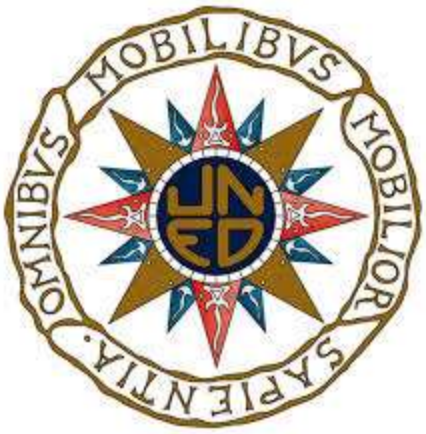


UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Grado en Ingeniería Informática

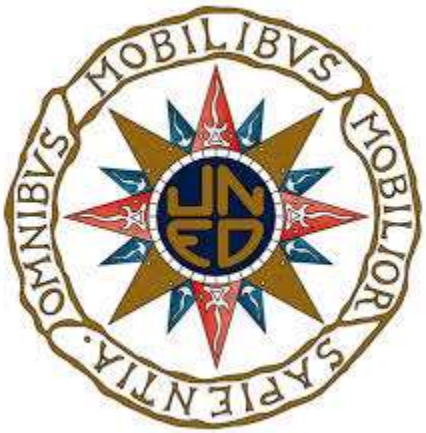
Sistema interactivo de apoyo a la docencia en circuitos eléctricos de corriente continua

Iñigo Atutxa Álvarez

Dirigido por: Carla Martín Villalba

Codirigido por: Alfonso Urquía Moraleda

Curso 2025/2026, convocatoria junio



Sistema interactivo de apoyo a la docencia en circuitos eléctricos de corriente continua

Proyecto de Fin de Grado en Ingeniería Informática de modalidad específica

Realizado por: Iñigo Atutxa Álvarez

Dirigido por: Carla Martín Villalba

Codirigido por: Alfonso Urquía Moraleda

Fecha de lectura y defensa: junio 2026

Resumen

Este trabajo presenta el diseño y desarrollo de una aplicación web orientada al apoyo en la enseñanza del análisis de circuitos eléctricos en corriente continua. La herramienta está especialmente dirigida a entornos educativos preuniversitarios y cursos introductorios de ingeniería, donde el aprendizaje de esta materia suele presentar dificultades debido a la falta de herramientas accesibles y específicas.

El sistema desarrollado integra en una única plataforma un editor gráfico de circuitos, un motor de resolución automática y una plataforma educativa con gestión de usuarios. El editor permite la creación interactiva de circuitos mediante la inserción y conexión de componentes eléctricos básicos, como resistencias y fuentes independientes. El motor de cálculo, basado en el método de Análisis Nodal Modificado, permite obtener de forma automática las magnitudes eléctricas relevantes, proporcionando retroalimentación inmediata al usuario.

Asimismo, la aplicación incorpora funcionalidades orientadas al ámbito docente, como la gestión de ejercicios por parte del profesorado, la resolución por parte del alumnado y la evaluación automática de las soluciones. Esto favorece un aprendizaje más activo y autónomo, al tiempo que facilita el seguimiento del progreso del alumnado.

Desde el punto de vista técnico, el sistema se ha implementado siguiendo una arquitectura cliente-servidor, utilizando tecnologías web modernas como React en el frontend y Node.js en el backend. Esto permite garantizar la accesibilidad desde distintos dispositivos sin necesidad de instalación.

Los resultados obtenidos muestran que la herramienta cumple con los objetivos planteados, ofreciendo una solución funcional, intuitiva y adaptada a las necesidades del entorno educativo. Finalmente, se identifican posibles líneas de mejora, como la ampliación a circuitos de corriente alterna, la incorporación de nuevos componentes y la evolución de la plataforma educativa.

Abstract

This project presents the design and development of a web-based application aimed at supporting the teaching and learning of direct current (DC) circuit analysis. The tool is particularly focused on pre-university education and introductory engineering courses, where students often face difficulties due to the lack of accessible, education-oriented tools.

The developed system integrates three main components within a single platform: a graphical circuit editor, an automatic solving engine, and an educational platform with user management. The editor allows users to create circuits interactively by inserting and connecting basic electrical components such as resistors and independent sources. The solving engine, based on the Modified Nodal Analysis (MNA) method, computes electrical magnitudes automatically, providing immediate feedback to the user.

Additionally, the application includes features specifically designed for educational use, such as exercise creation and management by instructors, student interaction with assigned exercises, and automatic evaluation of submitted solutions. This approach promotes active learning and facilitates student progress tracking.

From a technical perspective, the system follows a client-server architecture, using modern web technologies such as React for the frontend and Node.js for the backend. This ensures accessibility from different devices without requiring installation.

The results demonstrate that the application meets the initial objectives, providing a functional, intuitive, and education-oriented solution. Finally, several future improvements are identified, including support for alternating current circuits, the addition of new components, and further enhancements to the educational platform.

Palabras clave

Análisis de circuitos, corriente continua, aplicación web, educación, e-learning, simulación de circuitos, React, Node.js, Análisis Nodal Modificado.

Keywords

Circuit analysis, direct current, web application, education, e-learning, circuit simulation, React, Node.js, Modified Nodal Analysis.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción, objetivos y estructura

1.1 Introducción

El presente trabajo consiste en el desarrollo de un sistema web de apoyo a la docencia del análisis de circuitos eléctricos en corriente continua. El estudio de circuitos constituye una materia fundamental en niveles educativos preuniversitarios, tales como el bachillerato tecnológico y los ciclos formativos de la familia profesional de electricidad y electrónica, y resulta asimismo una base formativa esencial en numerosas titulaciones de ingeniería.

En determinados entornos educativos, el aprendizaje del análisis de circuitos se apoya en herramientas digitales en las que el profesorado proporciona enunciados en forma de imágenes, mientras que el alumnado debe completar formularios con los resultados obtenidos. Sin embargo, este enfoque presenta diversas limitaciones, como la ausencia de retroalimentación inmediata para el estudiante y la necesidad de emplear herramientas externas para el diseño de los circuitos por parte del profesorado.

En este contexto, se plantea el desarrollo de una herramienta accesible y de fácil uso que facilite tanto la creación de circuitos por parte del profesorado como su resolución por parte del alumnado. A diferencia de las herramientas profesionales existentes, que presentan una elevada curva de aprendizaje, el sistema propuesto prioriza la simplicidad y la usabilidad, adaptándose a las necesidades del entorno educativo.

El sistema desarrollado permite al profesorado definir circuitos personalizados de manera rápida y sencilla, mientras que el alumnado puede interactuar con ellos y obtener retroalimentación inmediata sobre sus resultados, favoreciendo así un aprendizaje más activo y eficaz. El alcance del trabajo se centra en circuitos de corriente

continua que incluyen resistencias y fuentes independientes de tensión y corriente, constituyendo una base adecuada para la formación inicial en análisis de circuitos.

Esta aplicación está por tanto especialmente orientada a su utilización en entornos educativos preuniversitarios, y como recurso de apoyo en cursos introductorios o cursos cero de titulaciones de ingeniería, contribuyendo a reforzar los conocimientos previos del alumnado.

1.2 Objetivos

Las motivaciones y necesidades que impulsan el desarrollo de la herramienta que se presenta en este trabajo, establecen los objetivos del sistema. Estos objetivos, tanto generales como específicos, deben guiar la planificación de la aplicación y garantizar la satisfacción de los requisitos de los usuarios finales.

1.2.1 Objetivo general

El objetivo general del trabajo, en coherencia con la motivación expuesta en la Sección 1.1, es el diseño y desarrollo de una herramienta web interactiva para el aprendizaje del análisis de circuitos en corriente continua. La aplicación contará con perfiles diferenciados para profesorado y alumnado, con el fin de facilitar la enseñanza, el seguimiento del progreso del estudiante y fomentar el aprendizaje autónomo.

1.2.2 Objetivos específicos

Para alcanzar el objetivo general, se han establecido los siguientes objetivos específicos:

Desarrollo del sistema y funcionalidades principales

- Diseñar y desarrollar una interfaz web interactiva que permita al profesorado y al alumnado interactuar con el sistema de forma intuitiva.
- Implementar un sistema que permita al profesorado la creación y gestión de circuitos eléctricos de corriente continua compuestos por resistencias y fuentes de tensión y corriente.
- Implementar la lógica necesaria para la resolución de circuitos eléctricos en corriente continua, obteniendo las magnitudes eléctricas relevantes (intensidad y tensión) de manera automática.

- Permitir la validación de las soluciones introducidas por el alumnado y la generación de retroalimentación automática.

Plataforma educativa y gestión de usuarios

- Desarrollar una plataforma orientada al ámbito educativo que integre la creación de ejercicios por parte del profesorado y su resolución por parte del alumnado.
- Implementar un sistema de gestión de usuarios basado en roles diferenciados (profesor y alumno).
- Permitir el almacenamiento y recuperación de circuitos, usuarios y resultados mediante un sistema de persistencia de datos.
- Facilitar el seguimiento del progreso del alumnado por parte del profesorado.

Accesibilidad y usabilidad

- Garantizar que la herramienta sea accesible desde distintos dispositivos sin necesidad de instalación, al tratarse de una aplicación web.
- Diseñar una interfaz centrada en la usabilidad, reduciendo la curva de aprendizaje de la aplicación, priorizando de esta manera la simplicidad de uso.
- Favorecer que el alumnado pueda centrarse en el aprendizaje de los conceptos de análisis de circuitos, minimizando la complejidad de la herramienta.

1.3 Estructura de la memoria

Este documento se organiza en los siguientes capítulos:

- **Capítulo 1: Introducción, objetivos y estructura.** En este capítulo se presenta el proyecto, su motivación, los objetivos planteados y la organización del documento.
- **Capítulo 2: Estado del arte.** Se analizan las aplicaciones existentes relacionadas con la resolución de circuitos eléctricos y las herramientas educativas en este u otros ámbitos.
- **Capítulo 3: Fundamentos teóricos de circuitos eléctricos.** Se revisan los conceptos y leyes básicas del análisis de circuitos en corriente continua necesarios para el desarrollo del trabajo. Asimismo, se describe el método de

análisis nodal modificado empleado por la aplicación para la resolución de los circuitos.

- **Capítulo 4: Análisis de requisitos.** Se especifican los requisitos funcionales y no funcionales del sistema, además de los casos de uso para los perfiles de alumno y profesor.
- **Capítulo 5: Tecnologías y metodologías de desarrollo.** Se describen las tecnologías utilizadas, incluyendo el entorno de programación, los lenguajes empleados, el sistema gestor de bases de datos y el entorno de despliegue.
- **Capítulo 6: Diseño del sistema.** Se detalla el diseño del sistema, la arquitectura cliente-servidor, la API REST y el diseño de la base de datos.
- **Capítulo 7: Implementación.** Se describen los aspectos más relevantes de la implementación, como los algoritmos desarrollados —con especial atención al módulo de resolución de circuitos— y el diseño de las interfaces de usuario.
- **Capítulo 8: Pruebas y validación.** Se presentan las pruebas realizadas y se analizan los resultados obtenidos para validar el sistema.
- **Capítulo 9: Conclusiones y trabajos futuros.** Se exponen las conclusiones del trabajo y se proponen posibles líneas de mejora y ampliación.
- **Bibliografía**
- **Anexo A: Manual de usuario**
- **Anexo B: Código fuente**
- **Anexo C: Estructura de la base de datos**

Capítulo 2

Estado del arte

2.1 Introducción

En este capítulo se analizan las herramientas y aplicaciones existentes relacionadas con el aprendizaje y la resolución de circuitos eléctricos, con el objetivo de contextualizar el desarrollo del sistema propuesto.

Para ello, se distinguen dos grandes categorías: por un lado, las aplicaciones educativas orientadas a la enseñanza, y por otro, las herramientas de simulación y resolución de circuitos utilizadas en entornos profesionales o formativos.

2.2 Aplicaciones educativas

En el ámbito educativo, el aprendizaje del análisis de circuitos eléctricos se apoya habitualmente en plataformas digitales de gestión del aprendizaje (Learning Management Systems, LMS), que permiten la distribución de contenidos, la realización de actividades y el seguimiento del progreso del alumnado.

Entre estas plataformas destaca Moodle [1] (Fig. 2.1), ampliamente utilizada en instituciones educativas y universidades. Este sistema permite la creación de cursos estructurados, la gestión de usuarios mediante roles (profesor y alumno), y la integración de diferentes tipos de actividades evaluables, como cuestionarios, tareas, foros o recursos multimedia.

Desde el punto de vista funcional, Moodle permite:

- distribución de contenidos teóricos en múltiples formatos
- creación de actividades evaluables
- gestión de calificaciones y seguimiento del alumnado
- control de acceso mediante autenticación de usuarios

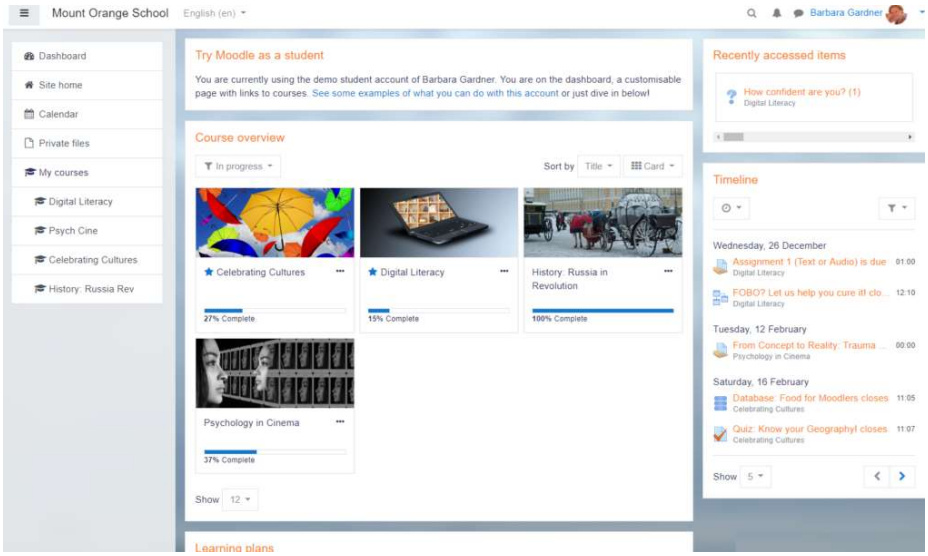


Figura 2.1: Captura de pantalla de Moodle

Sin embargo, en el contexto específico del análisis de circuitos eléctricos, los ejercicios suelen incorporarse como imágenes, diagramas o documentos externos, ya que la plataforma no dispone de herramientas nativas para la representación ni simulación de circuitos eléctricos.

Esto implica que el profesorado debe utilizar herramientas externas para el diseño de los circuitos y posteriormente integrarlos en la plataforma como recursos estáticos, lo que reduce la interactividad del proceso de aprendizaje. Desde el punto de vista del aprendizaje de circuitos eléctricos, esta aproximación presenta una limitación funcional, ya que la interacción con los circuitos no se realiza de forma nativa dentro del sistema, sino mediante recursos externos.

En cuanto a la instalación y despliegue, Moodle requiere un servidor web con soporte PHP y una base de datos relacional (MySQL, PostgreSQL o similar). Se trata de una plataforma de código abierto distribuida bajo licencia GPL, lo que permite su uso gratuito en entornos educativos.

2.3 Herramientas de simulación de circuitos orientados a la enseñanza

Existen diversas herramientas orientadas a la simulación y análisis de circuitos eléctricos, tanto en el ámbito educativo como profesional. Estas aplicaciones permiten

representar circuitos de forma visual y obtener resultados mediante simulación, facilitando la comprensión de los conceptos básicos de electrónica.

Una de las herramientas más conocidas es EveryCircuit [2], una aplicación orientada a la enseñanza de circuitos electrónicos. Permite construir circuitos de forma interactiva mediante una interfaz visual basada en arrastrar y soltar componentes. Su entorno está diseñado para la visualización intuitiva del comportamiento del circuito en tiempo real, mostrando magnitudes como corriente y tensión de forma dinámica en cada componente.

La versión gratuita está limitada a un número reducido de componentes (aproximadamente 5 elementos), mientras que la versión de pago amplía estas funcionalidades y permite la construcción de circuitos de mayor complejidad. La herramienta permite trabajar con circuitos en corriente continua y alterna, incluyendo elementos básicos como resistencias, fuentes y condensadores.

En la Figura 2.2 se muestra una captura de pantalla de esta herramienta.

