



UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de fin de Grado en Ingeniería en Tecnologías de la Información

Sistema de soporte a la decisión para la optimización logística en almacenes intermedios

Daniel Efrén Díaz Ureta

Dirigido por: Alfonso Urquía Moraleda

Codirigido por: Carla Martín Villalba

Curso 2025/2026, convocatoria septiembre



Sistema de soporte a la decisión para la optimización logística en almacenes intermedios

Proyecto de fin de Grado en Ingeniería en Tecnologías
de la Información
de modalidad específica

Realizado por: Daniel Efrén Díaz Ureta

Dirigido por: Alfonso Urquía Moraleda

Codirigido por: Carla Martín Villalba

Fecha de lectura y defensa: septiembre de 2026

Agradecimientos

*Quisiera dar las gracias a mis padres y a Aroa, por su paciencia y apoyo todos estos años,
haciendo el camino más llevadero.*

*A mi familia y amigos, que entendiendo que podíamos vernos menos han estado siempre
presentes. En especial a Roberto por animarme a retomar los estudios y acompañarme en este
viaje.*

*A todos aquellos compañeros de trabajo que han servido de motivación para estudiar y han
aportado su granito de arena. En particular a Víctor por su mentoría a lo largo de los años y
todo lo que me ha enseñado.*

Resumen

En muchos procesos productivos es necesario contar con almacenes que desacoplen el flujo de salida de las instalaciones productoras del flujo de entrada de las instalaciones consumidoras, permitiendo su funcionamiento de forma independiente. En estos casos, es responsabilidad de los almacenes intermedios proporcionar un almacenamiento temporal que garantice la trazabilidad del material y a su vez cumplir con estrictos requisitos de tiempo de servicio. Para ello, resulta necesario optimizar las diferentes operaciones que intervienen en la gestión del inventario y en el movimiento de los materiales.

Ante esta necesidad, se desarrolla GenStock, una aplicación de gestión de almacenes en el contexto siderúrgico donde el material almacenado serán desbastes. La solución se compone de varias aplicaciones web, incluyendo un cliente web desarrollado con React y TypeScript y un servidor en C# sobre .NET, cuya lógica se organiza siguiendo los principios de arquitectura limpia y diseño orientado al dominio. Sobre esta arquitectura, la aplicación integra sistemas de soporte a la decisión encargados de generar propuestas de ubicación y los movimientos de grúa necesarios para ejecutarlas. Para ello, se implementan múltiples sistemas adaptados a operaciones propias del almacén, como la descarga de transportes y la carga de programas, junto con un sistema extensible de reglas para definir los criterios operativos empleados en la clasificación del inventario.

La propuesta de ubicación se basa en un algoritmo de búsqueda informada que utiliza un heurístico construido mediante una relajación del problema y una secuencia de extracción teórica. Debido a la complejidad combinatoria de la búsqueda, la descarga de transportes se descompone en subproblemas que permiten reducir la exploración sin afectar a la calidad de los resultados. La solución diferencia entre la selección de las ubicaciones y la optimización de la secuencia de movimientos, agrupando movimientos en ganchadas para reducir la distancia total recorrida. Finalmente, se han realizado pruebas funcionales sobre diferentes configuraciones y niveles de ocupación del almacén, verificando las propuestas de ubicación y los movimientos generados. Los resultados han permitido demostrar la validez de las soluciones propuestas y la utilidad del heurístico para reducir el espacio de búsqueda.

Abstract

In many production processes, it is necessary to have warehouses that decouple the flow of output from production facilities from the flow of input to consuming facilities, allowing them to operate independently. In these cases, intermediate warehouses are responsible for providing temporary storage that guarantees material traceability while also meeting strict service time requirements. Therefore, it is necessary to optimize the different operations involved in inventory management and material movement.

In response to this need, GenStock is developed as a warehouse management application in the steel industry, where slabs are the stored material. The solution is composed of several web applications, including a web client developed with React and TypeScript and a C# server running on .NET, whose logic is organized according to the principles of Clean Architecture and Domain-Driven Design. On top of this architecture, the application integrates decision-support systems responsible for generating storage location proposals and the crane movements required to execute them. To this end, multiple systems adapted to warehouse-specific operations are implemented, such as transport unloading and program loading, together with an extensible rule system for defining the operational criteria used to classify the inventory.

The storage location proposal is based on an informed search algorithm that uses a heuristic constructed through a relaxation of the problem and a theoretical extraction sequence. Due to the combinatorial complexity of the search, transport unloading is decomposed into subproblems that reduce the search space without affecting the quality of the results. The solution differentiates between the selection of storage locations and the optimization of the movement sequence, grouping movements into crane lifts to reduce the total distance travelled. Finally, functional tests have been carried out using different warehouse configurations and occupancy levels, verifying the generated storage location proposals and crane movements. The results have demonstrated the validity of the proposed solutions and the usefulness of the heuristic in reducing the search space.

Palabras clave

Gestión de almacenes, parque de desbastes, sistemas de soporte a la decisión, propuesta de ubicación, algoritmo A*, optimización de movimientos.

Keywords

Warehouse management, slab yard, decision support systems, storage location assignment, A* algorithm, movement optimization.

Índice

Índice de tablas	XV
Índice de figuras	XXII
Índice de fragmentos de código	XXIII
1. Introducción, objetivos y estructura	1
1.1. Introducción	1
1.2. Objetivos	2
1.3. Estructura del documento	3
2. Contexto	5
2.1. Introducción	5
2.2. Sistemas de gestión de almacenes	5
2.3. El almacén de desbastes	8
2.4. Funcionamiento	9
2.4.1. Desbastes	9
2.4.2. Elementos de almacenamiento	10
2.4.3. Transportes	12
2.4.4. Grúa	14
2.4.5. Programas	15
2.5. Problemática específica	17
2.5.1. Variabilidad de los materiales	17
2.5.2. Gestión de la entrada / salida	18
2.5.3. Planes de movimientos	21
2.6. Tecnologías empleadas	23
2.7. Conclusiones	24

3. Diseño	25
3.1. Introducción	25
3.2. Arquitectura software	25
3.2.1. Arquitectura del servidor	27
3.2.2. Cliente	29
3.2.3. Aspire	30
3.2.4. Keycloak	33
3.2.5. SignalR	35
3.3. Gestión del inventario	36
3.3.1. Fichero de configuración del plano del parque	37
3.3.2. Modelo del inventario	41
3.4. Modelo de movimientos	42
3.5. Modelo de programas	44
3.6. Modelo del plan de movimientos	45
3.7. Modelo relacional	47
3.8. Simulación de gruistas	51
3.9. Simulación de programación	52
3.9.1. Creación de materiales	52
3.9.2. Programas	54
3.9.3. Retenciones	56
3.10. Especificaciones de la aplicación	56
3.10.1. Especificaciones funcionales	57
3.10.2. Especificaciones de los servicios de simulación	59
3.11. Conclusiones	59
4. Diseño propuesta de ubicación	61
4.1. Introducción	61

4.2. Algoritmo A*	61
4.3. Expansión de un nodo	63
4.4. Heurístico	65
4.5. Tipos de reglas	65
4.6. Reglas de validación	66
4.6.1. Máximo elementos	66
4.6.2. Seguridad de pila	66
4.6.3. Movimientos permitidos	67
4.7. Reglas globales	67
4.7.1. Pilas vacías	67
4.7.2. Distancia	69
4.7.3. Tamaño de ganchada	70
4.8. Reglas locales	71
4.8.1. Programas	71
4.8.2. Cubrir programa	77
4.8.3. Destino	78
4.8.4. Tipo	80
4.8.5. Familia	82
4.8.6. Anchos	85
4.9. Configuración de reglas	87
4.10. Conclusiones	88
5. Implementación	89
5.1. Introducción	89
5.2. Sincronización inventario y configuración	89
5.2.1. Idea inicial	89
5.2.2. Problemas encontrados	90

5.2.3. Solución	90
5.3. Implementación del estado de la propuesta de ubicación	90
5.3.1. Idea inicial	91
5.3.2. Problemas encontrados	91
5.3.3. Solución	91
5.4. Heurístico propuesta ubicación	93
5.4.1. Idea inicial	93
5.4.2. Problemas encontrados	93
5.4.3. Solución	94
5.5. Diagnóstico propuesta ubicación	101
5.5.1. Idea inicial	101
5.5.2. Problemas encontrados	101
5.5.3. Solución	101
5.6. Optimización de la distancia a recorrer	108
5.6.1. Idea inicial	108
5.6.2. Problemas encontrados	108
5.6.3. Solución	110
5.7. Descarga de trenes	112
5.7.1. Idea inicial	112
5.7.2. Problemas encontrados	113
5.7.3. Solución	113
5.8. Carga de programas	117
5.8.1. Idea inicial	117
5.8.2. Problemas encontrados	119
5.8.3. Solución	120
5.9. Flujo completo de la propuesta de ubicación	125

5.10. Conclusiones	127
6. Pruebas de aplicación	129
6.1. Introducción	129
6.2. Problemática	129
6.3. Pruebas automatizadas	131
6.4. Pruebas funcionales de la aplicación	135
6.4.1. Estadísticas	135
6.4.2. Transportes	136
6.4.3. Inventario	140
6.4.4. Planes de movimiento	143
6.4.5. Programas	151
6.4.6. Mapa	151
6.4.7. Movimientos	154
6.4.8. Gestión de destinos	156
6.4.9. Gestión de usuarios	161
6.4.10. Gestión de grúas	165
6.5. Pruebas de los servicios de simulación	167
6.5.1. Simulación de grúas	167
6.5.2. Generación de materiales	170
6.5.3. Generación de programas	170
6.5.4. Gestión de retenciones	170
6.6. Conclusiones	171
7. Verificación del soporte a la decisión	172
7.1. Introducción	172
7.2. Configuración de reglas	173

7.3. Ejemplo liberar pila de un desbaste	176
7.4. Ejemplo un desbaste y varias alternativas	181
7.5. Ejemplo varios desbastes	185
7.5.1. Estado S0	190
7.5.2. Estado S1	191
7.5.3. Estado S2	193
7.5.4. Estado S3	195
7.5.5. Estado S4	197
7.5.6. Estado S5	199
7.6. Ejemplo descarga de tren	201
7.6.1. Estado 1	204
7.6.2. Estado 2	204
7.6.3. Estado 3	205
7.6.4. Estado 4	206
7.6.5. Estado 5	207
7.6.6. Estado 6	208
7.7. Ejemplo carga de programa	209
7.8. Conclusiones	215
8. Conclusiones y trabajos futuros	216
8.1. Introducción	216
8.2. Conclusiones	216
8.3. Trabajos futuros	221
A. Manual de usuario	225
A.1. Visión general de la aplicación	225
A.2. Menú usuario	227

A.3. Transportes	228
A.4. Inventario	230
A.5. Plan de movimientos	232
A.5.1. Descarga de transporte	234
A.5.2. Liberar pilas	236
A.5.3. Cargar programa	237
A.5.4. Reubicar	238
A.6. Programas	238
A.7. Mapa	239
A.8. Administración	242
A.8.1. Usuarios	243
A.8.2. Destinos	243
A.8.3. Grúas	243
A.8.4. Asignación de materiales	244
B. Guía de instalación del entorno	245
B.1. Requisitos previos	245
B.2. Puesta en marcha	245

Índice de tablas

2.1. Número de formas de extracción según tamaño pila.	20
3.1. Ejemplo partes movimiento.	44
3.2. Probabilidades de producción de cada tipo de material.	53
3.3. Probabilidades de producción de cada familia de material.	53
3.4. Especificaciones funcionales de la aplicación (1 de 2).	57
3.5. Especificaciones funcionales de la aplicación (2 de 2).	58
3.6. Especificaciones de los servicios de simulación.	59
4.1. Penalización según tamaño ganchada.	70
4.2. Penalizaciones debidas a mezcla de programa.	73
4.3. Comparativa de penalizaciones de anchos.	85
4.4. Penalización según la diferencia (estrecho sobre ancho).	86
5.1. Recuentos de filas en tablas de diagnóstico.	105
5.2. Movimientos propuestos sin optimización de distancias.	109
5.3. Movimientos propuestos con optimización de distancias.	111
5.4. Datos máximo rendimiento propuesta ubicación en 3 min.	113
5.5. Algoritmo generación de subproblemas - Iteración 1.	115
5.6. Algoritmo generación de subproblemas - Iteración 2.	116
5.7. Algoritmo generación de subproblemas - Iteración 3.	116
5.8. Algoritmo generación de subproblemas - Iteración 4.	117
5.9. Ejemplo de composición de grupos en programa.	119
6.1. Distribución de las pruebas automatizadas.	132

7.1. Configuración utilizada para la regla «Pilas vacías».	173
7.2. Configuración utilizada para la regla «Distancia».	174
7.3. Configuración utilizada para la regla «Programas».	174
7.4. Configuración utilizada para la regla «Destino».	175
7.5. Configuración utilizada para la regla «Tipo».	175
7.6. Configuración utilizada para la regla «Familia».	175
7.7. Descripción del desbaste usado en el Ejemplo 7.3.	176
7.8. Datos de diagnóstico propuesta ubicación del Ejemplo 7.3.	178
7.9. Pilas permitidas del Ejemplo 7.3.	179
7.10. Penalización pilas más prometedoras del Ejemplo 7.3.	180
7.11. Descripción del desbaste usado en el Ejemplo 7.4.	182
7.12. Tabla de movimientos propuestos del Ejemplo 7.4.	182
7.13. Datos de diagnóstico propuesta ubicación del Ejemplo 7.4.	183
7.14. Penalización pilas más prometedoras del Ejemplo 7.4.	184
7.15. Resultados de evaluación pilas más prometedoras del Ejemplo 7.4.	184
7.16. Descripción del desbaste usado en el Ejemplo 7.5.	185
7.17. Tabla de movimientos propuestos del Ejemplo 7.5.	186
7.18. Datos de diagnóstico propuesta ubicación del Ejemplo 7.5.	187
7.19. Resumen de la exploración del árbol de búsqueda del Ejemplo 7.5.	189
7.20. Evaluación del estado S0 en el Ejemplo 7.5.	190
7.21. Expansión del estado S0 en el Ejemplo 7.5.	191
7.22. Evaluación del estado S1 en el Ejemplo 7.5.	191
7.23. Evaluación del movimiento realizado para alcanzar S1 en el Ejemplo 7.5.	192
7.24. Expansión del estado S1 en el Ejemplo 7.5.	193
7.25. Evaluación del estado S2 en el Ejemplo 7.5.	194
7.26. Evaluación del movimiento realizado para alcanzar S2 en el Ejemplo 7.5.	194

7.27. Expansión del estado S2 en el Ejemplo 7.5.	195
7.28. Evaluación del estado S3 en el Ejemplo 7.5.	195
7.29. Evaluación del movimiento realizado para alcanzar S3 en el Ejemplo 7.5.	196
7.30. Expansión del estado S3 en el Ejemplo 7.5.	196
7.31. Evaluación del estado S4 en el Ejemplo 7.5.	197
7.32. Evaluación del movimiento realizado para alcanzar S4 en el Ejemplo 7.5.	198
7.33. Expansión del estado S4 en el Ejemplo 7.5.	198
7.34. Evaluación del movimiento realizado para alcanzar S5 en el Ejemplo 7.5.	200
7.35. Descripción desbastes vagón 1 en el Ejemplo 7.6.	201
7.36. Descripción desbastes vagón 2 en el Ejemplo 7.6.	201
7.37. Descripción desbastes vagón 3 en el Ejemplo 7.6.	202
7.38. Grupos de desbastes para generación de subproblemas del Ejemplo 7.6.	203
7.39. Datos de diagnóstico propuesta ubicación del Ejemplo 7.6.	204
7.40. Movimientos calculados para el estado 1 del Ejemplo 7.6.	204
7.41. Movimientos calculados para el estado 2 del Ejemplo 7.6.	205
7.42. Movimientos calculados para el estado 3 del Ejemplo 7.6.	206
7.43. Movimientos calculados para el estado 4 del Ejemplo 7.6.	207
7.44. Movimientos calculados para el estado 5 del Ejemplo 7.6.	207
7.45. Movimientos calculados para el estado 6 del Ejemplo 7.6.	208
7.46. Tabla de movimientos propuestos del Ejemplo 7.7.	210

Índice de figuras

2.1. Grúa transportando desbastes. Autor: Borvan53, vía Wikimedia Commons, bajo licencia CC BY-SA 4.0 (link).	8
2.2. Ejemplo de pila de desbastes. Autor: Borvan53, vía Wikimedia Commons, bajo licencia CC BY-SA 3.0 (link).	11
2.3. Pilas en parque de desbastes. Autor: indeedou, Attribution vía Wikimedia Commons(link).	11
2.4. Pila con desbastes con ancho piramidal.	12
2.5. Zona de transportes representada en la aplicación.	13
2.6. Esquema pinza de desbastes con 4 materiales.	14
2.7. Representación de un programa de desbastes.	16
2.8. Vista lateral desbaste ancho recibido tarde.	18
2.9. Formas de extraer de una pila de tres desbastes.	20
3.1. Diagrama de arquitectura de la aplicación.	26
3.2. Dependencias entre capas en arquitectura limpia.	29
3.3. Página principal Aspire.	32
3.4. Ejemplo de traza en Aspire.	33
3.5. Clientes definidos en Keycloak.	34
3.6. Flujo simplificado de autenticación OpenID Connect.	35
3.7. Comunicación SignalR entre API y clientes.	36
3.8. Diagrama UML del modelo de inventario.	42
3.9. Modelo partes de movimiento.	43
3.10. Diagrama UML del modelo de movimientos.	44
3.11. Diagrama UML del modelo de programas.	45
3.12. Diagrama UML del modelo de planes de movimiento.	47

3.13. Diagrama ER materiales y programas.	48
3.14. Diagrama ER modelo de inventario.	49
3.15. Diagrama ER movimientos y programas de movimiento.	50
3.16. Composición y uso de tipos en un programa.	55
4.1. Terminología propuesta ubicación.	63
4.2. Diagrama expansión estados A^*	64
5.1. Uso de listas enlazadas para representar pilas de desbastes.	92
5.2. Situación de origen para cálculo secuencia teórica.	96
5.3. Ejemplo orden ascendente vs orden teórico.	96
5.4. Diagrama ER diagnóstico propuesta ubicación.	103
5.5. Resultado consulta estadísticas reglas propuesta.	106
5.6. Resultado consulta diagnóstico de propuestas de ubicación.	108
5.7. Caso ejemplo necesidad de optimización de distancias.	109
5.8. Realización del movimiento sin optimización de distancias.	110
5.9. Realización del movimiento con optimización de distancias.	111
5.10. Ejemplo de cabeceras que deberían descargarse juntas.	112
5.11. Ejemplo de grupos de descarga.	114
5.12. Ejemplo de grupos de descarga - Estado tras iteración 1.	115
5.13. Ejemplo de grupos de descarga - Estado tras iteración 2.	116
5.14. Ejemplo de grupos de descarga - Estado tras iteración 3.	116
5.15. Ejemplo de composición de grupos en carga de vagón.	118
5.16. Ejemplo objetivo de grupos en carga de programas en un vagón.	120
5.17. Ejemplo evolución de una carga de un vagón.	121
5.18. Ejemplo poda de composición incorrecta.	124
5.19. Ejemplo movimiento redundante 1.	124

5.20. Ejemplo movimiento redundante 2.	125
5.21. Diagrama de flujo ejecución propuesta de ubicación.	127
6.1. Resultados ejecución pruebas automáticas.	131
6.2. Verificación de estadísticas del almacén.	136
6.3. Estado anterior a la entrada del transporte.	137
6.4. Elección de materiales en entrada del transporte.	137
6.5. Estado posterior a la entrada del transporte.	138
6.6. Estado mapa posterior a la entrada del transporte.	138
6.7. Estado anterior a la salida del transporte.	139
6.8. Estado posterior a la salida del transporte.	139
6.9. Estado mapa posterior a la salida del transporte.	140
6.10. Inventario con material seleccionado.	141
6.11. Inventario con filtro de material.	142
6.12. Inventario con material seleccionado y búsqueda.	143
6.13. Alta de un plan de movimiento para descarga de transporte.	144
6.14. Listado tras el alta del plan de descarga de transporte.	144
6.15. Alta de un plan de movimiento para liberación de pila.	145
6.16. Listado tras el alta del plan de liberación de pila.	145
6.17. Alta de un plan de movimiento para carga de programa.	146
6.18. Listado tras el alta del plan de carga de programa.	146
6.19. Alta de un plan de movimiento para reubicación.	147
6.20. Listado tras el alta del plan de reubicación.	147
6.21. Confirmación de la cancelación de un plan de movimiento.	148
6.22. Listado de planes tras la cancelación.	148
6.23. Confirmación del borrado de un plan de movimiento.	149
6.24. Listado de planes tras el borrado.	149

6.25. Cambio de orden de un plan de movimiento.	150
6.26. Listado de planes tras el cambio de orden.	150
6.27. Listado con programa seleccionado.	151
6.28. Mapa del estado actual del almacén.	152
6.29. Mapa de estado actual del almacén con filtro de tipo.	153
6.30. Mapa del almacén con la visualización de movimientos activada.	154
6.31. Estado de la pila «B14» antes de realizar el movimiento.	155
6.32. Diálogo para realizar el movimiento de material.	155
6.33. Estado de las pila «A14» después de realizar el movimiento.	156
6.34. Formulario de alta de un destino.	157
6.35. Listado destinos con nuevo elemento.	157
6.36. Formulario de modificación de un destino.	158
6.37. Listado de destinos con el elemento modificado.	159
6.38. Confirmación del borrado de un destino.	160
6.39. Listado de destinos tras el borrado.	160
6.40. Formulario de alta de un usuario.	161
6.41. Listado de usuarios con nuevo elemento.	162
6.42. Formulario de modificación de un usuario.	163
6.43. Listado de usuarios con el elemento modificado.	163
6.44. Confirmación del borrado de un usuario.	164
6.45. Listado de usuarios tras el borrado.	165
6.46. Listado de grúas.	166
6.47. Formulario de modificación de una grúa.	166
6.48. Listado de grúas con el elemento modificado.	167
6.49. Grúa realizando depósito en la bahía camiones 3.	168
6.50. Grúa realizando la ganchada sobre la pila «C13».	168

6.51. Progreso de la operación sobre la pantalla de planes de movimiento.	169
6.52. Grúa tras la realización de los movimientos.	169
6.53. Menú de programas mostrando aquellos que están creados.	170
6.54. Registro de material con retención.	171
7.1. Ubicación inicial del desbaste en el Ejemplo 7.3.	176
7.2. Creación del plan de movimientos en el Ejemplo 7.3.	177
7.3. Plan de movimiento con propuesta para el Ejemplo 7.3.	177
7.4. Movimiento en mapa del Ejemplo 7.3.	178
7.5. Pilas descartadas del Ejemplo 7.3.	179
7.6. Pilas evaluadas del Ejemplo 7.3.	180
7.7. Estado final pila tras movimiento del Ejemplo 7.3.	181
7.8. Ubicación inicial del desbaste en el Ejemplo 7.4.	182
7.9. Movimiento en mapa del Ejemplo 7.4.	183
7.10. Ubicación inicial de desbastes en el Ejemplo 7.5.	186
7.11. Movimientos en mapa del Ejemplo 7.5.	187
7.12. Flujo ejecución A* del Ejemplo 7.5.	188
7.13. Configuración pila «C04» estado S1 en el Ejemplo 7.5.	192
7.14. Configuración pila «B10» estado S4 en el Ejemplo 7.5.	197
7.15. Configuración pila «F11» estado S5 en el Ejemplo 7.5.	199
7.16. Movimientos en mapa del Ejemplo 7.6.	202
7.17. Representación de criterios en estado inicial del Ejemplo 7.6.	203
7.18. Representación de criterios en estado 2 del Ejemplo 7.6.	205
7.19. Representación de criterios en estado 3 del Ejemplo 7.6.	205
7.20. Representación de criterios en estado 4 del Ejemplo 7.6.	206
7.21. Representación de criterios en estado 5 del Ejemplo 7.6.	207
7.22. Representación de criterios en estado 6 del Ejemplo 7.6.	208

7.23. Secuencia de movimientos propuesta en el Ejemplo 7.7.	209
7.24. Resultado de la carga del programa en el Ejemplo 7.7.	212
7.25. Resultado de la carga del vagón 1 en el Ejemplo 7.7.	213
7.26. Resultado de la carga del vagón 2 en el Ejemplo 7.7.	213
7.27. Resultado de la carga del vagón 3 en el Ejemplo 7.7.	214
A.1. Pantalla de acceso.	225
A.2. Pantalla de registro.	226
A.3. Pantalla gráficos.	226
A.4. Pantalla de ajustes.	227
A.5. Pantalla transportes sin selección.	228
A.6. Pantalla transportes con selección.	229
A.7. Diálogo incluir materiales.	229
A.8. Pantalla transportes indicando entrada de transporte.	230
A.9. Pantalla transportes indicando salida de transporte.	230
A.10. Pantalla inventario sin material seleccionado.	231
A.11. Pantalla inventario con material seleccionado.	232
A.12. Pantalla plan de movimientos vacía.	233
A.13. Pantalla para crear distintos tipos de planes de movimientos.	233
A.14. Diálogo crear plan de movimientos de tipo descarga de transporte.	234
A.15. Pantalla con los movimientos para la descarga de un camión.	234
A.16. Pantalla con varios elementos en ciertos movimientos.	235
A.17. Pantalla con movimientos en distintos estados.	236
A.18. Diálogo crear plan de movimientos de tipo liberar pila.	237
A.19. Diálogo crear plan de movimientos de tipo cargar programa.	237
A.20. Diálogo crear plan de movimientos de tipo reubicar.	238
A.21. Pantalla programas.	239

A.22.Visión general del mapa del parque.	239
A.23.Sección del mapa del parque.	240
A.24.Pantalla mapa con tipo seleccionado.	240
A.25.Menú botón derecho sobre una pila.	241
A.26.Pantalla detalle de una pila.	241
A.27.Pantalla para mover materiales.	242
A.28.Pantalla mapa mostrando los movimientos.	242
A.29.Pantalla para administración de usuarios.	243
A.30.Pantalla para administración de destinos.	243
A.31.Pantalla para administración de grúas.	244
A.32.Pantalla de administración para asignación de materiales.	244
B.1. Página principal Aspire en entorno de desarrollo.	247

Lista de fragmentos de código

3.1. Configuración dependencias en Aspire.	31
3.2. Definición objeto Yard.	37
3.3. Definición objeto Crane.	38
3.4. Definición objeto PileRack.	39
3.5. Definición objeto Zone.	40
4.1. Fichero de configuración de reglas.	87
5.1. Pseudocódigo cálculo heurístico de propuesta de ubicación.	94
5.2. Consulta estadísticas reglas propuesta.	106
5.3. Consulta diagnóstico de propuestas de ubicación.	107
5.4. Pseudocódigo algoritmo generación de subproblemas.	114
5.5. Pseudocódigo algoritmo expansión nodos carga de programas.	123
6.1. Ejemplo prueba de caso.	132
6.2. Ejemplo prueba comparativa.	133

Capítulo 1

Introducción, objetivos y estructura

1.1. Introducción

En la mayoría de los procesos industriales es necesario almacenar materiales. Sin embargo, la función y las características del almacenamiento dependen en gran medida del producto y proceso productivo. Según el punto del proceso en el que se encuentren, sus responsabilidades, su complejidad y los problemas que han de resolver variarán enormemente.

En este contexto son los almacenes intermedios los que suelen presentar una mayor complejidad operativa ya que han de balancear las necesidades de una o varias instalaciones productoras y una o varias instalaciones consumidoras. Su responsabilidad incluye actuar como elemento regulador, que evite paradas y cuellos de botella permitiendo un aumento de la flexibilidad y mayor independencia entre productor y consumidor.

En el caso concreto de los almacenes intermedios siderúrgicos hay varias características que añaden una mayor complejidad. La manipulación de piezas de grandes tonelajes conlleva limitaciones asociadas a la forma de almacenar, incrementa el tiempo de transporte y requiere requisitos adicionales de seguridad. Otro problema viene derivado de la variabilidad de los materiales en tipos y dimensiones impidiendo en muchos casos intercambiar una pieza por otra y obligando a la realización de más movimientos o desplazamientos más largos.

Dentro de los distintos almacenes intermedios presentes en una planta siderúrgica, los parques de desbastes constituyen un caso de especial interés. En ellos se almacenan temporalmente los desbastes producidos en la acería, productos semielaborados que posteriormente serán consumidos por las instalaciones de laminación. Como elemento de unión entre ambos procesos, el parque debe absorber las diferencias entre los ritmos de producción y consumo, garantizando simultáneamente la disponibilidad del material requerido y el uso eficiente de los recursos de manipulación y transporte.

Por este motivo, el presente trabajo se centra en el estudio de los almacenes intermedios de desbastes, donde el diseño de un sistema de soporte a la decisión tiene un mayor impacto en la mejora del almacén.

1.2. Objetivos

Con el objetivo de facilitar la toma de decisiones operativas involucradas en la operación de un almacén intermedio se desarrollará un sistema de optimización de la gestión de almacenes de desbastes basado en una aplicación web que permita:

1. **Asistencia a la ubicación de desbastes.** El objetivo principal de la aplicación será asistir a los operadores del almacén en la elección de ubicaciones donde almacenar los desbastes. Esta decisión es la más crítica y compleja que deben tomar y en la que, para alcanzar los criterios operativos, deben considerarse multitud de criterios.
2. **Consultar el estado del inventario en tiempo real.** El sistema permitirá visualizar en tiempo real el estado del inventario del parque en forma de mapa en dos dimensiones, así como la localización de cualquier desbaste en el mismo. De esta forma se garantizará la completa visibilidad de las operaciones y el seguimiento de los trabajos en curso.
3. **Garantizar la trazabilidad del material.** En el contexto de un almacén es vital contar con toda la información relativa a la manipulación de un desbaste para poder investigar y entender potenciales problemas de calidad. Para ello se mantendrá información de todas las ubicaciones y movimientos de los materiales presentes en el almacén.
4. **Gestionar los movimientos de las grúas.** El sistema proporcionará opciones a los operadores para la gestión del trabajo que han de realizar las grúas. Estas opciones integrarán la asistencia a la ubicación como parte de la toma de decisiones en la planificación de secuencias de movimientos. De esta forma se proporciona asistencia para tareas como la carga o descarga de transportes o la reorganización de material.

Como parte del desarrollo se implementarán herramientas de simulación que permitan observar la operación del parque en un entorno realista y la verificación de los sistemas de soporte a la decisión. Estas simulaciones cubrirán aspectos como el movimiento de las grúas, la generación de desbastes, la composición de transportes de entrada y planificaciones de salida de material.

El objetivo principal del trabajo es desarrollar una solución capaz de determinar la ubicación óptima de los materiales en cualquier caso, pero particularmente en la descarga de transportes. Para conseguir este objetivo se plantea un enfoque basado en la descomposición del problema en subproblemas de menor complejidad que contribuyan a la construcción de la solución final. Sobre esta estructura, se aplicará conocimiento de dominio para el desarrollo de heurísticas que permitan reducir significativamente el espacio de búsqueda y guiar la exploración hacia opciones más prometedoras. Como resultado se esperan obtener soluciones óptimas o cercanas al óptimo en tiempos de cálculo reducidos compatibles con la operación del almacén en un entorno industrial real.

Para alcanzar los objetivos planteados será necesario realizar las siguientes tareas:

1. Desarrollar la gestión de inventario.

- Implementar la persistencia de las ubicaciones de los materiales.
- Desarrollar la representación gráfica del estado del inventario.
- Implementar un mecanismo de actualización de la información ante cambios en el estado del almacén.

2. Implementar la trazabilidad de los materiales.

- Implementar la persistencia de los movimientos realizados sobre los materiales.
- Desarrollar funcionalidades para consultar el historial y la ubicación actual de los materiales.

3. Desarrollar el sistema de soporte a la decisión de la ubicación de desbastes.

- Identificar las reglas y restricciones que intervienen en la ubicación de materiales.
- Diseñar e implementar el algoritmo de búsqueda para la generación de propuestas de ubicación.
- Desarrollar heurísticas basadas en el conocimiento del dominio para reducir el espacio de búsqueda y orientar la exploración.
- Integrar las propuestas de ubicación en la planificación de movimientos.

4. Desarrollar la gestión de los movimientos de las grúas.

- Desarrollar las funcionalidades necesarias para planificar y persistir los planes de movimientos.
- Integrar los sistemas de soporte a la decisión en la generación de planes de movimientos.

La realización de estas tareas permitirá desarrollar y posteriormente verificar las funcionalidades necesarias para alcanzar los objetivos planteados.

1.3. Estructura del documento

Este documento está organizado en los siguientes capítulos:

- **Capítulo 1 - Introducción, objetivos y estructura:** En este capítulo se introduce el ámbito del proyecto, sus objetivos y la motivación detrás del desarrollo de esta aplicación.
- **Capítulo 2 - Contexto:** Aquí se explora el marco teórico del problema, se compara con otros estudios y trabajos de áreas similares y se introducen las tecnologías utilizadas en

el proyecto. En este capítulo se explica en profundidad el funcionamiento y problemáticas de un almacén de desbastes. Se explorarán procesos del parque que lo hacen diferente a otros casos y los problemas operativos que esto genera.

- **Capítulo 3 - Diseño:** A lo largo de este capítulo se explora cómo se ha diseñado la aplicación realizada para cumplir con los objetivos marcados. Se revisan y justifican las tecnologías implicadas revisando los aspectos que han motivado su inclusión. Se explora la arquitectura elegida, modelo de datos y se documentan las especificaciones que ha de cumplir la aplicación.
- **Capítulo 4 - Diseño propuesta ubicación:** En este capítulo se detalla el diseño del sistema encargado de generar propuestas de ubicación para los materiales en el almacén. Se presenta el planteamiento del problema de búsqueda, el modelo de estados y las reglas utilizadas para evaluar las distintas alternativas.
- **Capítulo 5 - Implementación:** Aquí se repasan los problemas más relevantes que se han encontrado durante la implementación y las soluciones planteadas para su corrección. Se ofrecerán descripciones detalladas de distintos aspectos técnicos y del proceso seguido para su solución.
- **Capítulo 6 - Pruebas de aplicación:** A lo largo de este capítulo se presenta la verificación del funcionamiento de la aplicación mediante las pruebas realizadas sobre sus principales funcionalidades. Se incluyen pruebas unitarias y pruebas de integración, comprobando el comportamiento de los distintos componentes y el cumplimiento de los requisitos definidos.
- **Capítulo 7 - Verificación del sistema de soporte a la decisión:** En este capítulo se analiza el funcionamiento de los sistemas de soporte a la decisión implementados, evaluando los algoritmos desarrollados sobre distintos casos y utilizando las métricas obtenidas para justificar la corrección de los resultados.
- **Capítulo 8 - Conclusiones y trabajos futuros:** En este capítulo se presentan las conclusiones del proyecto, reflexionando sobre la herramienta desarrollada y los problemas solucionados. Se discuten posibles mejoras y extensiones futuras.
- **Anexo I - Manual de usuario:** En este capítulo se revisan las funcionalidades de la aplicación a través de su interfaz de usuario.
- **Anexo II - Guía de Instalación:** En este capítulo se describirán los pasos necesarios para crear y configurar un entorno de desarrollo del proyecto.

Capítulo 2

Contexto

2.1. Introducción

En este capítulo, se revisa el marco teórico relacionado con el objetivo principal del proyecto de desarrollar un sistema de soporte a la decisión para optimizar el almacenamiento y los movimientos en un parque de desbastes. Se establecerá el marco teórico necesario para el trabajo, se revisarán problemas similares, así como, trabajos centrados en almacenes de desbastes y finalmente se realizará un resumen de las tecnologías que se usarán en el proyecto.

2.2. Sistemas de gestión de almacenes

En el contexto de la industria la gestión de los almacenes es un tema de gran importancia debido a que su funcionamiento se encuentra íntimamente ligado a la productividad de las fábricas. Mantener un control eficaz, tanto de las materias primas como de los productos intermedios y terminados, permite a las compañías tomar decisiones para la gestión de la producción, optimizar los flujos de trabajo y satisfacer los pedidos. Por estos motivos, existen en el mercado numerosas soluciones genéricas conocidas como *Sistemas de Gestión de Almacenes (SGA) 2026* que responden a estas necesidades, proporcionando funcionalidades de gestión de inventario, recepción, empaquetado y distribución.

A pesar de lo anterior y por la evidente similitud, sobre todo en la gestión del inventario, cuando hablamos de almacenes intermedios siderúrgicos, aparecen factores diferenciales que hacen necesario el desarrollo de sistemas especializados en su problemática en lugar de aprovechar soluciones estándar. Por ejemplo:

- **Flujos de entrada/salida no previsibles.** Debidos a la dependencia de otras instalaciones productivas con diferentes objetivos.
- **Dificultades para la trazabilidad.** Al no ser productos terminados que pueden transformarse en varios artículos o formar parte de otros, la identificación se realiza de forma temporal, típicamente en forma de etiquetas o marcas.
- **Tamaño y forma.** Suele tratarse de productos de grandes tonelajes con necesidades

de almacenamiento y transporte especiales, que no encajan en los casos habituales de almacenamiento en palés y estanterías.

- **Optimización para producción en lugar de para entrega.** La perspectiva de un SGA típicamente considera la entrega a clientes finales de productos terminados, a la hora de optimizar un almacén. Sin embargo, en almacenes intermedios se optimiza para las necesidades de producción de la siguiente instalación de la cadena, lo que habitualmente conlleva relajar el orden de los pedidos.

Los problemas anteriores llevan al desarrollo de soluciones particulares para la gestión de las ubicaciones del material y la optimización de movimientos. En ambas áreas se requiere de toma de decisiones multicriterio en tiempos reducidos, por lo que es de vital importancia contar con sistemas de soporte a la decisión que asistan el trabajo de los operadores del almacén.

A continuación se realiza una revisión de la bibliografía más relevante para el almacenamiento de desbastes, así como otras investigaciones de las que se pueden extraer conocimientos aplicables al problema.

La revisión bibliográfica pone de manifiesto el reducido número de trabajos disponibles en comparación con casos similares, como el almacenamiento de contenedores de barco. Sin embargo, el escenario abordado en este trabajo presenta una complejidad adicional, debido a factores como la manipulación múltiple de los desbastes, que se almacenan formando pilas, el desconocimiento de los órdenes de entrada y salida de materiales y la ausencia de estandarización de sus dimensiones.

En L. Chen, Lin y Xu 2025 se revisa la optimización del almacenamiento de contenedores con el objetivo de minimizar el número de reubicaciones, comparando el uso de heurísticas y hiper-heurísticas para la obtención de soluciones. El problema abordado en este trabajo puede considerarse una generalización del problema de almacenamiento de contenedores. Si se consideran desbastes idénticos, una única grúa, un orden fijo de extracción y se minimizan únicamente las reubicaciones, se obtiene un caso equivalente al problema de almacenamiento de contenedores, conocido por ser NP-duro. Por tanto, el problema propuesto, al ser una generalización de un problema NP-duro, también es NP-duro. Otro estudio relevante en el contexto de los contenedores marítimos, es Zhu et al. 2012 que aplica el algoritmo IDA* al problema de la recuperación de contenedores en orden estricto, una restricción compartida con el problema abordado en este trabajo.

Ya en el ámbito de los parques de desbastes, Kofler et al. 2012 presenta de forma clara el problema y propone un modelo de base para la asignación de ubicaciones. Sin embargo, presenta limitaciones al no contemplar la recepción dinámica de material, restricciones dimensionales y cambios en la planificación.

En cambio, Mizuyama [2020](#) considera el caso en el que la llegada de los materiales no es conocida de antemano y propone una solución mediante algoritmos de aprendizaje por refuerzo. Sin embargo, permite la salida de los materiales para su procesamiento sin un orden estricto, lo cual no es válido en nuestro caso. De la misma forma, Fechter et al. [2015](#) se centra en minimizar el número de movimientos de reordenación pero permite flexibilidad en el orden de entrega.

En Song et al. [2023](#) proponen agrupar los materiales según la lógica de laminación para aumentar la probabilidad de que los materiales almacenados juntos sean extraídos de forma conjunta. Esta idea es coherente con lo expuesto en Boge y Knust [2023](#), donde se estudia una variante del problema de reubicación en la que los materiales se agrupan en familias para reducir el número de reordenaciones. Sin embargo, la propuesta de Song et al. [2023](#) utiliza los pedidos como forma de agrupación, mientras que en nuestro escenario resulta más relevante utilizar las características de laminación de cada desbaste. Esta decisión se justifica sobre la base de que la generación de secuencias de laminación constituye un problema complejo que puede abordarse mediante algoritmos genéticos multiobjetivo, como muestran Fardad, Safi-Esfahani y Barekattain [2024](#). De este modo, la clasificación de los desbastes no solo afecta al almacenamiento, sino también a la eficiencia del proceso de laminación.

En Roljic, Leitner y Doerner [2021](#) se cuantifica la distribución del tiempo empleado por los vehículos de transporte y apilamiento en un parque de desbastes. Los autores muestran que, en un almacén bien organizado, las operaciones de apilamiento representan aproximadamente entre un 40 % y un 50 % del tiempo total de operación, mientras que el resto corresponde al desplazamiento de los vehículos. Este resultado pone de manifiesto la importancia de mantener una disposición del material en el parque que limite las reordenaciones, ya que incrementan el tiempo dedicado a las operaciones de manipulación y, por tanto, reducen la productividad.

Otras investigaciones como Zhao et al. [2019](#) y Wang et al. [2025](#) incluyen el problema de la gestión del trabajo de múltiples grúas, lo que queda fuera de ámbito del presente trabajo. No obstante, ambos consideran grúas capaces de transportar hasta dos materiales simultáneamente mientras que, en nuestro caso, se trabajará con una capacidad de cuatro.

En conjunto, la literatura revisada trata aspectos como la asignación de ubicaciones, la reducción de reubicaciones, la agrupación de materiales o la coordinación de múltiples grúas. Sin embargo, no se ha encontrado ningún trabajo que combine simultáneamente la recepción dinámica de desbastes, las restricciones dimensionales, el mantenimiento del orden estricto de extracción, la clasificación basada en criterios de laminación y las características operativas consideradas en este trabajo. Este vacío motiva el desarrollo de la propuesta presentada en los capítulos siguientes.

2.3. El almacén de desbastes

En esta sección se introduce el almacén intermedio de desbastes y sus funciones, que se desarrollan en mayor detalle a lo largo de las próximas secciones.

Dentro del proceso productivo siderúrgico se encuentran entre las acerías, donde se producen los desbastes como resultado de las coladas, y las instalaciones consumidoras, que realizan la laminación en caliente para su transformación en bobinas. Como se describe en Zhao et al. 2019 su objetivo principal es proporcionar almacenamiento temporal y actuar como búfers de la producción.

El funcionamiento del parque está determinado por las necesidades de almacenamiento de material saliente de la acería y el material programado para ser laminado, y a su vez, limitado por las capacidades de las grúas y transportes para realizar los movimientos necesarios. En la Figura 2.1 observamos una grúa realizando un movimiento. Nótese que estas operaciones requieren de maquinaria de gran tonelaje.



Figura 2.1: Grúa transportando desbastes. Autor: Borvan53, vía Wikimedia Commons, bajo licencia CC BY-SA 4.0 ([link](#)).

Cumplir los objetivos de funcionamiento anteriores implica manipular y almacenar material de forma óptima, así como atender la entrada/salida manteniendo en todo momento la trazabilidad. Para conseguir estos objetivos los operarios deben balancear multitud de criterios como:

- Seguridad en el almacenamiento del material.
- Agrupación del material en función de sus características.
- Minimizar los movimientos de los elementos de transporte.
- Tiempos de servicio en entrada / salida de material.

Habiendo descrito los objetivos principales del almacén, a continuación, se procederá a describir detalladamente el funcionamiento de un parque de desbastes.

2.4. Funcionamiento

Para comprender el funcionamiento del parque comenzaremos revisando los elementos fundamentales de los que se compone el almacén. Comenzaremos nuestra revisión analizando las principales entidades del parque y profundizando en el detalle de las mismas y su encaje en el funcionamiento global.

2.4.1. Desbastes

Constituyen el elemento principal de almacenamiento y cuentan con las siguientes propiedades.

- **Identificador:** Se trata de un número de 9 dígitos que identifica completamente un material. Los dos primeros dígitos indican el año de producción y los siguientes se asignan de forma incremental según su fabricación. Por ejemplo, el identificador "250000839" nos indica que es el desbaste 839 del año 2025.
- **Propiedades dimensionales:** Se utilizan principalmente para determinar la seguridad del almacenamiento, a excepción del ancho que también es fundamental en los movimientos y en la agrupación del material. Son las siguientes:
 - Ancho: En el rango [700-2500] milímetros.
 - Largo: En el rango [5-12] metros.
 - Espesor: En el rango [150-300] milímetros.
 - Peso: En el rango [5-35] toneladas.
- **Programa:** Hace referencia a una planificación de fabricación asociada al material que establece el orden en el que debe entregarse. Esta propiedad se analiza en mayor detalle en la Subsección 2.4.5.
- **Tipo:** Clase de material que determina los programas en los que puede utilizarse.
- **Familia:** Es una agrupación de varios tipos. Un material de una familia puede asignarse a

uno de varios tipos posibles según su composición. Es relevante cuando el tipo de material es «Resto». La familia es el criterio empleado cuando un desbaste no tiene pedido asignado, la asignación implica la pertenencia a un tipo.

- **Arranques / Cabeceras:** Materiales cuyo tipo de material es respectivamente «Arranque» o «Cabecera». Se diferencian conceptualmente, ya que se utilizan para la composición de programas de otros tipos. Esta propiedad se analiza con mayor detalle en la Subsección 2.4.5.
- **Destino:** Material que abandonará el almacén sin programa. Pueden ser instalaciones auxiliares u otros parques.
- **Retenido:** Material que potencialmente tiene algún problema de calidad y que no puede pertenecer a ningún programa hasta que se resuelva.

Las propiedades relacionadas con los atributos físicos del material no se pueden modificar, sin embargo, el resto pueden recibir cambios:

- **Programa:** La asignación de materiales a un programa se realiza típicamente cuando el material ya se encuentra en el parque. También es posible que se reciba material de programas no prioritarios o que no se han comenzado a expedir.
- **Tipo:** Una vez asignado el tipo no se puede cambiar, el cambio que puede producirse es la asignación de tipo a material con criterio «Familia». En esos casos la familia del material determinará los posibles tipos que pueden ser asignados.
- **Destino:** Se trata de materiales que se envían a otros parques para su almacenamiento o procesamiento. No se dará el caso de que el destino sea nuestro parque. Dado ese caso, lo veremos sin destino para su agrupamiento por el criterio correspondiente. La asignación de destinos se realiza de forma externa a la operativa del parque por lo que pueden aparecer para cualquier desbaste que no tenga programa o esté retenido.
- **Retenido:** Puede retenerse cualquier material de cualquier criterio en cualquier momento, típicamente esto ocurre por la detección de problemas de calidad, daños o problemas de identificación. La información de retención incluye una categoría y un texto libre opcional.

Habiendo descrito las características de los elementos que son almacenados, continuaremos nuestra exposición con los detalles que afectan a su almacenamiento.

2.4.2. Elementos de almacenamiento

El almacenamiento en el parque se realiza usando pilas como la que podemos observar en la Figura 2.2. Su principal característica es que el almacenamiento o la extracción solo pueden realizarse a través de la parte superior.



Figura 2.2: Ejemplo de pila de desbastes. Autor: Borvan53, vía Wikimedia Commons, bajo licencia CC BY-SA 3.0 ([link](#)).

En la Figura 2.3 podemos observar una serie de pilas en un almacén interior.



Figura 2.3: Pilas en parque de desbastes. Autor: indeedou, Attribution vía Wikimedia Commons([link](#)).

Es habitual referirse a la parte de material accesible como material en el «tope» y al resto como material «enterrado».

Para garantizar que no se produce inestabilidad física de los materiales, sobre todo ante golpes con los elementos manipuladores, se establecen las siguientes normas de seguridad para el material en pilas:

- La diferencia de ancho entre cualquier desbaste de la pila no debe exceder 320 mm.
- La diferencia de ancho entre dos desbastes cualesquiera no debe exceder 250 mm.
- La diferencia de largo entre cualquier desbaste de la pila no debe exceder 4 m.
- La diferencia de largo entre dos desbastes cualesquiera no debe exceder 2 m.
- No se pueden colocar más de 12 desbastes en la misma pila, excepto cuando sea un apoyo temporal en el que se dejan elementos y se extraen en el siguiente movimiento.

Las reglas dimensionales son solo de aplicación cuando la pila es no piramidal, es decir, cuando se colocan elementos más grandes encima de más pequeños. En la Figura 2.4 observamos que el material es completamente estable cuando es piramidal y por tanto dicha opción es siempre segura.

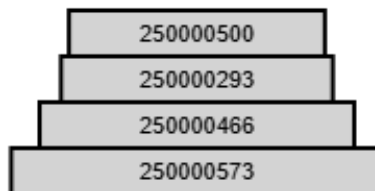


Figura 2.4: Pila con desbastes con ancho piramidal.

Las pilas se nombran usando letras para identificar la fila y números para las columnas, de forma que la pila «C14» corresponde a la tercera fila y a la columna catorce.

Habiendo descrito el almacenamiento permanente continuaremos nuestra exposición analizando el funcionamiento del almacenamiento temporal en transportes.

2.4.3. Transportes

Los transportes son los elementos que se utilizan para la entrada / salida de material en el parque y pueden ser de dos tipos: vagones o camiones. En el parque existen zonas diseñadas para los mismos y son los únicos puntos a los que pueden acceder. Al igual que las pilas, deben cumplir las normas de seguridad de apilamiento y una limitación adicional de peso máximo dependiente del transporte. En general suele ser:

- 105 tn peso máximo en un vagón.
- 52 tn peso máximo en un camión.

En la Figura 2.5 observamos un ejemplo de zona de transporte donde se encuentran dos camiones, uno con carga y otro descargado. Observamos también las bahías, que son las zonas designadas para el acceso de camiones, siendo más amplias que los mismos y permitiendo como máximo un vehículo.

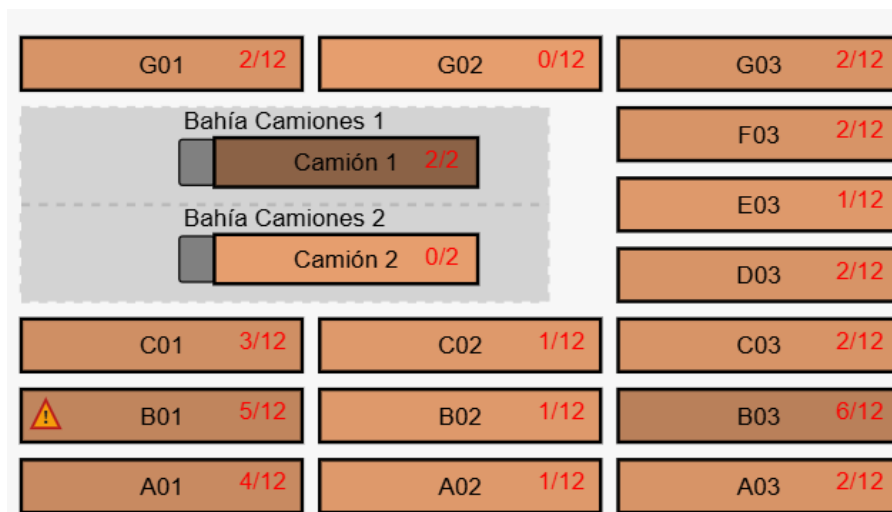


Figura 2.5: Zona de transportes representada en la aplicación.

El trabajo con transportes viene marcado por si se están utilizando para entrada o para salida. Según su uso la gestión que realiza el parque sobre los mismos es radicalmente diferente.

Cuando tratamos la entrada de material, el parque no tiene control sobre los desbastes recibidos, solo puede limitarse a descargarlos en las posiciones de almacenamiento que encajen mejor con las del desbaste. En este contexto, como se comenta en *Sistemas de Gestión de Almacenes (SGA) 2026*, es habitual usar sistemas de soporte a la decisión que ayuden a los operadores a tomar las decisiones multicriterio necesarias para la optimización del uso de almacenamiento.

Por el contrario, cuando tratamos la salida de material, a pesar de que nos encontremos limitados por las necesidades de las instalaciones consumidoras y otros parques, se dispone de una cierta flexibilidad sobre la elección de los materiales que van a ser cargados, en concreto la salida se produce con uno de los siguientes tipos:

- **Material de destino:** Se puede elegir cualquier desbaste disponible que tenga un destino que el transporte vaya a atravesar.
- **Material de programa:** Más restringido, el programa debe expedirse en orden estricto y es responsabilidad del parque que el transporte se encuentre perfectamente ordenado antes de su salida.

Evidentemente en ambos casos, la carga estará restringida por los criterios de seguridad y peso máximo del transporte.

Continuaremos nuestra explicación con las grúas, el elemento encargado de realizar los movimientos de los desbastes.

2.4.4. Grúa

Las grúas son los elementos encargados de desplazar los materiales en el parque. Pueden ser de diferentes tipos siendo los más comunes: puente, pórtico y carretillas. Sea cual sea su tipo, su propiedad más importante serán las toneladas que puede transportar, lo que, junto con el accesorio de desbastes, determinará el máximo de elementos que se pueden transportar.

Existen dos tipos principales de accesorios que comparten la restricción de solo ser capaces de manipular materiales que se encuentren en el tope del elemento de almacenamiento.

- **Electroimán:** Limitada a un único desbaste.
- **Pinza:** Su forma permite coger los desbastes aplicando fuerza en el lateral de la pieza. Según su tamaño pueden manipular entre uno y cuatro desbastes. Su aspecto se puede observar en la Figura 2.6.

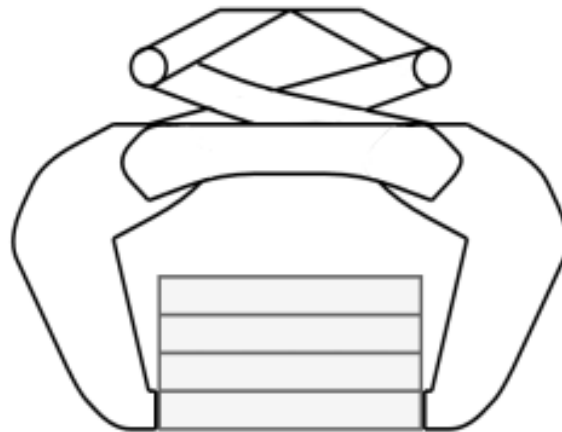


Figura 2.6: Esquema pinza de desbastes con 4 materiales.

Debido a la forma de la pinza, pueden existir restricciones dimensionales para los desbastes que se encuentren colocados encima del material que engancha, para garantizar que no se obstaculiza el contacto con la pieza inferior y que el gruista puede ver el enganche antes de izar el material. En nuestro caso las limitaciones son las siguientes:

- La diferencia de ancho entre el primer desbaste y cualquier otro de la pila no debe exceder

160 mm.

- La diferencia de ancho entre el primer desbaste y el segundo no debe exceder 80 mm.

Una consideración adicional que merece la pena resaltar es que las velocidades de movimiento, tanto de las grúas puente como pórtico, son mayores en el eje que no desplaza el puente, ya que no debe moverse el peso de la grúa. Este hecho es de importante consideración a la hora de evaluar las distancias que han de recorrer durante la manipulación del material.

A continuación, examinamos el funcionamiento de los programas, que son una parte vital del parque de desbastes ya que su principal responsabilidad es tratar con los mismos.

2.4.5. Programas

Los programas son listas ordenadas de desbastes que establecen el orden en que debe entregarse el material a la instalación que realiza el laminado en caliente. Cuando un desbaste está asignado a un programa cuenta con las siguientes propiedades:

- **Número de secuencia:** Indica la posición del material en la lista.
- **Grupo:** Indicador de compatibilidad que permite intercambiar números de secuencia dentro del mismo grupo. De esta forma se indican subgrupos compatibles y proporcionan flexibilidad al parque para reducir la necesidad de reordenar.

La composición de un programa no es aleatoria y sigue unos criterios específicos dictados principalmente por los rodillos encargados de la laminación en caliente. Una buena descripción de los mismos se encuentra en Y.-W. Chen et al. 2012. El proceso es el siguiente:

1. Se calientan los desbastes en un horno hasta temperaturas cercanas a los 900 °C.
2. Se hace pasar el desbaste múltiples veces a través de unos rodillos que reducirán su espesor y darán las dimensiones deseadas.
3. Una vez alcanzado el espesor objetivo se bobina el material y se deja enfriar a temperatura ambiente.

Se observa, por tanto, que los rodillos son la parte crucial del proceso y la que más sufre en el mismo, es por este motivo por el que se sustituyen cada programa. A continuación, describiremos brevemente las partes que diferenciamos dentro del programa y su relación con los requisitos dimensionales y sus tipos.

1. **Arranques:** este tipo de material comienza en anchos pequeños y tiene bajos requisitos de calidad, lo que permite empezar a calentar los rodillos. Puede tener uno de los siguientes

anchos (700, 800, 900, 1000, 1100) mm.

2. **Cabeceras:** este tipo cuenta con mayores necesidades de calidad, pero su función principal sigue siendo el calentamiento completo de los rodillos. Su ancho varía entre los 900 mm y los 1300 mm en incrementos de 50 mm.
3. **Tipo a laminar:** Corresponde al tipo concreto de material que se desea fabricar. Se comienza cuando los rodillos se encuentran en temperatura óptima. El ancho aumenta poco a poco hasta llegar al máximo del programa y empieza a decrecer. El mayor número de desbastes del programa pertenecerán a este tipo.

En la Figura 2.7 observamos los cambios de ancho de un programa y cómo va variando el ancho entre desbastes. En la parte denominada «Inicio» será donde encontraremos los «Arranques» y las «Cabeceras».

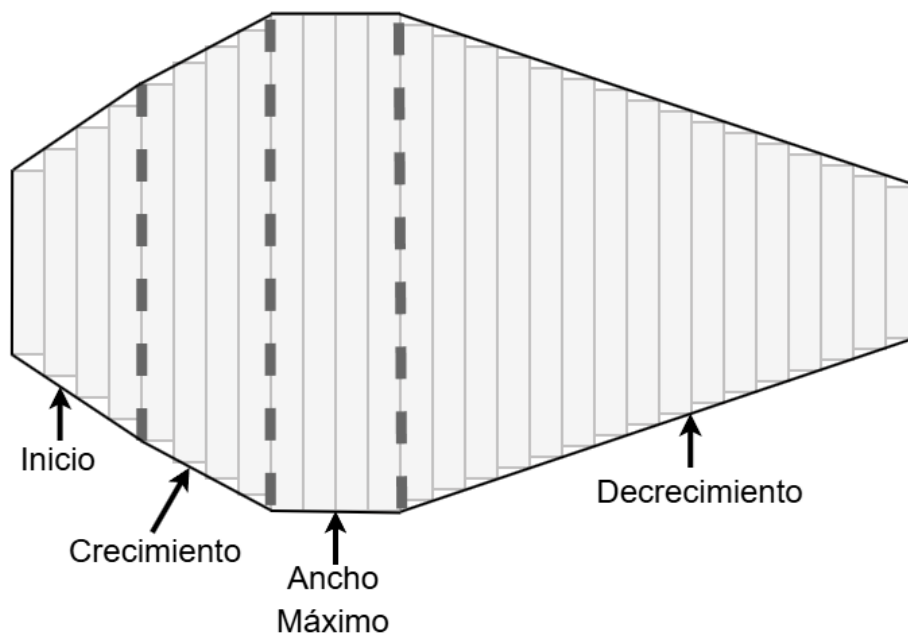


Figura 2.7: Representación de un programa de desbastes.

Como se deriva de la explicación anterior, el orden en el que se entregan los desbastes es muy importante, ya que determinará su entrada al horno y al proceso de laminación, en el cual, para garantizar la calidad, es necesario cumplir estrictamente los requisitos de orden.

Habiendo descrito el funcionamiento de las principales entidades del almacén, comentaremos ahora en mayor detalle las problemáticas que se encuentran dentro de la operativa del parque.

2.5. Problemática específica

El funcionamiento de un parque de desbastes debe dar respuesta a ciertas problemáticas mientras garantiza lo siguiente:

- La trazabilidad completa del material.
- Minimización de los tiempos de entrada y salida.
- Eficiencia de movimientos y uso de grúa.
- Ordenación de programas.

A continuación, se explorarán detalles propios de la operativa que dificultan la consecución de los objetivos del almacén. Estos elementos revisten de especial interés ya que su control, es una de las formas principales en las que un sistema de decisión puede ayudar a mejorar la operativa del parque.

2.5.1. Variabilidad de los materiales

Los materiales que se fabrican y que, por tanto, llegan al almacén, dependen de una gran variedad de criterios y responden a necesidades de producción y demanda de los clientes. Por ejemplo, es posible que se reciba un gran pedido de un tipo de material y anchos similares, lo que ocasione una abundancia de ciertos tipos de material, mientras que otros solo cuentan con unos pocos desbastes en el parque. De forma similar, podría ocurrir que se fabrican materiales sin pedido debido a necesidades de producción y que tardan mucho tiempo en ser utilizados. Mientras que el parque no forma parte de estas decisiones, toda la información de la que se disponga al respecto puede incorporarse para ayudar a la mejora de las decisiones a tomar en su almacenamiento.

También es importante tomar en cuenta el estado de las instalaciones productoras y consumidoras, dado que, cuando hay una parada de la producción de material, el parque tiende a vaciarse y, cuando hay una parada en el consumo, a llenarse. En estos casos se puede optimizar para la ordenación o para el almacenamiento respectivamente, cambiando la estrategia que se utiliza para abrir pilas y el número de alturas a utilizar.

Otro caso relativamente común es la recepción de piezas desordenadas en ancho entre diferentes transportes, lo que ocasiona casos en los que no es posible realizar la ordenación ideal. Por ejemplo, cuando se reciben los desbastes más anchos después de los estrechos, ya no podemos colocarlos debajo de los estrechos sin movimientos adicionales, lo que limita las opciones de agrupación. Este caso se ilustra en la Figura 2.8 donde el desbaste «3» llega después de haber formado la pila, la ordenación resultante se muestra a la izquierda y la ideal a la derecha.



Figura 2.8: Vista lateral desbaste ancho recibido tarde.

Del análisis anterior, deducimos que los siguientes criterios podrían ser de utilidad a la hora de tomar decisiones de cómo realizar el almacenamiento:

- Fecha estimada de laminación.
- Estado de las instalaciones productoras / consumidoras.
- Información de los pedidos (Número de desbastes de tipo y anchos).

Finalmente, el problema más importante de la variabilidad viene dado por los materiales en sí, ya que la combinación posible de características físicas y dimensionales resultan en un inventario que solo cuenta con uno o unos pocos materiales iguales. Si unimos eso a las restricciones de almacenamiento, seguridad y manipulación, tendremos la situación en la que cada apilamiento resulte en un compromiso de criterios en el que las decisiones tomadas no siempre resultan ser las óptimas.

2.5.2. Gestión de la entrada / salida

La función del almacén está ligada estrechamente a la entrada y salida de material, es por este motivo que su consideración tiene una gran importancia para la gestión y buen funcionamiento de este. Las consideraciones para decidir las zonas de transporte empiezan desde el momento en que se diseña el parque, estableciendo dentro de las restricciones de espacio constructivo, accesibilidad a las grúas y seguridad de los operarios. En particular se balancearán los siguientes criterios:

- **Distancia media a las pilas:** Como es evidente, cuanto más lejos estén situadas las zonas de transporte, más recorrido han de efectuar las grúas en la manipulación de material, lo que está directamente relacionado con el tiempo total de carga / descarga.
- **Espacio de almacenamiento:** Cuanto más espacio dediquemos a zonas de transportes, menos espacio estará disponible para el almacén, por lo que balancear correctamente la capacidad de almacenamiento y el espacio reservado para transportes es una decisión que dependerá de lo que queramos optimizar.

- **Interacción con las grúas:** Las grúas deben poder manipular elementos dentro de las zonas de transportes, lo que puede implicar que sus carriles entren en conflicto con vías de tren o pasos de camiones. Mientras que ambas situaciones son posibles, implican una gestión para prevenir colisiones y potencialmente un mayor coste de mantenimiento.
- **Seguridad de operarios:** La presencia de cargas suspendidas de gran tamaño y el movimiento de grúas de gran tonelaje requiere una especial atención a la seguridad de las personas que desempeñan su trabajo en el almacén. Suele ser habitual restringir la operación de las grúas cuando hay operarios en el parque, por este motivo, resulta más sencillo ubicar las zonas de transportes en laterales donde es más sencillo habilitar zonas seguras.

Para la toma de estas decisiones suelen realizarse múltiples estudios y simulaciones, ya que son decisiones difíciles de cambiar una vez que el almacén entra en funcionamiento. Además, hay que destacar que los transportes son un recurso generalmente limitado y, por lo tanto, se debe minimizar la duración de las operaciones de carga/descarga. Para reducir este problema es habitual establecer operativas para minimizar el tiempo de las operaciones. A continuación, se describen algunas de ellas:

- **Material de destino adyacente:** El establecimiento de zonas de almacenamiento destinadas únicamente a desbastes con destino, ubicadas al lado de zonas de transporte, con el objetivo de realizar una carga más rápida. El impacto evidente es la necesidad de bloquear espacio de almacenamiento, que potencialmente no puede ser ocupado a máxima capacidad debido a restricciones de seguridad.
- **Pilas reservadas para descarga:** Al igual que en el caso anterior, se reserva espacio para descargar los transportes según llegan, sin considerar la ubicación óptima, y posteriormente clasificar el material. El coste en este caso afecta tanto a la capacidad de almacenamiento como en tiempo de operación debido a la necesidad de realizar movimientos extra.
- **Zonas de ordenación de programas:** Se establecen zonas dedicadas a la clasificación de programas, con disponibilidad de pilas libres para la ordenación. Proporcionan mayor velocidad a la carga y a la ordenación ya que minimizan la distancia recorrida en los movimientos de ordenación. El coste que ha de asumirse es la reserva de espacio y el movimiento extra a destino. Merece la pena destacar que, ordenar programas fuera del activo también puede acarrear el riesgo de que el programa se cancele y sea un trabajo perdido.

Se observa por lo general que estas técnicas tienen la ventaja de ser adaptables a las necesidades del parque, ya que no dependen de ninguna forma de la construcción del parque, y pueden ser incorporadas y configuradas en los sistemas de soporte a la decisión.

El siguiente aspecto clave para la descarga viene dado por las diferentes formas de componer una ganchada. Cuando tenemos una pila con tres desbastes, podemos realizar los siguientes movimientos para su descarga, cada uno de los cuales resultará en un estado final del parque diferente.

- **Opción 1:** Coger los tres elementos en un solo movimiento.
- **Opción 2:** Coger los tres elementos en tres movimientos diferentes.
- **Opción 3:** Coger los dos primeros elementos juntos y un movimiento para el tercer elemento.
- **Opción 4:** Coger el primer elemento y otro movimiento para los dos siguientes.

Las opciones anteriores se ilustran en la Figura 2.9.

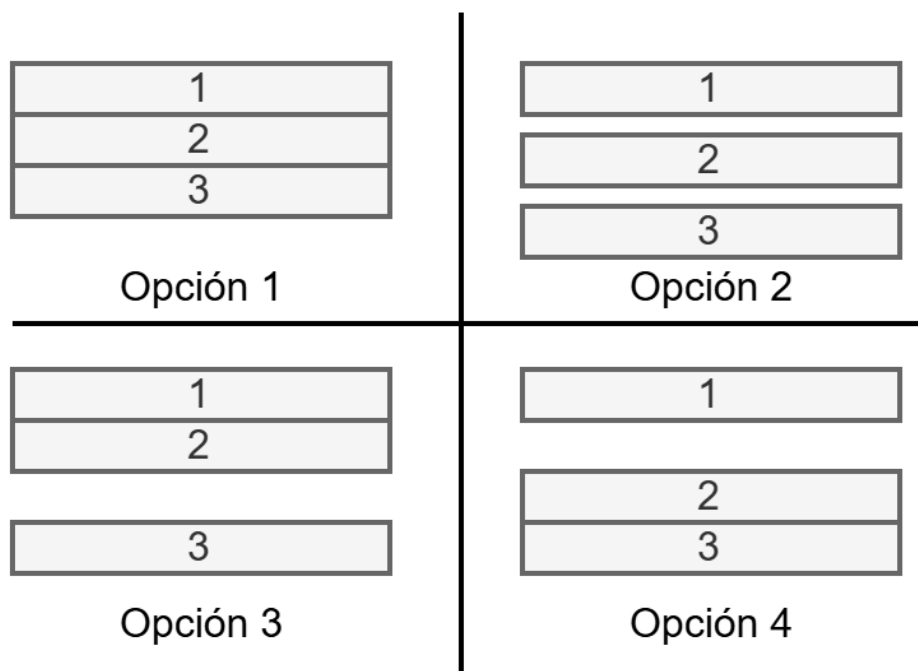


Figura 2.9: Formas de extraer de una pila de tres desbastes.

Teniendo en cuenta que en los transportes encontramos como máximo 4 desbastes, la Tabla 2.1 muestra el número de formas que hay de extraer los desbastes.

Tabla 2.1: Número de formas de extracción según tamaño pila.

Tamaño Pila	1	2	3	4
Nº Formas de extracción	1	2	4	8

Sin embargo, el problema de la combinatoria no termina aquí, ya que la descarga puede alterar

nar desbastes de cualquier pila del transporte, realizando extracciones de entre uno y cuatro desbastes. Por ejemplo, consideremos un caso de dos pilas con dos desbastes cada una.

Pila A: (2 desbastes) Extracción vía 2 movimientos.

Pila B: (2 desbastes) Extracción vía 2 movimientos.

Resultaría en las siguientes posibles combinaciones:

- AABB
- ABAB
- ABBA
- BAAB
- BABA
- BBAA

Resultando en un total de 6 opciones, a los que han de sumarse los derivados de haber elegido, para las dos pilas, la opción de extraer vía un único movimiento y en las que de una pila se extrae de un movimiento y la otra de dos, lo que resulta en un total de $6 + 2 + 3 + 3 = 14$. Observamos que incluso este caso sencillo resulta en muchas formas de mover los desbastes al parque.

No es objetivo de este trabajo la demostración matemática de la complejidad combinatoria, pero queda en evidencia que el número de opciones posibles es lo suficientemente alto para requerir del uso de sistemas de soporte a la decisión. La idea anterior se refuerza aún más cuando tenemos en cuenta que, para el caso habitual de un tren con seis vagones con cuatro desbastes cada uno, tenemos un total de $1,348 \cdot 10^{16}$ formas de llevar a cabo la descarga.

En esta sección se ha discutido aspectos operativos a considerar durante el flujo entrada / salida de desbastes, continuaremos la exposición del funcionamiento considerando cómo se organiza el trabajo de las grúas.

2.5.3. Planes de movimientos

En esta sección describiremos cómo se organiza el trabajo de las grúas, es decir, los movimientos que han de realizar. Los planes de movimientos son listas ordenadas que describen cada una de las operaciones que han de realizarse con los desbastes, indicando su origen, su destino y los materiales que deben moverse.

Las propiedades principales de un plan de movimientos son:

- **Orden:** Representa la prioridad de un plan de movimientos respecto a otros.
- **Nombre:** Texto libre que sirve a los operarios para describir las operaciones y diferenciarlas.
- **Estado:** Describe el estado de la operación.
 - Pendiente: El plan está definido pero aún no ha comenzado.
 - En curso: Alguno de los movimientos de los que consta el plan ha comenzado.
 - Completado: Todos los movimientos han sido realizados.
 - Cancelado: Se ha abortado su ejecución.
- **Tipo:** Especifica cuál de las cuatro operaciones principales del parque va a realizarse.
 - Descargar Transporte: Descargas de trenes o camiones que llegan al almacén.
 - Liberar Pilas: Retira todos los desbastes de una pila.
 - Cargar Programa: Carga de forma ordenada el programa hasta la capacidad máxima del transporte.
 - Reubicar: Movimientos libres entre cualquier elemento de almacenamiento.

Merece la pena repasar los tipos de plan ya que cada uno cuenta con su propio conjunto de problemas que debe afrontar. Comenzamos por la descarga de transportes, cuya complejidad se ha comentado en el apartado anterior. En este caso, cuando definimos planes, interesará tener en cuenta el orden en el que se descargarán los transportes presentes. Para tomar esta decisión, los operadores no solo atienden a criterios de ordenación sino también a necesidades sobre el uso de los transportes para la carga o urgencia para liberarlos.

La liberación de pilas también surge dentro de varias operativas y es una medida común, utilizada principalmente para compactar el espacio de almacenamiento, liberar zonas para destinarlas a ordenación o destino y garantizar la existencia de pilas donde ubicar los desbastes más anchos. La liberación de pilas es una decisión del operador pero una muy crítica, ya que en muchos casos la extracción del material de programa y destino ocasiona la ruptura de pilas. En este caso, la liberación de las pilas actúa como acción compensatoria.

El tipo carga de programa es también vital ya que se encarga de realizar la ordenación de los desbastes, que no se encuentran ya ordenados, para cargar las secuencias más prioritarias en el transporte. En esta operación, la complejidad radica en la diferenciación de tres tipos de movimientos, los destinados a apartar material que no es de programa o es de una secuencia que no va a ser cargada, los que reordenan los desbastes, para garantizar la entrega en orden, y los que finalmente los colocan en el transporte. Merece la pena destacar que este proceso puede necesitar usar pilas vacías para realizar la ordenación, cuando no es posible utilizar los propios transportes para ello.

Finalmente el tipo reubicar se encarga de manejar todos aquellos otros movimientos que los operadores deseen realizar con total libertad. Por ejemplo, cargas de material con destino, reubicaciones de materiales concretos, ordenación manual de programas,...

Además de lo anterior, debe considerarse que cada plan de movimientos tiene potencialmente dependencias con otros, al retirar o almacenar desbastes en pilas que son modificadas por otro plan. Esto añade una complejidad adicional que, no solo dificulta la definición de trabajo futuro a los operarios, sino la fragilidad del mismo ya que un cambio puede propagarse en cascada invalidando planes futuros.

2.6. Tecnologías empleadas

En esta sección se describen las tecnologías y herramientas seleccionadas para el desarrollo del sistema. De cada una de ellas se indica su finalidad general y el uso que se hace de la misma dentro del proyecto.

- **.NET:** Plataforma de desarrollo de Microsoft que ofrece un ecosistema de bibliotecas, herramientas y un entorno de ejecución para la creación de aplicaciones modernas. Se utiliza como base del sistema, implementando la lógica de negocio necesaria para el servidor y servicios de aplicación.
- **ASP.NET Core:** Framework de desarrollo de aplicaciones web incluido en .NET. Permite desarrollar servicios web altamente escalables, seguros y que son compatibles con múltiples servidores web. Se utilizará para desarrollar el acceso vía API a las funcionalidades del servidor.
- **React (TypeScript):** Biblioteca para el desarrollo de interfaces de usuario basada en componentes, utilizando TypeScript como lenguaje de programación tipado. Permite construir aplicaciones web dinámicas, mantenibles y con una mejor detección de errores durante el desarrollo. Se utiliza para implementar la interfaz web con la que interactúan los usuarios del sistema.
- **SQL Server:** Sistema de gestión de bases de datos relacional desarrollado por Microsoft. Permite almacenar, consultar y administrar información de forma segura y eficiente. Se utiliza para gestionar la persistencia de los datos del sistema, incluyendo la información de materiales, ubicaciones y operaciones realizadas.
- **Keycloak:** Plataforma de gestión de identidades y control de acceso de código abierto. Permite centralizar la autenticación de usuarios y la autorización mediante roles y permisos. Se utiliza para gestionar el acceso seguro a la aplicación, controlar los permisos de los distintos tipos de usuario y la implementación de proveedores de identidades externos.
- **.NET Aspire:** Conjunto de herramientas y componentes para facilitar el desarrollo y la

orquestración de aplicaciones distribuidas basadas en .NET. Permite configurar, coordinar y supervisar los distintos servicios que forman parte del sistema. Se utiliza para gestionar la ejecución conjunta de los diferentes componentes de la aplicación durante el desarrollo y facilitar su despliegue.

2.7. Conclusiones

En este capítulo se ha analizado el contexto en el que se enmarca el proyecto, revisando tanto las características de los almacenes intermedios siderúrgicos como las soluciones de gestión existentes y las principales investigaciones relacionadas con la optimización de su gestión. La revisión bibliográfica ha puesto de manifiesto que, aunque existen numerosas propuestas para problemas similares, no se ha encontrado una solución que integre simultáneamente las restricciones y necesidades propias del escenario planteado.

También se han revisado en detalle todos aquellos aspectos que diferencian el funcionamiento y las necesidades de un almacén de desbastes respecto a otros tipos de almacenes. Se ha explicado cómo encaja el parque dentro del proceso siderúrgico y sus responsabilidades y objetivos.

Se observa que, aunque en principio el sistema y sus elementos son relativamente sencillos, la variabilidad de los materiales, las restricciones de manipulación de desbastes y la gestión de movimientos generan problemas de gran complejidad. Los casos ilustrados claramente requieren de sistemas de soporte a la decisión, que faciliten la operación del almacén y que ayuden a cumplir los objetivos del parque.

Asimismo, se han presentado las tecnologías seleccionadas para el desarrollo del sistema, justificando su papel dentro de la arquitectura de la aplicación. La combinación de estas herramientas proporciona una base adecuada para implementar una solución moderna, mantenible y capaz de satisfacer los requisitos del proyecto.

En el siguiente capítulo abordaremos el diseño del sistema propuesto para reducir la complejidad de los procesos descritos junto con las necesidades genéricas de trazabilidad y control propias de un almacén.

Capítulo 3

Diseño

3.1. Introducción

En este capítulo se comienza revisando la arquitectura diseñada para GenStock, continuando con una revisión de las tecnologías utilizadas, justificando el uso de las mismas, y cómo se emplean para cubrir las necesidades propias del almacén.

Se continúa realizando una revisión de los modelos de dominio que dotan de la funcionalidad principal a la aplicación, haciendo énfasis en las entidades participantes y las reglas de negocio de las que son responsables.

Después se revisa el modelo relacional propuesto para la persistencia de la información de la aplicación en el sistema de base de datos.

A continuación, se abordan las necesidades de simulación, que son necesarias para dotar a la aplicación de un funcionamiento realista durante la ejecución de la misma fuera de un entorno productivo. Se proponen estrategias que permitan la realización de pruebas y la verificación funcional de la aplicación.

Finalmente, se realiza un inventario de las especificaciones que ha de cumplir GenStock para satisfacer el diseño realizado durante el capítulo, que servirán como referencia para las pruebas funcionales de la aplicación.

3.2. Arquitectura software

Las aplicaciones de gestión de inventario, como la mayoría de las aplicaciones modernas que se desarrollan para su uso en entornos corporativos, deben ser fiables, escalables y, sobre todo, modulares. En este diseño se pone el foco en esta característica, apostando por una solución en la que se utilizan múltiples aplicaciones web que colaboran para formar el producto final.

Partiendo de los objetivos anteriores, la solución adoptada estará compuesta por las siguientes aplicaciones web:

- **Cliente Web:** Aplicación web responsable de la interfaz de usuario. Será una SPA (Single Page Application) construida en TypeScript usando el framework React.
- **Simulación Operarios:** Aplicación responsable de ejecutar una simulación continua que imitará a un operario de grúa que realiza los movimientos de los desbastes en el almacén. Estará desarrollada en C# en forma de servicio de fondo.
- **Gestor de Programación:** Aplicación web responsable de gestionar la generación de programas así como la creación y cambios sobre los materiales del almacén. Estará desarrollada en C# en forma de servicio de fondo.
- **Servidor API:** Aplicación web responsable de la lógica de negocio de la aplicación y de la persistencia necesaria para garantizar la completa trazabilidad del estado del almacén. Será una aplicación web construida en C# y cuyo código estará organizado siguiendo los principios de arquitectura limpia establecidos por Martin [2012](#).
- **Servidor de Autenticación y Autorización:** Aplicación web responsable de la gestión de la autenticación y autorización, tanto para usuarios como aplicaciones. Para ello se configura KeyCloak, un servidor de código abierto que proporciona todas las funcionalidades necesarias.