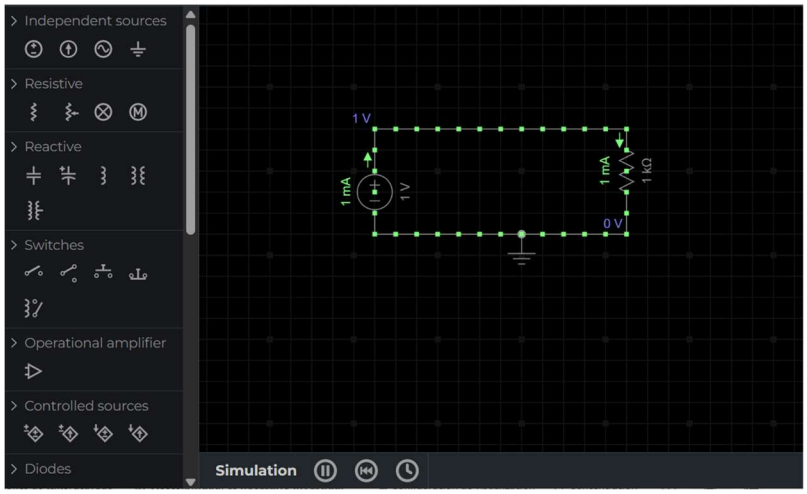


Figura 2.2: Circuito representado y resuelto mediante EveryCircuit

También existen herramientas más sencillas como Tinkercad Circuits [3], una plataforma web que permite simular circuitos electrónicos básicos directamente desde el navegador sin necesidad de instalación. Está orientada a la educación inicial y permite trabajar con componentes básicos como resistencias, LEDs, fuentes de alimentación y microcontroladores.

La Figura 2.3 muestra un ejemplo sencillo de uso con Tinkercad Circuits.

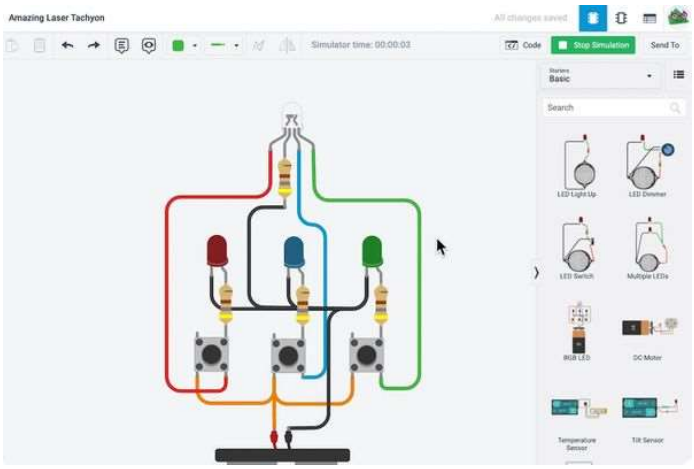


Figura 2.3: Ejemplo de uso de Tinkercad Circuits

Estas aplicaciones destacan por su facilidad de uso y por ofrecer una representación gráfica intuitiva de los circuitos, lo que favorece la comprensión de su funcionamiento. Sin embargo, suelen carecer de funcionalidades orientadas a la gestión docente, como la creación de actividades personalizadas, la evaluación automática o el seguimiento del progreso del alumnado.

2.4 Herramientas profesionales

Por otro lado, existen diversas herramientas de simulación y análisis de circuitos eléctricos ampliamente utilizadas en entornos profesionales y de ingeniería. Estas herramientas están orientadas al diseño, validación y análisis de circuitos electrónicos con un alto nivel de precisión.

Una de las herramientas más utilizadas en este ámbito es LTspice [4] (Fig. 2.4), desarrollada por Analog Devices. Se trata de un simulador de circuitos electrónicos de alto rendimiento que permite realizar análisis en dominio temporal y en frecuencia. Está especialmente orientado a circuitos analógicos complejos y es ampliamente utilizado en ingeniería electrónica.

Su funcionamiento se basa en la definición del circuito mediante netlists o esquemas eléctricos, lo que implica una curva de aprendizaje elevada en comparación con herramientas más visuales. Permite simular circuitos con múltiples tipos de componentes electrónicos y realizar análisis transitorios, de AC y DC.

En la Figura 2.4 se muestra un ejemplo de interfaz de esta herramienta.

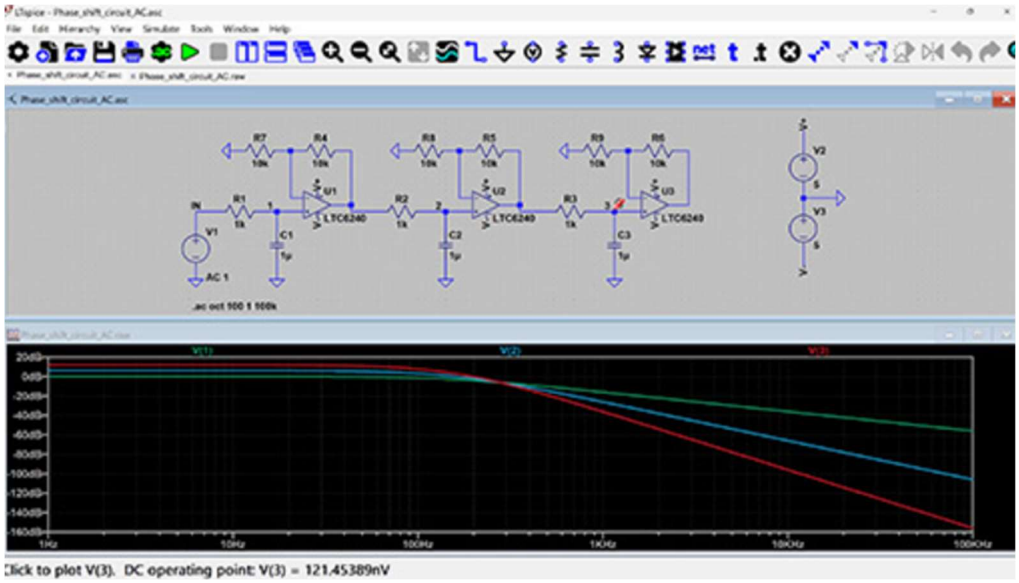


Figura 2.4: Ejemplo de simulador de circuitos LTspice

Asimismo, destaca NI Multisim [5] (Fig. 2.5), una herramienta de simulación desarrollada por National Instruments. Esta aplicación permite el diseño y análisis de circuitos electrónicos mediante una interfaz gráfica orientada a la educación y la ingeniería.

Incluye una amplia biblioteca de componentes electrónicos y permite realizar simulaciones tanto en corriente continua como en corriente alterna, así como análisis de transitorios. Además, incorpora instrumentos virtuales como osciloscopios y multímetros, lo que la acerca a un entorno de laboratorio real.

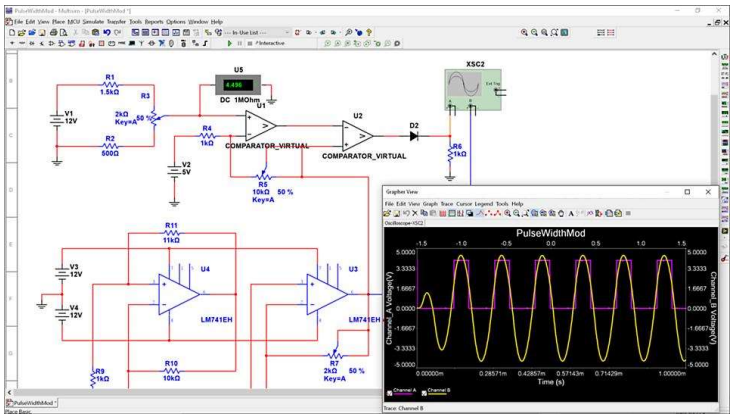


Figura 2.5: Captura de pantalla de NI Multisim

2.5 Comparación de herramientas de simulación de circuitos

La Tabla 2.1 presenta una comparación de las principales herramientas analizadas en este capítulo.

Tabla 2.1: Comparación de herramientas de simulación de circuitos

Herramienta	Tipo	Circuitos soportados	Interfaz	Instalación	Orientación	Funcionalidades educativas
EveryCircuit	Educativa	DC y AC	Visual (drag & drop)	Web / App	Educación	No
LTspice	Profesional	DC, AC, transitorio	Esquemas / netlist	Instalación local	Profesional	No
NI Multisim	Mixta	DC, AC, transitorio	Visual	Instalación local	Educación / ingeniería	Parcial
Tinkercad Circuits	Educativa básica	DC	Web visual	Navegador	Educación inicial	Muy limitada

2.6 Conclusiones

Del análisis realizado en este capítulo se desprende que las herramientas existentes relacionadas con el aprendizaje y el análisis de circuitos eléctricos pueden agruparse en tres grandes categorías: plataformas educativas, herramientas de simulación orientadas a la enseñanza y herramientas profesionales de simulación.

En primer lugar, las plataformas de gestión del aprendizaje como Moodle permiten estructurar el proceso educativo mediante la creación de cursos, la gestión de usuarios y la evaluación de actividades. Estas herramientas resultan adecuadas para la distribución de contenidos y el seguimiento del alumnado. Sin embargo, no incorporan de forma nativa funcionalidades específicas para la representación, simulación o resolución interactiva de circuitos eléctricos, lo que obliga a utilizar herramientas externas y reduce el nivel de integración del proceso de aprendizaje.

En segundo lugar, las herramientas educativas de simulación como EveryCircuit están diseñadas para facilitar la comprensión inicial de los circuitos eléctricos mediante interfaces visuales e interactivas. Estas soluciones permiten la construcción y simulación de circuitos básicos de forma intuitiva y accesible desde el navegador o dispositivos móviles. No obstante, presentan limitaciones en cuanto a la complejidad de los circuitos soportados y carecen de funcionalidades orientadas a la gestión educativa avanzada,

como la creación estructurada de ejercicios, la evaluación automática o el seguimiento del alumnado.

Por último, herramientas profesionales como LTspice o NI Multisim permiten el análisis de circuitos de mayor complejidad, incluyendo simulaciones en corriente continua, corriente alterna y análisis transitorio. Estas aplicaciones están orientadas principalmente a entornos de ingeniería y diseño electrónico, ofreciendo un alto nivel de precisión y capacidad de modelado. Sin embargo, su enfoque no está centrado en la docencia, y su uso requiere un nivel de conocimiento técnico elevado, lo que limita su aplicación en etapas educativas iniciales.

En conjunto, el análisis del estado del arte pone de manifiesto que, si bien podrían existir herramientas con estas características, no se han identificado en la bibliografía ni en las soluciones analizadas aquellas que integren de forma unificada la simulación de circuitos eléctricos con un entorno educativo completo que incluya la gestión de usuarios, la creación de ejercicios por parte del profesorado, la resolución interactiva por parte del alumnado y la evaluación automática de resultados.

En este contexto, el sistema propuesto en este trabajo se plantea como una solución orientada específicamente al ámbito educativo, combinando un entorno de diseño y resolución de circuitos con una plataforma de aprendizaje estructurada.

Capítulo 3

Fundamentos teóricos de circuitos eléctricos

3.1 Introducción

En este capítulo se presentan los fundamentos teóricos necesarios para la comprensión del sistema desarrollado. En particular, se revisan los conceptos básicos del análisis de circuitos eléctricos en corriente continua, así como las leyes y métodos empleados para su resolución, especialmente el método empleado en el algoritmo de resolución implementado.

El objetivo de este capítulo no es realizar un desarrollo exhaustivo de la teoría de circuitos, sino proporcionar el marco conceptual necesario para entender el funcionamiento del sistema.

3.2 Teoría de Circuitos

Un **circuito** es un camino completo formado por conductores y componentes a través del cual circula la corriente eléctrica, constituyendo una estructura que dirige y controla el flujo de energía eléctrica [6].

En este sentido, un circuito eléctrico es un sistema de elementos interconectados en el que la circulación de corriente se produce bajo la acción de una diferencia de potencial. El comportamiento de estos sistemas se rige por principios fundamentales como **la ley de Ohm** y **las leyes de Kirchhoff**, que describen las relaciones entre tensión, corriente y resistencia, así como los balances de corriente y tensión en los nodos y mallas del circuito. En la Figura 3.1 se muestra un ejemplo de circuito eléctrico diseñado y resuelto mediante la aplicación desarrollada.

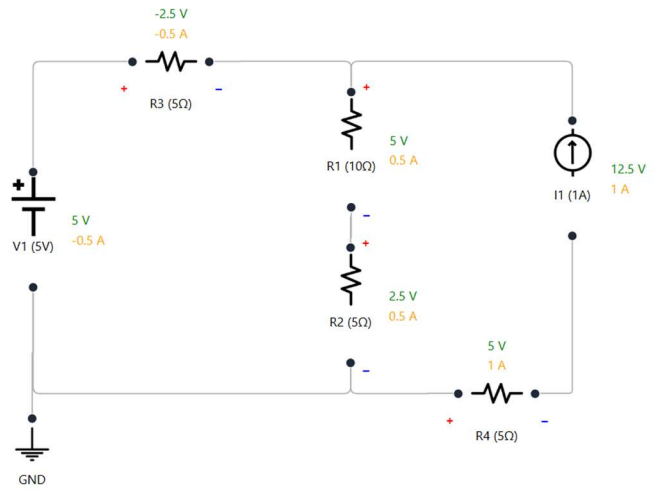


Figura 3.1: Circuito representado y resuelto mediante la aplicación

En este capítulo se describen los componentes eléctricos considerados en el trabajo, las principales magnitudes que intervienen en el análisis de circuitos, las leyes fundamentales que permiten su resolución y el método de resolución empleado.

3.3 Magnitudes eléctricas básicas

En esta sección se presentan las magnitudes eléctricas fundamentales consideradas, necesarias para el análisis de circuitos en corriente continua. Las definiciones expuestas se basan en bibliografía estándar de análisis de circuitos eléctricos [7], así como en diccionarios técnicos especializados [8].

- **Tensión (voltaje):** se define como la diferencia de potencial eléctrico entre dos puntos de un circuito. Representa la energía necesaria por unidad de carga para desplazar una carga eléctrica entre dichos puntos. Su unidad en el Sistema Internacional es el voltio (V). La tensión es la causa del campo eléctrico responsable del movimiento de las cargas.

- **Corriente eléctrica:** se define como el flujo de carga eléctrica que circula a través de un conductor por unidad de tiempo. Matemáticamente, puede expresarse como la variación de carga respecto al tiempo. Su unidad en el Sistema Internacional es el amperio (A). La corriente eléctrica refleja el movimiento ordenado de cargas dentro del circuito.

En el contexto de este trabajo, ambas magnitudes se consideran bajo régimen de corriente continua, por lo que sus valores se asumen constantes en el tiempo. Estas magnitudes están directamente relacionadas entre sí mediante las leyes fundamentales del análisis de circuitos, como la ley de Ohm y las leyes de Kirchhoff, que se introducen en la Sección 3.5.

3.4 Componentes del sistema y representación gráfica

En esta sección se describen los componentes eléctricos considerados en este trabajo, así como su representación dentro de la aplicación desarrollada. Los circuitos eléctricos de la aplicación están compuestos exclusivamente por elementos de corriente continua, en concreto resistencias, fuentes independientes de tensión, fuentes independientes de corriente y un nodo de referencia o tierra.

3.4.1 Resistencia

La **resistencia** es un elemento pasivo de un circuito eléctrico que se opone al paso de la corriente eléctrica, disipando energía en forma de calor. En términos físicos, representa la oposición al flujo de carga eléctrica a través de un conductor.

En la Figura 3.2. se muestra la representación gráfica de una resistencia en la aplicación desarrollada.

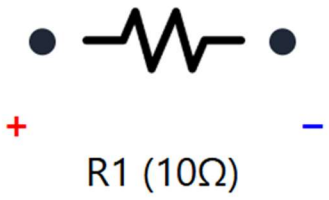


Figura 3.2: Representación gráfica de resistencia en la aplicación

En la aplicación, cada resistencia se modela mediante dos nodos claramente diferenciados, etiquetados como terminal positivo (+) y terminal negativo (-), lo que permite definir un sentido de referencia para la corriente que circula a través del elemento. Esta decisión de diseño resulta fundamental ya que evita ambigüedades en la asignación de signos de las magnitudes eléctricas.

El criterio para la representación del signo del valor de la intensidad en las resistencias es el siguiente: el valor de la intensidad es positivo si la corriente circula del polo positivo al negativo, negativo si circula en sentido contrario.

3.4.2 Fuente de tensión

Una **fuente de tensión** es un dipolo de una red de Kirchhoff cuya tensión es constante o función de otras variables [8]. La fuente de tensión es un elemento activo que en circuitos de corriente continua impone una diferencia de potencial fija entre sus terminales.

En la Figura 3.3. se muestra la representación gráfica de una fuente de tensión en la aplicación desarrollada.

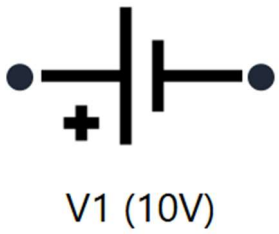


Figura 3.3: Representación gráfica de fuente de tensión en la aplicación

En la representación gráfica, se identifican igualmente los nodos, positivo y negativo, que determinan la polaridad de la fuente. Esta información es esencial para establecer correctamente las ecuaciones del circuito durante el proceso de resolución.

Al contrario de la resistencia, la fuente de tensión entrega energía al circuito por lo que el criterio para la representación del signo de la corriente es justo el contrario: el valor de la intensidad es positivo si la corriente circula del polo negativo al positivo, negativo si circula en el sentido contrario.

3.4.3 Fuente de intensidad

Una **fuente de intensidad** es un dipolo de una red de Kirchhoff cuya intensidad es constante o función de otras variables [8]. La fuente de intensidad es un elemento activo que en circuitos de corriente continua establece un valor de corriente constante entre sus terminales.

En la Figura 3.4. se muestra la representación gráfica de una fuente de tensión en la aplicación desarrollada.

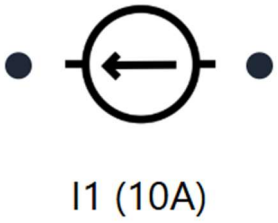


Figura 3.4:Representación gráfica de fuente de corriente en la aplicación

En este caso, la representación gráfica incluye una flecha que define el sentido de la corriente inyectada en el circuito. Al igual que en el caso anterior, al ser también una fuente que proporciona energía al circuito, se considera el siguiente criterio para el valor de la tensión: la tensión es positiva si ésta es mayor en el terminal de salida de la corriente, negativa en el caso contrario

3.4.4 Nodo de referencia

El **nodo de referencia (GND)** se define como el punto del circuito cuyo potencial se toma como referencia, asignándole un valor de 0 voltios. Como se verá posteriormente, este nodo es imprescindible para la aplicación del análisis nodal, ya que permite expresar el resto de las tensiones del circuito con relación a él.

En la Figura 3.5. se muestra la representación gráfica de un nodo de referencia en la aplicación desarrollada.

Figura 3.5: Representación gráfica del nodo de referencia en la aplicación

3.5 Leyes fundamentales del análisis de circuitos

El análisis de circuitos eléctricos se fundamenta en un conjunto de leyes que permiten establecer relaciones entre las magnitudes eléctricas descritas previamente. En particular, la ley de Ohm y las leyes de Kirchhoff constituyen la base para la formulación de los sistemas de ecuaciones que describen el comportamiento de un circuito.

3.5.1 Ley de Ohm

> "La intensidad de la corriente eléctrica que circula por un conductor eléctrico es directamente proporcional a la diferencia de potencial e inversamente proporcional a la resistencia del mismo."- Georg Simon Ohm.

Fuente: [9]

Así pues, la ley de Ohm establece la relación existente entre la tensión, la corriente y la resistencia en un elemento resistivo. Matemáticamente, se expresa como:

$$V = I \cdot R$$

Esta ley permite expresar la corriente que circula por cada resistencia en función de la diferencia de potencial entre sus terminales, lo cual resulta fundamental para el planteamiento del sistema de ecuaciones del circuito.

3.5.2 Ley de corrientes de Kirchhoff

La Ley de corrientes de Kirchhoff (LCK) establece que la suma algebraica de las corrientes que confluyen en un nodo es igual a cero:

$$\sum I = 0$$

La Figura 3.6 [10] ilustra esta ley, donde la corriente total que entra en el nodo es igual a la que sale del mismo, cumpliéndose la relación: $i_1 + i_4 = i_2 + i_3$

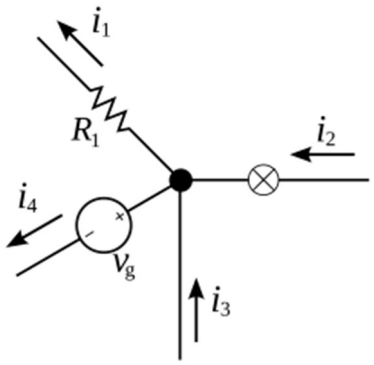


Figura 3.6: Ley de corrientes de Kirchhoff

Esta ley se basa en el principio de conservación de la carga y constituye el fundamento del análisis nodal.

3.5.3 Ley de tensiones de Kirchhoff

La Ley de Tensiones de Kirchhoff (LTK) establece que la suma algebraica de las tensiones en un lazo cerrado es igual a cero:

$$\sum \boldsymbol{V} = \mathbf{0}$$

La Figura 3.7 [10] ilustra esta ley. En el ejemplo representado se cumple que: $v_1 + v_2 + v_3 + v_4 = 0$. No se considera la tensión v_5, ya que no pertenece a la malla analizada.

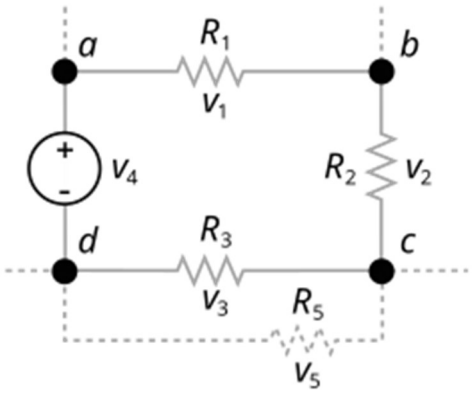


Figura 3.7: Ley de tensiones de Kirchhoff

Esta ley se basa en el principio de conservación de la energía y, aunque no se aplica de forma explícita en el método nodal, resulta implícita en la formulación del sistema de ecuaciones.

3.6 Métodos de análisis de circuitos

A partir de las leyes fundamentales descritas en la sección anterior, es posible desarrollar distintos métodos sistemáticos para el análisis de circuitos eléctricos. Estos métodos permiten obtener las tensiones y corrientes en los diferentes elementos del circuito mediante la formulación y resolución de sistemas de ecuaciones.

Entre los métodos más utilizados en el análisis de circuitos de corriente continua destacan el **método de mallas** y el **método de nodos**, ambos basados en la aplicación directa de las leyes de Kirchhoff.

3.6.1 Método de mallas

El método de mallas se basa en la aplicación de la LTK a los lazos independientes de un circuito. El procedimiento consiste en definir corrientes ficticias que circulan por cada malla y plantear ecuaciones que relacionan las caídas de tensión en los elementos que la componen. A partir de estas ecuaciones se obtiene un sistema lineal cuya resolución permite determinar las corrientes de malla y, posteriormente, calcular las corrientes reales en cada rama del circuito.

No obstante, este método presenta limitaciones importantes en el contexto de este trabajo.

En particular:

- La presencia de fuentes de corriente obliga a introducir ecuaciones adicionales.
- En circuitos con muchas ramas, el número de mallas crece rápidamente.
- Su formulación no es tan directa para su automatización en software.

Por estos motivos, aunque es un método ampliamente utilizado en análisis manual, resulta menos adecuado para su implementación en sistemas computacionales generalistas.

3.6.2 Método de nodos

El método de nodos se basa en la aplicación de la LCK en los nodos del circuito. En este enfoque, las incógnitas principales son las tensiones de los nodos respecto a un nodo de referencia, generalmente denominado tierra o GND.

El procedimiento consiste en:

- Seleccionar un nodo de referencia.
- Asignar una variable de tensión a cada uno de los nodos restantes.
- Aplicar la LCK en cada nodo, expresando las corrientes mediante la ley de Ohm.

De este modo se obtiene un sistema de ecuaciones lineales cuya resolución permite calcular las tensiones nodales. A partir de estas tensiones es posible determinar las corrientes en cada elemento del circuito.

Este método presenta ventajas significativas para su implementación computacional:

- La formulación es sistemática y escalable.
- La matriz resultante suele ser de tamaño reducido.
- Se adapta bien a circuitos con numerosas ramas y fuentes de corriente.

Por estas razones, el método nodal constituye la base de la mayoría de los algoritmos modernos de análisis de circuitos.

3.6.3 Selección del método de análisis

Si bien ambos métodos son válidos para el análisis de circuitos, el **método de nodos** resulta especialmente adecuado para el sistema desarrollado en este trabajo. Su formulación permite representar el circuito directamente en términos de sus nodos eléctricos, lo que facilita la automatización del proceso de resolución mediante técnicas de álgebra lineal.

Además, el método nodal admite extensiones como el **análisis nodal modificado (MNA)**, que permite incorporar de manera natural elementos que imponen relaciones de tensión.

3.7 Análisis nodal modificado

En este trabajo la resolución automática del circuito se basa en el Análisis Nodal, una técnica estándar en el análisis de circuitos lineales ampliamente utilizada por simuladores como SPICE. Este método permite formular de manera sistemática un sistema de ecuaciones lineales que describe el comportamiento eléctrico del circuito, a partir de su topología y de las leyes fundamentales de Kirchhoff [11].

Más concretamente, se emplea una formulación numérica del MNA, orientada al análisis de circuitos lineales resistivos con fuentes independientes de tensión y corriente, utilizando un nodo de referencia o masa. En esta técnica, el conjunto de incógnitas del sistema está formado por:

- Las tensiones nodales, excluyendo el nodo de referencia.
- Las corrientes a través de las fuentes independientes de tensión.

En esta sección se explican los pasos de los que se compone la técnica de Análisis Nodal Modificado, mientras que en la Sección 3.8, se presenta un caso práctico de resolución de un circuito concreto.

3.7.1 Selección del nodo de referencia

Para eliminar la indeterminación asociada al nivel absoluto de tensión, se define un nodo de referencia (GND) cuya tensión se fija a cero. La ecuación correspondiente a dicho nodo se elimina del sistema, reduciendo en una unidad el número de incógnitas de tensión nodal.

A partir de este punto, todas las tensiones nodales se expresan respecto al nodo de referencia, de acuerdo con el planteamiento estándar del análisis nodal.

3.7.2 Modelado del circuito mediante ramas

El circuito se modela como un conjunto de ramas que conectan pares de nodos eléctricos. Cada rama corresponde a un elemento lineal del circuito y se caracteriza por:

- Un nodo positivo y un nodo negativo.
- El tipo de elemento (en nuestro caso: resistencia, fuente de tensión o fuente de corriente).

- Un valor numérico asociado al elemento.
- Una orientación eléctrica, utilizada para definir el signo de las magnitudes calculadas.

Este modelo permite aplicar de forma sistemática las leyes de Kirchhoff y las ecuaciones constitutivas de cada elemento durante la formulación del sistema de ecuaciones.

3.7.3 Formulación del sistema de ecuaciones

El sistema de ecuaciones puede expresarse de forma compacta como:

$$\mathbf{M}\,\mathbf{x} = \mathbf{b}$$

Explicamos a continuación como se construyen los vectores y matrices del sistema de ecuaciones.

Vector de incógnitas (x)

El vector **x** incluye tensiones nodales y corrientes auxiliares. Se construye de la siguiente manera**:**

- Las tensiones nodales de todos los nodos del circuito, excepto el nodo de referencia.
- Las corrientes auxiliares asociadas a las fuentes independientes de tensión.

Si el circuito contiene **N** nodos eléctricos y **M** fuentes independientes de tensión, el vector de incógnitas tiene dimensión **N – 1 + M** y puede expresarse como:

$$\mathbf{x} = [\boldsymbol{v_1} \quad \boldsymbol{v_2} \quad \dots \quad \boldsymbol{v_{N-1}} \quad \boldsymbol{i_{V_1}} \quad \boldsymbol{i_{V_2}} \quad \dots \quad \boldsymbol{i_{V_M}}]^{\mathbf{T}}$$

donde $\boldsymbol{v_k}$representa la tensión del nodo $\boldsymbol{k}$ respecto al nodo de referencia e $\boldsymbol{i_{V_j}}$la corriente que circula por la fuente de tensión $\boldsymbol{j}$.

Matriz del sistema (M)

La matriz **M** recoge las ecuaciones del circuito obtenidas aplicando la KCL en los nodos y las ecuaciones de las fuentes independientes de tensión. Matemáticamente, la matriz **M** puede verse como:

$$\mathbf{M} = \begin{bmatrix} \mathbf{G} & \mathbf{B} \\ \mathbf{C} & \mathbf{0} \end{bmatrix}$$

Donde **G** contiene las contribuciones de las resistencias, y **B** y **C** la conexión entre nodos y fuentes de tensión.

Contribución de las resistencias

Cada resistencia **R** conectada entre dos nodos *a* y *b* aporta términos proporcionales a su **conductancia** $G = 1/R$:

- +G en las posiciones (a, a) y (b, b) de la matriz.
- -G en las posiciones (a, b) y (b, a) de la matriz.

Si uno de los nodos es el nodo de referencia, su fila y columna no se incluyen en la matriz, ajustándose los índices correspondientes.

Contribución de las fuentes independientes de tensión

Para cada fuente de tensión se añade una nueva **fila** a la matriz para imponer la ecuación de la fuente y una nueva **columna** asociada a la corriente que circula por dicha fuente. Si una fuente de tensión conecta los nodos p (positivo) y n (negativo):

- Se introduce +1 en la posición correspondiente al nodo p y a la corriente de la fuente.
- Se introduce -1 en la posición correspondiente al nodo n y a la corriente de la fuente.
- Se introducen los mismos coeficientes +1 y -1 en las columnas de los nodos p y n.

La orientación de la fuente determina cuál de los nodos se considera positivo y cuál negativo, lo que fija el signo de los coeficientes.

Vector de excitaciones independientes (b)

El vector **b** contiene las excitaciones independientes del circuito y se construye de la siguiente forma:

Las **fuentes independientes de corriente** contribuyen a las ecuaciones nodales, añadiendo términos en las posiciones correspondientes a los nodos afectados.

Las **fuentes independientes de tensión** aportan términos constantes en las ecuaciones adicionales introducidas por el Análisis Nodal Modificado.

3.7.4 Resolución del sistema y obtención de magnitudes eléctricas

Una vez formulado el sistema de ecuaciones del Análisis Nodal Modificado, éste se resuelve mediante un método directo para sistemas lineales, obteniéndose los valores numéricos del vector de incógnitas.

- Se reconstruyen las tensiones de todos los nodos del circuito, asignando tensión nula al nodo de referencia y utilizando los valores calculados para el resto.
- Se determinan las corrientes y tensiones en cada uno de los elementos del circuito, empleando las tensiones nodales obtenidas y las leyes constitutivas de cada componente (ley de Ohm para resistencias y definición de las fuentes independientes).

De este modo, se dispone de la información eléctrica completa del circuito, permitiendo analizar su comportamiento y validar las soluciones introducidas por el usuario.

3.8 Ejemplo de resolución del análisis nodal modificado

En esta sección se muestra, a modo de ejemplo, la resolución del circuito eléctrico de la Figura 3.8.

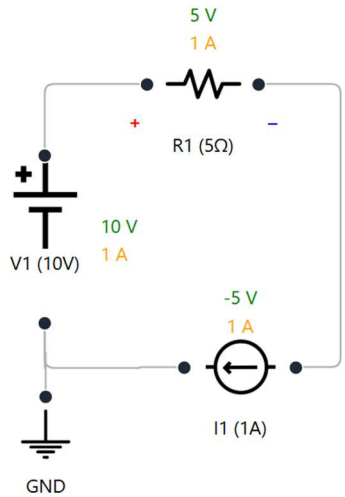


Figura 3.8: Circuito sencillo representado y resuelto mediante la aplicación

Se ha representado y resuelto este circuito como ejemplo por contener todos los componentes considerados en la aplicación: fuente de tensión ($V_1 = 10$ V), entre el nodo de referencia (GND) y el nodo superior izquierdo; resistencia ($R_1 = 5\Omega$) entre el nodo superior izquierdo y el nodo derecho; fuente de corriente ($I_1 = 1$ A), entre GND y el nodo derecho, con indicación de inyección de corriente de derecha a izquierda.

Identificación de los nodos del circuito y construcción del vector de incógnitas:

Mirando la topología del circuito, tenemos 3 nodos: el nodo de referencia (GND), v_1 (terminal positivo de la fuente V1) y v_2 (terminal derecho de la resistencia R2). Igualmente, definimos i_{V_1} la como corriente de la fuente de tensión V1.