A continuación, se muestra un diagrama de arquitectura de la aplicación donde se observan las distintas aplicaciones que componen la solución adoptada.

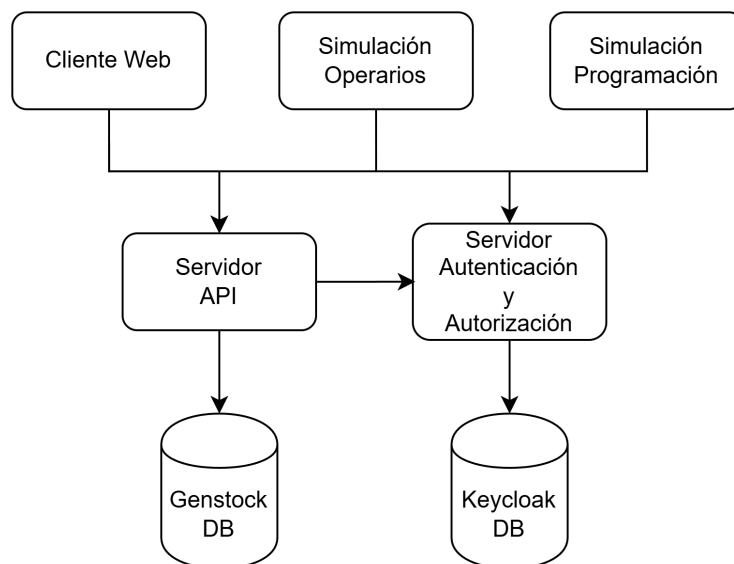


Figura 3.1: Diagrama de arquitectura de la aplicación.

3.2.1. Arquitectura del servidor

El servidor contiene la lógica de negocio del sistema y expone los servicios necesarios para la interacción con el cliente. En su diseño se ha buscado aislar las reglas de negocio de los aspectos tecnológicos, con el objetivo de facilitar la evolución y prueba del sistema. Para ello se emplea una arquitectura inspirada en los principios del diseño guiado por el dominio (Domain-Driven Design, DDD) Evans 2004 y arquitectura limpia (Clean Architecture).

La utilización de un modelo de dominio permite representar los conceptos propios del almacén, como materiales, ubicaciones, programas o planes de movimiento, evitando trasladar la lógica de negocio a las capas de infraestructura o presentación. Siguiendo este enfoque, las entidades son responsables de garantizar las reglas de negocio que les afectan, asegurando la consistencia del dominio y centralizando el conocimiento.

Dada la complejidad y el grado de interrelación existente entre los distintos procesos del almacén, el dominio se divide en varios contextos. Cada contexto encapsula un subconjunto del dominio, centrado en un problema concreto y exponiendo únicamente la funcionalidad necesaria. Organizar el conocimiento de dominio de esta forma permite reducir el acoplamiento y facilita la implementación de nuevas funcionalidades. A continuación, se presentan brevemente los contextos del modelo, que serán explicados en más detalle a lo largo del capítulo:

- **Inventario:** Responsable de la gestión de las ubicaciones donde se almacenan los materiales. Mantiene la información relativa a dónde se encuentra cada desbaste.
- **Movimientos:** Responsable de registrar y gestionar los desplazamientos de material que ocurren en el inventario. Mantiene información sobre los cambios de posición que se han realizado, generando trazabilidad de las operaciones.
- **Material:** Responsable de gestionar la información de los desbastes. Mantiene la información dimensional y de programación de cada material. Este contexto se mantiene separado de los contextos de inventario y programación porque su ciclo de vida es diferente. Un material existe de forma independiente de sus planificaciones, movimientos o asignaciones a programas.
- **Planes de Movimientos:** Responsable de gestionar las operaciones de movimiento que especifican la planificación de los desplazamientos de materiales.
- **Programas:** Responsable de gestionar los programas de laminación y la asociación de los desbastes a los mismos.
- **Propuesta de ubicación:** Responsable de gestionar los resultados de la propuesta de ubicación e información de diagnóstico de la ejecución del algoritmo A*. Debido a su complejidad, la implementación del algoritmo se realiza como un servicio externo.

El principal beneficio de esta división en contextos es la gestión de las relaciones existentes entre los distintos modelos del dominio. Conviene destacar que, prácticamente todos los contextos, mantienen algún tipo de relación con los materiales. Sin una separación, el modelo presentaría un elevado acoplamiento y una complejidad mucho mayor.

Las entidades únicamente se relacionan dentro de cada contexto y definen los siguientes elementos:

- **Agregados:** Utilizados como entidad raíz que protege la consistencia del conjunto de entidades y expone únicamente las operaciones necesarias para modificar su estado.
- **Objetos de valor:** Utilizados para imponer restricciones de dominio a los tipos básicos del lenguaje. Por ejemplo, la definición del ancho de un desbaste ha de garantizar que sólo los valores en el rango [700, 2500] mm son válidos.
- **Eventos:** Comunican la ocurrencia de cambios de estado que pueden utilizarse para orquestrar flujos de negocio entre distintos contextos.
- **Especificaciones:** Codifican las reglas para seleccionar objetos según criterios de dominio. Por ejemplo, describen cómo localizar movimientos pendientes.

De forma transversal a los contextos se definen las interfaces de servicio necesarias para la comunicación e interacción con la infraestructura.

Las dependencias entre capas se establecen de forma que las capas internas no conocen la existencia de las externas. De este modo, el modelo de dominio permanece agnóstico a la tecnología empleada para la persistencia o la capa de presentación.

La Figura 3.2 muestra las capas y las dependencias entre ellas.

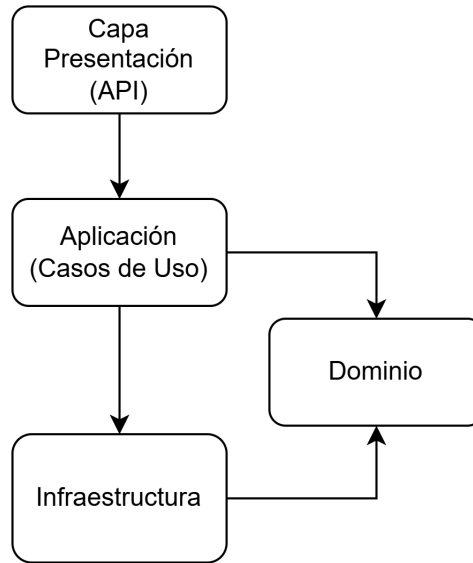


Figura 3.2: Dependencias entre capas en arquitectura limpia.

- **Presentación (API):** Formada por los controladores que exponen la funcionalidad del sistema a través de una API REST.
- **Aplicación (Casos de uso):** Implementa los casos de uso y coordina los servicios de infraestructura y los modelos de dominio.
- **Dominio:** Implementa los conceptos y reglas de negocio subyacentes a la aplicación. También define contratos que deben implementarse en la infraestructura.
- **Infraestructura:** Implementa la persistencia y comunicación con servicios externos, como el servidor de autenticación y la comunicación en tiempo real con el cliente.

Esta organización en capas permite desacoplar responsabilidades y facilita la evolución independiente de las distintas partes del sistema. En las siguientes secciones se describen con mayor detalle los elementos que componen cada una de ellas.

3.2.2. Cliente

El cliente permite la interacción del usuario con las funcionalidades de GenStock para la gestión del almacén. Se implementa como una aplicación web de página única (Single Page Application, SPA) desarrollada con React y TypeScript. Esta arquitectura permite separar la interfaz de usuario de la lógica del servidor, y proporciona navegación sin necesidad de recargar la página completa tras cada interacción. El uso de TypeScript permite el tipado de los contratos y facilita tanto el desarrollo de los componentes como la comunicación con el servidor.

La aplicación se organiza siguiendo una estructura modular. En un primer nivel, el código se

clasifica según su responsabilidad, diferenciando componentes visuales, servicios, proveedores de estado y utilidades comunes. Dentro de cada una de estas categorías, la funcionalidad se agrupa por contexto funcional, facilitando la localización del código relacionado con cada parte del sistema y reduciendo el acoplamiento entre módulos. Cabe destacar que, para reducir el acoplamiento y simplificar su uso, las páginas se encuentran definidas junto a los componentes que se utilizan en las mismas.

La navegación entre las distintas funcionalidades se realiza mediante enrutamiento en cliente utilizando la librería React Router.

La gestión del estado compartido se implementa mediante los elementos «context» de React a los que se accede mediante «hooks» personalizados. Este mecanismo permite desacoplar la lógica transversal de los componentes que la consumen y evita la propagación innecesaria de propiedades entre niveles de la jerarquía de componentes. Entre la información gestionada mediante proveedores se encuentran la autenticación del usuario, la conexión con SignalR, los temas visuales, la internacionalización y las notificaciones.

La comunicación con el servidor se realiza mediante servicios específicos para cada agregado del dominio. Estos servicios encapsulan el acceso a la API REST utilizando Axios, aislando los detalles de comunicación del resto de la aplicación. De forma complementaria, las actualizaciones en tiempo real se reciben mediante SignalR, permitiendo reflejar automáticamente en la interfaz los cambios producidos por otros usuarios o por procesos internos del sistema.

Para la construcción de la interfaz se emplea Material UI, que proporciona un conjunto de componentes reutilizables y un sistema de temas visuales, permitiendo mantener una apariencia homogénea en toda la aplicación. La representación gráfica del parque del almacén se implementa mediante Konva, una biblioteca especializada en dibujo 2D, elegida por facilitar la representación interactiva del mapa del almacén y de los elementos presentes en el mismo.

Las separaciones descritas permiten desacoplar las responsabilidades de presentación, gestión del estado y comunicación con el servidor, favoreciendo la evolución independiente de cada una de ellas y simplificando el mantenimiento del cliente.

3.2.3. Aspire

Para el desarrollo de aplicaciones distribuidas es muy útil contar con herramientas que nos permitan controlar las dependencias entre servicios y a la vez ofrezcan capacidades de monitorización. En este proyecto se emplea Aspire, Team 2026, para las siguientes funcionalidades auxiliares:

- Aprovisionamiento de los contenedores de Sql Server y Keycloak.

- Orquestador de la ejecución garantizando el orden de arranque.
- Paso de parámetros entre servicios.
- Herramienta de monitorización y gestión de trazas.

La forma en la que funciona esta herramienta es mediante la inclusión de un proyecto denominado *AppHost* donde definiremos los artefactos a incluir, las variables de entorno necesarias y el orden de dependencias entre los mismos. El ejemplo de Código 3.1 muestra una parte del fichero *AppHost* donde se define la necesidad de incluir un servidor Sql Server y se añade la dependencia con nuestra API:

```
1 var sql = builder.AddSqlServer("sql-server")
2     .WithOtlpExporter()
3     .WithLifetime(ContainerLifetime.Persistent);
4
5 var db = sql.AddDatabase("genstock-db");
6
7 var genstock_api = builder
8     .AddProject<Projects.GenStock_Api>("Backend")
9     // Incluye cadena de conexión
10    .WithReference(db)
11    // Espera a que la base de datos esté disponible
12    .WaitFor(db)
13    .WithReference(keycloak)
14    .WaitFor(keycloak)
15    // Configura la telemetría
16    .WithOtlpExporter()
17    // Añade variables de entorno
18    .WithEnvironment("SERVICE_CLIENT_ID", "genstock-api")
19    .WithEnvironment("SERVICE_CLIENT_SECRET", apiClientSecret);
```

Código 3.1: Configuración dependencias en Aspire.

El código anterior ilustra cómo se definen la estructura y las dependencias de los servicios del proyecto con Aspire, estableciendo dependencias, configuraciones de entorno y orden de arranque.

Otro de los beneficios que tiene la inclusión de Aspire en el proyecto es que pone a nuestra disposición una web de monitorización, donde podemos consultar en todo momento, el estado de nuestros servicios y toda la información de telemetría. En la Figura 3.3 se observa el panel de monitorización mostrando el estado de los servicios del proyecto.

Nombre	Estado	Hora de ini...	Origen	Direcciones URL	Acciones
keycloak-auth	Running	19:17:05	quay.io/keycl...	https://localhost:60830 +1	[Stop] [Logs] [More]
sql-server	Running	18:30:30	mcr.microsof...	tcp://localhost:60829	[Stop] [Logs] [More]
genstock-db	Running	18:30:30	-	-	[Logs] [More]
keycloak-db	Running	18:30:30	-	-	[Logs] [More]
Frontend	Running	19:32:25	npm run run	http://localhost:60831	[Stop] [Logs] [More]
Frontend-installer	Finished	19:32:21	npm install	-	[Play] [Logs] [More]
Backend	Running	19:32:24	GenStock.Ap...	https://localhost:7003/swagger +1	[Stop] [Logs] [More]
OperatorSimulation...	Running	19:32:25	GenStock.Op...	-	[Stop] [Logs] [More]
ProgramScheduling...	Running	19:32:25	GenStock.Pr...	-	[Stop] [Logs] [More]

Mostrando 9 recursos

Figura 3.3: Página principal Aspire.

La integración de la telemetría en Aspire utiliza el estándar abierto de OpenTelemetry Foundation 2019, cuya adopción en la industria es muy amplia y la mayoría de frameworks modernos lo soportan, esto simplifica enormemente su uso e instalación.

La monitorización del proyecto incluye los siguientes tipos de telemetría:

- **Trazas:** Las usaremos para seguir el flujo que sigue una petición a través de la aplicación. Además, OpenTelemetry soporta el concepto de unidades de trabajo, que permite la correlación de las operaciones originadas por otros servicios a través de una vinculación con la operación padre, formando una jerarquía. Esto nos permitirá entender fácilmente qué está ocurriendo en nuestros servicios.
- **Métricas:** Consisten en medidas obtenidas durante la ejecución de una aplicación. Por ejemplo, el número de peticiones activas, el porcentaje de uso de CPU, u otras.
- **Logs:** Son la forma de diagnóstico más utilizada históricamente. Consisten en una marca temporal acompañada de un nivel de criticidad y un mensaje o datos adicionales. La ventaja que ofrece OpenTelemetry al respecto es el soporte a las herramientas de log tradicionales de los lenguajes y la correlación de los datos con las trazas.

A continuación, en la Figura 3.4 se muestra un ejemplo de traza en Aspire donde se observa

cómo la petición llega al servicio de la API y ejecuta varias consultas a la base de datos antes de devolver la respuesta.

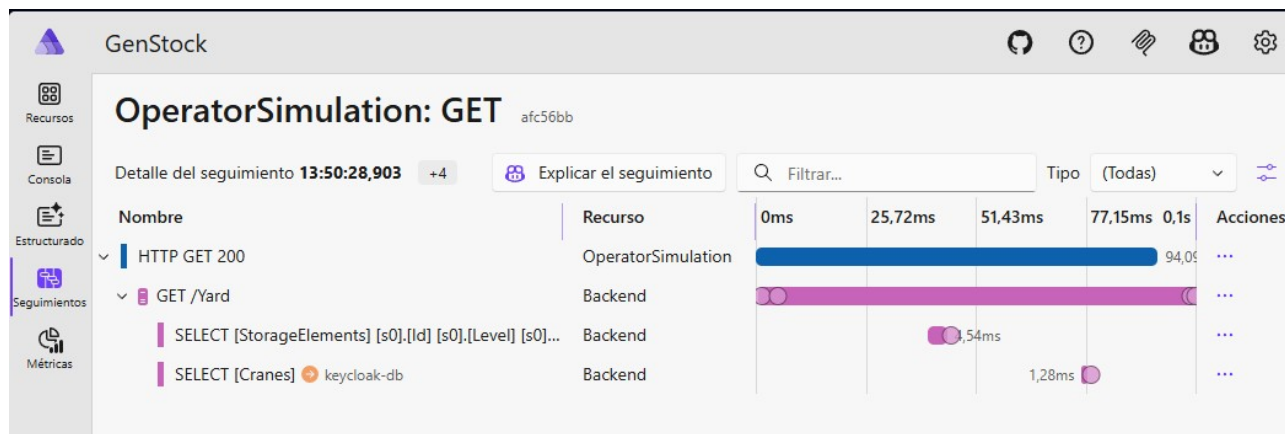


Figura 3.4: Ejemplo de traza en Aspire.

3.2.4. Keycloak

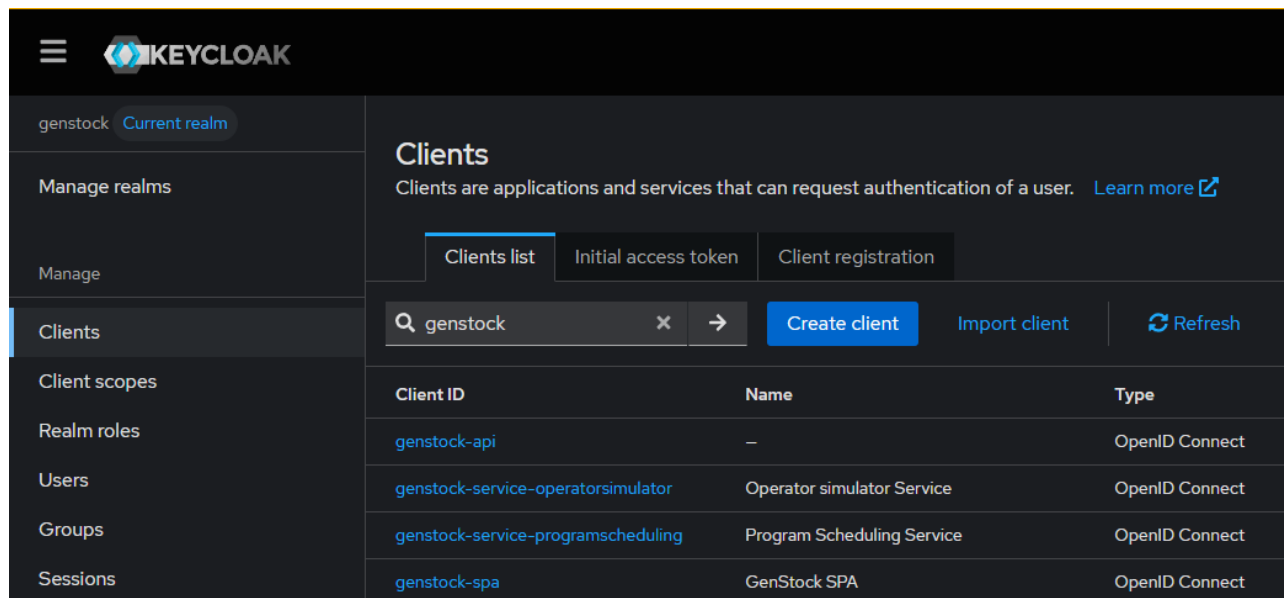
La arquitectura propuesta incorpora un servicio externo de identidad, encargado de la autenticación y autorización de usuarios y servicios. Para ello, se ha seleccionado Keycloak por ser un servicio de código abierto y amplia comunidad. La integración de este servicio nos proporciona las siguientes ventajas:

- **Delegación de la autenticación y autorización:** La aplicación no necesita implementar la gestión de usuarios, sesiones o permisos. De esta forma, la aplicación podrá centrarse en implementar la lógica de negocio.
- **Control de acceso entre servicios:** La utilización de este servidor permite autenticar los accesos a la API del cliente web y de los servicios de simulación usando diferentes flujos de autenticación.
- **Mayor nivel de seguridad:** La utilización de protocolos implementados en Keycloak como OAuth 2.0 y OpenID Connect evita la necesidad de añadir sistemas propios, que suelen introducir vulnerabilidades de seguridad. Además, el uso de JSON Web Tokens (JWT) facilita la comunicación entre el cliente y el servidor.
- **Implementación de permisos:** El servicio permite definir roles y asignarlos a los usuarios, por lo que la implementación de la seguridad en la aplicación puede limitarse a leer el token del usuario y verificar el rol necesario para autorizar una operación.
- **Facilidad de despliegue:** Es sencillo su despliegue e integración en nuestro sistema mediante la integración de un contenedor en Aspire.

Dentro de la arquitectura, el servicio de identidad asumirá las siguientes responsabilidades:

- Proveer de identidad a la aplicación principal y servicios de simulación.
- Proveer la funcionalidad de acceso y registro a la aplicación.
- Permitir el uso de cuentas de Microsoft como usuarios de la aplicación.

Para soportar los distintos flujos de autenticación, el servidor de identidad requiere la definición de diferentes clientes, roles y proveedores de identidad externos. Los clientes definidos para la solución pueden observarse en la Figura 3.5.



The screenshot shows the Keycloak administration interface. On the left is a sidebar with navigation options: 'genstock' (Current realm), 'Manage realms', 'Manage', 'Clients' (selected), 'Client scopes', 'Realm roles', 'Users', 'Groups', and 'Sessions'. The main panel is titled 'Clients' and includes a description: 'Clients are applications and services that can request authentication of a user. [Learn more](#)'. Below the title are tabs for 'Clients list', 'Initial access token', and 'Client registration'. A search bar contains 'genstock'. To the right of the search bar are buttons for 'Create client', 'Import client', and 'Refresh'. Below these is a table listing the defined clients.

Client ID	Name	Type
genstock-api	—	OpenID Connect
genstock-service-operatorsimulator	Operator simulator Service	OpenID Connect
genstock-service-programscheduling	Program Scheduling Service	OpenID Connect
genstock-spa	GenStock SPA	OpenID Connect

Figura 3.5: Clientes definidos en Keycloak.

Para la autenticación de usuarios se empleará el flujo Authorization Code con PKCE de OpenID Connect, recomendado para aplicaciones cliente ejecutadas en el navegador. Para ilustrar el proceso, la Figura 3.6 muestra una versión simplificada del flujo de autenticación. El usuario es redirigido a la web de Keycloak para la autenticación y, una vez completado el proceso, recibe un token JWT que se utilizará en las sucesivas peticiones al servidor de la aplicación.

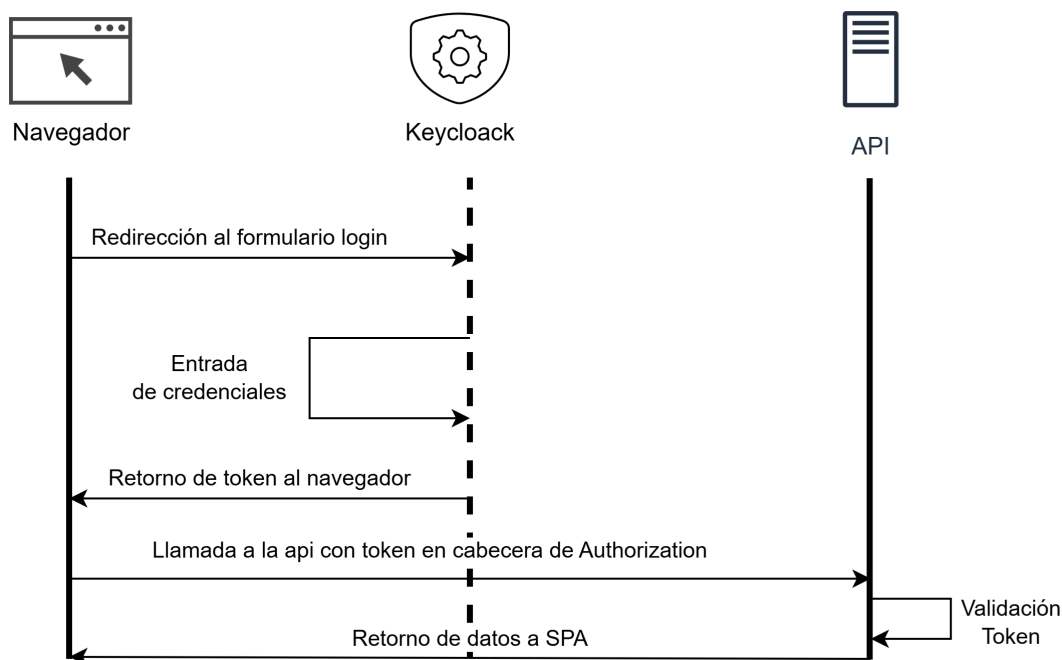


Figura 3.6: Flujo simplificado de autenticación OpenID Connect.

Este diseño nos permite desacoplar la autenticación del resto de la aplicación. El cliente redirige al servicio de autenticación y almacena el token, mientras que la aplicación principal se limitará a validar los tokens para autorizar las peticiones.

3.2.5. SignalR

La arquitectura propuesta incorpora un servicio para realizar comunicación en tiempo real entre cliente y servidor. Para ello, se ha seleccionado SignalR por su fácil integración en el ecosistema de .NET y su compatibilidad con distintos protocolos. La integración de este servicio nos proporciona las siguientes ventajas:

- **Permite el envío de eventos del servidor al cliente:** La aplicación servidora puede notificar fácilmente a los clientes de cambios que se produzcan en la información.
- **Reduce la carga del servidor:** Al ser el cliente notificado de eventos de forma proactiva, no se necesita consultar periódicamente al servidor para actualizar la información, lo que reduce el número de peticiones web.
- **Habilita el tiempo real:** Al realizar las comunicaciones mediante websockets se reduce la latencia en la recepción de los cambios, lo que permite la implementación de visualizaciones en tiempo real.
- **Soporte a autenticación:** Es posible configurar la conexión para que utilice el mismo mecanismo de autenticación que el resto del servidor. De esta forma, esta comunicación

también se realizará de forma segura.

En la aplicación, el servicio de SignalR tendrá las siguientes responsabilidades:

- Proveer la información de cambios en tiempo real que se utilizará en la visualización del estado del inventario, movimientos y grúas.
- Proveer de un mecanismo para recibir actualizaciones de progreso en tareas de larga duración.

En la Figura 3.7 se muestra el flujo de comunicación entre el servidor de aplicación (API) y los clientes.

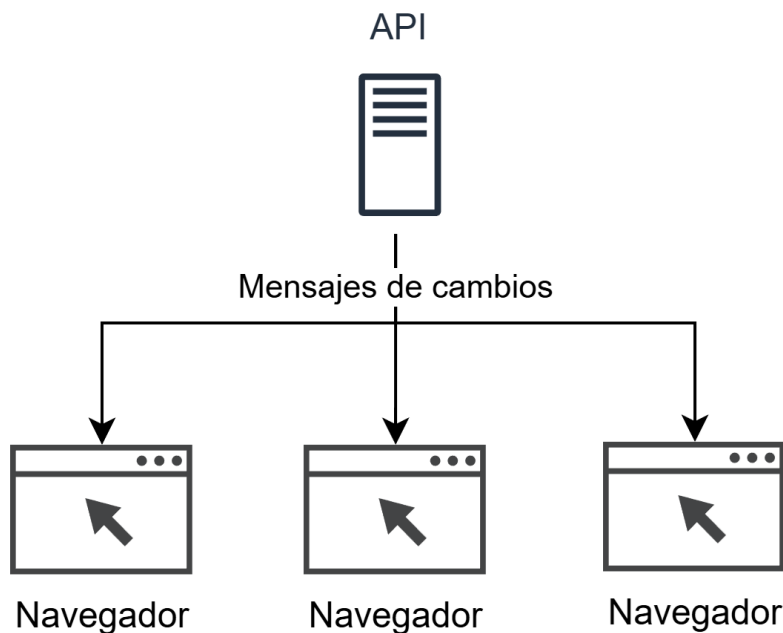


Figura 3.7: Comunicación SignalR entre API y clientes.

En ambos casos el servicio se emplea como canal de difusión, enviando eventos producidos en el servidor a todos los clientes activos.

3.3. Gestión del inventario

En esta sección se presenta el diseño del sistema de inventario que sirve como base a las funcionalidades de soporte a la decisión. La finalidad del sistema es proporcionar una estructura constituida por espacios de almacenamiento destinados a la ubicación de materiales, de manera que siempre se conozca la ubicación exacta de cada uno de ellos. En un almacén de desbastes, la gestión del inventario se realiza sobre un espacio físico finito que se divide en huecos de

almacenamiento localizados en el suelo y que se encuentran marcados con líneas o pivotes. Como consecuencia, el plano del parque puede requerir modificaciones por diversos motivos operativos, por ejemplo;

- Ajustar el tamaño del inventario.
- Modificar límites de seguridad entre elementos de almacenamiento o zonas de transporte.
- Modificar la ubicación o número de las bahías de camiones.

Además, debe garantizar la transaccionalidad de las operaciones cuando un material cambia de posición, evitar pérdidas de información y gestionar de forma consistente la entrada y salida de materiales.

Teniendo en cuenta la necesidad de flexibilidad en la definición del espacio del almacén, se opta por diseñar un sistema de almacenamiento basado en un fichero de configuración JSON para representar los elementos físicos. Esta solución permite la realización de cambios en el plano del parque sin necesidad de modificar el código de la aplicación. Su estructura se describe en el siguiente apartado.

3.3.1. Fichero de configuración del plano del parque

La estructura elegida para el fichero de configuración aprovecha la flexibilidad de la definición de datos y la posibilidad de anidar objetos propia de JSON. Como se observa en el Código 3.2, la configuración comienza mediante la definición del objeto «Yard» cuyas propiedades más relevantes son:

- **WidthInMeters:** Ancho en metros del área exterior del almacén. Se corresponde con el eje Y en la aplicación.
- **LengthInMeters:** Largo en metros del área exterior del almacén. Se corresponde con el eje X en la aplicación.
- **StorageElements:** Listado de objetos que pueden almacenar desbastes, puede contener elementos de tipo: «PileRack», «Truck» y «Wagon».
- **Zones:** Zonas que no pueden almacenar desbastes pero que tienen un significado lógico, se utiliza para definir áreas que tienen un significado lógico.
- **Cranes:** Definición de las grúas que operan en el almacén.

```
1 {  
2   "Yard": {  
3     "Id": "e527a4e5-be36-48da-b756-430b1eda6e39",
```

```

4      "Name": "Main Yard",
5      "WidthInMeters": 50,
6      "LengthInMeters": 280,
7      "StorageElements": [],
8      "Zones": []
9      "Cranes": []
10     },
11  }

```

Código 3.2: Definición objeto Yard.

En el Código 3.3 se muestra un ejemplo de la estructura que define el objeto «Crane» que representa una grúa del almacén. En la misma se identifican parámetros que serán críticos para su funcionamiento y representación en el sistema:

- **Id:** Cada una de las grúas debe estar identificada de forma única para permitir que los movimientos que realice queden inequívocamente asignados a la misma.
- **DefaultName:** Nombre inicial dado a la grúa, podrá ser modificado desde la aplicación.
- **CapacityInKg:** Refiere al máximo peso que la grúa puede levantar, no se incluye en el mismo el peso de elementos que forman parte de la grúa como la pinza. Este valor se utilizará para establecer limitaciones a los movimientos que puede realizar.
- **MaxItems:** Determina el número máximo de desbastes que pueden moverse a la vez.
- **BridgeOrientation:** Con valores «Horizontal» o «Vertical» especifica en cual de los ejes X o Y respectivamente se mueve el carro de la grúa.
- **MaxBridgeSpeedInMPerS:** Establece la velocidad máxima que tiene el carro de la grúa en metros por segundo.
- **MaxTrolleySpeedInMPerS:** Establece la velocidad máxima del puente de la grúa en metros por segundo.
- **WidthInMeters:** Indica el ancho que tiene la grúa en la representación del mapa en la aplicación.
- **LengthInMeters:** Indica el largo que tiene la grúa en la representación del mapa en la aplicación.
- **WorkingArea:** Este objeto se encarga de establecer el área de trabajo sobre la que la grúa puede operar tomando como base el punto superior izquierdo del área del almacén definido en el objeto «Yard».

```

1  {
2      "Id": "123e4567-e89b-12d3-a456-426614174001",

```

```
3  "DefaultName": "Crane A",
4  "CapacityInKg": 82000,
5  "MaxItems": 4,
6  "BridgeOrientation": "Vertical",
7  "MaxBridgeSpeedInMPerS": 2.4,
8  "MaxTrolleySpeedInMPerS": 1.2,
9  "WidthInMeters": 10,
10 "LengthInMeters": 10,
11 "WorkingArea": {
12   "TopLeftX": 3,
13   "TopLeftY": 3,
14   "WidthInMeters": 50,
15   "LengthInMeters": 280
16 }
17 }
```

Código 3.3: Definición objeto Crane.

En el Código 3.4 se define el objeto «PileRack» que representa un elemento de almacenamiento donde se colocarán desbastes. Las propiedades más relevantes son:

- **Id:** Cada uno de los elementos de almacenamiento debe identificarse de forma única para un funcionamiento correcto del inventario.
- **Name:** Nombre que se utilizará para visualizar e identificar el elemento en el mapa de la aplicación.
- **Position:** Establece la ubicación del punto superior izquierdo del almacenamiento en coordenadas relativas al espacio del parque.
- **WidthInMeters y LengthInMeters:** Indica el ancho y el largo que tiene el elemento de almacenamiento.
- **MaxElements:** Define el número máximo de elementos que puede almacenar este elemento.

```
1  {
2    "$type": "PileRack",
3    "Id": "79281b07-a5b4-424a-ad3a-8e6d66f46cb3",
4    "Name": "A01",
5    "Position": {
6      "X": 5,
7      "Y": 5
8    },
```

```
9      "WidthInMeters": 3,  
10     "LengthInMeters": 16,  
11     "MaxElements": 12  
12 }
```

Código 3.4: Definición objeto PileRack.

En el Código 3.5 se ilustra el objeto «Zone» que representa una zona lógica de almacenamiento diseñada para la ubicación de transportes móviles. La idea detrás de la misma es delimitar la zona donde se colocarán los transportes de un determinado tipo y que será visible en el mapa de la aplicación. Las propiedades más importantes son:

- **Id:** Identificador único de cada una de las zonas.
- **Name:** Nombre que se utilizará para visualizar la zona en el mapa de la aplicación.
- **TransportElement:** Enumeración de tipos de transporte soportados: «Truck» para camiones y «Wagon» para vagones de tren.
- **Position:** Establece la ubicación del punto superior izquierdo de la zona en coordenadas relativas al espacio del parque.
- **WidthInMeters y LengthInMeters:** Indican el ancho y el largo que tiene la zona.

```
1 {  
2     "Id": "9fe19005-855b-4332-9298-6c23ce8f8990",  
3     "Name": "Bahia Camiones 1",  
4     "TransportElement": "Truck",  
5     "Position": {  
6         "X": 5,  
7         "Y": 22.5  
8     },  
9     "WidthInMeters": 5.5,  
10    "LengthInMeters": 30  
11 }
```

Código 3.5: Definición objeto Zone.

A través de la configuración del fichero anterior obtenemos un sistema robusto que permite una fácil reconfiguración del espacio de almacenamiento, así como el uso de la aplicación en diferentes almacenes. En el siguiente apartado se presenta el modelo del inventario, encargado de representar el estado relativo al almacenamiento de los materiales.

3.3.2. Modelo del inventario

Mientras que el fichero de configuración describe la estructura física del almacén, el estado del inventario es dinámico y cambia continuamente, como consecuencia de las operaciones realizadas sobre los materiales. Por ello, el sistema requiere un modelo que represente la ubicación actual de cada material y garantice la consistencia del estado del inventario.

El modelo debe hacer cumplir las siguientes reglas:

- Una ubicación (pila + altura) solo puede contener un material.
- Un material solo puede ocupar una ubicación.
- El número de materiales puede exceder el máximo de elementos definido en la configuración del almacén. En este caso, consideraremos el apilamiento no seguro, aunque se permite para mantener la trazabilidad.
- Los transportes pueden estar o no en el parque.

Estas restricciones también constituyen la base para las operaciones de entrada, salida y movimiento de materiales descritas en los apartados siguientes. A continuación, se presentan las clases encargadas de mantener la información del inventario, representadas en la Figura 3.8:

- **StorageSlot:** Representa la ocupación de un nivel de almacenamiento por un material concreto. Además de la referencia al material, mantiene la altura que ocupa dentro del elemento de almacenamiento, siendo el 1 el nivel más bajo de cada elemento.
- **StorageElement:** Representa un elemento físico de almacenamiento. Proporciona el comportamiento común de los distintos tipos de almacenamiento. Entre sus responsabilidades destacan:
 - Mantener los materiales almacenados.
 - Permitir operaciones de extracción.
 - Permitir operaciones de almacenamiento.
- **Crane:** Representa el almacenamiento temporal en una grúa durante los movimientos.
- **PileRack:** Representa el almacenamiento en forma de pila.
- **Truck:** Representa el transporte en camión. Mantiene información sobre si se encuentra en el almacén o no.
- **Wagon:** Representa el transporte en un vagón de tren. Mantiene información sobre si se encuentra en el almacén o no.

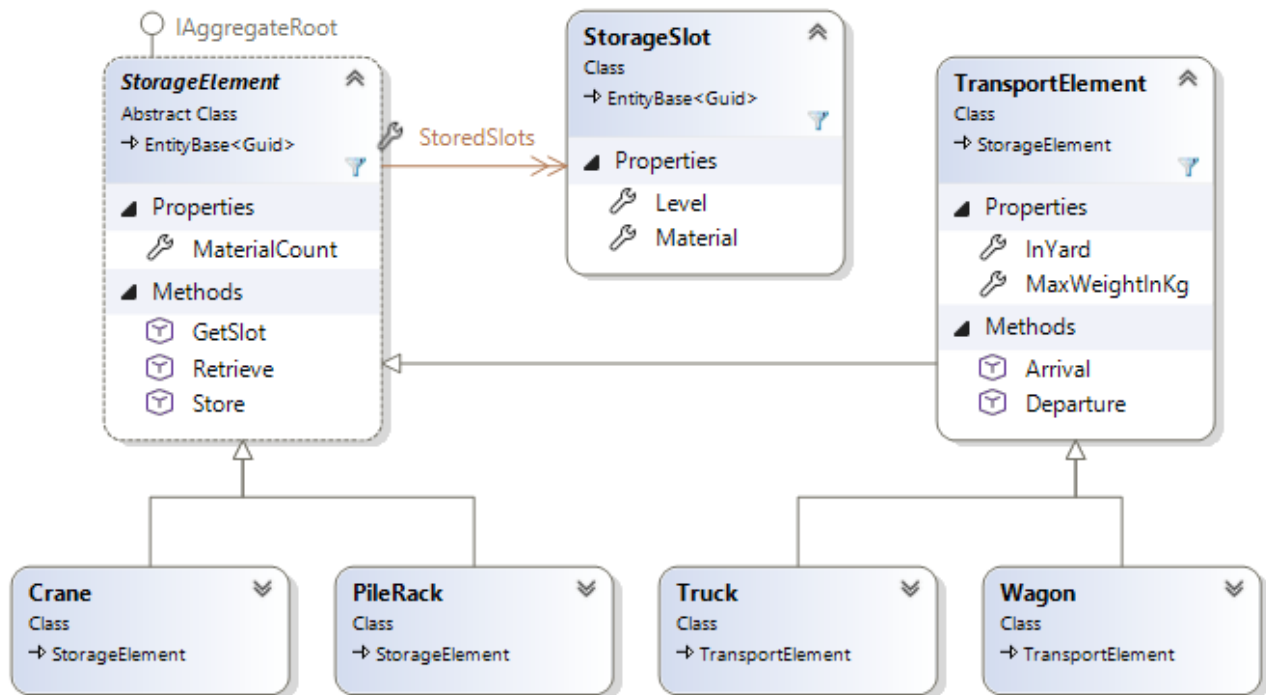


Figura 3.8: Diagrama UML del modelo de inventario.

A diferencia del modelo del plano del parque, que representa la distribución física del almacén, el modelo del inventario representa el estado operativo de dichos elementos mediante la ocupación de las distintas ubicaciones.

3.4. Modelo de movimientos

En esta sección se presenta el diseño del sistema de movimientos, encargado de mantener la trazabilidad del inventario, mediante el registro de todos los cambios de ubicación de los desbastes. En particular, deben registrarse dos tipos de movimientos: los movimientos realizados por elementos móviles como grúas y los realizados de forma lógica, desde la aplicación, para corregir posiciones. La información almacenada debe permitir dar respuesta a las siguientes necesidades operativas:

- Determinar dónde ha estado un desbaste en un momento dado y la composición de la pila en la que se encontraba. Esta información es de gran utilidad a la hora de investigar problemas de calidad que pueden haber surgido por golpes o la presencia de piedras u otros elementos.
- Trazar los movimientos realizados tanto por las grúas como los movimientos lógicos.
- Medir el rendimiento del parque a través del número de movimientos realizados en diferentes intervalos de tiempo.

El modelo debe registrar todos los cambios de ubicación en el inventario, incluyendo aquellos que involucran a las posiciones de la grúa. Este problema se puede plantear de dos formas:

- **Movimiento con dos elementos:** Se consideran únicamente un origen y destino. En el caso de que el movimiento lo realice una grúa, existirá un movimiento para coger el material y otro para soltarlo.
- **Movimiento con tres elementos:** Se incluye el elemento móvil además del origen y el destino. En caso de que se trate de una regularización, el elemento móvil no se informa.

Finalmente se ha optado por el modelo de dos elementos representado en la Figura 3.9 debido a que presenta las siguientes ventajas:

- Se puede implementar mediante sólo inserción, sin modificar registros ya almacenados.
- Evita el uso de campos nulos y la necesidad de introducir lógica específica para distinguir ambos tipos de movimientos.
- Requiere menos campos para almacenamiento de posiciones e información de fechas.

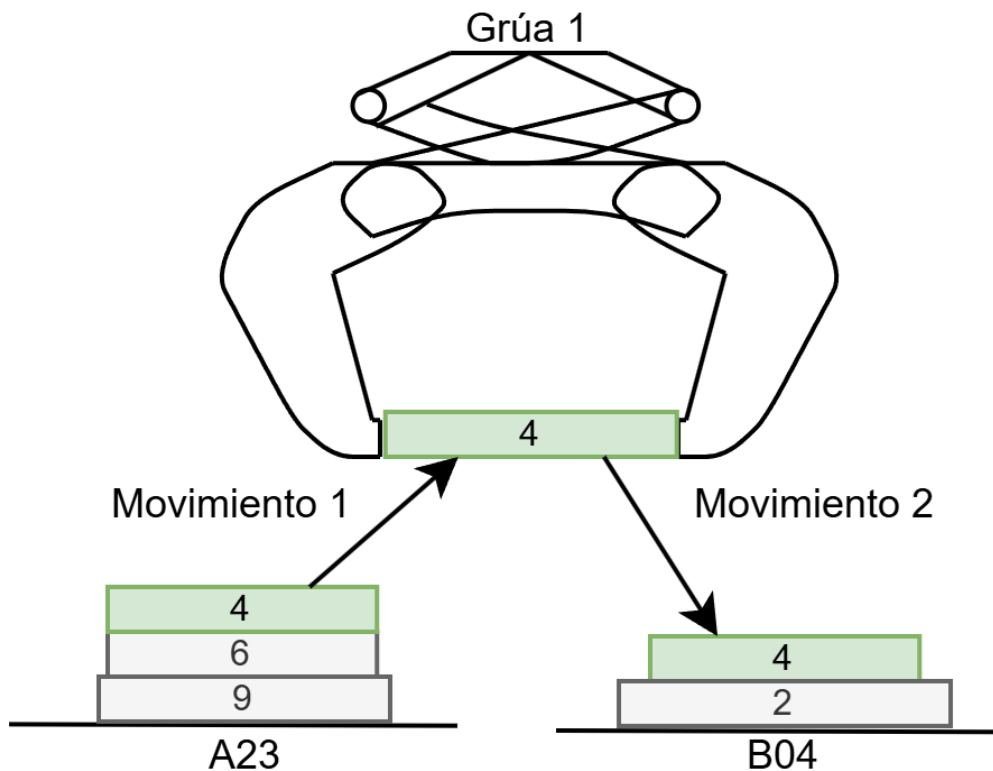


Figura 3.9: Modelo partes de movimiento.

Mediante este modelo, un movimiento realizado por una grúa queda representado por dos registros consecutivos. En la Tabla 3.1 se muestra cómo se registraría el movimiento representado

en 3.9.

Tabla 3.1: Ejemplo partes movimiento.

Movimiento	Material	Origen	Altura Origen	Destino	Altura Destino
Movimiento 1	4	A23	3	Grúa 1	1
Movimiento 2	4	Grúa 1	1	B04	2

A continuación, se presentan las clases encargadas de mantener la información de movimientos representadas en la Figura 3.10:

- **StockPosition:** Representa una posición del inventario junto a su altura. Puede ser una pila, una grúa, un camión o un vagón.
- **Movement:** Representa el traslado de un desbaste entre dos posiciones «StockPosition» del parque. Agrupa la identificación de los elementos implicados junto con información temporal asociada al movimiento.

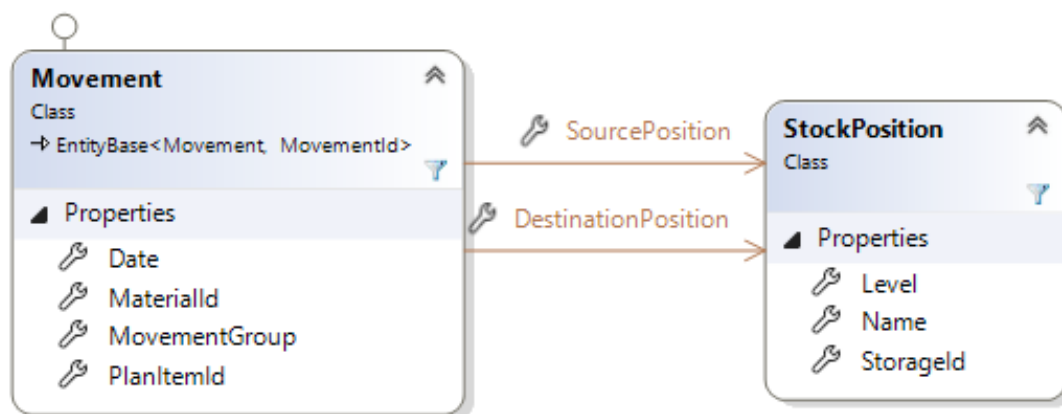


Figura 3.10: Diagrama UML del modelo de movimientos.

Mediante este modelo es posible recuperar el histórico completo de ubicaciones de cualquier desbaste, analizar los movimientos realizados en el parque y cumplir los requisitos operativos descritos al inicio de esta sección.

3.5. Modelo de programas

En esta sección se presenta el diseño de los programas de laminación, encargados de mantener la lista de desbastes programados y gestionar su expedición. La información de los programas se recibe externamente y el almacén se encarga de la expedición de los desbastes. El modelo debe hacer cumplir las siguientes reglas:

- Un material solo puede estar como máximo en un programa activo.
- La expedición de todos los desbastes que se encuentran en el parque, debe completar el programa.
- El orden de prioridad de los programas puede variar.

A continuación, se presentan las clases encargadas de mantener la información de los programas, representadas en la Figura 3.11:

- **ProgramItem:** Representa una secuencia de un programa, vinculando un material con su número de secuencia y grupo. Además, mantiene la información de si la secuencia ha sido expedida o no.
- **SchedulingProgram:** Representa un programa de laminación. Proporciona el comportamiento para ordenar, completar y modificar las secuencias de un programa.

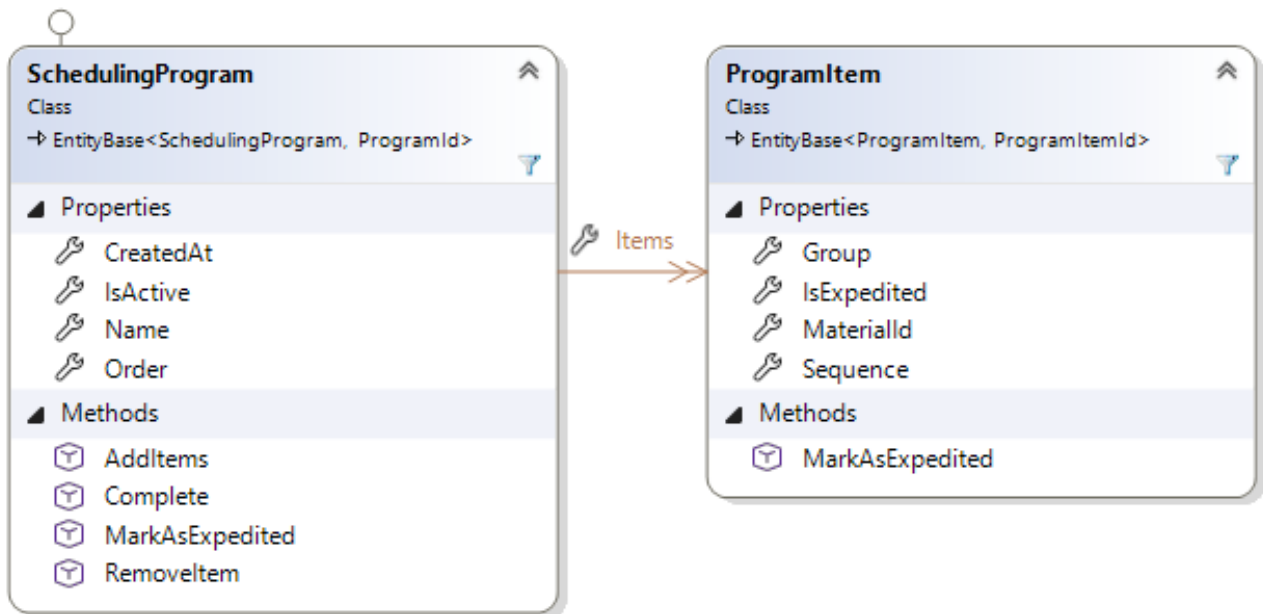


Figura 3.11: Diagrama UML del modelo de programas.

Mediante este modelo es posible recuperar la información relativa a todos los programas de laminación que han existido en el parque. El cumplimiento de los requisitos operativos se reparte entre el modelo y los casos de uso que lo utilizan.

3.6. Modelo del plan de movimientos

En esta sección se presenta el diseño de los planes de movimientos. Constituyen la unidad de trabajo utilizada por los operadores para ejecutar una determinada operación sobre el parque.

Agrupar una secuencia ordenada de movimientos, que pueden ser definidos manualmente por el usuario o creados a petición de los operadores con el apoyo del sistema de soporte a la decisión. El modelo implementa los siguientes tipos de planes en función de su objetivo:

- **Descarga transporte:** Opción diseñada para generar un plan de movimientos que lleva a cabo la descarga completa de un tren o camión. Se sirve del uso de la propuesta para elegir las mejores ubicaciones y genera una secuencia de movimientos seguros que optimizan la descarga.
- **Carga programa:** Opción diseñada para cargar transportes con material de programa. Se sirve de un algoritmo de planificación que ordena las secuencias en el orden requerido por el programa. En caso de necesitar apartar material que no sea de programa, se sirve de la propuesta de ubicación para generar los movimientos.
- **Liberar pila:** Opción diseñada para dejar una pila sin desbastes. Se sirve de la propuesta de ubicación para decidir la nueva ubicación de los materiales.
- **Reubicación:** Opción diseñada para la introducción manual de planes de movimientos, para reorganizar materiales del parque cuando el operador desea intervenir directamente en la planificación. Esta opción permite introducir movimientos de materiales sobre cualquier par de ubicaciones del parque.

A continuación, se presentan las clases encargadas de mantener la información de los planes de movimientos, representadas en la Figura 3.12:

- **StockPosition:** Representa una posición del inventario junto a su altura. Se utiliza para representar el punto del que se debe retirar el material como parte de un movimiento.
- **MaterialStockPosition:** Representa un material junto a su altura. Se utiliza para representar el punto donde se debe ubicar un material concreto.
- **MovementPlanItem:** Representa un movimiento que deberá realizarse como parte del plan de movimientos. Incluye el orden relativo al plan en el que debe realizarse y referencia a los conceptos de «StockPosition» y «MaterialStockPosition» para especificar, respectivamente, el origen y el destino de cada material.
- **MovementPlan:** Representa un plan de movimientos. Actúa como agregado y proporciona el comportamiento necesario para añadir movimientos, cancelar, completar y ordenar el plan.

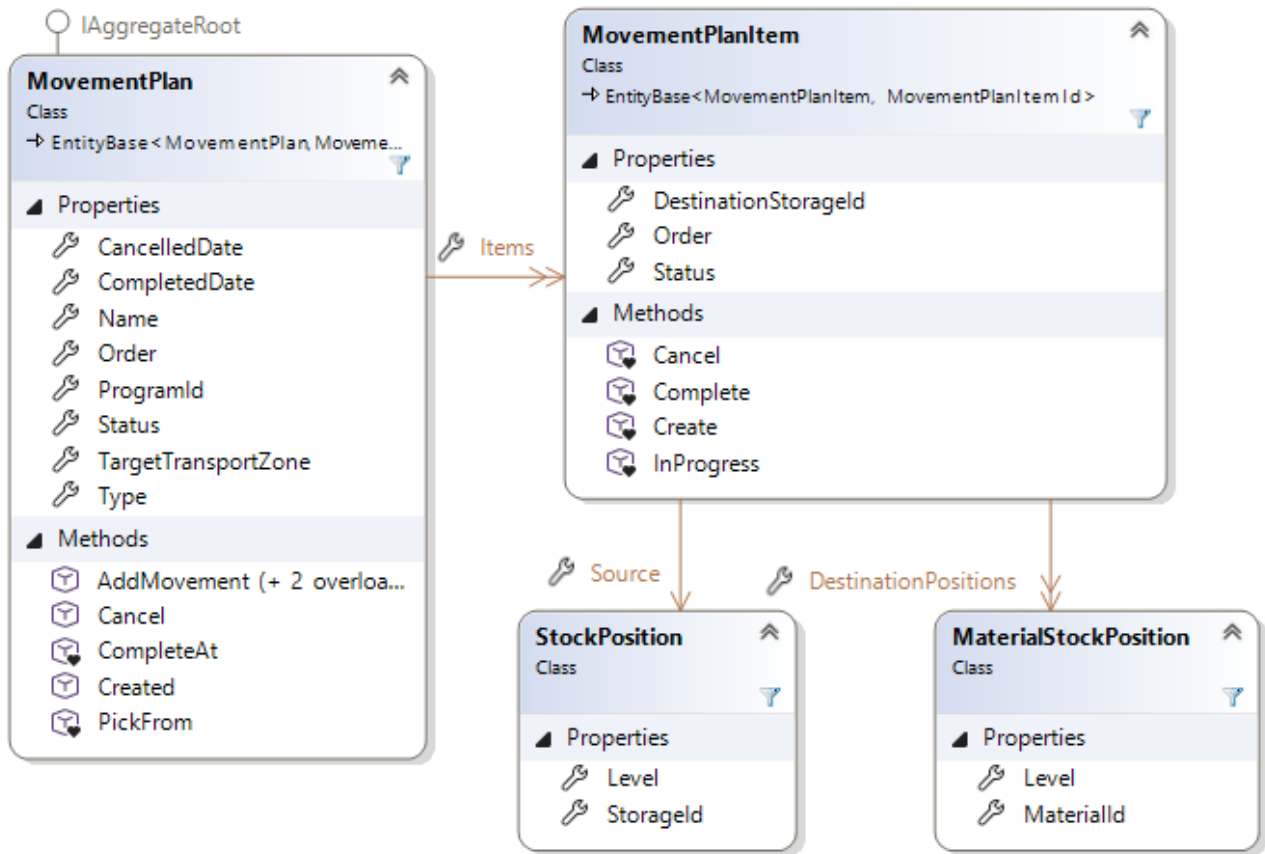


Figura 3.12: Diagrama UML del modelo de planes de movimiento.

Nótese que se distinguen los conceptos de «StockPosition» y «MaterialStockPosition» porque el origen de un movimiento hace referencia a una posición física del almacén, mientras que el destino identifica la ubicación concreta asignada a cada uno de los materiales transportados.

Además de servir como soporte para la ejecución de movimientos, este modelo permite mantener el histórico de todos los planes generados y consultar su estado durante cualquier fase de ejecución. El cumplimiento de los requisitos operativos se reparte entre el propio agregado y los casos de uso encargados de su creación y gestión.

3.7. Modelo relacional

El modelo relacional se ha diseñado siguiendo la separación de responsabilidades establecida en el modelo de dominio. Las distintas entidades representan los conceptos principales del parque, manteniendo relaciones entre ellos y garantizando la integridad de la información mediante el uso de claves externas. Todo el modelo se diseña siguiendo prácticas de normalización, que evitan la existencia de información duplicada.

Debido al tamaño del modelo completo, éste se presenta dividido en varios diagramas agrupados

según su responsabilidad.

La Figura 3.13 muestra el modelo relacional correspondiente a los materiales y a la planificación de programas.

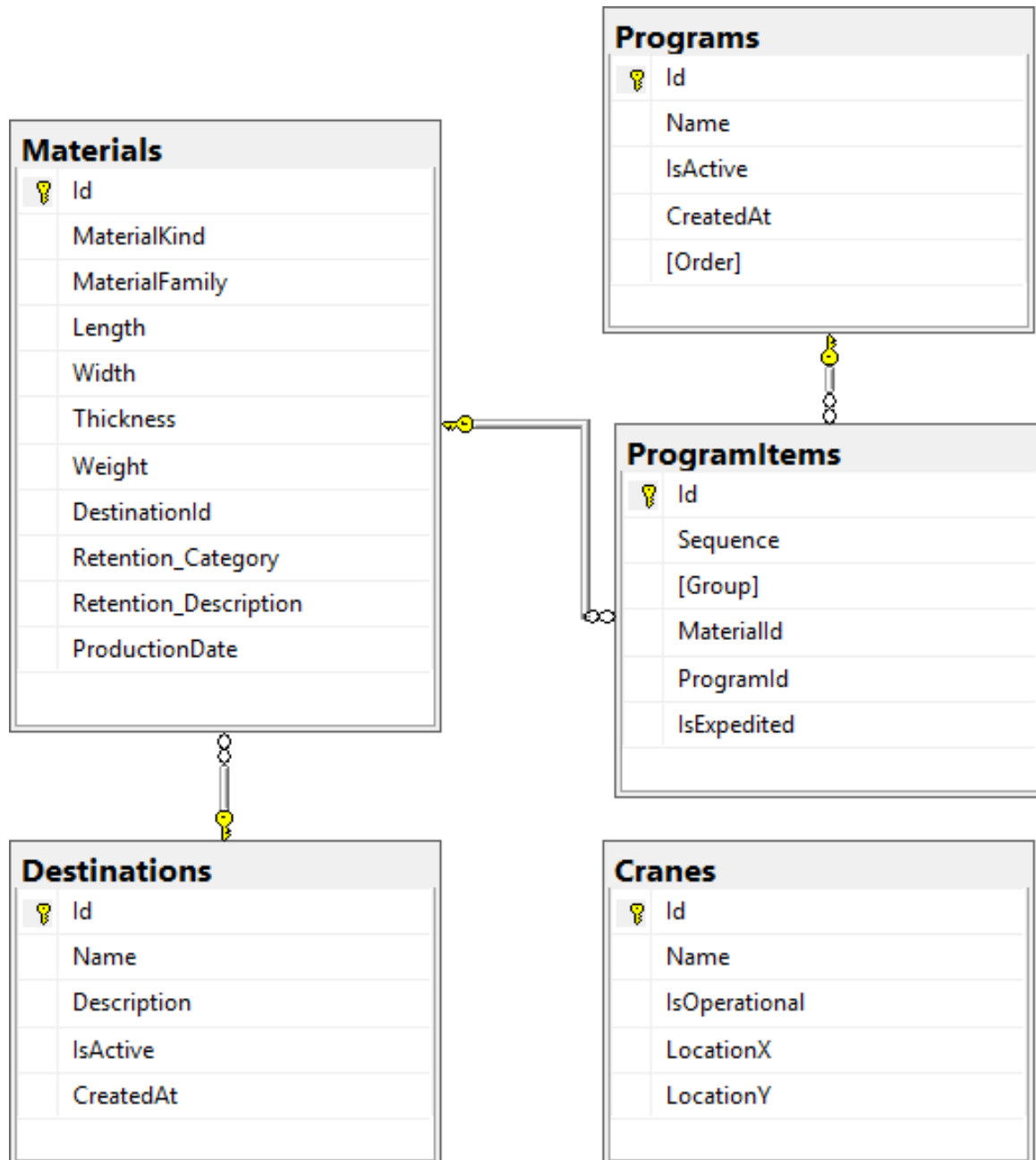


Figura 3.13: Diagrama ER materiales y programas.

La entidad «Materials» almacena las propiedades físicas de cada desbaste, junto con la información necesaria para su clasificación y planificación.

La entidad «Programs» representa los programas de laminación. Cada programa se compone de múltiples elementos almacenados en «ProgramItems», estableciendo una relación uno a muchos entre ambas entidades. La entidad «ProgramItems» representa la pertenencia de un material a

un programa e incluye atributos propios de dicha relación, como la secuencia de laminación, el grupo o la marca de expedición. Estos atributos no pertenecen ni al material ni al programa de forma independiente, sino a la asociación entre ambos. De esta forma, se mantiene separada la información de planificación de la existencia del desbaste, ya que un material puede estar presente en el parque sin estar asociado a un programa de laminación.

La entidad «Destinations» almacena el catálogo de destinos posibles y es referenciada desde los materiales para mantener la integridad referencial.

Finalmente, la entidad «Cranes» almacena la posición y el estado operacional de las grúas del almacén.

La Figura 3.14 muestra el modelo relacional correspondiente al modelo de inventario.

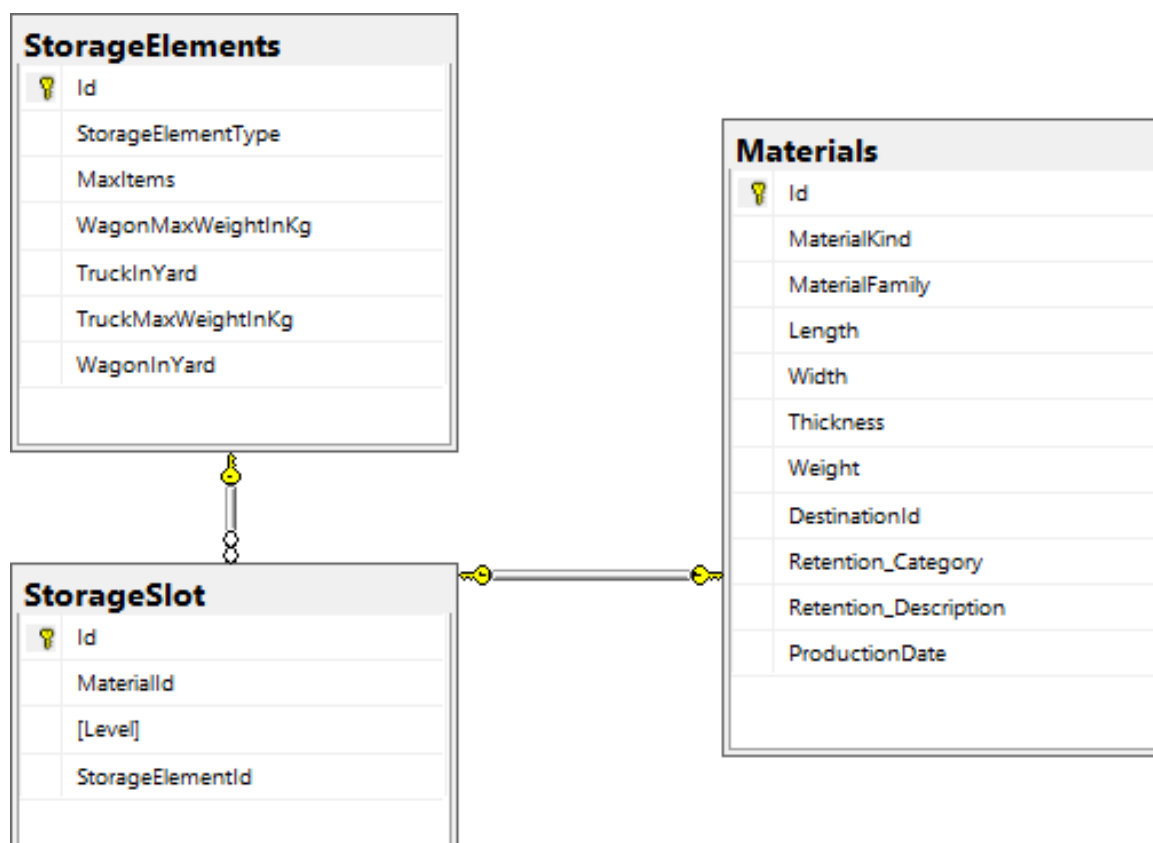


Figura 3.14: Diagrama ER modelo de inventario.

La entidad «StorageElements» representa los elementos de almacenamiento definidos en el fichero de configuración del inventario. De esta forma, se dispone de una representación que permite garantizar la integridad referencial. También incluye un tipo que determina los campos opcionales que son necesarios.

La entidad «StorageSlot» representa la presencia de un material en un elemento de almacenamiento e incluye la altura en la que se encuentra. De esta forma, se mantiene separada la información de un material y un elemento de almacenamiento respecto a la ubicación del prime-

ro en el segundo. Esta entidad también es responsable de garantizar, a través de una restricción de unicidad, que un material no puede encontrarse en más de un elemento de almacenamiento.

La Figura 3.15 muestra el modelo relacional correspondiente a los modelos de movimientos y planes de movimiento.

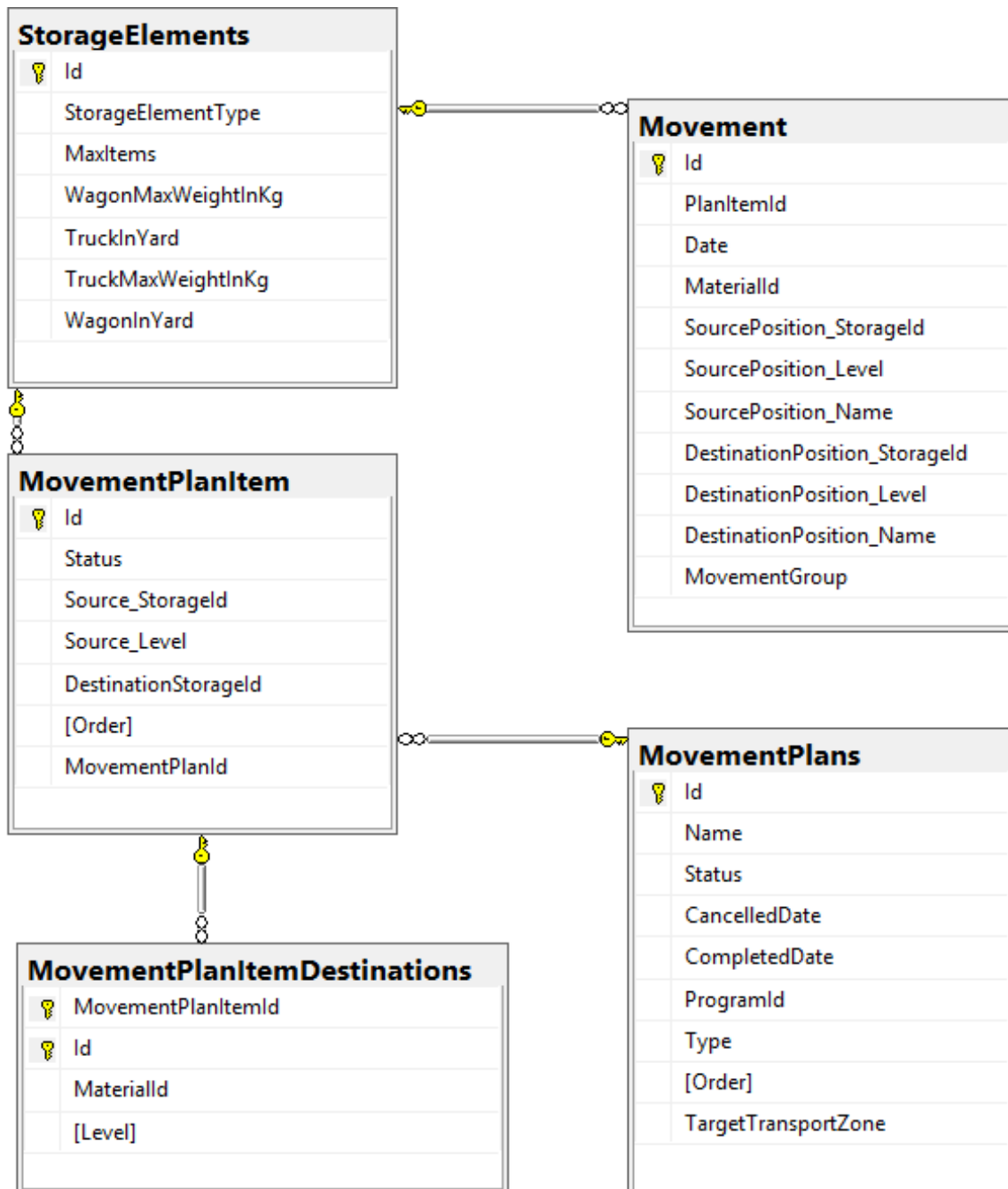


Figura 3.15: Diagrama ER movimientos y programas de movimiento.

La entidad «Movement» representa el histórico de movimientos realizados sobre los elementos de almacenamiento. Incluye información suficiente para identificar las partes involucradas y metadatos informativos, como la fecha o su opcional vinculación a un plan de movimientos. La relación entre un movimiento y un elemento de almacenamiento es doble, ya que debe

garantizarse tanto la integridad del origen como la del destino.

Finalmente, el grupo de entidades «MovementPlans», «MovementPlanItem» y «MovementPlanItemDestinations» representa la información de las planificaciones de movimientos. La finalidad de las mismas es recoger los potenciales movimientos a realizar dentro del plan especificando los orígenes, destinos y alturas en las que se debe colocar cada material, para su cumplimiento. Por simplicidad en el diagrama, se han omitido las relaciones con la entidad «Materials» y «Programs».

3.8. Simulación de gruistas

Con el objetivo de simular el funcionamiento del almacén, se diseña un proceso encargado de emular el comportamiento de los gruistas. La finalidad de este proceso no es, únicamente, reproducir el movimiento de las grúas, sino también permitir la validación y evaluación de los planes de movimientos definidos en el sistema a partir de la propuesta de ubicación.

La simulación utiliza los planes de movimientos generados en la aplicación como fuente de las tareas que deben ejecutar las grúas. El proceso ejecutará continuamente la siguiente secuencia:

1. Seleccionar el siguiente movimiento pendiente de ejecución. Para ello se utiliza la lista de planes de movimientos y se selecciona el primer movimiento no realizado del plan más prioritario. En caso de no existir movimientos, la grúa quedará a la espera de su creación.
2. Mover la grúa al centro de la pila de origen.
3. Recogida de los materiales indicados, que han de estar siempre en el tope de la pila.
4. Mover la grúa al centro de la pila de destino.
5. Depósito de los materiales en la posición indicada.

Con el fin de aproximar el comportamiento del sistema a la realidad, la simulación ha utilizado los siguientes parámetros extraídos de un almacén real:

- Velocidad máxima del carro de la grúa: 90 m/min.
- Velocidad máxima del puente grúa: 120 m/min.
- Tiempo para recogida de materiales de una pila una vez posicionados sobre la misma: entre 15 y 26 s, reducido a entre 10 y 18 s cuando la operación se realiza inmediatamente después de un depósito.
- Tiempo de depósito de materiales: entre 20 y 31 s.

De esta forma, a la hora de probar el sistema podremos evitar tener que realizar los movimientos

manualmente y podremos centrarnos en la evaluación del soporte a la decisión.

3.9. Simulación de programación

Como se ha comentado anteriormente, el almacén de desbastes se encuentra en una posición intermedia entre la línea de colada continua, que produce los desbastes, y la de laminación en caliente, que los transforma en bobinas. En un entorno real, cada una de estas aplicaciones dispondrá de su propio sistema de gestión y contará con mecanismos para compartir información. Como punto de unión entre ambas operativas, debemos recibir información tanto de los materiales que se producen como de los programas de laminación que debemos entregar. Adicionalmente, existen procesos de calidad que revisan potenciales problemas en los materiales. En esos casos el material sospechoso se bloquea con una retención. En esta sección se describen los mecanismos que se han utilizado para simular la recepción de esta información de forma que nuestro almacén sea funcional.

3.9.1. Creación de materiales

La creación de materiales depende, en gran medida, de los pedidos de los clientes y, por lo tanto, es difícil de simular. Para su generación se utilizan distribuciones probabilísticas según la naturaleza de cada atributo. El objetivo es obtener un conjunto de datos sintéticos con características similares a las observadas en un entorno de producción.

Para el tipo de material y la familia, se utiliza una distribución discreta ponderada que elige sobre las categorías configuradas. Para ello, se define una probabilidad asociada a cada categoría, se genera un número aleatorio en el intervalo $[0, 1]$ y se selecciona la primera categoría cuya probabilidad acumulada supera dicho valor.

En las Tablas 3.2 y 3.3 se presentan las probabilidades, configuradas a partir del histórico de producción de una fábrica real durante los últimos veinte años.

No obstante, la distribución de los materiales en el almacén depende en gran medida de los pedidos realizados por los clientes, por lo que puede variar con el tiempo. El establecimiento de una distribución de probabilidades permite generar escenarios más realistas y, a la vez, facilitar la reproducción de los resultados obtenidos.

Para la obtención de los valores presentados a continuación, algunas clasificaciones han sido agrupadas con el objetivo de reducir el número total de tipos utilizados. Las tablas presentadas representan únicamente un caso concreto y no deben tomarse como referencia para otras instalaciones. Los algoritmos y las soluciones propuestas son aplicables a escenarios con otras

distribuciones de probabilidad.

Tabla 3.2: Probabilidades de producción de cada tipo de material.

Tipo	Probabilidad
Ancho	8,04 %
Arranque	5,15 %
Cabecera	17,81 %
Chapa	2,85 %
DWI	3,07 %
FGP	0,33 %
Fino	1,47 %
Hojalata	16,25 %
Microaleado	27,17 %
Refosforado	4,54 %
Ultrabajo	9,67 %
CalidadZ	1,25 %
Resto	2,3 %

Tabla 3.3: Probabilidades de producción de cada familia de material.

Tipo	Probabilidad
Aleados	44 %
Ferrosos	36 %
No Ferrosos	14 %
Otros	6 %

Por el contrario, los atributos numéricos continuos, como la longitud, el ancho, el espesor y el peso, se generan siguiendo una distribución normal (gaussiana). Finalmente, los valores que exceden los límites establecidos se truncan al intervalo válido.

El ancho del material constituye una excepción respecto al resto de atributos numéricos, ya que se trabaja en diferentes rangos según el tipo del material.

- Materiales de tipo Arranque únicamente pueden adoptar uno de varios anchos prefijados (700, 800, 900, 1000, 1100) mm.
- Materiales de tipo Cabecera utilizan una distribución uniforme dentro del intervalo [900, 1300] mm y posteriormente se redondean a múltiplos de 50 mm.

- Materiales del resto de tipos: se genera inicialmente un valor siguiendo la distribución normal descrita y después se redondea al siguiente múltiplo de 10 mm.

Finalmente, la fecha de producción se obtiene mediante una distribución uniforme dentro del último año, asignando a cada material una fecha aleatoria comprendida entre la fecha actual y los 365 días anteriores. De esta forma, simulamos la llegada de materiales que se han fabricado en un periodo más amplio.

3.9.2. Programas

En una situación real, el parque recibiría la información de los programas de un sistema externo. Como eso no ocurre, se hace necesario simularlo internamente. La generación de programas de laminación constituye un problema de optimización complejo. Puede ser abordado mediante la utilización de algoritmos genéticos como se describe en Fardad, Safi-Esfahani y Barekatin [2024](#). Dado que el objetivo de este trabajo no es desarrollar un optimizador de programas de laminación, sino disponer de datos suficientemente representativos para evaluar el sistema, se diseña un algoritmo simplificado basado en la selección aleatoria de desbastes.

La simulación mantiene un máximo de tres programas activos en un momento dado. Cuando el número de programas se encuentra por debajo de este valor, se procede a la creación de un programa. Se comienza el proceso eligiendo el tamaño del programa, que se elige de forma aleatoria en el rango $[80, 120]$ desbastes. La generación continúa si se cumplen las siguientes condiciones:

- Existe un tipo de material que tenga por lo menos el 90 % del tamaño del programa.
- Existen suficientes desbastes de tipo arranque para representar el 5 % del tamaño del programa.
- Existen suficientes desbastes de tipo cabecera para representar el 5 % del tamaño del programa.

Los porcentajes utilizados se han seleccionado de forma empírica para obtener programas de laminación con una morfología similar a la observada en parques reales, sin pretender reproducir exactamente el proceso de planificación.

En caso de cumplir las condiciones, elegiremos el tipo con un mayor número de desbastes y comenzaremos el proceso de la generación de la forma del programa. La Figura 3.16 ilustra la terminología y los tipos utilizados en cada parte del programa.

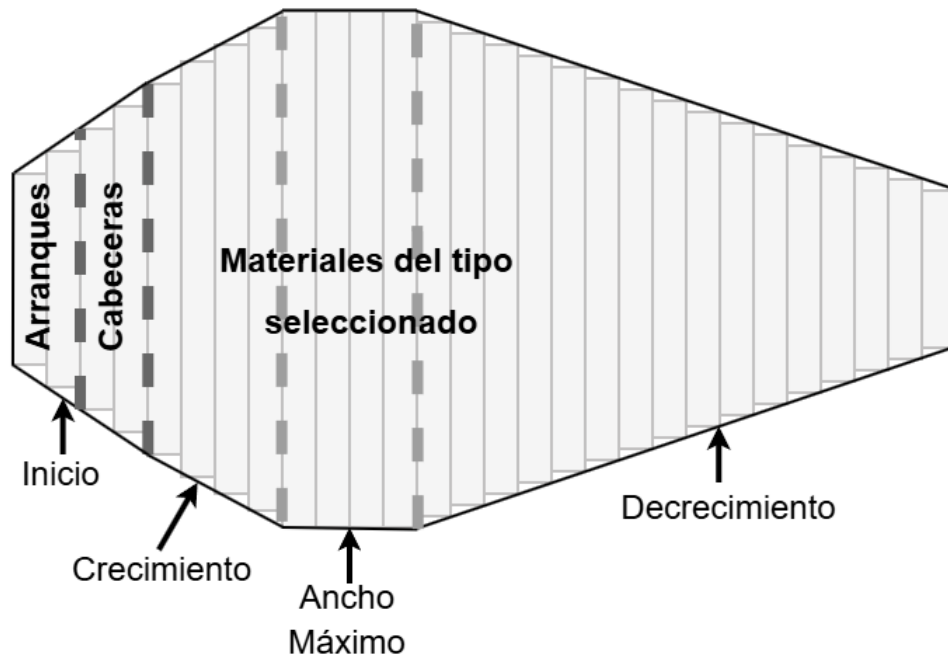


Figura 3.16: Composición y uso de tipos en un programa.

Para la generación se siguen los siguiente pasos:

1. **Preparación:** Se elige de forma aleatoria el punto de ancho máximo entre el 45 % y el 65 % del tamaño del programa. Sobre los materiales del tipo seleccionado se calcula el ancho mínimo y el ancho medio.
2. **Crecimiento:** Para seleccionar el primer desbaste, se selecciona un valor aleatorio entre el mínimo y la media. Sobre este, se eligen sucesivamente desbastes cuyo ancho sea entre 0 y 25 mm superior al anterior. La elección se repite hasta llegar al punto máximo, salvo que nos quedemos sin materiales candidatos. En ese caso, se continuará intentando llegar al punto máximo, insertando materiales entre los desbastes que ya han sido seleccionados.
3. **Decrecimiento:** Partiendo del ancho máximo se realiza una selección similar a la anterior salvo que en este caso se utiliza un ancho decreciente y se repite hasta alcanzar el número de desbastes de tipo necesarios para el programa.
4. **Cabeceras:** Partiendo del ancho mínimo calculado al inicio, se seleccionan cabeceras sucesivamente cuyo ancho sea entre 0 y 100 mm inferior al anterior. La elección se repite hasta llegar al 5 % de cabeceras, salvo que nos quedemos sin candidatos. En ese caso, se continuará intentando llegar al número necesario, insertando materiales entre los desbastes que ya han sido seleccionados.
5. **Arranques:** Partiendo del ancho mínimo de las cabeceras, se seleccionan arranques suce-

sivamente cuyo ancho sea entre 0 y 100 mm inferior al anterior. La elección se repite hasta llegar al 5 % de arranques, salvo que nos quedemos sin candidatos. En ese caso, se continuará intentando llegar al número necesario, insertando materiales entre los desbastes que ya han sido seleccionados.