De esta manera, el vector de incógnitas resultante es:

$$\mathbf{x} = \begin{bmatrix} v_1 \\ v_2 \\ i_{V_1} \end{bmatrix}$$

Construcción de la submatriz G:

La única resistencia R1, representada entre el nodo 1 y el nodo 2, tiene una conductancia de $G = \frac{1}{5} = 0{,}2\ \mathrm{S}$. Como se ha indicado en la sección anterior, aporta +G en las posiciones (a, a) y (b, b) de la matriz y -G en las posiciones (a, b) y (b, a) de la matriz.

De esta manera, la submatriz G resultante es:

$$G = \begin{bmatrix} +0{,}2 & -0{,}2 \\ -0{,}2 & +0{,}2 \end{bmatrix}$$

Construcción de las submatrices B y C:

Existe una única fuente de tensión, entre GND y el nodo 1, por lo que las submatrices B y C resultantes son:

$$\mathbf{B} = \begin{bmatrix} +1 \\ 0 \end{bmatrix}$$

$$\mathbf{C} = [+1 \quad 0]$$

Matriz M completa

Construimos la matriz **M** completa a partir de las submatrices G, B y C.

$$\mathbf{M} = \begin{bmatrix} 0{,}2 & -0{,}2 & 1 \\ -0{,}2 & 0{,}2 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Vector de excitaciones independientes

La fuente de corriente inyecta corriente de 1 A desde el nodo 2 a GND ($b_2 = -1$), y una fuente de tensión de 10V (añade una ecuación adicional $b_3 = 10$).

De esta manera, el vector b resultante es:

$$\mathbf{b} = \begin{bmatrix} 0 \\ -1 \\ 10 \end{bmatrix}$$

Resolución del sistema

Construimos el sistema de ecuaciones a partir de la matriz M y de los vectores x y b:

$$\begin{bmatrix} 0{,}2 & -0{,}2 & 1 \\ -0{,}2 & 0{,}2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ i_{V_1} \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 10 \end{bmatrix}$$

Resultando: $v_1 = 10$ V, $v_2 = 5$ V e $i_{V_1} = -1$ A

En la aplicación, el signo de la corriente se interpreta tras la resolución del sistema.

3.9 Conclusiones

En este capítulo se han presentado los fundamentos teóricos necesarios para el análisis de circuitos eléctricos en corriente continua, incluyendo las magnitudes básicas, los componentes considerados y las leyes fundamentales que rigen su comportamiento.

Asimismo, se han descrito los principales métodos de análisis de circuitos, justificando la elección del método nodal modificado como base para la resolución automática implementada en el sistema.

Estos conceptos constituyen el marco teórico sobre el que se apoya el desarrollo de la aplicación, permitiendo comprender tanto su funcionamiento interno como la validez de los resultados obtenidos.

Capítulo 4

Análisis de requisitos

4.1 Introducción

En este capítulo se analizan los requisitos que debe cumplir el sistema desarrollado, con el objetivo de definir de forma precisa su comportamiento y las condiciones bajo las que debe operar. Este análisis constituye una base fundamental para el posterior diseño e implementación del sistema.

En primer lugar, se describen los requisitos funcionales, que determinan las funcionalidades que debe ofrecer la aplicación. A continuación, se presentan los requisitos no funcionales, que establecen los criterios de calidad del sistema. Finalmente, se incluye una síntesis de ambos tipos de requisitos en forma de tablas, así como la descripción de los diagramas de casos de uso.

4.2 Requisitos funcionales

Los requisitos funcionales definen el comportamiento del software y las funcionalidades que debe ofrecer al usuario. En el caso de esta aplicación, estos requisitos están directamente relacionados con la creación, resolución y gestión de circuitos eléctricos, así como con la interacción entre los distintos perfiles de usuario. En la Tabla 4.1 se muestran los requisitos funcionales del sistema.

Tabla 4.1: Requisitos funcionales del sistema

Código	Descripción
Diseño de circuitos	
[RF-DIS-01]	Debe disponer de un área de edición de circuitos.
[RF-DIS-02]	Debe permitir la inserción de componentes eléctricos en el circuito.
[RF-DIS-03]	Los componentes deben poder conectarse mediante conexiones.
[RF-DIS-04]	Debe ser posible mover, rotar y eliminar elementos del circuito.
[RF-DIS-05]	Debe mostrar el circuito de forma visual e interactiva.
Resolución y almacenamiento de circuitos	

Código	Descripción
[RF-RES-01]	Debe validar la viabilidad del circuito antes de su resolución.
[RF-RES-02]	Debe calcular tensiones y corrientes en los elementos del circuito.
[RF-RES-03]	Debe mostrar los resultados de forma clara al usuario.
[RF-RES-04]	Debe permitir el almacenamiento de circuitos.
Plataforma educativa	
[RF-EDU-01]	Debe permitir el acceso de usuarios con roles diferenciados.
[RF-EDU-02]	El profesorado debe poder publicar, cerrar y eliminar ejercicios.
[RF-EDU-03]	El alumnado debe poder acceder a ejercicios y resolverlos.
[RF-EDU-04]	Debe permitir la evaluación automática de las respuestas del alumnado.
[RF-EDU-05]	El profesorado debe poder acceder a las respuestas del alumnado.
Persistencia y gestión de datos	
[RF-PER-01]	Debe almacenar la información en una base de datos persistente.
[RF-PER-02]	Debe permitir la recuperación de circuitos y resultados.

4.2.1 Diseño de circuitos

La aplicación debe permitir que el usuario con rol de profesor diseñe circuitos eléctricos mediante la inserción de componentes y su conexión a través de nodos. Asimismo, debe posibilitar la modificación de la posición y orientación de los elementos, así como su eliminación.

4.2.2 Resolución y almacenamiento de circuitos

El sistema debe permitir al usuario con rol de profesor la resolución automática de los circuitos diseñados, así como su almacenamiento. Para ello, la aplicación debe validar la consistencia del circuito y calcular las magnitudes eléctricas relevantes, como tensiones y corrientes.

4.2.3 Plataforma educativa

El sistema tiene un carácter educativo, por lo que debe permitir la interacción entre profesorado y alumnado.

Por un lado, el profesor debe poder gestionar los circuitos diseñados, incluyendo su publicación, cierre y eliminación, así como realizar el seguimiento del progreso de sus alumnos.

Por otro lado, el alumno debe poder acceder a los ejercicios propuestos, resolverlos y enviar sus soluciones, recibiendo una evaluación automática de las mismas.

4.2.4 Persistencia y gestión de datos

El sistema debe garantizar el almacenamiento persistente de la información generada. En concreto, el sistema debe almacenar los circuitos creados por el profesorado, el estado del circuito, las entregas realizadas por el alumnado junto a sus resultados y calificaciones y la información de los usuarios.

4.3 Requisitos no funcionales

Los requisitos no funcionales definen las características de calidad que debe cumplir el sistema, estableciendo cómo debe comportarse, más allá de las funcionalidades que ofrece. En la Tabla 4.2 se muestran los requisitos no funcionales del sistema.

Tabla 4.2: Requisitos no funcionales del sistema

Usabilidad	La aplicación debe disponer de una interfaz sencilla e intuitiva, adecuada para usuarios sin experiencia previa. Debe proporcionar retroalimentación visual de las acciones realizadas.
Rendimiento	El sistema debe responder en tiempos reducidos, permitiendo la edición fluida de circuitos y la resolución en tiempo razonable sin bloqueos.
Portabilidad	La aplicación debe ser accesible desde distintos sistemas operativos (Windows, Linux y macOS) al tratarse de una aplicación web.
Mantenibilidad	El código debe estar estructurado de forma modular y documentado, facilitando su mantenimiento y evolución.
Escalabilidad	El sistema debe permitir la incorporación futura de nuevas funcionalidades, como nuevos tipos de componentes o métodos de análisis.
Seguridad	El sistema debe proteger los datos de los usuarios, garantizando autenticación y control de acceso según roles.

4.3.1 Usabilidad

La aplicación debe presentar una interfaz sencilla, intuitiva y fácil de utilizar, especialmente orientada a estudiantes sin experiencia previa en herramientas de simulación de circuitos eléctricos.

Asimismo, el sistema debe proporcionar retroalimentación visual inmediata ante las acciones del usuario, facilitando la comprensión de las operaciones realizadas y reduciendo la posibilidad de errores.

4.3.2 Rendimiento

El sistema debe ser capaz de responder a las acciones del usuario en tiempos reducidos, garantizando una experiencia de uso fluida.

En particular:

- El diseño del circuito debe realizarse de forma interactiva y sin retardos apreciables.
- La resolución de los circuitos debe ejecutarse en un tiempo razonable para su uso en entornos educativos.
- La interfaz no debe presentar bloqueos ni interrupciones durante su utilización.

4.3.3 Portabilidad

La aplicación debe ser accesible desde distintos sistemas operativos y navegadores web, al tratarse de una aplicación basada en tecnologías web.

En concreto, debe ser compatible con los principales sistemas operativos (Windows, Linux y macOS) y con los navegadores más utilizados, garantizando así su accesibilidad sin necesidad de instalación adicional.

4.3.4 Mantenibilidad

El sistema debe estar diseñado de forma modular para facilitar su mantenimiento y evolución.

Para ello:

- El código debe estar estructurado en módulos independientes.
- Debe mantenerse una correcta documentación del código.
- La arquitectura debe facilitar la incorporación de nuevas funcionalidades.

4.3.5 Seguridad

Dado que el sistema gestiona usuarios y datos educativos, se deben garantizar medidas básicas de seguridad. Esto incluye la protección de datos de usuario, el acceso restringido mediante autenticación y la separación de permisos entre roles.

4.4 Diagramas de casos de uso

En esta sección se identifican los principales casos de uso del sistema en función de los actores que interactúan con la aplicación. Se presentan los diagramas UML de casos de uso junto con una tabla resumen que describe las principales funcionalidades del sistema.

Los actores identificados en el sistema son dos: el profesor y el alumno, cada uno con funcionalidades diferenciadas dentro de la aplicación.

Se presentan dos diagramas UML de casos de uso, uno para cada tipo de actor identificado. Los diagramas de casos de uso permiten representar de forma gráfica la interacción entre los actores y el sistema, mostrando las funcionalidades principales sin entrar en detalles de implementación

4.4.1 Casos de uso del Profesor

La Figura 4.1 muestra el diagrama UML de casos de uso del profesor. La Tabla 4.3 presenta el resumen de los casos de uso del profesor.

Figura 4.1: Diagrama de casos de uso del profesor

Tabla 4.3: Casos de uso del profesor

Código	Caso de uso	Descripción breve
CU-PROF-01	Crear circuito	El profesor diseña un nuevo circuito en el editor.
CU-PROF-02	Resolver circuito	El profesor obtiene la solución automática del circuito.
CU-PROF-03	Guardar ejercicio	El profesor almacena el circuito en el sistema.
CU-PROF-04	Publicar ejercicio	El profesor publica un circuito como ejercicio.
CU-PROF-05	Cerrar ejercicio	El profesor cierra un ejercicio, impidiendo entregas.
CU-PROF-06	Eliminar ejercicio	El profesor elimina un ejercicio en el sistema.
CU-PROF-07	Consultar respuestas	El profesor consulta las respuestas de los alumnos.

4.4.2 Casos de uso del Alumno

La Figura 4.2 muestra el diagrama UML de casos de uso del alumno. La Tabla 4.4 presenta el resumen de los casos de uso del alumno.

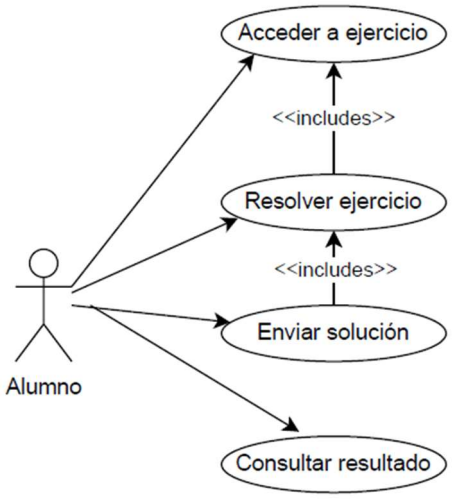


Figura 4.2: Diagrama de casos de uso del Alumno

Tabla 4.4: Casos de uso del alumno

Código	Caso de uso	Descripción breve
CU-ALUM-01	Acceder a ejercicio	El alumno visualiza los ejercicios disponibles y accede.
CU-ALUM-02	Resolver ejercicio	El alumno interactúa con el circuito para resolverlo.
CU-ALUM-03	Enviar solución	El alumno envía su respuesta al sistema.
CU-ALUM-04	Consultar resultado	El alumno visualiza su calificación y resultados.
CU-ALUM-05	Revisar solución	El alumno consulta su solución tras la entrega.

4.5 Conclusiones

El análisis de requisitos permite definir de forma clara las funcionalidades y restricciones del sistema antes de su diseño e implementación. Este proceso resulta fundamental para garantizar que el desarrollo del software se ajuste a los objetivos planteados, asegurando la coherencia entre la solución propuesta y las necesidades del entorno educativo, así como facilitando las fases posteriores de diseño, implementación y validación.

Capítulo 5

Tecnologías y metodologías de desarrollo

5.1 Introducción

En este capítulo se describen las tecnologías, herramientas y metodologías empleadas en el desarrollo del sistema. La selección de dichas tecnologías se ha realizado atendiendo a criterios como la facilidad de desarrollo, la eficiencia y la adecuada integración entre los distintos componentes de la aplicación. Esta elección resulta clave para garantizar un desarrollo sólido, mantenible y alineado con los objetivos del proyecto.

A continuación, se detallan los lenguajes de programación utilizados, las tecnologías del frontend y backend, el sistema de persistencia de datos y los entornos de desarrollo y despliegue. Asimismo, se describe la metodología seguida durante la implementación del proyecto, basada en un enfoque incremental e iterativo que ha permitido la evolución progresiva del sistema.

5.2 Lenguajes de programación

El desarrollo del sistema se ha realizado empleando principalmente tecnologías del ecosistema web.

5.2.1 Tecnologías del frontend

El **frontend** de la aplicación se ha desarrollado como una aplicación web interactiva, basada en una arquitectura de componentes.

5.2.1.1 TypeScript

Se ha empleado **TypeScript** [12] para el desarrollo de la lógica más compleja del sistema, el módulo de resolución de circuitos.

TypeScript es un superconjunto de JavaScript que añade tipado estático, lo que permite definir de forma explícita la estructura de los datos utilizados en la aplicación. En este proyecto se han definido tipos específicos para representar los distintos elementos del circuito, como nodos, conexiones y ramas, lo que aporta diversas ventajas:

- Mayor seguridad en el código, al detectar errores en tiempo de desarrollo.
- Mejor comprensión de las estructuras de datos utilizadas.
- Facilidad de mantenimiento y escalabilidad del sistema.
- Reducción de errores lógicos en la manipulación de datos complejos.

5.2.1.2 React

Se ha utilizado **React** [13] como biblioteca principal para la construcción de la interfaz de usuario, empleándose principalmente en la construcción de los dashboard de alumno y profesor.

React permite el desarrollo de interfaces basado en componentes reutilizables, la así como la gestión del estado de la aplicación y el renderizado dinámico en función de los datos. Estas características resultan especialmente adecuadas para aplicaciones web interactivas como la desarrollada en este proyecto.

5.2.1.3 React Flow

Para la representación visual de los circuitos eléctricos se ha utilizado **React Flow** [14], una librería externa integrada en React, especializada en la creación de interfaces basadas en grafos compuestos por nodos y conexiones.

Esta herramienta resulta especialmente adecuada para la representación visual de circuitos eléctricos, ya que permite modelar de forma natural elementos como componentes y conexiones eléctricas dentro del editor.

Gracias a esta librería, el usuario puede: añadir y eliminar componentes eléctricos, conectarlos mediante enlaces y manipular su posición y orientación.

5.2.1.4 React Router

Para la gestión de la navegación entre las distintas vistas de la aplicación se ha utilizado **React Router** [15], una librería externa que se integra con React y permite implementar enrutamiento en aplicaciones de una sola página (SPA).

Esta herramienta permite asociar rutas específicas a componentes concretos, facilitando la navegación entre vistas como el panel del alumno, el panel del profesor o el editor de circuitos, sin necesidad de recargar la página.

5.2.1.5 Tailwind CSS

El diseño de la interfaz se ha realizado utilizando **Tailwind CSS** [16], un framework basado en clases utilitarias que permite la creación de interfaces limpias y consistentes. Se ha empleado principalmente en el diseño y estilizado de tablas, botones, paneles y modales, contribuyendo a una interfaz visual coherente y adaptada a distintos dispositivos.

5.2.2 Tecnologías del backend

El **backend** del sistema se encarga de gestionar la lógica de negocio de la aplicación, así como el acceso y la persistencia de los datos.

5.2.2.1 Node.js y Express

El backend se ha desarrollado utilizando **Node.js** [17] junto con el framework Express [18], que facilita la creación de servidores web de forma sencilla.

Express permite:

- Definir rutas de acceso a los recursos
- Gestionar peticiónes HTTP (GET, POST, PUT, DELETE).
- Implementación de forma estructurada una API REST.

En el sistema desarrollado, estas tecnologías se utilizan para gestionar aspectos fundamentales de la aplicación, como la administración de circuitos, la gestión de usuarios, el registro de entregas del alumnado y la publicación y cierre de ejercicios.

Aunque estos elementos se describen con mayor detalle en el Capítulo 6, dedicado al diseño del sistema, su implementación se basa en la definición de distintos endpoints que permiten la comunicación entre el cliente y el servidor.

5.2.3 API REST

La comunicación entre frontend y backend se realiza mediante una **API REST**, que actúa como interfaz entre ambos componentes.

Este tipo de arquitectura se caracteriza por:

- El uso de métodos HTTP estándar para realizar operaciones sobre los recursos.
- El intercambio de datos en formato estructurado, en este caso JSON.
- La separación clara entre cliente y servidor, lo que mejora la modularidad del sistema.

5.3 Sistema de persistencia

El sistema de persistencia se encarga del almacenamiento y gestión de los datos de la aplicación, permitiendo su recuperación y manipulación de forma eficiente.

5.3.1 PostgreSQL

Se ha utilizado **PostgreSQL** [19] como sistema gestor de bases de datos.

La elección viene motivada tanto por sus características técnicas como por su integración con la plataforma de despliegue utilizada (Render) que ofrece soporte nativo para este tipo de bases de datos.

Entre las ventajas técnicas de PostgreSQL que hacen especialmente adecuado para este trabajo destacan:

- Alta fiabilidad y robustez
- Soporte para datos estructurados y almacenamiento en formato JSON.
- Buen rendimiento en aplicaciones web.

5.3.2 Almacenamiento de circuitos

Los circuitos se almacenan en la base de datos en formato **JSON**, incluyendo los nodos y componentes eléctricos, las conexiones entre ellos, los resultados generados por el módulo de resolución de circuitos, la calificación y respuestas de los alumnos.

Este enfoque permite almacenar estructuras complejas de forma flexible sin necesidad de un modelo relacional excesivamente fragmentado. Asimismo, facilita la reconstrucción del circuito en el frontend y su posterior procesamiento.

5.4 Entorno de desarrollo

El desarrollo del proyecto se ha realizado en un entorno basado en el editor de código **Visual Studio Code** [20] sobre el sistema operativo Ubuntu Linux.

El uso de un entorno basado en Linux facilita la compatibilidad con herramientas de desarrollo web y entornos de despliegue, así como la ejecución de aplicaciones basadas en Node.js.

Por su parte, Visual Studio Code proporciona diversas funcionalidades que han resultado especialmente útiles durante el desarrollo del proyecto, entre las que destacan:

- Integración nativa con TypeScript.
- Amplio ecosistema de extensiones orientadas al desarrollo web.
- Herramientas de depuración que permiten analizar el comportamiento de la aplicación.

5.5 Entorno de despliegue

La aplicación ha sido desplegada utilizando servicios en la nube, en concreto **Netlify** [21] para el frontend y **Render** [22] para el backend y la base de datos. La aplicación se ha desplegado y se encuentra accesible en la dirección http://iatutxa.netlify.app .

5.5.1 Netlify

Netlify es una plataforma de desarrollo web y alojamiento en la nube diseñada para aplicaciones frontend modernas.

Permite desplegar sitios de forma rápida, automatizando la construcción y publicación de páginas web. Además, ofrece distribución de contenido mediante redes de distribución (CDN), lo que mejora el rendimiento y la accesibilidad de la aplicación.

En este proyecto se ha utilizado Netlify para el despliegue del frontend debido a:

- Su facilidad de integración con aplicaciones desarrolladas en React.
- La automatización del proceso de despliegue.
- Su disponibilidad en modalidad gratuita

5.5.2 Render

Render es una plataforma en la nube de tipo PaaS (Platform as a Service) que permite el despliegue del backend y la base de datos ya que proporciona:

- Soporte para la ejecución de aplicaciones desarrolladas en Node.js.
- Gestión integrada de bases de datos PostgreSQL.
- Configuración sencilla de servicios y entornos.

5.6 Metodología de desarrollo

El desarrollo del proyecto se ha llevado a cabo siguiendo una metodología inspirada en **Scrum**, adaptada a un entorno de trabajo individual.

Aunque no se ha aplicado Scrum de forma estricta, se han adoptado varios de sus principios fundamentales, entre los que destacan:

- División del trabajo en tareas de pequeño tamaño.
- Desarrollo basado en iteraciones.
- Priorización de las funcionalidades principales.
- Evaluación continua del progreso del proyecto.

Scrum es un marco de trabajo ágil orientado al desarrollo incremental e iterativo de software, cuyos principios se recogen en la *Scrum Guide* [23].

5.6.1 Desarrollo incremental

El sistema se ha construido de forma progresiva, desarrollando en primer lugar los componentes fundamentales y ampliando posteriormente sus funcionalidades.

Las principales etapas del desarrollo han sido:

1. Desarrollo del editor visual de circuitos.
2. Implementación del módulo de resolución de circuitos.
3. Creación del panel de profesor.
4. Implementación del panel de alumno.
5. Integración completa del sistema.

Este enfoque ha permitido disponer en cada fase de una versión funcional del sistema, facilitando la validación progresiva de los distintos componentes.

5.6.2 Desarrollo iterativo

Cada una de las fases anteriores ha sido refinada mediante un proceso iterativo, basado en la mejora continua del sistema.

Este proceso ha incluido:

- Realización de pruebas continuas.
- Corrección de errores detectados.
- Mejora progresiva de la interfaz de usuario.

El enfoque iterativo ha permitido adaptar el sistema a medida que se avanzaba en su desarrollo, mejorando tanto su funcionalidad como la experiencia de uso.

5.7 Conclusiones

Las tecnologías y metodologías empleadas han permitido desarrollar una aplicación web completa, funcional y adaptable al entorno educativo. La combinación de herramientas modernas del ecosistema web con un enfoque incremental ha facilitado la implementación progresiva del sistema y la integración de sus diferentes componentes.

Capítulo 6

Diseño del sistema

6.1 Introducción

En este capítulo se describe el diseño del sistema desarrollado, partiendo de los requisitos previamente definidos y estableciendo la estructura que permite su correcta implementación. A lo largo del capítulo se detalla la arquitectura del sistema, basada en un modelo cliente-servidor, y se presenta el diseño de la API REST que facilita la comunicación entre el frontend y el backend. Asimismo, se define la estructura de la base de datos encargada de la persistencia de la información.

Por otro lado, se describe el diseño de los principales componentes del sistema, como el editor de circuitos, prestando especial atención a la representación interna de los circuitos y su integración con el módulo de resolución. En conjunto, este capítulo establece las bases técnicas que permiten la implementación del sistema, garantizando su coherencia, mantenibilidad y adecuación al entorno educativo para el que ha sido concebido.

6.2 Visión general del sistema

El sistema desarrollado sigue una arquitectura cliente-servidor orientada a aplicaciones web, en la que se separan claramente las responsabilidades de la interfaz de usuario, la lógica de negocio y la persistencia de datos. Esta separación permite mejorar la mantenibilidad, escalabilidad y claridad estructural del sistema.

La aplicación está diseñada para soportar dos tipos de usuarios con funcionalidades diferenciadas: profesorado y alumnado. El profesorado dispone de herramientas para la creación y gestión de ejercicios, mientras que el alumnado puede acceder a los circuitos propuestos, resolverlos y consultar sus resultados.

La comunicación entre cliente y servidor se realiza mediante una **API REST**, utilizando el formato **JSON** para el intercambio de información.

6.3 Arquitectura del sistema

6.3.1 Arquitectura cliente-servidor

El sistema se estructura en los siguientes componentes principales:

- **Frontend**: aplicación web desarrollada con React, encargada de la interacción con el usuario.
- **Backend**: servidor desarrollado con Node.js y Express, que gestiona la lógica de negocio y el acceso a datos.
- **Base de datos**: sistema gestor relacional encargado de almacenar la información de forma persistente.

Esta arquitectura permite desacoplar la interfaz de usuario de la lógica interna del sistema, facilitando su evolución y mantenimiento.

Tal y como se muestra en la Figura 6.1, el cliente accede a la aplicación a través de un navegador web y se comunica con el frontend mediante el protocolo HTTP/HTTPS. A su vez, el frontend actúa como intermediario entre el cliente y el backend, realizando peticiones a la API REST mediante HTTP y utilizando el formato JSON para el intercambio de información.

Finalmente, el backend procesa dichas peticiones, aplica la lógica de negocio correspondiente y accede a la base de datos mediante consultas SQL para obtener o modificar la información solicitada.

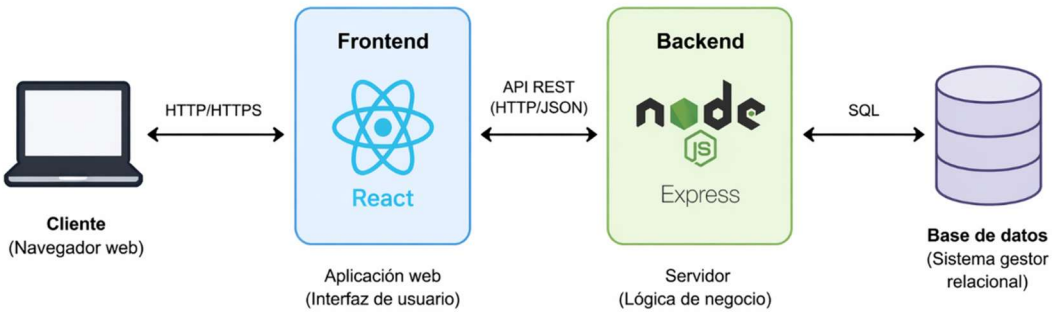


Figura 6.1: Arquitectura cliente-servidor

6.3.2 División en capas

El sistema puede entenderse como una arquitectura de tres capas:

- **Capa de presentación:** implementada en el frontend, gestiona la interacción con el usuario. Incluye los dashboards del profesorado y del alumnado, así como editor y visor de circuitos.
- **Capa de lógica de negocio:** implementada en el backend, gestiona operaciones como la publicación de circuitos y la evaluación de respuestas. Cabe señalar que si bien el cálculo de resultados es parte de la lógica de negocio se realiza en el frontend mediante un módulo específico, con el objetivo de reducir la latencia en la interacción del usuario.
- **Capa de datos:** gestionada mediante una base de datos relacional almacena circuitos, usuarios y entregas.

Esta organización modular facilita la mantenibilidad del sistema y la incorporación de nuevas funcionalidades.

6.3.3 Diseño de la API REST

La comunicación entre el frontend y el backend del sistema se implementa mediante una API REST desarrollada con Node.js y Express. Esta API sigue una organización basada en recursos, dividiéndose en distintos módulos funcionales que agrupan los **endpoints** según su propósito: autenticación de usuarios, gestión de circuitos y gestión de entregas. El intercambio de información se realiza mediante el formato JSON, lo que facilita la integración con el frontend desarrollado en React.

A continuación, se describen los principales endpoints del sistema organizados por módulos.

Autenticación de usuarios

Este módulo gestiona el inicio de sesión y la generación de tokens de autenticación mediante JWT, estándar abierto en JSON propuesto por IETF (RFC7519) [24] para la creación de tokens de acceso que permiten la propagación de identidad y privilegios.

El endpoint presente en el módulo valida las credenciales del usuario mediante comparación de contraseñas cifradas con bcrypt y, en caso de éxito, genera un token JWT con una duración limitada. La Tabla 6.1 muestra el endpoint presente en el módulo.

Tabla 6.1: Endpoint del módulo de autenticación de usuarios

Método	Endpoint	Descripción	Parámetros	Respuesta
POST	/auth/login	Autentica un usuario en el sistema	Nombre y password	Token JWT, rol de usuario, id y nombre

Gestión de circuitos

Este módulo agrupa las operaciones relacionadas con la creación, consulta, publicación y eliminación de circuitos eléctricos. El conjunto de endpoints del módulo permite gestionar todo el ciclo de vida de un circuito, desde su creación por parte del profesorado hasta su resolución y evaluación por el alumnado. La Tabla 6.2 muestra los endpoints agrupados en el módulo de gestión de circuitos.

Tabla 6.2: Enpoints del módulo de gestión de circuitos

Método	Endpoint	Descripción	Parámetros	Respuesta
POST	/circuits	Crea un nuevo circuito en estado borrador	name, nodes, edges, result, profesor_id	id del circuito
GET	/circuits/:id	Obtiene un circuito por su identificador	id del circuito	objeto circuito
GET	/circuits/profesor/:id	Obtiene los circuitos de un profesor	id del profesor	lista de circuitos
DELETE	/circuits/:id	Elimina un circuito	id del circuito	mensaje de confirmación
PUT	/circuits/publish/:id	Publica un circuito y asigna fecha límite	due_date	mensaje de confirmación
PUT	/circuits/finish/:id	Cambia el estado del circuito a cerrado	id del circuito	mensaje de confirmación
GET	/circuits/student/:id	Obtiene los circuitos disponibles para un alumno	id del alumno	lista de circuitos con estado y nota
GET	/circuits/:id/submissions	Obtiene las entregas asociadas a un circuito	id del circuito	soluciones, notas y datos correctos

Gestión de entregas (submissions)

Este módulo gestiona el envío y consulta de las soluciones realizadas por el alumnado. Este módulo permite registrar las soluciones de los estudiantes, actualizar entregas existentes y consultar resultados individuales o agregados. La Tabla 6.3 muestra los endpoints presentes en el módulo de gestión de entregas.

Tabla 6.3: Endpoints del módulo de gestión de entregas

Método	Endpoint	Descripción	Parámetros	Respuesta
POST	/submissions/submit	Crea la entrega de un alumno	student_id, circuit_id, solution, grade	mensaje de confirmación
GET	/submissions/:id/submissions	Obtiene las entregas de un circuito	id del circuito	lista de entregas con notas
GET	/submissions/student/:studentId/circuit/:circuitId	Obtiene la entrega de un alumno en un circuito concreto	studentId, circuitId	solución y nota

6.4 Modelo de datos

El sistema desarrollado requiere un mecanismo de persistencia que permita almacenar de forma estructurada la información relativa a usuarios, circuitos y entregas realizadas por el alumnado. Para ello, se ha diseñado un modelo de datos basado en una base de datos relacional, que garantiza la integridad y coherencia de la información.

El modelo se compone de cuatro entidades principales: **usuarios**, **circuitos**, **entregas** y **relación profesor–alumno**, cuyas relaciones se representan en la Figura 6.2 mediante el correspondiente diagrama entidad–relación.

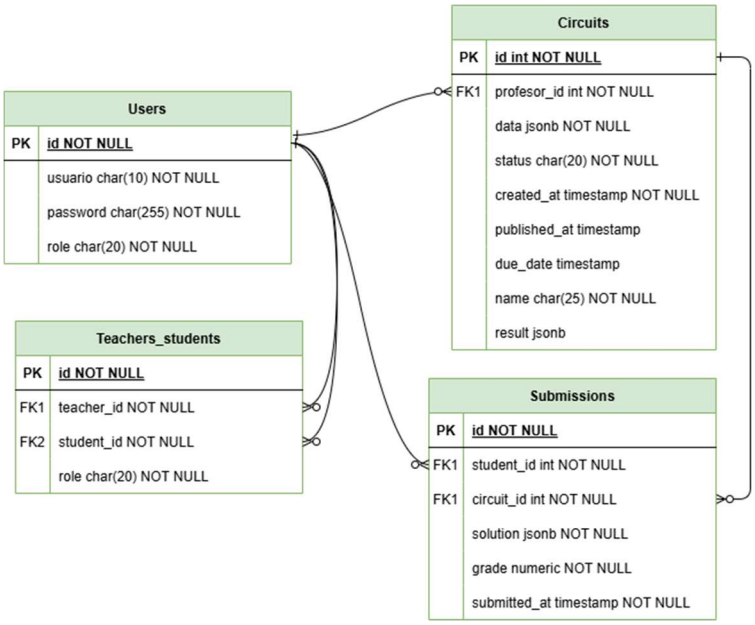


Figura 6.2: Diagrama Entidad-Relación

6.4.1 Usuarios

La entidad de usuarios (users) almacena la información de los usuarios registrados en la aplicación, tanto profesorado como alumnado. Cada usuario dispone de un identificador único, un nombre de usuario, una contraseña cifrada y un rol que determina su tipo dentro del sistema.