Como puede observarse, el algoritmo no persigue obtener una planificación óptima, sino generar una secuencia de materiales que respete determinadas restricciones dimensionales y reproduzca la forma característica de un programa de laminación.

Esta aproximación presenta limitaciones, ya que no considera aspectos como los posibles aprovechamientos de material almacenado en ubicaciones próximas, la planificación global de la producción ni la información procedente de otros parques. No obstante, en el contexto de este trabajo, la selección aleatoria puede aproximar un escenario desfavorable, generando programas dispersos en múltiples pilas del almacén. Este comportamiento resulta útil para evaluar la eficacia de la propuesta de ubicación en situaciones desfavorables.

3.9.3. Retenciones

Para la simulación de las retenciones se propone un proceso que se ejecute cada hora y que lleve a cabo las siguientes acciones:

- Sobre los materiales con una retención, se retirará con una probabilidad del 20 %.
- Sobre los materiales sin retención ni programa, se asignarán retenciones aleatoriamente hasta alcanzar un 5 % de todos los materiales conocidos (no solo los del parque).

Los materiales con programa no pueden tener retenciones ya que al estar listos para su laminación, no pueden presentar problemas de calidad. La categoría de la retención se asignará aleatoriamente usando un generador de números aleatorios.

3.10. Especificaciones de la aplicación

A partir de las funcionalidades descritas a lo largo de este capítulo, se establecen las siguientes especificaciones para GenStock. Para facilitar la trazabilidad durante la verificación de las mismas, cada especificación se identifica mediante un código compuesto por una abreviatura del área funcional a la que pertenece y un número incremental.

3.10.1. Especificaciones funcionales

La Tablas 3.4 y 3.5 recogen las especificaciones funcionales relacionadas con las operaciones que pueden realizarse desde la aplicación.

Tabla 3.4: Especificaciones funcionales de la aplicación (1 de 2).

Código	Sección	Funcionalidad
EST-01	Estadísticas	Visualización de estadísticas del almacén.
TRN-01	Transportes	Listado de transportes.
TRN-02	Transportes	Entrada de un transporte.
TRN-03	Transportes	Salida de un transporte.
TRN-04	Transportes	Consulta del detalle de un transporte.
INV-01	Inventario	Listado de materiales en el almacén.
INV-02	Inventario	Búsqueda y filtrado de materiales.
INV-03	Inventario	Consulta del detalle de un material.
PMO-01	Planes de movimiento	Listado de planes de movimiento.
PMO-02	Planes de movimiento	Alta de un plan de movimiento para descarga de transporte.
PMO-03	Planes de movimiento	Alta de un plan de movimiento para liberación de pila.
PMO-04	Planes de movimiento	Alta de un plan de movimiento para carga de programa.
PMO-05	Planes de movimiento	Alta de un plan de movimiento para reubicación.
PMO-06	Planes de movimiento	Consulta del detalle de un plan.
PMO-07	Planes de movimiento	Cancelación de un plan.
PMO-08	Planes de movimiento	Borrado de un plan.
PMO-09	Planes de movimiento	Cambio de orden de un plan.
PRG-01	Programas	Listado de programas.
PRG-02	Programas	Consulta del detalle de un programa.

Tabla 3.5: Especificaciones funcionales de la aplicación (2 de 2).

Código	Sección	Funcionalidad
MAP-01	Mapa	Visualización del estado del almacén.
MAP-02	Mapa	Búsqueda de materiales.
MAP-03	Mapa	Consulta de la información de una pila.
MAP-04	Mapa	Visualización de movimientos.
MOV-01	Movimientos	Movimiento de material entre pilas.
DES-01	Gestión de destinos	Listado de destinos.
DES-02	Gestión de destinos	Alta de un destino.
DES-03	Gestión de destinos	Modificación de un destino.
DES-04	Gestión de destinos	Baja de un destino.
USU-01	Gestión de usuarios	Listado de usuarios.
USU-02	Gestión de usuarios	Alta de usuario.
USU-03	Gestión de usuarios	Modificación de usuario.
USU-04	Gestión de usuarios	Baja de usuario.
GRU-01	Gestión de grúas	Listado de grúas.
GRU-02	Gestión de grúas	Modificación de una grúa.

3.10.2. Especificaciones de los servicios de simulación

Además de las funcionalidades accesibles desde la interfaz de usuario, la aplicación incorpora servicios destinados a simular procesos externos que ejecutan operaciones sobre el almacén. Estas funcionalidades permiten disponer de un entorno con la información necesaria para la verificación de la aplicación. En la Tabla 3.6 se recogen las especificaciones de las funcionalidades de simulación.

Tabla 3.6: Especificaciones de los servicios de simulación.

Código	Servicio	Funcionalidad
SIM-01	Simulación de grúas	Ejecución automática de los planes de movimientos pendientes mediante la simulación de la grúa.
SIM-02	Generación de materiales	Generación periódica de nuevos materiales según la configuración establecida.
SIM-03	Generación de programas	Generación automática de programas de producción.
SIM-04	Gestión de retenciones	Generación y eliminación automática de retenciones sobre los materiales del almacén.

3.11. Conclusiones

En este capítulo se ha presentado el diseño de la solución propuesta para la aplicación GenStock. El objetivo de la aplicación es la gestión de un almacén de desbastes y el apoyo a su gestión mediante la utilización de sistemas de soporte a la decisión. Se ha definido la arquitectura software de la aplicación, así como los principales modelos del dominio que soportan sus funcionalidades.

La arquitectura planteada persigue separar claramente las responsabilidades de cada componente, aislando la lógica de negocio de los aspectos tecnológicos y facilitando la evolución independiente de las distintas partes del sistema. La aplicación de los principios de arquitectura limpia y del diseño guiado por el dominio ha permitido estructurar el problema en contextos bien definidos, reduciendo el acoplamiento y favoreciendo la mantenibilidad de la solución.

Sobre esta arquitectura se han diseñado los modelos que representan los conceptos fundamentales del dominio, como el inventario, los movimientos, los programas de laminación y los planes de movimientos, definiendo las reglas de negocio asociadas a cada uno de ellos. También se ha definido el modelo relacional y los servicios de simulación necesarios para disponer de un entorno que permita realizar las pruebas de la aplicación.

Finalmente, se han definido las especificaciones que debe cumplir GenStock, estableciendo una referencia para la posterior verificación de sus funcionalidades.

Este diseño de la aplicación proporciona la base para el siguiente capítulo, donde se plantea el diseño de la propuesta de ubicación. En donde se definirá el problema, las reglas de evaluación y el algoritmo de búsqueda empleados para generar recomendaciones de almacenamiento. La propuesta de ubicación constituye el principal elemento sobre el que se estructurarán las funcionalidades de soporte a la decisión.

Capítulo 4

Diseño propuesta de ubicación

4.1. Introducción

En este capítulo se describe el diseño de la funcionalidad responsable de elegir las mejores ubicaciones para un grupo de desbastes. Se realizará una revisión, comenzando con una descripción breve de la implementación del algoritmo A^* para pasar a describir el procedimiento utilizado en la generación de nodos vecinos, continuando con los requisitos que ha de cumplir el heurístico y finalmente se revisa el diseño de las reglas que dirigen el algoritmo, junto con sus distintos niveles de aplicación.

Es importante destacar que la propuesta de ubicación únicamente está encargada de buscar las mejores ubicaciones para un estado concreto del parque y por lo tanto, no se encarga de sugerir reorganizaciones ni de mover materiales que no estén incluidos en la petición de ubicación.

4.2. Algoritmo A^*

El algoritmo A^* es un algoritmo empleado tradicionalmente en la búsqueda de caminos y brilla, principalmente, en los casos en los que se puede guiar su ejecución, mediante el uso de un algoritmo heurístico que permita elegir los nodos más prometedores en la búsqueda.

Como parte de la búsqueda, se necesita mantener un estado capaz de gestionar, de forma eficiente, la información relativa al estado de las ubicaciones. Contendrá al menos la siguiente información:

- **Pilas:** Mantiene información relativa al estado de las pilas, que difiere del estado inicial del parque, ya que, en cada rama, el estado realiza una secuencia de movimientos diferentes sobre los elementos de almacenamiento. Por lo tanto, las pilas tendrán una composición diferente en cada estado del árbol de evaluación.
- **Coste:** Coste total de la solución que representa este estado. Es importante destacar que es un coste parcial, mientras que aún queden materiales por colocar y final, cuando ya no queden más elementos.
- **Orígenes pendientes:** Almacena información sobre las pilas donde se encuentran los

materiales para los que se han pedido propuesta y los materiales que quedan pendientes de extraer de las mismas.

- **Movimientos:** Representa la secuencia de movimientos que se han realizado para llegar al estado actual. Es la información más importante que se extraerá de la ejecución del algoritmo.

El cálculo de los costes de cada estado se realiza durante la exploración de los nodos, mediante la aplicación de las reglas que se describirán en los apartados 4.7 y 4.8. Las reglas se basan en la evaluación de criterios asociados a los desbastes, en función de sus propiedades físicas y de programación. Los criterios para realizar el agrupamiento son mutuamente exclusivos y se evalúan en orden, quedando el desbaste asignado al criterio más prioritario en el siguiente orden:

1. **Programa:** La asignación a programa representa la necesidad de salir del parque en un orden concreto en un horizonte de tiempo cercano, por este motivo, es la más prioritaria.
2. **Destino:** Los desbastes con destino saldrán del parque siempre que quede capacidad libre en la salida de programas, lo que los hace menos prioritarios.
3. **Tipo:** El desbaste cuenta con un pedido y en algún punto se incluirá en un programa de laminación con otros de su mismo tipo.
4. **Familia:** El desbaste no tiene aún pedido y según su familia se le pueden asignar diferentes tipos.

Dentro de cada criterio, siempre se considerarán las propiedades dimensionales basadas en el largo y el ancho de los desbastes.

En la Figura 4.1 se representan los términos comunes que se emplearán durante la descripción del funcionamiento de la propuesta de ubicación.

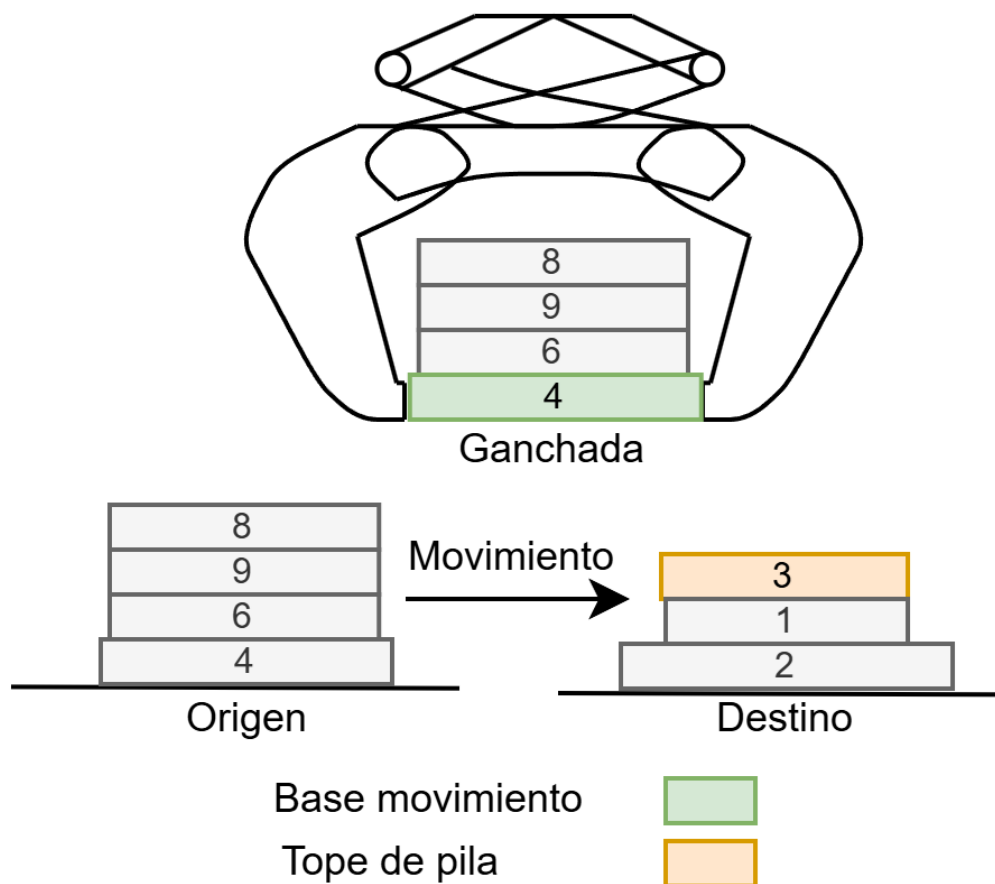


Figura 4.1: Terminología propuesta ubicación.

A continuación, se comienza la explicación del algoritmo describiendo cómo se generan los nodos sucesivos de un estado.

4.3. Expansión de un nodo

En esta sección se expone la forma en la que el algoritmo expande el espacio de búsqueda de una solución, añadiendo movimientos pendientes.

Como parte de la ejecución del A* debemos de ser capaces de generar los estados derivados de un estado dado, para continuar la exploración del espacio de soluciones. Se implementa una solución que, sobre el estado descrito en la sección anterior, busca generar todos los posibles movimientos para retirar los elementos pendientes de los orígenes. En la Figura 4.2 se muestra el árbol de expansión parcial del nodo inicial de una propuesta de ubicación para un único desbaste.

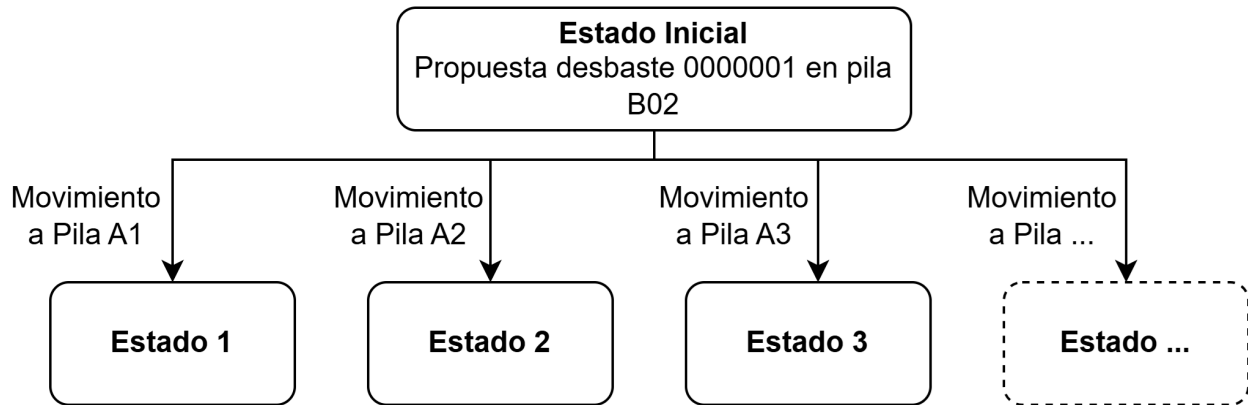


Figura 4.2: Diagrama expansión estados A*.

El algoritmo propuesto comienza revisando cada pila que contiene materiales para los que se ha pedido propuesta. Nos referiremos a esta agrupación de desbastes en pilas como orígenes. Otra consideración fundamental es que la propuesta sobre una pila no incluye necesariamente toda la pila, sino que funciona a nivel de desbastes individuales, por lo que siempre considerará un número X contando desde el tope hacia la base. De esta forma, la propuesta puede ignorar los materiales que quedan debajo y simular que estamos vaciando las pilas de origen. De lo anterior, se deduce que dichas pilas no pueden utilizarse en la solución ya que asumimos que un material para el que se pide propuesta debe de ser movido. Los orígenes pueden tener más elementos que los que la grúa puede mover en un movimiento, por lo que nunca se extraerán en una expansión más elementos de los que puede retirar la grúa. Esta decisión se justifica en que esos movimientos no son descendientes directos y, por lo tanto, no pueden ser realizados sin haber realizado un movimiento para acceder a ellos.

Por cada origen, seleccionaremos todas las ganchadas posibles, entre uno y cuatro desbastes, y únicamente si son seguras procederemos a generar las opciones de movimiento de cada una a:

- **La pila vacía más cercana a la pila de origen:** Considerar la pila vacía más cercana es suficiente ya que todas las pilas vacías son «iguales» y hacer el filtrado en este punto reduce la ramificación de la exploración.
- **El resto de pilas con materiales.**

Es importante destacar que el factor de ramificación de este algoritmo es elevado. Por ejemplo, en el caso de 60 pilas, de las cuales 12 están vacías, y 4 desbastes en un origen, la ramificación del estado inicial resulta en $(60 - (12 - 1)) * 4 = 196$ estados candidatos directos.

En el siguiente apartado se revisará la lógica heurística que se utiliza para priorizar la exploración de aquellos estados que resultan más prometedores, de forma que nos ayude a navegar el espacio de búsqueda.

4.4. Heurístico

La función de la heurística es obtener una estimación del coste de ir desde un estado hasta la meta. Esta estimación dirigirá la ejecución del algoritmo, de forma que serán evaluadas primero las opciones más prometedoras. Dado que la propuesta utiliza el algoritmo A^* de búsqueda heurística, resulta deseable que la función tenga las siguientes propiedades:

- **Admisibilidad:** El coste que se obtiene del algoritmo nunca será mayor que el coste real de alcanzar la meta. Cumpliendo esta propiedad se garantiza que el algoritmo encuentre una solución óptima, determinada por la función de coste del problema.
- **Monotonía:** Favorece que para un nodo y uno de sus hijos, el coste de la meta no sea superior a ir al hijo y luego a la meta. Esta propiedad evita inconsistencias durante la exploración y hace que el coste estimado evolucione de forma coherente entre estados consecutivos.

El desarrollo del heurístico es complejo, requiriendo de la realización de pruebas y ajustes experimentales. Por lo tanto, su definición y análisis de decisiones adoptadas se realizará dentro del Capítulo 5.

4.5. Tipos de reglas

Para la toma de decisiones durante la ejecución del algoritmo diferenciamos tres tipos de reglas, según su ámbito de aplicación:

- **Reglas de validación:** Evalúan si la realización de un movimiento a una pila es válido o debe descartarse.
- **Reglas globales:** Realizan un cálculo de coste en función del estado del parque, sin limitarse a la pila de destino.
- **Reglas locales:** Realizan un cálculo de coste en función de la pila y el movimiento que se propone.

El resultado de la evaluación de las reglas locales y globales se suma para componer el coste de un nodo dentro del A^* , mientras que, en aquellos estados que no pasen la validación, su coste no será evaluado. En los siguientes apartados se describen las reglas definidas para cada tipo.

4.6. Reglas de validación

Este tipo de reglas tienen un resultado binario que se utilizará para validar la posibilidad del movimiento a la pila y determinar si puede ejecutarse o no. En caso de que alguna de ellas determine que el movimiento no es viable, este será descartado y el algoritmo detendrá la exploración de esa rama.

4.6.1. Máximo elementos

Cuando realizamos el almacenamiento de desbastes en pilas es habitual establecer un límite operativo a la altura máxima que puede alcanzarse con el apilamiento, con el fin de garantizar la seguridad y optimizar los niveles de almacenamiento. Para la gestión de la optimización, se establece un parámetro que limita el máximo de elementos en función del volumen de material que se espera alojar en el parque. Para tomar una decisión sobre este valor, se tendrá en cuenta la altura máxima que puede alcanzar una pila de desbastes en relación a la distancia mínima del suelo de la pinza, cuando lleva el máximo de desbastes. Además, es habitual añadir un margen adicional de seguridad, que no requiera de alcanzar la elevación máxima, para garantizar la seguridad y que dé un margen suficiente para evitar colisiones durante el izado del inicio de los movimientos, en los que pueden existir puntos ciegos. Finalmente, se ha elegido configurar esta regla con un número de desbastes en lugar de con una altura física, debido a que las diferencias dimensionales de espesor no son tan grandes como para justificar la complejidad para los usuarios de contar con números de alturas variables, en función de los elementos almacenados. La implementación de la regla comprobará que al realizar un movimiento a una pila, ésta no excede el número máximo de elementos que ha sido configurado.

4.6.2. Seguridad de pila

Un factor crítico, que está presente en toda la operativa del parque, es el concepto de la seguridad de pila, por la que deben evitarse situaciones en las que el apilamiento cause riesgos. Durante la operación manual, es posible que un operador cometa un error y genere una pila que no sea segura, sin embargo, la propuesta debe evitar generar estas situaciones. Para evitarlo, se introduce la regla de seguridad, encargada de hacer cumplir las reglas dimensionales definidas en 2.4.2 y evitar la exploración de esos estados.

4.6.3. Movimientos permitidos

El objetivo de esta regla es evitar configuraciones de apilamiento que dificulten la recuperación posterior de los materiales. En particular, impide que un material, de un criterio de menor prioridad, quede situado sobre otro de un criterio de mayor prioridad, ya que ello obligaría a realizar movimientos adicionales para acceder al material inferior. Para ello, se establece la siguiente jerarquía de prioridades:

- **Programa:** constituye el criterio de mayor prioridad. Un material clasificado por programa puede situarse sobre materiales de cualquier criterio.
- **Destino:** puede colocarse sobre materiales clasificados por destino, tipo o familia, pero no sobre materiales clasificados por programa.
- **Tipo:** únicamente puede colocarse sobre materiales clasificados por tipo o familia.
- **Familia:** constituye el criterio de menor prioridad y solo puede colocarse sobre materiales clasificados por familia.

Siguiendo la anterior jerarquía evitamos generar opciones claramente perjudiciales, pero aún tenemos que tratar la mezcla de criterios en las pilas. Sin embargo, ese aspecto se tratará según el criterio definido en Subapartado 4.8 que ya no implementa reglas binarias sino una asignación de coste a cada caso.

4.7. Reglas globales

Este tipo de reglas realizan un cálculo, teniendo en cuenta el estado del parque y el movimiento, para obtener un coste parcial, que luego se sumará a la evaluación de las reglas locales. Su elemento diferencial es la capacidad para considerar el escenario completo en lugar de centrarse en la pila destino del movimiento que se está evaluando. Estas reglas se ejecutan siempre que el estado sea válido.

4.7.1. Pilas vacías

La gestión de pilas vacías es un aspecto clave en el parque, ya que el espacio de almacenamiento es finito y una asignación inadecuada puede generar casos en los que no se aprovecha toda la altura de almacenamiento o se genera la necesidad de realizar movimientos de reordenación.

Los aspectos más importantes para tomar la decisión de asignación son:

- **El ancho de los desbastes a colocar:** De acuerdo con las restricciones establecidas

en 2.4.2, cuando colocamos un desbaste de ancho X mm la pila solo podrá aceptar desbastes cuyo ancho se encuentre entre el mínimo ancho de desbaste y $X + 320$ mm (piramidal siempre es seguro). En consecuencia, asignar una pila vacía a un material estrecho reduce el conjunto de desbastes que podrán almacenarse posteriormente en esa pila. Por el contrario, si abrimos pilas para los materiales muy anchos se favorecerá el aprovechamiento dimensional, pero también aumenta la probabilidad de que estos queden cubiertos por materiales más estrechos, incrementando las reubicaciones necesarias para su extracción.

- **El número de pilas vacías disponibles:** Como recurso finito debe gestionarse adecuadamente para no agotarlas. Cuando esto ocurre, la propuesta se ve obligada a mezclar material que idealmente se almacenaría separado, degradando la calidad de la propuesta.

La solución propuesta considera ambos criterios de forma independiente y los modela mediante una función de penalización normalizada, cuyo resultado se suma al coste total de la regla. En ambos casos se ha decidido emplear una función lineal por su sencillez, facilidad de parametrización y porque permite aumentar la penalización progresivamente.

4.7.1.1. Ancho de desbastes

Para el caso del ancho de los desbastes se emplea la fórmula

$$\left(1 - \frac{A_d - B}{C - B}\right) \cdot A \quad (4.1)$$

donde:

- A_d : Representa el ancho del desbaste a ubicar en la base de la pila.
- A : Parámetro configurable de la regla «WidthDiffMaxPenalty» que representa la máxima penalización que se puede obtener por este criterio.
- B : Es una constante con valor 700 mm que representa el mínimo ancho que puede tener un desbaste.
- C : Es una constante que representa el máximo ancho de un desbaste. Aquí diferenciamos según el tipo, Arranques (1100 mm), Cabeceras (1300 mm) y cualquier otro (2500 mm).

Nótese que este criterio únicamente favorece la apertura de pilas para desbastes de mayor anchura. Se delega la responsabilidad del control de la piramidalidad a la regla 4.8.6.

4.7.1.2. Número de pilas disponibles

En cuanto al número de pilas disponibles, la penalización se calcula considerando el estado del parque tras realizar la asignación. Por este motivo, el número de pilas libres se reduce en una unidad al evaluar el coste de abrir una nueva pila. La penalización se obtiene mediante la expresión

$$\left(1 - \frac{A-1}{B}\right) \cdot Z \quad (4.2)$$

donde:

- A : Representa el número total de pilas libres antes de realizar la asignación.
- B : Representa el número de pilas de las que dispone el almacén.
- Z : Parámetro configurable de la regla «EmptyPileUsageMaxPenalty» que representa la máxima penalización que se puede obtener por este criterio.

De esta forma, a medida que el número de pilas ocupadas aumenta, también lo hace el coste de abrir pilas, lo que permite balancear el uso del espacio y el mantenimiento de pilas libres.

Como se ha comentado en la introducción de la sección, es responsabilidad de la operación del parque, y no de la propuesta de ubicación, realizar movimientos para agrupar materiales, especialmente después de realizar extracciones.

4.7.2. Distancia

La distancia recorrida en los movimientos de las grúas es un factor directamente vinculado al tiempo de ejecución de un movimiento, lo que limita cuántas peticiones puede atender el parque y, por lo tanto, su eficiencia operativa. La propuesta debe considerar este factor para tomar decisiones de forma que, a costes de solución similares, se elija la opción que minimice la distancia a recorrer.

Para el diseño de la regla se ha optado por evaluar de forma independiente las componentes X e Y de la distancia. Esta decisión se debe a que, en una grúa puente, el movimiento del carro suele ser más rápido y menos costoso que el desplazamiento del puente, por lo que ambas direcciones tienen un impacto diferente sobre el tiempo total del movimiento. Para obtener una fórmula flexible y fácil de parametrizar, ambas componentes se modelan mediante funciones de penalización normalizadas. Se ha decidido usar funciones de penalización lineal por su sencillez. La Fórmula empleada es 4.3:

$$\left(\frac{|O_x - P_x|}{A} \cdot C\right) + \left(\frac{|O_y - P_y|}{B} \cdot D\right) \quad (4.3)$$

donde:

- O : Representa el punto de origen del movimiento con sus componentes X e Y .
- P : Representa el punto de la pila que está siendo evaluada con sus componentes X e Y .
- A : Representa la anchura total del almacén.
- B : Representa el alto total del almacén.
- C : Parámetro configurable de la regla «MaxPenaltyX» que representa la máxima penalización que puede obtener un movimiento sobre el eje X .
- D : Parámetro configurable de la regla «MaxPenaltyY» que representa la máxima penalización que puede obtener un movimiento sobre el eje Y .

La normalización mediante las dimensiones del almacén permite que el valor obtenido sea independiente del tamaño del parque y que la influencia de la regla pueda ajustarse únicamente mediante los parámetros C y D .

De esta forma, se obtiene una evaluación consistente de la distancia recorrida, que ayuda a priorizar aquellas ubicaciones que requieren un menor desplazamiento de las grúas, contribuyendo a mejorar la eficiencia operativa del parque.

4.7.3. Tamaño de ganchada

Junto con la distancia recorrida, el número de materiales que se transportan en la grúa en un movimiento es un factor vinculado a la optimización de los movimientos. La propuesta debe considerar este factor para tomar decisiones que, a costes de solución similares, se elija la opción que maximice el número de desbastes del movimiento.

Para el diseño de esta regla se opta por establecer unas puntuaciones estáticas según el número de materiales que van a ser incluidos en la ganchada. En la Tabla 4.1 se observan las penalizaciones establecidas.

Tabla 4.1: Penalización según tamaño ganchada.

Número Desbastes	Penalización
4	0
3	15
2	30
1	40

De esta forma, a igualdad del resto de criterios, la propuesta tenderá a priorizar aquellos mo-

vimientos que transporten un mayor número de desbastes. Las diferencias entre penalizaciones son pequeñas para que esta regla actúe como un criterio de desempate y no prevalezca sobre otras restricciones de mayor importancia.

4.8. Reglas locales

Este tipo de reglas realizan un cálculo teniendo en cuenta el estado de la pila y los materiales que se propone colocar en la misma, para obtener el coste parcial que luego se sumará a la evaluación de las reglas globales. Su elemento diferencial es centrarse en la pila destino y los nuevos materiales, sin considerar el contexto del almacén. Estas reglas se ejecutan siempre que el estado sea válido y que se cumplan los requisitos establecidos por cada regla.

4.8.1. Programas

El objetivo de esta regla es favorecer una organización de las pilas que facilite la expedición de los programas, reduciendo el número de movimientos necesarios para ordenar y cargar los desbastes. Para que esta regla sea aplicable el movimiento ha de contener material de programa.

Cuando es de aplicación, se evalúan distintos aspectos de la pila de forma independiente, asignando una penalización a cada uno de ellos. La penalización total se obtiene como la suma de todas las penalizaciones parciales:

$$P_{total} = P_{cobertura} + P_{mezcla} + P_{cohesion} + P_{orden} + P_{enterramiento} + P_{bloqueo} + P_{programas} \quad (4.4)$$

Cada uno de estos términos representa un criterio diferente de evaluación. Su justificación se desarrolla en los siguientes apartados.

4.8.1.1. Cobertura de programas prioritarios

Uno de los aspectos más importantes consiste en evitar que materiales pertenecientes a programas de mayor prioridad queden cubiertos por otros de menor prioridad. Cuando esta situación ocurre, la penalización aumenta con la profundidad a la que queda enterrado el programa prioritario, ya que su recuperación requerirá un mayor número de movimientos. Es fácil interpretar que cubrir un programa con otro de mayor prioridad, equivale a generar una necesidad de reubicación cuando llegue el momento de expedir u ordenar el programa prioritario.

Se ha decidido usar una fórmula de penalización lineal por su fácil configuración y porque representa un crecimiento predecible de la penalización. La fórmula empleada es:

$$P_{cobertura} = \begin{cases} N_C \cdot A & \text{si existe cobertura de un programa prioritario,} \\ 0 & \text{en caso contrario.} \end{cases} \quad (4.5)$$

donde:

- N_C : Profundidad máxima a la que queda cubierto el programa prioritario, calculada como número de desbastes que quedan por encima del prioritario.
- A : Parámetro configurable «BurriedMorePriorityProgramsPenalty» que representa la penalización por cada nivel de profundidad. Debido a su importancia a la hora de separar grupos de soluciones, se recomienda asignar a este parámetro un valor alto en relación al resto, por ejemplo, 4000.

4.8.1.2. Mezcla de programas

Siempre que sea posible, se intenta evitar almacenar materiales pertenecientes a distintos programas dentro de una misma pila. Cuantos más programas diferentes aparecen en la pila, el factor de desorden se incrementa, por tanto, mayor será la penalización obtenida. Un factor diferencial en esta comprobación es que, aunque mezclamos programas, coincidan en el tope de la pila y la base del movimiento, ya que en este caso tendremos al menos dos elementos del mismo programa juntos. Este hecho se trata como una circunstancia ligeramente más positiva que la no coincidencia.

Se ha decido usar fórmulas de penalización lineal ya que representan un crecimiento apto para diferenciar el incremento de coste de mezcla entre soluciones. La fórmula utilizada es:

$$P_{mezcla} = \begin{cases} 2 \cdot A \cdot N_{pm} & \text{si el movimiento mezcla programas y el tope de la pila no coincide con la base del movimiento,} \\ 1,5 \cdot A \cdot N_{pm} + 0,5 \cdot N_{pm} & \text{si el movimiento mezcla programas y el tope de la pila coincide con la base del movimiento,} \\ 0 & \text{si no hay mezcla de programas.} \end{cases} \quad (4.6)$$

donde:

- N_{pm} : Número de programas existentes en el movimiento.

- **A:** Parámetro configurable «MixedProgramsInMoveBasePenalty» que representa la penalización por cada programa mezclado.

En la Tabla 4.2 encontramos el desglose de penalizaciones. De esta forma, se ilustran las diferencias de la fórmula aplicada según el caso.

Tabla 4.2: Penalizaciones debidas a mezcla de programa.

Programas en el movimiento	Sin coincidencia base	Coincidencia base	Diferencia
1	N/A	N/A	N/A
2	$2N_{pm}$	$3,5N_{pm}$	$0,5N_{pm}$
3	$4N_{pm}$	$5N_{pm}$	$1N_{pm}$
4	$8N_{pm}$	$6,5N_{pm}$	$1,5N_{pm}$

4.8.1.3. Cohesión de grupos

Esta penalización evalúa el grado de dispersión de los grupos pertenecientes al programa prioritario. Cuando existen huecos entre grupos consecutivos, se penalizan las distribuciones en las que los grupos se encuentran más alejados de la media, favoreciendo así configuraciones más compactas.

$$P_{cohesion} = \begin{cases} A \cdot \frac{1}{\sqrt{N}} \cdot \left(\frac{1}{M} \cdot \sum_{i=1}^M |g_i - \bar{g}| \right) & \text{si existen huecos entre grupos} \\ 0 & \text{en otro caso.} \end{cases} \quad (4.7)$$

donde:

- **A:** Parámetro configurable «GroupsDispersionsPenalty» que representa la penalización a aplicar en función de la dispersión de los grupos.
- **N:** Representa el número de materiales en la ganchada.
- **M:** Número de grupos considerados.
- **g_i :** Representa el valor del grupo del elemento i .
- **$\bar{g}$:** Representa la media de los valores de grupo presentes en la pila y en el movimiento.

La Fórmula 4.7 se divide en dos partes. La parte incluida entre paréntesis mide la desviación absoluta, para obtener cómo de dispersos se encuentran los grupos independientemente del número de grupos existente. Por otro lado, $\frac{1}{\sqrt{N}}$ se encarga de reducir la penalización, a medida que el movimiento evaluado contiene más materiales. De esta forma, para la misma dispersión, los movimientos más grandes se penalizan menos.

La dispersión únicamente se evalúa cuando existen huecos entre grupos consecutivos. Si los grupos presentes forman una secuencia continua, independientemente de su numeración absoluta, se considera que la organización es adecuada y no se aplica ninguna penalización adicional.

4.8.1.4. Orden de grupos

Además de mantener agrupados los materiales, resulta conveniente favorecer el orden ascendente de los grupos de laminación. Para ello, se compara la secuencia de grupos que se produciría en la pila resultante de efectuar el movimiento, con la secuencia ideal. Una vez calculada, se penaliza linealmente en función del número de desbastes que ocupan una posición distinta de la esperada, mediante la siguiente Fórmula 4.8:

$$P_{orden} = N_{des} \cdot A \quad (4.8)$$

donde:

- N_{des} : Número de grupos que no se encuentran en su orden ideal.
- A : Parámetro configurable «UnorderedGroupsPenalty» que representa la penalización por cada grupo que no está ordenado.

La siguiente comparativa entre dos casos ilustra la aplicación de este coste:

- **Grupos: 1, 2, 2, 3:** Resulta en $0 \cdot A = 0$ que al ser el ideal, no acumula ningún coste.
- **Grupos: 1, 2, 3, 2:** Resulta en $2 \cdot A = 2A$ que tiene dos elementos fuera de orden, por lo que se multiplica la penalización base por dos.

En la comparativa anterior, se ignorarán todos los grupos cuya diferencia exceda la configurada en «IgnoreGroupsPenaltyAfter». Por ejemplo, si estamos juntando grupos del 1 al 3 y se ignoran grupos a partir de una diferencia de 14, no se tendría en cuenta el orden del grupo 18 para la penalización. De esta forma, ignoramos aquellos desbastes cuyo grupo está lo suficientemente alejado como para no compensar un esfuerzo de ordenación y dejando que el resto de criterios decida su idoneidad.

Mediante la aplicación de este coste obtenemos mejores puntuaciones en las pilas que favorezcan un mejor orden de laminación, y reduce la necesidad de reordenar para expedir, lo que ayuda a cumplir los objetivos operativos del parque.

4.8.1.5. Enterramiento de materiales con programa

Los materiales que todavía no pertenecen a ningún programa de laminación pueden dificultar la extracción de materiales programados, cuando quedan intercalados entre ellos. El objetivo de esta penalización es limitar el número de intercalaciones de materiales sin programa entre materiales programados y establecer un número máximo de elementos que se pueden quedar enterrados. En el primer caso, se busca reducir las intercalaciones cuando sea posible mientras que en el segundo, se busca limitar las potenciales reordenaciones.

Para el cálculo de la penalización asociada se ha elegido una función lineal que aplica la penalización según el número de elementos enterrados, según la siguiente fórmula:

$$P_{enterramiento\ programa} = \begin{cases} A & N_e > B, \\ N_e \cdot C & 0 < N_e \leq B, \\ 0 & N_e = 0 \end{cases} \quad (4.9)$$

donde:

- N_e : Número de materiales enterrados.
- A : Parámetro configurable «BurriedExceededPenalty» que representa la penalización que obtienen los casos que exceden el enterramiento máximo.
- B : Parámetro configurable «MaxBurrialAllowed» que marca el máximo de enterramiento.
- C : Parámetro configurable «BurriedProgramPenaltyPerItem» que representa la penalización por cada elemento enterrado.

Esta es una regla de afinamiento, cuyo objetivo es establecer preferencias secundarias sobre la regla de programas, por lo que se recomienda utilizar factores pequeños en sus valores.

El enterramiento en el parque ocurre de forma natural cuando una pila de materiales con criterio Tipo recibe una asignación de programa que no cubre todos sus elementos. En estas situaciones, no siempre es conveniente evitar completamente el enterramiento, ya que puede resultar preferible continuar agrupando materiales similares aunque ello implique asumir una reordenación futura. Por este motivo, se le añade un pequeño coste que permite influir en la decisión, sin competir con criterios de mayor importancia.

4.8.1.6. Bloqueo de materiales sin programa

Los materiales que todavía no pertenecen a ningún programa de laminación pueden cubrirse con materiales de programa. El objetivo de esta penalización es ayudar a la propuesta a seleccionar

qué pilas de materiales sin programa resulta más conveniente cubrir. De esta forma, tratamos de garantizar la existencia de soluciones para material de programa, minimizando el número de materiales sin programa que dejamos bloqueados.

Para el cálculo de la penalización asociada se ha elegido una función lineal que aplica la penalización según el número de elementos bloqueados, según la siguiente fórmula:

$$P_{\text{bloqueo}} = N_e \cdot A \quad (4.10)$$

donde:

- N_e : Número de materiales sin programa que quedarían bloqueados.
- A : Parámetro configurable «BurriedNonProgramPenaltyPerItem» que representa la penalización por cada elemento enterrado.

Esta regla se considera una regla de afinamiento, cuyo objetivo es establecer preferencias secundarias dentro de la regla de programas. Por este motivo, se recomienda utilizar valores reducidos para su parámetro de configuración, de forma que su influencia no llegue a competir con los criterios principales de la regla.

El bloqueo de estos materiales puede ser aceptable cuando es por materiales pertenecientes a un programa, ya que estos tienen una prioridad superior y, en principio, serán expedidos antes. Por este motivo, la penalización no pretende impedir este tipo de situaciones, sino introducir un pequeño coste adicional que permita diferenciar entre varias alternativas, favoreciendo aquellas que bloqueen un menor número de materiales sin programa.

4.8.1.7. Número de programas presentes

Finalmente, también se penaliza el coste organizativo de la presencia de varios programas distintos dentro de una misma pila, favoreciendo así que cada pila contenga materiales pertenecientes al menor número posible de programas. La penalización es nula cuando únicamente existe un programa y aumenta linealmente por cada programa adicional presente. La fórmula es la siguiente:

$$P_{\text{programas}} = (N_p - 1) \cdot A \quad (4.11)$$

donde:

- N_p : Número de programas presentes en la pila.

- **A:** Parámetro configurable «BurriedProgramsPenalty» que representa la penalización por cada programa presente en la pila.

Al igual que en el criterio anterior, la coexistencia de varios programas en una misma pila es una situación que aparece de forma natural, cuando materiales sin programa son asignados a uno menos prioritario. La presencia de varios programas se considera una situación más desfavorable, debido al riesgo de que futuras modificaciones en la planificación alteren las prioridades de extracción y obliguen a realizar movimientos adicionales. En cambio, mientras un material permanece sin programa, su reubicación puede planificarse con mayor flexibilidad y realizarse únicamente cuando resulte necesario.

4.8.2. Cubrir programa

El objetivo de esta regla es penalizar aquellas propuestas en las que materiales sin programa quedan situados por encima de materiales ya asignados a un programa. Esta situación incrementa el número de movimientos necesarios para acceder al material programado y, por tanto, debe evitarse siempre que existan alternativas viables.

No obstante, cuando el nivel de ocupación del parque es elevado, puede resultar inevitable cubrir temporalmente materiales programados con materiales sin programa. Por este motivo, la regla no prohíbe esta situación, sino que les asigna una penalización para que únicamente sean consideradas cuando no existan alternativas claramente mejores. En la operación del parque suele permitirse un máximo equivalente al número de desbastes que puede coger la grúa en una ganchada. De esta forma, se permite desordenar pero asegurando que puede corregirse en un único movimiento.

La penalización se calcula mediante la siguiente expresión que emplea una función lineal:

$$P_{cubrir} = \begin{cases} A & \text{si el movimiento contiene material programado que queda} \\ & \text{situado sobre un material sin programa,} \\ 0 & \text{si no existe cobertura de programa,} \\ \frac{A}{B+1} \cdot N_c & \text{en cualquier otro caso.} \end{cases} \quad (4.12)$$

donde:

- **N_c :** Número de materiales sin programa que quedan situados sobre materiales programados.

- *A*: Parámetro configurable «CoveringProgramPenalty» que representa la penalización máxima de este criterio.
- *B*: Parámetro configurable «AllowedCoverageOfProgram» que representa el número máximo de materiales sin programa que pueden cubrir un programa, antes de alcanzar la penalización máxima.

Nótese que la penalización aumenta linealmente con el número de materiales sin programa que cubren un programa. La división entre $B + 1$ garantiza que, cuando se supera el límite configurado, la penalización alcanza exactamente el valor máximo establecido por la regla.

Existe además una situación considerada inválida: cuando dentro de la propia petición de ubicación, un material programado queda situado bajo un material sin programa. En este caso, la regla devuelve directamente la penalización máxima, ya que la propuesta estaría generando una cobertura de programa desde el mismo momento de la asignación.

Cuando todos los materiales implicados pertenecen a un programa, o ninguno de ellos pertenece a un programa, la regla no aplica ninguna penalización adicional, ya que no se produce ninguna cobertura de materiales programados. En estos casos, serán el resto de reglas las encargadas de generar costes para la situación.

4.8.3. Destino

El objetivo de esta regla es favorecer, en primer lugar, la agrupación de material de destino y en segundo lugar, agrupar desbastes del mismo destino, reduciendo el número de movimientos necesarios para cargar los desbastes. Para que esta regla sea aplicable, el movimiento o la pila han de contener material de destino, pero no programa. Cuando es de aplicación, se evalúan distintos aspectos de la pila de forma independiente, asignando una penalización a cada uno de ellos. La penalización total se obtiene como la suma de todas las penalizaciones parciales:

$$P_{total} = P_{mezclaMovimiento} + P_{coincidenciaTopeBase} + P_{estadoFinal} \quad (4.13)$$

La regla evalúa la agrupación de destinos desde tres perspectivas complementarias: la homogeneidad del movimiento, la compatibilidad entre el movimiento y la pila existente y la homogeneidad del estado final tras realizar la asignación. Las distintas perspectivas se desarrollan en los siguientes apartados.

4.8.3.1. Mezcla del movimiento

Esta penalización se centra en evaluar el grado de mezcla de los destinos que aportará el movimiento a la pila. Para esto, se emplea la entropía de Shannon, una medida ampliamente utilizada para evaluar la dispersión de una distribución. Esta medida presenta un valor nulo cuando todos los materiales comparten el mismo destino y aumenta a medida que la distribución es más uniforme. La fórmula utilizada es:

$$P_{mezclaMovimiento} = -A \cdot \sum_{i=1}^X p_i \ln(p_i) \quad (4.14)$$

donde:

- A : Parámetro configurable «MixedInMoveMaxPenalty» que representa la penalización máxima de este criterio.
- X : Representa el número de destinos distintos que existen en el movimiento.
- p_i : Representa la proporción de materiales cuyo destino es i .

De esta forma se obtiene una penalización que depende únicamente de la distribución de destinos presente en el movimiento, sin considerar el orden en el que aparecen los materiales. Así, movimientos con destinos $\{A, A, A, B\}$, $\{A, B, A, A\}$ o $\{B, A, A, A\}$ reciben la misma penalización al presentar el mismo grado de homogeneidad. La evaluación del orden de los materiales se delega al resto de penalizaciones parciales que componen la regla.

4.8.3.2. Coincidencia entre tope y base

Representada como $P_{coincidenciaTopeBase}$ está diseñada para penalizar los casos en los que el destino de la base del movimiento no coincide con el destino presente en el tope de la pila. Esta situación no es la óptima, pero puede ocurrir a partir de niveles de ocupación medios a elevados. Por este motivo, se permite la situación pero se asigna una penalización para que únicamente se considere cuando no existan alternativas mejores. La penalización se configura a través del parámetro «MoveAndTopMismatchPenaly».

4.8.3.3. Estado final

Esta penalización se centra en evaluar el estado final de los destinos en la pila, favoreciendo la existencia del destino mayoritario. Para ello se propone, por su sencillez de parametrización, la siguiente fórmula lineal:

$$P_{estadoFinal} = (1 - X) \cdot A \quad (4.15)$$

donde:

- **X**: Representa la proporción de materiales que pertenecen al destino mayoritario en la pila, una vez realizado el movimiento. Este porcentaje se obtiene considerando conjuntamente los materiales que ya se encuentran almacenados en la pila y los que aporta el movimiento.
- **A**: Parámetro configurable «FinalStateMaxPenalty» que representa la penalización máxima de este criterio.

De esta forma, se favorece que cada pila tienda a contener materiales de un único destino o, al menos, que exista un destino claramente predominante. Este criterio actúa de forma complementaria a la penalización por mezcla del movimiento ya que, mientras aquella evalúa únicamente los materiales que se incorporan, ésta considera el estado final de la pila tras realizar la asignación.

4.8.4. Tipo

El objetivo de esta regla es favorecer que las pilas contengan materiales de un único tipo. Esta agrupación facilita la posterior asignación a programas, ya que estos se generan a partir del tipo y de la anchura de los desbastes. Como consecuencia, mantener juntos los materiales de un mismo tipo ayuda a aumentar la probabilidad de que se encuentren en la misma pila cuando sean asignados a programa. Para que esta regla sea de aplicación, el movimiento o la pila han de contener material de tipo, pero la pila no debe contener materiales asignados a un programa ni a destino.

La evaluación se realiza mediante cuatro criterios evaluados de forma secuencial y en los que, bajo ciertas circunstancias, cada uno puede determinar la puntuación de la regla. La regla se ha diseñado para satisfacer los siguientes objetivos, ordenados de mayor a menor prioridad:

1. Evitar mezclar tipos en el movimiento.
2. Evitar apilar un tipo distinto al presente en el tope.
3. Si hay mezcla, que aparezca en bloques continuos y no alternados.

4.8.4.1. Mezcla de tipos en el movimiento

El primer criterio penaliza aquellos movimientos que contienen materiales de distintos tipos, favoreciendo así a aquellos casos de la propuesta cuyos movimientos no mezclan tipos.

$$P_{movimiento} = (N_{tipos} - 1) \cdot A \quad (4.16)$$

donde:

- **A:** Parámetro configurable «MixedTypesInMovePenalty» que representa la penalización a aplicar por la mezcla de tipos en un movimiento.
- **N_{tipos} :** Representa el número de tipos distintos que tienen los desbastes del movimiento. La penalización aplica cuando hay al menos dos tipos, por lo que se resta uno para que ese caso obtenga exactamente la penalización definida en el parámetro.

Cuando el movimiento tiene dos o más tipos, la puntuación de este criterio determinará el valor de la regla.

4.8.4.2. Compatibilidad con la pila

Cuando el movimiento contiene un único tipo, se evalúa su compatibilidad con la pila existente. Si la pila está vacía, no se aplica penalización. En caso contrario, cuando el tipo situado en la base del movimiento no coincide con el tipo en el tope de la pila, se aplica la penalización máxima:

$$P_{compatibilidad} = \begin{cases} 0 & \text{si ambos tipos coinciden o la pila está vacía,} \\ B & \text{en otro caso.} \end{cases} \quad (4.17)$$

donde B corresponde al parámetro «MixedTypesMaxPenalty».

Cuando la pila está vacía o ambos tipos no coinciden, la puntuación de este criterio determinará el valor de la regla.

4.8.4.3. Número máximo de tipos

Cuando el número de tipos presentes en la pila alcanza o supera el valor configurado mediante el parámetro «MaxMixedTypesInPile», la regla devuelve directamente la penalización máxima definida en «MixedTypesMaxPenalty». Esta restricción evita usar pilas en las que las transiciones obligarían a costosas reorganizaciones cuando los desbastes sean asignados a programas.

4.8.4.4. Organización de la pila

Finalmente, cuando el movimiento resulta compatible con la pila, el último criterio evalúa la organización del estado final obtenido.

Para ello, se utiliza el número de transiciones entre tipos consecutivos presentes en la pila. Cuanto menor sea este número, mayor será el grado de agrupación de materiales del mismo tipo.

La penalización viene dada por

$$P_{organizacion} = \begin{cases} 0 & \text{si la pila solo contiene un único grupo,} \\ \frac{N_{tran}}{C} \cdot B & \text{en cualquier otro caso} \end{cases} \quad (4.18)$$

donde:

- N_{tran} : Representa el número de transiciones entre tipos presentes en la pila. Por ejemplo, la secuencia A, B, A tiene dos transiciones y A, A, B, B únicamente una.
- C : Parámetro configurable «MaxMixedTypesInPile» que representa el número máximo de tipos que debe tener la pila para obtener la penalización máxima.
- B : Parámetro configurable «MixedTypesMaxPenalty» que representa la penalización máxima que puede obtener la pila debido a la mezcla de tipos.

En conjunto, esta regla favorece la creación de pilas formadas por bloques continuos de un mismo tipo. Se intentan evitar casos en los que la mezcla aparece en el movimiento, y se continúa intentando favorecer configuraciones donde los materiales del mismo tipo permanecen agrupados.

4.8.5. Familia

La agrupación por este criterio aplica cuando los materiales no tienen pedido y, por tanto, no tienen un tipo asignado. El problema que se deriva de este hecho es que un material de una familia, al recibir pedido, puede ser asignado a varios tipos. Lo que hace que la agrupación por este criterio, no tenga garantía de generar pilas ordenadas una vez que se asigna el pedido. El objetivo de esta regla es favorecer, en primer lugar, la agrupación de material de familia y en segundo lugar, agrupar desbastes de la misma familia. De esta forma, se maximiza la posibilidad de que, al asignarse un tipo,, se generen el menor número de mezclas en la pila, minimizando los movimientos para su reorganización. Para que esta regla sea aplicable, el movimiento o la pila han de tener material de familia, pero no de ningún otro criterio más prioritario. Cuando

es de aplicación, se evalúan distintos aspectos de la pila de forma independiente, asignando una penalización a cada uno de ellos. La penalización total se obtiene como la suma de todas las penalizaciones parciales:

$$P_{total} = P_{mezclaMovimiento} + P_{coincidenciaTopeBase} + P_{estadoFinal} \quad (4.19)$$

La regla evalúa la agrupación de familias desde tres perspectivas complementarias: la homogeneidad del movimiento, la compatibilidad entre el movimiento y la pila existente y la homogeneidad del estado final tras realizar la asignación. Las distintas perspectivas se desarrollan en los siguientes apartados.

4.8.5.1. Mezcla del movimiento

Esta penalización se centra en evaluar el grado de mezcla de las familias que aportará el movimiento a la pila. Para esto, se emplea la entropía de Shannon, una medida ampliamente utilizada para evaluar la dispersión de una distribución. Esta medida presenta un valor nulo, cuando todos los materiales comparten la misma familia, y aumenta a medida que la distribución es más uniforme. La fórmula utilizada es:

$$P_{mezclaMovimiento} = -A \cdot \sum_{i=1}^{N_f} p_i \ln(p_i) \quad (4.20)$$

donde:

- A : Parámetro configurable «MixedInMoveMaxPenalty» que representa la penalización máxima de este criterio.
- N_f : Representa el número de familias distintas que existen en el movimiento.
- p_i : Representa la proporción de materiales cuya familia es i .

De esta forma, se obtiene una penalización que depende únicamente de la distribución de familias presente en el movimiento, sin considerar el orden en el que aparecen los materiales. Así, movimientos con familias $\{A, A, A, B\}$, $\{A, B, A, A\}$ o $\{B, A, A, A\}$ reciben la misma penalización, al presentar el mismo grado de homogeneidad. La evaluación del orden de los materiales se delega al resto de penalizaciones parciales que componen la regla.

4.8.5.2. Coincidencia entre tope y base

Representada como $P_{coincidenciaTopeBase}$ está diseñada para penalizar los casos en los que la familia de la base del movimiento no coincide con la familia presente en el tope de la pila. Esta situación no es la óptima, pero puede ocurrir a partir de niveles de ocupación medios a elevados. Por este motivo, se permite la situación pero se asigna una penalización para que únicamente se considere cuando no existan alternativas mejores. La penalización se configura a través del parámetro «MoveAndTopMismatchPenaly».

4.8.5.3. Estado final

Esta penalización se centra en evaluar el estado final de las familias en la pila, favoreciendo la existencia de la familia mayoritaria. Para ello se propone, por su sencillez de parametrización, la siguiente fórmula lineal:

$$P_{estadoFinal} = (1 - N_f) \cdot A \quad (4.21)$$

donde:

- N_f : Representa la proporción de materiales que pertenecen a la familia mayoritaria en la pila, una vez realizado el movimiento. Este porcentaje se obtiene considerando conjuntamente los materiales que ya se encuentran almacenados en la pila y los que aporta el movimiento.
- A : Parámetro configurable «FinalStateMaxPenalty» que representa la penalización máxima de este criterio.

De esta forma, se favorece que cada pila tienda a contener materiales de una única familia o, al menos, que exista una claramente predominante. Este criterio actúa de forma complementaria a la penalización por mezcla del movimiento ya que, mientras aquella evalúa únicamente los materiales que se incorporan, ésta considera el estado final de la pila tras realizar la asignación.

Para la configuración de esta regla se utilizan valores bajos de puntuación, debido a que la importancia de un buen orden de las familias no es tan importante como que se agrupen juntas. Una de las tareas que han de realizar los operadores del parque es la reorganización, mediante la propuesta de ubicación, una vez que los desbastes de familia se asignan a pedido y por tanto, se les asigna un tipo.

4.8.6. Anchos

El objetivo de esta regla es favorecer una organización piramidal de ancho de los desbastes dentro de cada pila. Una pila piramidal, en ancho descendente a medida que aumenta la altura, evita las limitaciones dimensionales de seguridad y aumenta la flexibilidad para futuras ubicaciones.

La evaluación de la regla se realiza mediante la evaluación de cada pareja de desbastes consecutivos, comenzando por la base del movimiento hacia su tope. La penalización total se obtiene sumando todas las penalizaciones consecutivas a través de la siguiente fórmula:

$$P_{anchos} = \sum_{i=1}^{N_d} P_i \quad (4.22)$$

donde:

- N_d : Representa el número de desbastes del movimiento.
- P_i : Representa la penalización obtenida de la comparación del desbaste i -ésimo con el que se encuentra inmediatamente debajo del mismo. En caso de que la pila se encuentre vacía, el primer material no generará ninguna penalización, ya que no existe un desbaste con el que compararlo.

Es importante destacar que esta regla no evalúa el estado final completo, sino las transiciones de ancho que ocurren entre cada par de desbastes consecutivos. En la Tabla 4.3 encontramos dos casos representativos. En el caso A, la pila presenta una piramidalidad segura, donde se coloca material estrecho sobre ancho. Como penalizamos cada transición y son pequeñas variaciones en piramidalidad segura, su penalización será pequeña. En el caso B, encontramos varios desbastes anchos sobre estrechos, lo que acumula una penalización por cada transición, resultando en una penalización mayor.

Tabla 4.3: Comparativa de penalizaciones de anchos.

Caso A	Caso B
1800	2150
1850	2000
1900	2150
2000	2000

A continuación describimos los casos que pueden ocurrir durante la evaluación de las transiciones.

4.8.6.1. Igual anchura

Cuando ambos materiales tienen la misma anchura, nos encontramos ante el caso más favorable. Estaremos manteniendo la forma de pila y, por lo tanto, la penalización de este caso será nula.

$$P = 0 \quad (4.23)$$

4.8.6.2. Estrecho sobre ancho

Cuando el material es más estrecho que el anterior, estaremos generando el caso de piramidalidad decreciente, que resulta aceptable. Sin embargo, las diferencias muy elevadas reducen el aprovechamiento de la altura de las pilas. Dado que bloquean la introducción de material del ancho original, será necesario buscar otra pila para materiales de esas características. Por este motivo, se aplica una penalización proporcional a la diferencia de anchuras existente:

$$P = A \cdot \frac{\Delta W}{W_{max}} \quad (4.24)$$

donde:

- A : Representa un factor de penalización variable según la diferencia de anchos. La Tabla 4.4 recoge dichos valores.
- ΔW : Representa la diferencia de ancho entre el desbaste anterior y el evaluado.
- W_{max} : Representa el ancho máximo que puede haber entre dos desbastes consecutivos, según las reglas de seguridad.

Tabla 4.4: Penalización según la diferencia (estrecho sobre ancho).

Diferencia	Penalización
$\leq 30mm$	50
$30 - 100mm$	100
$100 - 200mm$	200
$> 200mm$	300

La utilización de distintos factores introduce una penalización creciente a medida que aumenta la diferencia de ancho.

4.8.6.3. Ancho sobre estrecho

La ubicación de un material ancho sobre estrecho es una circunstancia a evitar porque dificulta futuras ubicaciones. Este caso es más grave que el anterior ya que, debido a las restricciones de seguridad, sería equivalente a establecer un ancho máximo y mínimo en la pila. Para este caso, se propone la Fórmula 4.25 análoga a la Fórmula 4.24:

$$P = 400 \cdot \frac{\Delta W}{W_{max}} \quad (4.25)$$

donde:

- ΔW : Representa la diferencia de ancho entre el desbaste anterior y el evaluado.
- W_{max} : Representa el ancho máximo que puede haber entre dos desbastes consecutivos, según las reglas de seguridad.

Obsérvese que el factor utilizado (400) es superior al máximo empleado en el caso anterior (300), reflejando que estamos ante una situación más perjudicial.

4.9. Configuración de reglas

Las reglas y sus parámetros son objeto de continuos ajustes, tanto durante las pruebas como durante la operativa del almacén. Por este motivo, se ha optado por definir su configuración mediante un fichero JSON, aprovechando su flexibilidad para representar distintos tipos de reglas y sus parámetros asociados.

En el Código 4.1 se puede observar un extracto del fichero utilizado.

```
1 {  
2   "AllowedRules": [  
3     { "$type": "MaxItemsRule" }  
4   ],  
5   "GlobalPenaltyRules": [  
6     {  
7       "$type": "EmptyPileRule",  
8       "EmptyPileUsageMaxPenalty": 300,  
9       "WidthDiffMaxPenalty": 300  
10    }  
11  ],  
12  "LocalPenaltyRules": [  
13    {
```

```
13  {
14      "$type": "BlockingProgramRule",
15      "CoveringProgramPenalty": 10000,
16      "AllowedCoverageOfProgram": 4
17  }
18 ]
19 }
```

Código 4.1: Fichero de configuración de reglas.

Las reglas se agrupan, según su función, en reglas de validación, reglas globales y reglas locales. Cada regla está identificada mediante el campo «\$type», que identifica la regla, mientras que el resto de campos corresponden a los parámetros configurables descritos en las Secciones 4.6, 4.7 y 4.8.

De esta forma, se consigue activar o desactivar reglas simplemente añadiéndolas o eliminándolas del fichero de configuración, así como modificar sus parámetros, sin necesidad de realizar cambios en el código de la aplicación.

En el Capítulo 7 se presenta la configuración empleada durante la evaluación de la propuesta, incluyendo el conjunto de reglas utilizadas y los valores asignados a cada uno de sus parámetros.

4.10. Conclusiones

En este capítulo se ha definido la forma en que la propuesta de ubicación explorará las posibles soluciones, sirviéndose del conjunto de criterios aplicados a través de las reglas para la toma de decisiones.

Para abordar el problema de búsqueda, se plantea una solución mediante un algoritmo A*, definiendo un modelo de estados junto a la lógica para la expansión de los mismos. Se han definido tres tipos de reglas, cuyo comportamiento permite evaluar por separado la validez, la situación global o local para cada problema. De esta forma, se proporciona una capa de abstracción sobre los problemas que han de abordarse. La separación de estas reglas respecto del algoritmo de búsqueda, permite configurar y ampliar los criterios de evaluación sin modificar la estructura general del algoritmo.

El diseño presentado constituye la base sobre la que se desarrollará el siguiente capítulo, donde se describirá su implementación y las principales decisiones adoptadas durante el desarrollo.

Capítulo 5

Implementación

5.1. Introducción

En este capítulo se describirán las dificultades encontradas durante la implementación y que han requerido el desarrollo de soluciones alternativas. Aunque el diseño presentado en el Capítulo 3 sirvió para guiar el desarrollo, durante la implementación surgieron diversos problemas que obligaron a buscar alternativas para cumplir los requisitos. En todos los casos, el objetivo tras las soluciones ha sido la obtención de código no solo funcional sino que cumple los requisitos de usabilidad, fiabilidad y tiempo de respuesta.

Para la implementación del proyecto, se ha empleado la metodología Extreme Programming (XP) Beck [2004](#), basada en iteraciones frecuentes. La aplicación de este enfoque, en el que se abordan los problemas de forma incremental mientras que se valida y refina el sistema, permitió alcanzar los objetivos definidos.

En los apartados siguientes describiremos la evolución de distintos componentes desde su idea inicial, las dificultades encontradas y cómo se resolvieron finalmente.

5.2. Sincronización inventario y configuración

En este apartado se revisa el planteamiento de la sincronización del inventario con la configuración del parque.

5.2.1. Idea inicial

El planteamiento inicial consistía en permitir la configuración del inventario mediante un fichero JSON. Durante el inicio de la aplicación, la información definida en dicho fichero serviría para sincronizar la base de datos, de forma que ésta mantuviera la información necesaria para preservar la integridad referencial con el resto de entidades del sistema.

5.2.2. Problemas encontrados

Durante la implementación se comprobó que mantener la consistencia entre el fichero de configuración y la base de datos requería la gestión de distintos tipos de cambios, entre ellos:

- La eliminación de elementos del fichero debe reflejarse en la base de datos, eliminando no solo el elemento de almacenamiento «StorageElement» sino las ubicaciones almacenadas «StorageSlot».
- La modificación de la capacidad máxima de un elemento puede provocar que exista material almacenado que exceda la nueva capacidad permitida.

5.2.3. Solución

Como solución, se incorporó un proceso de sincronización ejecutado durante el inicio de la aplicación. Este proceso compara la configuración, definida en el fichero JSON, con la información almacenada en la base de datos y aplica las modificaciones necesarias para mantener su consistencia.

Se considera el fichero de configuración como fuente única de verdad. En consecuencia, cuando un elemento es eliminado del fichero, las ubicaciones asociadas dejan de existir y cualquier material almacenado en ellas se considera retirado del parque, eliminándose también su ubicación. Sin embargo, la información correspondiente al material y al historial de movimientos asociados, se conserva sin modificaciones.

Esta aproximación evita discrepancias entre el fichero de configuración y el estado persistido en la base de datos. El criterio aplicado, considera que las modificaciones de la configuración representan cambios físicos en el parque. Por tanto, cuando éstas se realizan, los elementos de almacenamiento afectados deberían encontrarse vacíos, por lo que la eliminación de sus ubicaciones no supone una situación esperable durante la operación del sistema.

5.3. Implementación del estado de la propuesta de ubicación

En este apartado se revisa la implementación de los estados del algoritmo A* utilizado en la propuesta de ubicación.

5.3.1. Idea inicial

El algoritmo A^* , durante su ejecución, necesita mantener información sobre la evolución del parque, dentro de cada posible solución evaluada. Este estado debe permitir al algoritmo modificar el contenido de las pilas sin afectar a otras soluciones. El diseño de los estados debe soportar la expansión de nodos que se ha descrito en la Sección 4.3. El planteamiento inicial consistía en representar el estado mediante un objeto que contenía una colección de pilas, implementadas utilizando las listas proporcionadas por la biblioteca estándar de .NET.

5.3.2. Problemas encontrados

Durante la implementación se comprobó que esta aproximación presentaba un inconveniente importante. Los estados compartían referencias a los mismos objetos en memoria, por lo que, una modificación realizada durante la exploración de una solución, afectaba también al resto de estados derivados.

Para evitar que los estados se vieran afectados entre sí, fue necesario realizar una copia en profundidad del estado que garantizase el aislamiento en memoria.

Sin embargo, se observó que la realización continua de copias profundas incrementaba considerablemente el consumo de memoria y el tiempo necesario para generar nuevos estados, afectando directamente al rendimiento del algoritmo.

5.3.3. Solución

Debido a esta situación, se ha decidido cambiar las estructuras utilizadas por una implementación de pila basada en listas enlazadas. De esta forma, se puede mantener compartido el estado general del parque, que no cambia y constituye la mayor parte de la información de estado. Cada solución, al modificar una pila, añade un nuevo objeto que referencia al anterior tope de pila y deja de ser necesaria la realización de una copia en profundidad del estado del parque.

En la Figura 5.1 se observa cómo la realización de diferentes movimientos modifican el estado de la pila, sin afectar a otros estados derivados del inicial.

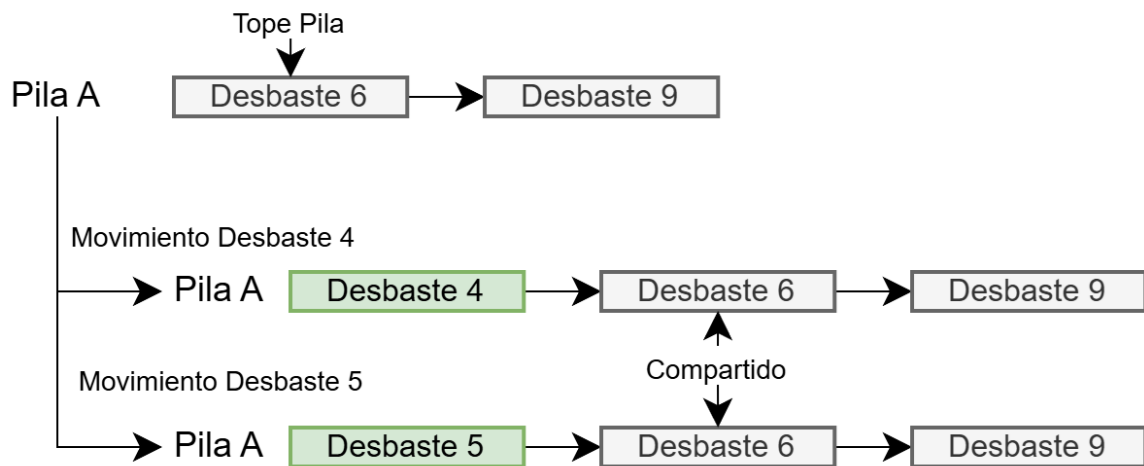


Figura 5.1: Uso de listas enlazadas para representar pilas de desbastes.

Además, cada pila mantiene actualizado un objeto de metadatos con la siguiente información:

- **Count:** Mantiene el número de desbastes contenidos en la pila. Evita recorrer la lista enlazada para obtener el valor.
- **HasProgram:** Indicador binario que refleja si algún material de la pila tiene programa.
- **HasDestination:** Indicador binario que refleja si algún material de la pila tiene destino.
- **HasKind:** Indicador binario que refleja si algún material de la pila tiene tipo.
- **MinWidth:** Mantiene el menor ancho encontrado en la pila.
- **MaxWidth:** Mantiene el mayor ancho encontrado en la pila.
- **MinLength:** Mantiene la menor longitud encontrada en la pila.
- **MostPrioritaryProgram:** Mantiene el orden del programa más prioritario encontrado en la pila.
- **DestinationCounts:** Mantiene un mapa de cada destino y el número de elementos pertenecientes al mismo en la pila.
- **FamilyCounts:** Mantiene un mapa de cada familia y el número de elementos pertenecientes a la misma en la pila.

La actualización de estos metadatos se realiza al modificar el tope de la pila, momento en que es posible computar esta información, a partir del estado anterior, con un coste lineal. De esta forma, se evita tener que recalcularlos durante la evaluación de las reglas, reduciendo el coste de ejecución y simplificando la evaluación.

5.4. Heurístico propuesta ubicación

En este apartado se revisa la implementación del heurístico del algoritmo A^* utilizado en la propuesta de ubicación. En la fase de diseño, en la Sección 4.4, únicamente se definieron los criterios de admisibilidad y monotonía que debía cumplir. En ese punto, no se realizó una propuesta formal debido a la complejidad y necesidad de ajuste experimental.

5.4.1. Idea inicial

La mayor parte de las pruebas, durante la construcción de las reglas y la propuesta, se realizaron utilizando un heurístico que devolvía siempre 0. Esto, efectivamente, convierte el A^* en el algoritmo de Dijkstra, en el que siempre se expande el estado con menor coste acumulado. Esta aproximación, a pesar de ser admisible y monótona, pierde los beneficios de la búsqueda informada de A^* .

El primer punto de partida para la implementación buscaba proporcionar una estimación sencilla del coste restante, utilizando dos criterios:

- **Coste restante de las ganchadas.** Se estimaba la penalización mínima inevitable asociada al tamaño de la última ganchada pendiente, reutilizando la misma función de coste definida para la regla 4.7.3. De esta forma, se dirigía el algoritmo hacia soluciones con ganchadas más grandes.
- **Distancia optimista pendiente.** Se estimaba el número mínimo de movimientos restantes, suponiendo ganchadas completas, y se multiplicaba por la menor penalización por distancia existente entre el origen del movimiento y cualquiera de las pilas del parque.

Este heurístico cumplía la propiedad de admisibilidad, ya que ambos componentes constituían cotas inferiores del coste restante, además de presentar un coste computacional muy reducido.

5.4.2. Problemas encontrados

Durante las pruebas se observó que este heurístico presentaba problemas para diferenciar entre estados distintos. El coste restante de las ganchadas únicamente dependía del número de materiales pendientes, mientras que la componente de distancia suponía que todos los movimientos restantes podrían realizarse hacia la pila más cercana, lo que no resulta realista.

Como consecuencia, numerosos estados obtenían valores heurísticos muy similares, reduciendo la capacidad del algoritmo para priorizar las alternativas más prometedoras. En la práctica, el

algoritmo no era capaz de encontrar una solución, debido al gran número de ramificaciones del problema, motivando el diseño de un heurístico más informativo.

5.4.3. Solución

Los problemas observados durante las pruebas pusieron de manifiesto que el heurístico necesitaba ser significativamente más informativo. El nuevo diseño perseguía tres objetivos principales:

- Reflejar mejor el coste real restante.
- Conservar un coste computacional reducido para no convertirse en un cuello de botella de la búsqueda.
- Mantener las propiedades de admisibilidad y monotonía requeridas.

A continuación se presenta la solución elegida.

5.4.3.1. Idea general

Para evitar caer en los problemas del algoritmo inicial, cuyos resultados eran demasiado planos, debido a la incapacidad de evaluar cómo de prometedor resulta un estado, se busca inspiración en ejemplos típicos de A* en búsqueda de caminos, donde se utiliza la distancia euclídea, ignorando obstáculos, como heurístico.

Partiendo de esta idea, se plantea una forma optimista de calcular, relajando los criterios de decisión, cómo de bien podríamos ubicar los materiales pendientes del estado actual. Se plantea una aproximación que busca determinar la mejor distribución de ancho posible sobre el estado actual del parque. En lugar de intentar determinar directamente los movimientos que deben realizarse, se construye primero una distribución teórica de los materiales, que representa una situación cercana a la ideal, sin considerar todas las restricciones existentes. Posteriormente, esta distribución se utiliza como referencia para la evaluación el estado actual.

Como puede observarse en el Código 5.1, la obtención de la estimación se realiza en tres fases. La primera, construye una distribución teórica de referencia con los materiales pendientes. La segunda, estima el coste pendiente de las ganchadas necesarias para completar la operación. Finalmente, la tercera realiza un desempate entre estados con estimaciones similares.

```

1  1. Construcción de la distribución teórica
2      Generar la mejor secuencia teórica de extracción
3      Crear una representación simplificada del estado mediante pilas
   virtuales
4      Asignación voraz con criterios relajados

```

```

5
6 2. Cálculo del coste pendiente de ganchada
7     Calcular el coste mínimo de realización de ganchadas
8
9 3. Desempate entre estados
10    Calcular el criterio de desempate
11
12 Obtener el coste heurístico a partir de los costes anteriores

```

Código 5.1: Pseudocódigo cálculo heurístico de propuesta de ubicación.

Para el cálculo de la puntuación heurística se utiliza la siguiente fórmula lineal:

$$C_{\text{heurístico}} = C_{\text{ancho}} + C_{\text{ganchada}} - (0,0001 \cdot A) \quad (5.1)$$

donde:

- C_{ancho} : Corresponde al mínimo coste de anchos que podemos obtener al mover al parque los desbastes pendientes. Se explicará en mayor detalle en la Sección 5.4.3.2.
- C_{ganchada} : Corresponde al mínimo coste de realización de ganchadas que podemos obtener para mover al parque los desbastes pendientes. Su funcionamiento se explicará en mayor detalle en la Sección 5.4.3.3.
- A : Corresponde a un criterio empleado para el desempate de casos similares, basado en la finalización de grupos. Se revisará en la Sección 5.4.3.4.

5.4.3.2. Construcción de la distribución teórica

La construcción de la distribución teórica se realiza en varias etapas. Primero, se determina una secuencia teórica de extracción, buscando aproximarse a la distribución ideal de anchos. A continuación, se construye una representación simplificada del estado, mediante pilas virtuales, y se asignan sobre ella los materiales de la secuencia. Finalmente, en función de la asignación realizada, se obtiene el coste asociado a los anchos. Estas etapas se describen a continuación.

5.4.3.2.1 Secuencia teórica

La primera fase de esta operación consiste en la construcción de la mejor secuencia de extracción de los materiales, siguiendo lo que llamaremos la secuencia teórica. La mejor ordenación de anchos consiste en la construcción de pilas con un orden descendente de ancho desde la base hasta el tope de la pila. Siguiendo esa filosofía, se busca calcular cuál sería la mejor forma posible de estructurar el material recibido. Para ello, para cada uno de los tipos de material

pendientes de extraer, se determina su mejor orden de extracción. En la Figura 5.2 se observa un ejemplo de dos pilas, en las que para cada desbaste se muestra su ancho. Todos ellos pertenecen al mismo tipo.

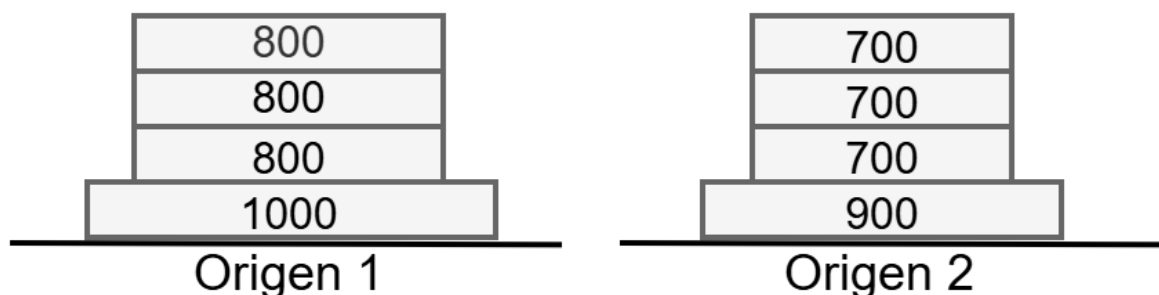


Figura 5.2: Situación de origen para cálculo secuencia teórica.

Sobre el anterior caso, la Figura 5.3 muestra la diferencia entre el orden estrictamente descendente y el mejor orden teórico alcanzable.

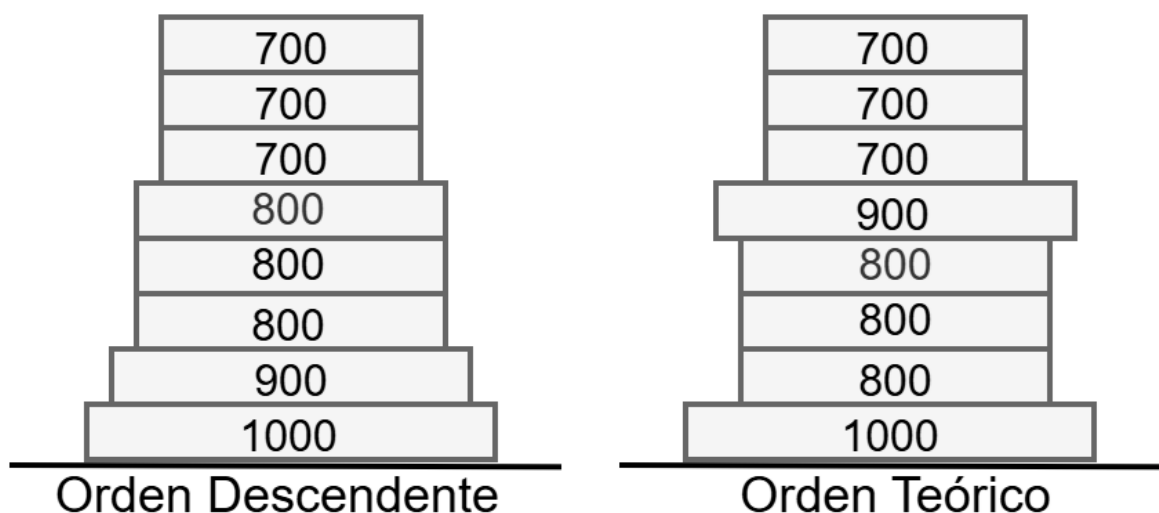


Figura 5.3: Ejemplo orden ascendente vs orden teórico.

Se observa que, mediante la utilización de ganchadas de máximo 4 desbastes, en este caso no es posible obtener el orden estrictamente descendente, asumiendo que se está construyendo una única pila de descarga. El motivo es que la secuencia de extracción está condicionada por la posición de los desbastes en las pilas de origen. En particular, los tres desbastes de ancho 800 se encuentran sobre el desbaste de ancho 1000. Por tanto, para mantener su posición relativa al trasladarlos a una única pila, estos cuatro desbastes deben considerarse conjuntamente en una

misma ganchada. De esta forma, el 1000 queda situado en la base de la nueva pila y los tres 800 sobre él.

La misma situación ocurre en la segunda pila, de forma que también deben trasladarse conjuntamente. Como consecuencia, el 900 debe quedar situado entre los desbastes de ancho 800 y los de ancho 700, dando lugar al orden teórico mostrado en la Figura 5.3.

La secuencia obtenida proporciona una referencia sobre la mejor organización que puede plantearse bajo las restricciones consideradas en esta primera fase. A partir de esta secuencia, se construye una representación del estado actual, sobre la que se realiza la asignación de los materiales para la estimación de los costes de ancho.

Esta aproximación permite que el heurístico evalúe la proximidad del estado actual a una distribución favorable, sin necesidad de construir una solución completa para cada estado explorado.