El atributo *role* permite diferenciar entre los perfiles de profesor y alumno, lo que facilita la implementación de control de acceso a las distintas funcionalidades de la aplicación.

6.4.2 Circuitos

La entidad de circuitos (circuits) almacena los ejercicios creados por el profesorado. Cada circuito incluye información descriptiva, como su nombre, estado y fechas asociadas, así como la representación interna del circuito y su solución.

La estructura del circuito se almacena en formato JSON mediante el atributo *data*, que contiene los nodos y conexiones que definen el grafo del circuito. Asimismo, el atributo *result* almacena los resultados obtenidos tras la resolución del circuito.

Cada circuito está asociado a un único profesor mediante el atributo *profesor_id*, estableciendo una relación uno a muchos entre usuarios (profesores) y circuitos.

6.4.3 Entregas

La entidad de entregas (submissions) recoge las soluciones enviadas por el alumnado para cada circuito. Incluye la solución propuesta, la calificación obtenida y la fecha de envío.

Cada entrega está asociada tanto a un alumno como a un circuito, mediante los atributos *student_id* y *circuit_id*, respectivamente. Esto establece relaciones de tipo uno a muchos entre usuarios (alumnos) y entregas, así como entre circuitos y entregas.

Cabe destacar que el sistema impone una restricción lógica que garantiza una única entrega por alumno y circuito, gestionada mediante una operación de actualización en caso de reenvío.

6.4.4 Relación profesor–alumno

La entidad intermedia profesor-alumno (teacher_students) permite modelar la relación entre profesorado y alumnado, estableciendo qué estudiantes están asociados a cada profesor. Se trata de una relación de tipo muchos a muchos, ya que un profesor puede tener múltiples alumnos y un alumno puede estar asociado a varios profesores.

6.5 Diseño de la interacción usuario-sistema

El diseño de la interacción usuario–sistema define la forma en que los distintos tipos de usuarios utilizan la aplicación y acceden a sus funcionalidades. El diseño de la interacción se ha basado en los principios de **simplicidad** (reduciendo al mínimo las acciones necesarias en cada tarea), **claridad** (mostrando la información de forma explícita) y **consistencia** (manteniendo una estructura homogénea de las interfaces).

El sistema distingue dos perfiles de usuario principales: profesorado y alumnado, cada uno con un conjunto de funcionalidades adaptadas a sus necesidades. En esta sección se muestra el diagrama de interacción principal de cada perfil.

6.5.1 Interacción del profesorado

El profesorado utiliza el sistema principalmente para la creación y gestión de ejercicios basados en circuitos eléctricos. La interacción se estructura en torno a un panel desde el cual se accede a las distintas funcionalidades.

Con el objetivo de representar de forma visual la interacción del profesorado con el sistema, en la Figura 6.3 se muestra el flujo principal de uso de la aplicación.

El proceso comienza con el inicio de sesión del usuario y el acceso al panel de control. A partir de este punto, el profesorado puede realizar distintas acciones principales: la creación de nuevos ejercicios, la gestión de ejercicios existentes o la consulta de las respuestas enviadas por el alumnado.

En el caso de la creación de ejercicios, el flujo incluye el diseño del circuito mediante el editor, su resolución y su almacenamiento en estado de borrador, permitiendo su posterior publicación. Por otro lado, la gestión de ejercicios permite realizar acciones como publicar, cerrar o eliminar circuitos previamente creados. Finalmente, la consulta de respuestas posibilita la revisión de las entregas realizadas por los estudiantes.

Este flujo refleja el ciclo completo de uso del sistema por parte del profesorado, integrando las principales funcionalidades descritas en el diseño.

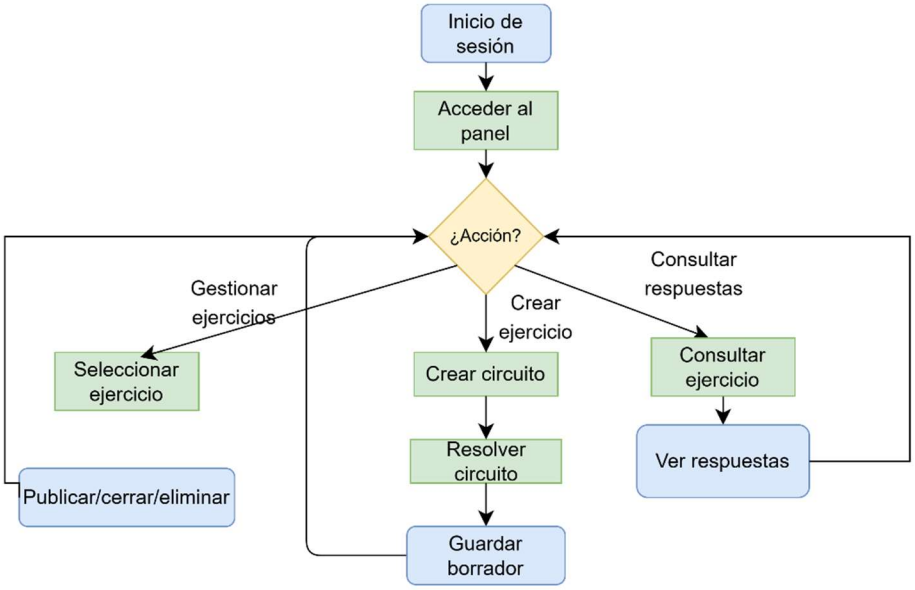


Figura 6.3: Diagrama de interacción del profesorado

6.5.2 Interacción del alumnado

Con el objetivo de representar de forma visual la interacción del alumnado con el sistema, en la Figura 6.4 se muestra el flujo principal de uso de la aplicación.

El proceso comienza con el inicio de sesión del usuario y el acceso al panel de control, donde se muestran los ejercicios disponibles. A partir de este punto, el alumnado puede realizar dos acciones principales: resolver un ejercicio publicado o consultar el resultado de un ejercicio previamente entregado.

En el caso de los ejercicios disponibles, el alumno accede a la interfaz de resolución, donde introduce los valores solicitados y envía su solución. Esta queda registrada en el sistema junto con la calificación obtenida.

Por otro lado, para los ejercicios ya resueltos o cerrados, el alumno puede acceder a su respuesta y consultar la nota obtenida, así como revisar la solución enviada.

Este flujo refleja las principales interacciones del alumnado con el sistema, centradas en la resolución de ejercicios y el seguimiento de su progreso.

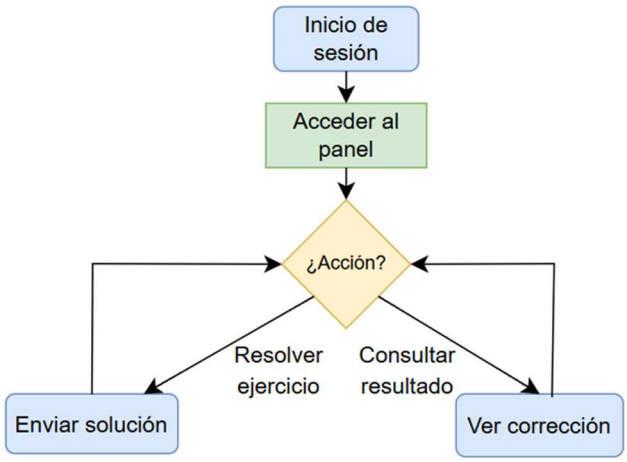


Figura 6.4: Diagrama de interacción del alumnado

6.6 Diseño del editor de circuitos

El editor de circuitos constituye uno de los componentes principales del sistema, ya que permite al usuario la creación y visualización de circuitos eléctricos de forma sencilla. Su diseño está orientado a simplificar la construcción de esquemas eléctricos, evitando la necesidad de herramientas profesionales complejas.

El editor se diseña como una interfaz gráfica interactiva basada en grafos, en la que el circuito eléctrico se representa mediante **nodos**, que representan los componentes

eléctricos (resistencias, fuentes de tensión, fuentes de corriente y nodos de referencia), y **aristas (edges)**, que representan las conexiones eléctricas entre componentes.

Internamente, el circuito se modela mediante dos estructuras principales:

- **Nodes**: array de elementos que representan los componentes eléctricos. Cada elemento contiene la información relevante del componente:
  - Identificador único
  - Tipo de componente
  - Posición en el plano
  - Valor eléctrico (resistencia, tensión o corriente, según el caso)
  - Orientación
- **Edges**: array de conexiones entre componentes. Estas conexiones se almacenan como relaciones entre identificadores, lo que permite reconstruir la topología completa del circuito.

Este modelo simplificado permite mantener el sistema enfocado en el análisis básico de circuitos sin introducir complejidad innecesaria, además de facilitar su serialización en formato JSON para su almacenamiento en base de datos y su posterior reconstrucción.

6.6.1 Interacción del usuario con el editor

El editor proporciona una interfaz visual basada en interacción directa, permitiendo al usuario añadir, conectar, mover y rotar elementos en tiempo real, lo que proporciona una experiencia altamente interactiva.

Los valores eléctricos de los componentes (resistencia, tensión o corriente) se introducen mediante ventanas modales, lo que mejora la usabilidad y evita la manipulación directa sobre el lienzo.

El editor se apoya en la librería **React Flow**, que proporciona la funcionalidad necesaria para la creación y manipulación de grafos interactivos en aplicaciones React. Sobre esta base se ha desarrollado una capa adicional que adapta su comportamiento al dominio específico de los circuitos eléctricos.

6.7 Persistencia del circuito

El circuito se serializa en formato JSON y se almacena en la base de datos. Esta estructura incluye tanto los nodos como las conexiones, lo que permite reconstruir completamente el circuito en cualquier momento.

Este diseño garantiza la persistencia completa del circuito, independiza el frontend del sistema de almacenamiento y permite su reconstrucción y visualización posterior sin pérdida de información.

6.8 Seguridad y autenticación

El sistema implementa un **mecanismo de autenticación** basado en tokens JSON Web Token (JWT), que permite gestionar el acceso de los usuarios de forma segura y eficiente. Este enfoque es ampliamente utilizado en aplicaciones web modernas, ya que facilita la autenticación sin estado y simplifica la comunicación entre cliente y servidor.

El uso de roles permite implementar un **control de acceso** básico, restringiendo determinadas funcionalidades en función del tipo de usuario. De este modo, el profesorado dispone de permisos para la creación y gestión de circuitos, mientras que el alumnado únicamente puede acceder a la resolución de ejercicios y consulta de resultados.

6.9 Conclusiones

En este capítulo se ha presentado el diseño del sistema desarrollado, estableciendo las bases estructurales y funcionales que permiten su correcta implementación.

Se ha definido una arquitectura cliente-servidor que separa claramente las responsabilidades del sistema, favoreciendo su mantenibilidad y escalabilidad. Asimismo, se ha descrito la API REST como mecanismo de comunicación entre el frontend y el backend, permitiendo el intercambio de información de forma estructurada mediante el uso de JSON.

Por otro lado, se ha detallado el diseño de componentes clave como el editor de circuitos, cuya representación basada en grafos permite modelar de forma sencilla los

elementos eléctricos y sus conexiones, facilitando tanto su manipulación como su almacenamiento.

Además, se han descrito los flujos de interacción de los distintos tipos de usuario, evidenciando la diferenciación entre las funcionalidades del profesorado y del alumnado, y poniendo de manifiesto el enfoque orientado al ámbito educativo del sistema.

Finalmente, se han presentado los mecanismos de autenticación y control de acceso, que garantizan un uso adecuado de la aplicación por parte de los distintos perfiles de usuario.

En conjunto, el diseño propuesto proporciona una base sólida y coherente para la implementación del sistema, permitiendo cumplir con los requisitos planteados y facilitando futuras ampliaciones o mejoras.

Capítulo 7

Implementación

7.1 Introducción

En este capítulo se describen los aspectos más relevantes de la implementación del sistema, atendiendo a la arquitectura definida en el capítulo anterior. Se detallan los componentes principales tanto del backend como del frontend, así como la organización del código y los módulos clave de la aplicación.

La implementación se ha llevado a cabo utilizando tecnologías web modernas, siguiendo una arquitectura cliente-servidor. El frontend, desarrollado con React, gestiona la interacción con el usuario, mientras que el backend, basado en Node.js y Express, se encarga de la lógica de negocio y del acceso a la base de datos.

Asimismo, se presta especial atención a los módulos que presentan mayor complejidad técnica, como el editor de circuitos y el sistema de resolución automática, que constituyen los elementos centrales de la aplicación.

7.2 Implementación del backend

En esta sección se presenta la estructura de directorios y los principales ficheros que componen el backend del sistema, junto con algunos fragmentos de código representativos.

El backend se ha desarrollado utilizando Node.js junto con el framework Express, lo que permite la construcción de una API REST ligera, modular y eficiente.

La aplicación se organiza en torno a un directorio principal `routes`, que contiene la implementación de los distintos endpoints de la API REST, agrupados según su funcionalidad. Esta organización mejora la claridad del código y facilita su mantenimiento y ampliación.

El servidor se inicializa en el fichero principal `server.js`, encargado de configurar la aplicación, registrar las rutas disponibles y gestionar las peticiones HTTP procedentes del frontend. Este servidor actúa como intermediario entre la interfaz de usuario y la base de datos, implementando la lógica de negocio asociada a cada operación.

La conexión con la base de datos se realiza mediante un pool de conexiones definido en el fichero `db.js`, lo que permite gestionar múltiples peticiones concurrentes de forma eficiente. En la Figura 7.1, se muestra un fragmento representativo de dicha configuración:

```
import { Pool } from "pg";

export const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  ssl: { rejectUnauthorized: false },
});
```

Figura 7.1: Código de pool de conexiones

La API REST se organiza en distintos módulos en función de la funcionalidad, siguiendo una estructura basada en rutas. Cada módulo agrupa endpoints relacionados con una entidad concreta del sistema:

- **auth.js**: gestión de autenticación de usuarios.
- **circuits.js**: operaciones sobre circuitos.
- **submissions.js**: gestión de entregas realizadas por el alumnado.

Cada endpoint sigue un patrón común de implementación:

1. Definición del método HTTP (GET, POST, PUT o DELETE).
2. Recepción de parámetros mediante la URL o el cuerpo de la petición.
3. Ejecución de la lógica de negocio correspondiente.
4. Acceso a la base de datos mediante consultas SQL.
5. Envío de la respuesta en formato JSON.

A modo de ejemplo, la Figura 7.2 muestra el endpoint encargado de recuperar los circuitos asociados a un profesor:

```
router.get("/profesor/:id", async (req, res) => {
  try {
    const result = await pool.query(
      "SELECT * FROM circuits WHERE profesor_id = $1 ORDER BY created_at DESC",
      [req.params.id]
    );

    res.json(result.rows);

  } catch (err) {
    console.error(err);
    res.status(500).json({ error: "Error obteniendo circuitos" });
  }
});
```

Figura 7.2: Código de enpoint

7.3 Implementación del frontend

En esta sección se presenta la estructura de directorios y los principales ficheros que componen el frontend del sistema, así como algunos de los fragmentos de código más representativos.

El frontend se ha desarrollado utilizando la librería **React**, lo que permite construir una interfaz de usuario dinámica, modular y basada en componentes. Esta aproximación facilita la reutilización de código y la separación de responsabilidades, aspectos clave para la mantenibilidad del sistema.

El proyecto se ha organizado siguiendo un enfoque modular orientado a componentes, que favorece la escalabilidad del sistema y una clara separación de responsabilidades entre los distintos elementos de la aplicación.

7.3.1 Estructura de directorios

El frontend se organiza de forma modular, agrupando los distintos elementos en función de su responsabilidad dentro de la aplicación. La estructura principal se compone de tres directorios:

- **components/**: contiene los componentes reutilizables de la interfaz, como los nodos del circuito. Incluye el subdirectorio *ui/*, que agrupa componentes de interfaz de usuario como ventanas modales, formularios y elementos de notificación.
- **logic/**: alberga la lógica específica de la aplicación, destacando el módulo de resolución de circuitos.