5.4.3.2.2 Pilas virtuales

Para simplificar la gestión del estado y evitar problemas de memoria como los comentados en la Sección 5.3, se ha implementado una representación simplificada de las pilas. De esta forma, no se mantienen todos los materiales que forman cada pila, sino únicamente la información necesaria para realizar la evaluación. Para cada pila, se mantiene el material situado en el tope y la capacidad restante de la misma.

La representación utiliza únicamente las pilas reales que no se encuentran vacías en el estado y, adicionalmente, incorpora cuatro pilas vacías que representan posiciones sobre las que podrían comenzar a ubicarse los materiales pendientes.

Para la representación interna se utilizan vectores de .NET, que serán iterados durante el proceso de asignación, descrito a continuación. De esta forma, la evaluación se realiza sin modificar ni almacenar una copia completa del estado original.

5.4.3.2.3 Asignación voraz con criterios relajados

Una vez construido el estado, los materiales de la secuencia teórica se asignan de forma voraz sobre las pilas representadas. Para cada material, se evalúan las diferentes pilas candidatas y se estima el coste que tendría colocarlo sobre el material situado actualmente en el tope de cada una de ellas. El material se asigna a la pila con el menor coste, actualizando posteriormente la representación de la pila, para continuar con el siguiente material de la secuencia.

Para el cálculo del coste se emplea una versión simplificada de las reglas de la propuesta de ubicación que únicamente compara las características del desbaste con las del tope de la pila.

Los criterios considerados son:

- **Ancho:** se mantiene la estructura de penalización de la regla de anchos, incluyendo los diferentes costes en función de la magnitud y el sentido de la diferencia. Su funcionamiento completo se ha descrito en la Subsección 4.8.6.
- **Tipo, cobertura de programa y familia:** para estos tres criterios únicamente se considera la existencia de una diferencia entre el desbaste y el tope de la pila, aplicando la penalización correspondiente. Las reglas completas están descritas, respectivamente, en las Subsecciones 4.8.4, 4.8.2 y 4.8.5.

De esta forma, el coste de colocación conserva los principales criterios usados por la propuesta de ubicación, pero omitiendo la mayor parte de sus costes. Esta relajación, permite obtener una cota inferior del coste sin ejecutar todas las comprobaciones del cálculo completo.

Cuando la pila candidata se encuentra vacía, no es posible comparar el material con un elemento situado en el tope. En este caso, se aplica la penalización correspondiente al criterio de ancho del desbaste de la regla de pilas vacías descrita en la Sección 4.7.1.1.

El coste final de la solución se obtiene realizando la suma de los costes de la asignación voraz realizada para cada material.

5.4.3.3. Cálculo del coste pendiente de ganchada

El cálculo del coste pendiente se realiza de forma equivalente a la obtención del coste de la regla de tamaño de ganchada, definida en la Sección 4.7.3. Durante su evaluación, se obtiene el coste asociado a los materiales que quedan pendientes de extraer en cada pila de origen.

Para cada origen, se calcula el resto de dividir su número de materiales pendientes entre los cuatro desbastes que puede tener como máximo una ganchada. Cuando el número de materiales restantes es múltiplo de cuatro, todas las ganchadas pueden realizarse con el tamaño máximo y no se incurre en penalización alguna. En caso contrario, queda una última ganchada de tamaño inferior al máximo, a la que se asigna un coste, según la Tabla 4.1.

De esta forma, para cada pila de origen, se obtiene un coste y la suma de estos valores da como resultado el coste pendiente de ganchada asociado al estado. Es importante recordar que la propuesta no considera la realización de movimientos intermedios o de reordenación de los materiales a descargar.

Esta estimación favorece los estados en los que los materiales pendientes pueden agruparse en ganchadas de mayor tamaño. De esta forma, se dirige la exploración hacia estados que permiten transportar un mayor número de desbastes en cada movimiento.

5.4.3.4. Desempate entre estados

Finalmente, se introduce un criterio de desempate para distinguir entre estados que presentan costes heurísticos similares. Consiste en calcular, sobre los desbastes pendientes de colocar, un número de grupos determinado por los criterios de la propuesta, de la siguiente forma:

- **Programa:** Se cuenta un grupo para cada uno de los programas presentes.
- **Destino:** Se cuenta un grupo para cada uno de los destinos presentes.
- **Tipo:** Se cuenta un grupo para cada uno de los tipos presentes.
- **Familia:** Se cuenta un grupo para cada una de las familias presentes.

El número de grupos es comparado con un valor máximo de referencia, correspondiente a 24 grupos. Este valor asume el caso peor de la descarga de un tren en el que cada desbaste, de los 24 que puede tener, pertenece a un grupo distinto. El valor de A se obtiene a partir de la siguiente fórmula:

$$A = (24 - N_{\text{grupos}}) \cdot 100 \quad (5.2)$$

donde N_{grupos} representa el número de grupos que permanecen entre los materiales pendientes.

De esta forma, los estados en los que se han completado un mayor número de grupos, reciben un pequeño beneficio frente a otros estados con un coste heurístico equivalente. Al tratarse de un valor tan pequeño, y sólo utilizarse como resta, evitamos que este pueda afectar a la admisibilidad y únicamente sirva para diferenciar entre alternativas similares. Por este motivo, su contribución a la función heurística se mantiene deliberadamente pequeña, mediante el factor 0.0001.

5.4.3.5. Propiedades del heurístico

La solución propuesta busca ofrecer una estimación optimista del coste, mientras que proporciona suficiente información para guiar el algoritmo hacia la solución. A continuación, se analizan sus características en relación con las propiedades planteadas durante el diseño.

5.4.3.5.1 Admisibilidad

La admisibilidad constituye la principal propiedad requerida para preservar la optimalidad de la búsqueda. Un heurístico es admisible cuando no sobreestima el coste necesario para alcanzar una solución desde un estado determinado. En este caso, el coste de una solución se obtiene mediante la relajación de los mismos criterios que usan las reglas, lo que permite obtener una cota inferior del coste.

En particular, el coste asociado a los anchos se calcula sobre una versión simplificada del estado y mediante una asignación que no considera otros aspectos evaluados por las reglas. Por ejemplo, se omite cualquier coste relacionado con el estado de ordenación de la pila. De esta forma, el coste de una solución real no será inferior al estimado.

De la misma forma, el coste pendiente de ganchada únicamente considera la estructura de las pilas de origen y la posibilidad de formar ganchadas de tamaño máximo. No se consideran las restricciones de seguridad ni otros factores que podrían obligar a realizar ganchadas menos favorables durante la ejecución. En consecuencia, este componente tampoco puede introducir un coste superior al de una solución real.

Por tanto, para cualquier nodo del estado S , se cumple:

$$C_{\text{ancho}}(S) + C_{\text{ganchada}}(S) \leq C_{\text{restante}}(S) \quad (5.3)$$

donde $C_{\text{restante}}(S)$ representa el coste hasta la meta aplicando las reglas completas.

Finalmente, el criterio de desempate se incorpora mediante el término A restado al coste principal y como $A(S) \geq 0$, este únicamente puede reducir la estimación. Se concluye que el heurístico constituye una cota inferior del coste restante y por lo tanto, es admisible.

5.4.3.5.2 Capacidad de discriminación

Además de resultar admisible, el nuevo diseño mejora la capacidad del heurístico para diferenciar entre estados. El heurístico inicial producía valores demasiado similares entre estados, reduciendo su capacidad para dirigir la búsqueda hacia las alternativas más prometedoras.

La nueva estimación incorpora información sobre la mejor distribución de ancho, que constituye uno de los criterios relevantes en la evaluación de las soluciones. Esto permite diferenciar estados que, a pesar de presentar un coste acumulado similar, pueden ofrecer diferentes posibilidades para completar la operación.

El criterio de desempate permite, además, establecer un orden entre estados con estimaciones similares, favoreciendo aquellos que han conseguido completar un mayor número de grupos.

Durante la implementación se comprobó que la introducción de este nuevo heurístico permitía encontrar soluciones en casos en los que la implementación inicial no lo conseguía. Sin embargo, la mejora del heurístico por sí sola no evita la explosión combinatoria en casos complejos de descarga de trenes, donde el tiempo de cómputo puede seguir siendo elevado. Por ello, para tratar estos casos, se introduce el mecanismo descrito en la Sección 5.7, basado en la división del problema en subproblemas.

5.5. Diagnóstico propuesta ubicación

Durante el desarrollo de la propuesta de ubicación se hizo evidente que la complejidad del algoritmo impedía evaluar completamente el comportamiento mediante pruebas de funcionamiento. Por este motivo, fue necesario diseñar un sistema de diagnóstico que permitiera registrar la información generada durante la búsqueda y facilitar, tanto la depuración como el análisis del comportamiento del algoritmo.

5.5.1. Idea inicial

El planteamiento inicial consistía en evaluar la utilidad de la propuesta basándose únicamente en la realización de pruebas en diferentes estados del parque. Para ello, se pretendía utilizar el conocimiento de dominio disponible para evaluar manualmente casos de descarga de transportes y reubicaciones sobre el almacén.

5.5.2. Problemas encontrados

A medida que aumentaba el número de reglas y el tamaño del espacio de búsqueda, la depuración mediante puntos de interrupción dejó de ser una herramienta útil. Una única propuesta podía generar un gran volumen de estados, cada uno con cientos de evaluaciones de reglas y distintos valores heurísticos. Analizar manualmente esta información resultaba prácticamente imposible, especialmente porque la evolución del parque únicamente existía como estados internos del algoritmo y no podía visualizarse desde la aplicación.

Esto impedía responder a preguntas como:

- ¿Por qué no se ha movido el material a la pila X?
- ¿Qué valores de penalización se están obteniendo al aplicar la regla X?
- ¿Cómo evolucionan los valores del heurístico durante la búsqueda?

A continuación, se revisa la solución propuesta para poder responder a estas preguntas y guiar el desarrollo y ajuste de la propuesta de ubicación.

5.5.3. Solución

Para resolver el problema se incorporó un sistema de recolección de métricas de uso que permitiese evaluar el comportamiento de la propuesta, sin necesidad de la depuración manual. Para

ello, se procedió a diseñar un modelo de diagnóstico que pudiera ser persistido y utilizado, posteriormente, para analizar el comportamiento mediante consultas a la base de datos.

Los objetivos del diagnóstico realizado son:

- Poder examinar los resultados de la evaluación de las reglas, tanto de validación, como globales y locales.
- Obtener métricas de los valores del heurístico.
- Obtener métricas de la función «f» ejecutada para cada nodo, compuesta por la suma del coste desde el estado inicial hasta el nodo y del resultado del heurístico para ese nodo.
- Reconstruir la evolución de la búsqueda y los caminos seguidos.

La Figura 5.4 muestra el modelo relacional correspondiente a la información de diagnóstico de la propuesta de ubicación.

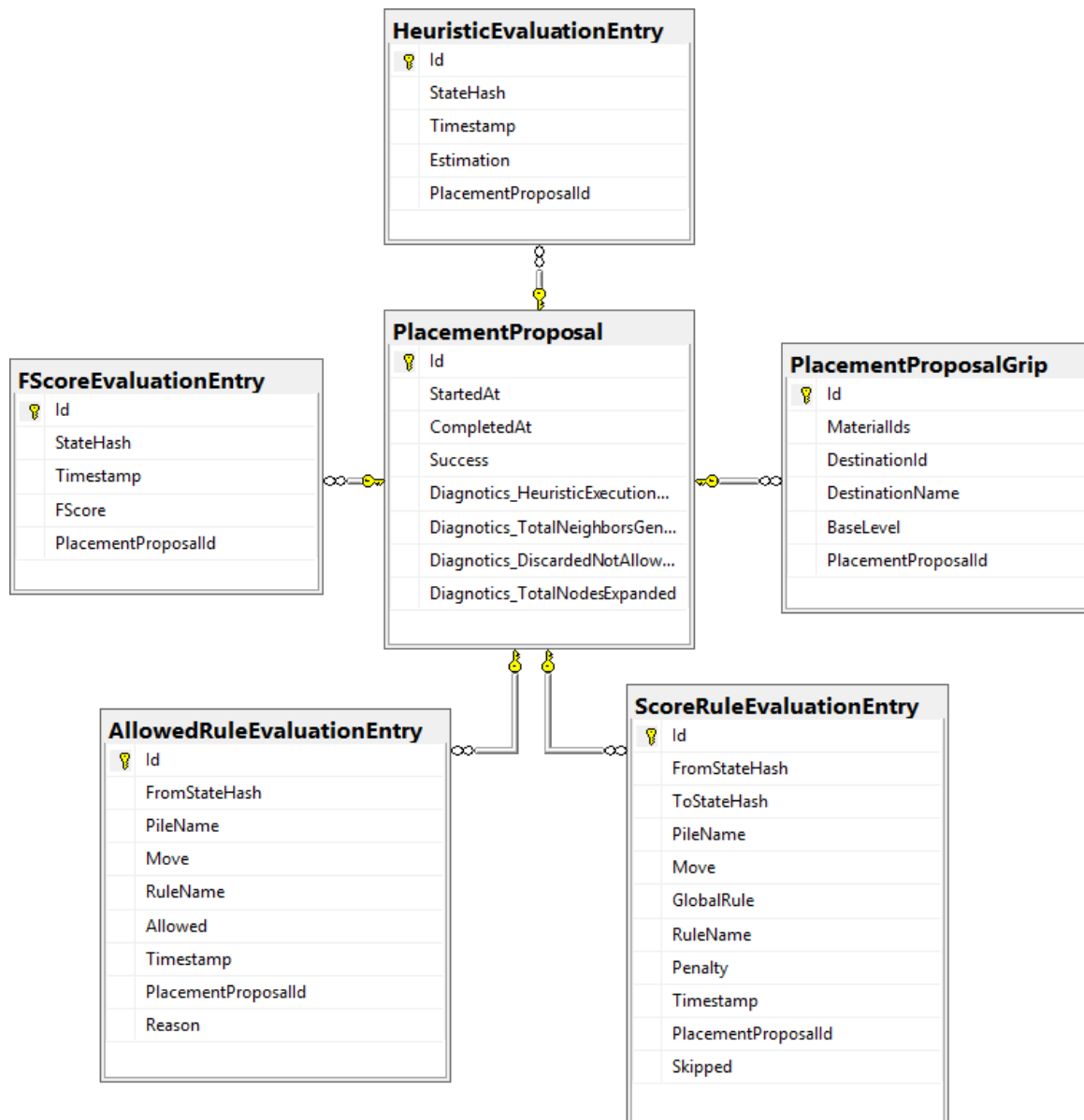


Figura 5.4: Diagrama ER diagnóstico propuesta ubicación.

El modelo se organiza alrededor de la entidad «PlacementProposal», que representa una ejecución completa del algoritmo. A partir de ella, se relacionan las distintas entidades encargadas de almacenar, tanto la solución obtenida como las métricas generadas durante la ejecución del algoritmo. Las propiedades de esta entidad son:

- **Id:** Identificador único de cada propuesta ejecutada en el sistema.
- **StartedAt:** Marca temporal del inicio de la ejecución.
- **CompletedAt:** Marca temporal de la finalización de la ejecución.
- **Success:** Indicador de si se ha encontrado una solución o no.
- **Diagnostics HeuristicExecutionCount:** Número de ejecuciones del algoritmo heurístico.

- **Diagnostics TotalNeighborsGenerated:** Número total de vecinos generados en la expansión de estados.
- **Diagnostics DiscardedNotAllowedCount:** Número total de estados descartados debido a las reglas de validación descritas en la Sección 4.6.
- **Diagnostics TotalNodesExpanded:** Número total de nodos expandidos en el proceso de búsqueda.

La entidad «PlacementProposalGrip» almacena la solución final propuesta por el algoritmo. Cada registro representa una de las ganchadas generadas, indicando tanto los materiales que la componen como la pila y altura propuestas para su almacenamiento. Sus propiedades son:

- **Id:** Identificador único de cada ganchada.
- **MaterialIds:** Concatenación de los identificadores de los materiales de la ganchada.
- **DestinationId:** Identificador del «StorageElement» destino de la ganchada.
- **DestinationName:** Nombre del destino, utilizado para facilitar la visualización.
- **BaseLevel:** Altura a la que se propone colocar el primer material de la ganchada.

Debido al gran número de estados generados, no resulta viable persistir el estado completo de cada uno de ellos. En su lugar, para la identificación de los nodos en la persistencia, se utiliza el hash interno que se mantiene para diferenciarlos durante la ejecución del algoritmo. Este identificador nos permite reconstruir la secuencia de movimientos de cada estado a través de la entidad «ScoreRuleEvaluationEntry», que almacena referencias al estado anterior y posterior además de las siguientes propiedades:

- **Id:** Identificador único de cada evaluación.
- **FromStateHash:** Hash del estado anterior al evaluado.
- **ToStateHash:** Hash del estado actual.
- **PileName:** Nombre de la pila sobre la que se evalúa la regla.
- **Move:** Secuencia de identificadores de material, concatenados por comas, que conforman el movimiento a la pila evaluada.
- **GlobalRule:** Indicador de si la evaluación corresponde a una regla global o local.
- **RuleName:** Nombre de la regla evaluada.
- **Penalty:** Valor de penalización obtenido de la evaluación de la regla.
- **Timestamp:** Fecha en la que se realizó la evaluación.
- **Skipped:** Indica si la regla se ejecutó o no, según los criterios de aplicabilidad de la misma. Válido solo para reglas locales.

La entidad «AllowedRuleEvaluationEntry» almacena la información sobre la ejecución de las reglas de validación. Ésta nos permite analizar, no solo las soluciones exploradas, sino también aquellas que fueron descartadas antes de incorporarse a la búsqueda. Sus propiedades son las siguientes:

- **Id:** Identificador único de cada evaluación.
- **FromStateHash:** Identificador del estado previo al evaluado.
- **PileName:** Nombre de la pila sobre la que se evalúa la regla.
- **Move:** Secuencia de identificadores de material, concatenados por comas, que conforman el movimiento a la pila evaluada.
- **RuleName:** Nombre de la regla de validación evaluada.
- **Allowed:** Indicador del resultado obtenido. Si alguna de las reglas del estado devuelve que no está permitido, no se continuará la exploración del nodo.
- **Timestamp:** Fecha en la que se realizó la evaluación.
- **Reason:** Texto descriptivo del motivo, en caso de que la evaluación devuelva no permitido.

Finalmente, las entidades «HeuristicEvaluationEntry» y «FScoreEvaluationEntry» almacenan los valores obtenidos durante la evaluación del heurístico y de la función $f(n) = g(n) + h(n)$ para cada estado explorado. Gracias a esta información es posible analizar, posteriormente, cómo evoluciona la búsqueda y comprobar si el heurístico está guiando el algoritmo de la forma esperada.

Tras las primeras pruebas se observó que el volumen de información generado por los diagnósticos era muy elevado. Por ejemplo, en la siguiente ejecución de la propuesta, sobre 5 materiales se obtuvieron:

Tabla 5.1: Recuentos de filas en tablas de diagnóstico.

Tabla	Número Filas
AllowedRuleEvaluationEntry	102498
FScoreEvaluationEntry	47920
HeuristicEvaluationEntry	47921
ScoreRuleEvaluationEntry	431280
PlacementProposalGrip	5
PlacementProposal	1
Total	629624

Es decir, sobre un caso de dimensiones reducidas pueden generarse aproximadamente 630 k

filas de diagnóstico. Este resultado pone de manifiesto la necesidad de realizar la persistencia de la información de forma asíncrona, para evitar una sobrecarga significativa en el tiempo de ejecución de la propuesta.

Para evitar que el diagnóstico afectase al rendimiento del algoritmo, se desacopló completamente la generación de información de su persistencia. Los eventos de diagnóstico se almacenan inicialmente en memoria y son procesados, posteriormente, por un servicio en segundo plano encargado de realizar inserciones masivas en la base de datos. Además, se incorporó una librería especializada en operaciones en lotes, reduciendo considerablemente el coste asociado a la escritura de grandes volúmenes de información.

Una vez almacenada la información de diagnóstico, es posible realizar consultas que permiten analizar el comportamiento del algoritmo durante su ejecución. Por ejemplo, la consulta incluida en el Código 5.2, obtiene estadísticas de las penalizaciones obtenidas por cada regla, facilitando la identificación de aquellas que tienen mayor influencia sobre el coste de los estados explorados.

```

1 SELECT
2     RuleName ,
3     COUNT(*) AS EvaluationCount ,
4     AVG(Penalty) AS AveragePenalty ,
5     MIN(Penalty) AS MinPenalty ,
6     MAX(Penalty) AS MaxPenalty
7 FROM ScoreRuleEvaluationEntry
8 WHERE Skipped = 0
9 GROUP BY RuleName
10 ORDER BY AveragePenalty DESC;
```

Código 5.2: Consulta estadísticas reglas propuesta.

La Figura 5.5 muestra el resultado de la ejecución de la consulta anterior para el análisis de las penalizaciones.

RuleName	EvaluationCount	AveragePenalty	MinPenalty	MaxPenalty
KindRule	6772383	4998,36409133978	0	18000
ProgramRule	136135	897,599548415386	0	5250
WidthPenaltyRule	6908581	537,704383505093	0	1659,375
BlockingProgramRule	136135	376,508612774085	0	6000
FamilyRule	62	242,096774193548	0	660
GripsRule	6908581	32,4796944553447	0	40
DistanceRule	6908581	9,79894290018394	0,8	30,5214285714286
EmptyPileRule	6908581	9,02923681063605	0	492,631578947368

Figura 5.5: Resultado consulta estadísticas reglas propuesta.

Podemos observar otro ejemplo en la consulta incluida en el Código 5.3. Se muestra, para cada propuesta de ubicación, su información de diagnóstico junto con la media y máximo del heurístico. Esta información se puede utilizar para evaluar el heurístico y cómo está guiando la búsqueda.

```

1 WITH HeuristicMeasures AS
2 (
3     SELECT
4         PlacementProposalId,
5         AVG(Estimation) AS HeuristicAverage,
6         MAX(Estimation) AS HeuristicMax
7     FROM HeuristicEvaluationEntry
8     GROUP BY PlacementProposalId
9 )
10 SELECT TOP (1000) pa.[Id]
11     , DATEDIFF(MILLISECOND, pa.[StartedAt], pa.[CompletedAt]) AS
    Elapsed
12     , (
13         SELECT COUNT(DISTINCT LTRIM(RTRIM(s.value)))
14         FROM [genstock-db].[dbo].[PlacementProposalGrip] pg
15         CROSS APPLY STRING_SPLIT(pg.MaterialIds, ',') s
16         WHERE pg.PlacementProposalId = pa.Id
17             AND NULLIF(LTRIM(RTRIM(s.value)), '') IS NOT NULL
18     ) AS NumMaterials
19     , pa.[Success]
20     , pa.[Diagnostics_HeuristicExecutionCount] AS
    HeuristicExecutionCount
21     , pa.[Diagnostics_TotalNeighborsGenerated] AS
    TotalNeighborsGenerated
22     , pa.[Diagnostics_DiscardedNotAllowedCount] AS
    DiscardedNotAllowedCount
23     , pa.[Diagnostics_TotalNodesExpanded] AS TotalNodesExpanded
24     , hm.HeuristicAverage
25     , hm.HeuristicMax
26 FROM [genstock-db].[dbo].[PlacementProposal] pa
27 INNER JOIN HeuristicMeasures hm on pa.Id = hm.PlacementProposalId
28 WHERE Success = 1 AND Diagnostics_HeuristicExecutionCount IS NOT NULL
29 ORDER BY StartedAt ASC

```

Código 5.3: Consulta diagnóstico de propuestas de ubicación.

La Figura 5.6 muestra el resultado de la ejecución de la consulta anterior para el análisis de las propuestas de ubicación.

Id	Elapsed	NumMaterials	Success	HeuristicExecutionCount	TotalNeighborsGenerated	DiscardedNotAllowedCount	TotalNodesExpanded	HeuristicAverage	HeuristicMax
DE3F...	73224	8	1	1014889	1651453	636565	8497	156,21593889552	543,52
F186...	1936	5	1	69159	169340	100182	1135	37,0839501728004	119,77
AD74...	163	4	1	916	3730	2815	28	31,030534934498	79,77
7D35...	53	1	1	36	62	27	1	1,10472222222222	39,77
8459...	154	2	1	69	311	243	4	9,07724637681159	39,77
50EA...	76	1	1	5	63	59	1	7,954	39,77
709B...	50	1	1	6	63	58	1	6,62833333333333	39,77
B320...	6045	6	1	125976	204416	78441	1093	117,002602559204	453,52
FAF6...	366	5	1	9298	21010	11713	165	36,9369369756953	119,78
35D6...	17	1	1	17	64	48	1	2,33941176470588	39,77

Figura 5.6: Resultado consulta diagnóstico de propuestas de ubicación.

En el Capítulo 7 se utilizará la información recopilada por este sistema para evaluar los resultados de diferentes ejemplos, analizando el comportamiento del algoritmo y el funcionamiento de las distintas reglas y del heurístico.

5.6. Optimización de la distancia a recorrer

Durante el desarrollo, se detectó que el comportamiento de los movimientos generados resultaba diferente al realizado típicamente por los grúas para reducir la distancia recorrida con la grúa. La propuesta de ubicación determina el destino óptimo para los desbastes y la secuencia para alcanzarlo. Sin embargo, esta secuencia no minimiza el recorrido que es necesario realizar para su ejecución. En este apartado se presenta una optimización aplicada a la solución del algoritmo, para reducir la distancia recorrida sin modificar el estado final.

5.6.1. Idea inicial

Durante la fase de diseño no se había detectado esta necesidad, por lo que se consideraba que los movimientos generados por la propuesta de ubicación ya optimizarían la distancia.

5.6.2. Problemas encontrados

Para ilustrar el problema utilizaremos el ejemplo de la Figura 5.7, donde observamos el movimiento de dos desbastes, que se encuentran en la misma pila, a dos destinos distintos. Asumimos, en este caso, que los desbastes no irán juntos debido a alguno de los criterios evaluados.

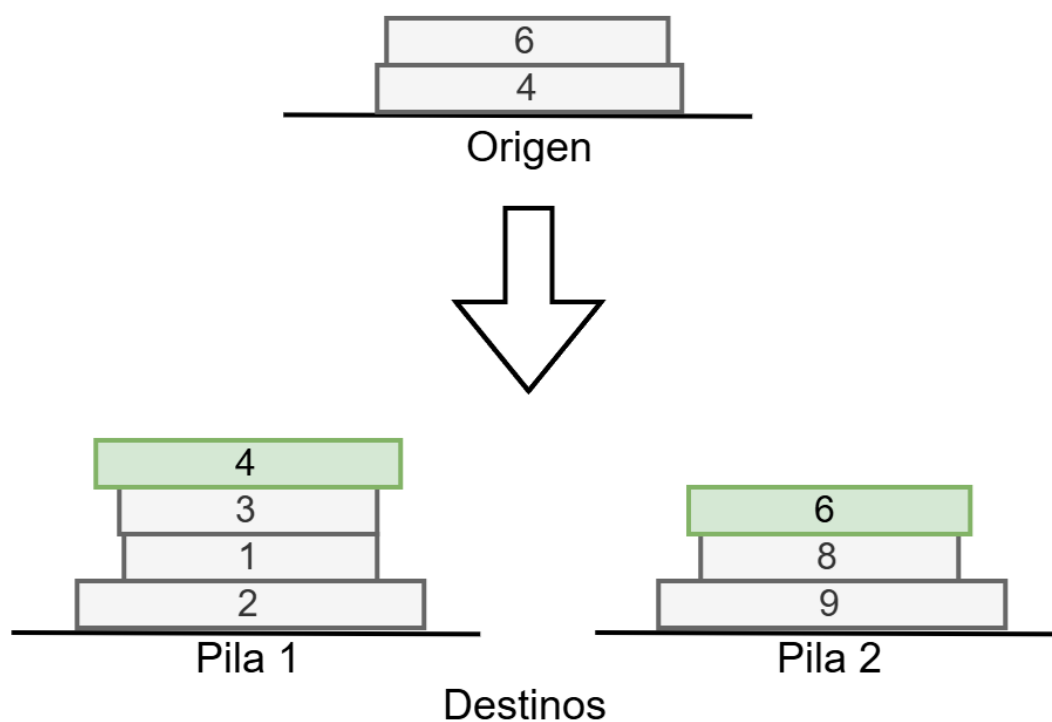


Figura 5.7: Caso ejemplo necesidad de optimización de distancias.

Teniendo en cuenta que la propuesta considerada no optimiza la distancia, el algoritmo genera la secuencia mostrada en la Tabla 5.2.

Tabla 5.2: Movimientos propuestos sin optimización de distancias.

Orden	Desbaste	Origen	Destino
1º	6	Origen (Altura 2)	Pila 2 (Altura 3)
2º	4	Origen (Altura 1)	Pila 1 (Altura 4)

Aunque esta solución es correcta y deja los materiales en su ubicación óptima, si nos fijamos en el recorrido que realiza la grúa para llevar a cabo los movimientos en la Figura 5.8, observamos que el primer movimiento implica una vuelta de vacío al origen.

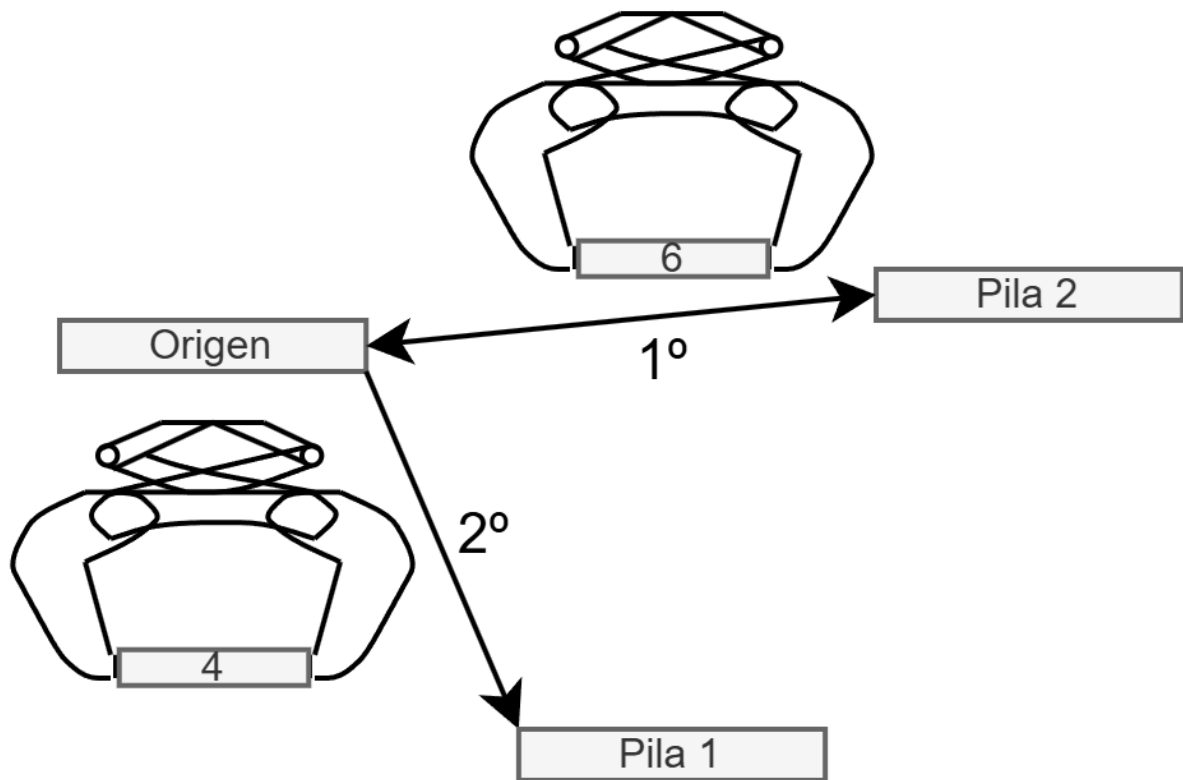


Figura 5.8: Realización del movimiento sin optimización de distancias.

Es decir, la secuencia es:

1. Transporte de desbaste 6 de Origen a Pila 2.
2. Vuelta de vacío a Origen.
3. Transporte de desbaste 4 de Origen a Pila 1.

La distancia recorrida por la grúa viene dada por $2d_{Origen,Pila2} + d_{Origen,Pila1}$ donde el término adicional $d_{Origen,Pila2}$ corresponde al desplazamiento de vacío al origen para ejecutar el segundo movimiento.

5.6.3. Solución

La distancia recorrida puede optimizarse teniendo en cuenta que el estado final obtenido por la propuesta no depende del orden en el que se ejecuten ambos movimientos. Aprovechando esta propiedad, es posible transformar determinadas secuencias de movimientos para reducir el recorrido de la grúa, sin modificar la distribución final del parque.

Esta optimización será aplicable cuando:

- Ambos movimientos parten de la misma pila de origen.
- El número de materiales de ambos movimientos no utiliza la capacidad completa de la ganchada.
- El material transportado en el primer movimiento puede incorporarse, temporalmente, al segundo destino, sin alterar el estado final.

La optimización de distancia recorrida realiza los siguientes cambios en la secuencia de movimientos, descrita anteriormente:

- Se invierte el orden de los movimientos.
- Los materiales correspondientes al segundo movimiento se incorporan a la ganchada del primero.
- Tras depositar la ganchada completa en el primer destino, únicamente se transporta el subconjunto restante hasta el segundo destino.

De esta forma el ejemplo anterior quedaría como se observa en la Figura 5.3.

Tabla 5.3: Movimientos propuestos con optimización de distancias.

Orden	Desbaste	Origen	Destino
1º	6, 4	Origen (Altura 2)	Pila 1 (Altura 4)
2º	6	Pila 1 (Altura 5)	Pila 2 (Altura 3)

El recorrido correspondiente puede observarse en la Figura 5.9.

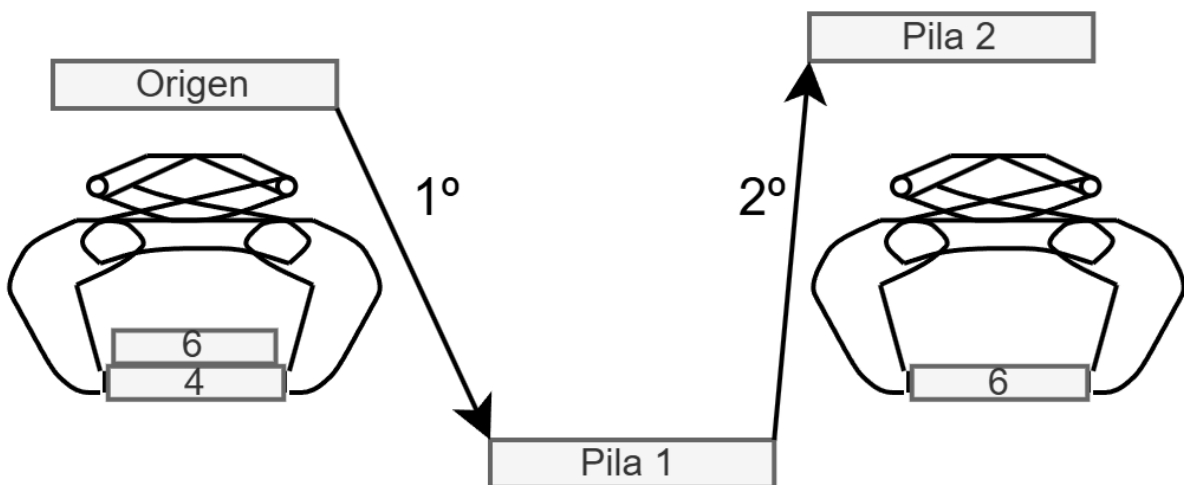


Figura 5.9: Realización del movimiento con optimización de distancias.

Con esta optimización, la distancia recorrida pasa a ser $d_{OrigenPila2} + d_{OrigenPila1}$ dado que se

ha eliminado el movimiento de vacío de la grúa.

Además de reducir la distancia, también mejora el tiempo total de la secuencia de movimientos. El segundo desplazamiento comienza desde una pila en la que la grúa acaba de depositar material, por lo que únicamente es necesario elevar la pinza para recoger el bloque superior, evitando un nuevo posicionamiento sobre la pila de origen.

Esta optimización se realiza en una etapa posterior a la propuesta de ubicación, por lo que no se modifica el estado final obtenido por el algoritmo A*.

5.7. Descarga de trenes

Durante el desarrollo y pruebas de la propuesta de ubicación se hizo evidente que, el gran número de casos generados en la descarga de trenes, derivaba en un número de cálculos demasiado elevado. El algoritmo no era capaz de obtener una solución en un intervalo de tiempo aceptable para su uso en las operaciones del parque. En este apartado se presenta una descomposición en subproblemas, que permite gestionar estos casos, manteniendo la calidad de las soluciones obtenidas.

5.7.1. Idea inicial

El planteamiento inicial para la descarga consistía en realizar peticiones de propuesta de ubicación individuales para cada uno de los vagones del tren. El problema es que, para el caso que se observa en la Figura 5.10, los materiales indicados en verde sería ideal que fueran descargados juntos y se ubicasen en la misma pila. Con la propuesta de ubicación individual, esto no ocurriría al realizarse la descarga del primer vagón antes que la del segundo.

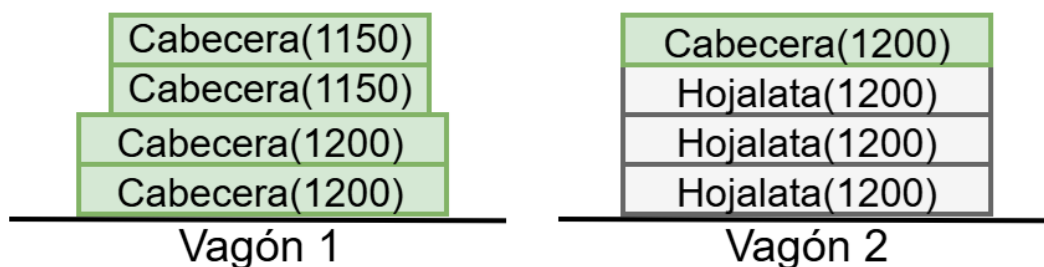


Figura 5.10: Ejemplo de cabeceras que deberían descargarse juntas.

Para evitar este problema, se intenta que sea la propuesta la encargada de organizar la descarga, incluyendo en la petición todo el transporte. Sin embargo, con esta aproximación, el algoritmo

tiene que tratar con la explosión combinatoria resultante, que, como se comentó en la Sección 2.5.2, para 6 vagones de 4 elementos puede alcanzar $1,348 \cdot 10^{16}$ formas de llevar a cabo la descarga.

5.7.2. Problemas encontrados

A pesar de los múltiples intentos de optimización del heurístico para tratar de dirigir estos casos, no se consigue un algoritmo fiable que permita obtener resultados en un tiempo de cálculo reducido.

Se exploró la realización de optimizaciones sobre el cómputo y uso de memoria, con el objetivo de aumentar el número de estados que podían explorarse. Sin embargo, a pesar de que se consiguió llegar a casi dos millones de ejecuciones del heurístico en tres minutos, no fue suficiente para que el algoritmo alcanzase una solución. Los datos de rendimiento obtenido pueden observarse en la Tabla 5.4.

Tabla 5.4: Datos máximo rendimiento propuesta ubicación en 3 min.

Propiedad	Valor
Total vecinos generados	2.823.760
Nodos descartados (no permitidos)	837.446
Nº Ejecuciones Heurístico	1.986.315
Nodos expandidos	7.288

La solución al problema debe cumplir:

- La descarga considera materiales de todo el transporte.
- El tiempo de cálculo se encuentra en un intervalo aceptable (inferior a 30 segundos).

A continuación, se revisa la solución final para obtener la mejor propuesta posible, en un tiempo de cálculo aceptable.

5.7.3. Solución

La estrategia propuesta consiste en transformar el problema de una búsqueda global sobre todos los materiales del transporte, en una secuencia de subproblemas de menor tamaño. Cada subproblema, agrupa materiales según el criterio que usaría la propuesta (programa, destino, tipo o familia) y se consideran las dependencias necesarias para su extracción. La agrupación

favorece aquellos grupos que representan un mayor tamaño en relación al número de materiales que los tapan.

De esta forma, el algoritmo deja de explorar todas las combinaciones posibles de descarga y pasa a seleccionar en base a los conjuntos que usará la propuesta. Con ello, se reduce el tamaño de la búsqueda, manteniendo al mismo tiempo la posibilidad de descargar conjuntamente el mayor número posible de materiales pertenecientes al mismo grupo.

Examinemos el algoritmo propuesto a partir de un ejemplo. En la Figura 5.11 se muestra un ejemplo de un transporte donde se han simplificado los criterios a agrupaciones representadas mediante letras. Los colores representan los grupos de descarga que genera el algoritmo propuesto.

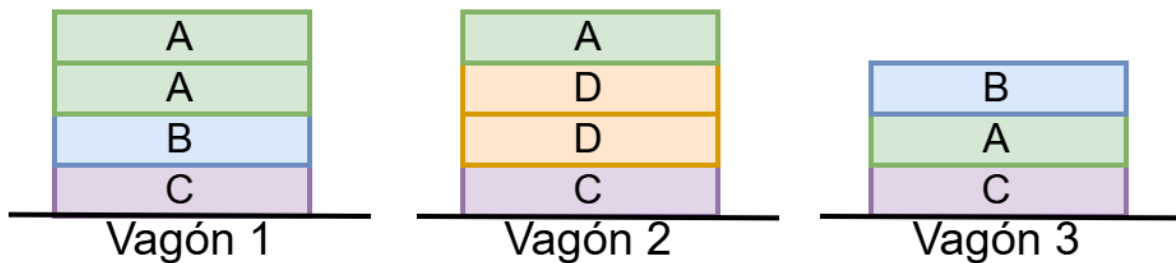


Figura 5.11: Ejemplo de grupos de descarga.

El algoritmo propuesto utiliza los siguientes conceptos:

- **Grupo:** Se asigna a cada material siguiendo el orden de prioridad:
 1. Programa X: Un grupo por cada programa existente.
 2. Destino X: Un grupo por cada destino existente.
 3. Tipo X: Un grupo por cada tipo existente.
 4. Familia X: Un grupo por cada familia existente.
- **Dependencias:** Se considera como dependencia cualquier material que esté situado encima de un desbaste objetivo, independientemente de su grupo. Es una medida de cuántos materiales deben retirarse para permitir su extracción.
- **Lote:** Bloque generado como resultado, incluye las ubicaciones de origen con los materiales que deben extraerse.

El algoritmo realiza una selección iterativa en base al pseudocódigo descrito en el Código 5.4.

```

1 mientras materiales pendientes:
2     candidatos <= materiales accesibles en una ganchada
  
```

```

3   grupos <= agrupar(candidatos)
4   para cada grupo:
5       num materiales <= materiales del grupo
6       dependencias <= materiales que los bloquean
7       ratio <= |num materiales| / |dependencias|
8
9   mejorGrupo <= grupo con mayor ratio
10  crear lote(mejorGrupo + dependencias)
11  eliminar materiales del lote de pendientes

```

Código 5.4: Pseudocódigo algoritmo generación de subproblemas.

A continuación, se revisa la ejecución para el caso de ejemplo presentado en la Figura 5.11. La primera iteración genera los grupos representados en la Tabla 5.5.

Tabla 5.5: Algoritmo generación de subproblemas - Iteración 1.

Grupo	Materiales	Dependencias	Ratio
A	4	2	2
B	2	2	1
C	3	8	0.375
D	2	2	1

En la primera iteración el grupo con un mayor ratio es **A**, nótese que el algoritmo se ve obligado a seleccionar un elemento de B para poder acceder a todos los elementos del grupo A. El nuevo estado del transporte se representa en la Figura 5.12.

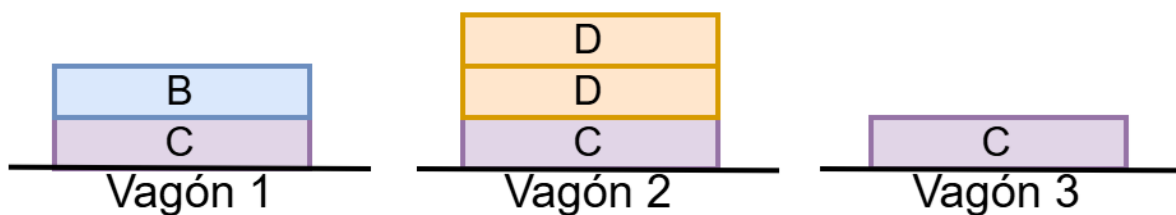


Figura 5.12: Ejemplo de grupos de descarga - Estado tras iteración 1.

La segunda iteración genera los grupos representados en la Tabla 5.6.

Tabla 5.6: Algoritmo generación de subproblemas - Iteración 2.

Grupo	Materiales	Dependencias	Ratio
B	1	0	∞
C	3	3	1
D	2	1	2

En la segunda iteración el grupo extraído es **B** debido a que está marcado como que no tiene dependencias, por lo que se considera más importante. En este caso, el grupo B no ha podido ser considerado completamente, pero aún así, la decisión de la ubicación del segundo B, se realiza nada más que el algoritmo decida el primero, maximizando las posibilidades de juntarlos, si procede. El nuevo estado del transporte se representa en la Figura 5.13.

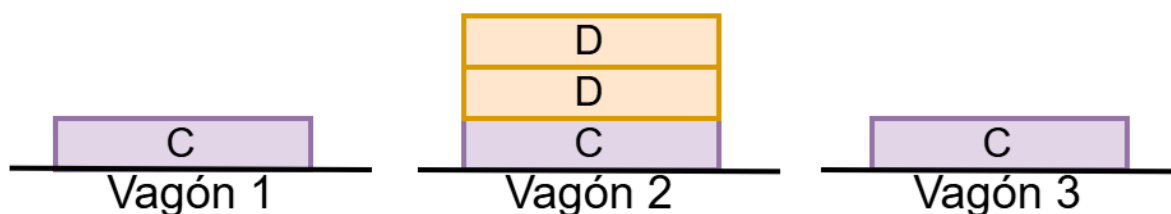


Figura 5.13: Ejemplo de grupos de descarga - Estado tras iteración 2.

La tercera iteración genera los grupos representados en la Tabla 5.7.

Tabla 5.7: Algoritmo generación de subproblemas - Iteración 3.

Grupo	Materiales	Dependencias	Ratio
C	3	2	1.5
D	2	1	2

En la tercera iteración el grupo extraído es **D**, ya que tiene el mejor ratio y desbloquea el siguiente grupo. El nuevo estado del transporte se representa en la Figura 5.14.

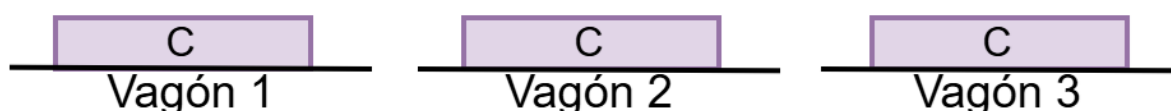


Figura 5.14: Ejemplo de grupos de descarga - Estado tras iteración 3.

La cuarta iteración genera los grupos representados en la Tabla 5.8.

Tabla 5.8: Algoritmo generación de subproblemas - Iteración 4.

Grupo	Materiales	Dependencias	Ratio
C	3	0	∞

Finalmente se selecciona el grupo **C**, ya que es el único que queda.

La propuesta no garantiza la optimalidad global de la secuencia completa de descarga. En su lugar, selecciona iterativamente el grupo con mejor relación beneficio/coste. La reducción de complejidad se consigue por tres motivos:

- Solo se consideran materiales de la parte superior, limitando por la ganchada máxima de la grúa.
- Las propuestas se realizan para cada lote y no sobre un transporte completo, reduciendo el espacio de búsqueda.
- Cada lote reduce el tamaño del problema restante.

En consecuencia, el espacio de búsqueda se reduce al disminuir el número total de materiales de cada propuesta, pasando a depender principalmente del número de grupos y de las dependencias entre pilas.

5.8. Carga de programas

A medida que avanzaba el desarrollo, empezó a resultar evidente que solo estábamos considerando la descarga y reordenación de material en el parque. Sin embargo, una responsabilidad clave del almacén es la que consiste en la ordenación y expedición de material correspondiente a programas de laminación. En este apartado se presenta la solución adoptada para cubrir la necesidad de la planificación de carga de programas.

5.8.1. Idea inicial

Durante la fase de diseño no se había considerado esta necesidad, por lo que la propuesta de ubicación no está pensada para cubrir esta funcionalidad.

La funcionalidad necesaria para realizar la carga de programas debe tener en cuenta los siguientes requisitos:

- Los desbastes deben colocarse en el transporte en orden ascendente, determinado por

su grupo de laminación. Pueden existir huecos únicamente cuando algún grupo no tiene ningún material en el parque.

- Adicionalmente al criterio de orden, se busca cargar el máximo de elementos de programa en el transporte, respetando el orden de grupos.
- El material de programa puede encontrarse enterrado y ser necesario realizar movimientos adicionales para acceder al mismo.
- La carga puede necesitar del uso de pilas, con el objetivo de reordenar los desbastes antes de su carga en el transporte.

En la Figura 5.15 observamos una distribución de grupos de laminación sobre un tren de cuatro vagones. En ella, se muestran los grupos en orden ascendente en un caso en el que todos los grupos se encuentran en el parque.

1	4	7	9
2	5	8	10
3	6	8	11
4	6	8	12
Vagón 1	Vagón 2	Vagón 3	Vagón 4

Figura 5.15: Ejemplo de composición de grupos en carga de vagón.

Los materiales presentados en la Tabla 5.9 sirven de ejemplo de definición de un programa. En ella se observa claramente cómo los grupos se vinculan a uno o más desbastes, que son intercambiables a la hora de cargarse en los vagones.

Tabla 5.9: Ejemplo de composición de grupos en programa.

Secuencia	Grupo	Identificador Material
1	1	260000032
2	2	260000233
3	3	260006035
4	4	260404472
5	4	260007038
6	5	260000039
7	6	260340031
8	6	260345032
9	7	260002037
10	8	260000032
11	8	260050634
12	8	260672033
13	9	260404532
14	10	260055642
15	11	260054654
16	12	260671063
...	...	...

A continuación, exploraremos los problemas detectados al intentar utilizar la propuesta de ubicación para resolver este problema.

5.8.2. Problemas encontrados

Revisando el funcionamiento de la propuesta de ubicación para añadir esta funcionalidad, observamos que tanto los objetivos de la propuesta, como las consideraciones realizadas en su diseño, recogidas en el Capítulo 4, evidencian que fue diseñada para un problema distinto. Por ejemplo, la propuesta de ubicación no se encarga de sugerir reorganizaciones ni de mover materiales que no estén incluidos en la petición de ubicación.

La adaptación de la propuesta para cubrir esta responsabilidad se descarta, debido a los siguientes motivos:

- La composición del destino es conocida al inicio, por lo que no se requiere el cálculo de destinos que lleva a cabo la propuesta de ubicación.
- Puede realizarse más de un movimiento para un desbaste para que alcance el orden objetivo.
- En la reordenación, el material no objetivo no es importante y puede apartarse. Además, pueden moverse materiales que no estén incluidos en la petición para acceder a los que sí.

Debido a lo anterior, nos vemos obligados a diseñar una solución específica para este problema, capaz de proporcionar soporte a la compleja decisión de llevar a cabo la carga de transportes de programa, una de las principales responsabilidades del almacén.

5.8.3. Solución

La solución propuesta consiste en la introducción de un nuevo algoritmo de búsqueda, diseñado específicamente para resolver este problema. En concreto, se plantea una búsqueda de tipo primero el mejor, cuyo objetivo es encontrar la secuencia de carga que requiera el menor número de movimientos.

Para generar una solución, lo primero es calcular, en función de los desbastes presentes en el parque, cómo debe realizarse la carga de grupos. En la Figura 5.16 observamos un ejemplo de objetivo y estado inicial de los desbastes en una pila del parque.

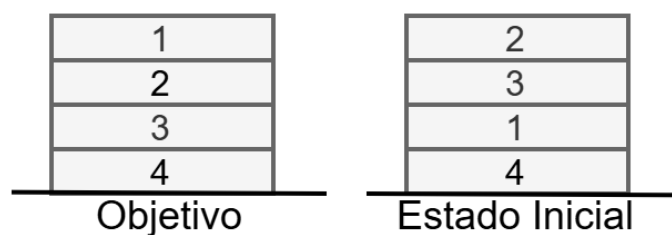


Figura 5.16: Ejemplo objetivo de grupos en carga de programas en un vagón.

Para realizar la carga de estos materiales de forma que se encuentren colocados como dicta el objetivo, han de realizarse los pasos recogidos en la Figura 5.17, en la que se observan los movimientos de los desbastes entre las pilas utilizadas.

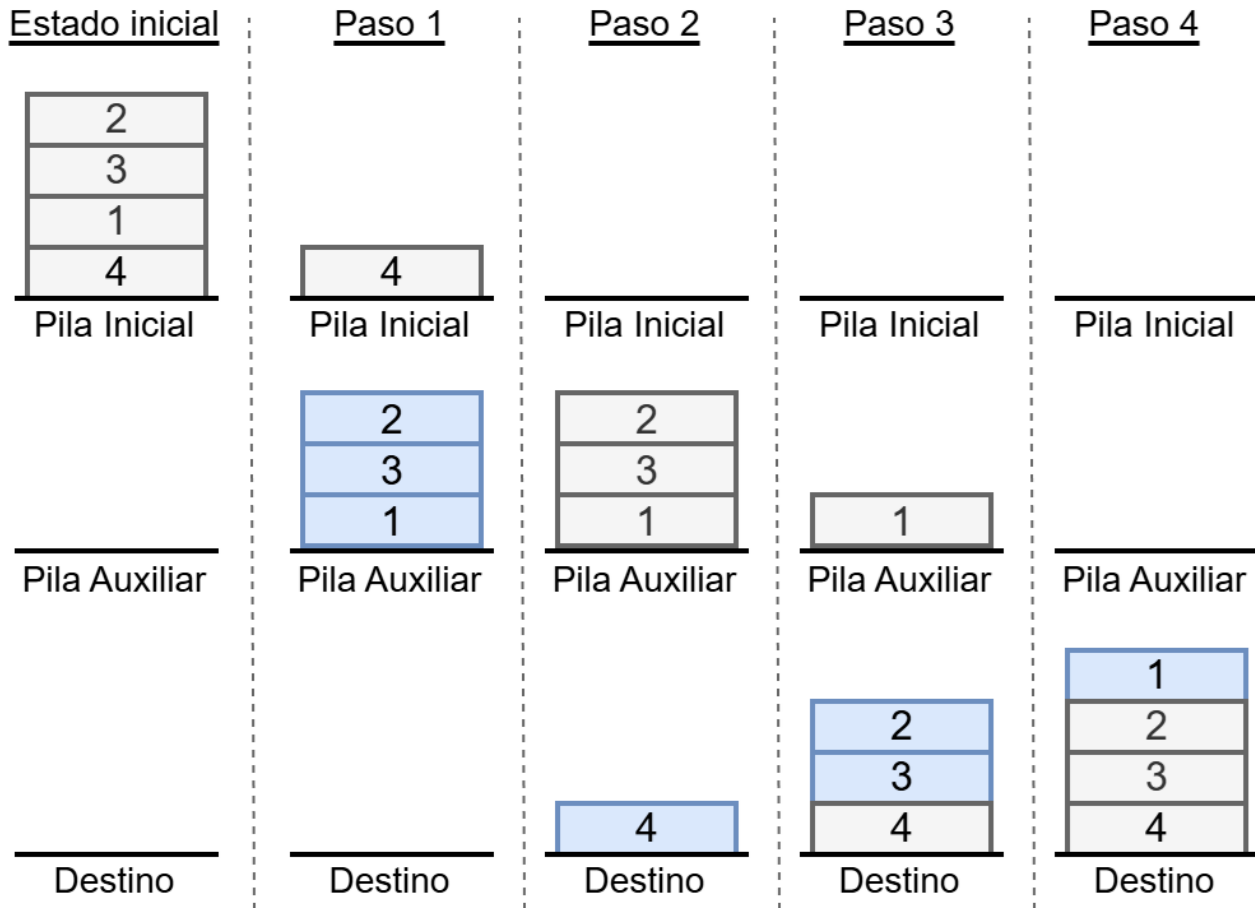


Figura 5.17: Ejemplo evolución de una carga de un vagón.

El algoritmo se diseña para trabajar con un estado parcial del parque, donde únicamente se utilizan las pilas que contienen desbastes de los grupos objetivo. El estado del parque utiliza la siguiente información:

- **Pilas:** Pilas que contienen los grupos objetivo y que constituyen los orígenes del cálculo.
- **Destinos:** Ubicaciones del transporte establecidas como destino final del material. La ejecución del algoritmo finaliza cuando, en estas ubicaciones, se encuentran los grupos objetivo en el orden establecido.
- **Pilas auxiliares:** Pilas adicionales, empleadas como movimiento intermedio, en los casos en los que no es posible realizar los movimientos de ordenación llevando el material directamente al transporte. Estas pilas vacías se reservan en el parque para cumplir con esta función y no se utilizan para el almacenamiento.
- **Objetivo grupos:** Estructura que mantiene la ordenación objetivo de los grupos de laminación y que se utiliza para determinar la idoneidad de los movimientos realizados.

Cada movimiento calculado por el algoritmo se corresponde con una de las siguientes operaciones:

- **A pila objetivo:** Mueve un material o conjunto de materiales a su ubicación objetivo en el destino.
- **A pila auxiliar:** Mueve un material o conjunto de materiales a una pila auxiliar para la realización de un movimiento intermedio de reordenación.
- **Al almacén:** Mueve un material o conjunto de materiales a pilas del almacén. Para determinar las pilas concretas, se usará la propuesta de ubicación. Esta operación únicamente se genera para el material que debe apartarse. El algoritmo no decide su ubicación final, sino que delega esa decisión en la propuesta de ubicación.

Inicialmente se decidió implementar el algoritmo como una búsqueda iterativa en profundidad. Sin embargo, la expansión de movimientos resultó tan grande que este enfoque incurría en un coste demasiado elevado. Como alternativa, se adoptó una búsqueda primero el mejor, guiada mediante una cola de prioridad. A partir del estado inicial, el algoritmo genera iterativamente los movimientos válidos y explora los nodos según la prioridad, obtenida mediante la siguiente fórmula de evaluación:

$$\text{Prioridad} = E_{\text{pendientes}} \cdot P_{\text{actual}} \quad (5.4)$$

donde:

- $E_{\text{pendientes}}$: representa el número de desbastes que faltan por colocar.
- P_{actual} : representa la profundidad alcanzada por la solución, que se corresponde con el número de movimientos realizados.

La idea subyacente a la prioridad propuesta es favorecer estados que hayan colocado muchos materiales utilizando pocos movimientos. Se escogió el producto porque penaliza simultáneamente soluciones profundas y con numerosos materiales pendientes.

La generación de movimientos responsable de expandir un estado de búsqueda, tiene en cuenta las siguientes restricciones:

- Todas las ganchadas generadas han de ser seguras.
- Los movimientos con destino al almacén solo contendrán desbastes que no pertenezcan a los grupos objetivos.
- Las pilas auxiliares solo pueden utilizarse para la ubicación temporal de materiales objetivos.
- Una vez que un material se ubica en destino, no volverá a considerarse su movimiento.

El algoritmo de generación de movimientos expande un nodo siguiendo el pseudocódigo mostrado en el Código 5.5.

```

1 Para cada pila origen
2     Para ganchadas de tamaño 4,3,2,1
3         Si la ganchada es válida
4             Si todos son de grupos objetivo
5                 Generar movimientos hacia pilas destino
6                 Generar movimientos hacia pilas auxiliares
7             Si ninguno pertenece a los grupos objetivo
8                 Generar movimiento al almacén
9
10 Para cada pila auxiliar
11     Para ganchadas de tamaño 4,3,2,1
12         Si la ganchada es válida
13             Generar movimientos hacia pilas destino

```

Código 5.5: Pseudocódigo algoritmo expansión nodos carga de programas.

Este pseudocódigo refleja las restricciones descritas anteriormente y define el flujo de desplazamientos permitido durante la expansión de un estado. Adicionalmente, cuando el algoritmo genera movimientos al almacén que han de ser realizados por la propuesta, el generador de movimientos solo generará la opción más grande posible. Por ejemplo, si hay que apartar tres desbastes consecutivos, no se generarán estados para apartarlos en una combinación de $2 + 1$ movimientos. Esto se debe a que, la responsabilidad de tomar esa decisión ha de recaer en la propuesta de ubicación, este algoritmo solo se encarga de indicar que necesita apartarlos y la propuesta se encargará de definir cómo realizar esa reubicación.

Para controlar el crecimiento del espacio de estados se aplican las siguientes estrategias, que permiten reducir significativamente el número de estados explorados:

- **Control de ciclos:** Se implementa un control de nodos visitados que, a través del uso de una función hash, almacena la firma de cada estado visitado para evitar explorarlo nuevamente.
- **Limitación de la profundidad de búsqueda:** Como estrategia de control del algoritmo se implementa un límite a la profundidad que puede alcanzarse.
- **Poda de estados:** Se implementan estrategias que permiten descartar nodos en cuanto se observa que no van a obtener una buena solución. Estas estrategias se describen a continuación.

5.8.3.1. Poda de composiciones incorrectas

Al detectarse que se han realizado movimientos a un destino que incumplen la ordenación objetivo, hemos de descartar la solución, ya que no es posible que llegue a obtener una solución

válida.

Por ejemplo, ante el estado actual representado en la Figura 5.18, el desbaste de grupo 1 ya se encuentra en una posición que impide que alcancemos los grupos en el orden objetivo.

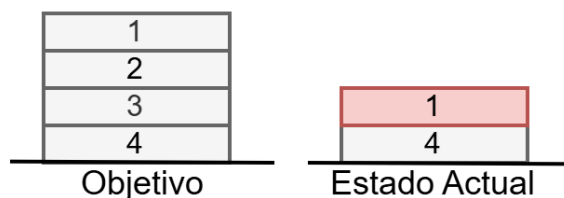


Figura 5.18: Ejemplo poda de composición incorrecta.

Cuando detectamos esta situación, no merece la pena explorar esta secuencia de movimientos y por lo tanto la rama se poda inmediatamente.

5.8.3.2. Poda movimientos redundantes

Cuando se detecta que se están realizando movimientos que son redundantes, se descarta la rama y no se exploran nodos hijos de la misma.

El caso de la Figura 5.19 muestra un movimiento realizado entre pilas auxiliares. Este movimiento no nos acerca a una solución válida, por lo que puede descartarse.

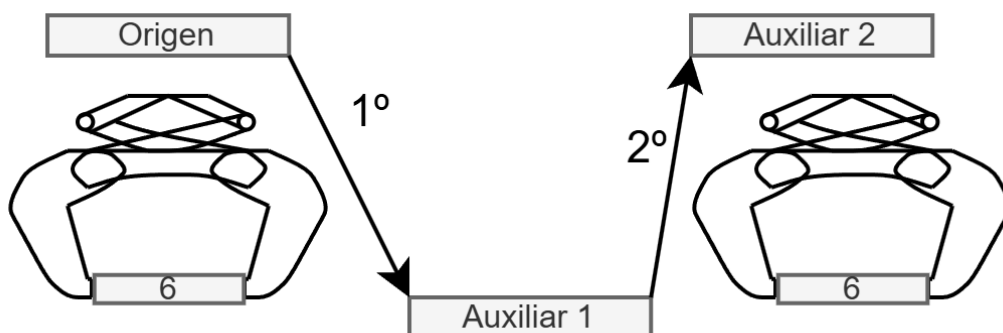


Figura 5.19: Ejemplo movimiento redundante 1.

Otro ejemplo sería el representado en la Figura 5.20, en el que se observa un material que pasa de origen a pila auxiliar y, a continuación, a destino, lo que implica que ya estamos evaluando ese caso en otro estado.

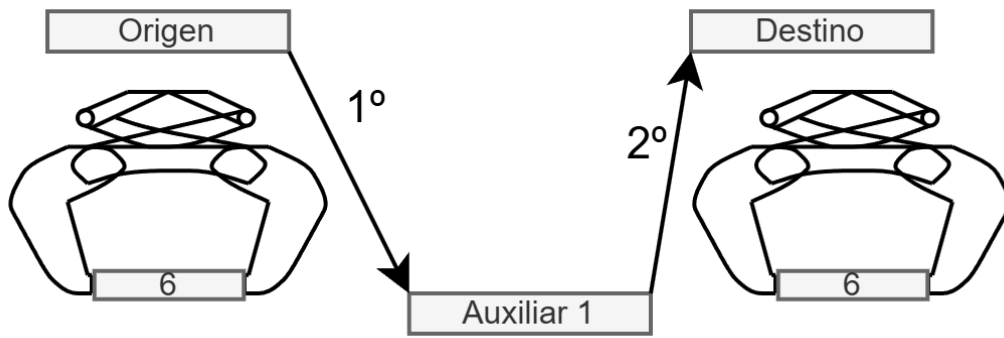


Figura 5.20: Ejemplo movimiento redundante 2.

Por lo tanto, en cuanto se detectan estas situaciones, podemos realizar la poda de ese árbol de estados.

Finalmente, se alcanza el estado final cuando no quedan materiales en las pilas auxiliares y todos los destinos contienen exactamente la secuencia objetivo.

De esta forma, se obtiene una estrategia de generación de movimientos para la carga de programas que, aunque únicamente utiliza la propuesta de ubicación para reubicar el material apartado, permite obtener soluciones de alta calidad, minimizando el número de movimientos.

5.9. Flujo completo de la propuesta de ubicación

Una vez descritos los algoritmos y procesos auxiliares que complementan la propuesta de ubicación, en esta sección se presenta el flujo completo seguido por el sistema en la generación de una propuesta. El funcionamiento final integra tanto el algoritmo principal como los mecanismos auxiliares y optimizaciones desarrollados a lo largo de este capítulo.

El procesamiento de una petición tiene las siguientes etapas:

1. **Recuperación de información:** Antes de iniciar el algoritmo, se obtiene la información necesaria sobre el estado actual del parque, incluyendo los materiales almacenados y sus ubicaciones.
2. **Preparación de las estructuras de la propuesta:** A partir de la configuración de reglas y de la información recuperada en la etapa anterior, se inicializan las estructuras de datos, que representan el estado inicial del parque y que se utilizarán durante la ejecución del algoritmo.
3. **Descomposición en subproblemas:** Cuando la propuesta de ubicación involucra múltiples elementos de origen, se aplica el algoritmo de generación de lotes descrito en la Sección

- 5.7, dividiendo el problema en un conjunto de subproblemas de menor tamaño.
4. **Ejecución del algoritmo:** El algoritmo de búsqueda se ejecuta para cada lote generado, manteniendo el estado del parque compartido entre ejecuciones. De este modo, el estado resultante de un subproblema constituye el estado inicial del siguiente, evitando reconstruir las estructuras del parque y reduciendo el tiempo y coste de ejecución.
 5. **Optimización de distancias:** Una vez obtenida la secuencia de movimientos, se aplica el algoritmo descrito en la Sección 5.6 para minimizar la distancia que será recorrida al ejecutar los movimientos, sin modificar en ningún caso el estado final de la solución.
 6. **Almacenamiento de la solución:** Finalmente, la secuencia optimizada de ganchadas se transforma en elementos «MovementPlanItem», que representan los movimientos que formarán parte del plan de movimientos resultante.

La Figura 5.21 resume gráficamente este proceso, mostrando la secuencia completa, desde la recepción de la petición hasta la generación del plan de movimientos.

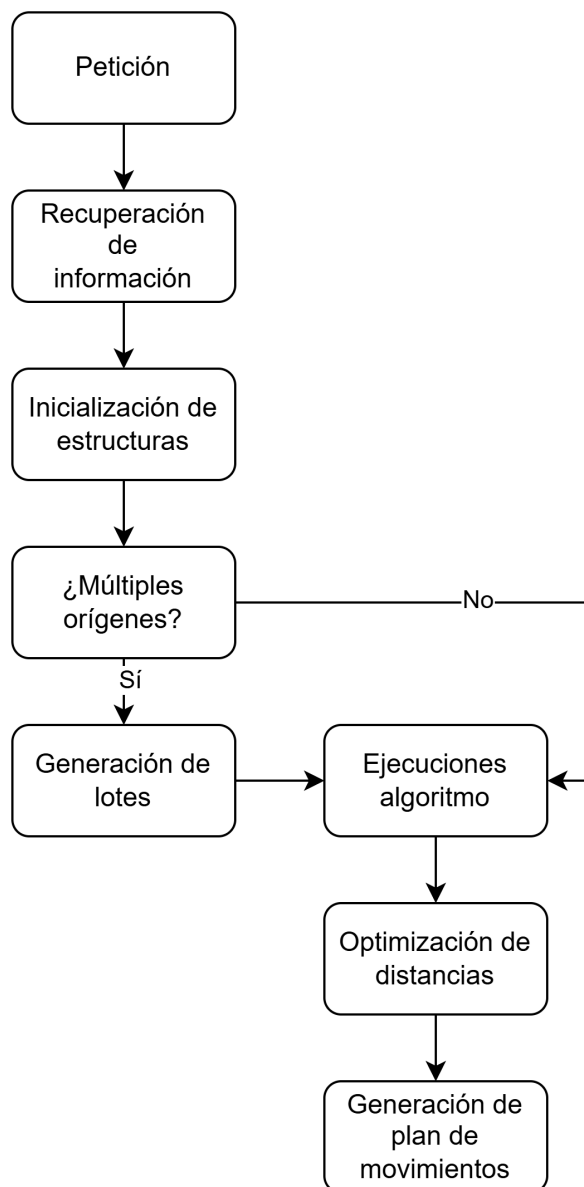


Figura 5.21: Diagrama de flujo ejecución propuesta de ubicación.

El resultado final es un algoritmo con capacidad de adaptación a distintos escenarios, mediante la combinación de técnicas de descomposición, búsqueda y optimización, garantizando la calidad de las soluciones. La integración de los distintos mecanismos descritos a lo largo del capítulo, permite generar propuestas de ubicación de forma eficiente, haciendo viable su utilización como sistema de apoyo a la decisión en un entorno operativo.

5.10. Conclusiones

En este capítulo se ha descrito el proceso de implementación de los principales componentes de la aplicación, particularmente en aquellos en los que ha sido necesario revisar y adaptar el diseño

inicial. Para cada uno de ellos, se ha presentado la idea de partida, los problemas detectados durante el desarrollo y la solución final, justificando las decisiones de diseño correspondientes.

La implementación ha incorporado mecanismos adicionales, que no se habían contemplado en el diseño, como la descomposición en subproblemas para la descarga de trenes, la optimización de la distancia recorrida o la información de diagnóstico de las propuestas de ubicación. La integración de estas estrategias se combina para funcionar como una unidad y resulta lo suficientemente flexible para satisfacer las necesidades del almacén.

Una vez completada la implementación, el siguiente paso consiste en comprobar el funcionamiento de la aplicación y verificar el cumplimiento de los requisitos definidos durante el diseño. Para ello, en el próximo capítulo se presentan las pruebas realizadas sobre la aplicación, comprobando el correcto funcionamiento de sus principales funcionalidades y el cumplimiento de los requisitos establecidos.

Capítulo 6

Pruebas de aplicación

6.1. Introducción

Este capítulo presenta el proceso seguido para la verificación de la aplicación GenStock. El objetivo es comprobar la corrección de las funcionalidades implementadas en la aplicación, dejando para el Capítulo 7 la validación de los sistemas de soporte a la decisión.

Para la validación del sistema se emplean dos niveles de pruebas. El primer nivel, compuesto por pruebas automatizadas que permiten comprobar el comportamiento del código y detectar errores o regresiones durante el desarrollo. El segundo nivel está compuesto por pruebas funcionales, que permiten verificar el comportamiento de la aplicación en los distintos casos de uso definidos. De este modo, se combinan comprobaciones sobre componentes individuales con otras realizadas sobre el sistema en su conjunto.

En primer lugar, se presenta la problemática asociada al problema que ha motivado la verificación mediante pruebas y la estrategia seguida en la implementación de las mismas. A continuación, se describen las pruebas automatizadas y las pruebas funcionales realizadas sobre las principales funcionalidades de GenStock. Finalmente, se presentan las pruebas de los servicios de simulación.

La verificación de los sistemas de soporte a la decisión requiere, además, considerar una problemática diferente, relacionada con la cantidad de situaciones y alternativas que pueden presentarse en la toma de decisiones. Esta cuestión se aborda específicamente en el Capítulo 7.

6.2. Problemática

La validación de las soluciones desarrolladas para el soporte a la decisión sobre un almacén representa un reto, debido a la gran variedad de situaciones posibles. El problema consiste en la asignación de una posición, a cada desbaste a almacenar, dentro de un conjunto de pilas con capacidad limitada. Esta asignación tendrá en cuenta, tanto las características del material como las posibles operaciones de extracción que afectarán al desbaste. En resumen, se trata de optimizar el paso del desbaste a través del parque, minimizando el coste de su almacenamiento

y su extracción.

La distribución de los materiales entre las pilas genera un espacio de estados con una elevada combinatoria. El estado del almacén no queda determinado únicamente por el número de desbastes almacenados, sino también por la pila en la que se encuentra cada uno y por el orden que ocupa dentro de ella. Además, los materiales presentan características diferentes y pueden estar sujetos a distintos criterios de agrupación, almacenamiento o extracción. Como consecuencia, dos estados con un número similar de materiales pueden requerir decisiones completamente diferentes.

A esta variabilidad se añade el carácter dinámico del almacén, donde las decisiones tomadas pueden condicionar las operaciones posteriores, porque una ubicación que resulta adecuada para una operación concreta puede dificultar la realización de operaciones futuras. Asimismo, las necesidades operativas pueden variar a lo largo del tiempo, como consecuencia de cambios de planificación y paradas en las instalaciones productoras o consumidoras.

La consecuencia es que la verificación mediante un conjunto reducido de casos aislados, resulta insuficiente para validar el comportamiento global de la propuesta. Una determinada regla o estrategia puede proporcionar buenos resultados en un escenario y comportarse de forma diferente en otro. Por lo tanto, no basta con comprobar que el algoritmo encuentra una solución válida en una configuración, sino que es necesario comprobar su comportamiento ante diferentes distribuciones de material, niveles de ocupación y estados del almacén.

Por estos motivos, no resulta viable realizar una verificación completa de todas las configuraciones posibles. En su lugar, se ha empleado una combinación de diferentes estrategias de prueba, que permiten evaluar el sistema desde distintos puntos de vista:

- Pruebas automatizadas unitarias, de integración y funcionales, para comprobar la implementación de las reglas y el comportamiento de los diferentes componentes.
- Pruebas sobre estados aleatorizados del almacén, en los que los materiales presentan diferentes distribuciones entre las pilas y no necesariamente siguen la propuesta de ubicación.
- Pruebas de llenado, partiendo de un almacén vacío y realizando sucesivas recepciones de material y analizando el comportamiento, a medida que aumenta el número de desbastes almacenados.
- Pruebas sobre un almacén gestionado mediante el soporte a la decisión, permitiendo observar el efecto acumulado de las decisiones de entrada y salida, sobre el estado del almacén.

La combinación de estas estrategias permite comprobar la corrección del dominio, las reglas y evaluar el comportamiento de la aplicación en diferentes estados del almacén.

No obstante, debido al elevado número de escenarios posibles, no se puede afirmar que todas las situaciones hayan sido cubiertas. La validación realizada busca, por tanto, proporcionar evidencias sobre el comportamiento del sistema, mediante un conjunto representativo de casos y escenarios, en lugar de intentar demostrar exhaustivamente su comportamiento para todas las configuraciones posibles.

La verificación específica de los mecanismos de soporte a la decisión se aborda en el Capítulo 7 mediante escenarios diseñados para analizar el comportamiento de las soluciones generadas, ante diferentes condiciones del almacén. Esta verificación, se centra en los resultados obtenidos y en su coherencia con los criterios establecidos, sin limitarse a la comprobación de la implementación interna de los algoritmos.

6.3. Pruebas automatizadas

Con el objetivo de verificar el funcionamiento de la aplicación y la propuesta de ubicación, se incorporó una batería de pruebas automatizadas utilizando el framework xUnit para .NET.

Las pruebas se organizaron en tres proyectos independientes, según el tipo de prueba realizada:

- **Pruebas unitarias.** Verifican de forma aislada el comportamiento de las clases que implementan la lógica de negocio, incluyendo clases de dominio, reglas de validación, reglas de penalización, heurísticos, evaluadores de seguridad y otros componentes.
- **Pruebas de integración.** Comprueban la correcta interacción entre varios componentes de la aplicación, verificando especialmente el acceso a la base de datos y la persistencia de información.
- **Pruebas funcionales.** Evalúan el comportamiento de la API desde el punto de vista del usuario, ejecutando peticiones HTTP reales sobre una instancia de la aplicación, para comprobar que los casos de uso principales producen los resultados esperados.

La Figura 6.1 muestra el resultado de la ejecución completa de la batería de pruebas desarrollada.

Test	Duration
>  GenStock.FunctionalTests (34)	31,7 sec
>  GenStock.IntegrationTests (3)	4,7 sec
>  GenStock.UnitTests (400)	9,4 sec

Figura 6.1: Resultados ejecución pruebas automáticas.

En la Tabla 6.1 se observa la distribución de las 437 pruebas automatizadas desarrolladas para

la aplicación.

Tabla 6.1: Distribución de las pruebas automatizadas.

Tipo	Nº Pruebas
Unitarias	400
Integración	3
Funcionales	34
Total	437

El reducido número de pruebas de integración se debe a que la mayor parte del comportamiento puede validarse mediante pruebas unitarias o pruebas funcionales. Las pruebas de integración existentes verifican la integración de los repositorios de base de datos.

Las pruebas unitarias constituyen la mayor parte del conjunto de pruebas debido a la naturaleza del proyecto, donde gran parte de la complejidad reside en la implementación de algoritmos y reglas de evaluación. Este tipo de pruebas permitió verificar, de forma independiente, el comportamiento de cada componente, simplificando el desarrollo y la integración.

Las pruebas de integración y funcionales complementan esta verificación, asegurando que los distintos componentes colaboran correctamente y que los principales casos de uso de la aplicación funcionan de forma conjunta.

Las pruebas unitarias resultaron clave durante el desarrollo de la propuesta de ubicación. Permitieron validar de forma aislada cada regla, antes de integrarla en el algoritmo completo, y facilitaron la corrección de errores mediante la incorporación de nuevos casos de prueba.

En la validación de la propuesta se emplearon dos aproximaciones. La primera consiste en pruebas de caso, donde se verifica que un escenario concreto produce la penalización esperada. La segunda corresponde a pruebas comparativas, en las que se evalúan dos escenarios distintos y se comprueba que aquel considerado más favorable obtiene una penalización inferior.

En el Código 6.1 se muestra una prueba de caso, concretamente una utilizada para evaluar la ubicación de dos desbastes de distinto programa sobre una pila vacía.

```

1 [Fact]
2 public void
   Evaluate_WhenEmptyPile2ProgramsInMove_ReturnsExpectedPenalty()
3 {
4     // Arrange
5     var rule = new ProgramRule() { MixedProgramsInMoveBasePenalty =
      2000 };
6     var material1 = new MaterialBuilder()

```

```

7         .WithSecuencialId(2)
8         .WithProgramItem(x =>
9             x.WithPriority(1)
10            .WithSequence(1)
11            .WithGroup(1)
12        ).Build();
13    var material2 = new MaterialBuilder()
14        .WithSecuencialId(3)
15        .WithProgramItem(x =>
16            x.WithPriority(2)
17            .WithSequence(1)
18            .WithGroup(1)
19        ).Build();
20    var pile = new PileBuilder().Build();
21    // Act
22    var penalty = rule.EvaluatePenalty(pile, [material1, material2]);
23    // Assert
24    penalty.ShouldBe(6800);
25 }

```

Código 6.1: Ejemplo prueba de caso.