- **pages/**: define las distintas vistas de la aplicación, tales como los paneles de usuario (profesor y alumno), el editor de circuitos y las interfaces de resolución y revisión.

Esta organización permite separar claramente la lógica de presentación, la lógica de negocio en el cliente y la estructura de navegación, mejorando la escalabilidad y el mantenimiento del sistema.

La aplicación se estructura como una *Single Page Application* (SPA), en la que la navegación entre las distintas vistas se gestiona mediante rutas, evitando recargas completas de la página y mejorando la experiencia de usuario.

7.3.2 Gestión de rutas y navegación

La navegación entre las distintas vistas se gestiona mediante la librería **React Router**, que permite definir rutas asociadas a componentes específicos.

Las rutas se definen en el fichero **App.tsx**, ubicado en el directorio raíz del frontend, de forma clara y centralizada. Entre las principales rutas implementadas se encuentran: la página de autenticación (*Login*), los paneles de profesor (*ProfesorDashboard*) y de alumno (*AlumnoDashboard*), el editor de circuitos (*Editor*) y las vistas de resolución (*ResolveCircuit*), revisión (*ReviewCircuit*) y consulta (*ViewCircuit*).

En la Figura 7.3 se muestra un fragmento representativo del fichero App.tsx.

```
export default function App() {
  return (
    <Routes>
      <Route path="/" element={<Login />} />
      <Route path="/profesor" element={<ProfesorDashboard />} />
      <Route path="/alumno" element={<AlumnoDashboard />} />
      <Route path="/editor" element={<Editor />} />
      <Route path="/circuit/:id" element={<ViewCircuit />} />
      <Route path="/resolver/:id" element={<ResolveCircuit />} />
       <Route path="/review/:id" element={<ReviewCircuit />} />
    </Routes>
  );
}
```

Figura 7.3: Código de gestión de rutas

7.3.3 Componentes de interfaz

El frontend se basa en una arquitectura orientada a componentes, donde cada elemento de la interfaz se implementa como una unidad independiente y reutilizable. Este enfoque favorece la modularidad del sistema y reduce la duplicación de código.

Entre los componentes más relevantes destaca **ComponentNode.tsx**, encargado de la representación visual de los componentes eléctricos dentro del editor de circuitos. Asimismo, el directorio **components/ui** agrupa componentes reutilizables de interfaz, como ventanas modales, formularios y elementos de interacción con el usuario.

El fragmento de código mostrado en la Figura 7.4 ilustra la definición de los tipos de componentes eléctricos:

```
const images: Record<string, string> = {
  V: "/images/fuente.png",
  R: "/images/resistencia.png",
  I: "/images/fuenteCorriente.png",
  GND: "/images/ground.png",
};

export type Orientation = "right" | "down" | "left" | "up";

export type ComponentData = {
  label: string;
  type: "R" | "V" | "I"| "GND";
  orientation: Orientation;
  value?: string;
  voltage?: number;
  current?: number;
  solverResult?: {
    voltages: Record<string, number>;
    currents: Record<string, number>;
  };
  showSolverResult?: boolean;
  isConnecting?: boolean;
};
```

Figura 7.4: Código de definición de tipos de componentes

Esta estructura permite añadir nuevos tipos de componente de forma sencilla, favoreciendo la extensibilidad del sistema.

Asimismo, los componentes modales se han diseñado de forma genérica, permitiendo su reutilización en distintos contextos.

La Figura 7.5 muestra la parte más representativa del código de los componentes modales.

```
type ModalProps = {
  open: boolean;
  title?: string;
  children: ReactNode;
  onClose: () => void;
  fullScreen?: boolean;
};

export function Modal({ open, title, children, onClose, fullScreen }: ModalProps) {
  if (!open) return null;

  return (
    <div className="fixed inset-0 z-50 flex items-center justify-center bg-black bg-
opacity-40">
  ....
    </div>
  );
};
```

Figura 7.5: Código de componente modal

7.3.4 Editor de circuitos e interacción gráfica

Uno de los elementos clave del frontend es el editor de circuitos, cuya implementación se basa en la librería **React Flow**, que permite la creación y manipulación de grafos interactivos.

Esta librería facilita la representación de los circuitos eléctricos mediante nodos (componentes) y conexiones (aristas), permitiendo al usuario añadir, conectar, mover y rotar componentes (resistencias, fuentes de tensión, fuentes de corriente y nodo de referencia) de forma intuitiva.

El estado del editor se gestiona mediante los *hooks* (Fig 7.6) proporcionados por React Flow:

```
const [nodes, setNodes, onNodesChange] = useNodesState([]);
const [edges, setEdges, onEdgesChange] = useEdgesState([]);
```

Figura 7.6: Código de gestión de estado del Editor

La creación de conexiones entre componentes se implementa mediante el siguiente callback (Fig. 7.7):

```
const onConnect = useCallback((params) => {
  setEdges((eds) => addEdge({ ...params, type: "smoothstep" }, eds));
}, []);
```

Figura 7.7: Código de creación de componentes

Estas estructuras permiten mantener actualizado el estado del circuito en tiempo real, en función de las acciones del usuario. Además, se ha definido un tipo de nodo personalizado que permite representar los distintos componentes eléctricos.

La rotación de los componentes se implementa modificando su orientación de forma cíclica (Fig. 7.8).

```
const rotateNode = (nodeId) => {
  setNodes((nds) =>
    nds.map((node) => {
      if (node.id !== nodeId) return node;
      const order = ["right", "down", "left", "up"];
      const index = order.indexOf(node.data.orientation);
      return {
        ...node,
        data: { ...node.data, orientation: order[(index + 1) % 4] },
      };
    })
  );
};
```

Figura 7.8: Código de rotación de componentes

7.3.5 Comunicación con el backend

La comunicación entre el frontend y el backend se realiza mediante peticiones HTTP a la API REST descrita en el capítulo anterior. Estas peticiones permiten intercambiar información en formato JSON, facilitando la interoperabilidad entre ambos componentes del sistema.

Desde el frontend, las solicitudes se realizan utilizando la API nativa **fetch**, lo que permite enviar y recibir datos de forma asíncrona. Este enfoque garantiza una experiencia de usuario fluida, evitando bloqueos en la interfaz durante la comunicación con el servidor.

Las operaciones principales que requieren comunicación con el backend incluyen: La **autenticación de usuarios**, mediante el envío de credenciales al endpoint de login; la **gestión de circuitos**, incluyendo la creación, recuperación, publicación y eliminación; el **envío de soluciones**, donde el alumnado remite sus respuestas para su almacenamiento y evaluación; la **consulta de resultados**, tanto por parte del alumnado como del profesorado.

La Figura 7.9 muestra un fragmento representativo de una petición al backend para obtener los circuitos asociados a un profesor.

```
  const fetchCircuits = async () => {
    setLoading(true);
    try {
      const res = await fetch(`${API_URL}/circuits/profesor/${userId}`);
      const data = await res.json();
      setCircuits(data);
    } catch {
    } finally {
      setLoading(false);
    }
  };
```

Figura 7.9: Código de comunicación frontend-backend

En aquellas operaciones que requieren el envío de datos, como la creación de circuitos o el envío de soluciones, se utilizan peticiones de tipo POST o PUT, incluyendo el cuerpo de la solicitud en formato JSON (Fig. 7.10).

```
await fetch(`${API_URL}/submissions/submit`, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        student_id: studentId,
        circuit_id: id,
        solution: answers,
        grade: finalGrade,
      }),
    });
```

Figura 7.10: Código de envío de datos por POST

En conjunto, el diseño de la comunicación entre frontend y backend se basa en principios REST, manteniendo una clara separación de responsabilidades y facilitando la escalabilidad y mantenibilidad del sistema.

7.3.6 Estilado y diseño visual de la interfaz

El diseño visual de la aplicación se ha implementado mediante el uso de **Tailwind CSS**, un framework de estilos basado en clases utilitarias que permite construir interfaces de forma rápida, consistente y mantenible.

El uso de Tailwind permite definir directamente en los componentes las propiedades visuales (espaciado, colores, tipografía o alineación), evitando la necesidad de hojas de estilo separadas y facilitando la coherencia del diseño en toda la aplicación.

En la Figura 7.11 se muestra un fragmento representativo del uso de Tailwind en un componente modal.

```
<div className="fixed inset-0 z-50 flex items-center justify-center bg-black bg-opacity-
40">
  <div className="bg-white rounded-2xl shadow-xl p-6 w-full max-w-md">
    <h2 className="text-xl font-semibold mb-4">{title}</h2>
    {children}
  </div>
</div>
```

Figura 7.11: Diseño con Tailwind

En este ejemplo se puede observar cómo se definen el posicionamiento y layout (fixed, flex, items-center, justify-center), el estilo visual (bg-white, rounded-2xl, shadow-xl) y el espaciado y tipografía (p-6, text-xl, font-semibold)

Este enfoque permite mantener una **consistencia visual** en todos los componentes de la aplicación, ya que se reutilizan patrones de estilos similares.

Además, el uso de Tailwind facilita la creación de **interfaces responsivas**, lo que permite que la aplicación se visualice correctamente en distintos tamaños de pantalla.

En conjunto, el uso de Tailwind CSS contribuye a una implementación eficiente del diseño visual, reduciendo la complejidad del código de estilos y mejorando la mantenibilidad del frontend.

7.4 Implementación del solver de circuitos

En esta sección se describe la implementación del módulo encargado de la resolución automática de circuitos eléctricos. Este módulo constituye el núcleo lógico de la aplicación, ya que permite transformar la representación gráfica del circuito en un sistema de ecuaciones lineales que es resuelto mediante técnicas de álgebra lineal.

La implementación sigue el enfoque de Análisis Nodal Modificado (MNA) descrito en la Sección 3.6, adaptado a un entorno computacional. El funcionamiento general del módulo se puede resumir en las siguientes 6 etapas que se describirán en detalle:

1. Transformación del circuito gráfico en nodos y ramas
2. Identificación de nodos eléctricos mediante Union-Find
3. Selección del nodo de referencia
4. Construcción del sistema MNA
5. Resolución del sistema lineal
6. Reconstrucción de resultados

7.4.1 Transformación del circuito gráfico en nodos y ramas

El solver opera sobre la representación del circuito generada por el editor gráfico, basada en nodos y conexiones. Estas estructuras se definen de la forma mostrada en la Figura 7.12.

```
export type RFNode = {
  id: string;
  data: {
    label: string;
    value?: string;
    type: "R" | "V" | "I" | "GND";
    orientation?: string;
  };
};

export type RFEdge = {
  source: string;
  target: string;
  sourceHandle?: string | null;
  targetHandle?: string | null;
};
```

Figura 7.12: Estructuras de datos del solver

A partir de estas estructuras se construye una representación interna basada en ramas eléctricas (Fig. 7.13). Cada rama conecta dos nodos eléctricos e incluye la información necesaria para la formulación del sistema de ecuaciones.

```
export type Branch = {
  a: number;
  b: number;
  type: "R" | "V" | "I";
  value: number;
  id: string;
  orientation?: string;
};
```

Figura 7.13: Código de definición de rama eléctrica

7.4.2 Identificación de nodos eléctricos

Uno de los pasos fundamentales consiste en agrupar terminales conectados mediante cables en un mismo nodo eléctrico. Para ello se utiliza una estructura de tipo **Union-Find**, que permite identificar conjuntos de nodos conectados de forma eficiente (Fig.7.14).

```
function find(x: string): string {
  if (parent[x] !== x) parent[x] = find(parent[x]);
  return parent[x];
}

function union(a: string, b: string) {
  const pa = find(a);
  const pb = find(b);
  if (pa !== pb) parent[pb] = pa;
}
```

Figura 7.14: Código de identificación de nodos eléctricos

Este proceso permite identificar automáticamente los nodos del circuito, construir un mapeo a índices numéricos y preparar la formulación matricial

7.4.3 Selección del nodo de referencia

Se identifica el nodo de referencia (GND) introducido por el usuario (Fig. 7.15).

```
let gndNode: number | null = null;
nodes.forEach((n) => {
  if (n.data.type === "GND") gndNode = nodeOf(`${n.id}_A`);
});
```

Figura 7.15: Código de identificación del nodo de referencia

Posteriormente, se elimina del sistema lo que permite reducir el número de incógnitas y simplificar el sistema de ecuaciones (Fig. 7.16).

```
function mapNode(n: number) {
  if (n === gndNode) return -1;
  return n < gndNode! ? n : n - 1;
}
```

Figura 7.16: Código de eliminación del nodo de referencia del sistema

7.4.4 Construcción del sistema MNA

El sistema se construye siguiendo la formulación:

$$M \cdot x = b$$

Donde **M** es la matriz del sistema, **x** el vector de incógnitas y **b** (variable **bVec** en el código) el vector de excitaciones.

La Figura 7.17 muestra como se construyen la matriz **M** y el vector **bVec** a partir de las resistencias, fuentes de corriente y de tensión presentes en el circuito.

```
if (br.type === "R") {
  const g = 1 / br.value;
  if (pN >= 0) M[pN][pN] += g;
  if (nN >= 0) M[nN][nN] += g;
  if (pN >= 0 && nN >= 0) {
    M[pN][nN] -= g;
    M[nN][pN] -= g;
  }
}
else if (br.type === "I") {
  if (pN >= 0) bVec[pN] += br.value;
  if (nN >= 0) bVec[nN] -= br.value;
}
else if (br.type === "V") {
  const row = vIdxMap.get(br.id)!;

  if (pN >= 0) {
    M[row][pN] = 1;
    M[pN][row] = -1;
  }
  if (nN >= 0) {
    M[row][nN] = -1;
    M[nN][row] = 1;
  }
```

```
  bVec[row] = br.value;
}
```

Figura 7.17: Código de construcción del sistema lineal

7.4.5 Resolución del sistema lineal

Una vez construido el sistema, se resuelve mediante la librería **mathjs**, utilizando descomposición LU (Fig. 7.18).

```
const xRaw = lusolve(M, bVec) as number[][];
const x = xRaw.map((v) => v[0]);
```

Figura 7.18: Resolución del sistema lineal

7.4.6 Reconstrucción de resultados

A partir del vector solución se reconstruyen las tensiones nodales y se calculan las corrientes y tensiones en cada componente (Fig. 7.19).

```
for (let i = 0; i < nNodes; i++) {
  if (i === gndNode) {
    nodeVoltages[i] = 0;
  } else {
    nodeVoltages[i] = x[xi++];
  }
}
const V_diff = Vpos - Vneg;

if (br.type === "R") {
  currents[br.id] = V_diff / br.value;
}
```

Figura 7.19: Código de construcción de resultados

7.4.7 Consideraciones sobre la orientación

La orientación de los componentes se tiene en cuenta para determinar correctamente la polaridad (Fig. 7.20).

```
function getTerminalsByOrientation(br: Branch) {
  const negIsA = br.orientation === "left" || br.orientation === "up";
  return {
    pos: negIsA ? br.b : br.a,
    neg: negIsA ? br.a : br.b,
  };
}
```

Figura 7.20: Código de corrección de la orientación

7.5 Conclusiones

En este capítulo se ha descrito la implementación del sistema desarrollado, detallando tanto los componentes del backend como del frontend, así como los módulos más relevantes de la aplicación.

Se ha puesto de manifiesto cómo la arquitectura cliente-servidor definida en el diseño se materializa mediante el uso de tecnologías web modernas, permitiendo una separación clara de responsabilidades entre la interfaz de usuario, la lógica de negocio y la persistencia de datos.

Asimismo, se ha presentado la implementación del editor de circuitos y del módulo de resolución automática, evidenciando la viabilidad técnica del sistema y su capacidad para integrar de forma eficiente la interacción gráfica con el análisis de circuitos eléctricos.

En conjunto, la implementación realizada permite cumplir los requisitos establecidos, proporcionando una base funcional sobre la que se apoyan las fases de validación y evaluación del sistema.

Capítulo 8

Pruebas y validación

8.1 Introducción

El desarrollo de software requiere una fase de verificación y validación con el objetivo de garantizar la calidad del producto final. A través de este proceso se pretende detectar posibles errores de ejecución, resultados inesperados, anomalías o cualquier comportamiento que pueda comprometer tanto la funcionalidad como la fiabilidad del sistema.

En entornos profesionales, es habitual complementar estas fases con pruebas en condiciones reales de uso, permitiendo identificar fallos que no han sido detectados durante el desarrollo. En este contexto, la validación adquiere un papel fundamental para asegurar que el sistema cumple con los requisitos definidos y responde adecuadamente a las necesidades para las que ha sido diseñado.