Este tipo de prueba resulta especialmente útil cuando la penalización esperada puede calcularse manualmente a partir de la definición de la regla. Por el contrario, en el caso de la prueba comparativa definida en el Código 6.2, se definen dos escenarios que son evaluados por la misma regla para comparar sus resultados.

```

1 [Fact]
2 public void
3     Evaluate_WhenProgramIsCoveredBy5NonProgram_PenalizeMoreThan4()
4 {
5     // Arrange
6     var rule = new ProgramRule();
7     var pileWith4 = new PileBuilder()
8         .AddOnTop(m => m
9             .WithSecuencialId(40)
10            .WithProgramItem(x =>
11                x.WithPriority(9)
12                .WithSequence(1)
13                .WithGroup(24)
14            )
15        )
16        .AddOnTop(m => m.WithSecuencialId(41))
17        .AddOnTop(m => m.WithSecuencialId(42))
18        .AddOnTop(m => m.WithSecuencialId(43))
19        .AddOnTop(m => m.WithSecuencialId(44))

```

```

19         .Build();
20     var pileWith5 = new PileBuilder()
21         .AddOnTop(m => m
22             .WithSecuencialId(50)
23             .WithProgramItem(x =>
24                 x.WithPriority(9)
25                 .WithSequence(1)
26                 .WithGroup(24)
27             )
28         )
29         .AddOnTop(m => m.WithSecuencialId(51))
30         .AddOnTop(m => m.WithSecuencialId(52))
31         .AddOnTop(m => m.WithSecuencialId(53))
32         .AddOnTop(m => m.WithSecuencialId(54))
33         .AddOnTop(m => m.WithSecuencialId(55))
34         .Build();
35     var move = new MaterialBuilder()
36         .WithSecuencialId(56)
37         .WithProgramItem(x =>
38             x.WithPriority(9)
39             .WithSequence(1)
40             .WithGroup(18)
41         ).Build();
42     // Act
43     var penaltyWith4 = rule.EvaluatePenalty(pileWith4, [move]);
44     var penaltyWith5 = rule.EvaluatePenalty(pileWith5, [move]);
45     // Assert
46     penaltyWith5.ShouldBeGreaterThan(penaltyWith4);
47 }

```

Código 6.2: Ejemplo prueba comparativa.

Este enfoque, especialmente útil en las reglas de penalización, reduce el acoplamiento con los valores numéricos concretos de la penalización y verifica directamente el comportamiento esperado.

Para el desarrollo de todas las pruebas de la aplicación se han seguido las siguientes convenciones:

- **Estructura de la prueba:** Se utiliza la siguiente organización del código:
 - Arrange: Código para la configuración de la prueba.
 - Act: Ejecución del código que queremos probar.
 - Assert: Verificación de los resultados.
- **Nomenclatura:** *NombreMétodo_WhenSituación_ResultadoEsperado*, el objetivo de la mis-

ma es que el nombre de cada prueba sirva como documentación de la verificación que lleva a cabo.

- **Builders:** Para la construcción de los objetos a probar se han creado clases auxiliares que siguen el patrón «Builder» para permitir una configuración legible. Estas clases cuentan con valores por defecto que permiten centrarse, en cada caso, en los atributos que son relevantes a la prueba.
- **Aserciones:** Se ha utilizado la librería Shouldly, debido a la legibilidad que aporta a las aserciones con estilo funcional.

En conjunto, esta batería de pruebas permitió validar el comportamiento individual de los distintos componentes y su integración dentro de la aplicación. Una vez verificada la corrección de la implementación, en los siguientes apartados se lleva a cabo una verificación de las especificaciones establecidas en la fase de diseño.

6.4. Pruebas funcionales de la aplicación

En esta sección se recogen las pruebas funcionalidades que se han realizado sobre la parte visible de la aplicación, para garantizar que su funcionamiento cumple con las especificaciones que se han definido en el diseño.

6.4.1. Estadísticas

En este apartado se recogen las pruebas funcionales realizadas sobre la sección de estadísticas.

6.4.1.1. Visualización de estadísticas del almacén

Esta prueba verifica la especificación *EST-01*, correspondiente a la visualización de estadísticas del almacén.

Para realizar la prueba se verifica que, sobre un estado del parque, se muestra la información correspondiente al estado actual, permitiendo observar aspectos como la evolución del número de toneladas o la distribución de tipos.

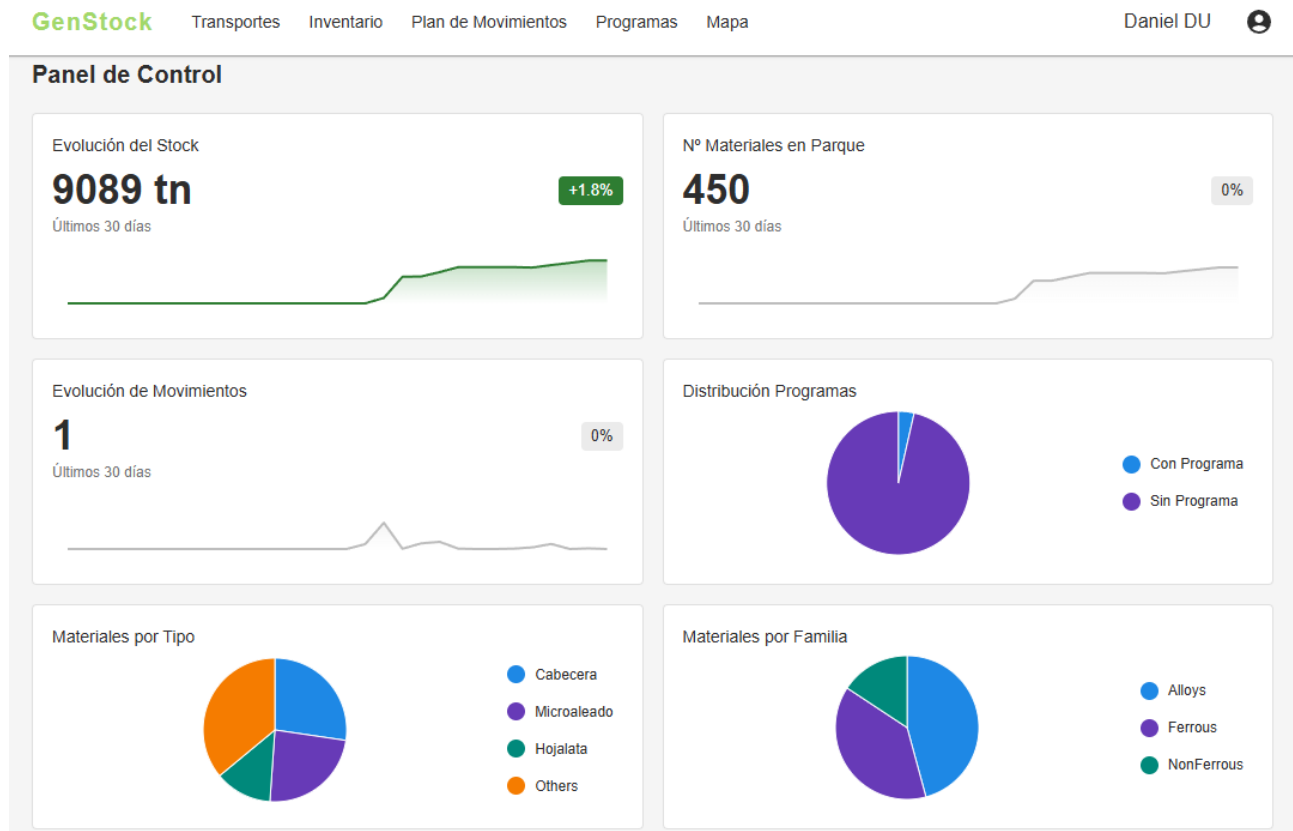


Figura 6.2: Verificación de estadísticas del almacén.

Se comprueba que los valores mostrados coinciden con la información del estado actual, verificando el cumplimiento de la especificación *EST-01*.

6.4.2. Transportes

A continuación, se describen las pruebas funcionales realizadas sobre la sección de transportes.

6.4.2.1. Entrada de un transporte

Esta prueba verifica la especificación *TRN-02*, correspondiente a la funcionalidad de entrada de un transporte en el almacén.

Para realizar la prueba se accede a la sección de transportes, se localiza un transporte que no se encuentre en el parque, utilizando para ello el listado que se observa en la Figura 6.3.

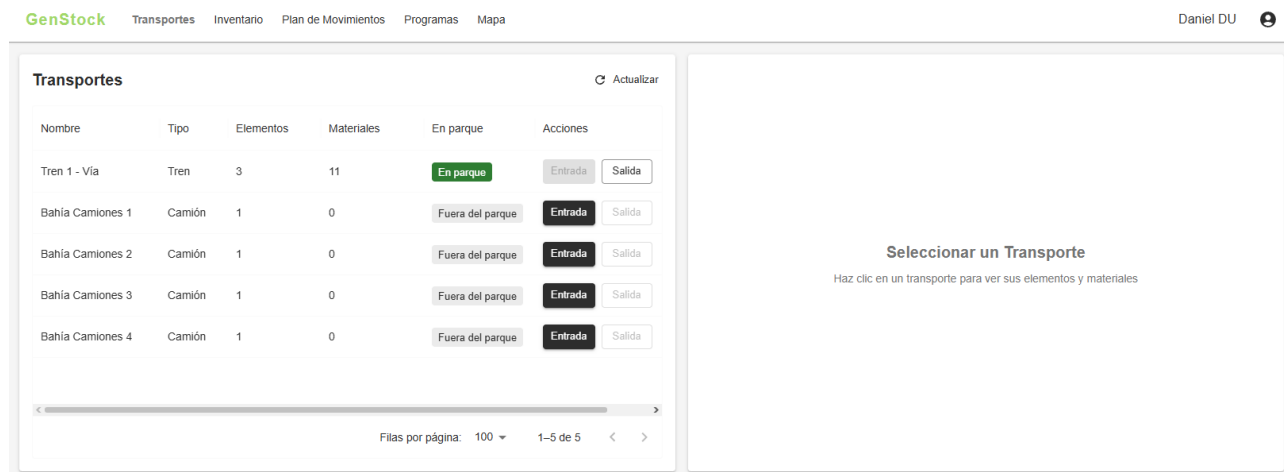


Figura 6.3: Estado anterior a la entrada del transporte.

En este caso, seleccionamos la última fila, correspondiente a «Bahía Camiones 4», pulsamos sobre el botón «Entrada» y se muestra el diálogo de la Figura 6.4 para elegir si el transporte incluirá materiales o no.

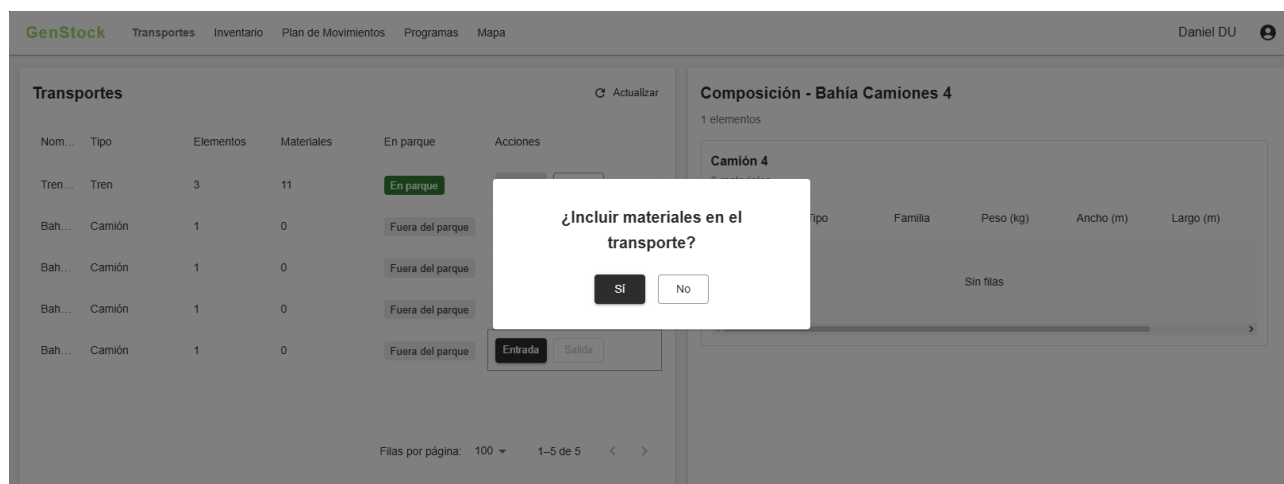


Figura 6.4: Elección de materiales en entrada del transporte.

En este caso, seleccionamos que se incluyan los materiales y se obtiene la confirmación de que la operación se ha realizado, como podemos observar en la Figura 6.5. El estado del transporte aparece ahora modificado, mostrándose con el estado «En el parque». Además, en el panel derecho de detalle de la composición, se visualiza la lista de los materiales que contiene.

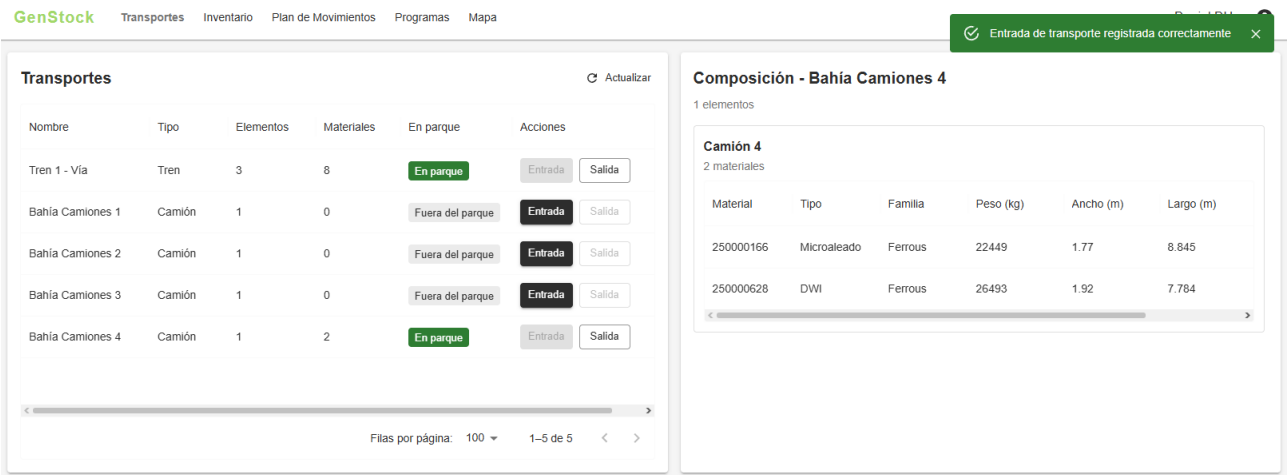


Figura 6.5: Estado posterior a la entrada del transporte.

Adicionalmente, se comprueba el estado en el mapa de la aplicación y la Figura 6.6 permite observar la aparición del transporte.

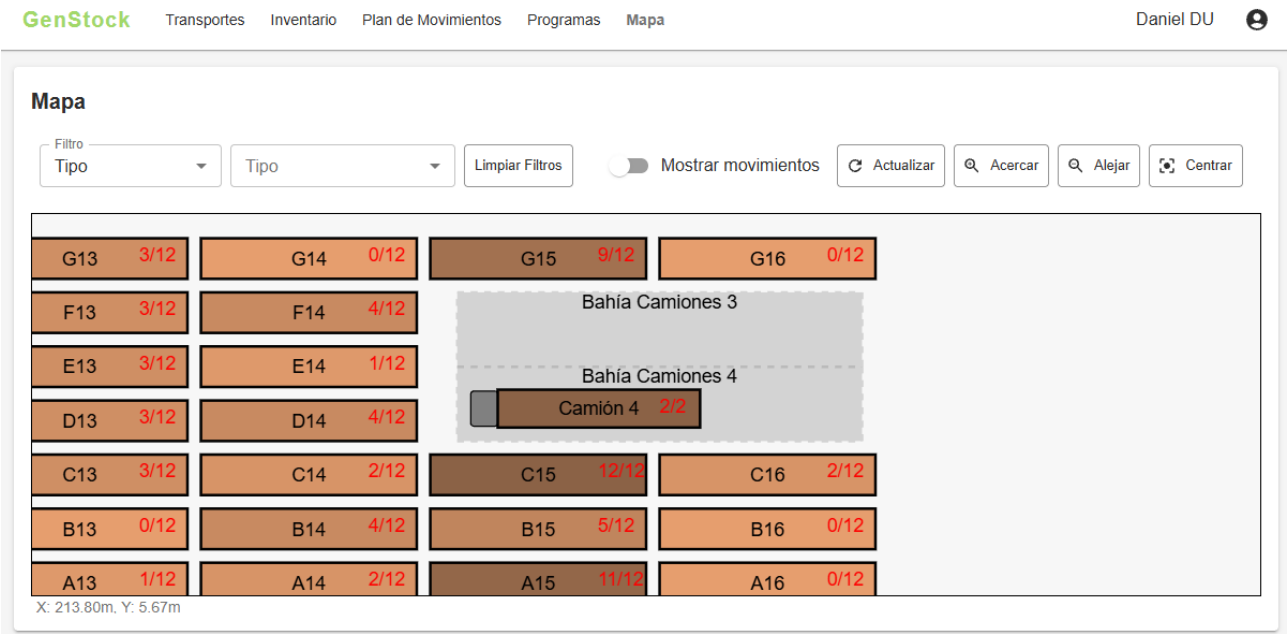


Figura 6.6: Estado mapa posterior a la entrada del transporte.

El resultado obtenido permite verificar la especificación *TRN-02* y además las funcionalidades *TRN-01* y *TRN-04* correspondientes, respectivamente, al funcionamiento del listado de transportes y la consulta del detalle de un transporte.

6.4.2.2. Salida de un transporte

Esta prueba verifica la especificación *TRN-03*, correspondiente a la funcionalidad de salida de un transporte del almacén.

Para realizar la prueba se accede a la sección de transportes, se localiza un transporte que se encuentre en el parque, utilizando para ello el listado que se observa en la Figura 6.7.

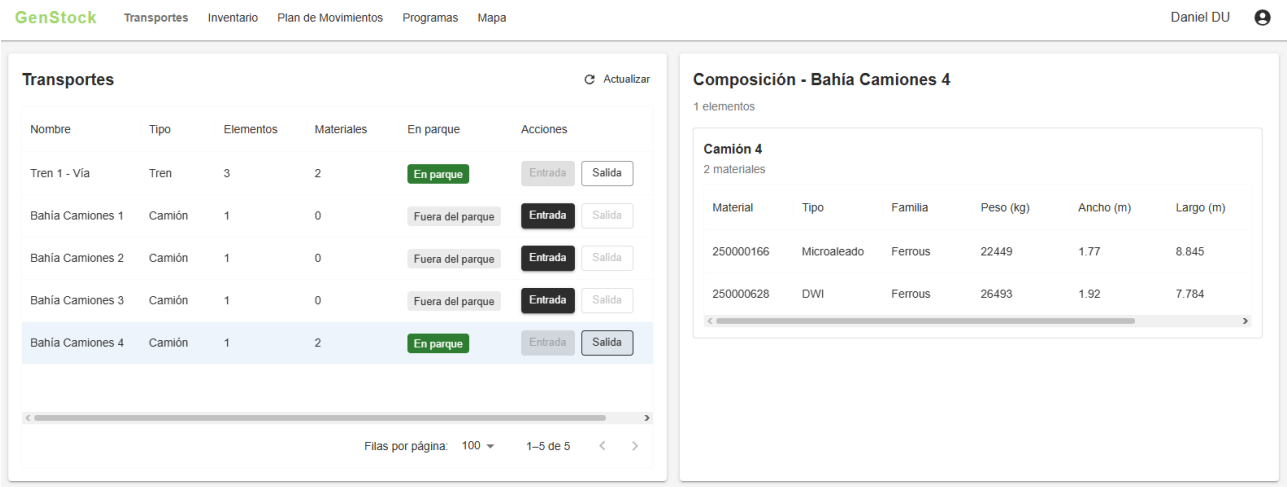


Figura 6.7: Estado anterior a la salida del transporte.

En este caso, seleccionamos la última fila correspondiente a «Bahía Camiones 4», pulsamos sobre el botón «Salida» y se muestra la confirmación de que la operación ha sido realizada, como podemos observar en la Figura 6.8. El estado del transporte aparece ahora modificado, mostrándose con el estado «Fuera del parque». Además, en el panel derecho de detalle de la composición, se observa que ya no contiene materiales.

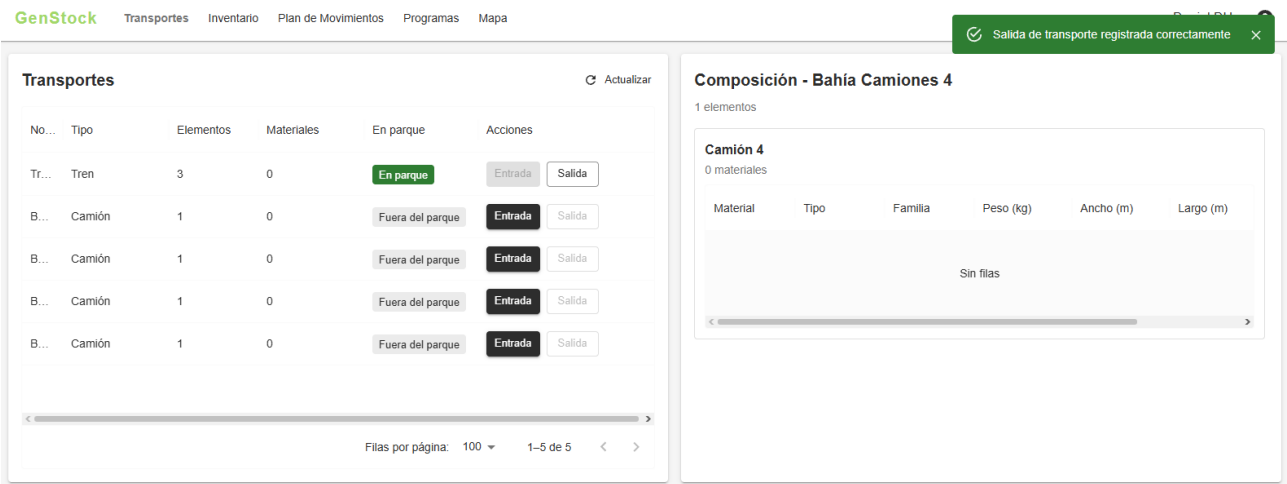


Figura 6.8: Estado posterior a la salida del transporte.

Adicionalmente, se comprueba el estado en el mapa de la aplicación y la Figura 6.9 permite observar que el transporte ya no se encuentra presente.

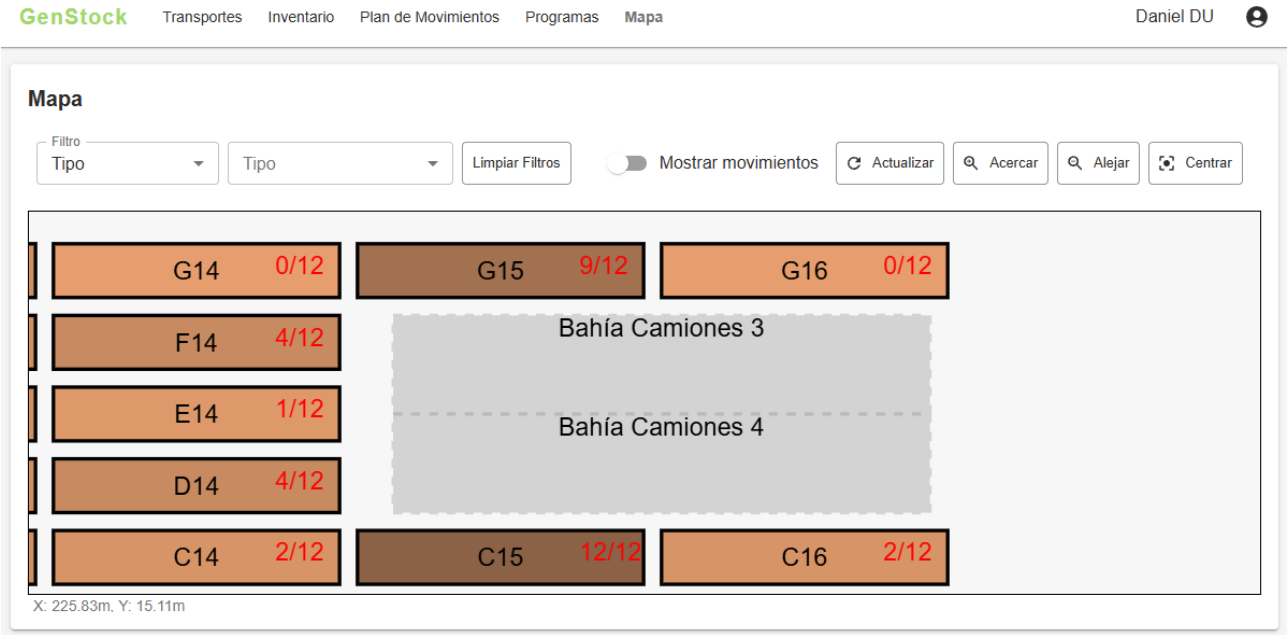


Figura 6.9: Estado mapa posterior a la salida del transporte.

El resultado obtenido permite verificar la especificación *TRN-03* y además las funcionalidades *TRN-01* y *TRN-04* correspondientes, respectivamente, al funcionamiento del listado de transportes y la consulta del detalle de un transporte.

6.4.3. Inventario

En este apartado se describen las pruebas realizadas para la verificación de la sección de inventario.

6.4.3.1. Listado de materiales

Esta prueba verifica el listado de información de los materiales del almacén, recogido en la especificación *INV-01* y la *INV-03* correspondiente a la consulta del detalle de un material.

Para realizar la prueba se accede a la sección de inventario y se observa que se muestra un listado de información, con los materiales que se encuentran en el almacén. El listado incluye la información para identificar el desbaste y su ubicación en el almacén. Al seleccionar una de las filas, en este caso la correspondiente al material 250000002, se muestra el detalle del material en el panel derecho. Podemos observar el resultado de las acciones anteriores en la Figura 6.10.

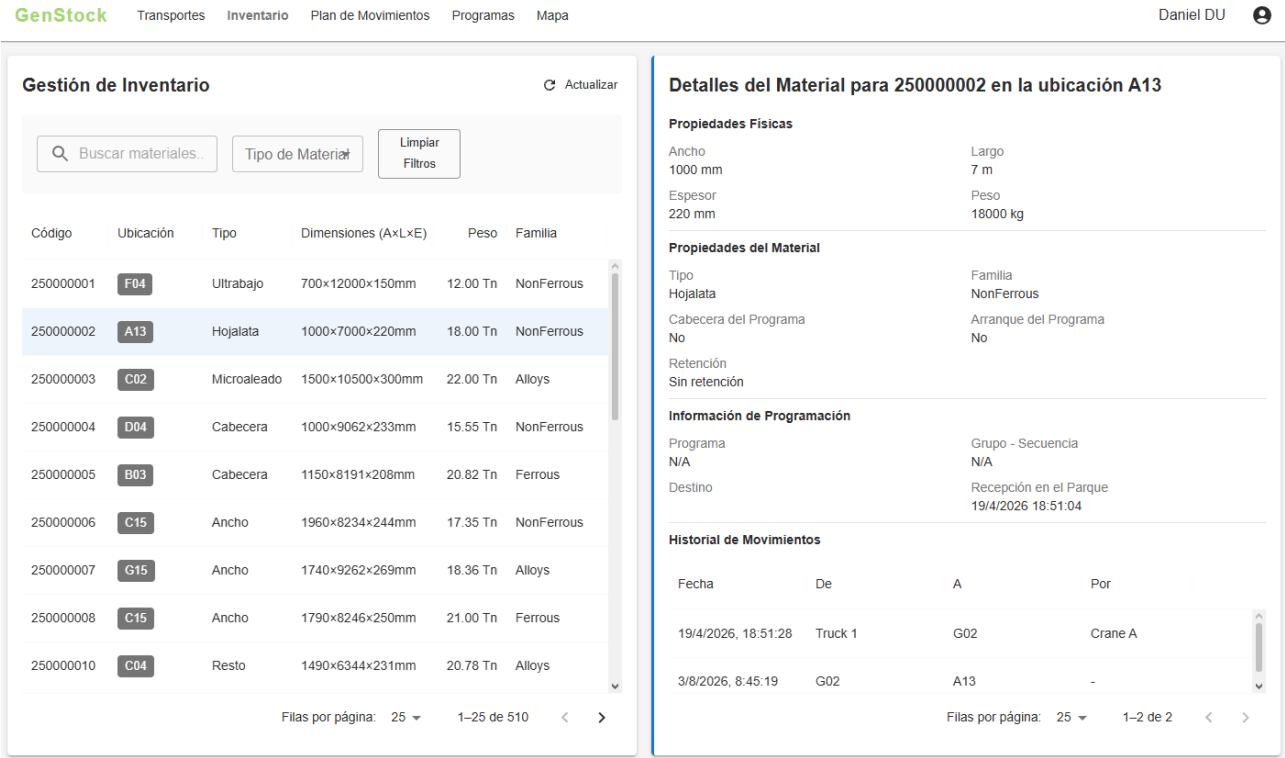


Figura 6.10: Inventario con material seleccionado.

Se aprecia que la información de detalle expande no solo la información propia del material, sino que también se puede consultar los movimientos en los que el material ha estado involucrado. El resultado obtenido permite verificar las especificaciones *INV-01* y *INV-03*.

6.4.3.2. Búsqueda y filtrado

Esta prueba verifica la especificación *INV-02* correspondiente a la búsqueda y filtrado de materiales.

Para realizar la prueba se accede a la sección de inventario y se selecciona un tipo de material en el desplegable superior. Los resultados de esta acción se muestran en la Figura 6.11.

Gestión de Inventario

Actualizar

Buscar materiales...

Tipo de Material
CalidadZ

Limpiar Filtros

Código	Ubicación	Tipo	Dimensiones (A×L×E)	Peso	Familia
250000013	G07	CalidadZ	1680×9227×184mm	18.61 Tn	Alloys
250000032	G07	CalidadZ	1650×7700×185mm	16.64 Tn	NonFerrous
250000072	F08	CalidadZ	1840×5600×188mm	16.11 Tn	Alloys
260000343	F12	CalidadZ	2050×8668×211mm	17.77 Tn	Ferrous

Filas por página: 25 1–4 de 4

Figura 6.11: Inventario con filtro de material.

La prueba continúa introduciendo «003» como criterio de búsqueda, en la Figura 6.12 se observa el resultado de la acción y se observa que se muestran los materiales cuyo identificador contiene la búsqueda introducida.

Gestión de Inventario

🔍 003

Tipo de Material

CalidadZ

Limpiar Filtros

Actualizar

Código	Ubicación	Tipo	Dimensiones (A×L×E)	Peso	Familia
250000032	G07	CalidadZ	1650×7700×185mm	16.64 Tn	NonFerrous
260000343	F12	CalidadZ	2050×8668×211mm	17.77 Tn	Ferrous

Filas por página: 25

1-2 de 2

< >

Figura 6.12: Inventario con material seleccionado y búsqueda.

El resultado que se ha obtenido permite verificar la especificación *INV-02*.

6.4.4. Planes de movimiento

En esta sección se comprueba la funcionalidad relacionada con la gestión de los planes de movimiento del almacén. A continuación, se detallan las pruebas funcionales realizadas para verificar su correcto funcionamiento.

6.4.4.1. Alta de un plan para descarga de transporte

Esta prueba verifica la especificación *PMO-02*, correspondiente al alta de un plan de movimiento para la descarga de un transporte.

Para realizar la prueba se accede a la sección de planes de movimiento y se selecciona la opción para crear un nuevo plan. Se selecciona la operación de descarga de transporte y se introducen los datos necesarios para generar el plan de movimiento. La Figura 6.13 muestra el estado del

formulario antes del guardado.

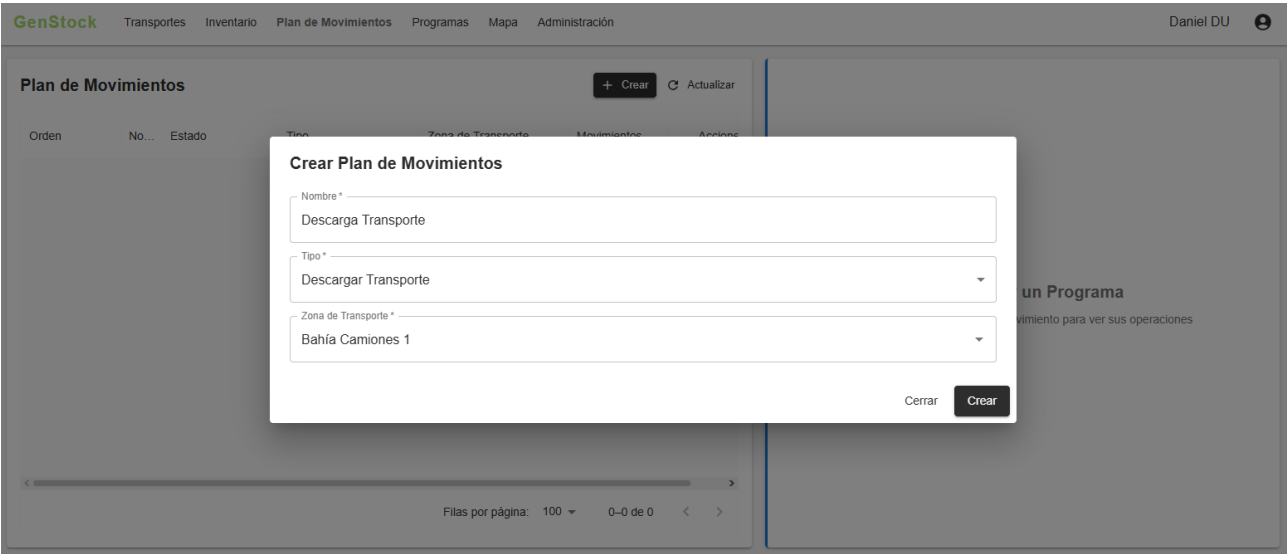


Figura 6.13: Alta de un plan de movimiento para descarga de transporte.

Tras confirmar la operación, el plan de movimiento se genera correctamente y, como se observa en la Figura 6.14, aparece en el listado de planes.

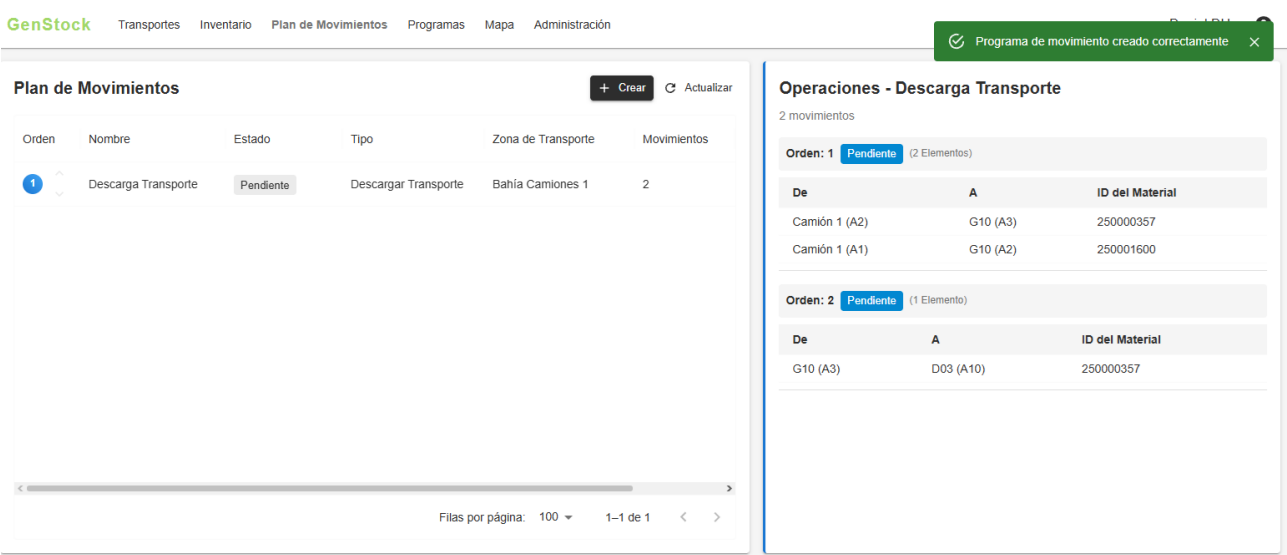


Figura 6.14: Listado tras el alta del plan de descarga de transporte.

El resultado obtenido permite verificar el cumplimiento de la especificación *PMO-02* y, a su vez, las especificaciones *PMO-01* y *PMO-06* correspondientes, respectivamente, al listado de planes de movimiento y a la consulta del detalle del plan.

6.4.4.2. Alta de un plan para liberación de pila

Esta prueba verifica la especificación *PMO-03*, correspondiente al alta de un plan de movimiento para la liberación de una pila.

Para realizar la prueba se accede a la sección de planes de movimiento y se selecciona la opción para crear un nuevo plan. Se selecciona la operación de liberación de pila y se introducen los datos necesarios para generar el plan de movimiento. La Figura 6.15 muestra el estado del formulario antes de guardar el plan.

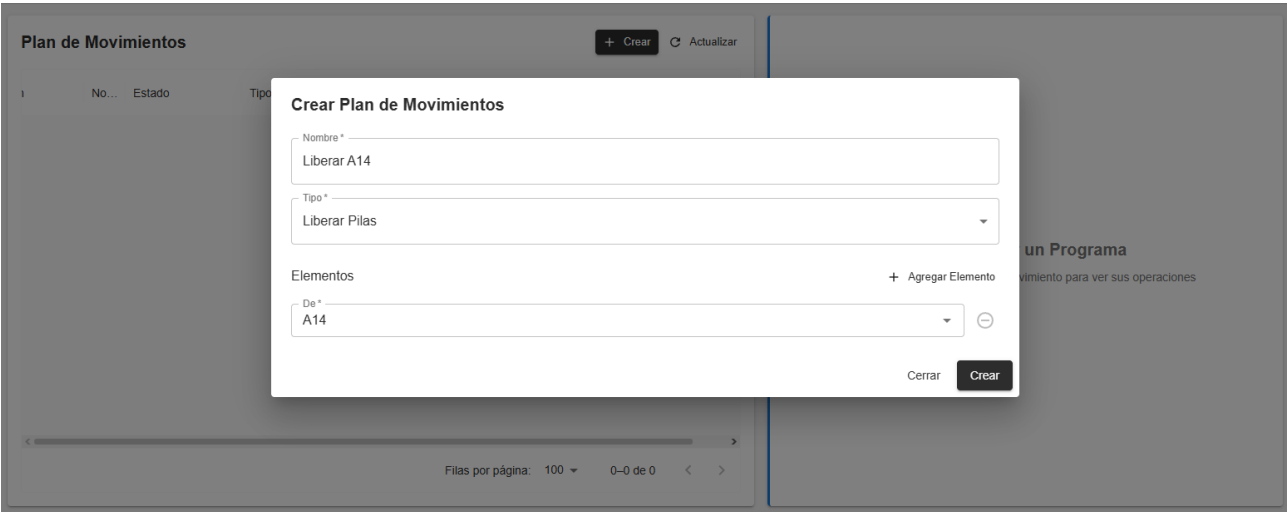


Figura 6.15: Alta de un plan de movimiento para liberación de pila.

Tras confirmar la operación, el plan de movimiento se genera correctamente y, como se observa en la Figura 6.16, aparece en el listado de planes.

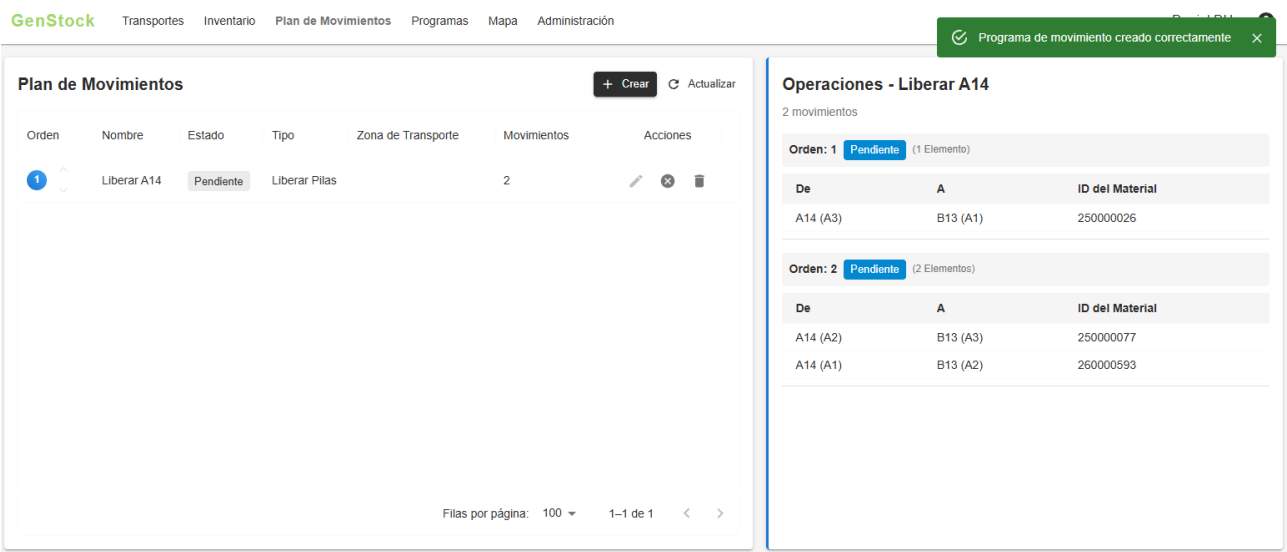


Figura 6.16: Listado tras el alta del plan de liberación de pila.

El resultado obtenido permite verificar el cumplimiento de la especificación *PMO-03*.

6.4.4.3. Alta de un plan para carga de programa

Esta prueba verifica la especificación *PMO-04*, correspondiente al alta de un plan de movimiento para la carga de un programa.

Para realizar la prueba se accede a la sección de planes de movimiento y se selecciona la opción para crear un nuevo plan. Se selecciona la operación de carga de programa y se introducen los datos necesarios para generar el plan de movimiento. La Figura 6.17 muestra el estado del formulario antes de guardar el plan.

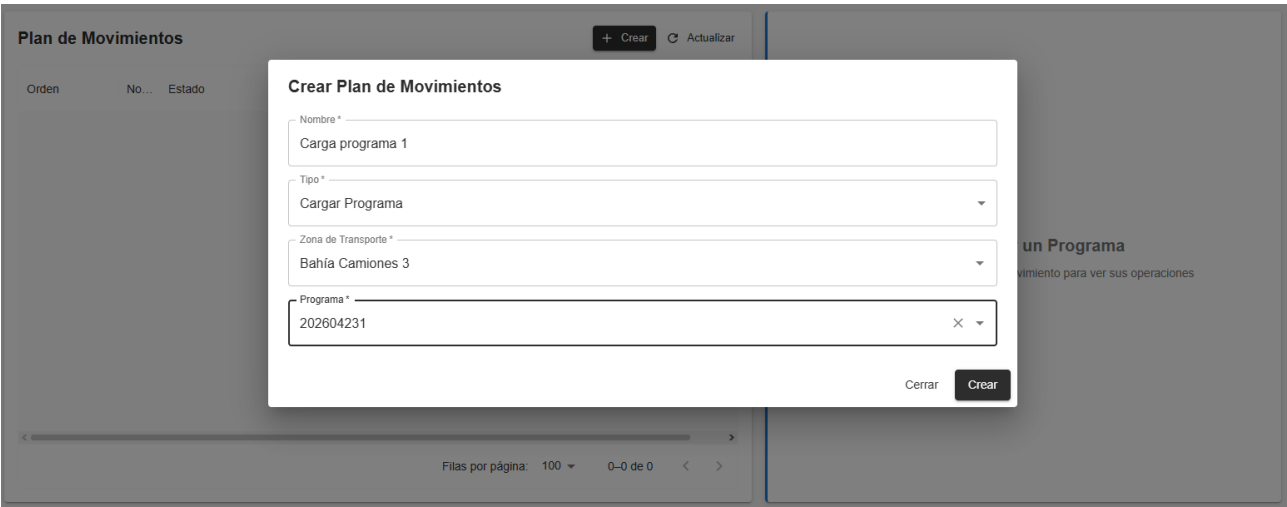


Figura 6.17: Alta de un plan de movimiento para carga de programa.

Tras confirmar la operación, el plan de movimiento se genera correctamente y, como se observa en la Figura 6.18, aparece en el listado de planes.

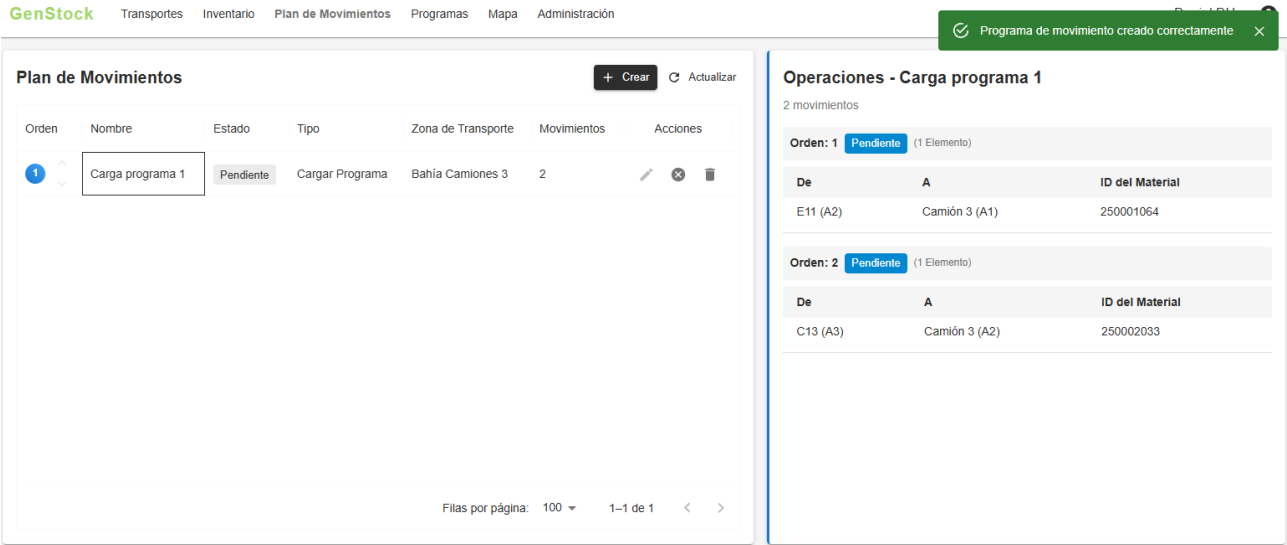


Figura 6.18: Listado tras el alta del plan de carga de programa.

El resultado obtenido permite verificar el cumplimiento de la especificación *PMO-04*.

6.4.4.4. Alta de un plan para reubicación

Esta prueba verifica la especificación *PMO-05*, correspondiente al alta de un plan de movimiento para la reubicación de material.

Para realizar la prueba se accede a la sección de planes de movimiento y se selecciona la opción para crear un nuevo plan. Se selecciona la operación de reubicación y se introducen los datos necesarios para generar el plan de movimiento. La Figura 6.19 muestra el estado del formulario antes de guardar el plan.

Plan de Movimientos

+ Crear Actualizar

Operaciones - Carga programa 1

Orden No... Estado

1 C... Pendiente

Crear Plan de Movimientos

Nombre *
Reubicacion

Tipo *
Reubicar

Elementos + Agregar Elemento

De * A03	A * A04	ID del Material * 250000223 (A3)	⊖
De * A05	A * A06	ID del Material * 250000960 (A6)	⊖

Cerrar Crear

Filas por página: 100 1-1 de 1

Figura 6.19: Alta de un plan de movimiento para reubicación.

Tras confirmar la operación, el plan de movimiento se genera correctamente y, como se observa en la Figura 6.20, aparece en el listado de planes.

GenStock Transportes Inventario Plan de Movimientos Programas Mapa Administración

Programa de movimiento creado correctamente

Plan de Movimientos + Crear Actualizar

Orden	Nombre	Estado	Tipo	Zona de Transporte	Movimientos	Acciones
1	Carga programa 1	Pendiente	Cargar Programa	Bahía Camiones 3	2	
2	Reubicacion	Pendiente	Reubicar		2	

Filas por página: 100 1-2 de 2

Operaciones - Reubicacion

2 movimientos

Orden: 1 Pendiente (1 Elemento)

De	A	ID del Material
A03 (A3)	A04 (A10)	250000223

Orden: 2 Pendiente (1 Elemento)

De	A	ID del Material
A05 (A6)	A06 (A10)	250000960

Figura 6.20: Listado tras el alta del plan de reubicación.

El resultado obtenido permite verificar el cumplimiento de la especificación *PMO-05*.

6.4.4.5. Cancelación de un plan

Esta prueba verifica la especificación *PMO-07*, correspondiente a la cancelación de un plan de movimiento.

Para realizar la prueba se accede a la sección de planes de movimiento y se selecciona el plan que se desea cancelar. Se selecciona la opción de cancelación y se confirma la operación.

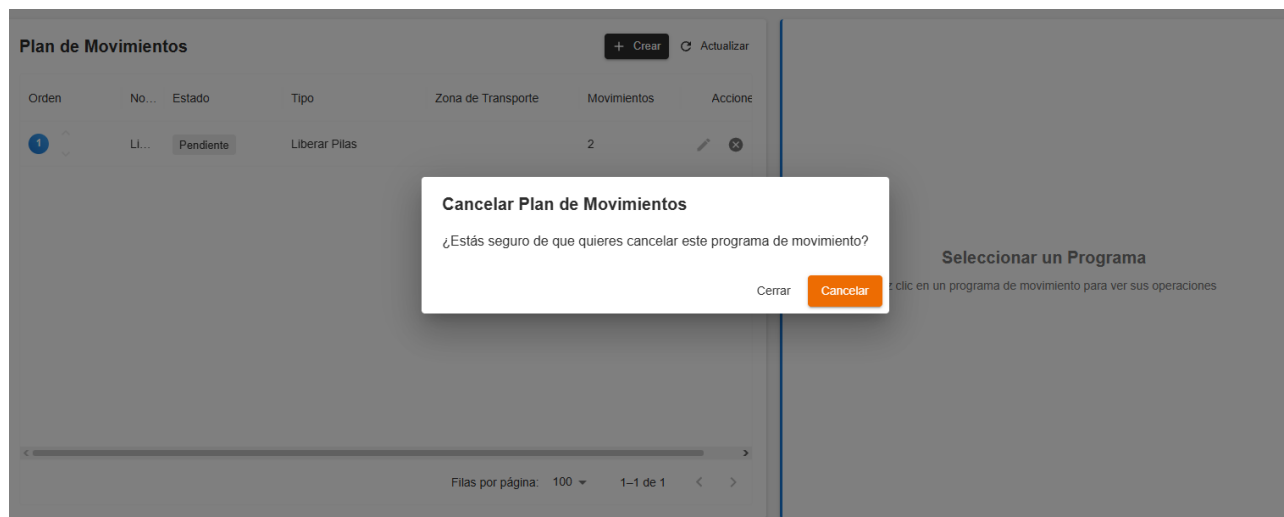


Figura 6.21: Confirmación de la cancelación de un plan de movimiento.

Tras confirmar la operación, aparece una notificación para indicar el éxito de la cancelación y el plan desaparece del listado de planes de movimiento, como se muestra en la Figura 6.22.

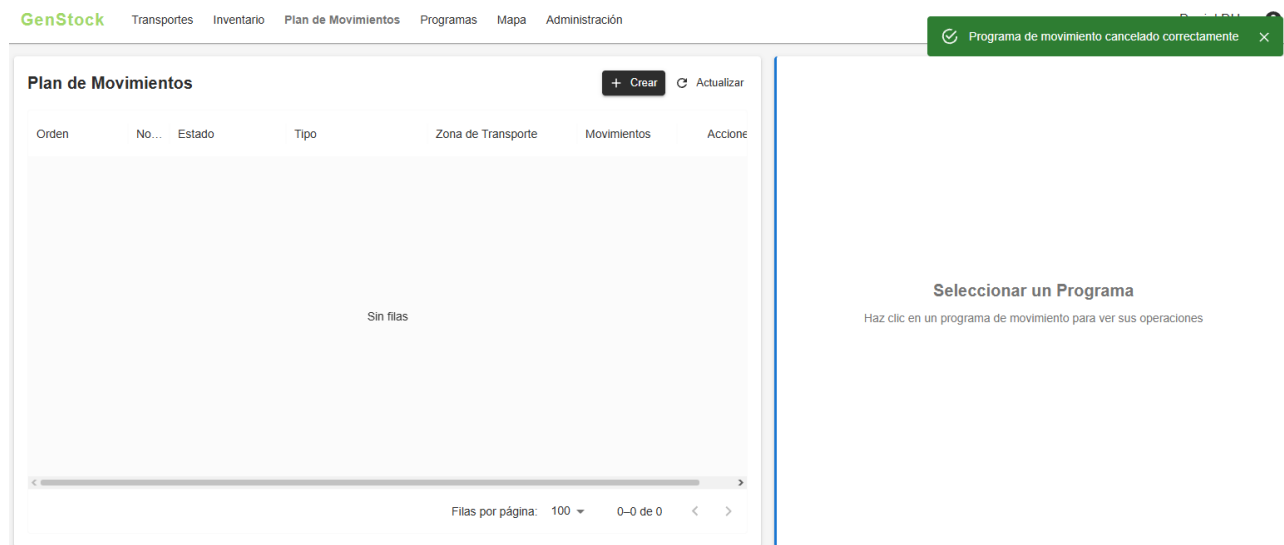


Figura 6.22: Listado de planes tras la cancelación.

El resultado obtenido permite verificar el cumplimiento de la especificación *PMO-07*.

6.4.4.6. Borrado de un plan

Esta prueba verifica la especificación *PMO-08*, correspondiente al borrado de un plan de movimiento.

Para realizar la prueba se accede a la sección de planes de movimiento y se selecciona el plan que se desea eliminar. Al seleccionar la opción de borrado aparece una ventana de confirmación de la operación, como se muestra en la Figura 6.23.

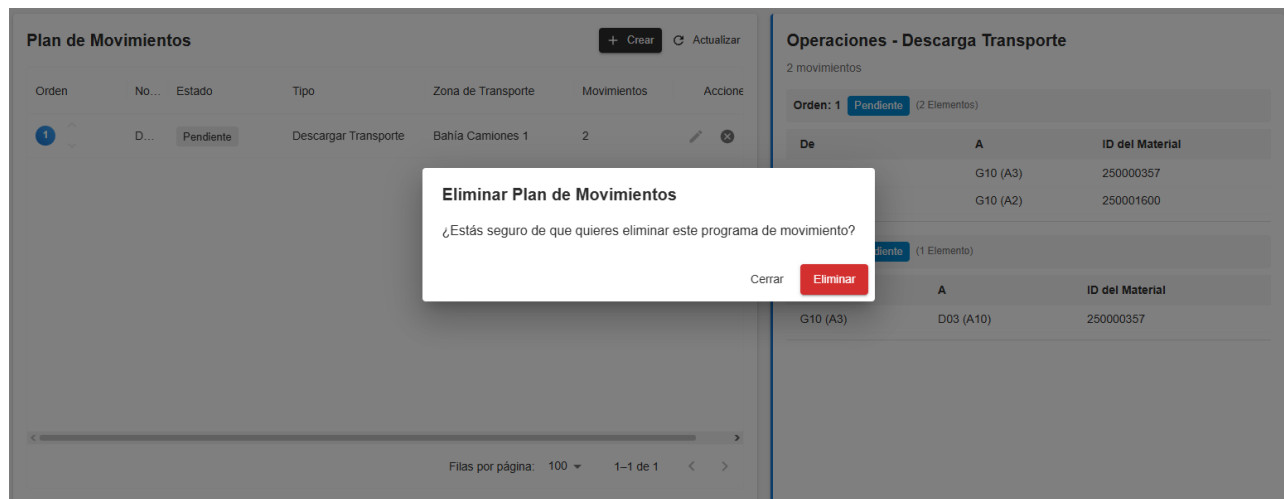


Figura 6.23: Confirmación del borrado de un plan de movimiento.

Tras confirmar la operación, aparece una notificación para indicar el éxito del borrado y el plan deja de aparecer en el listado de planes, como se muestra en la Figura 6.24.

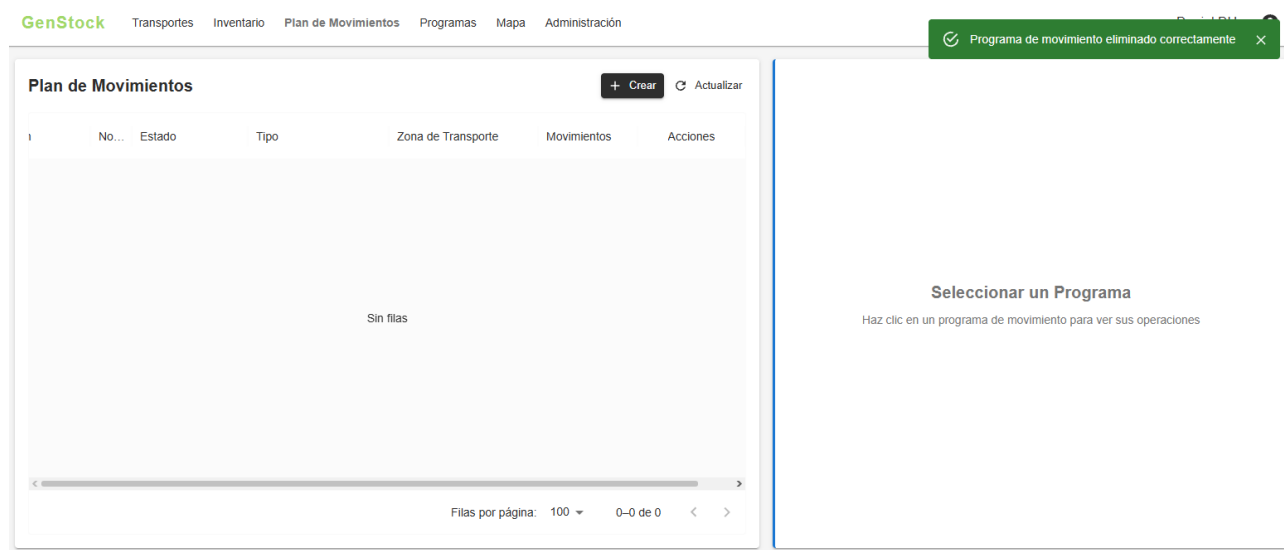


Figura 6.24: Listado de planes tras el borrado.

El resultado obtenido permite verificar el cumplimiento de la especificación *PMO-08*.

6.4.4.7. Cambio de orden de un plan

Esta prueba verifica la especificación *PMO-09*, correspondiente al cambio de orden de un plan de movimiento.

Para realizar la prueba se accede a la sección de planes de movimiento y se modifica la posición de uno de los planes dentro del listado. Para ello, se hace clic sobre la flecha arriba del segundo elemento para moverlo a la primera posición. La operación permite cambiar el orden de los planes de movimiento. La Figura 6.25 muestra el estado inicial.

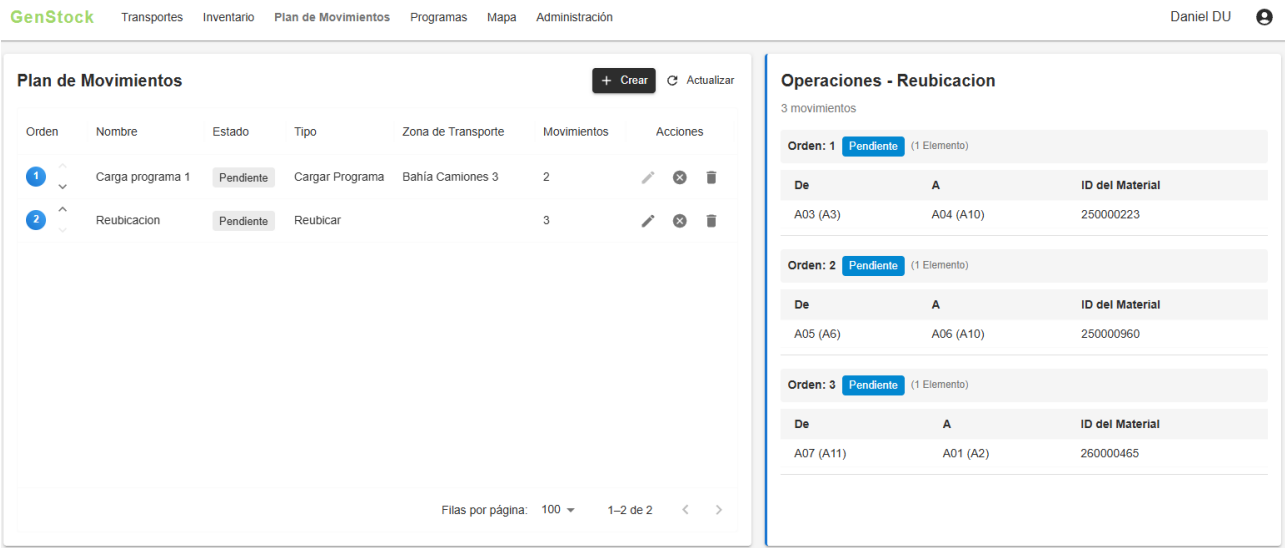


Figura 6.25: Cambio de orden de un plan de movimiento.

Tras realizar la operación, el nuevo orden se refleja en el listado de planes de movimiento, como se muestra en la Figura 6.26.

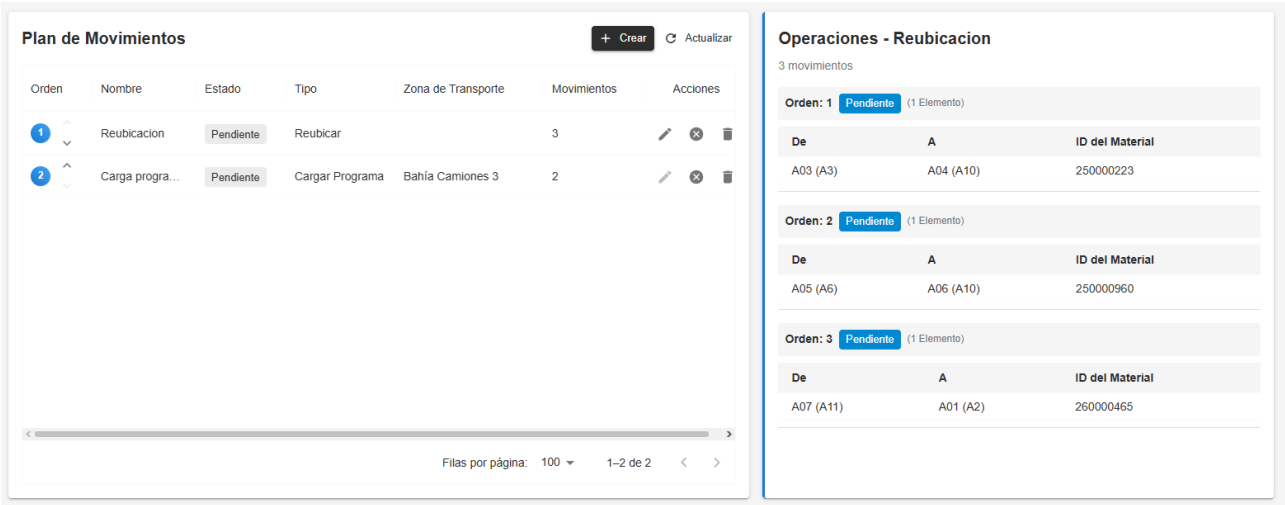


Figura 6.26: Listado de planes tras el cambio de orden.

El resultado obtenido permite verificar el cumplimiento de la especificación *PMO-09*.

6.4.5. Programas

Este apartado explora las pruebas funcionales realizadas sobre la sección de programas.

6.4.5.1. Listado de programas

Esta prueba verifica el listado de información de programas, recogido en la especificación *PRG-01*, así como *PRG-02* correspondiente a la consulta del detalle de un programa.

Para la realización de la prueba se accede al menú de programas, donde se observan los programas existentes en la aplicación. Estos programas se generan por una funcionalidad de simulación, por lo que no existen opciones en la interfaz para su gestión. A continuación, hacemos clic sobre el programa «202604231» de forma que se carguen los detalles del mismo en el menú lateral. En la Figura 6.27 se observan los detalles de las secuencias que forman parte del programa, así como su ubicación y altura en el almacén.



The screenshot shows the GenStock application interface. At the top, there is a navigation bar with the following items: GenStock, Transportes, Inventario, Plan de Movimientos, Programas, and Mapa. On the right side of the navigation bar, the user's name 'Daniel DU' and a profile icon are visible.

The main content area is divided into two panels. The left panel, titled 'Programas', contains a table with the following columns: Orden, Nombre, % Expedición, En parque, and Últ. Sec. Exp. The table has one row selected, with the following data: Orden 1, Nombre 202604231, % Expedición 24%, En parque 48 / 72, and Últ. Sec. Exp. 24. Below the table, there is a pagination bar showing 'Filas por página: 100' and '1-1 de 1'.

The right panel, titled 'Elementos - 202604231', shows the details of the selected program. It indicates there are 72 elements. Below this, there is a table with the following columns: Secuencia, Grupo, Material, Ubicación, and Altura. The table displays the following data:

Secuencia	Grupo	Material	Ubicación	Altura
28	28	250001898	D12	2
29	29	250001181	E11	1
30	30	260000017	D06	1
31	31	250001581	E06	2
32	32	260000863	F13	3

Below the table, there is a pagination bar showing 'Filas por página: 100' and '1-72 de 72'.

Figura 6.27: Listado con programa seleccionado.

De esta forma concluimos que las especificaciones *PRG-01* y *PRG-02* están verificadas.

6.4.6. Mapa

En esta sección se muestra en todo momento el estado actual del almacén. A continuación, se detallan las pruebas funcionales que garantizan el correcto funcionamiento.

6.4.6.1. Visualización del estado del almacén

Esta prueba comprueba la visualización del estado del almacén a través del mapa, cuya funcionalidad está recogida en la especificación *MAP-01*.

Para la realización de la prueba se accede al menú del mapa, donde se observa que la representación de pilas, transportes y grúa coinciden con el estado actual del almacén. La Figura 6.28, demuestra el funcionamiento de la visualización, para la que se ha comprobado que muestra correctamente el estado actual.

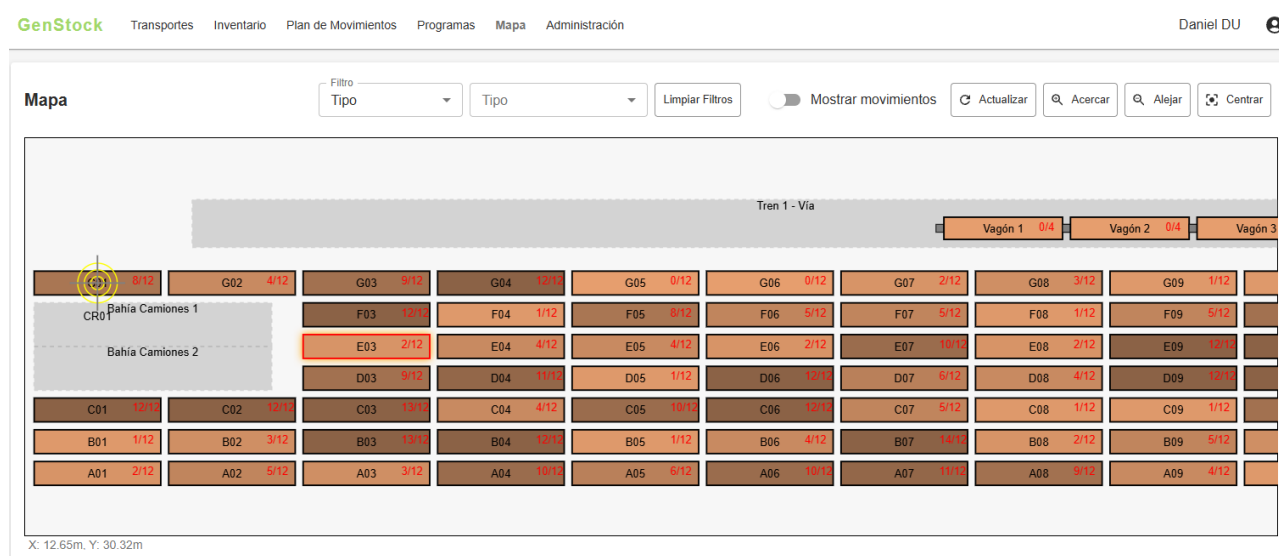


Figura 6.28: Mapa del estado actual del almacén.

Concluimos por tanto que se cumple la especificación *MAP-01*.

6.4.6.2. Búsqueda de materiales

En esta prueba se verificará la búsqueda de material, descrita en la especificación *MAP-02*, junto con la visualización de detalles de una pila, descrita en *MAP-03*.

La prueba se realiza accediendo al menú del mapa, seleccionando el filtro superior de tipo y seleccionando «Arranque» como tipo a filtrar. Al realizar esta acción, se observa que las pilas que contienen material de tipo arranque se muestran resaltadas en azul. Una de las pilas con este tipo de material es la pila «G02», haciendo clic en ella, se abre el panel lateral que muestra la información detallada del contenido de la pila. En la Figura 6.29 se puede observar el resultado de la prueba y comprobar que, efectivamente, la pila está compuesta por elementos de tipo «Arranque».

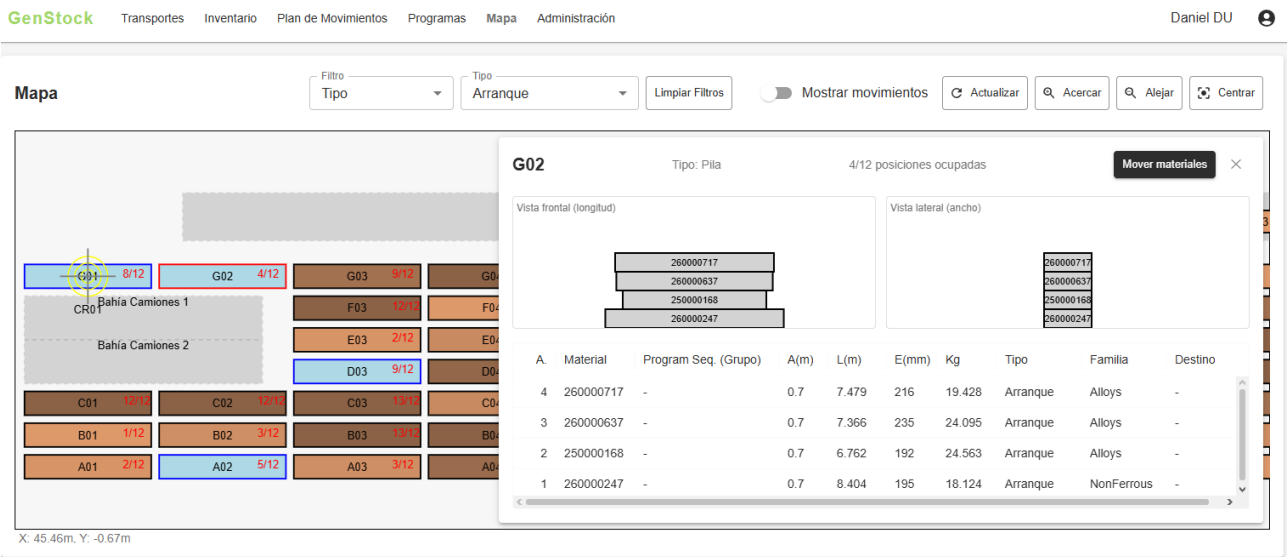


Figura 6.29: Mapa de estado actual del almacén con filtro de tipo.

De esta forma concluimos que se cumplen tanto la especificación *MAP-02* como *MAP-03*.

6.4.6.3. Visualización de movimientos

En esta prueba se verificará la visualización de los movimientos del almacén descrita en la especificación *MAP-04*.

La prueba se realiza accediendo al menú del mapa y activando la opción «Mostrar movimientos», disponible en la barra superior. Al realizar esta acción, se muestran sobre el mapa los movimientos de material pendientes entre las diferentes pilas del almacén, permitiendo visualizar gráficamente los desplazamientos que se deben realizar. En la Figura 6.30 se puede observar el resultado de la prueba.



Figura 6.30: Mapa del almacén con la visualización de movimientos activada.

De esta forma concluimos que se cumple la especificación *MAP-04*.

6.4.7. Movimientos

En esta sección se comprueba la funcionalidad para la realización manual de movimientos de material entre las distintas pilas del almacén. A continuación, se detallan las pruebas funcionales realizadas para verificar su correcto funcionamiento.

6.4.7.1. Movimiento de material entre pilas

En esta prueba se verificará la realización de movimientos de material descrita en la especificación *MOV-01*.

La prueba se realiza accediendo al menú del mapa y seleccionando la pila «B14». En el panel lateral se muestra la información detallada del contenido de la pila, como se puede observar en la Figura 6.31. Desde este panel se selecciona la opción «Mover materiales» para iniciar la operación.

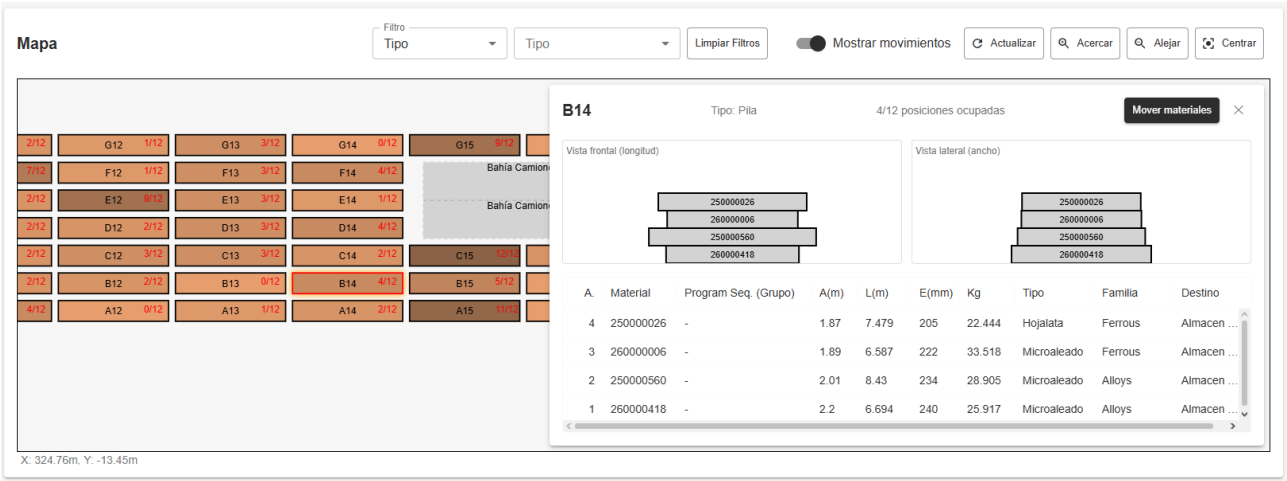


Figura 6.31: Estado de la pila «B14» antes de realizar el movimiento.

Al seleccionar la opción «Mover materiales» se abre un diálogo en el que se muestra la pila de origen y se permite seleccionar la pila de destino y el número de materiales a mover. En este caso, se selecciona la pila «A14» como destino y se indica el movimiento de un desbaste, como se muestra en la Figura 6.32.

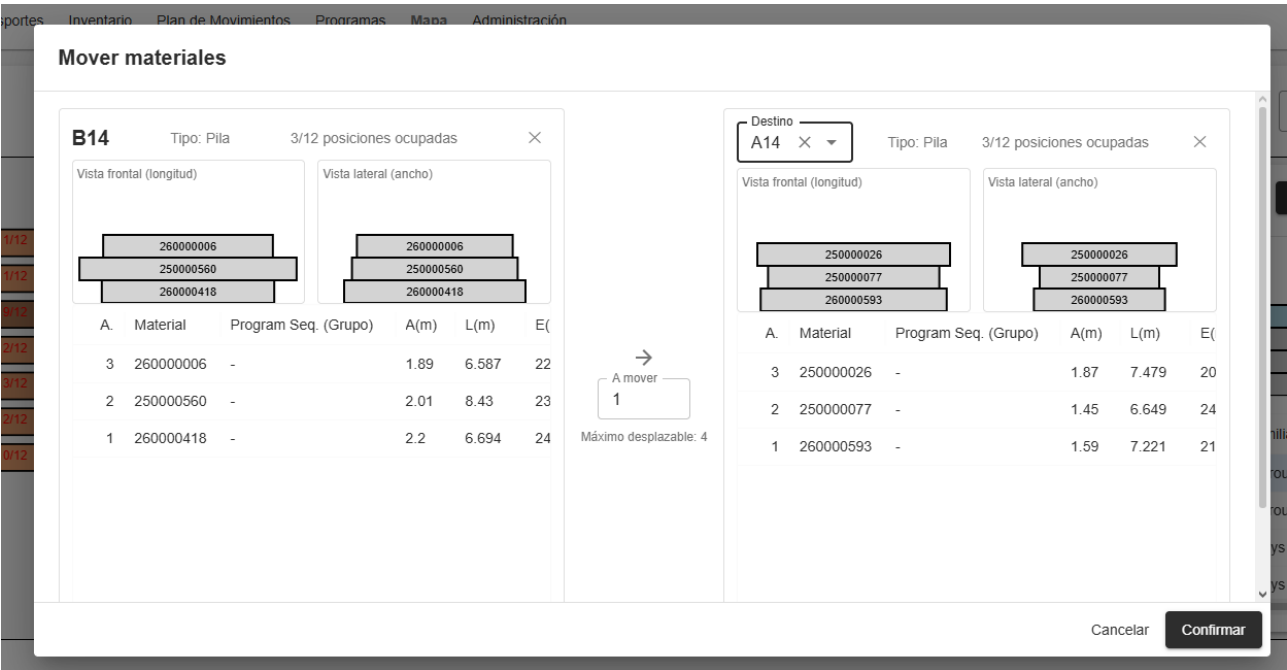


Figura 6.32: Diálogo para realizar el movimiento de material.

Tras confirmar la operación, el movimiento se realiza y se actualiza el estado de las pilas. El desbaste pasa de la pila «B14» a la pila «A14», como se puede comprobar en la Figura 6.33. Cabe destacar que el movimiento ha producido que la pila deje de ser segura, lo que se visualiza mediante el indicador de advertencia que se encuentra sobre la pila.

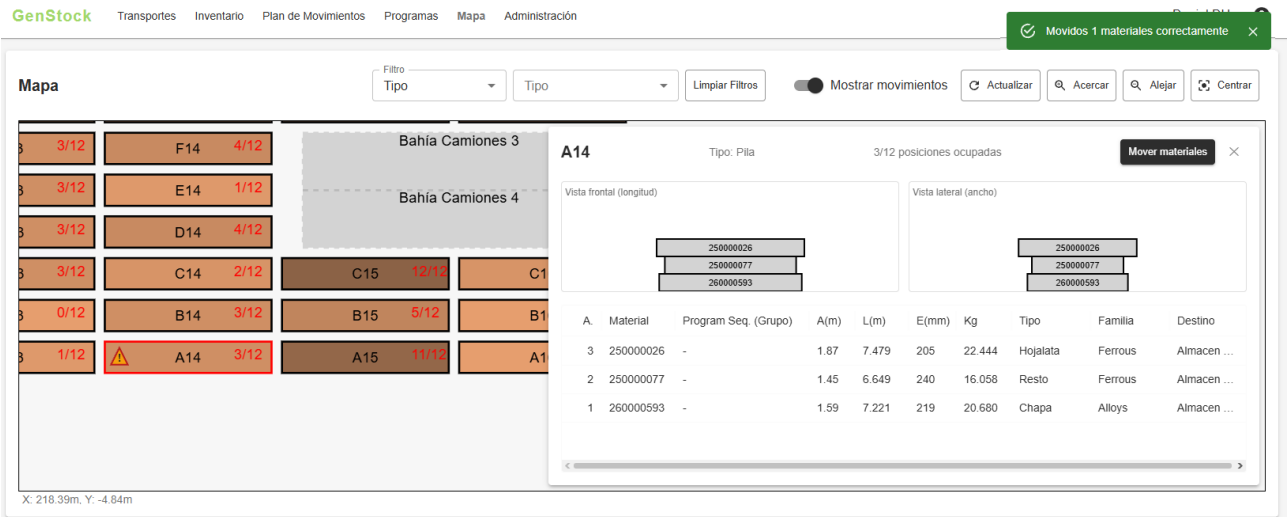


Figura 6.33: Estado de las pila «A14» después de realizar el movimiento.

El resultado obtenido permite verificar el cumplimiento de la especificación *MOV-01*, correspondiente a la realización de movimientos de material entre pilas.

6.4.8. Gestión de destinos

A través de esta sección se pueden administrar los destinos que pueden tener los materiales del almacén. A continuación, se muestran las pruebas funcionales realizadas.

6.4.8.1. Alta de destino

Esta prueba verifica la especificación *DES-02*, correspondiente al alta de destinos.

Para realizar la prueba se accede a la sección de administración y después a la opción de destinos, se selecciona la opción «Agregar Destino», aparece el formulario de la Figura 6.34 donde introducimos los datos de un nuevo destino.

The screenshot shows the 'GenStock' application interface. The top navigation bar includes 'Transportes', 'Inventario', 'Plan de Movimientos', 'Programas', 'Mapa', 'Administración', and a user profile 'Daniel DU'. The left sidebar under 'Administración' lists 'Usuarios', 'Destinos' (highlighted), 'Grúas', and 'Asignación de Materiales'. A modal window titled 'Crear Destino' is open, containing two input fields: 'Nombre*' with the value 'Nuevo destino' and 'Descripción' with the value 'Prueba funcional'. At the bottom of the modal are 'Cancelar' and 'Guardar' buttons. In the background, a table with 'Acciones' (edit and delete icons) is partially visible.

Figura 6.34: Formulario de alta de un destino.

Tras pulsar sobre guardar, aparece una notificación para indicar el éxito de la operación y el destino aparece en el listado de destinos.

The screenshot shows the 'Destinos' list in the GenStock application. A green notification banner at the top right states 'Destino creado correctamente'. The left sidebar is the same as in Figure 6.34. The main content area displays a table with the following data:

Nombre	Descripción	Acciones
Almacen Externo A	Almacén principal sur	
Puerto	Zona de carga en puerto	
Nuevo destino	Prueba funcional	

At the bottom of the table, there is a pagination control showing 'Filas por página: 100' and '1-3 de 3' with navigation arrows. A '+ Agregar Destino' button is located at the top right of the table area.

Figura 6.35: Listado destinos con nuevo elemento.

El resultado obtenido permite verificar el cumplimiento de la especificación *DES-02* y, a su vez, la especificación *DES-01*, correspondiente al listado de destinos.

6.4.8.2. Modificación de un destino

Esta prueba verifica la especificación *DES-03*, correspondiente a la modificación de destinos.

Para realizar la prueba se accede a la sección de administración y después a la opción de destinos. En el listado de destinos, se selecciona el destino pulsando el icono de lápiz de la fila y se accede a su formulario de edición. En este formulario se modifican los datos del destino, como se muestra en la Figura 6.36.

The screenshot displays the GenStock application interface. At the top, a navigation bar includes links for Transportes, Inventario, Plan de Movimientos, Programas, Mapa, and Administración, along with a user profile for Daniel DU. The left sidebar under 'Administración' lists 'Usuarios', 'Destinos' (highlighted), 'Grúas', and 'Asignación de Materiales'. The main content area features a modal window titled 'Editar Destino'. This modal contains two text input fields: 'Nombre*' with the value 'Nuevo destino - mod' and 'Descripción' with the value 'modificacion'. At the bottom right of the modal are 'Cancelar' and 'Guardar' buttons. In the background, a table of destinations is visible, with each row having edit and delete icons. The bottom of the page shows pagination information: 'Filas por página: 100' and '1-3 de 3'.

Figura 6.36: Formulario de modificación de un destino.

Tras pulsar sobre guardar, aparece una notificación para indicar el éxito de la operación y los datos modificados se reflejan en el listado de destinos, como se muestra en la Figura 6.37.

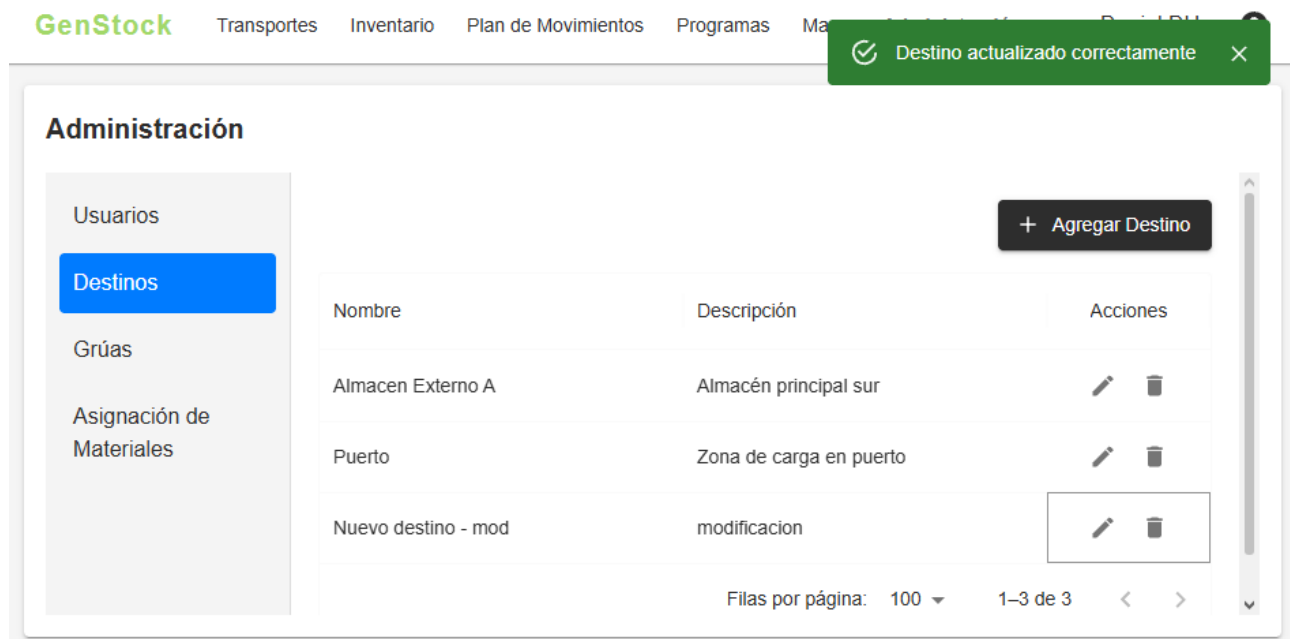


Figura 6.37: Listado de destinos con el elemento modificado.

El resultado obtenido permite verificar el cumplimiento de la especificación *DES-03* y, a su vez, la especificación *DES-01*, correspondiente al listado de destinos.

6.4.8.3. Baja de un destino

Esta prueba verifica la especificación *DES-04*, correspondiente a la baja de destinos.

Para realizar la prueba se accede a la sección de administración y después a la opción de destinos. En el listado de destinos, se selecciona el destino que se desea eliminar. Al hacer clic sobre el icono de papelera disponible en la fila, aparece una ventana de confirmación de la operación, como se muestra en la Figura 6.38.

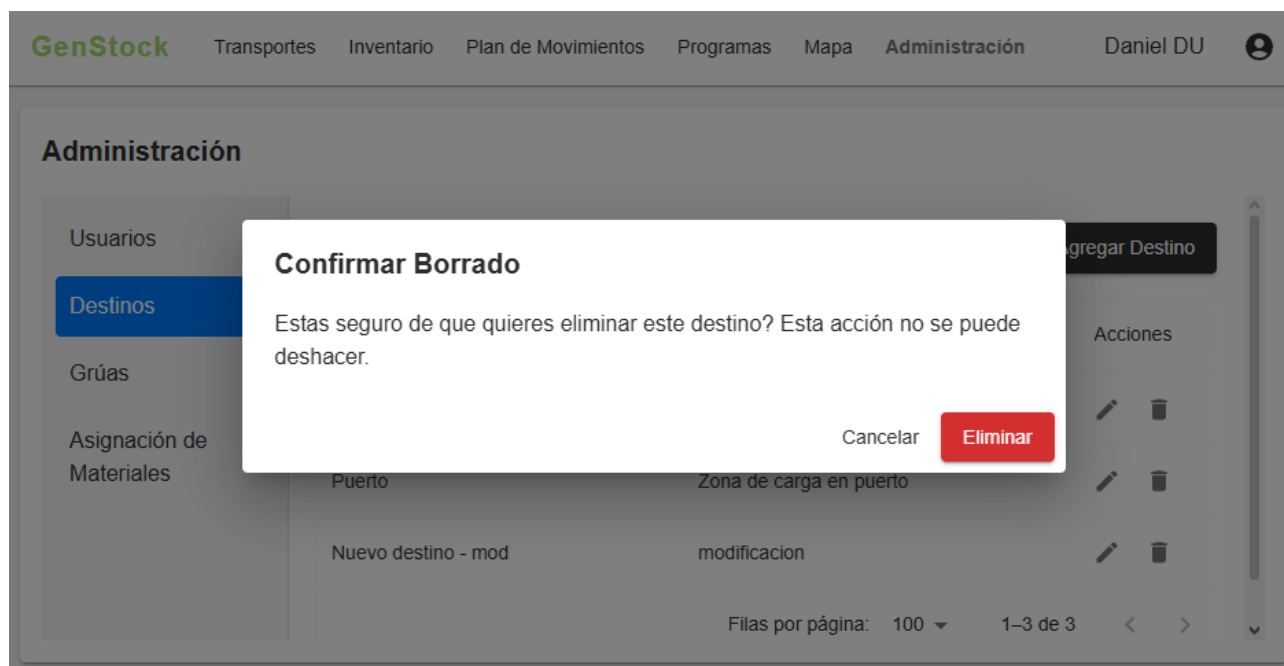


Figura 6.38: Confirmación del borrado de un destino.

Tras confirmar la operación, aparece una notificación para indicar el éxito del borrado y el destino deja de aparecer en el listado de destinos. La Figura 6.39 muestra el listado de destinos, una vez realizado el borrado.

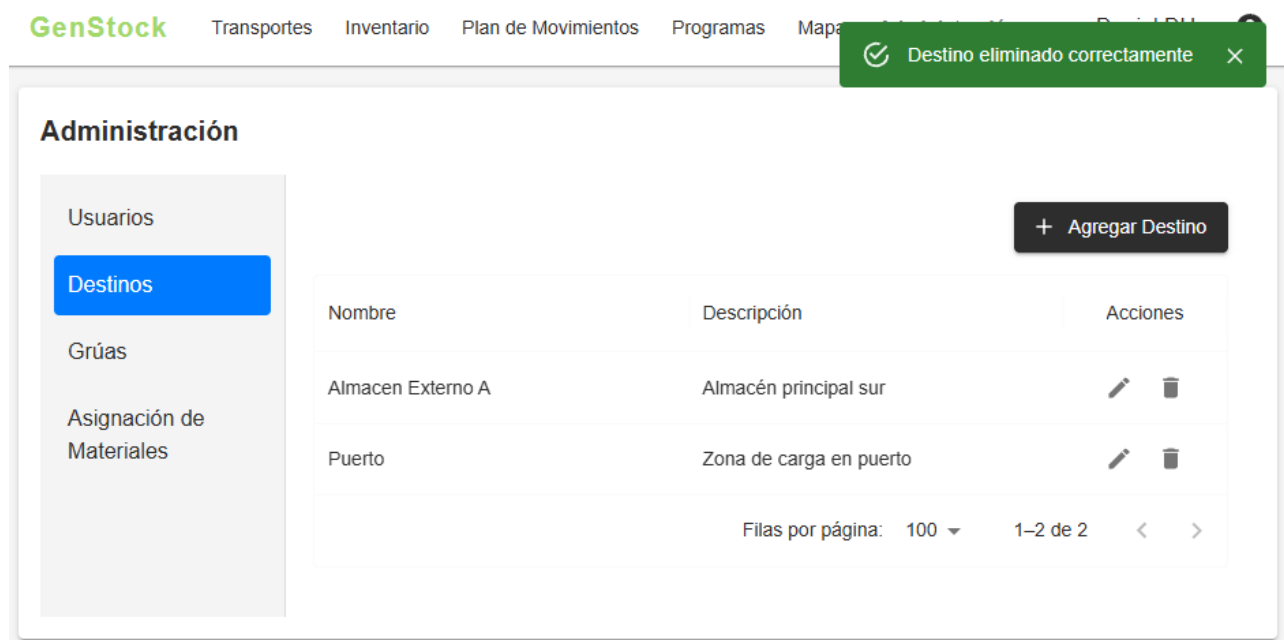


Figura 6.39: Listado de destinos tras el borrado.

El resultado obtenido permite verificar el cumplimiento de la especificación *DES-04* y, a su vez, la especificación *DES-01*, correspondiente al listado de destinos.

6.4.9. Gestión de usuarios

En este apartado se exploran las pruebas funcionales realizadas sobre la gestión de usuarios de la aplicación.

6.4.9.1. Alta de usuario

Esta prueba verifica la especificación *USU-02*, correspondiente al alta de usuarios.

Para realizar la prueba se accede a la sección de administración y después a la opción de usuarios, se selecciona la opción «Agregar Usuario», aparece el formulario de la Figura 6.40, donde introducimos los datos de un nuevo usuario.

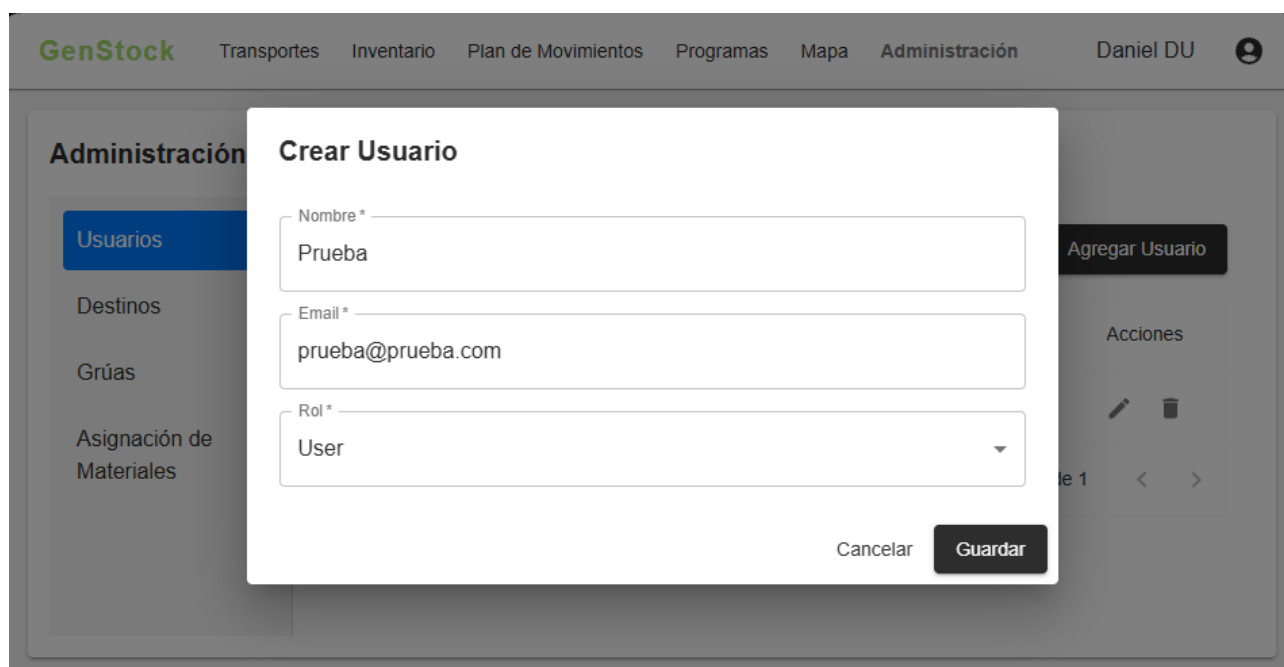


Figura 6.40: Formulario de alta de un usuario.

Tras pulsar sobre guardar, el usuario aparece en el listado de usuarios, como se muestra en la Figura 6.41.

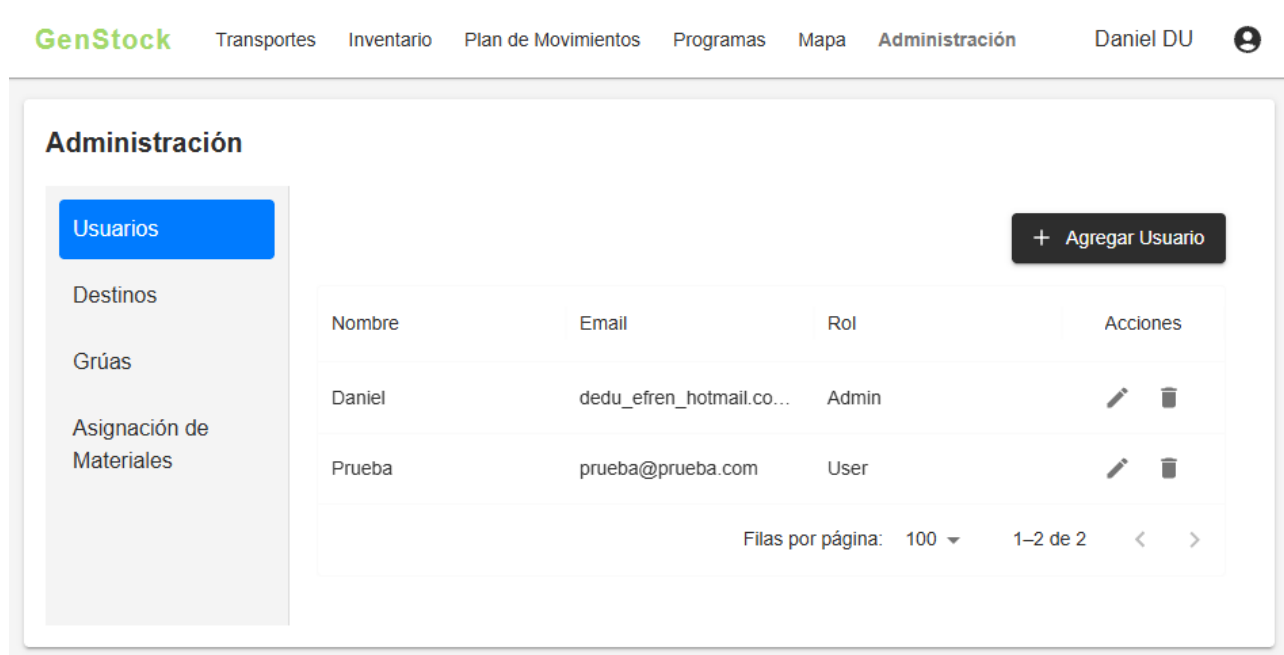


Figura 6.41: Listado de usuarios con nuevo elemento.

El resultado obtenido permite verificar el cumplimiento de la especificación *USU-02* y, a su vez, la especificación *USU-01*, correspondiente al listado de usuarios.

6.4.9.2. Modificación de un usuario

Esta prueba verifica la especificación *USU-03*, correspondiente a la modificación de usuarios.

Para realizar la prueba se accede a la sección de administración y después a la opción de usuarios. En el listado de usuarios, se selecciona el usuario que se desea modificar y se accede a su formulario de edición. En este formulario se modifica el rol, único dato modificable del usuario, como se muestra en la Figura 6.42.

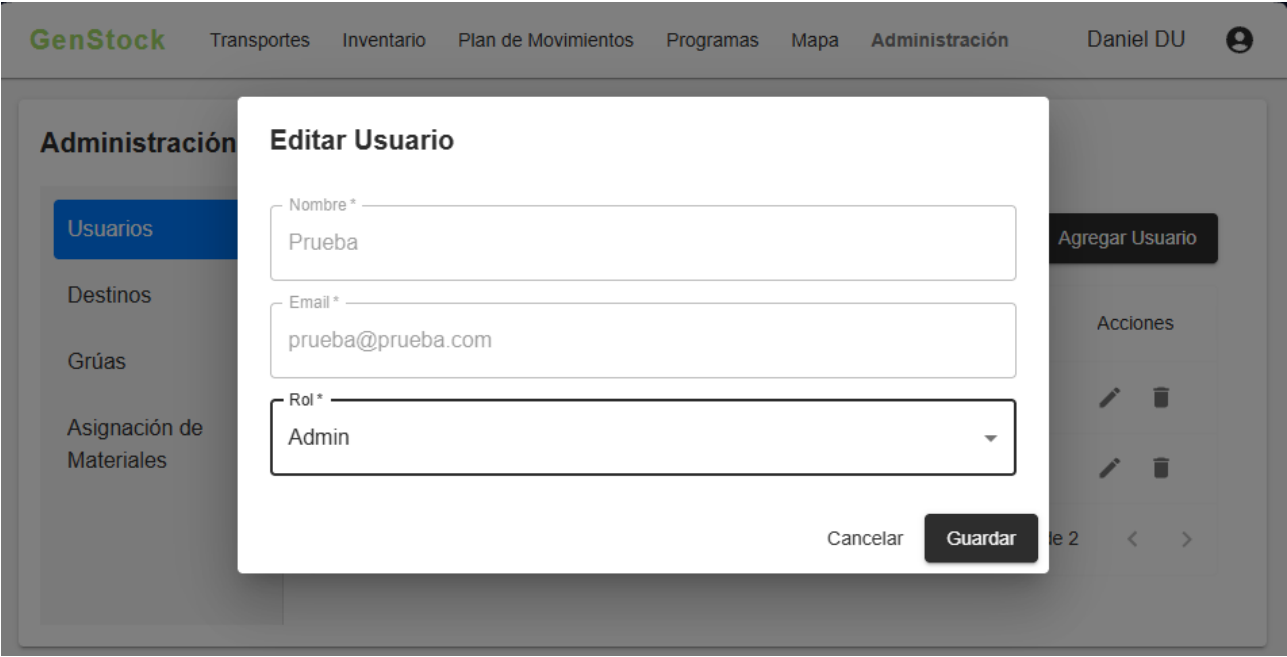


Figura 6.42: Formulario de modificación de un usuario.

Tras pulsar sobre guardar, aparece una notificación para indicar el éxito de la operación y el rol de usuario modificado se refleja sobre el listado de usuarios, como se muestra en la Figura 6.43.

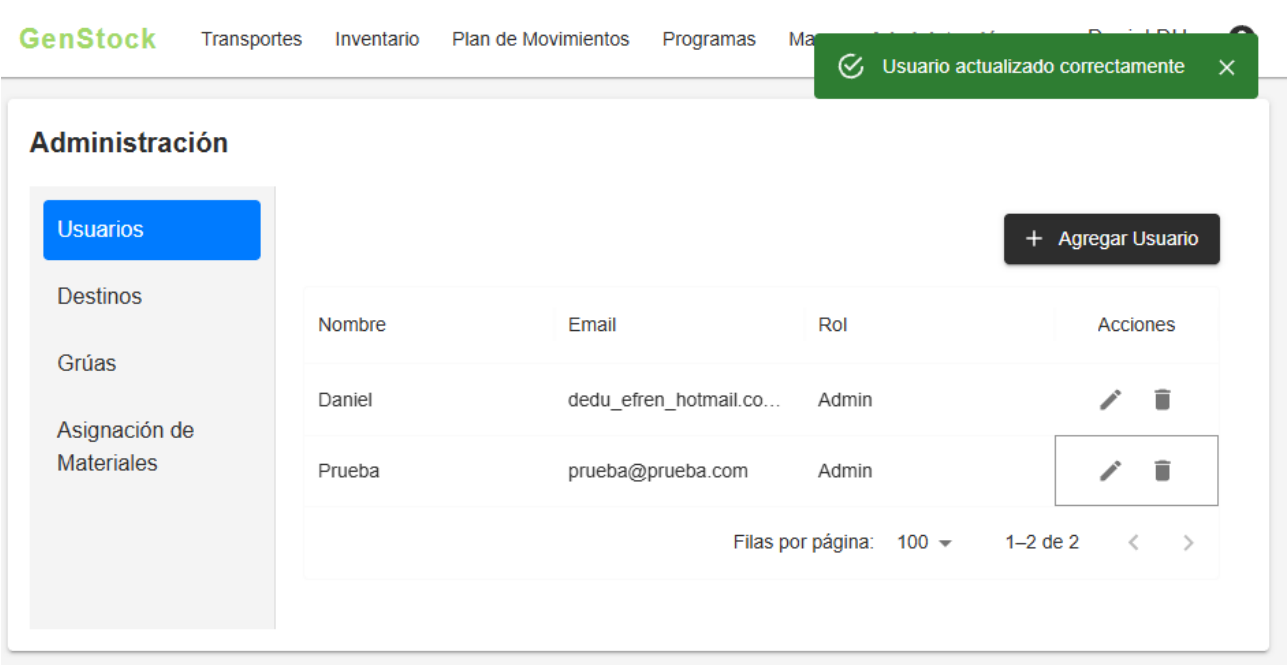


Figura 6.43: Listado de usuarios con el elemento modificado.

El resultado obtenido permite verificar el cumplimiento de la especificación *USU-03* y, a su vez, la especificación *USU-01*, correspondiente al listado de usuarios.

6.4.9.3. Baja de un usuario

Esta prueba verifica la especificación *USU-04*, correspondiente a la baja de usuarios.

Para realizar la prueba se accede a la sección de administración y después a la opción de usuarios. En el listado de usuarios, se selecciona el usuario que se desea eliminar. Al seleccionar la opción de borrado aparece una ventana de confirmación de la operación, como se muestra en la Figura 6.44.

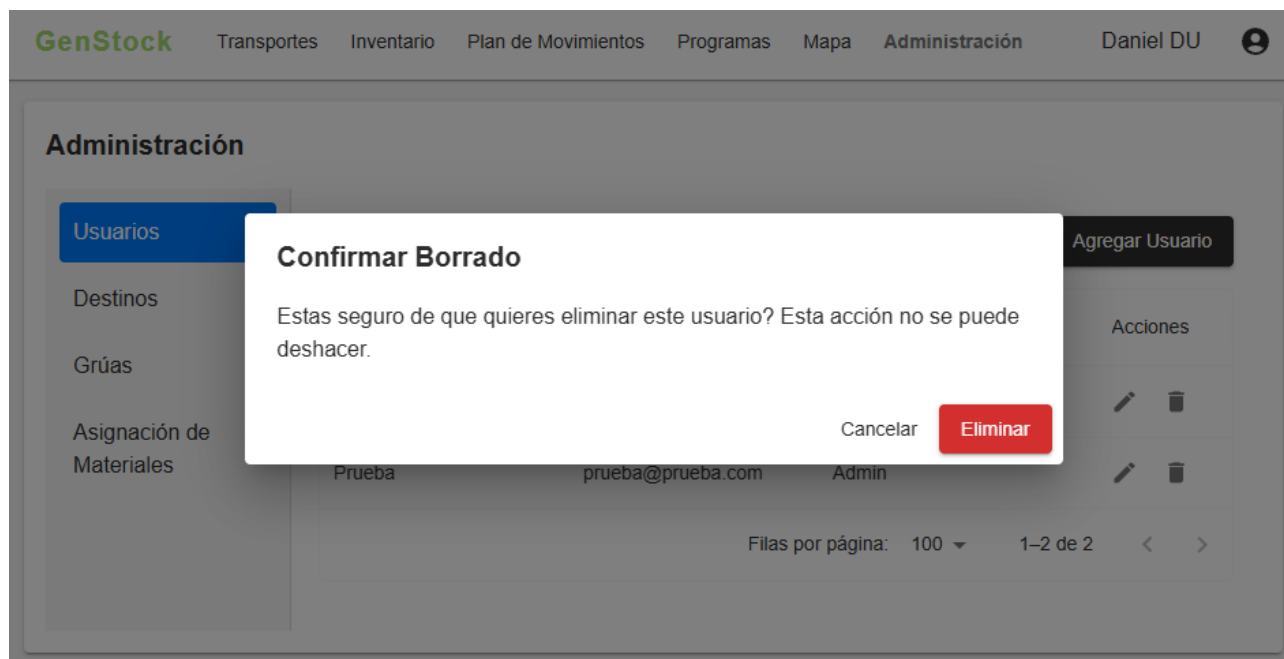


Figura 6.44: Confirmación del borrado de un usuario.

Tras confirmar la operación, aparece una notificación para indicar el éxito del borrado y el usuario deja de aparecer en el listado de usuarios, como se muestra en la Figura 6.45.

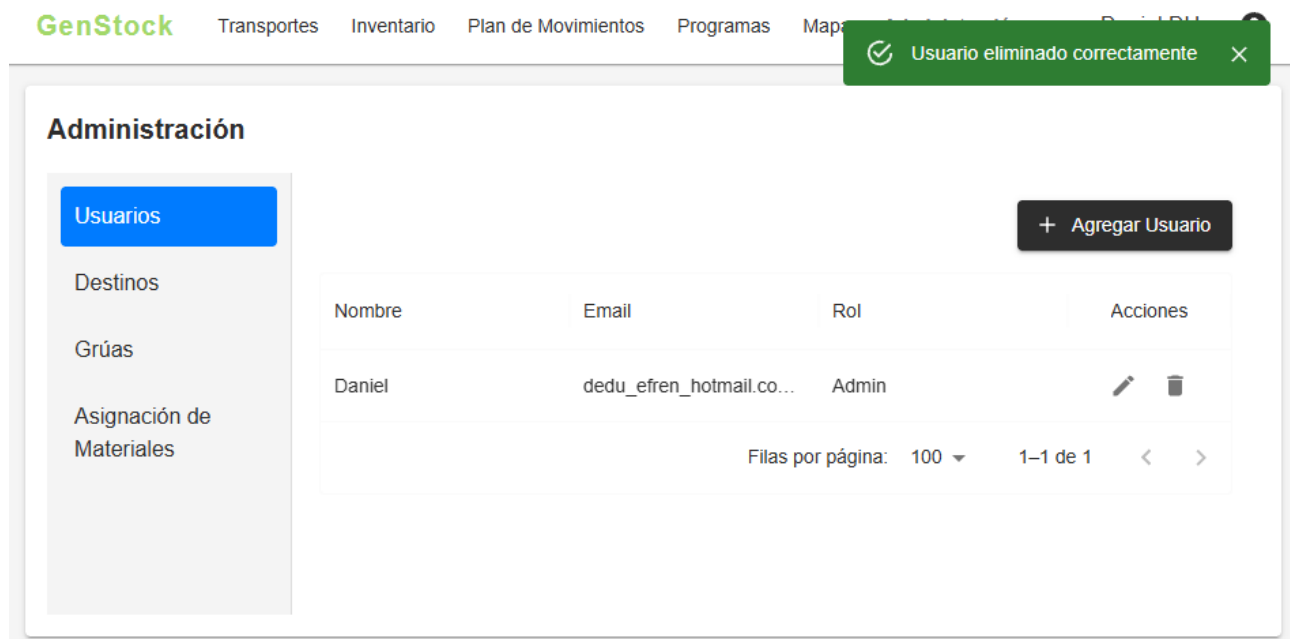


Figura 6.45: Listado de usuarios tras el borrado.

El resultado obtenido permite verificar el cumplimiento de la especificación *USU-04* y, a su vez, la especificación *USU-01*, correspondiente al listado de usuarios.

6.4.10. Gestión de grúas

6.4.10.1. Modificación de una grúa

Esta prueba verifica la especificación *GRU-02*, correspondiente a la modificación de grúas.

Para realizar la prueba se accede a la sección de administración y después a la opción de grúas. La Figura 6.46 muestra el estado actual de la única grúa configurada en el almacén.

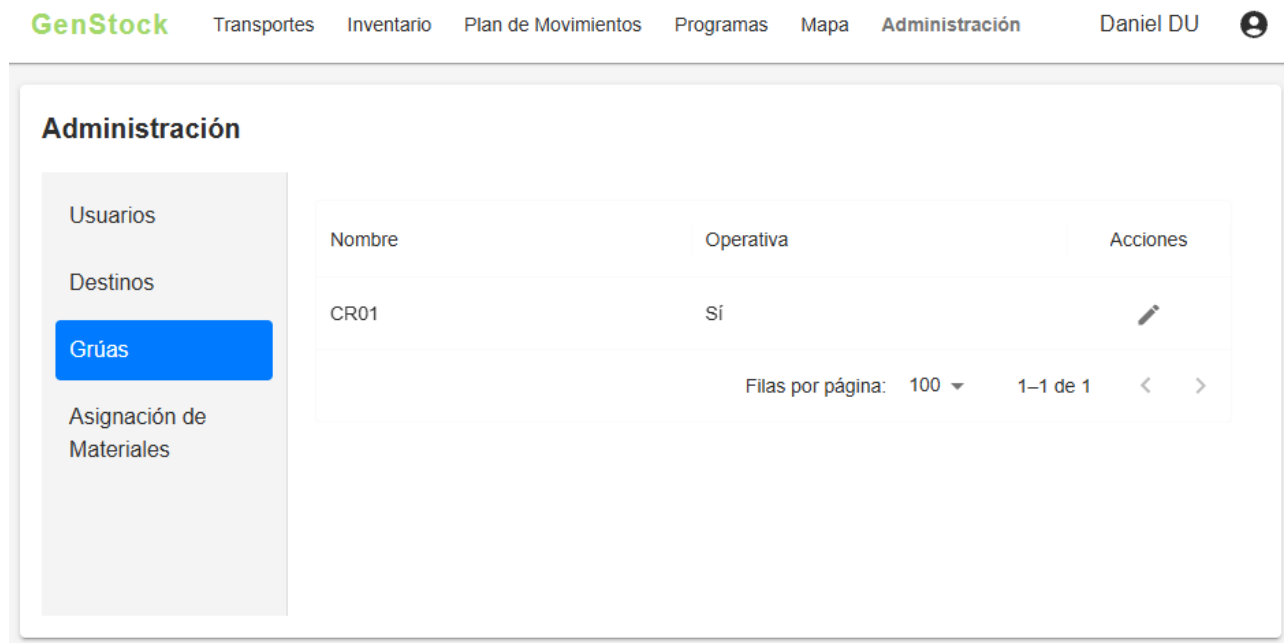


Figura 6.46: Listado de grúas.

En el listado de grúas, se selecciona la grúa que se desea modificar y se accede a su formulario de edición, mediante el icono del lápiz. En este formulario, se modifican los datos de la grúa, cambiando su nombre y estado operativo, como se muestra en la Figura 6.47.

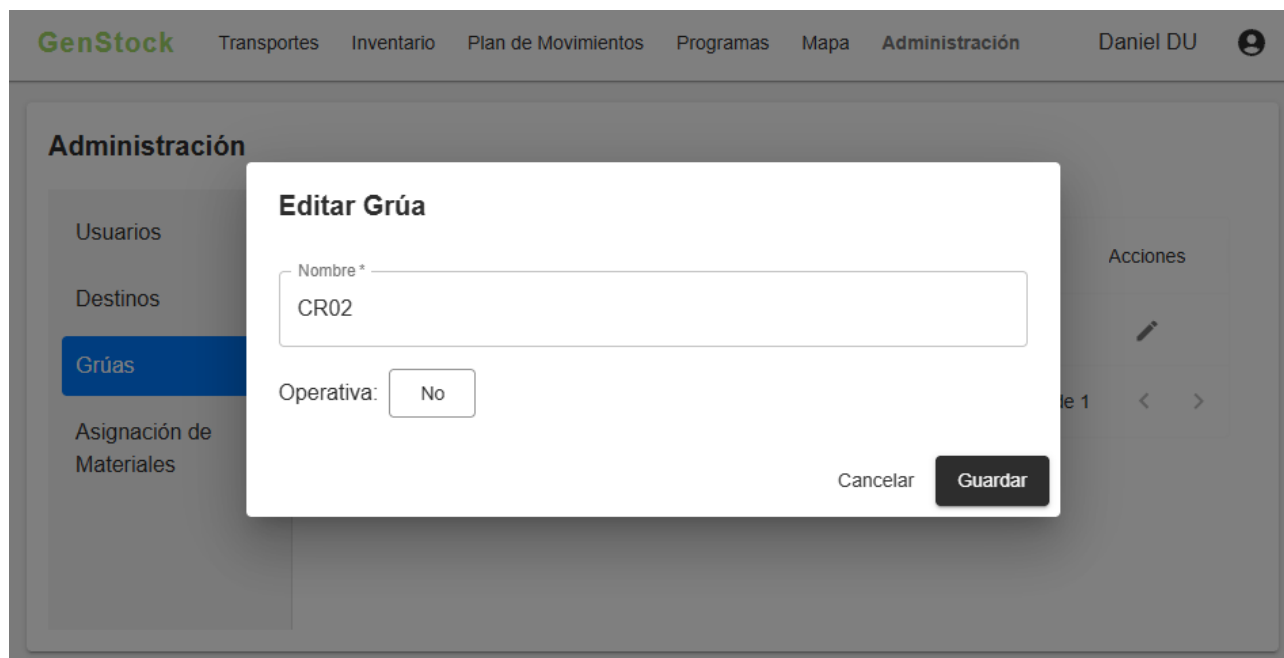


Figura 6.47: Formulario de modificación de una grúa.

Tras pulsar sobre guardar, aparece una notificación para indicar el éxito de la operación y los datos modificados se reflejan en el listado de grúas, como se muestra en la Figura 6.48.

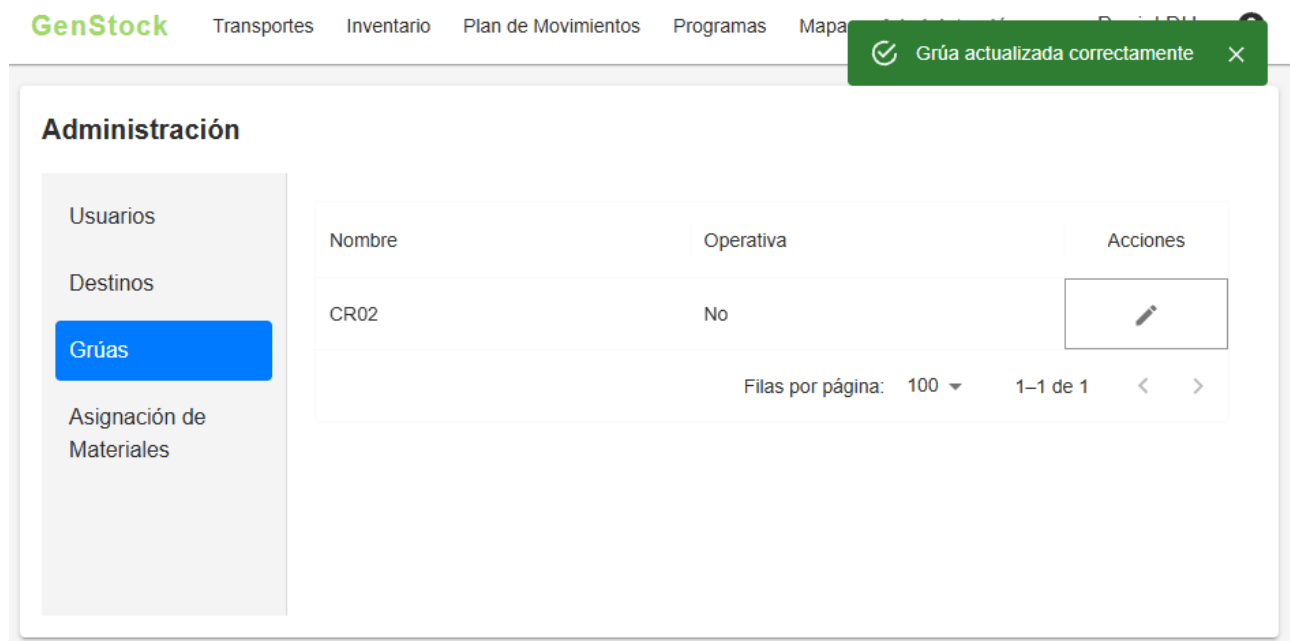


Figura 6.48: Listado de grúas con el elemento modificado.

El resultado obtenido permite verificar el cumplimiento de la especificación *GRU-02* y, a su vez, la especificación *GRU-01*, correspondiente al listado de grúas.

6.5. Pruebas de los servicios de simulación

En este apartado se recogen las pruebas de las funcionalidades que permiten disponer de un entorno con la información necesaria para la verificación de la aplicación.

6.5.1. Simulación de grúas

En esta prueba se verificará la simulación de grúa, correspondiente a la especificación *SIM-01*, establecida para la realización de los movimientos, emulando un operador de grúa.

Para la realización de esta prueba, se ha creado un plan de movimientos de carga de programa a la bahía camiones 3. Una vez creado el plan en la aplicación y arrancada la simulación, se observa cómo la grúa empieza a moverse en el mapa realizando el transporte del material. En la Figura 6.49, se observa cómo la grúa está completando el movimiento de un desbaste de la pila «E11» a la bahía.

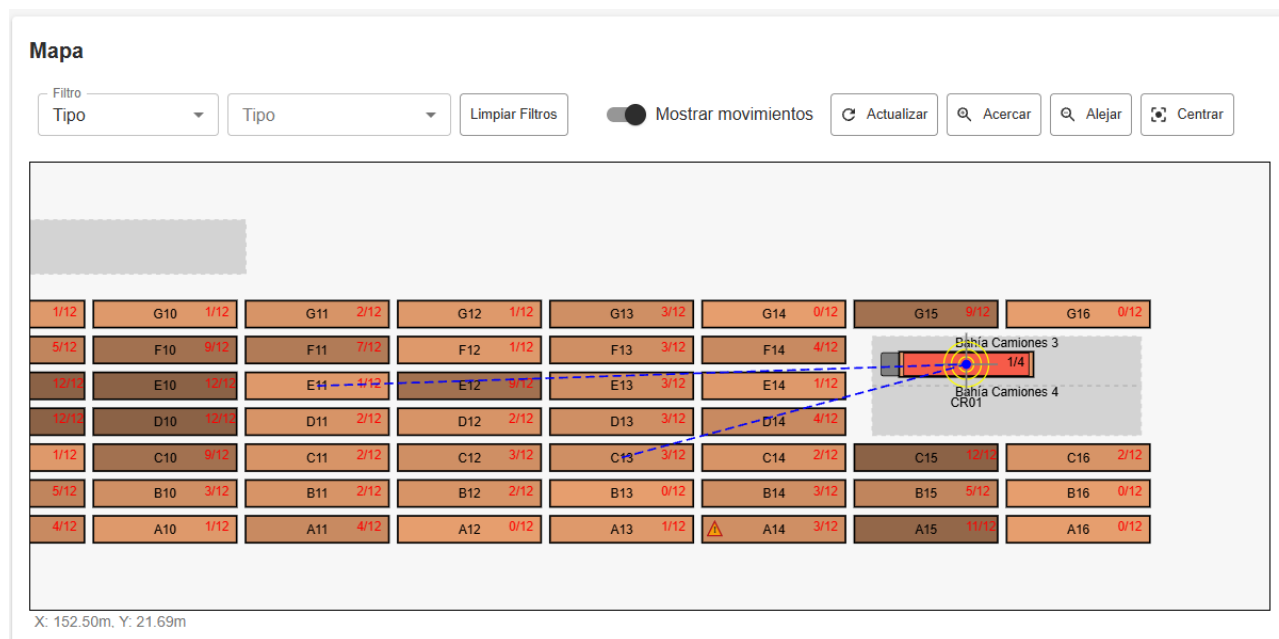


Figura 6.49: Grúa realizando depósito en la bahía camiones 3.

Una vez depositado el material, se observa que ya aparece en el contador de ocupación de la bahía y la simulación continúa con el siguiente material. En este caso, el origen es la pila «C13», en la Figura 6.50 observamos cómo se está retirando el desbaste para el segundo movimiento.

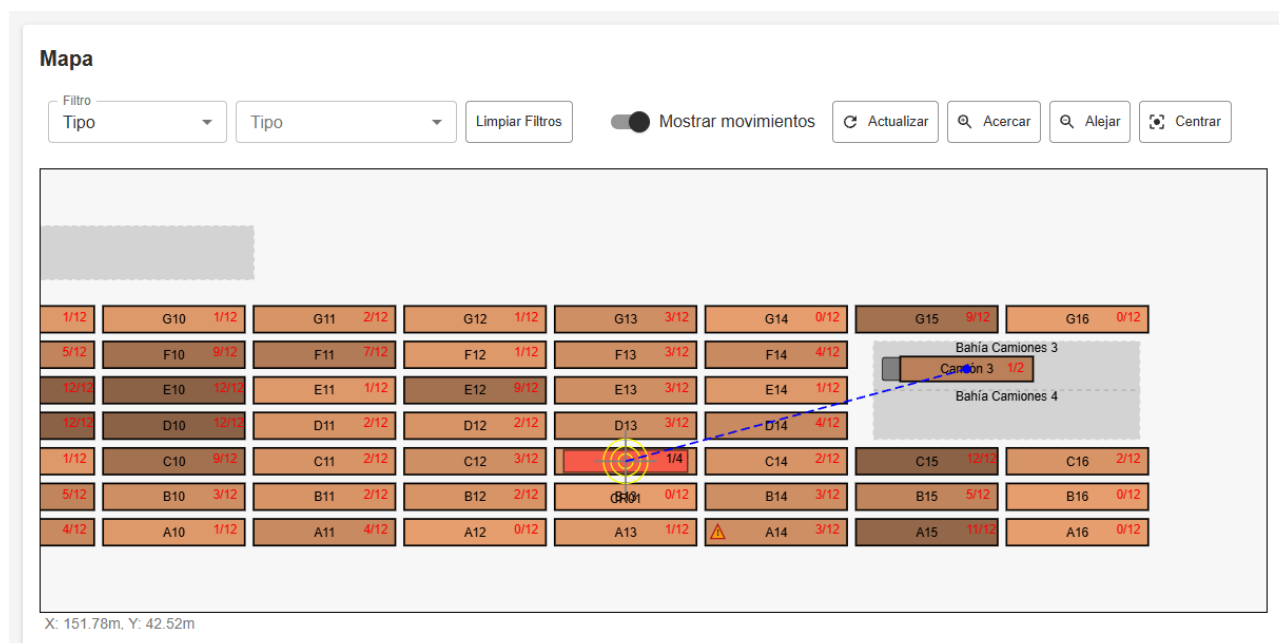


Figura 6.50: Grúa realizando la ganchada sobre la pila «C13».

Mientras que lo anterior está ocurriendo, la Figura 6.51 muestra cómo podemos seguir el avance de la operación, en la pantalla de planes de movimientos, a través de los estados de las ganchadas.

Operaciones - Carga programa 1

2 movimientos

Orden: 1

Completado

(1 Elemento)

De	A	ID del Material
E11 (A2)	Camión 3 (A1)	250001064

Orden: 2

En Curso

(1 Elemento)

De	A	ID del Material
C13 (A3)	Camión 3 (A2)	250002033

Figura 6.51: Progreso de la operación sobre la pantalla de planes de movimiento.

Cuando se completa el plan, y en caso de que no haya un nuevo plan, la grúa queda a la espera sobre la ubicación donde se ha realizado el último depósito. Esta situación puede observarse en la Figura 6.52.

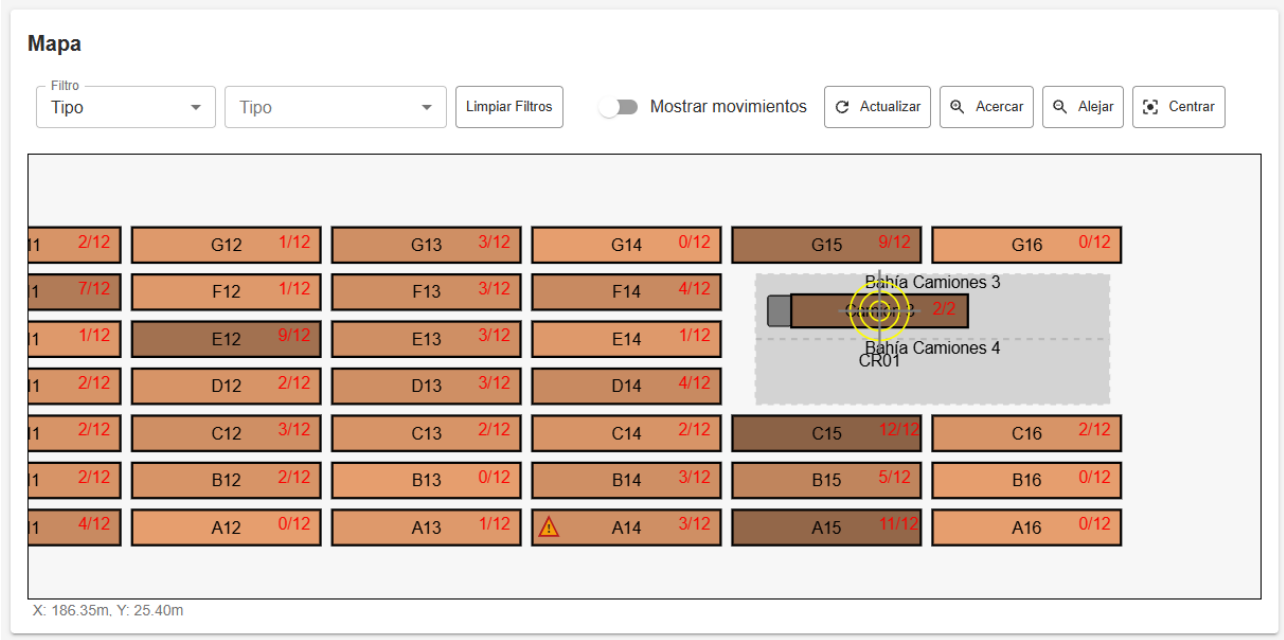


Figura 6.52: Grúa tras la realización de los movimientos.

En base a la prueba anterior, se determina que se cumple la especificación *SIM-01*.

6.5.2. Generación de materiales

La generación de materiales, especificación *SIM-02*, corresponde a la generación automática de materiales para su uso en el almacén. Su verificación se ha realizado implícitamente a través de todas las pruebas realizadas, ya que todos los materiales usados en las pruebas han sido creados mediante esta funcionalidad. Por lo tanto, se considera que se cumple la especificación.

6.5.3. Generación de programas

La generación de programas, correspondiente a la especificación *SIM-03*, es un proceso que ocurre automáticamente cuando se dan las condiciones en el almacén. Ya que los programas no pueden crearse manualmente, la prueba de su funcionamiento viene por la existencia de programas, que podemos visualizar desde el menú correspondiente. En la Figura 6.53 pueden observarse los programas que se han creado automáticamente en el almacén.

The screenshot shows the GenStock application interface. At the top, there is a navigation bar with the following items: GenStock, Transportes, Inventario, Plan de Movimientos, Programas, Mapa, and Administración. On the right side of the navigation bar, the user's name 'Daniel DU' and a profile icon are displayed.

The main content area is divided into two panels. The left panel is titled 'Programas' and contains a table with the following columns: Orden, Nombre, % Expedición, En parque, and Últ. Sec. Exp. The table lists two programs:

Orden	Nombre	% Expedición	En parque	Últ. Sec. Exp.
1	202604231	24%	48 / 72	24
2	202609121	0%	10 / 93	

At the bottom of the 'Programas' panel, there is a pagination control showing 'Filas por página: 100' and '1-2 de 2'.

The right panel is titled 'Elementos - 202609121' and contains a table with the following columns: Secuencia, Grupo, Material, Ubicación, and Altura. The table lists 93 elements, with the first 7 elements visible:

Secuencia	Grupo	Material	Ubicación	Altura
1	1	250000921	-	
2	2	260000408	A07	8
3	3	250000854	A07	4
4	4	250000321	G01	3
5	5	250001199	E10	6
6	6	250000894	-	
7	7	260000932	-	

At the bottom of the 'Elementos' panel, there is a pagination control showing 'Filas por página: 100' and '1-93 de 93'.

Figura 6.53: Menú de programas mostrando aquellos que están creados.

Se concluye que se cumple la especificación *SIM-03*.

6.5.4. Gestión de retenciones

La funcionalidad de gestión de retenciones, correspondiente a la especificación *SIM-04*, consiste en el marcado aleatorio de materiales como retenido, junto con la eliminación de retenciones bajo cierta probabilidad. La prueba realizada consiste en verificar, para un material en el que ha

aparecido una retención, como el que se observa en la Figura 6.54, que la retención desaparece transcurrido un tiempo.

	Id	MaterialKind	MaterialFamily	Length	Width	Thickness	Weight	DestinationId	Retention_Category	Retention_Description	ProductionDate
1	250000067	11	1	7907	1920	221	18413	NULL	2	Retention for ShortTerm	2025-05-01 18:56:43.7355728

Figura 6.54: Registro de material con retención.

Consultando la información del mismo material transcurrido un tiempo, en el que la simulación se ha ejecutado múltiples veces, se observa que el material ya no se encuentra retenido y, por lo tanto, no aparece cuando se consultan materiales con valor en el campo de retención. Por lo tanto, se considera que se cumple la especificación *SIM-04*.

6.6. Conclusiones

En este capítulo se ha presentado la estrategia empleada para verificar la aplicación GenStock mediante diferentes niveles de pruebas, para comprobar tanto el comportamiento individual de sus componentes como su funcionamiento conjunto. Las pruebas automatizadas permiten realizar comprobaciones repetibles sobre el código y detectar regresiones durante la evolución de la aplicación, mientras que las pruebas funcionales permiten verificar el comportamiento de las principales funcionalidades desde la perspectiva del usuario.

Esta estrategia se ha aplicado a las diferentes partes de la aplicación y se ha complementado con las pruebas de los servicios de simulación, permitiendo comprobar el funcionamiento de los componentes que intervienen en las operaciones principales de GenStock.

Con esta verificación se demuestra que la aplicación funciona y se establece una base para el análisis del comportamiento de los sistemas de soporte a la decisión. En este caso, la dificultad no se limita a comprobar que el código funciona correctamente, sino que es necesario determinar si las decisiones generadas son adecuadas ante las diferentes alternativas y estados posibles del almacén. La verificación específica de estos sistemas se aborda en el siguiente capítulo.

Capítulo 7

Verificación del soporte a la decisión

7.1. Introducción

Este capítulo presenta el proceso seguido para la verificación de los sistemas de soporte a la decisión integrados en GenStock. El objetivo es comprobar que, tanto las propuestas de ubicación como los planes de movimientos generados, son coherentes con las reglas y criterios establecidos en el Capítulo 4.

La problemática combinatoria descrita en el capítulo anterior adquiere una especial relevancia en estos sistemas, ya que una misma situación puede presentar múltiples alternativas, en función del estado del almacén, y la decisión adoptada puede condicionar las operaciones posteriores. La comprobación exhaustiva de todas las posibles configuraciones no resulta viable, por lo que la verificación se realiza mediante escenarios representativos, que permiten incrementar progresivamente la complejidad del problema y analizar el comportamiento del sistema ante diferentes situaciones.

Para ello, se presentan varios escenarios representativos de la operativa del almacén. Los ejemplos se presentan con un grado de complejidad creciente, comenzando por situaciones sencillas, que permiten comprobar el comportamiento individual de los criterios de la propuesta, y avanzando hacia escenarios en los que intervienen varios materiales o requieren de varios movimientos. De esta forma, se comprueba el comportamiento de los sistemas de soporte a la decisión ante diferentes situaciones de operación.

Este capítulo se estructura de la siguiente forma:

- **7.2 Configuración de reglas:** Realiza un inventario de las reglas y parámetros utilizados en los ejemplos.
- **7.3 Ejemplo liberar pila de un desbaste:** Introduce un ejemplo sencillo en el que la liberación de una pila cuenta con pocas opciones.
- **7.4 Ejemplo un desbaste y varias alternativas:** Presenta un escenario con mayor complejidad a la hora de tomar la decisión de ubicación.
- **7.5 Ejemplo varios desbastes:** Explora el funcionamiento del A^* , junto con el heurístico, a la hora de tomar una decisión sobre varios desbastes.

- **7.6 Ejemplo descarga de tren:** Ilustra el algoritmo de descomposición en subproblemas utilizado para la descarga de trenes.
- **7.7 Ejemplo carga de programa:** Demuestra el comportamiento del algoritmo diseñado para la carga de programas.

7.2. Configuración de reglas

El objetivo de este apartado es documentar la configuración utilizada, de forma que el lector pueda reproducir los resultados obtenidos. Las siguientes reglas estaban activadas:

- Reglas de validación:
 - Máximo de elementos 4.6.1.
 - Seguridad de pila 4.6.2.
 - Movimientos permitidos 4.6.3.
- Reglas globales:
 - Pilas vacías 4.7.1.
 - Distancia 4.7.2.
 - Tamaño de ganchada 4.7.3.
- Reglas locales:
 - Programas 4.8.1.
 - Destino 4.8.3.
 - Tipo 4.8.4.
 - Familia 4.8.5.
 - Anchos 4.8.6.

A continuación, para las reglas parametrizables, se describen los valores utilizados para las pruebas.

Tabla 7.1: Configuración utilizada para la regla «Pilas vacías».

Parámetro	Valor	Descripción
WidthDiffMaxPenalty	300	Penalización máxima que se puede obtener por el criterio de ancho de pila.
EmptyPileUsageMaxPenalty	300	Penalización máxima que se puede obtener por el criterio de pilas vacías.

Tabla 7.2: Configuración utilizada para la regla «Distancia».

Parámetro	Valor	Descripción
YardWidth	280	Representa la anchura total del almacén.
YardHeight	50	Representa la altura total del almacén.
MaxPenaltyX	40	Representa la máxima penalización que puede obtener un movimiento sobre el eje X.
MaxPenaltyY	7	Representa la máxima penalización que puede obtener un movimiento sobre el eje Y.

Tabla 7.3: Configuración utilizada para la regla «Programas».

Parámetro	Valor	Descripción
BurriedMorePrioritaryProgramsPenalty	4000	Representa la penalización por cada nivel de profundidad.
MixedProgramsInMoveBasePenalty	2000	Representa la penalización por cada programa mezclado.
GroupsDispersionsPenalty	10	Representa la penalización en función de la dispersión de los grupos.
UnorderedGroupsPenalty	100	Representa la penalización por cada grupo que no está ordenado.
IgnoreGroupsPenaltyAfter	14	Configura el número de grupos a partir del cual se ignora el orden de grupos para un desbaste.
BurriedExceededPenalty	1500	Representa la penalización que obtienen los casos que exceden el enterramiento máximo.
MaxBurrialAllowed	4	Configura el máximo de enterramiento permitido.
BurriedProgramPenaltyPerItem	250	Representa la penalización que se obtiene por cada elemento enterrado.
BurriedNonProgramPenaltyPerItem	200	Representa la penalización que se obtiene por cada elemento bloqueado.
BurriedProgramsPenalty	800	Representa la penalización por cada programa presente en la pila.

Tabla 7.4: Configuración utilizada para la regla «Destino».

Parámetro	Valor	Descripción
MixedInMoveMaxPenalty	300	Representa la máxima penalización por mezcla de destino en el movimiento.
MoveAndTopMismatchPenaly	100	Representa la penalización obtenida cuando el tope de la pila no coincide con la base del movimiento.
FinalStateMaxPenalty	700	Representa la máxima penalización que se puede obtener por el estado final de la pila.

Tabla 7.5: Configuración utilizada para la regla «Tipo».

Parámetro	Valor	Descripción
MixedTypesInMovePenalty	6000	Representa la máxima penalización por mezcla de tipos en el movimiento.
MixedTypesMaxPenalty	5000	Representa la penalización obtenida cuando el tope de la pila no coincide con la base del movimiento.
MaxMixedTypesInPile	3	Representa el número máximo de tipos con el que se obtendrá la máxima penalización de mezcla.

Tabla 7.6: Configuración utilizada para la regla «Familia».

Parámetro	Valor	Descripción
MixedInMoveMaxPenalty	300	Representa la máxima penalización por mezcla de familias en el movimiento.
MoveAndTopMismatchPenaly	100	Representa la penalización obtenida cuando el tope de la pila no coincide con la base del movimiento.
FinalStateMaxPenalty	700	Representa la máxima penalización que se puede obtener por el estado final de la pila.

Todos los valores presentados se determinaron según la importancia relativa de los criterios y realizando pruebas de comportamiento. En las mismas, se simuló el llenado del parque y desde varios estados iniciales y se revisó manualmente la idoneidad de los resultados hasta obtener una calidad aceptable.

Finalmente, la configuración utilizada excluye las pilas (G05, G06 y G16) para su uso por

parte de la propuesta de ubicación. Esta reserva permite garantizar un pequeño espacio de almacenamiento para las operaciones de reordenación de material, necesarias para la carga de programas.

7.3. Ejemplo liberar pila de un desbaste

Este primer ejemplo tiene un doble objetivo. Por un lado, mostrar el flujo completo de utilización de la propuesta de ubicación dentro de la aplicación y, por otro, explorar la liberación de una pila de un desbaste, con el objetivo de compactar el almacenamiento. Este caso pretende vaciar una pila ocupada por un único desbaste para dejarla disponible para futuros movimientos, consolidando el material en una pila existente. Las propiedades del desbaste elegido pueden observarse en la Tabla 7.7.

Tabla 7.7: Descripción del desbaste usado en el Ejemplo 7.3.

Propiedad	Valor
Identificador	250001724
Programa - Secuencia - Grupo	No tiene
Destino	No tiene
Ancho	1800mm
Largo	10,396m
Tipo	Refosforado
Familia	Alloys

Como se muestra en la Figura 7.1, es el único desbaste que se encuentra en la pila «D11».

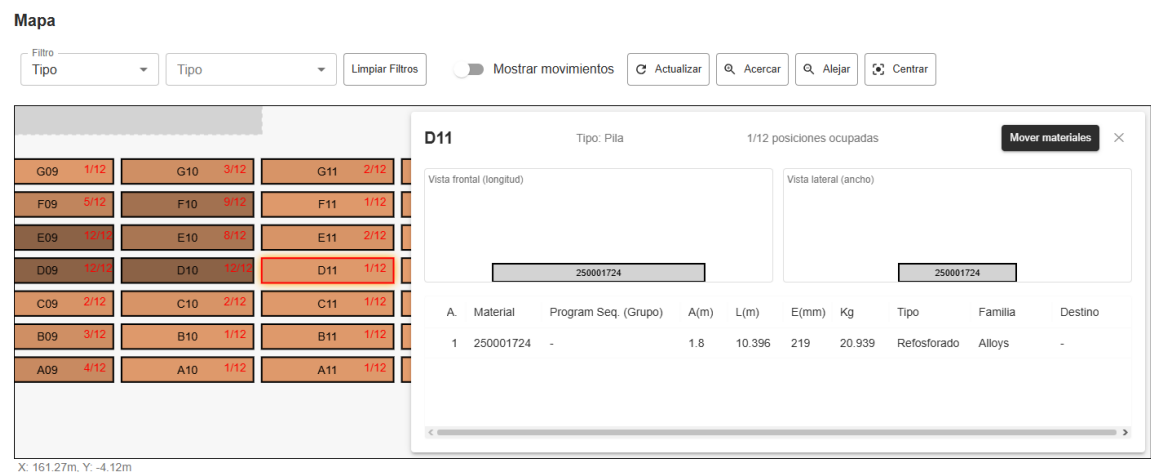


Figura 7.1: Ubicación inicial del desbaste en el Ejemplo 7.3.

Se procede a la creación del plan de movimientos para la liberación de la pila «D11». En la Figura 7.2 se observa el formulario utilizado para la creación.

Crear Plan de Movimientos

Nombre *

Ejemplo 1

Tipo *

Liberar Pilas

Elementos

+ Agregar Elemento

De *

D11

Cerrar

Crear

Figura 7.2: Creación del plan de movimientos en el Ejemplo 7.3.

En la Figura 7.3 se visualiza el plan de movimientos que se ha calculado, formado por un único movimiento, para mover el desbaste desde la pila «D11» hasta la altura seis de la pila «F06», propuesta por el algoritmo.

Plan de Movimientos

+ Crear

Actualizar

Orden	Nombre	Estado	Tipo
1	Ejemplo 1	Pendiente	Liberar Pilas

Filas por página: 100 1-1 de 1

Operaciones - Ejemplo 1

1 movimientos

Orden: 1

Pendiente

(1 Elemento)

De	A	ID del Material
D11 (A1)	F06 (A6)	250001724

Figura 7.3: Plan de movimiento con propuesta para el Ejemplo 7.3.

La Figura 7.4 demuestra la utilidad del mapa de la aplicación para visualizar los movimientos pendientes. El sombreado azul corresponde al filtro aplicado de tipo «Refosforado», que es el tipo correspondiente al desbaste.

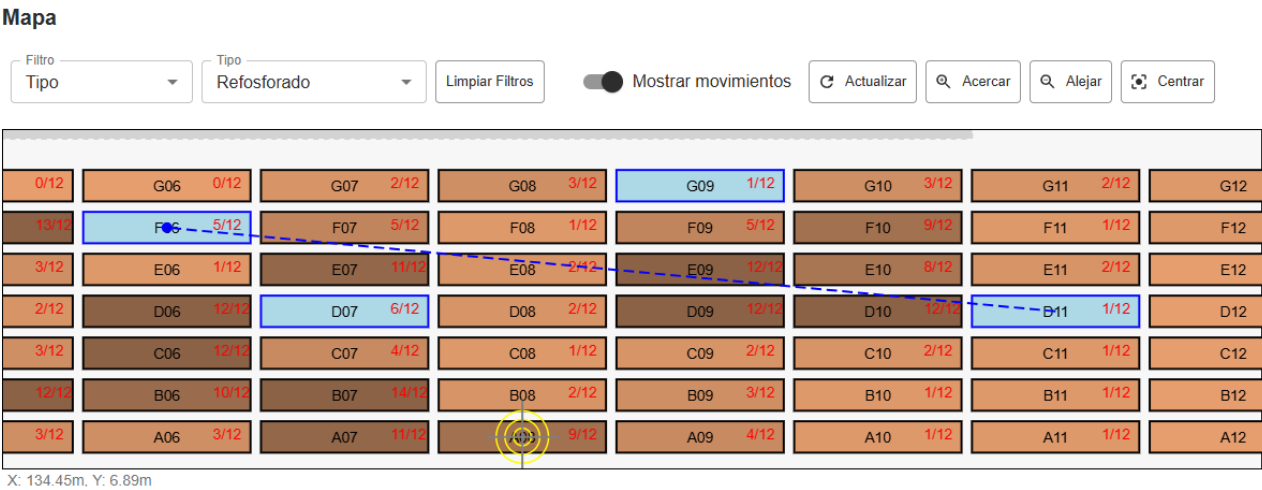


Figura 7.4: Movimiento en mapa del Ejemplo 7.3.

Analizando la información de diagnóstico disponible para la propuesta, descrita en la Sección 5.5, se determina que la ejecución ha durado 9 milisegundos y ha explorado 63 opciones sobre las 96 pilas existentes en el almacén. En la Tabla 7.8 se analizan los resultados globales del diagnóstico.

Tabla 7.8: Datos de diagnóstico propuesta ubicación del Ejemplo 7.3.

Propiedad	Valor	Descripción
Total vecinos generados	63	Corresponde al número de pilas que se evaluaron como candidatas.
Nodos descartados (no permitidos)	51	Sobre los vecinos generados representa el número de nodos que no cumplieron alguna de las reglas de validación.
Nº Ejecuciones Heurístico	13	Las ejecuciones corresponden a la expansión del nodo inicial y el cálculo heurístico para las $63 - 51 = 12$ pilas candidatas.
Nodos expandidos	1	Corresponde al nodo inicial, ya que el resto son estados finales que no se expanden.

En este caso, el problema únicamente requiere un movimiento, por lo que el árbol de búsqueda tiene profundidad uno. El algoritmo genera un nodo sucesor por cada pila candidata y selecciona aquella con menor coste total, sin necesidad de expandir estados adicionales.

Como se observa en la Tabla 7.9 de las doce pilas que resultan válidas, solo una comparte tipo con el desbaste.

Tabla 7.9: Pilas permitidas del Ejemplo 7.3.

Pila	Comentario
A10	
A11	
B11	
C04	
C07	
C16	
D12	
F06	Pila con mismo tipo (refosforado)
F07	
F12	
G07	
G08	

La mayoría de las pilas permitidas no contienen desbastes del mismo tipo. Esto obliga al algoritmo a valorar el compromiso entre mantener la homogeneidad del almacenamiento y el resto de criterios considerados.

En la Figura 7.5 se observan algunos de los motivos por los que se han descartado pilas, principalmente porque la ubicación en las mismas habría generado problemas de seguridad.

PileName	Move	RuleName	Allowed	Reason
A02	250001724	MaxItemsRule	1	NULL
A02	250001724	SafetyRule	0	Move top material (1800mm) exceeds width 250 compared to pile top material (800mm)
A03	250001724	MaxItemsRule	1	NULL
A03	250001724	SafetyRule	0	Move top material (1800mm) exceeds width 250 compared to pile top material (1450mm)
A05	250001724	MaxItemsRule	1	NULL
A05	250001724	SafetyRule	0	Move top material (10396mm) exceeds length 2000 compared to pile top material (6831mm)

Figura 7.5: Pilas descartadas del Ejemplo 7.3.

En la Figura 7.6 podemos observar la evaluación de cada una de las reglas para las dos pilas más prometedoras.

PileName	GlobalRule	RuleName	Penalty
D12	1	DistanceRule	1,82142857142857
D12	1	EmptyPileRule	357,291666666667
D12	1	GripsRule	40
D12	0	BlockingProgramRule	0
D12	0	WidthPenaltyRule	0
D12	0	ProgramRule	0
D12	0	KindRule	0
D12	0	DestinationRule	0
D12	0	FamilyRule	0
F06	1	DistanceRule	10,7071428571429
F06	1	EmptyPileRule	0
F06	1	GripsRule	40
F06	0	BlockingProgramRule	0
F06	0	WidthPenaltyRule	112,5
F06	0	ProgramRule	0
F06	0	KindRule	0
F06	0	DestinationRule	0
F06	0	FamilyRule	0

Figura 7.6: Pilas evaluadas del Ejemplo 7.3.

Aunque «D12» constituye una ubicación válida, recibe una penalización mayor al requerir la apertura de una pila. En cambio, «F06» presenta una mejor combinación, debido a que está formada por una composición homogénea del mismo tipo que el desbaste.

La penalización total obtenida se recoge en la Tabla 7.10, donde se observa que «F06» es la mejor opción.

Tabla 7.10: Penalización pilas más prometedoras del Ejemplo 7.3.

Pila	Penalización
D12	399,113
F06	163,207

Una vez se ha completado el movimiento, es posible utilizar el mapa de la aplicación para visualizar el estado final de la pila. La Figura 7.7 muestra que la pila es uniforme en tipo y sus anchos se encuentran en el rango [1,8 - 2,04] m, siguiendo piramidalidad segura.

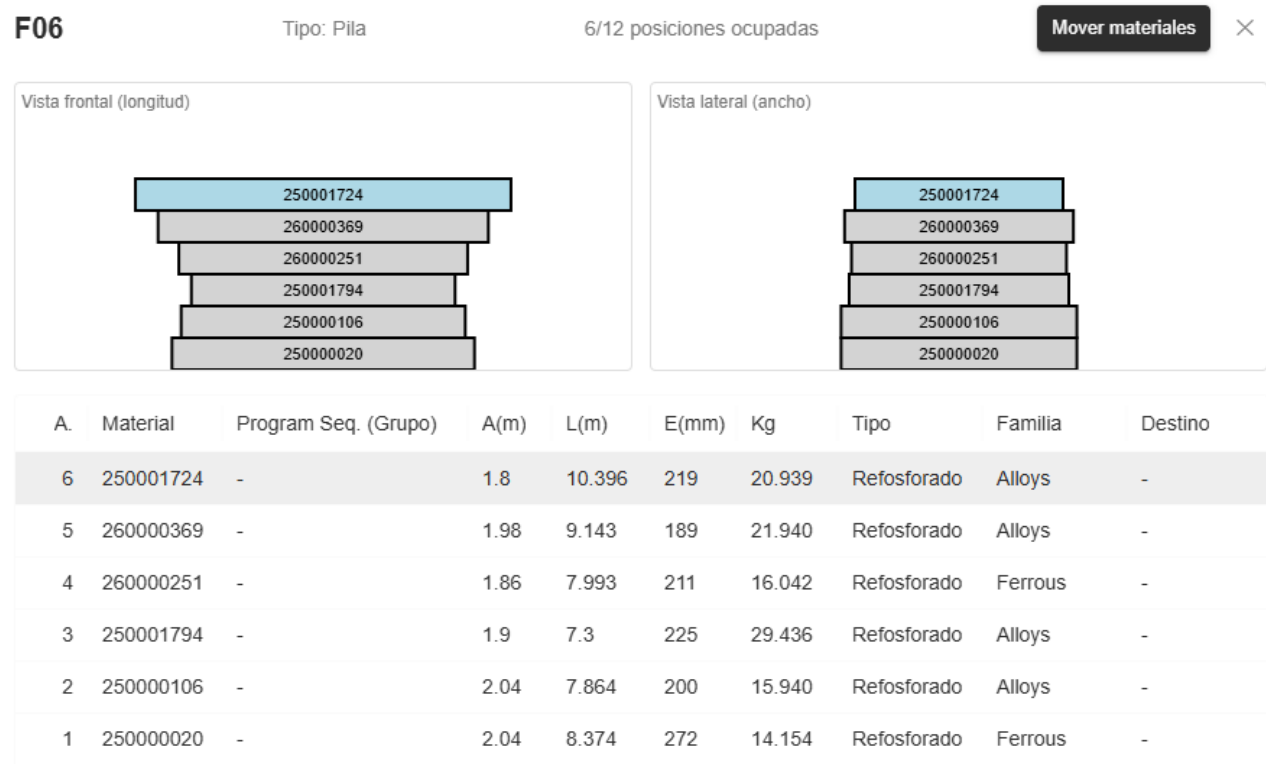


Figura 7.7: Estado final pila tras movimiento del Ejemplo 7.3.

Este ejemplo muestra el funcionamiento de la propuesta en un problema de profundidad uno. El algoritmo descarta la mayoría de las alternativas mediante las reglas de validación y únicamente evalúa el resto. Entre ellas, se identifica como mejor opción la pila «F06» que comparte tipo y anchos similares al desbaste, obteniendo una solución coherente con los criterios definidos durante el diseño. De esta forma, se consigue liberar la pila «D11» y compactar el almacenamiento, sin incumplir ninguna restricción de seguridad.

7.4. Ejemplo un desbaste y varias alternativas

Este segundo ejemplo evalúa la propuesta en un escenario donde un único desbaste ocupa una pila completa y existen varias ubicaciones válidas para su traslado. El objetivo es liberar dicha pila, para favorecer la compactación del almacén, y analizar la capacidad del algoritmo para seleccionar la mejor alternativa, entre varias opciones viables. Las propiedades del desbaste elegido pueden observarse en la Tabla 7.11.

Tabla 7.11: Descripción del desbaste usado en el Ejemplo 7.4.

Propiedad	Valor
Identificador	260000739
Programa - Secuencia - Grupo	No tiene
Destino	No tiene
Ancho	800mm
Largo	6,824m
Tipo	Arranque
Familia	NonFerrous

Como se muestra en la Figura 7.8 es el único desbaste que se encuentra en la pila «B05».

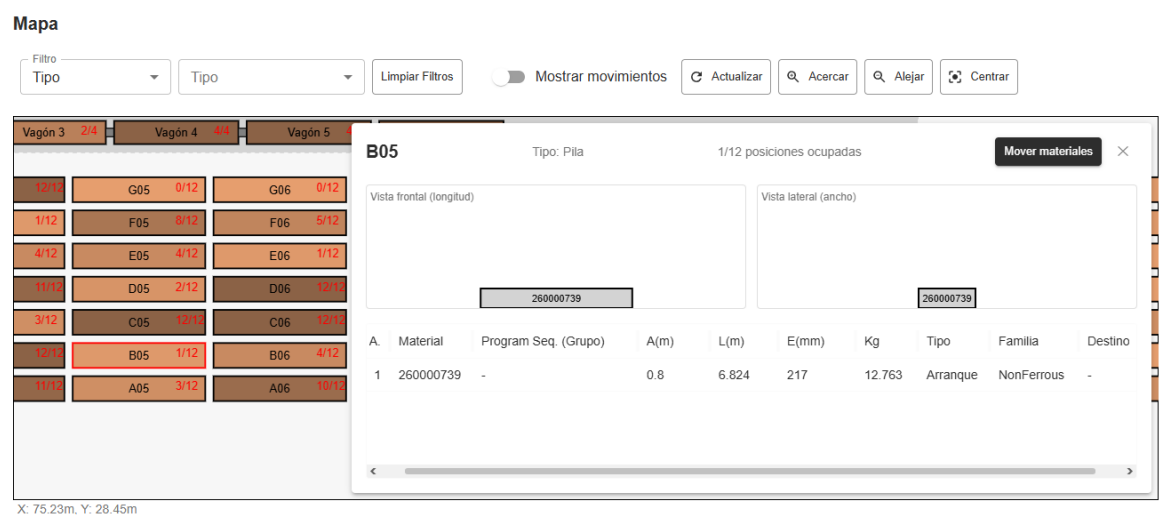


Figura 7.8: Ubicación inicial del desbaste en el Ejemplo 7.4.

En la Tabla 7.12 se puede observar el plan de movimientos que se ha calculado, formado por un único movimiento, para mover el desbaste desde la pila «B05» hasta la altura nueve de la pila «F05», propuesta por el algoritmo.

Tabla 7.12: Tabla de movimientos propuestos del Ejemplo 7.4.

Orden	De	A	Material
1	B05 (A1)	F05 (A9)	260000739

La Figura 7.9 muestra el mapa de la aplicación con el filtro del tipo del desbaste («Arranque») para visualizar el movimiento pendiente y el resto de pilas potencialmente candidatas.

Mapa

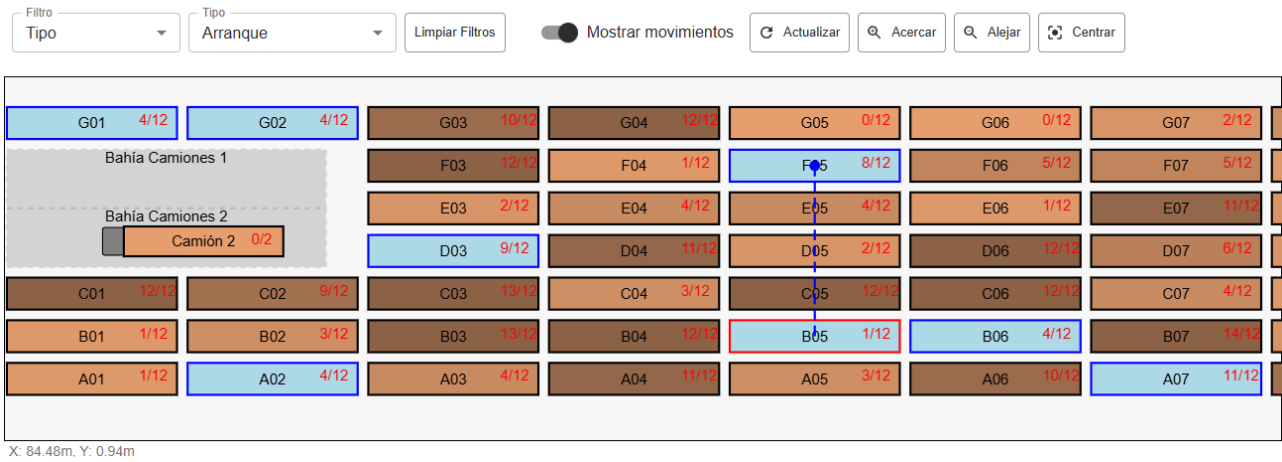


Figura 7.9: Movimiento en mapa del Ejemplo 7.4.

Analizando la información de diagnóstico disponible para la propuesta, descrita en la Sección 5.5, se determina que la ejecución ha durado 11 milisegundos y ha explorado 66 opciones sobre las 96 pilas existentes en el almacén. En la Tabla 7.13 se analizan los resultados globales del diagnóstico.

Tabla 7.13: Datos de diagnóstico propuesta ubicación del Ejemplo 7.4.

Propiedad	Valor	Descripción
Total vecinos generados	66	Corresponde al número de pilas que se evaluaron como candidatas.
Nodos descartados (no permitidos)	13	Sobre los vecinos generados representa el número de nodos que no cumplieron alguna de las reglas de validación.
Nº Ejecuciones Heurístico	54	Las ejecuciones corresponden a la expansión del nodo inicial y el cálculo heurístico para las $66 - 13 = 53$ pilas candidatas.
Nodos expandidos	1	Corresponde al nodo inicial, ya que el resto son estados finales que no se expanden.

En este caso el problema únicamente requiere un movimiento, por lo que el árbol de búsqueda tiene profundidad uno. El algoritmo genera un nodo sucesor por cada pila candidata y selecciona aquella con menor coste total, sin necesidad de expandir estados adicionales.

Aunque el algoritmo evaluó 53 pilas, únicamente se muestran las cinco con menor penalización, por ser las más relevantes para justificar la decisión final. La penalización total que han obtenido se recoge en la Tabla 7.14, donde se observa que «F05» es la mejor opción.

Tabla 7.14: Penalización pilas más prometedoras del Ejemplo 7.4.

Pila	Penalización
A07	170,417
D03	77,227
A02	47,846
B06	42,429
F05	42,24

A continuación, en la Tabla 7.15 observamos los resultados de la evaluación para las cinco pilas más prometedoras. Se muestran los valores obtenidos del cálculo o un guion, cuando la regla no se ha evaluado debido a que no se cumplían los criterios para su evaluación. Por ejemplo, la regla de programas solo se aplica cuando el movimiento o la pila contienen materiales pertenecientes a un programa.

Tabla 7.15: Resultados de evaluación pilas más prometedoras del Ejemplo 7.4.

Regla / Pila	A07	D03	A02	B06	F05
Pila vacía	0	0	0	0	0
Distancia	5,417	5,977	7,846	2,429	2,24
Tamaño de Ganchada	40	40	40	40	40
Programas	-	-	-	-	-
Cubrir Programa	-	-	-	-	-
Destino	-	-	-	-	-
Tipo	0	0	0	0	0
Familia	-	-	-	-	-
Anchos	125	31,25	0	0	0
Total	170,417	77,227	47,846	42,429	42,24

Del análisis de los resultados se identifican los siguientes aspectos:

- **A07:** Obtiene una penalización de 125 en la regla de anchos, debido a que el tope de la pila tiene un ancho de 700 mm, por lo que ubicar un material de 800 mm implica un crecimiento de ancho en el tope, lo que no constituye una situación deseable.
- **D03:** Obtiene una penalización de 31,25 en la regla de anchos, debido a que el tope de la pila tiene un ancho de 900 mm, lo que implica un decrecimiento. Esta situación se penaliza débilmente.

- **A02:** Tiene material de ancho 800 mm en el tope por lo que no se le penaliza por ancho, sin embargo, se encuentra a mayor distancia que otras pilas.
- **B06 y F05:** Ambas tienen un ancho de 800 mm en el tope y la única diferencia entre ambas viene por la regla de la distancia. La distancia entre centros es de 17 m para «B06» y de 16 m «F05». Aunque ambas pilas satisfacen las mismas restricciones de tipo y ancho, la regla de distancia permite priorizar la alternativa que implica un menor coste de desplazamiento de la grúa. De acuerdo con el diseño de la regla de distancia 4.7.2, se justificó que los movimientos del carro de la grúa (eje Y) deberían ser menos costosos.

Este ejemplo muestra que, incluso cuando existen varias alternativas válidas, la propuesta no selecciona una ubicación arbitraria, sino que aplica, de forma consistente, las reglas de evaluación definidas en el diseño. La solución elegida minimiza la penalización total, al combinar compatibilidad de ancho con un menor coste de desplazamiento, permitiendo además liberar completamente la pila original y contribuir a la compactación del almacén. La penalización de tipo no ha aplicado a las cinco pilas más prometedoras, debido a que éstas eran uniformes en sus tipos de material.

7.5. Ejemplo varios desbastes

Este tercer ejemplo evalúa la propuesta en un escenario de descarga de un vagón, cuando existen varias ubicaciones válidas para cada uno de los cuatro desbastes existentes en el mismo. El objetivo es vaciar el vagón, como parte de la descarga del transporte, y analizar la capacidad del algoritmo para encontrar la mejor solución global. Las propiedades de los desbastes elegidos pueden observarse en la Tabla 7.16.

Tabla 7.16: Descripción del desbaste usado en el Ejemplo 7.5.

Propiedad	Altura 4	Altura 3	Altura 2	Altura 1
Identificador	250000010	250000699	260000106	250001920
Programa - Secuencia - Grupo	No tiene	202604231 S33 (G33)	202604231 S67 (G67)	202604231 S69 (G69)
Destino	No tiene	No tiene	No tiene	No tiene
Ancho	1490mm	1870mm	2330mm	2300mm
Largo	6,344m	6,571m	8,074m	8,57m
Tipo	Resto	Microaleado	Microaleado	Microaleado
Familia	Alloys	NonFerrous	NonFerrous	NonFerrous

Como se muestra en la Figura 7.10 son los desbastes que se encuentran en la pila «Vagón 1».

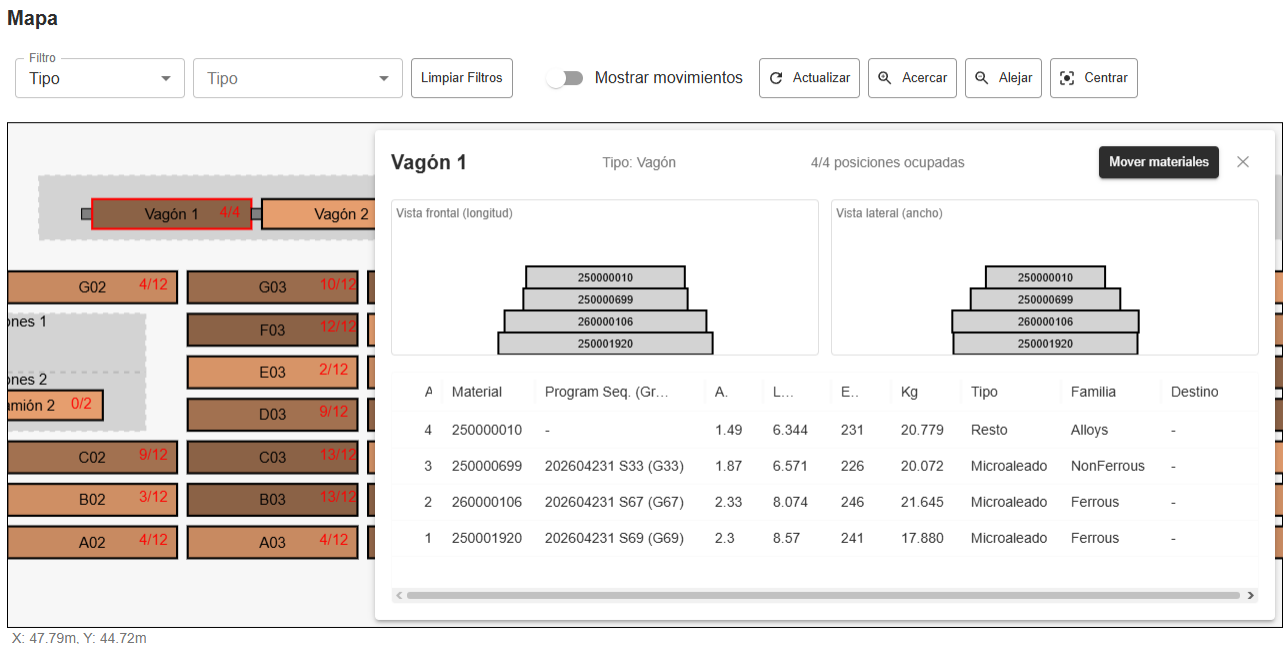


Figura 7.10: Ubicación inicial de desbastes en el Ejemplo 7.5.

En la Tabla 7.17 se puede observar el plan de movimientos que se ha calculado, formado por tres movimientos.

Tabla 7.17: Tabla de movimientos propuestos del Ejemplo 7.5.

Orden: 1		
De	A	ID del Material
Vagón 1 (A4)	E06 (A2)	250000010
Vagón 1 (A3)	E06 (A1)	250000699

Orden: 2		
De	A	ID del Material
E06 (A2)	C04 (A4)	250000010

Orden: 3		
De	A	ID del Material
Vagón 1 (A2)	F11 (A3)	26000106
Vagón 1 (A1)	F11 (A2)	250001920

La Figura 7.11 demuestra la utilidad del mapa de la aplicación para visualizar los movimientos pendientes. El sombreado azul corresponde al filtro aplicado para el programa «202604231» al que pertenecen tres de los cuatro desbastes.

Mapa

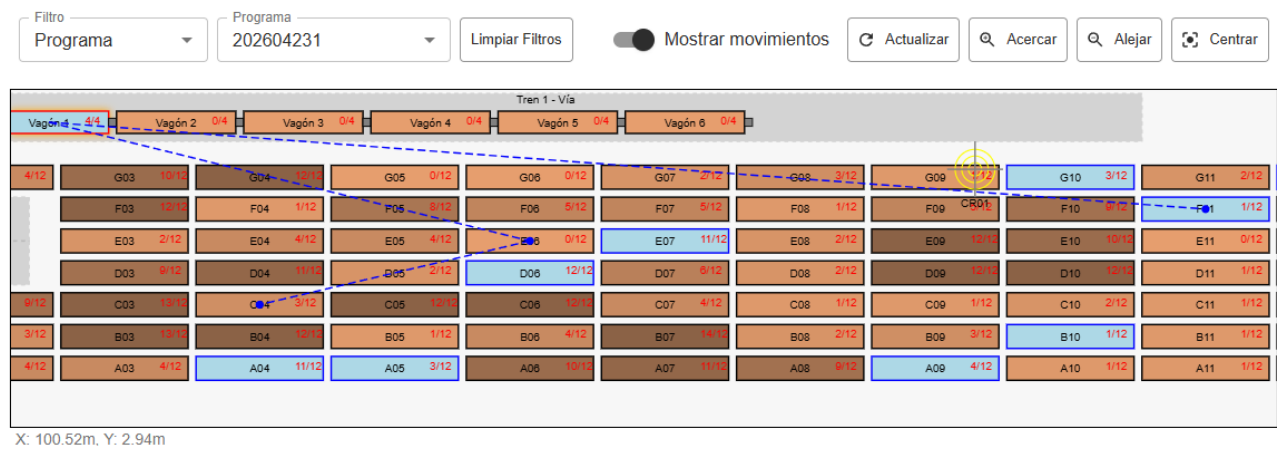


Figura 7.11: Movimientos en mapa del Ejemplo 7.5.

Analizando la información de diagnóstico disponible para la propuesta, descrita en la Sección 5.5, se determina que la ejecución ha durado 81 milisegundos y ha explorado 1039 opciones sobre las 96 pilas existentes en el almacén. En la Tabla 7.18 se analizan los resultados globales del diagnóstico.

Tabla 7.18: Datos de diagnóstico propuesta ubicación del Ejemplo 7.5.

Propiedad	Valor	Descripción
Total vecinos generados	1039	Corresponde al número de pilas que se evaluaron como candidatas.
Nodos descartados (no permitidos)	809	Sobre los vecinos generados representa el número de nodos que no cumplieron alguna de las reglas de validación.
Nº Ejecuciones Heurístico	231	Las ejecuciones corresponden a la expansión del nodo inicial y el cálculo heurístico para las $1039 - 809 = 230$ pilas candidatas.
Nodos expandidos	5	Compuesto por el nodo inicial y cuatro exploraciones adicionales.

La Figura 7.12 muestra una representación simplificada del árbol de búsqueda generado durante la ejecución del algoritmo A*. Con el objetivo de mantener la figura legible, únicamente se representan mediante nodos circulares aquellos estados que son efectivamente expandidos por el algoritmo, es decir, los estados que, en algún momento, son seleccionados de la lista abierta por presentar el menor valor de la función de evaluación f .

Los nodos triangulares representan alternativas de movimiento generadas durante la expansión de un estado. Cada uno de ellos, resume un posible movimiento y muestra el número de estados sucesores que se habrían generado, si dicha alternativa hubiese sido seleccionada para su

expansión. Esta información se indica mediante la etiqueta *Nodos Hijos*. Dado que estos nodos nunca alcanzan el menor valor de f dentro de la lista abierta, no son expandidos y, por tanto, sus descendientes no se representan en la figura.

Esta representación permite visualizar el orden real de exploración seguido por A*, sin necesidad de mostrar el árbol completo de búsqueda, que en este ejemplo estaría formado por 231 nodos de estado. De este modo, la figura resulta más fácil de interpretar, al tiempo que conserva la información necesaria para comprender las decisiones tomadas por el algoritmo durante la búsqueda.

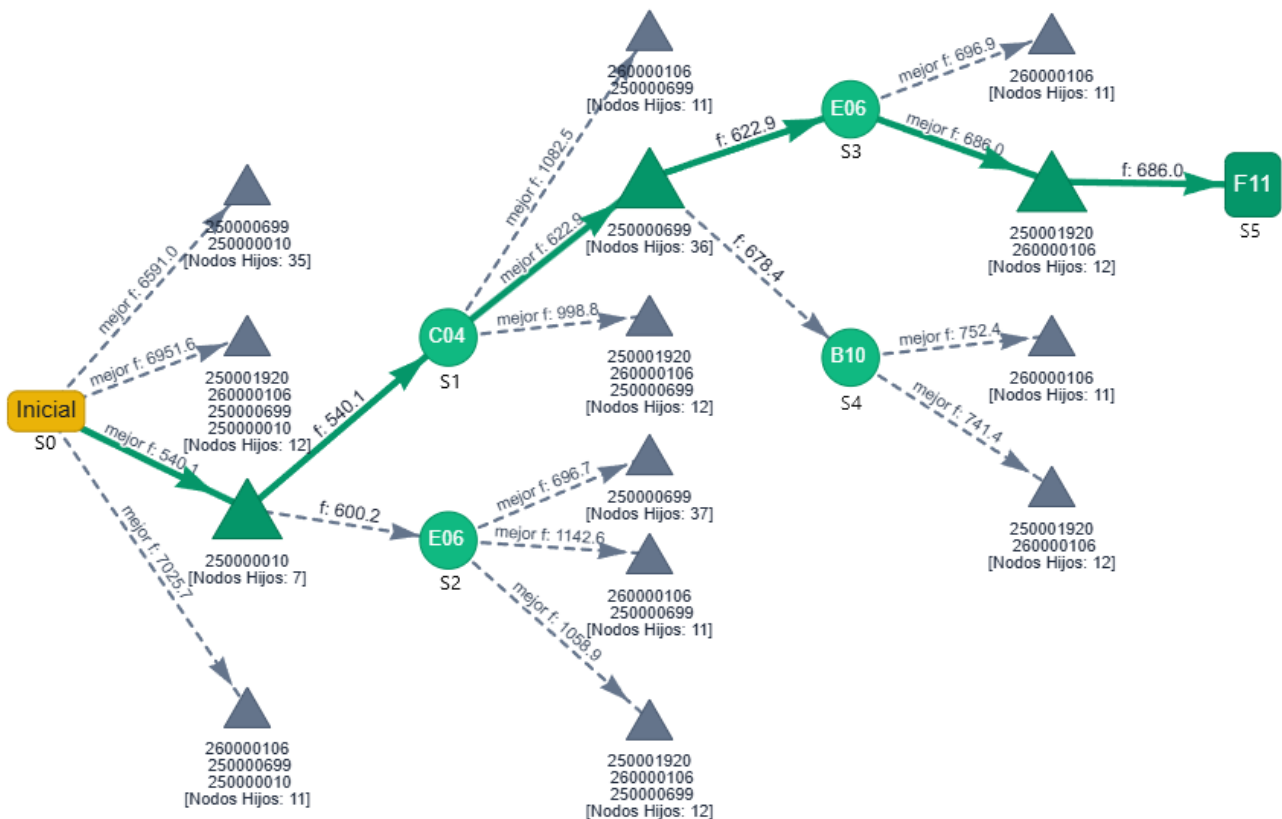


Figura 7.12: Flujo ejecución A* del Ejemplo 7.5.

Puede observarse que, aunque durante la búsqueda se generan numerosas alternativas, únicamente una pequeña parte de ellas llega a expandirse. En varias ocasiones el algoritmo cambia de rama, al detectar en la lista abierta otro estado con un menor valor de f , posponiendo la exploración de alternativas que inicialmente parecían prometedoras. Este comportamiento ilustra el funcionamiento de A* y permite reducir, significativamente, el número de estados que es necesario explorar para alcanzar la solución óptima.

Con el fin de facilitar el seguimiento de la búsqueda, cada estado expandido mostrado en la Figura 7.12 se identifica mediante un código S_i , que se utiliza también en la Tabla 7.19 para describir el orden de exploración seguido por el algoritmo.

Tabla 7.19: Resumen de la exploración del árbol de búsqueda del Ejemplo 7.5.

Iteración	Estado expandido	Movimiento representado por el estado	Resultado de la expansión
1	S0	Estado inicial.	Se generan cuatro alternativas de movimientos para mover los desbastes de la pila, agrupando un total de 65 estados. El mejor sucesor es el estado S1, correspondiente al movimiento del desbaste 250000010 a la pila «C04», con un coste de $f = 540, 1$.
2	S1	Movimiento del desbaste 250000010 a la pila «C04».	Se generan tres alternativas de movimientos para continuar la búsqueda, agrupando un total de 59 estados. El mejor sucesor es el estado S3, correspondiente al movimiento del desbaste 250000699 a la pila «E06», con un coste de $f = 622, 9$.
3	S2	Movimiento del desbaste 250000010 a la pila «E06».	Este estado es seleccionado antes que S3 por presentar un menor valor de f . Se generan tres alternativas de movimientos para mover los tres desbastes pendientes (250001920, 260000106 y 250000699), agrupando un total de 60 estados. Sin embargo, todos los estados generados presentan un coste superior al más prometedor del algoritmo, siendo el mejor de ellos $f = 696, 7$.
4	S3	Movimiento del desbaste 250000699 a la pila «E06», partiendo del estado S1.	Se generan dos alternativas de movimientos para continuar la búsqueda para los dos desbastes pendientes (250001920, 260000106), agrupando un total de 23 estados. La expansión genera una alternativa más favorable, el estado S5, en el que los desbastes 250001920 y 260000106 se trasladan conjuntamente a la pila «F11», obteniendo un coste de evaluación $f = 686, 0$.
5	S4	Movimiento del desbaste 250000699 a la pila «B10», partiendo del estado S1.	Este estado es seleccionado antes que S5 por presentar un menor valor de f . Se generan dos alternativas de movimientos para continuar la búsqueda para los dos desbastes pendientes (250001920, 260000106), agrupando un total de 23 estados. Sin embargo, todos los estados generados presentan un coste superior al más prometedor del algoritmo, siendo el mejor de ellos $f = 741, 4$.
6	S5	Movimiento de los desbastes 250001920 y 260000106 a la pila «F11».	La expansión alcanza un estado objetivo con un coste total de $f = 686, 0$, finalizando la búsqueda y obteniendo la secuencia de movimientos mostrada en la Tabla 7.17.

La Tabla 7.19 resume el orden de expansión de los nodos. A continuación se describe cada iteración justificando las decisiones de cada nodo.

7.5.1. Estado S0

Este estado corresponde al inicio del problema, antes de realizar ningún movimiento. A partir de este estado, el algoritmo genera todas las alternativas válidas para mover los desbastes del vagón, representados en la Figura 7.10.

7.5.1.1. Evaluación del estado

El algoritmo A* evalúa cada estado mediante la función:

$$f(n) = g(n) + h(n) \quad (7.1)$$

donde $g(n)$ representa el coste acumulado desde el estado inicial y $h(n)$ la estimación heurística del coste restante, calculada mediante la implementación descrita en la Sección 5.4.3 .

En la Tabla 7.20 se muestran los resultados de la función de evaluación.

Tabla 7.20: Evaluación del estado S0 en el Ejemplo 7.5.

Componente	Valor	Descripción
Coste acumulado (g)	0	Al tratarse del nodo inicial, no existe coste acumulado.
Heurística (h)	121,98	
Función de evaluación ($f = g + h$)	121,98	

7.5.1.2. Resultado de la expansión

Durante la expansión del estado S0 se generan 65 estados, agrupados en las cuatro ganchadas posibles para extraer el material, que se presentan en la Tabla 7.21.

El movimiento del desbaste 250000010 a la pila «C04», que en adelante denominaremos estado S1, presenta el menor coste, con un valor de $f = 540,1$, por lo que será el siguiente estado seleccionado para su expansión.

Tabla 7.21: Expansión del estado S0 en el Ejemplo 7.5.

Ganchadas	Número de estados	Mejor puntuación (f)
2500000010, 2500000699, 2600000106, 2500001920	12	6951,6
2500000010, 2500000699, 2600000106	11	7025,7
2500000010, 2500000699	35	6591,0
2500000010	7	540,1

7.5.2. Estado S1

Este estado está compuesto por el movimiento del desbaste 2500000010, ubicado en el tope de la pila inicial, a la pila «C04». Tras la expansión del nodo inicial, el estado S1 presenta el menor valor de la función de evaluación entre todos los estados de la lista abierta, por lo que es seleccionado para su expansión.

La información de las funciones de coste para el estado actual se encuentran recogidas en la Tabla 7.22.

Tabla 7.22: Evaluación del estado S1 en el Ejemplo 7.5.

Componente	Valor
Coste acumulado (g)	403,13
Heurística (h)	136,97
Función de evaluación ($f = g + h$)	540,1

7.5.2.1. Configuración de la pila

La Figura 7.13 muestra el estado de la pila «C04» tras aplicar el movimiento del estado S1. El material del movimiento se muestra seleccionado en azul.



Figura 7.13: Configuración pila «C04» estado S1 en el Ejemplo 7.5.

El resultado de evaluar los criterios de la propuesta previo al movimiento, obtiene los resultados recogidos en la Tabla 7.23.

Tabla 7.23: Evaluación del movimiento realizado para alcanzar S1 en el Ejemplo 7.5.

Regla	Puntuación
Pila vacía	0
Distancia	6,88
Tamaño de Ganchada	40
Programas	-
Cubrir Programa	-
Destino	-
Tipo	-
Familia	0
Anchos	356,25
Total	403,13

Donde se observa que la ubicación encaja perfectamente, al quedar compuesta por cuatro materiales que comparten el criterio familia. Sin embargo, el ancho no es la mejor opción ya que hay una diferencia de 380 mm, que es válida y segura debido al criterio de piramidalidad descendente.

Analizando el resto de pilas del parque, únicamente se observa la pila «C11» con material de

familia, sin embargo al explorarla se observa:

- En cuanto al criterio de anchos, está compuesta por un desbaste de ancho 2370 mm lo que ocasionaría una diferencia de 880 mm.
- En cuanto a la distancia, está más alejada del origen del movimiento.

Debido a lo anterior, la pila obtiene una penalización de 888,88 por lo que claramente la opción elegida representa una solución mejor.

7.5.2.2. Resultado de la expansión

Durante la expansión del estado S1 se generan 59 estados, agrupados en las tres ganchadas posibles para extraer el material, que se presentan en la Tabla 7.24.

Tabla 7.24: Expansión del estado S1 en el Ejemplo 7.5.

Ganchadas	Número de estados	Mejor puntuación (f)
250000699, 260000106, 250001920	12	998,8
250000699, 260000106	11	1082,5
250000699	36	622,9

En este punto, el mejor nodo existente no es un sucesor de S1, sino el movimiento del desbaste 250000010 a la pila «E06», que en adelante denominaremos estado S2, presenta el menor coste con un valor de $f = 600,2$, por lo que será el siguiente estado seleccionado para su expansión.

7.5.3. Estado S2

Este estado está compuesto por el movimiento del desbaste 250000010, ubicado en el tope de la pila inicial, a la pila «E06». Aunque S2 no es sucesor del estado expandido en el paso anterior, se encuentra el primero en la lista abierta, por lo que es seleccionado para su expansión.

La información de las funciones de coste para el estado actual se encuentran recogidas en la Tabla 7.25.

Tabla 7.25: Evaluación del estado S2 en el Ejemplo 7.5.

Componente	Valor
Coste acumulado (g)	463,28
Heurística (h)	136.97
Función de evaluación ($f = g + h$)	600,25

7.5.3.1. Configuración de la pila

La pila «E06» se encuentra vacía, por lo que la configuración corresponde únicamente al desbaste 250000010, por tanto no es necesario incluir una representación del estado. El resultado de evaluar los criterios de la propuesta previo al movimiento, obtiene los resultados recogidos en la Tabla 7.26.

Tabla 7.26: Evaluación del movimiento realizado para alcanzar S2 en el Ejemplo 7.5.

Regla	Puntuación
Pila vacía	412,66
Distancia	10,62
Tamaño de Ganchada	40
Programas	-
Cubrir Programa	-
Destino	-
Tipo	-
Familia	0
Anchos	0
Total	463,28

Donde se observa que la penalización se obtiene principalmente por el criterio de pila vacía, que evalúa tanto el ancho del desbaste como el número de pilas libres, para obtener su penalización.

Analizando el resto de pilas del parque, únicamente se observan pilas donde se produciría una mezcla de familias, lo que conlleva una penalización mayor.

7.5.3.2. Resultado de la expansión

Durante la expansión del estado S2 se generan 60 estados, agrupados en las tres ganchadas posibles para extraer el material, que se presentan en la Tabla 7.27.

Tabla 7.27: Expansión del estado S2 en el Ejemplo 7.5.

Ganchadas	Número de estados	Mejor puntuación (f)
250000699, 260000106, 250001920	12	1058,9
250000699, 260000106	11	1142,6
250000699	37	696,7

En este punto el mejor nodo existente no es un sucesor de S2, sino el movimiento del desbaste 250000699 a la pila «E06» sucesor de S1, que en adelante denominaremos estado S3, presenta el menor coste con un valor de $f = 622,9$, por lo que será el siguiente estado seleccionado para su expansión.

7.5.4. Estado S3

Este estado está compuesto por el movimiento del desbaste 250000699, ubicado en la altura tres de la pila inicial, a la pila «E06». Tras la expansión de los estados previamente seleccionados, S3 pasa a ser el estado con menor valor de f dentro de la lista abierta y, por tanto, es el siguiente en expandirse.

La Tabla 7.28 muestra el resultado de la evaluación de las funciones de coste.

Tabla 7.28: Evaluación del estado S3 en el Ejemplo 7.5.

Componente	Valor
Coste acumulado (g)	575,91
Heurística (h)	46,97
Función de evaluación ($f = g + h$)	622,88

7.5.4.1. Configuración de la pila

La pila «E06» se encuentra vacía, por lo que la configuración corresponde únicamente al desbaste 250000699, por tanto no es necesario incluir una representación del estado. El resultado de evaluar los criterios de la propuesta previo al movimiento, obtiene los resultados recogidos en la Tabla 7.29.

Tabla 7.29: Evaluación del movimiento realizado para alcanzar S3 en el Ejemplo 7.5.

Regla	Puntuación
Pila vacía	122,16
Distancia	10,62
Tamaño de Ganchada	40
Programas	0
Cubrir Programa	0
Destino	-
Tipo	-
Familia	-
Anchos	0
Total	172,78

Donde se observa que la penalización se obtiene principalmente por el criterio de pila vacía, que evalúa tanto el ancho del desbaste como el número de pilas libres, para obtener su penalización. Sin embargo, al tratarse de un desbaste de programa, observamos que la penalización de pila vacía se reduce y que entran en aplicación las reglas de programas y cubrir programa.

Explorando otras pilas del parque, únicamente se observan pilas donde la colocación de este desbaste, que tiene el grupo 33 dentro del programa, obstruiría la colocación de secuencias más cercanas.

7.5.4.2. Resultado de la expansión

Durante la expansión del estado S3 se generan 23 estados, agrupados en las dos ganchadas posibles para extraer el material, que se presentan en la Tabla 7.30.

Tabla 7.30: Expansión del estado S3 en el Ejemplo 7.5.

Ganchadas	Número de estados	Mejor puntuación (f)
260000106, 250001920	12	686,0
260000106	11	696,9

En este punto el mejor nodo existente no es un sucesor de S3, sino el movimiento del desbaste 250000699 a la pila «B10», que en adelante denominaremos estado S4, presenta el menor coste con un valor de $f = 678,4$, por lo que será el siguiente estado seleccionado para su expansión.

7.5.5. Estado S4

Este estado está compuesto por el movimiento del desbaste 250000699, ubicado en la altura tres de la pila inicial, a la pila «B10». Aunque S4 no es sucesor del estado expandido inmediatamente anterior, pasa a ocupar la primera posición de la lista abierta, debido a que su valor de f es inferior al del resto de alternativas disponibles.

La evaluación del estado se muestra en la Tabla 7.31.

Tabla 7.31: Evaluación del estado S4 en el Ejemplo 7.5.

Componente	Valor
Coste acumulado (g)	631,39
Heurística (h)	46,97
Función de evaluación ($f = g + h$)	678,36

7.5.5.1. Configuración de la pila

La Figura 7.14 muestra el estado de la pila «B10» tras aplicar el movimiento del estado S4. El material del movimiento se muestra seleccionado en azul.

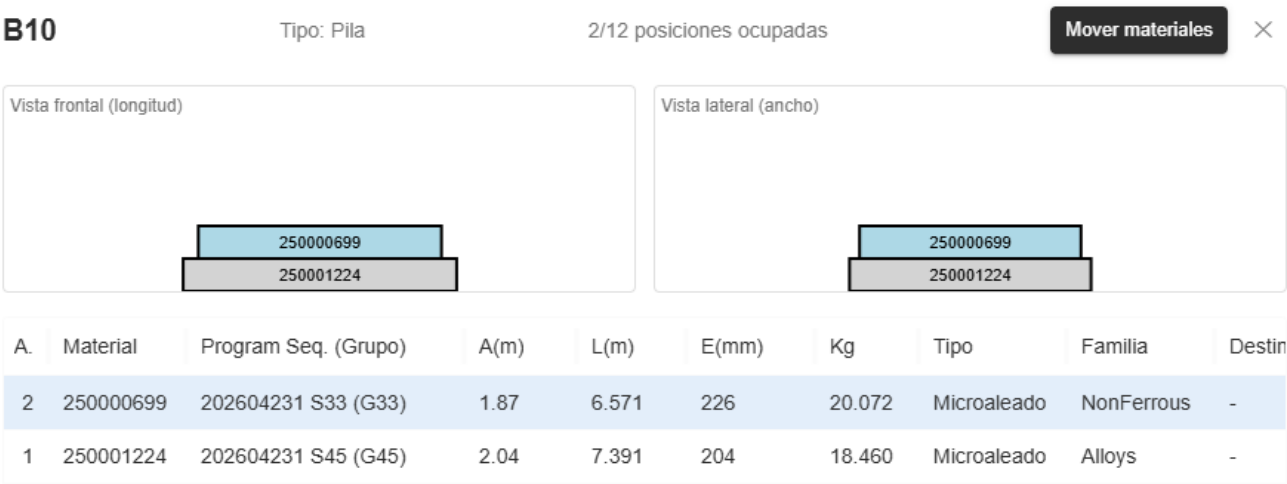


Figura 7.14: Configuración pila «B10» estado S4 en el Ejemplo 7.5.

El resultado de evaluar los criterios de la propuesta previo al movimiento, obtiene los resultados recogidos en la Tabla 7.32.

Tabla 7.32: Evaluación del movimiento realizado para alcanzar S4 en el Ejemplo 7.5.

Regla	Puntuación
Pila vacía	0
Distancia	22,01
Tamaño de Ganchada	40
Programas	60
Cubrir Programa	0
Destino	-
Tipo	-
Familia	-
Anchos	106,25
Total	228,26

Donde se observa que la ubicación es penalizada ligeramente tanto por la regla de programas como por la de ancho. En ambos casos, se debe a la diferencia de grupo con el material existente en la pila, lo que implica una diferencia de ancho. El único desbaste presente en la pila, comparte programa y tiene el grupo 45, lo que implica una diferencia de doce grupos, lo que puede ser mayor de 12 desbastes. Realizar esta ubicación obligaría a mezclar secuencias que irán potencialmente en diferentes cargas o ubicar las mismas en otras pilas, limitando el crecimiento de la pila «B10». En consecuencia, aunque la ubicación es físicamente válida, resulta menos adecuada, al dificultar futuras cargas de programa.

7.5.5.2. Resultado de la expansión

Durante la expansión del estado S4 se generan 23 estados, agrupados en las dos ganchadas posibles para extraer el material, que se presentan en la Tabla 7.33.

Tabla 7.33: Expansión del estado S4 en el Ejemplo 7.5.

Ganchadas	Número de estados	Mejor puntuación (f)
260000106, 250001920	12	741,4
260000106	11	752,4

En este punto el mejor nodo existente no es un sucesor de S4, sino el movimiento del desbaste 250000699 a la pila «E06» sucesor de S3, que en adelante denominaremos estado S5, presenta

el menor coste con un valor de $f = 686,0$, por lo que será el siguiente estado seleccionado para su expansión.

7.5.6. Estado S5

Este estado está compuesto por el movimiento de los desbastes 260000106, 250001920, ubicados en las alturas 2 y 1 respectivamente, de la pila inicial a la pila «F11». En este punto de la búsqueda, el estado S5 presenta el menor valor de la función de evaluación entre todos los estados de la lista abierta, por lo que es seleccionado para su expansión.

Tras aplicar esta secuencia se alcanza un estado objetivo, por lo que no se calcula la información de las funciones de coste para el estado actual.

7.5.6.0.1 Configuración de la pila

La Figura 7.15 muestra el estado de la pila «F11» tras aplicar el movimiento del estado S5. El material del movimiento se muestra seleccionado en azul.

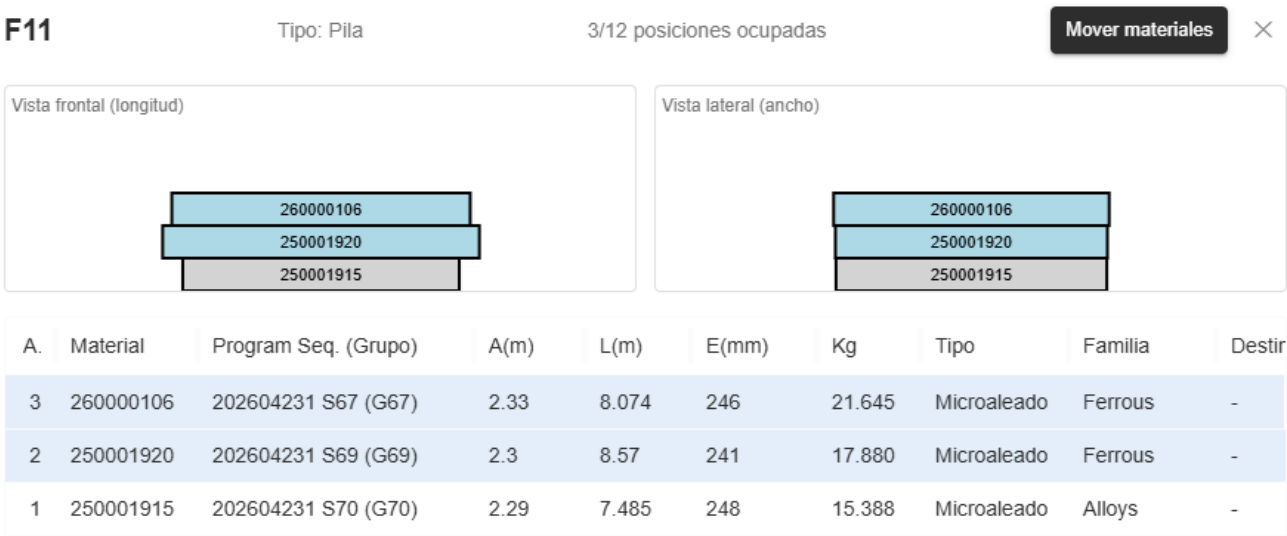


Figura 7.15: Configuración pila «F11» estado S5 en el Ejemplo 7.5.

El resultado de evaluar los criterios de la propuesta previo al movimiento, obtiene los resultados recogidos en la Tabla 7.34.

Tabla 7.34: Evaluación del movimiento realizado para alcanzar S5 en el Ejemplo 7.5.

Regla	Puntuación
Pila vacía	0
Distancia	22,2
Tamaño de Ganchada	30
Programas	7,8
Cubrir Programa	0
Destino	-
Tipo	-
Familia	-
Anchos	50
Total	110

Donde se observa que la ubicación es penalizada tanto por la regla de programas como por la de ancho y que, en ambos casos, la penalización es inferior a la que observamos en el estado S4. El único desbaste presente en la pila, comparte programa y tiene el grupo 70, lo que implica una diferencia de 1 y 3 grupos respectivamente. Lo que constituye una buena solución, dejando grupos muy próximos juntos.

7.5.6.1. Estado objetivo

Al tratarse de un estado objetivo, no se generan nuevos estados sucesores. La búsqueda finaliza y el algoritmo devuelve la secuencia de movimientos mostrada en la Tabla 7.17.

Este ejemplo permite validar el comportamiento de la propuesta de ubicación en un caso donde existen múltiples alternativas válidas para reubicar los desbastes del vagón. El análisis del árbol de búsqueda muestra que el algoritmo evalúa todas las configuraciones generadas, pero únicamente expande aquellos estados que presentan el menor valor de la función de evaluación en cada iteración. De este modo, la búsqueda se centra en las soluciones más prometedoras, sin necesidad de desarrollar completamente todas las ramas del árbol.

Las decisiones observadas durante la exploración son consistentes con los criterios de evaluación definidos en la propuesta. En cada estado, la ubicación seleccionada responde al equilibrio determinado por los criterios definidos. Como resultado, el algoritmo obtiene una secuencia de movimientos que resuelve la descarga del vagón, manteniendo la organización de los materiales en el almacén.

7.6. Ejemplo descarga de tren

Este ejemplo tiene como objetivo evaluar el algoritmo de descarga de trenes, descrito en la Sección 5.7. Con ese objetivo, se explora el caso de la descarga de un tren compuesto por tres vagones, con cuatro desbastes cada uno. A lo largo del ejemplo se evaluará la utilidad del algoritmo de división en subproblemas utilizado para realizar la descarga. Las propiedades de los desbastes elegidos pueden observarse en las Tablas 7.35, 7.36 y 7.37.

Tabla 7.35: Descripción desbastes vagón 1 en el Ejemplo 7.6.

Propiedad	Altura 4	Altura 3	Altura 2	Altura 1
Identificador	260000723	260000949	260000991	250001989
Programa - Secuencia - Grupo	No tiene	No tiene	No tiene	No tiene
Destino	No tiene	No tiene	No tiene	No tiene
Ancho	950mm	1050mm	1000mm	1000mm
Largo	6,671m	5,972m	8,75m	9,382m
Tipo	Cabecera	Cabecera	Arranque	Arranque
Familia	Alloys	NonFerrous	Alloys	Ferrous

Tabla 7.36: Descripción desbastes vagón 2 en el Ejemplo 7.6.

Propiedad	Altura 4	Altura 3	Altura 2	Altura 1
Identificador	260000545	250000725	260000915	260000575
Programa - Secuencia - Grupo	No tiene	No tiene	No tiene	No tiene
Destino	No tiene	No tiene	Almacén Ex- terno A	No tiene
Ancho	1800mm	1820mm	1930mm	2110mm
Largo	7,385m	8,63m	7,883m	10,09m
Tipo	Chapa	Chapa	DWI	Ancho
Familia	Ferrous	Ferrous	Alloys	Alloys

Tabla 7.37: Descripción desbastes vagón 3 en el Ejemplo 7.6.

Propiedad	Altura 4	Altura 3	Altura 2	Altura 1
Identificador	260000354	250000068	250001761	250000881
Programa - Secuencia - Grupo	No tiene	No tiene	No tiene	No tiene
Destino	No tiene	No tiene	No tiene	No tiene
Ancho	1920mm	1820mm	1920mm	2000mm
Largo	6,266m	7,77m	7,669m	7,414m
Tipo	Ultrabajo	Chapa	Ancho	Ancho
Familia	Alloys	NonFerrous	Alloys	Alloys

La Figura 7.16 muestra los movimientos propuestos para la descarga del tren y permite visualizar sus destinos. La representación permite identificar los movimientos cuyos materiales se trasladan a una misma pila de destino.

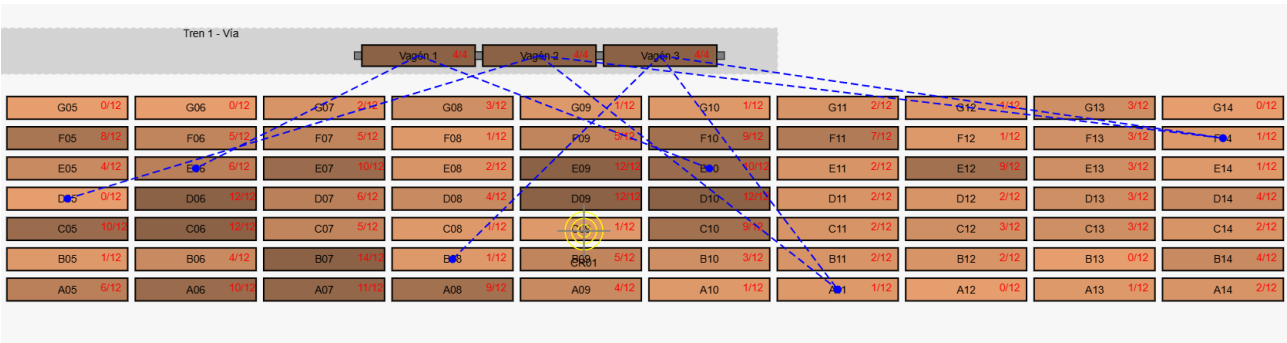


Figura 7.16: Movimientos en mapa del Ejemplo 7.6.

Para analizar cómo se realiza la descarga, el primer paso consiste en identificar los grupos que se emplearán para generar los subproblemas. En la Tabla 7.38, se muestra el detalle y el número de desbastes de cada uno de los grupos identificados en el transporte.

Tabla 7.38: Grupos de desbastes para generación de subproblemas del Ejemplo 7.6.

Grupo	Número de desbastes	Comentario
Tipo Cabecera	2	
Tipo Arranque	2	
Tipo Chapa	3	Distribuido entre los vagones 2 y 3.
Destino Almacén Externo A	1	Al ser el único material de destino nos referiremos a este únicamente como «Destino».
Tipo Ancho	3	Distribuido entre los vagones 2 y 3.
Tipo Ultrabajo	1	

En la Figura 7.17 se representan los grupos presentes en el transporte en el estado inicial. Se ha asignado un color único para cada grupo.



Figura 7.17: Representación de criterios en estado inicial del Ejemplo 7.6.

A partir de la información de diagnóstico disponible para la propuesta, descrita en la Sección 5.5, se observa que la descarga se ha dividido en seis subproblemas, uno por cada grupo identificado. La información básica de diagnóstico, en la que se asigna un identificador a cada estado según el orden en el que se realiza la descarga, se muestra en la Tabla 7.39. Es importante destacar que, debido al funcionamiento de la generación de movimientos, cada una de estas propuestas parte de un estado del parque que asume la ejecución de los movimientos calculados por las propuestas anteriores.

Tabla 7.39: Datos de diagnóstico propuesta ubicación del Ejemplo 7.6.

Estado	Grupo	Nº Ma- teriales	Tiempo ejecu- ción	Vecinos genera- dos	Nodos des- carta- dos	Ejec. Heurísti- co	Nodos expan- didos
1	Tipo Ultrabajo	1	14ms	68	44	25	1
2	Tipo Chapa	3	177ms	408	240	169	3
3	Destino	1	7ms	73	52	22	1
4	Tipo Ancho	3	12ms	680	511	170	6
5	Tipo Cabecera	2	9ms	136	29	108	1
6	Tipo Arranque	2	9ms	136	60	77	1

A continuación, se irá mostrando la evolución de la descarga a través de los estados correspondientes a cada subproblema.

7.6.1. Estado 1

El proceso de descarga comienza retirando el desbaste de tipo ultrabajo, ubicado en la altura 4 del vagón 3, ya que de esta forma todos los materiales de tipo chapa quedan accesibles para su descarga. El movimiento calculado por la primera propuesta de ubicación se muestra en la Tabla 7.40.

Tabla 7.40: Movimientos calculados para el estado 1 del Ejemplo 7.6.

Orden: 1		
De	A	ID del Material
Vagón 3 (A4)	B08 (A2)	260000354

7.6.2. Estado 2

Una vez ejecutado el primer movimiento, el estado de la descarga queda como se muestra en la Figura 7.18.



Figura 7.18: Representación de criterios en estado 2 del Ejemplo 7.6.

En este caso, el algoritmo de descarga decide retirar los desbastes de tipo chapa, ubicados en las alturas 3 y 4 del vagón 2 y en la altura 3 del vagón 3. Los movimientos calculados se muestran en la Tabla 7.41, donde se observa que los tres desbastes se llevan a la misma pila de destino.

Tabla 7.41: Movimientos calculados para el estado 2 del Ejemplo 7.6.

Orden: 2		
De	A	ID del Material
Vagón 3 (A3)	F14 (A2)	250000068
Orden: 3		
De	A	ID del Material
Vagón 2 (A4)	F14 (A4)	260000545
Vagón 2 (A3)	F14 (A3)	250000725

7.6.3. Estado 3

En el siguiente subproblema, el estado de la descarga es el que se representa en la Figura 7.19.

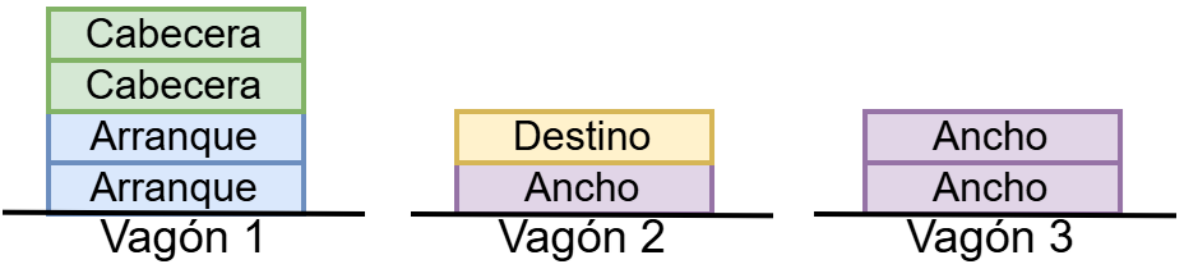


Figura 7.19: Representación de criterios en estado 3 del Ejemplo 7.6.

En este estado, se descarga el desbaste de criterio destino, para permitir la propuesta conjunta de los tres desbastes de tipo ancho. El movimiento calculado, que realiza una apertura de pila para este material, puede observarse en la Tabla 7.42.

Tabla 7.42: Movimientos calculados para el estado 3 del Ejemplo 7.6.

Orden: 4		
De	A	ID del Material
Vagón 2 (A2)	D05 (A1)	260000915

7.6.4. Estado 4

Tras la descarga del material de destino, en la Figura 7.20 se muestra el estado de los vagones.

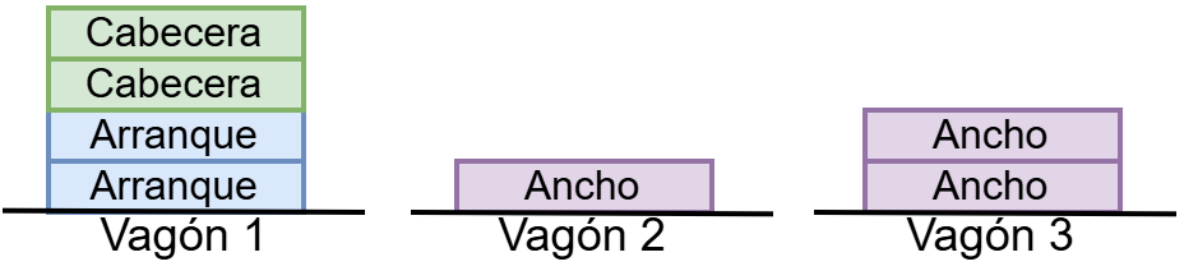


Figura 7.20: Representación de criterios en estado 4 del Ejemplo 7.6.

En este estado el algoritmo de descarga decide descargar los tres desbastes de tipo ancho que han quedado accesibles. Los desbastes están ubicados en la altura 1 del vagón 2 y en las alturas 1 y 2 del vagón 3. Los movimientos que se han calculado para este caso se muestran en la Tabla 7.43, donde se observa que los tres desbastes se llevan a la misma pila de destino.

Tabla 7.43: Movimientos calculados para el estado 4 del Ejemplo 7.6.

Orden: 5		
De	A	ID del Material
Vagón 2 (A1)	A11 (A2)	260000575
Orden: 6		
De	A	ID del Material
Vagón 3 (A2)	A11 (A4)	250001761
Vagón 3 (A1)	A11 (A3)	250000881

7.6.5. Estado 5

En esta etapa, habiendo terminado la descarga de los vagones 2 y 3, el estado es el que se ilustra en la Figura 7.21.

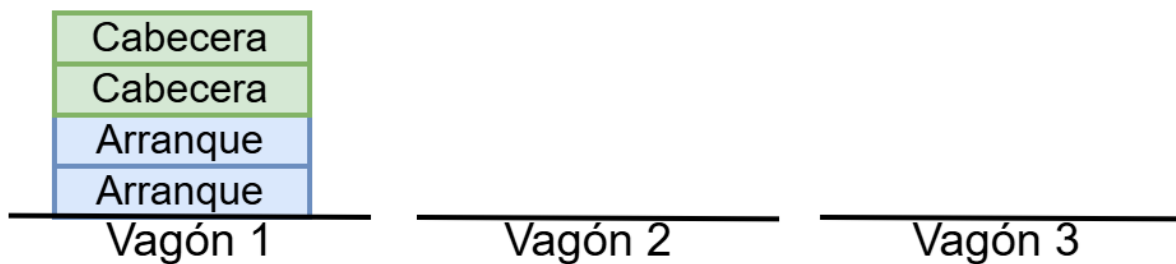


Figura 7.21: Representación de criterios en estado 5 del Ejemplo 7.6.

En esta ocasión se descargan los desbastes de tipo cabecera, ubicados en las alturas 3 y 4. Los movimientos calculados se pueden observar en la Tabla 7.44. En este caso, ambos desbastes se descargan como parte de una misma ganchada.

Tabla 7.44: Movimientos calculados para el estado 5 del Ejemplo 7.6.

Orden: 7		
De	A	ID del Material
Vagón 1 (A4)	E10 (A12)	260000723
Vagón 1 (A3)	E10 (A11)	260000949

7.6.6. Estado 6

Finalmente, para los últimos materiales que quedan por descargar el estado de la descarga es el representado en la Figura 7.22.

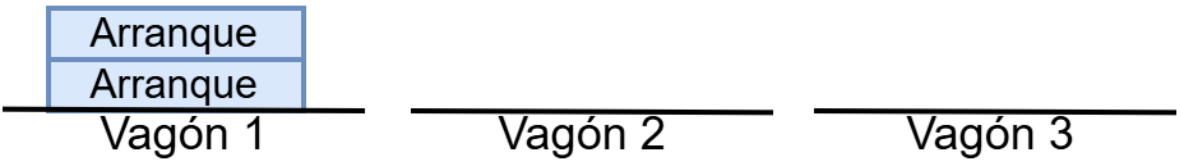


Figura 7.22: Representación de criterios en estado 6 del Ejemplo 7.6.

En este caso se descargan los desbastes de tipo arranque, ubicados en las alturas 1 y 2. El movimiento generado puede observarse en la Tabla 7.45.

Tabla 7.45: Movimientos calculados para el estado 6 del Ejemplo 7.6.

Orden: 8		
De	A	ID del Material
Vagón 1 (A2)	E06 (A8)	260000991
Vagón 1 (A1)	E06 (A7)	250001989

Este ejemplo permite observar cómo el algoritmo de descarga de trenes descompone un problema en una secuencia de subproblemas. En este caso, los doce desbastes del tren se agrupan en seis subproblemas que se resuelven de forma secuencial, utilizando como punto de partida de cada propuesta el estado resultante de la anterior. El orden obtenido, permite liberar progresivamente los materiales necesarios para abordar los siguientes grupos, mientras que la propuesta de ubicación determina los movimientos concretos para cada subproblema. De este modo, el ejemplo muestra cómo ambas soluciones trabajan conjuntamente: el algoritmo de descarga determina qué grupo de materiales debe abordarse en cada etapa y la propuesta de ubicación calcula cómo realizar los movimientos necesarios para llevarlo a cabo.

Además, los datos de diagnóstico muestran que la búsqueda se realiza de forma individual para cada subproblema sobre el estado correspondiente, limitando la exploración a los materiales relevantes en cada etapa y evitando abordar en una única búsqueda la totalidad de las combinaciones posibles de la descarga.

7.7. Ejemplo carga de programa

Este último ejemplo demuestra el funcionamiento del sistema de soporte a la decisión, desarrollado para asistir a los operarios en la carga de programas. Esta funcionalidad se ha descrito en la Sección 5.8 y el objetivo de este ejemplo es mostrar su comportamiento en una situación real. La funcionalidad permite obtener automáticamente una secuencia de movimientos para realizar la carga de un programa en un transporte, respetando la ordenación descendente de grupos y resolviendo las situaciones en las que los materiales necesarios no se encuentren directamente accesibles.

El ejemplo se ejecuta bajo las siguientes condiciones:

- Se realizará la carga del único programa presente en el parque, el programa 202604231.
- El último grupo expedido es el 12, por lo que únicamente se expedirán grupos posteriores a este.
- Los desbastes necesarios para la carga se encuentran disponibles en el parque.
- El transporte empleado para la carga es un tren compuesto por tres vagones.
- Las pilas auxiliares coinciden con las excluidas de la propuesta, que se han definido en el Subapartado 7.2.

A partir de estas condiciones, se genera el plan de movimientos correspondiente. El algoritmo primero determina a partir de qué secuencia cargar, planifica la composición objetivo de cada vagón y finalmente calcula la secuencia de movimientos necesaria para realizar la carga del programa. La Figura 7.23 muestra la secuencia de movimientos calculados sobre el mapa del parque.

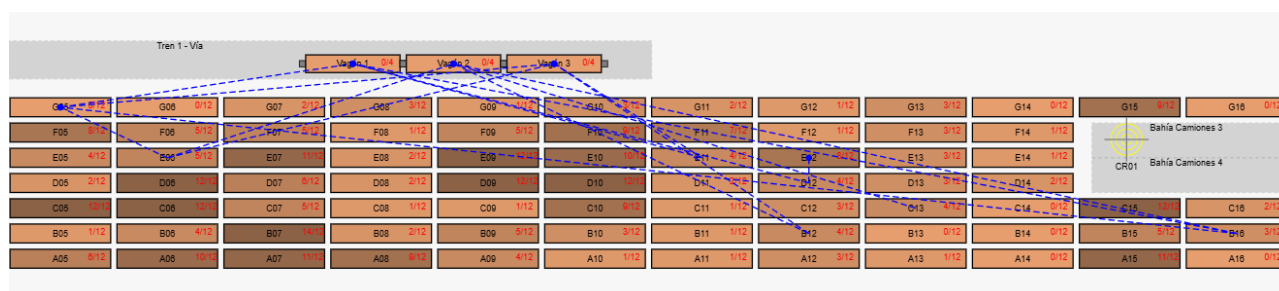


Figura 7.23: Secuencia de movimientos propuesta en el Ejemplo 7.7.

El detalle de los movimientos propuestos se muestra en la Tabla 7.46. La secuencia combina la colocación de los desbastes en los vagones con los movimientos auxiliares que resultan necesarios en cada momento. Los movimientos auxiliares se derivan de las siguientes circunstancias:

- Cuando el desbaste que debe cargarse está tapado por materiales que no han de cargarse,

el algoritmo genera los movimientos necesarios para liberar su acceso y continúa con la carga, según el orden establecido.

- Cuando un material necesario bloquea el acceso a otro que debe cargarse antes, el algoritmo puede utilizar temporalmente una pila auxiliar para modificar el orden de extracción.

Tabla 7.46: Tabla de movimientos propuestos del Ejemplo 7.7.

Orden	De	A	ID del Material	Grupo
1	D12 (A4)	E12 (A9)	260000537	-
2	D12 (A3)	Vagón 1 (A1)	250001902	16
3	B16 (A3)	G05 (A1)	260000123	13
4	B16 (A2)	Vagón 1 (A2)	250001343	15
5	E11 (A4)	Vagón 1 (A3)	250000835	14
6	G05 (A1)	Vagón 1 (A4)	260000123	13
7	E06 (A5)	Vagón 2 (A1)	260000756	20
8	B12 (A4)	Vagón 2 (A2)	260000580	19
9	C13 (A4)	Vagón 2 (A3)	260000004	18
10	B16 (A1)	Vagón 2 (A4)	250001267	17
11	E06 (A4)	G05 (A1)	250001399	21
12	E06 (A3)	Vagón 3 (A1)	250001365	24
13	E11 (A3)	Vagón 3 (A2)	250000294	23
14	B12 (A3)	Vagón 3 (A3)	260000357	22
15	G05 (A1)	Vagón 3 (A4)	250001399	21

En particular, se observan los tres tipos de movimientos generados por el algoritmo de carga:

- **Al almacén:** Este tipo de movimiento permite apartar materiales que impiden acceder a los desbastes que deben cargarse. En el movimiento de orden 1, se desplaza el material situado en la cabecera de la pila «D12» hasta «E12», liberando el acceso al material situado inmediatamente debajo. Como consecuencia, en el movimiento de orden 2 es posible extraer el desbaste 250001902, correspondiente al grupo 16, y colocarlo en la base del vagón 1.
- **A pila auxiliar:** Estos movimientos permiten apartar temporalmente un material cuando su extracción es necesaria para acceder a otro desbaste que debe cargarse antes. En el movimiento de orden 3, se desplaza el material 260000123 desde «B16» hasta la pila auxiliar «G05», permitiendo extraer el material que debe ocupar la segunda posición del

vagón 1. Una vez colocados los materiales que deben situarse por debajo, el movimiento de orden 6 recupera el material 260000123 desde «G05» y lo coloca en su posición definitiva, correspondiente a la cuarta altura del vagón.

- **A pila objetivo:** Son los movimientos mediante los cuales un desbaste se desplaza hasta su posición definitiva en el vagón. Este es el tipo de movimiento más frecuente en la secuencia propuesta. Los movimientos 4, 5 y 6 son ejemplos de este tipo, ya que colocan los materiales en las posiciones segunda, tercera y cuarta del primer vagón, respectivamente. En particular, el movimiento 6 corresponde a la colocación definitiva del material 260000123, que previamente había sido desplazado a una pila auxiliar mediante el movimiento 3.

La Figura 7.24 muestra, a través de la pantalla de programas, el resultado de la carga. En ella puede observarse cómo se ha trasladado el orden de elementos del programa a la composición del transporte, posicionando los desbastes en los tres vagones según el orden establecido por el programa.

Elementos - 202604231

72 elementos

Secuencia	Grupo	Material	Ubicación	Altura
12	12	250000489	-	
13	13	260000123	Vagón 1	4
14	14	250000835	Vagón 1	3
15	15	250001343	Vagón 1	2
16	16	250001902	Vagón 1	1
17	17	250001267	Vagón 2	4
18	18	260000004	Vagón 2	3
19	19	260000580	Vagón 2	2
20	20	260000756	Vagón 2	1
21	21	250001399	Vagón 3	4
22	22	260000357	Vagón 3	3
23	23	250000294	Vagón 3	2
24	24	250001365	Vagón 3	1
25	25	250002033	C13	3
26	26	250001064	E11	2

Figura 7.24: Resultado de la carga del programa en el Ejemplo 7.7.

Las Figuras 7.25, 7.26 y 7.27 muestran el estado final de cada uno de los vagones. En los tres casos se observa que los desbastes han quedado dispuestos conforme al orden establecido por el programa. Además, se observa una progresión creciente en el ancho de los desbastes, característica propia de la fase de crecimiento del programa.

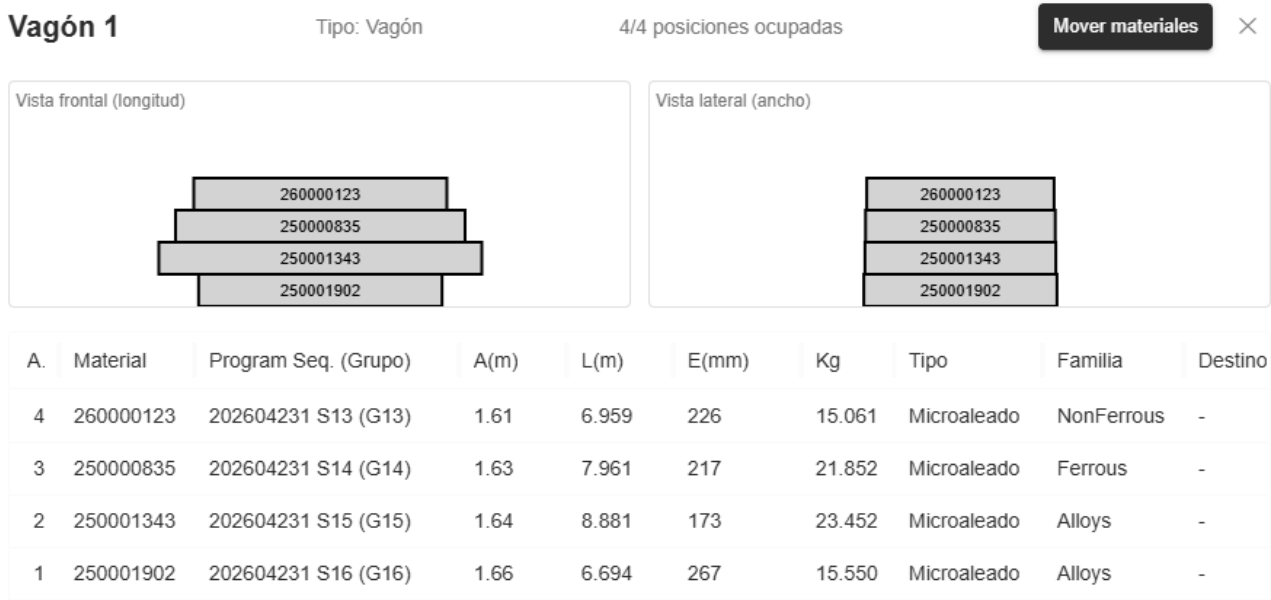


Figura 7.25: Resultado de la carga del vagón 1 en el Ejemplo 7.7.

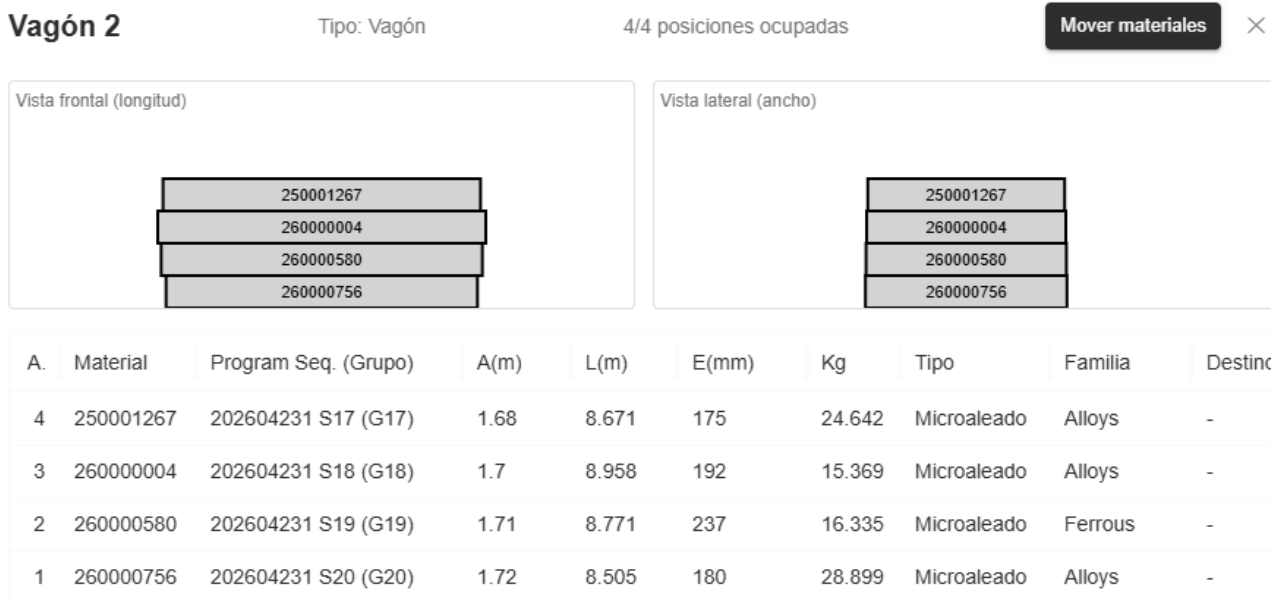


Figura 7.26: Resultado de la carga del vagón 2 en el Ejemplo 7.7.

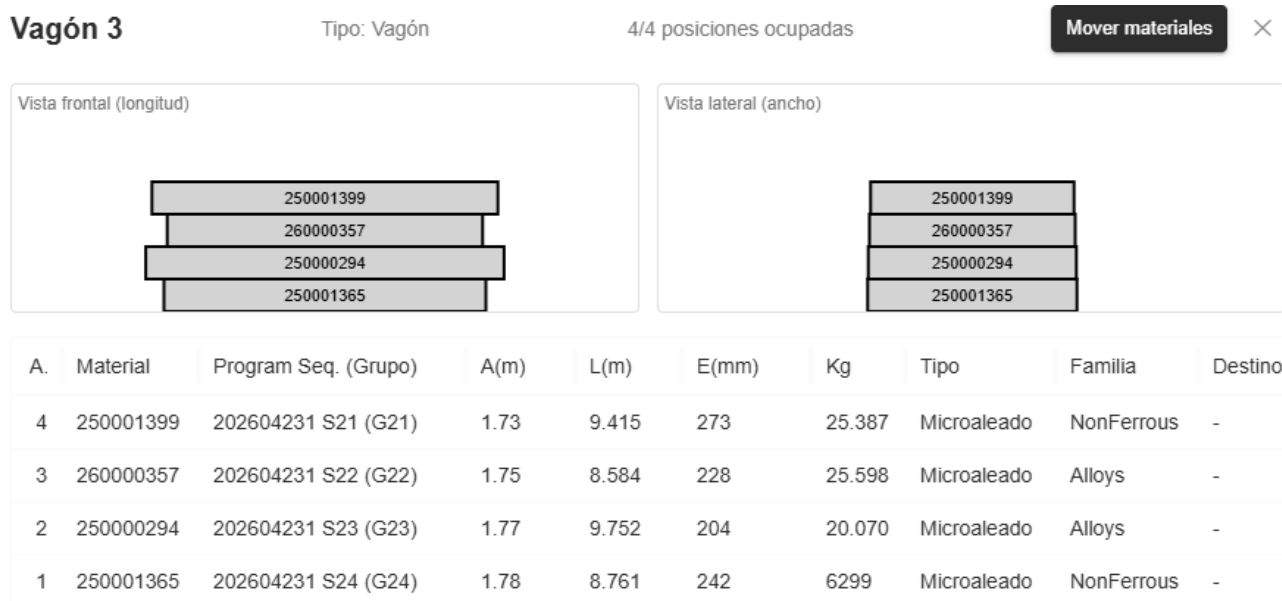


Figura 7.27: Resultado de la carga del vagón 3 en el Ejemplo 7.7.

Como puede observarse, la secuencia generada requiere un número de movimientos superior al número de desbastes cargados. Esto se debe a que algunos materiales no se encontraban inicialmente en una posición desde la que pudieran ser cargados directamente, siendo necesaria la realización de movimientos auxiliares para liberar su acceso o modificar temporalmente su orden de extracción.

Esta situación está relacionada con el momento en el que se generó el programa. En ese momento, únicamente se encontraba disponible en el parque aproximadamente el 15 % de las secuencias necesarias para su carga. Por tanto, la ordenación por el criterio de tipo no pudo influir, proporcionando una distribución adecuada del conjunto de materiales del programa. El resto del material fue recibido posteriormente mediante diferentes transportes, por lo que, los desbastes necesarios para un mismo programa, pueden quedar desordenados en el parque, ya que su ubicación es una decisión local realizada según su orden de recepción.

Por este motivo, es habitual realizar trabajos de ordenación de los programas antes de su carga, con el objetivo de ubicar los materiales en posiciones que faciliten su expedición. Estos trabajos pueden requerir movimientos similares a los mostrados en este ejemplo, ya que deben resolver las mismas restricciones de acceso y ordenación que aparecen durante la carga. La ventaja de realizarlos antes es que permite una mayor velocidad de carga del transporte.

Este ejemplo permite comprobar la utilidad de la funcionalidad de carga de programas. El sistema de soporte a la decisión, genera automáticamente una secuencia de movimientos que respeta el orden de carga requerido y resuelve las situaciones en las que los materiales necesarios no se encuentran directamente accesibles. De este modo, los operadores pueden disponer de una secuencia de carga calculada por el sistema, sin tener que realizar manualmente este proceso.

7.8. Conclusiones

En este capítulo se ha presentado la verificación de los sistemas de soporte a la decisión, mediante diferentes escenarios representativos de las situaciones que puede afrontar GenStock. Los casos planteados permiten analizar progresivamente el comportamiento de las propuestas de ubicación y de los planes de movimientos, a medida que aumenta la cantidad de materiales, las alternativas disponibles y las operaciones necesarias.

Los resultados obtenidos permiten comprobar que las decisiones generadas se ajustan a las reglas y criterios definidos, incluso en situaciones en las que es necesario considerar diferentes alternativas o realizar movimientos auxiliares para alcanzar la solución. Los escenarios planteados no permiten cubrir de forma exhaustiva todas las configuraciones posibles del almacén, pero proporcionan una verificación dirigida de los principales mecanismos que intervienen en la toma de decisiones.

Finalmente, se destaca la importancia de los sistemas de diagnóstico definidos en la Sección 5.5, ya que permiten justificar y argumentar las decisiones tomadas por los sistemas de soporte. Además, proporcionan una base para la identificación de situaciones particulares y para el ajuste y refinamiento de las reglas empleadas en la generación de propuestas. De esta forma, la verificación no se limita a comprobar el resultado obtenido, sino que permite analizar el comportamiento del sistema y detectar posibles aspectos susceptibles de mejora.

Capítulo 8

Conclusiones y trabajos futuros

8.1. Introducción

El objetivo de este trabajo ha sido desarrollar un sistema de gestión de almacenes (SGA), aplicado al caso de los almacenes de desbastes y enriquecido con sistemas de apoyo a la toma de decisiones. Estos sistemas debían ser capaces de generar propuestas de ubicación y movimientos, teniendo en cuenta las restricciones del almacén y los criterios operativos empleados en su gestión. De esta forma, se busca garantizar los objetivos de almacenamiento temporal del parque, manteniendo la trazabilidad del material y cumpliendo los requisitos de tiempo de servicio a los que se encuentra sometido.

Durante el desarrollo, se ha comprobado que la propuesta de ubicación no puede plantearse como una simple búsqueda de ubicaciones válidas para los materiales. La gran cantidad de combinaciones posibles derivadas de las características del material almacenado, así como la convivencia de distintos tipos de operaciones, con requisitos relacionados entre sí, hacen necesario emplear estrategias específicas para cada situación. En esta línea, el principal resultado del trabajo ha sido la construcción de una solución de propuesta de ubicación, que combina estrategias de descomposición, algoritmos de búsqueda y reglas de evaluación, para alcanzar la capacidad de resolución necesaria para abordar los problemas del almacén.

En este capítulo se presentan las principales conclusiones y aprendizajes obtenidos durante el diseño e implementación de GenStock. El análisis se centra especialmente en aquellos aprendizajes relacionados con la creación de sistemas de soporte a la decisión, contextualizándolos en las necesidades propias del SGA.

8.2. Conclusiones

Como resultado del trabajo se ha desarrollado GenStock, una aplicación funcional para la gestión de almacenes de desbastes que integra, tanto las funcionalidades necesarias para la gestión del inventario y de las operaciones del almacén, como diferentes sistemas de soporte a la decisión. Estos sistemas, permiten asistir al operador en decisiones que presentan una elevada complejidad, especialmente en la elección de ubicaciones y en la planificación de movimientos

de las grúas. La solución desarrollada permite gestionar el estado del inventario, representar las operaciones del almacén en tiempo real, mantener la trazabilidad de los materiales y planificar los principales tipos de movimientos que pueden realizarse sobre el inventario.

El grado de cumplimiento de estos objetivos se ha comprobado mediante dos procesos de verificación complementarios. Por una parte, la funcionalidad general de la aplicación se ha verificado mediante pruebas automatizadas y funcionales, que comprueban el comportamiento de sus diferentes componentes y operativas. Por otra, los sistemas de soporte a la decisión se han verificado sobre diferentes escenarios, comprobando su comportamiento ante distintos estados del almacén. Estas últimas pruebas, se han complementado con herramientas de diagnóstico que permiten analizar las decisiones generadas y evaluar los factores que intervienen en los resultados obtenidos.

Con respecto a los objetivos planteados al inicio del trabajo, se ha conseguido lo siguiente:

- **Asistencia a la ubicación de desbastes:** La propuesta de ubicación desarrollada permite asistir al operador en la selección de las posiciones donde almacenar los desbastes. La propuesta combina diferentes criterios de evaluación y mecanismos de búsqueda, para seleccionar las ubicaciones más adecuadas, de acuerdo con el estado del almacén y los criterios operativos definidos. Este sistema se integra en los diferentes procesos que requieren tomar decisiones sobre la ubicación del material, proporcionando asistencia durante la planificación de los movimientos.
- **Consultar el estado del inventario en tiempo real:** Se han implementado mecanismos de actualización en tiempo real que permiten mantener actualizada la información mostrada a los operadores. El mapa de la aplicación representa el estado actual del inventario, junto con la posición de las grúas, mientras que la pantalla de planes de movimiento refleja la evolución de las operaciones, a medida que se ejecutan los movimientos. Adicionalmente, la capa de movimientos pendientes permite visualizar, sobre el plano del almacén, los movimientos planificados y las posiciones involucradas en ellos.
- **Garantizar la trazabilidad del material:** Se ha implementado un sistema de persistencia que mantiene, tanto el estado actual del inventario, como la información histórica asociada a los movimientos realizados. Los movimientos se almacenan de forma independiente de los planes y de las ubicaciones actuales, permitiendo consultar el histórico de movimientos de cada material. De esta forma, el sistema conserva la información necesaria para reconstruir la secuencia de manipulaciones realizadas sobre los desbastes y facilita la investigación de las operaciones realizadas sobre el inventario.
- **Gestionar los movimientos de las grúas:** Se han desarrollado cuatro tipos de planes de movimiento, que cubren las principales operaciones consideradas en el sistema:
 - **Descarga de transportes:** Se ha desarrollado un mecanismo de planificación que

descompone la operación en diferentes subproblemas y combina la propuesta de ubicación con la generación de la secuencia de movimientos. La solución permite obtener, automáticamente, una secuencia de descarga a las ubicaciones óptimas que, además, agrupa movimientos para reducir la distancia recorrida por la grúa.

- **Liberación de pilas:** Este tipo de plan permite generar una secuencia de movimientos para vaciar una pila y facilitar las tareas de reorganización y compactación del material. La propuesta de ubicación se utiliza para determinar las posiciones adecuadas para los desbastes que deben ser trasladados.
- **Carga de programas:** Se ha desarrollado una solución específica para planificar la carga de los materiales asociados a un programa, incluyendo la reordenación de las secuencia,s que ha de realizarse antes de la carga, y el traslado del material hasta el transporte de destino. Cuando es necesario apartar material para realizar la carga, esta solución utiliza la propuesta de ubicación, para determinar los movimientos necesarios.
- **Reubicación:** Se ha incorporado un tipo de plan destinado a proporcionar una planificación bajo demanda para movimientos que no quedan cubiertos por las operaciones anteriores. A diferencia del resto de planes, este flujo no incorpora un sistema de soporte a la decisión, ya que su finalidad es proporcionar al operador una herramienta para gestionar necesidades adicionales de movimiento.

En conjunto, los resultados obtenidos muestran que se han alcanzado los objetivos planteados al inicio del trabajo, tanto en lo relativo a la funcionalidad de la aplicación como a los sistemas de soporte a la decisión. La solución desarrollada, integra los sistemas de soporte a la decisión en las principales operaciones que requieren tomar decisiones sobre el inventario y los movimientos de las grúas. Además, las abstracciones y configuraciones utilizadas en su desarrollo, permiten ampliar el comportamiento de estos sistemas e incorporar nuevos criterios o funcionalidades sin necesidad de modificar su estructura general.

El desarrollo de la solución también ha permitido obtener varios aprendizajes, relacionados con las decisiones de diseño adoptadas y con los problemas encontrados durante la implementación de los sistemas de soporte a la decisión.

La decisión de diseño más importante en la construcción de GenStock ha sido su sistema de almacenamiento, que permite configurar el diseño del almacén mediante un fichero en formato JSON. De esta forma, la ubicación de los elementos de almacenamiento en el parque puede modificarse fácilmente, proporcionando una forma sencilla de adaptar el sistema a diferentes configuraciones del almacén, sin necesidad de cambiar la lógica de la aplicación. En cuanto a la persistencia del inventario, ha resultado particularmente efectiva la solución adoptada, que requiere mantener únicamente las filas correspondientes a los materiales que se encuentran actualmente ubicados en el parque.

Pasando al área de los aprendizajes relacionados con los sistemas de apoyo a la decisión, uno de los principales resultados obtenidos tiene que ver con la construcción del heurístico que orienta la búsqueda del algoritmo A*. Durante el desarrollo, se comprobó la dificultad de establecer un heurístico que evaluase correctamente la distancia a la meta, cuando no se conoce exactamente la secuencia de movimientos que permitirá alcanzarla, ni qué características tendrá el estado final. La heterogeneidad de las propiedades de los materiales y el orden en el que pueden ser extraídos para formar movimientos, condicionan los posibles resultados. Para abordar esta dificultad, se introduce una relajación del problema basada en la construcción de una secuencia de extracción teórica de materiales y su asignación voraz sobre las pilas del parque. A partir de la misma, se obtiene un coste heurístico mediante la evaluación simplificada de la compatibilidad entre el material y el tope de la pila, a la que se incorpora la penalización derivada de las diferencias de ancho entre ambos. Esta aproximación, permite estimar el coste pendiente sin necesidad de resolver el problema completo, manteniendo una elevada capacidad para discriminar estados. El principal aprendizaje de este desarrollo ha sido comprobar que el ancho constituye una excelente guía heurística. Su presencia como restricción común en las clasificaciones del almacén, hace que se convierta en un indicador especialmente fiable para estimar la calidad de un estado y orientar la búsqueda hacia soluciones prometedoras.

Otro aspecto que ha condicionado profundamente el diseño de la solución es la complejidad combinatoria presente en el algoritmo de búsqueda. El número de estados posibles aumenta rápidamente, debido principalmente a las diferentes formas de extracción de los materiales. Sin embargo, el análisis del problema permite identificar grupos de materiales que, por sus características y criterios de agrupación, resulta conveniente descargar conjuntamente. Empleando este conocimiento, se introduce una estrategia de descomposición en subproblemas, que permite dividir la descarga del material y reducir la combinatoria asociada a la extracción, evitando que la búsqueda tenga que explorar de forma conjunta alternativas que pueden evaluarse por separado. De esta forma, se consigue reducir el número de cálculos necesarios, sin sacrificar la calidad de las soluciones obtenidas. En esta estrategia, cada subproblema determina el conjunto de materiales que debe abordarse en una determinada etapa de la descarga, mientras que la propuesta de ubicación se encarga de determinar los movimientos necesarios para realizarla. Esta propuesta, se calcula sobre el estado resultante de la etapa anterior, de forma que los movimientos generados en cada subproblema modifican el estado sobre el que se resolverá el siguiente. Las pruebas realizadas sobre la descarga de transportes, han permitido comprobar la utilidad de esta estrategia para limitar la exploración y reducir los cálculos necesarios, sin renunciar a la calidad de las soluciones obtenidas.

Durante el desarrollo también se ha identificado la necesidad de optimizar la distancia recorrida en los movimientos realizados. La problemática se origina en que la propuesta de ubicación genera una secuencia de movimientos, basada en la evolución de los estados, para ubicar los desbastes objetivo en las posiciones elegidas y, una vez determinada esta secuencia, no reali-

za consideraciones adicionales sobre su ejecución. Aunque la distancia es uno de los criterios empleados para determinar estas posiciones, sigue siendo posible optimizar, tanto el tiempo de movimiento, como la distancia recorrida, mediante la ampliación de las ganchadas. Esto permite transportar varios desbastes en una misma secuencia de movimiento, realizando las descargas y nuevas recogidas necesarias durante el recorrido, para evitar desplazamientos adicionales. Por este motivo, se introduce un cálculo diferenciado que optimiza la secuencia de movimientos, separando la decisión de qué ubicación resulta más adecuada, de la decisión de cómo alcanzar dicha ubicación de forma eficiente. Esta separación, junto con la división en subproblemas introducida anteriormente, se unen para formar una estructura de generación de soluciones con tres capas independientes, que trabajan de forma conjunta. Primero, se determinan los grupos que deben abordarse, luego se seleccionan las ubicaciones más adecuadas, mediante la aplicación de los criterios de la propuesta y, finalmente, se optimiza el coste de la secuencia de movimientos necesaria para ejecutar la solución. De esta forma, la misma solución es capaz de integrar decisiones con objetivos diferentes en un único resultado.

Otro aprendizaje se deriva del hecho de que no todos los planes de movimiento pueden gestionarse mediante una misma solución. La carga de programas constituye un ejemplo de esta situación, en el que se opta por una solución específica, en lugar de tratar de forzar su comportamiento como parte de la propuesta de ubicación. Este algoritmo se encarga de generar el estado final de la carga y planificar los movimientos necesarios para alcanzarla, siendo capaz, a diferencia de la propuesta de ubicación, de realizar reordenaciones y apartar material, cuando resulta necesario. Esta solución puede apoyarse, a su vez, en la propuesta de ubicación, utilizando sus capacidades para determinar los movimientos necesarios para apartar material. De esta forma, se consigue resolver el problema mediante la utilización de herramientas adecuadas a sus características, permitiendo además la colaboración entre las diferentes soluciones, cuando resulta necesario, en lugar de tratar de concentrar todos los comportamientos en un único algoritmo.

Finalmente, el desarrollo de diferentes reglas de evaluación permite representar los múltiples criterios operativos que se combinan para construir una solución. El uso de un sistema extensible de criterios, permite comparar diferentes alternativas y determinar cuáles resultan más favorables para la ubicación. De esta forma, la propuesta no queda limitada a un conjunto fijo de criterios, sino que se convierte en un sistema flexible, que admite la introducción y configuración de nuevos criterios de evaluación.

En resumen, el principal aprendizaje que se deriva del trabajo no es la implementación de la propuesta de ubicación en sí, sino la identificación y el tratamiento de las diferentes circunstancias que intervienen en la gestión del inventario y de los movimientos. La solución desarrollada permite adaptar el tratamiento de cada situación, mediante la combinación de diferentes estrategias de evaluación y búsqueda, utilizando las soluciones implementadas. Las

pruebas realizadas permiten verificar el funcionamiento de estas estrategias, en diferentes escenarios, y muestran la utilidad de la solución desarrollada como herramienta de apoyo para la gestión de las operaciones del almacén

8.3. Trabajos futuros

El desarrollo realizado deja abiertas diferentes líneas de trabajo, orientadas a ampliar las operaciones soportadas por GenStock, mejorando la gestión del almacén. Entre las principales líneas de trabajo futuro se encuentran:

- **Implementación de la carga por destino:** Actualmente la generación de planes de movimiento solo soporta la carga de programas. Se podrían ampliar las operaciones de carga para incluir la carga por destino, que constituye el segundo flujo más importante de salida de material. Esta operación presenta características diferentes a la carga de programas, pero resulta más sencilla al no requerir un orden estricto ni una secuencia de carga prefijada.
- **Implementación de zonas dinámicas:** Permitir la definición de zonas lógicas de tamaño configurable, asociadas a diferentes criterios y restricciones de ubicación. Esto permitiría a la propuesta aplicar diferentes penalizaciones, implementando las restricciones relativas a las zonas en los criterios de evaluación. Por ejemplo, se podría diseñar una zona dedicada al material con programa, en la que la ubicación de otros tipos de materiales derivase en mayores penalizaciones.
- **Ordenación del material dentro del parque:** Esta es una tarea importante ante cambios de planificación o de criterios en los materiales almacenados. El desarrollo de estrategias para la reorganización del inventario existente, más allá de las operaciones de ubicación asociadas a la entrada o salida de materiales, permitiría mejorar la distribución del material y preparar el parque para futuras operaciones.
- **Simulación para la evaluación de las propuestas:** Para evaluar la calidad de la propuesta de ubicación, se requiere la realización de múltiples pruebas y comparaciones con otros estados. Su efectividad podría comprobarse mediante el desarrollo de un sistema de simulación, que permita generar escenarios de forma controlada y medir el impacto de las decisiones. Esto permitiría evaluar, no solo el resultado de una propuesta concreta, sino también el comportamiento global de la solución ante diferentes circunstancias.
- **Soporte multigrúa:** La solución actual se enfoca en un almacén que cuenta con una única grúa. El estudio de soluciones multigrúa obligaría a considerar las interacciones entre ellas y el establecimiento de estrategias globales de coordinación de los movimientos.

En conjunto, estas líneas de trabajo permitirían ampliar las capacidades desarrolladas en este trabajo y avanzar hacia un sistema de apoyo a la toma de decisiones capaz de abordar un mayor número de operaciones y escenarios de gestión del almacén.

Bibliografía

- Beck, Kent (2004). *Extreme Programming Explained: Embrace Change*.
- Boge, Sven y Sigrid Knust (jun. de 2023). «The blocks relocation problem with item families minimizing the number of reshuffles». En: *OR Spectrum* 45.2, págs. 395-435. ISSN: 1436-6304. DOI: [10.1007/s00291-022-00703-x](https://doi.org/10.1007/s00291-022-00703-x). URL: <https://doi.org/10.1007/s00291-022-00703-x>.
- Chen, Lifen, Jiajun Lin y Shihao Xu (jun. de 2025). «A Study on Container Storage Optimization in Yards Based on a Hyper-Heuristic Algorithm with a Q-Learning Mechanism». En: *International Journal of Computational Intelligence Systems* 18.1, pág. 24. ISSN: 1875-6883. DOI: [10.1007/s44196-025-00905-5](https://doi.org/10.1007/s44196-025-00905-5). URL: <https://doi.org/10.1007/s44196-025-00905-5>.
- Chen, Yu-Wang et al. (2012). «Development of hybrid evolutionary algorithms for production scheduling of hot strip mill». En: *Computers & Operations Research* 39.2, págs. 339-349. ISSN: 0305-0548. DOI: <https://doi.org/10.1016/j.cor.2011.04.009>. URL: <https://www.sciencedirect.com/science/article/pii/S0305054811001109>.
- Evans, Eric (2004). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. 1st. Boston, MA: Addison-Wesley Professional. ISBN: 978-0321125217.
- Fardad, Hamidreza, Faramarz Safi-Esfahani y Behrang Barekattain (28 de oct. de 2024). «Optimizing production planning and sequencing in hot strip mills: an approach using multi-objective genetic algorithms». En: *The Journal of Supercomputing* 81.1, pág. 88. ISSN: 1573-0484. DOI: [10.1007/s11227-024-06469-z](https://doi.org/10.1007/s11227-024-06469-z). URL: <https://doi.org/10.1007/s11227-024-06469-z> (visitado 27-04-2026).
- Fechter, Judith et al. (dic. de 2015). «Modeling a Lot-Aware Slab Stack Shuffling Problem». En: págs. 334-341. ISBN: 978-3-319-27339-6. DOI: [10.1007/978-3-319-27340-2_42](https://doi.org/10.1007/978-3-319-27340-2_42).
- Foundation, Cloud Native Computing (2019). *OpenTelemetry*. URL: <https://opentelemetry.io/es/> (visitado 01-05-2026).
- Kofler, Monika et al. (sep. de 2012). «Modelling and optimizing storage assignment in a steel slab yard». En: págs. 101-106. ISBN: 978-1-4673-4520-0. DOI: [10.1109/LINDI.2012.6319469](https://doi.org/10.1109/LINDI.2012.6319469).
- Martin, Robert C. (2012). *Arquitectura Limpia*. URL: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html> (visitado 28-05-2026).
- Mizuyama, Hajime (2020). «Agent-Based Modeling and Analysis of Dynamic Slab Yard Management in a Steel Factory». En: *Advances in Production Management Systems. The Path to Digital Transformation and Innovation of Production Management Systems*. Ed. por Bojan Lalic et al. Cham: Springer International Publishing, págs. 37-44. ISBN: 978-3-030-57993-7.
- Roljic, Biljana, Sebastian Leitner y Karl F. Doerner (2021). «Stacking and transporting steel slabs using high-capacity vehicles». En: *Procedia Computer Science* 180. Proceedings of

- the 2nd International Conference on Industry 4.0 and Smart Manufacturing (ISM 2020), págs. 843-851. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2021.01.334>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050921003884>.
- Sistemas de Gestión de Almacenes (SGA)* (2026). URL: https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_almacenes (visitado 25-06-2026).
- Song, Lulu et al. (abr. de 2023). «Improved Differential Evolution Algorithm for Slab Allocation and Hot-Rolling Scheduling Integration Problem». En: *Mathematics* 11, pág. 2050. DOI: [10.3390/math11092050](https://doi.org/10.3390/math11092050).
- Team, ASP.NET Core (2026). *¿Qué es Aspire?* URL: <https://aspire.dev/get-started/what-is-aspire/> (visitado 30-04-2026).
- Wang, Xu et al. (2025). «Optimal Assignment and Scheduling of Cranes in Slab Yard for Iron and Steel Production Enterprises». En: *IEEE Transactions on Automation Science and Engineering* 22, págs. 7349-7360. DOI: [10.1109/TASE.2024.3490039](https://doi.org/10.1109/TASE.2024.3490039).
- Zhao, Guodong et al. (nov. de 2019). «Model and Heuristic Solutions for the Multiple Double-Load Crane Scheduling Problem in Slab Yards». En: *IEEE Transactions on Automation Science and Engineering* PP, págs. 1-13. DOI: [10.1109/TASE.2019.2946196](https://doi.org/10.1109/TASE.2019.2946196).
- Zhu, Wenbin et al. (2012). «Iterative Deepening A* Algorithms for the Container Relocation Problem». En: *IEEE Transactions on Automation Science and Engineering* 9.4, págs. 710-722. DOI: [10.1109/TASE.2012.2198642](https://doi.org/10.1109/TASE.2012.2198642).

Apéndice A

Manual de usuario

A.1. Visión general de la aplicación

Al acceder al cliente en el navegador, se muestra la pantalla de acceso que puede observarse en la Figura A.1, en la que se deberá introducir el usuario y la contraseña para acceder a GenStock. Podrá utilizarse una cuenta registrada en la propia aplicación o bien una cuenta de Microsoft.

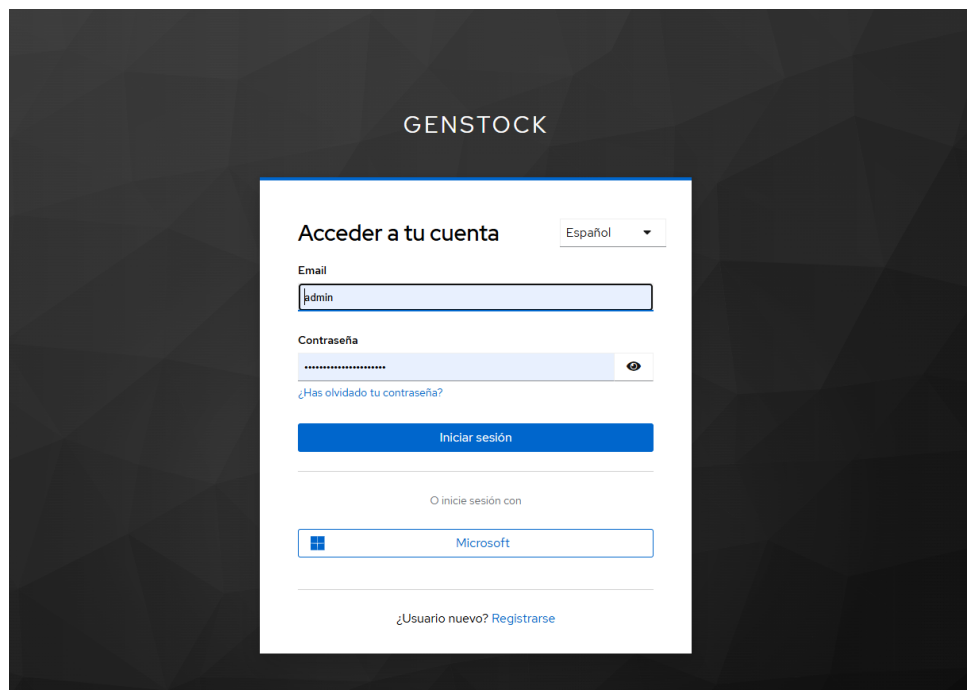


Figura A.1: Pantalla de acceso.

Como en cualquier pantalla de acceso, existe la opción de recuperar contraseña o registrarse como nuevo usuario como podemos ver en la Figura A.2.

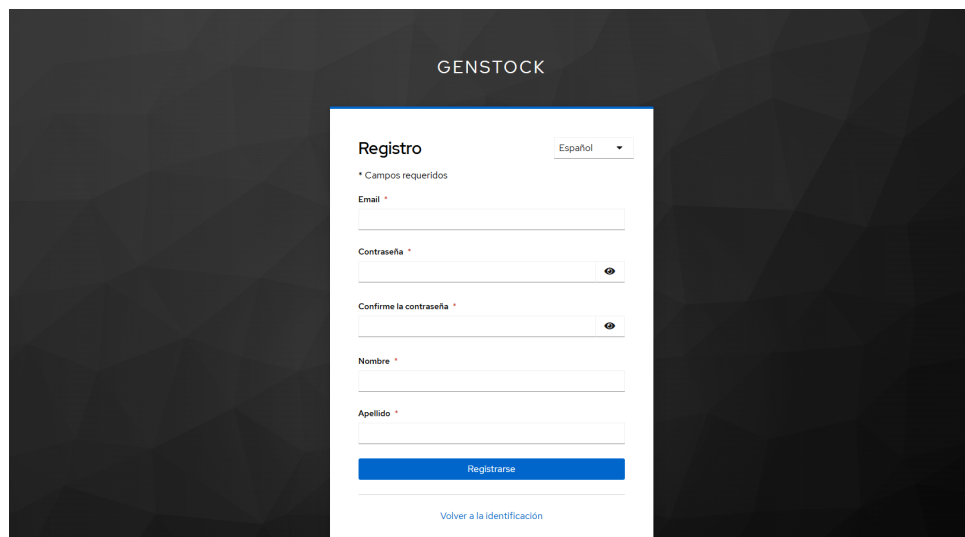


Figura A.2: Pantalla de registro.

Una vez validado el acceso, se mostrará una pantalla de inicio, que puede verse en la Figura A.3, donde se visualizan una serie de gráficos relacionados con el estado actual del parque.

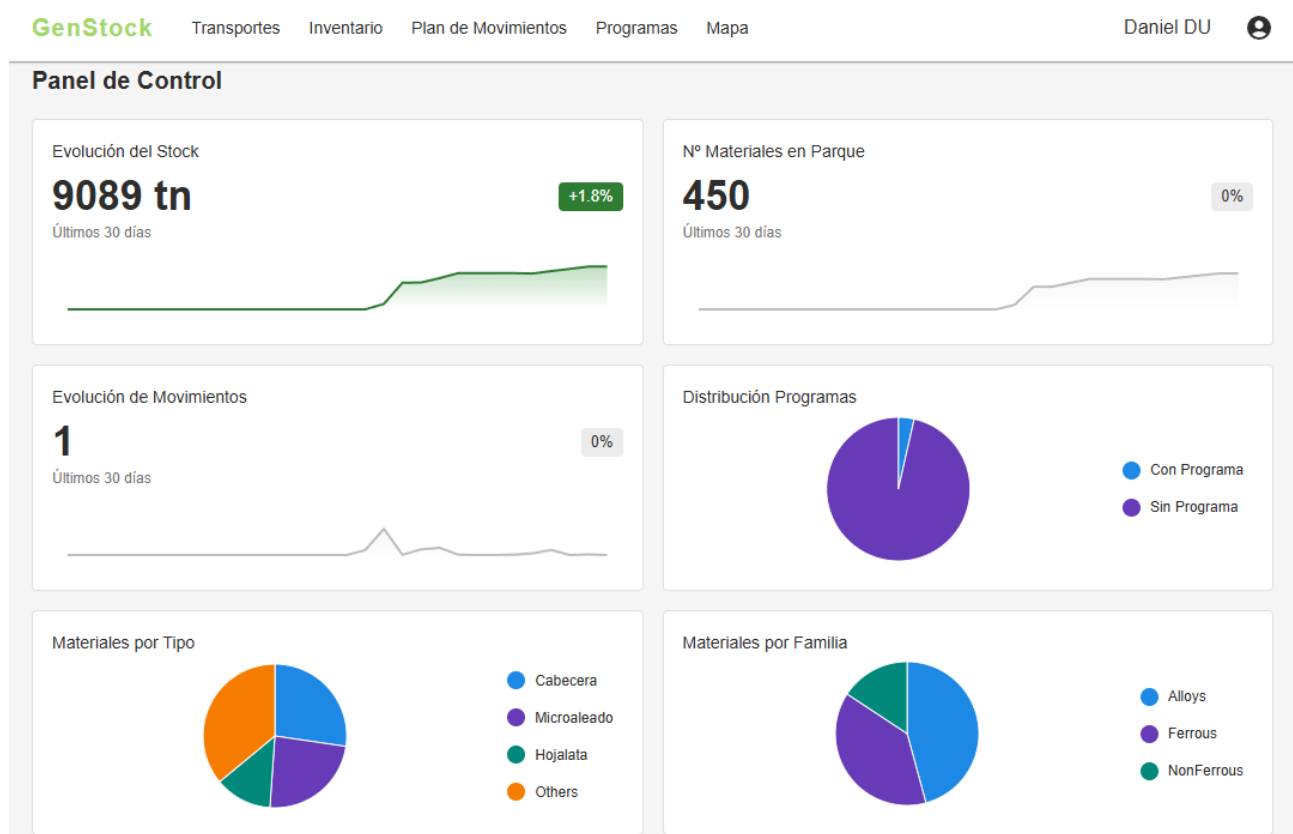


Figura A.3: Pantalla gráficos.

A modo de resumen visual tendremos los siguientes:

- **Evolución del inventario:** En este gráfico veremos el número total de toneladas re-

cibidas en el parque en los últimos 30 días, su distribución a lo largo del tiempo y el porcentaje de aumento o disminución con respecto al día anterior.

- **Número de materiales en el parque:** También para los últimos 30 días, se observa en este gráfico la distribución del número de materiales que se han recibido en el parque, junto con el porcentaje de aumento o disminución con respecto al día anterior.
- **Evolución de movimientos:** Como en los anteriores, se ve la distribución de los movimientos realizados en los últimos 30 días y el porcentaje de aumento o disminución con respecto al día anterior.
- **Distribución de programas:** En este gráfico se muestra el porcentaje de materiales con o sin programa.
- **Materiales por tipo:** Se visualiza la distribución de los materiales entre los distintos tipos permitidos.
- **Materiales por familia:** De igual manera, en este gráfico se verá la distribución de los materiales entre las distintas familias.

También puede verse en la Figura A.3 que en la parte superior de la aplicación aparecen las distintas opciones de menú disponibles en GenStock que pasaremos a explicar en los siguientes apartados.

A.2. Menú usuario

Empezaremos por las opciones de ajustes de usuario con las que se podrá personalizar la aplicación GenStock. Se observa en la Figura A.4 que se podrán modificar el tema, seleccionando si se desea la pantalla clara u oscura, y el idioma, estando disponibles español e inglés.

Ajustes

Apariencia

Tema
Claro

Idioma
Español

Cuenta

Nombre
Daniel DU

Guardar

Figura A.4: Pantalla de ajustes.

A.3. Transportes

Este menú de GenStock permite consultar el estado en el que se encuentran las diferentes zonas de transportes existentes en el parque, indicando para cada una, si el transporte se encuentra en el almacén y, si es el caso, si tiene materiales, tal y como vemos en la Figura A.5.

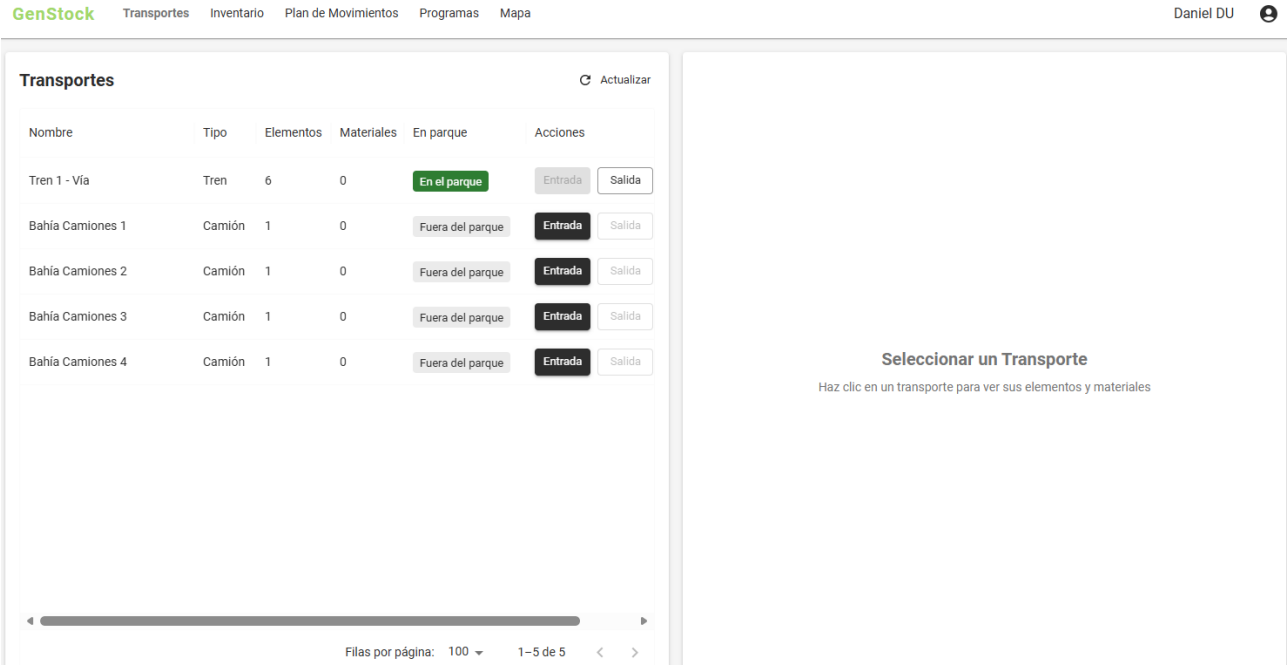


Figura A.5: Pantalla transportes sin selección.

Si seleccionamos un transporte con materiales, se visualizará un listado de los mismos, como podemos observar en la Figura A.6.

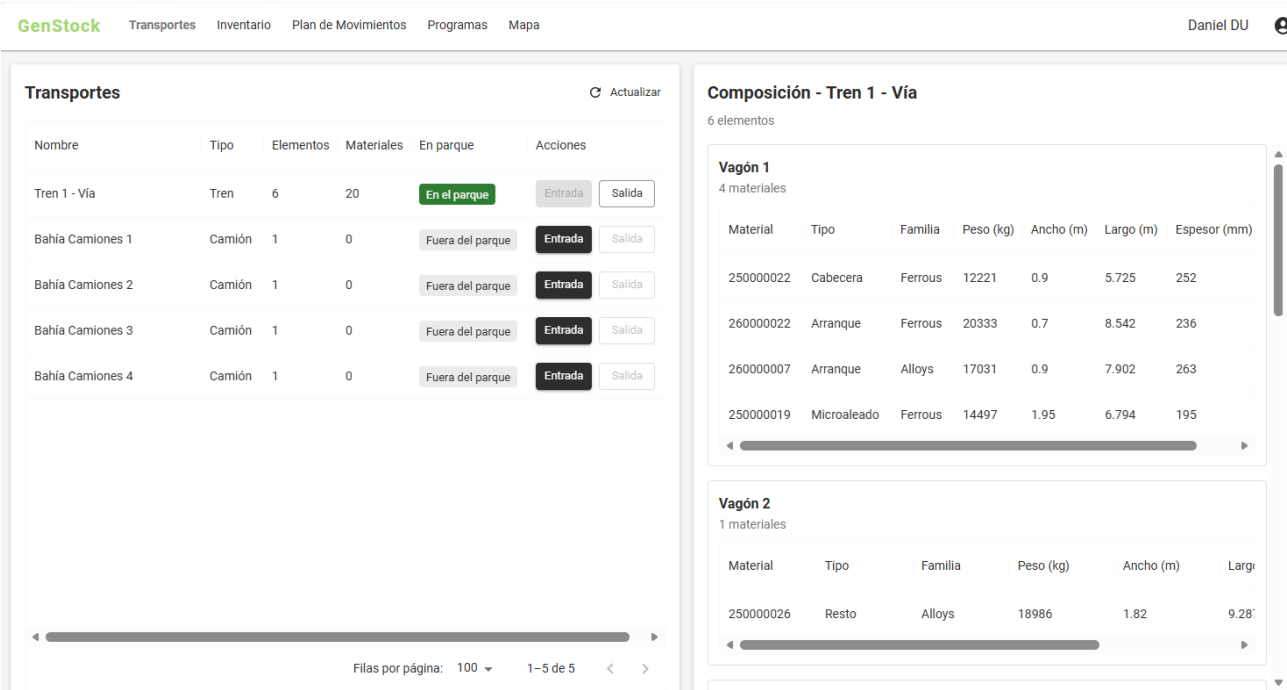


Figura A.6: Pantalla transportes con selección.

Además en este menú y como necesidad para este trabajo, se incorpora, a modo de simulación, la posibilidad de pedir la entrada de un transporte al parque pudiendo seleccionar si ha de venir con materiales o no. Esta acción se realiza a través del botón «Entrada» disponible para cada zona de transportes.

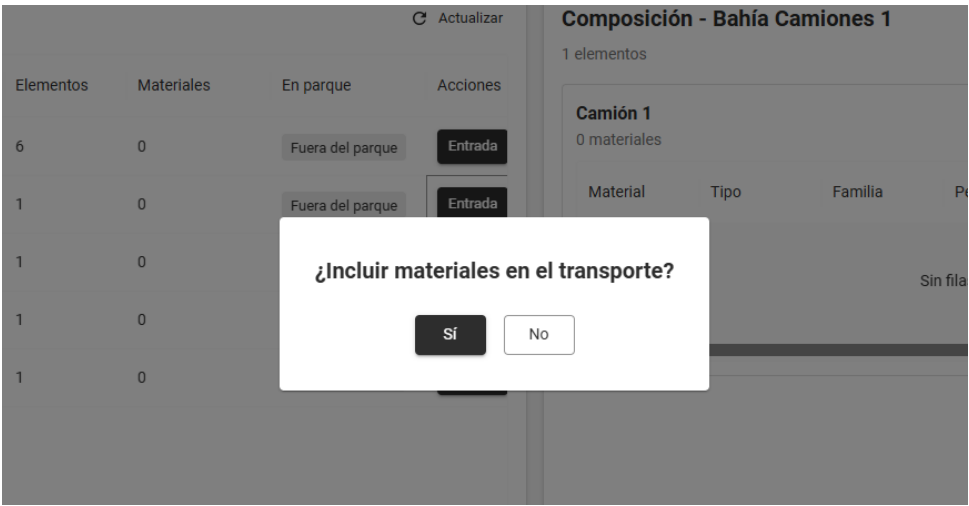


Figura A.7: Diálogo incluir materiales.

Indicando que «Sí» en el menú que se observa en la Figura A.7, el transporte accederá al parque, lo que se visualizará en el menú de mapa, se notificará la llegada y se visualizará la lista de materiales según se observa en la Figura A.8.

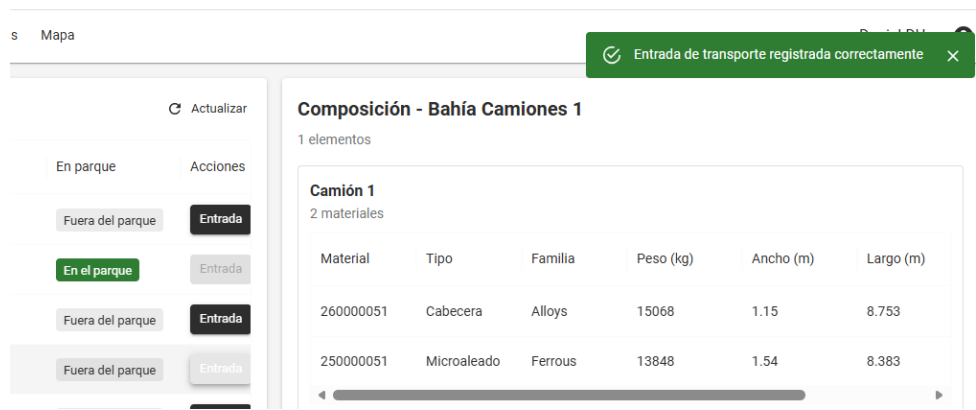


Figura A.8: Pantalla transportes indicando entrada de transporte.

De la misma manera, también como necesidad de este trabajo, existe, a modo de simulación, la posibilidad de pedir la salida de un transporte, la cual se notificará de manera similar a la entrada, tal y como podemos observar en la Figura A.9.

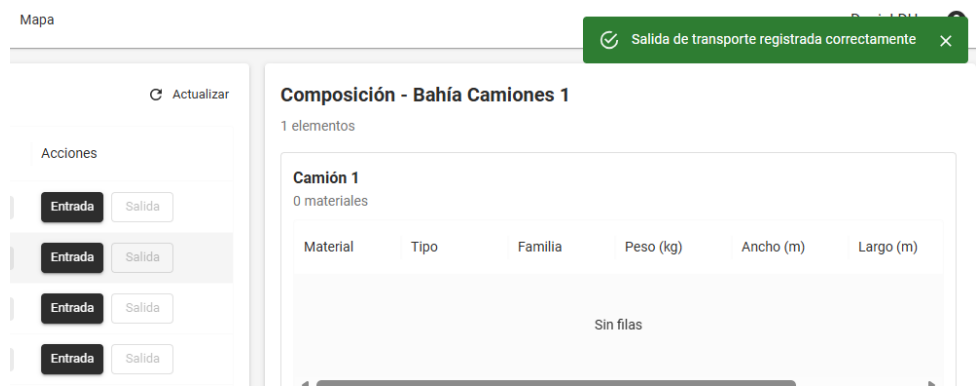


Figura A.9: Pantalla transportes indicando salida de transporte.

El transporte dejará de visualizarse en el menú mapa, del que se hablará con más detenimiento en la Sección A.7.

A.4. Inventario

Con esta opción se consulta la lista de materiales en el parque. Como puede observarse en la Figura A.10 puede buscarse un material concreto o bien filtrar por tipo de material. La búsqueda admite parte del código identificador del material.

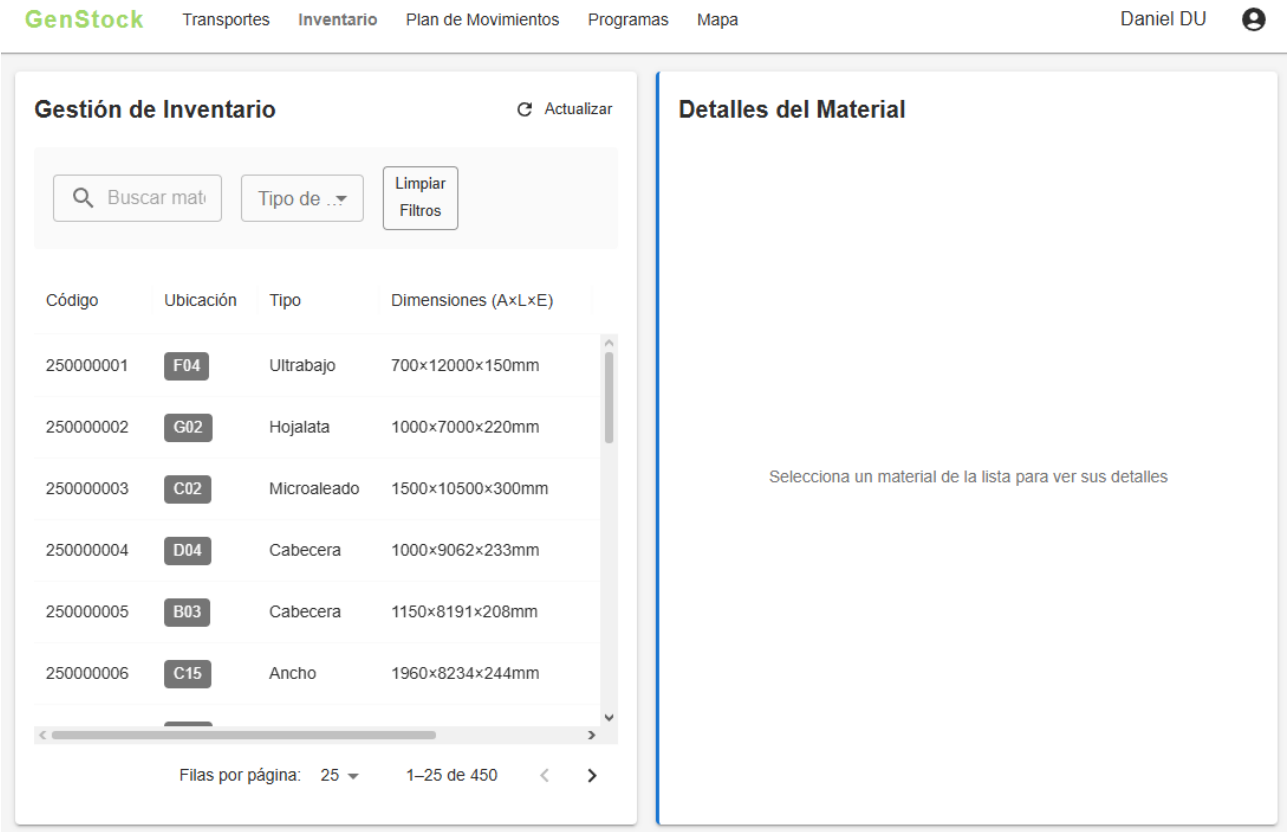


Figura A.10: Pantalla inventario sin material seleccionado.

Seleccionando uno de los materiales en el inventario, se visualizan detalles del mismo tales como sus dimensiones, el tipo, la información de programación disponible o su historial de movimientos, como podemos ver en la Figura A.11.

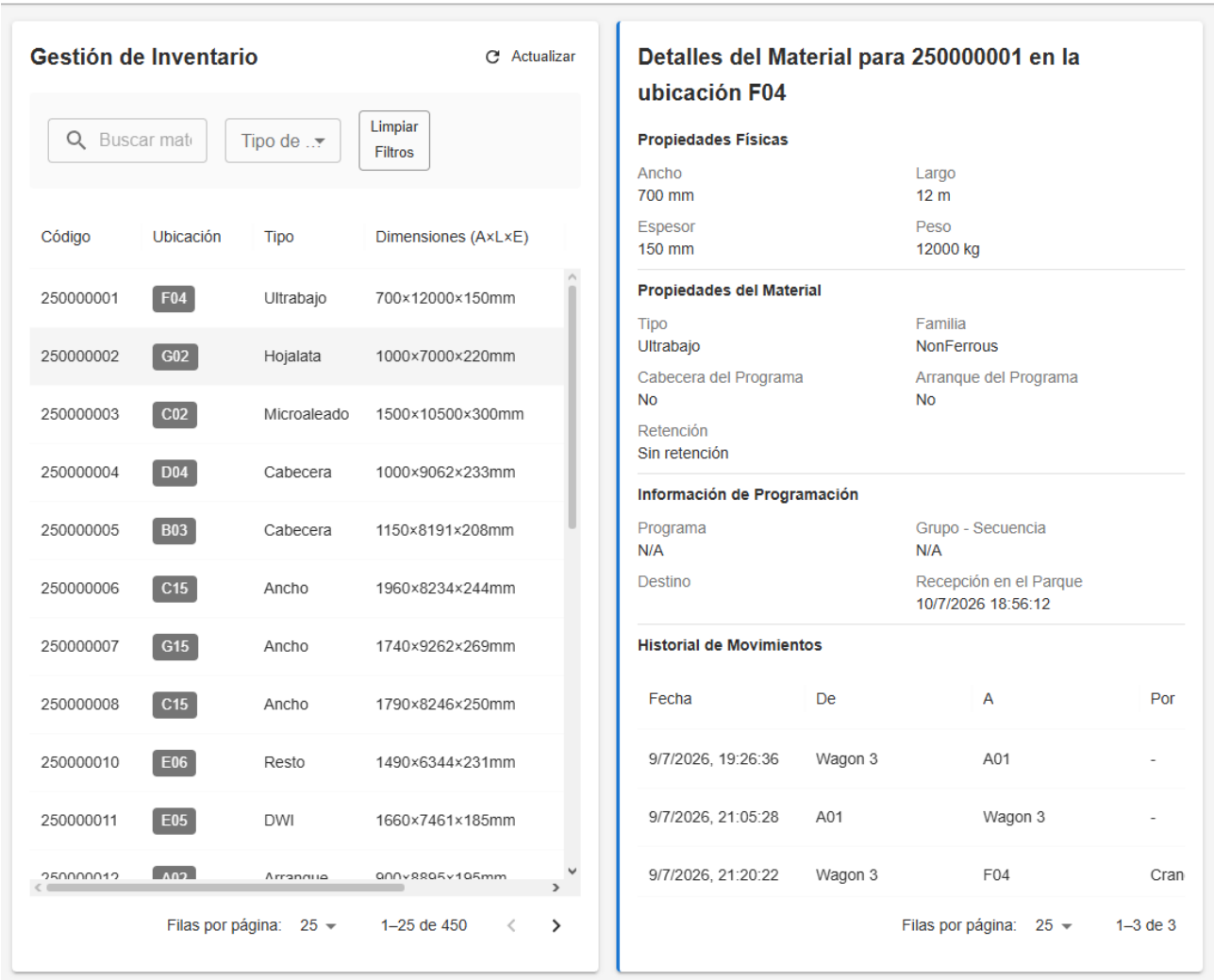


Figura A.11: Pantalla inventario con material seleccionado.

A.5. Plan de movimientos

Este menú se utilizará para crear los planes de movimientos dentro de GenStock. Para crear uno, deberá pulsarse el botón «Crear» que se observa en la Figura A.12.

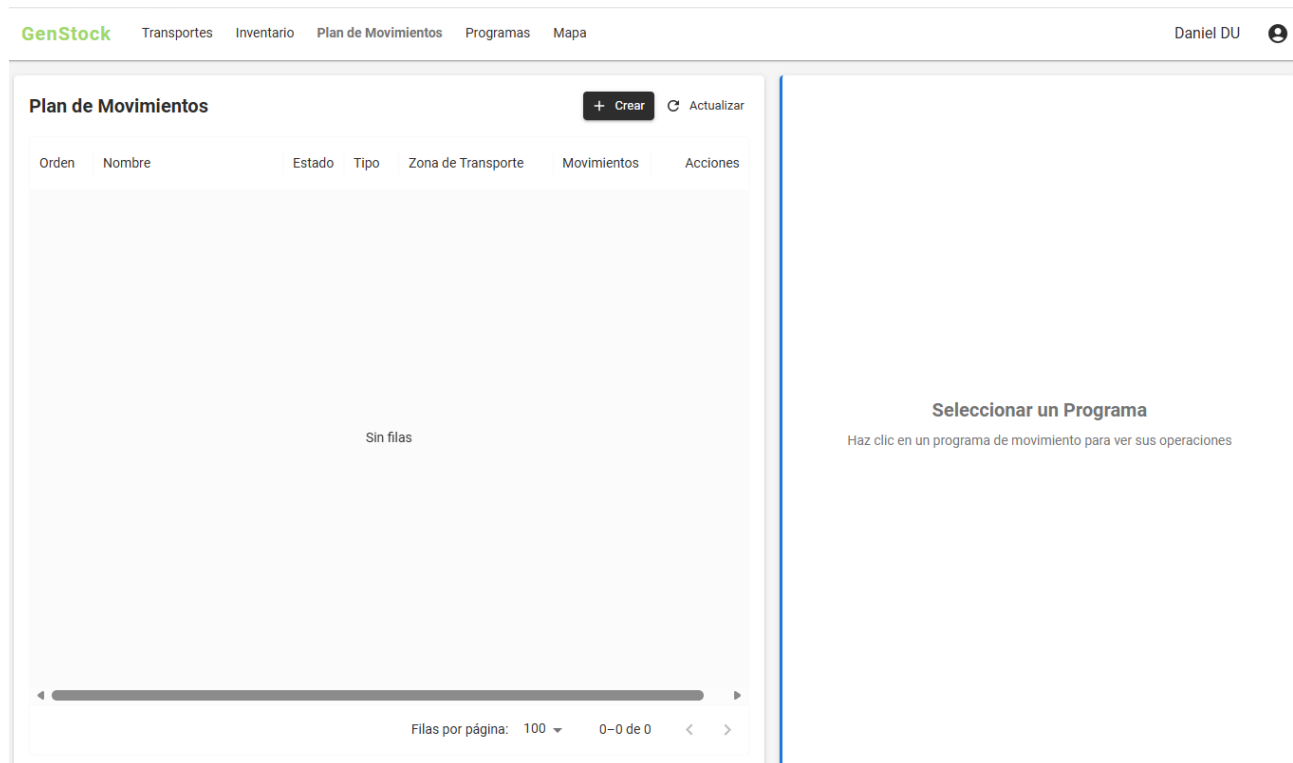


Figura A.12: Pantalla plan de movimientos vacía.

Puede verse en la Figura A.13 que existen cuatro tipos de planes de movimientos:

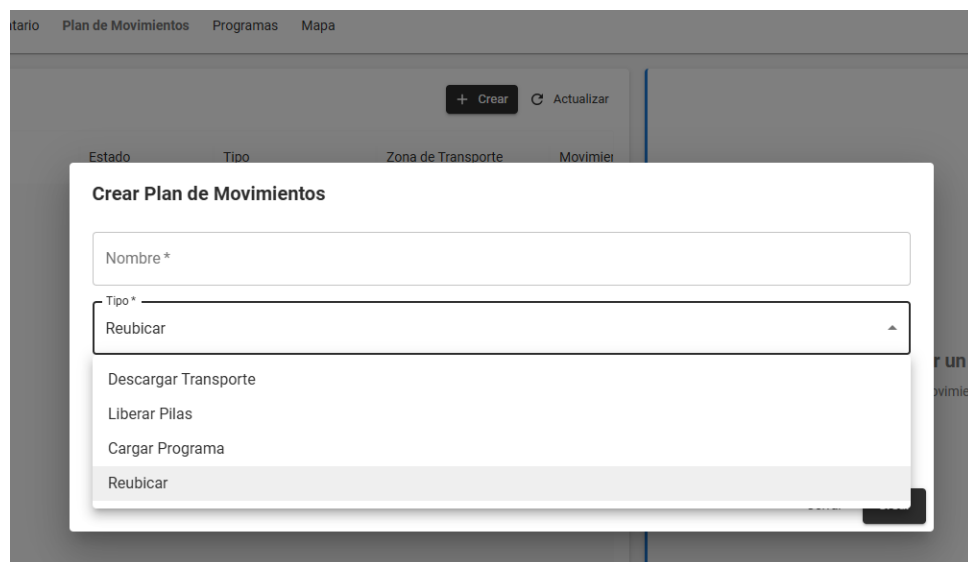


Figura A.13: Pantalla para crear distintos tipos de planes de movimientos.

- **Descargar transporte:** Se utiliza para la recepción de nuevo material y su descarga a las pilas según el algoritmo diseñado para la optimización del parque.
- **Liberar pilas:** Esta opción está disponible para permitir al operador vaciar una pila si estima que es necesario.
- **Cargar programa:** Empleado para la generación de planes para la carga de un programa

de laminación en un transporte.

- **Reubicar:** Permite al operador realizar una planificación manual moviendo materiales entre pilas.

A.5.1. Descarga de transporte

Para realizar la descarga de un transporte le asignaremos un nombre al plan de movimientos y deberá elegirse el transporte a descargar entre los presentes en el parque que se mostrarán en el desplegable. Podemos ver un ejemplo en la Figura A.14.

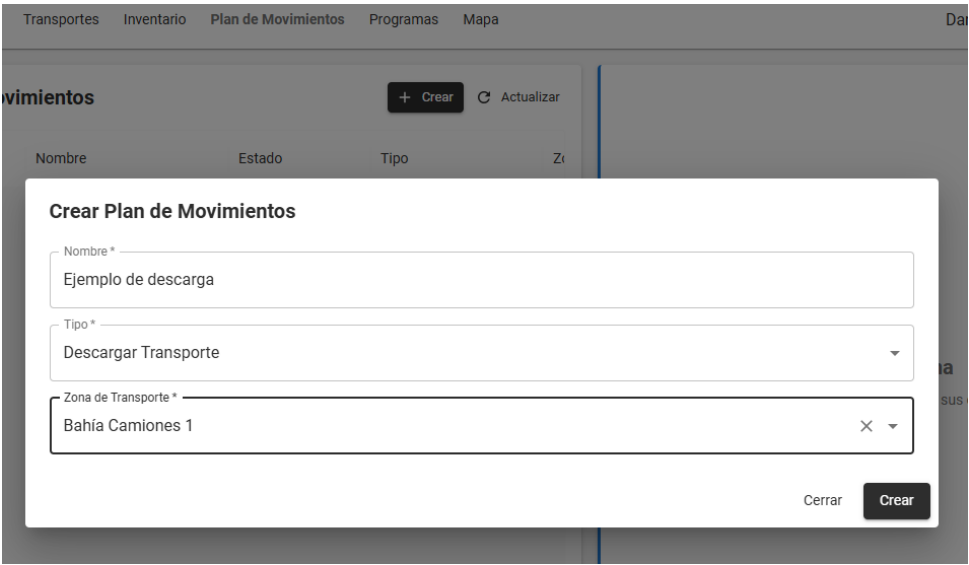


Figura A.14: Diálogo crear plan de movimientos de tipo descarga de transporte.

Una vez creado el plan, en la pantalla principal del menú plan de movimientos, podremos ver, tal y como se observa en la Figura A.15, la lista de los distintos movimientos con su estado, el número de elementos a mover, pila de origen, pila de destino y el identificador de cada material. Tanto en el origen como en el destino, además de la pila, se indica entre paréntesis la altura del material dentro de la misma.

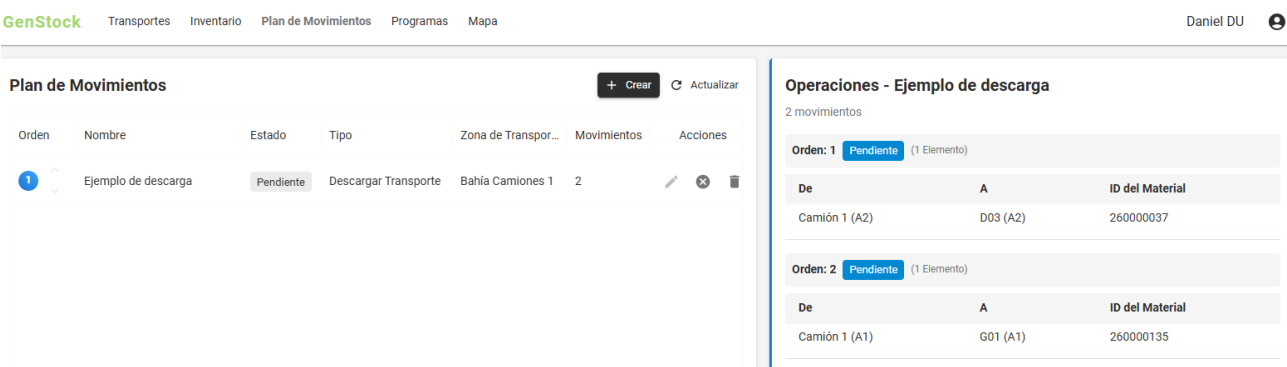


Figura A.15: Pantalla con los movimientos para la descarga de un camión.

Obsérvese que cada bloque, con su número de elementos a mover, representan una ganchada. En el ejemplo de la Figura A.16 puede verse como para cada movimiento, en la ganchada podrá ir más de un material.

Operaciones - Ejemplo descarga

20 movimientos

Orden: 1

Pendiente

(2 Elementos)

De	A	ID del Material
Vagón 2 (A4)	C04 (A5)	250000240
Vagón 2 (A3)	C04 (A4)	260000439

Orden: 2

Pendiente

(1 Elemento)

De	A	ID del Material
C04 (A5)	E05 (A4)	250000240

Orden: 3

Pendiente

(2 Elementos)

De	A	ID del Material
Vagón 3 (A4)	E10 (A10)	260000745
Vagón 3 (A3)	E10 (A9)	250000488

Figura A.16: Pantalla con varios elementos en ciertos movimientos.

Según la grúa vaya moviendo materiales, irán actualizándose los estados de cada movimiento. Así, en la Figura A.17 puede apreciarse que el primer movimiento ya se ha completado, el segundo está en curso y el resto están pendientes.

Operaciones - Ejemplo descarga

20 movimientos

Orden: 1 Completado (2 Elementos)		
De	A	ID del Material
Vagón 2 (A4)	C04 (A5)	250000240
Vagón 2 (A3)	C04 (A4)	260000439

Orden: 2 En Curso (1 Elemento)		
De	A	ID del Material
C04 (A5)	E05 (A4)	250000240

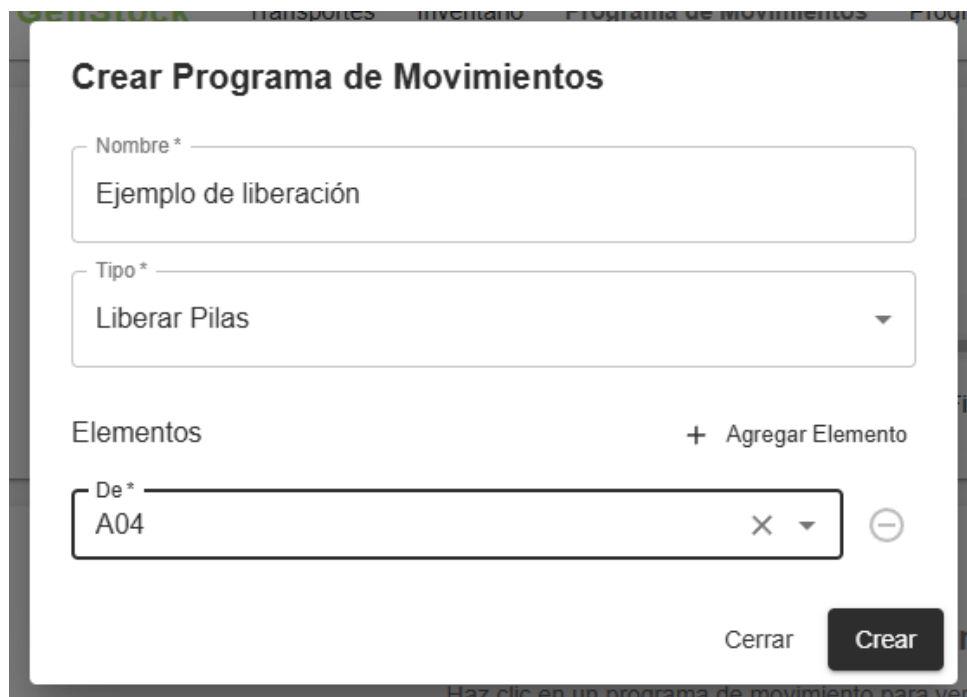
Orden: 3 Pendiente (2 Elementos)		
De	A	ID del Material
Vagón 3 (A4)	E10 (A10)	260000745
Vagón 3 (A3)	E10 (A9)	250000488

Figura A.17: Pantalla con movimientos en distintos estados.

Una vez finalizados todos los movimientos, el plan dejará de visualizarse.

A.5.2. Liberar pilas

De la misma manera si lo que se desea es liberar una pila, además de asignarle un nombre al plan habrá que seleccionar la pila a liberar, como se observa en la Figura A.18. Además, podrán indicarse varias pilas a vaciar utilizando la opción agregar elemento.



Crear Programa de Movimientos

Nombre *
Ejemplo de liberación

Tipo *
Liberar Pilas

Elementos + Agregar Elemento

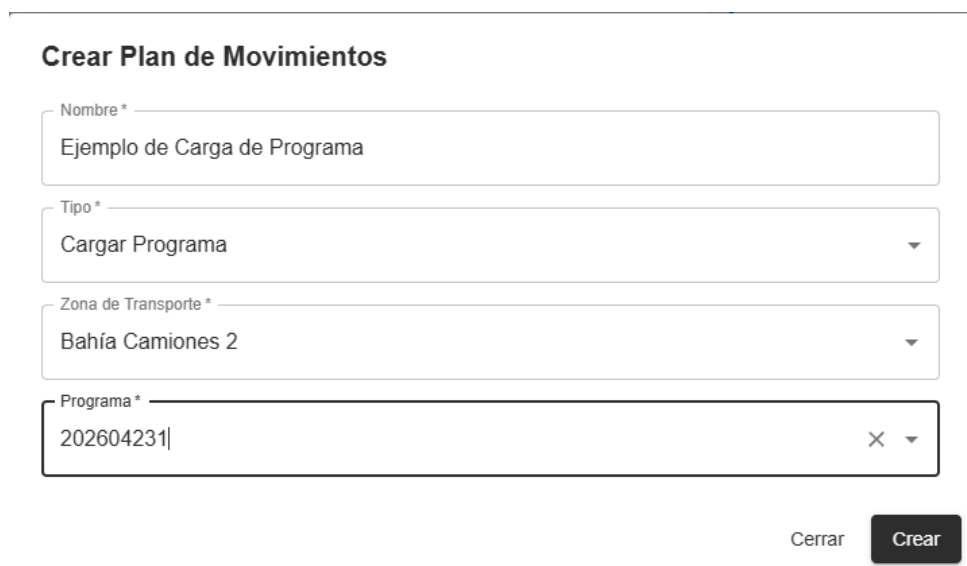
De *
A04

Cerrar Crear

Figura A.18: Diálogo crear plan de movimientos de tipo liberar pila.

A.5.3. Cargar programa

Para la creación de un plan de movimientos que cargue un programa en un transporte, habrá que asignarle un nombre y seleccionar el programa a cargar según se observa en la Figura A.19.



Crear Plan de Movimientos

Nombre *
Ejemplo de Carga de Programa

Tipo *
Cargar Programa

Zona de Transporte *
Bahía Camiones 2

Programa *
202604231

Cerrar Crear

Figura A.19: Diálogo crear plan de movimientos de tipo cargar programa.

A.5.4. Reubicar

Si se hace necesario reubicar algún material, en este caso, tras asignarle un nombre, deberemos seleccionar el origen, el destino y el identificador del material a reubicar. Como puede observarse en la Figura A.20 podrá reubicarse una lista de materiales utilizando la opción agregar elemento.

Crear Plan de Movimientos

Nombre *
Ejemplo Reubicación

Tipo *
Reubicar

Elementos

+ Agregar Elemento

De * A * ID del Material *

A04 A08 250001009 (A8)

A06 A04

ID del Material *

250001324 (A1)

260000265 (A2)

250000215 (A3)

Close

Crear

Figura A.20: Diálogo crear plan de movimientos de tipo reubicar.

A.6. Programas

En este menú se visualizará la lista de programas disponibles en el parque y la lista de elementos que compone el programa que hayamos seleccionado, tal y como se ve en la Figura A.21.

Programas

Actualizar

Orden	Nombre	% Expedición	En parque	Últ. Sec. Exp.
1	202604231	25%	16 / 72	66

Elementos - 202604231

72 elementos

Secuencia	Grupo	Material	Ubicac
1	1	250000882	-
2	2	250000462	-
3	3	250001718	-
4	4	250000247	-
5	5	260000262	-
6	6	250000107	-
7	7	250001759	-

Figura A.21: Pantalla programas.

En la lista de los elementos, aparecerán en gris los ya expedidos y en blanco los que todavía no han sido expedidos.

A.7. Mapa

En esta sección se visualizará el parque como puede observarse en la Figura A.22.

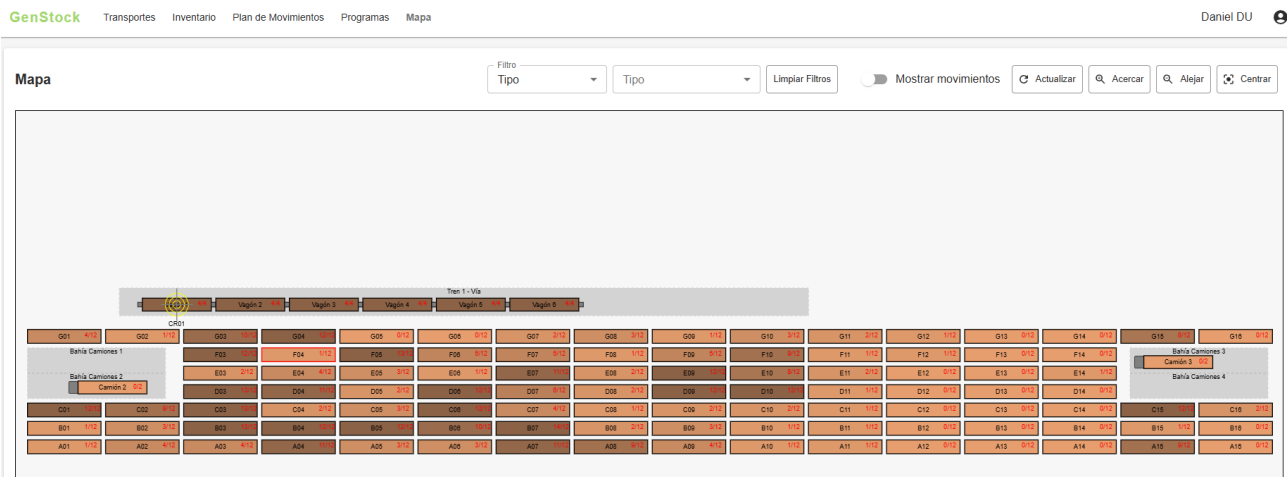


Figura A.22: Visión general del mapa del parque.

Tanto en las pilas como en los transportes, se verá el número de elementos ubicados. Además, como puede apreciarse en la Figura A.23 las distintas ubicaciones aparecerán en diferentes tonos

según estén vacías o con pocos elementos (pilas «G05», «F04»), próximas a completarse (pilas «G03», «D04») o ya completas (pilas «G04», «F03»).

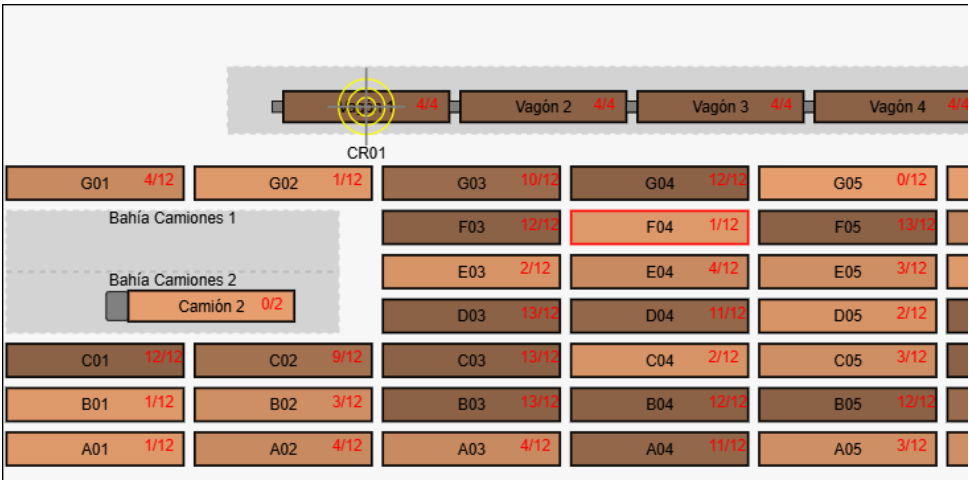


Figura A.23: Sección del mapa del parque.

También se verán en distinto color las pilas con materiales que cumplan el filtro seleccionado en la pantalla. En el ejemplo de la Figura A.24, se visualizan en azul las pilas con materiales de tipo cabecera.

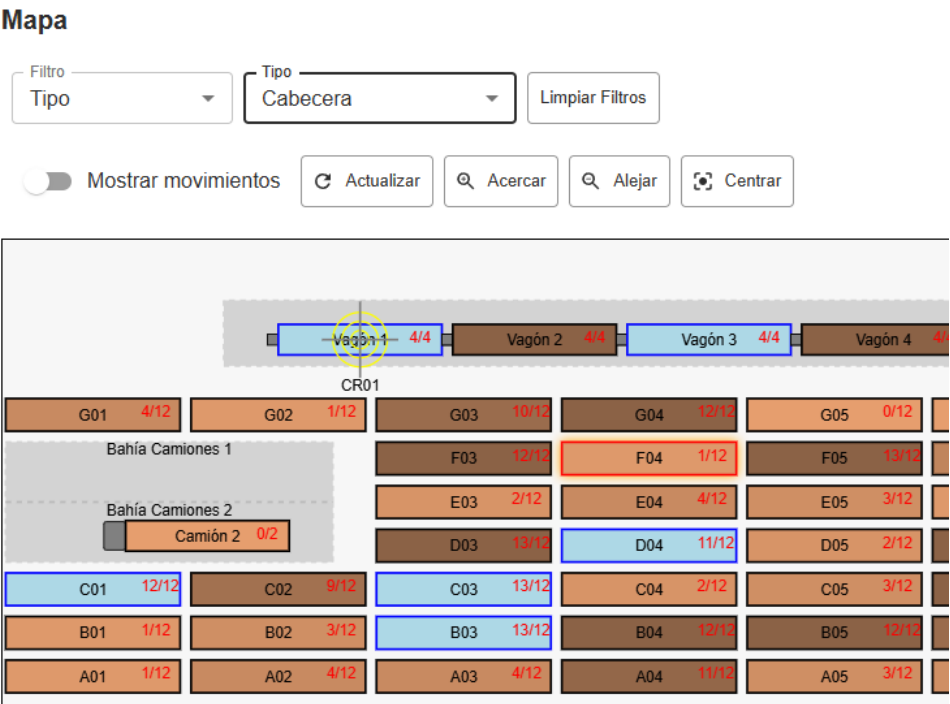


Figura A.24: Pantalla mapa con tipo seleccionado.

Para cada pila tendremos varias opciones en el menú contextual como se observa en la Figura A.25.

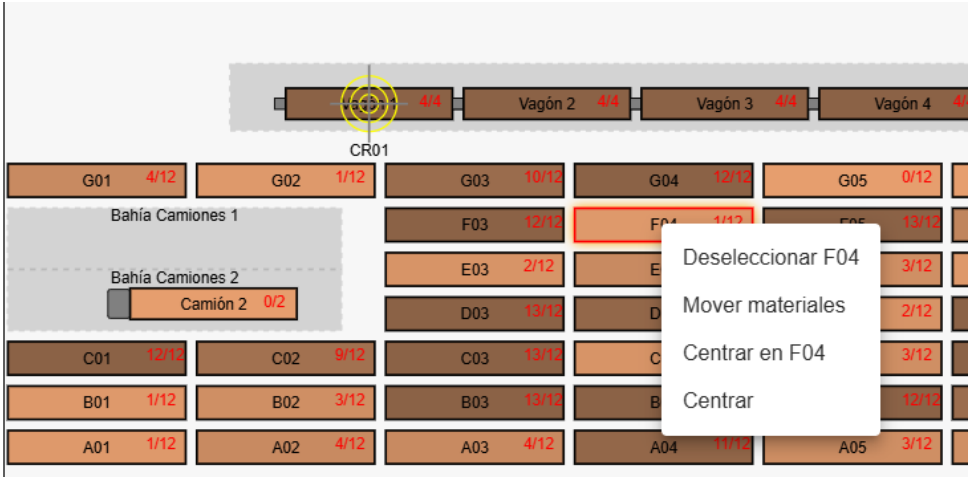


Figura A.25: Menú botón derecho sobre una pila.

Además puede seleccionarse cualquier pila y veremos el detalle de los materiales ubicados en la misma, con una vista frontal (largo de los desbastes) y lateral (ancho de los materiales), sus dimensiones, peso, tipo, familia y destino tal y como puede verse en la Figura A.26.

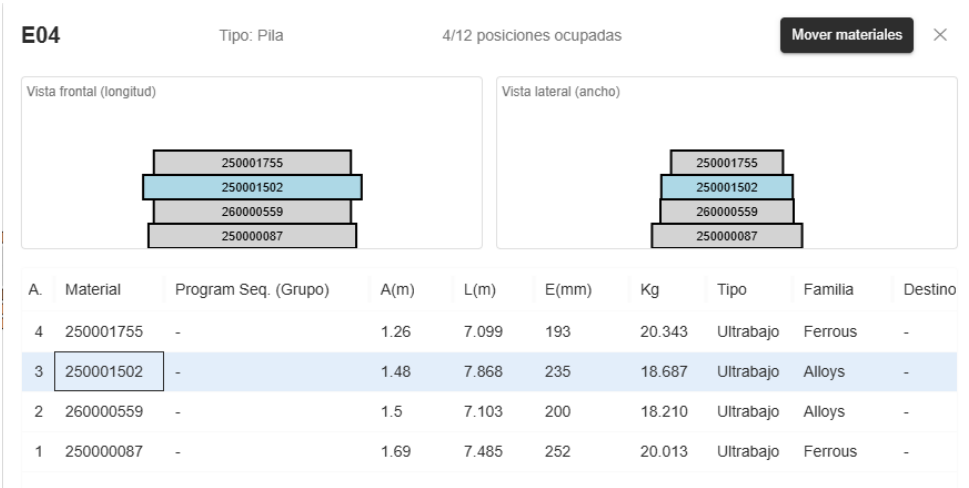


Figura A.26: Pantalla detalle de una pila.

Tanto en el menú contextual como en el detalle de una pila, existe la opción de mover materiales. Como se ve en la Figura A.27 en la parte izquierda de la pantalla aparecen los materiales en la pila seleccionada con sus dimensiones y en la parte derecha se seleccionará la pila destino de la que visualizaremos los materiales ya ubicados también con sus dimensiones. En el centro de la pantalla podremos indicar el número de materiales a mover.

Mover materiales

B08 Tipo: Pila 1/12 posiciones ocupadas

Vista frontal (longitud)

Vista lateral (ancho)

A.	Material	Prog...	A.	L.	E...	Destino
1	260000651	-	1.98	7.12	216	-

A01 Tipo: Pila 2/12 posiciones ocupadas

Vista frontal (longitud)

Vista lateral (ancho)

A.	Material	Prog...	A.	L.	E...	Destino
2	260000354	-	1.92	6.266	206	-
1	260000047	-	1.53	8.032	212	-

→

A mover

1

Máximo desplazable: 2

Cancelar Confirmar

Figura A.27: Pantalla para mover materiales.

Cuando existan planes de movimiento, existe la posibilidad de visualizar los movimientos restantes mediante flechas de origen a destino como puede observarse en la Figura A.28.

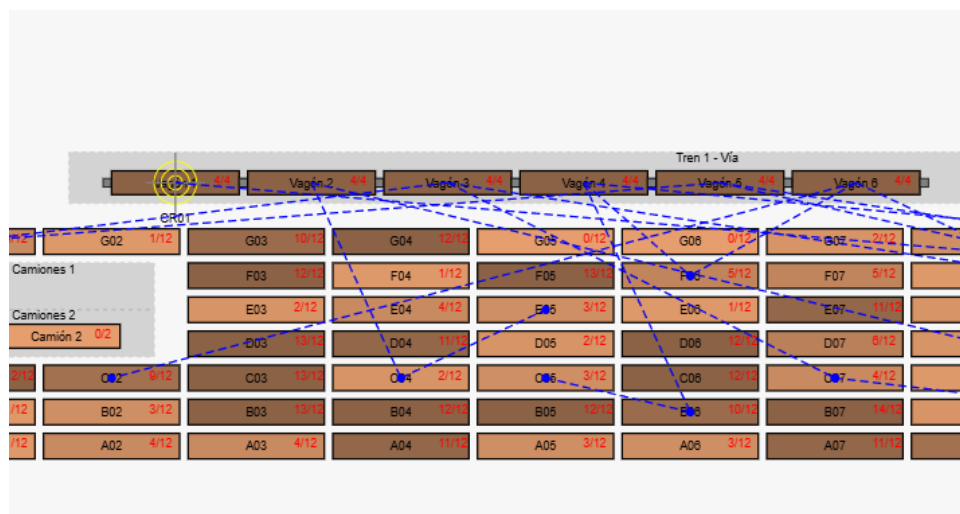


Figura A.28: Pantalla mapa mostrando los movimientos.

A.8. Administración

Si el usuario con el que se accede a GenStock es administrador, existirá un menú adicional. En esta opción administración se podrá gestionar:

- Usuarios.
- Destinos.
- Grúas.

■ Asignación de materiales.

A.8.1. Usuarios

Como se observa en la Figura A.29 en este apartado podrán darse de alta, editar o eliminar usuarios.

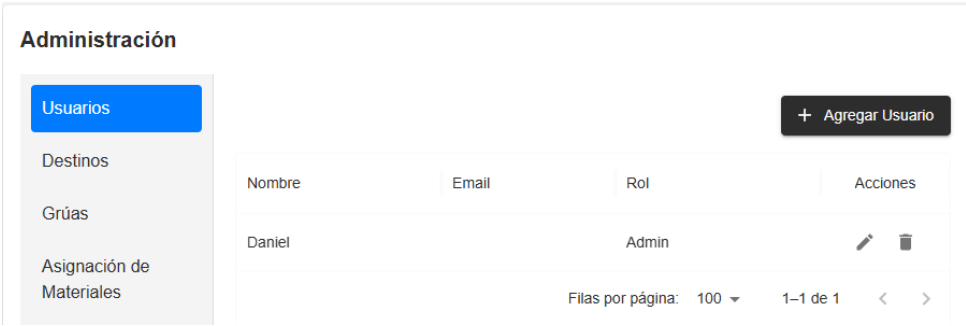


Figura A.29: Pantalla para administración de usuarios.

A.8.2. Destinos

En este caso se gestionan los destinos que pueden asignarse a los materiales del inventario. La pantalla de gestión es la que vemos en la Figura A.30.



Figura A.30: Pantalla para administración de destinos.

A.8.3. Grúas

En este menú podrán gestionarse las grúas dentro del sistema. Como se ve en la Figura A.31, la única opción existente es la edición para indicar si la grúa está o no operativa.

Administración

- Usuarios
- Destinos
- Grúas**
- Asignación de Materiales

Nombre	Operativa	Acciones
CR01	Sí	

Filas por página: 100 1-1 de 1

Figura A.31: Pantalla para administración de grúas.

A.8.4. Asignación de materiales

Dentro de esta opción podremos hacer varias operaciones. Por un lado, como vemos en la Figura A.32 podremos seleccionar un número de elementos a los que asignar destinos y/o tipos de materiales. Para el primer caso, se podrá asignar destino a cualquier material que no esté en un programa. Para el caso del tipo, se podrá asignar únicamente a materiales de una familia.

Administración

- Usuarios
- Destinos
- Grúas
- Asignación de Materiales**

A asignar: 1
Max asignable: 20

Destino

+ Asignar Destino

Tipo de Material

+ Asignar Tipo

+ Generar Programa

Figura A.32: Pantalla de administración para asignación de materiales.

Por otro lado, y de manera totalmente independiente a las asignaciones de tipo y destino, también existe la posibilidad de generar un programa. Para este caso no hay que hacer nada más que pulsar el botón.

Apéndice B

Guía de instalación del entorno

El uso de Aspire facilita la configuración y puesta en marcha del entorno de desarrollo de la aplicación. Como se ha descrito en la Sección 3.2.3, Aspire permite definir los diferentes servicios que componen la aplicación junto con las relaciones entre los mismos. Además, proporciona herramientas para facilitar la ejecución y gestión de aplicaciones distribuidas y su posterior despliegue en diferentes entornos.

En el caso de GenStock, esta configuración permite iniciar conjuntamente los diferentes componentes de la aplicación sin necesidad de configurar y ejecutar cada uno de ellos de forma independiente.

B.1. Requisitos previos

Para configurar el entorno de desarrollo son necesarios los siguientes componentes:

- **Git:** utilizado para obtener el código fuente del proyecto del sistema de control de versiones alojado en Azure DevOps.
- **Visual Studio 2022:** entorno de desarrollo utilizado para trabajar con el código y ejecutar el proyecto.
- **.NET 10 SDK:** necesario para compilar y ejecutar los proyectos desarrollados.
- **Podman:** utilizado como motor de ejecución de contenedores empleados por los diferentes servicios de la aplicación.
- **Node.js 22.x:** entorno de ejecución utilizado por el cliente de la aplicación.
- **Node Package Manager (npm):** gestor de paquetes utilizado para instalar las librerías de las que depende el cliente.

B.2. Puesta en marcha

Una vez instalados los requisitos previos, la configuración del entorno se realiza mediante los siguientes pasos:

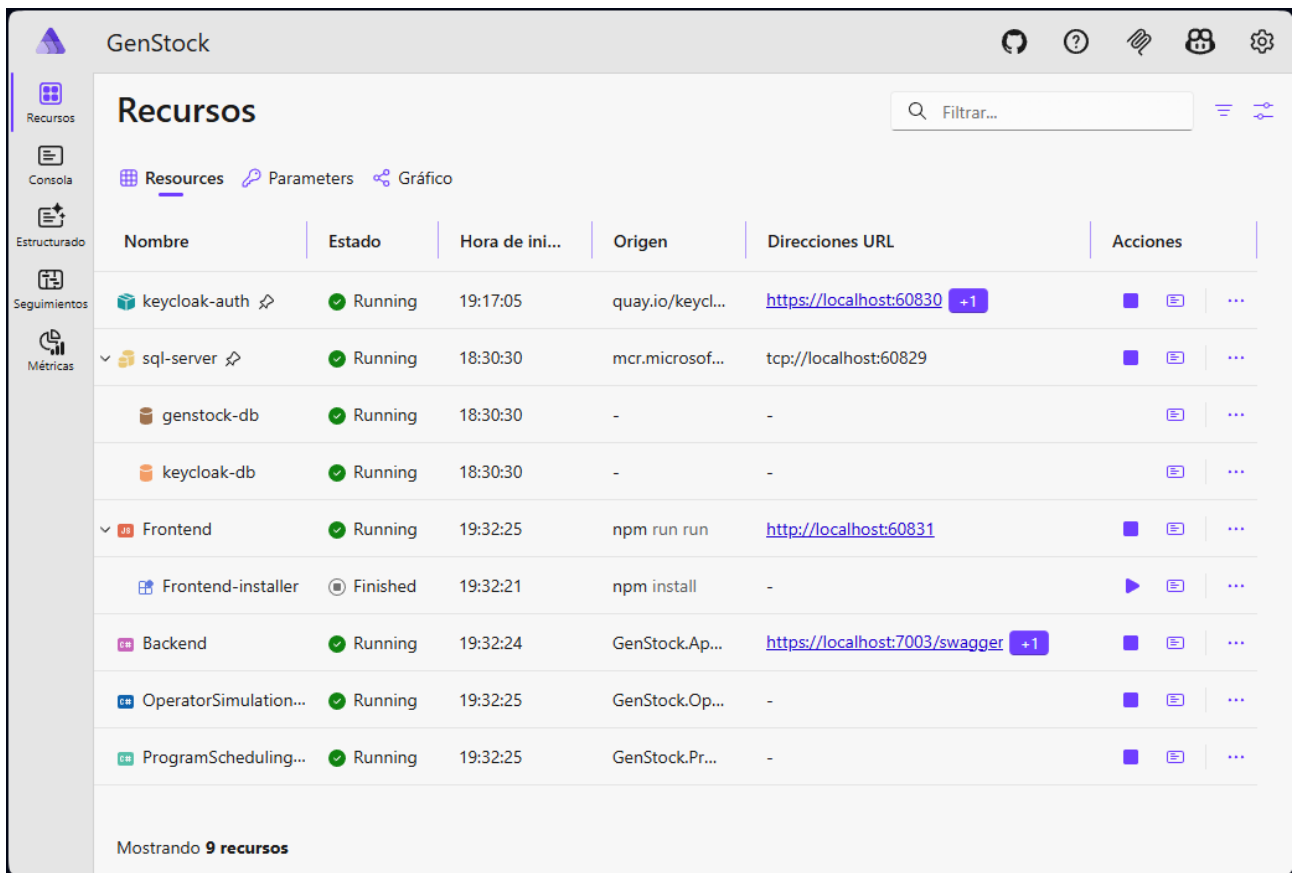
1. **Clonar el repositorio:** obtener el código fuente del proyecto mediante Git.
2. **Configurar los secretos de usuario:** establecer mediante la herramienta de secretos de usuario de .NET los valores de configuración sensibles necesarios para la ejecución de la aplicación. Las claves requeridas son:

- `GenStock:ServiceClients:ProgramScheduling`
- `GenStock:ServiceClients:OperatorSimulator`
- `GenStock:Api`
- `GenStock:AzureMSAuth`

Los tres primeros secretos se utilizan para configurar las credenciales de los clientes de servicio empleados en la comunicación segura entre los diferentes componentes de la aplicación. Esta configuración se utiliza para crearlos en el arranque de Keycloak y para autenticar los servicios. El último se corresponde con el secreto utilizado para configurar la autenticación mediante Microsoft. Para su obtención se ha configurado una aplicación en Azure.

3. **Ejecutar el AppHost:** iniciar el proyecto *AppHost* desde Visual Studio. Aspire se encarga de iniciar y orquestar los diferentes servicios definidos en él, incluyendo los servicios del servidor, API, cliente, sistema de autenticación y base de datos.

Una vez iniciado el proyecto, Aspire orquesta el arranque de GenStock y permite acceder a los diferentes recursos y servicios de la aplicación desde su interfaz de gestión. El cliente web estará disponible a través de la dirección: `http://localhost:3000`.



The screenshot displays the GenStock Aspire dashboard. The left sidebar contains navigation icons for Recursos, Consola, Estructurado, Seguimientos, and Métricas. The main area is titled 'Recursos' and includes a search bar and tabs for Resources, Parameters, and Gráfico. A table lists the following resources:

Nombre	Estado	Hora de ini...	Origen	Direcciones URL	Acciones
keycloak-auth	Running	19:17:05	quay.io/keycl...	https://localhost:60830	[Stop] [Logs] [More]
sql-server	Running	18:30:30	mcr.microsof...	tcp://localhost:60829	[Stop] [Logs] [More]
genstock-db	Running	18:30:30	-	-	[Logs] [More]
keycloak-db	Running	18:30:30	-	-	[Logs] [More]
Frontend	Running	19:32:25	npm run run	http://localhost:60831	[Stop] [Logs] [More]
Frontend-installer	Finished	19:32:21	npm install	-	[Play] [Logs] [More]
Backend	Running	19:32:24	GenStock.Ap...	https://localhost:7003/swagger	[Stop] [Logs] [More]
OperatorSimulation...	Running	19:32:25	GenStock.Op...	-	[Stop] [Logs] [More]
ProgramScheduling...	Running	19:32:25	GenStock.Pr...	-	[Stop] [Logs] [More]

Mostrando 9 recursos

Figura B.1: Página principal Aspire en entorno de desarrollo.