En el caso de este proyecto, la validación se ha llevado a cabo mediante la ejecución de distintas pruebas sobre la aplicación desarrollada, evaluando su comportamiento en escenarios representativos de uso. Estas pruebas permiten comprobar el cumplimiento de los requisitos funcionales y no funcionales definidos en el capítulo 4, así como verificar el correcto funcionamiento global de la plataforma educativa.

8.2 Pruebas funcionales

En esta sección se describen las **pruebas de caja negra** realizadas con el objetivo de verificar el cumplimiento de los requisitos funcionales definidos en el Capítulo 4.

Estas pruebas se centran en evaluar el comportamiento del sistema desde el punto de vista del usuario, comprobando que las funcionalidades implementadas responden correctamente ante distintos escenarios de uso.

8.2.1 Pruebas del diseño de circuitos

Las pruebas del módulo de diseño permiten comprobar que el usuario puede crear y editar circuitos eléctricos de forma interactiva mediante la inserción y conexión de componentes.

Los casos de prueba realizados se recogen en la Tabla 8.1.

Tabla 8.1: Casos de prueba del diseño de circuitos

ID	Descripción	Pasos a realizar	Resultado esperado	Estado
Test-01	Área de edición	Acceder al editor de circuitos	Se muestra correctamente el área de trabajo	OK
Test-02	Inserción de componentes	Añadir un componente de cada tipo al circuito	Los componentes aparecen en el área de edición	OK
Test-03	Conexión de componentes	Conectar varios componentes entre sí	Se crean las conexiones correctamente	OK
Test-04	Movimiento	Desplazar componentes dentro del área de trabajo	Los componentes cambian de posición	OK
Test-05	Rotación	Rotar varios componentes	Los componentes cambian de orientación	OK
Test-06	Eliminación	Eliminar un componente	El componente desaparece del área de trabajo	OK

En las siguientes figuras se muestran las pruebas realizadas, en concreto:

- Figura 8.1: muestra el resultado de realizar el Test-01.
- Figura 8.2: muestra el resultado de realizar el Test-02.
- Figura 8.3: muestra el resultado de realizar el Test-03.
- Figura 8.4: muestra el resultado de realizar el Test-04.
- Figura 8.5: muestra el resultado de realizar el Test-05.
- Figura 8.6: muestra el resultado de realizar el Test-06.

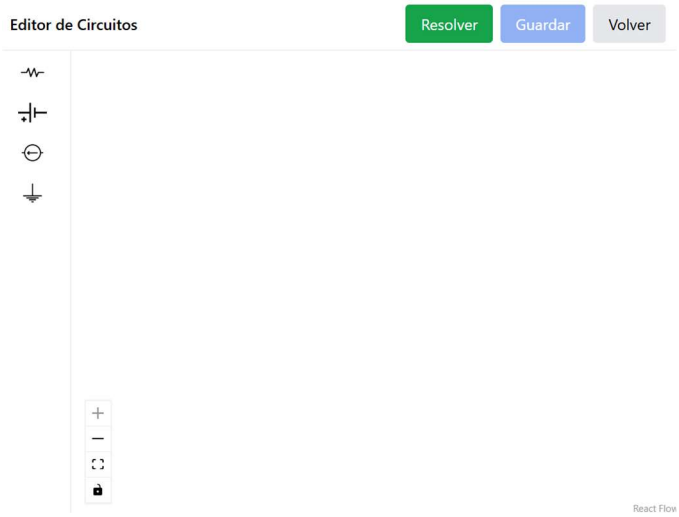


Figura 8.1: Acceso al editor de circuitos

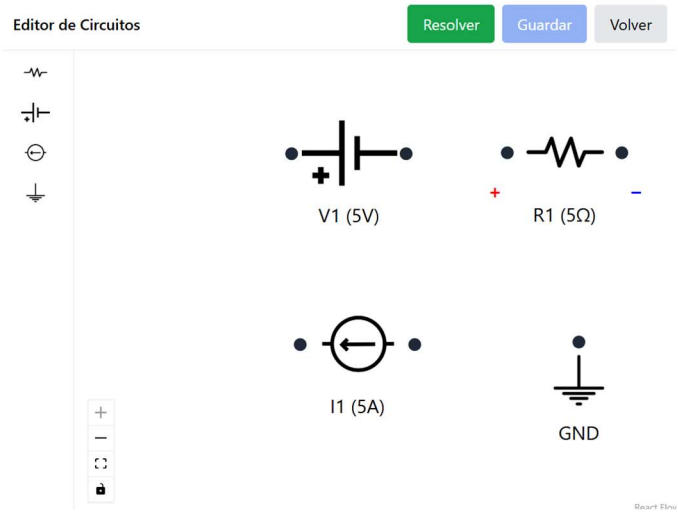


Figura 8.2: Inserción de componentes

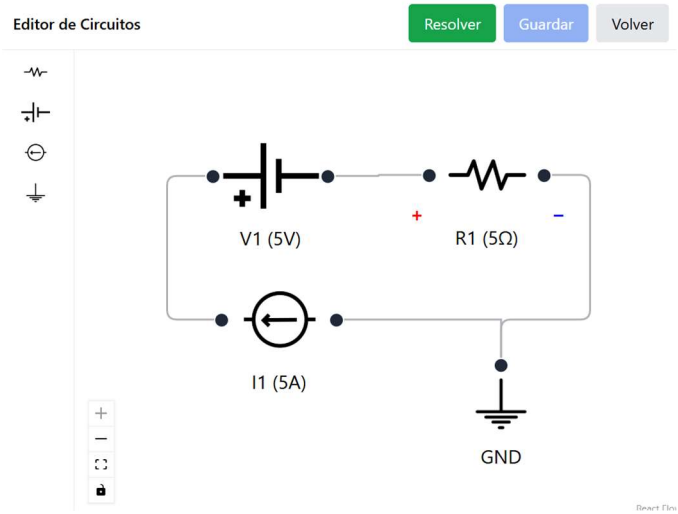


Figura 8.3: Conexión de componentes

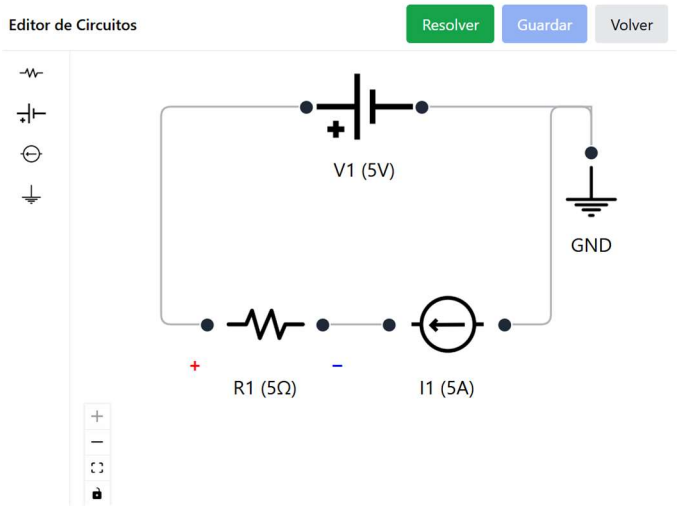


Figura 8.4: Movimiento de componentes

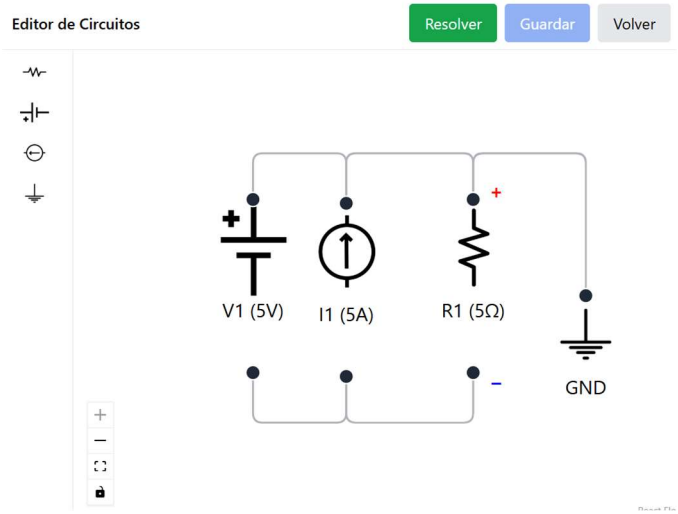


Figura 8.5: Rotación de componentes

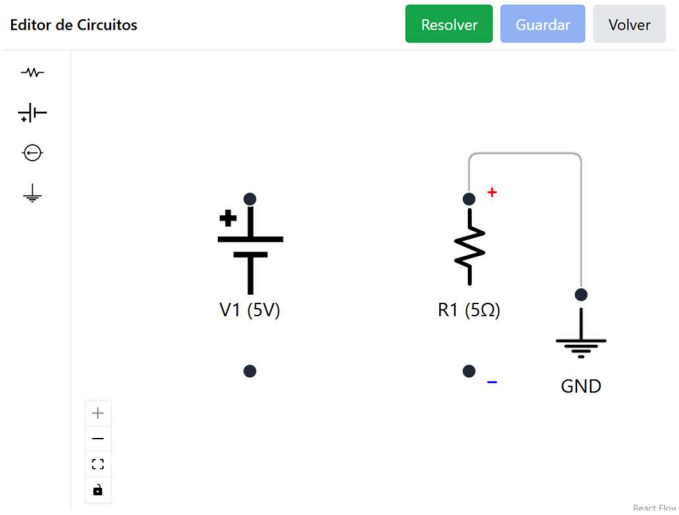


Figura 8.6: Eliminación de componente

8.2.2 Pruebas de resolución de circuitos

Las pruebas del módulo de resolución permiten verificar que el sistema es capaz de validar la viabilidad de los circuitos diseñados, calcular las magnitudes eléctricas correspondientes y mostrar los resultados de forma adecuada al usuario.

Asimismo, se comprueba el correcto almacenamiento de los circuitos una vez han sido resueltos.

Los casos de prueba realizados se recogen en la Tabla 8.2.

Tabla 8.2: casos de prueba de resolución de circuitos

ID	Descripción	Pasos a realizar	Resultado esperado	Estado
Test-07	Validación de circuito	Intentar resolver circuito inválido	Se muestra mensaje de error	OK
Test-08	Cálculo de magnitudes	Resolver circuito válido	Se calculan tensiones y corrientes	OK
Test-09	Visualización de resultados	Resolver un circuito válido	Los resultados se muestran automáticamente en la interfaz	OK
Test-10	Guardado de circuito	Guardar circuito	El circuito aparece en la lista de circuitos	OK
Test-11	Recuperación de circuito	Acceder a un circuito guardado	El circuito se carga correctamente con sus datos	OK

Cabe destacar que el proceso de cálculo y la visualización de resultados se realizan de forma conjunta en una única acción, mostrándose automáticamente los resultados tras la resolución del circuito.

En las siguientes figuras se muestran las pruebas realizadas:

- Figura 8.7: muestra el resultado de realizar el Test-07.
- Figura 8.8: muestra el resultado de realizar el Test-08 y el Test-09.
- Figura 8.9: muestra el resultado de realizar el Test-10.
- Figura 8.10: muestra el resultado de realizar el Test-11.

Figura 8.7: Validación de circuito incorrecto

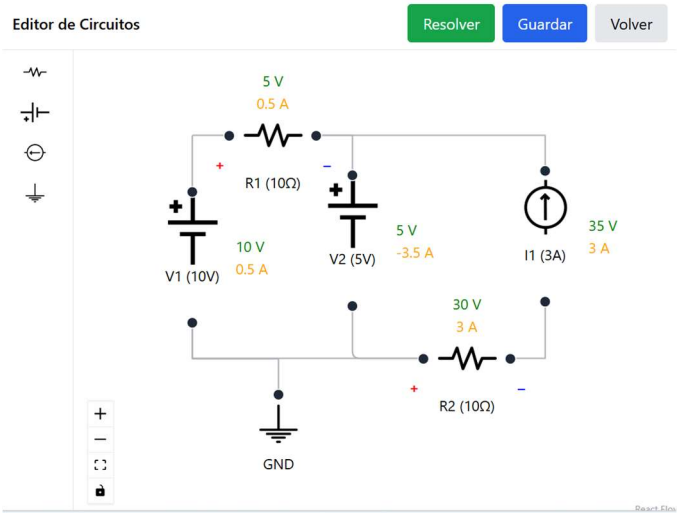


Figura 8.8: Resolución de circuito válido

Figura 8.9: Listado de circuito

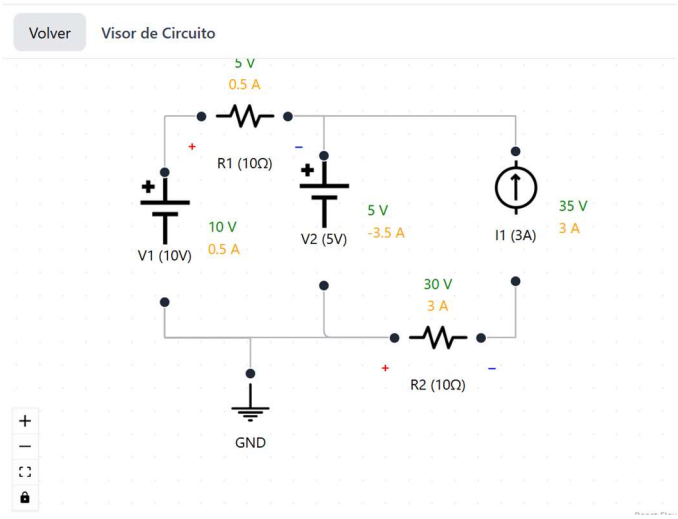


Figura 8.10: Carga de circuito guardado

8.2.3 Pruebas de la plataforma educativa

Las pruebas de la plataforma educativa permiten verificar la correcta interacción entre los distintos tipos de usuario del sistema, así como la gestión de ejercicios y la evaluación de las respuestas del alumnado.

En particular, se valida el funcionamiento de los roles diferenciados (profesor y alumno), la creación y publicación de ejercicios, el acceso por parte del alumnado, así como la evaluación automática de las soluciones y la gestión de las respuestas.

Los casos de prueba realizados se recogen en la Tabla 8.3.

Tabla 8.3: Casos de prueba de la plataforma educativa

ID	Descripción	Pasos a realizar	Resultado esperado	Estado
Test-12	Acceso con rol profesor	Iniciar sesión como profesor	El sistema permite el acceso con el rol de profesor	OK
Test-13	Acceso con rol alumno	Iniciar sesión como alumno	El sistema permite el acceso con el rol de alumno	OK
Test-14	Creación de ejercicios	Acceder como profesor y crear un ejercicio	El ejercicio aparece en el listado de ejercicios en borrador	OK
Test-15	Publicación de ejercicios	Acceder como profesor y publicar el ejercicio	El ejercicio aparece disponible para los alumnos	OK
Test-16	Acceso a ejercicios	Acceder como alumno y consultar ejercicios	Se muestran correctamente el ejercicio publicado	OK
Test-17	Resolución de ejercicios	Acceder como alumno y resolver el ejercicio	El sistema permite interactuar con el circuito y enviar la solución	OK

ID	Descripción	Pasos a realizar	Resultado esperado	Estado
Test-18	Evaluación automática	Enviar la solución del ejercicio	El sistema calcula y muestra la calificación automáticamente	OK
Test-19	Visualización de solución	Acceder como alumno y visualizar el ejercicio resuelto	Se muestra el circuito con los resultados correctos y los enviados	OK
Test-20	Consulta de respuestas	Acceder como profesor y revisar respuestas	El profesor puede visualizar las soluciones enviadas por los alumnos	OK
Test-21	Cierre de ejercicio	Acceder como profesor y cerrar el ejercicio	El ejercicio pasa a estado cerrado y no permite nuevas entregas	OK
Test-22	Eliminación de ejercicio	Acceder como profesor y eliminar el ejercicio	El ejercicio desaparece de todos los listados del sistema	OK

En las siguientes figuras se muestran las pruebas realizadas:

- Figura 8.11: muestra el resultado de realizar el Test-12.
- Figura 8.12: muestra el resultado de realizar el Test-13.
- Figura 8.13: muestra el resultado de realizar el Test-14.
- Figura 8.14: muestra el resultado de realizar el Test-15.
- Figura 8.15: muestra el resultado de realizar el Test-16.
- Figura 8.16: muestra el resultado de realizar el Test-17.
- Figura 8.17: muestra el resultado de realizar el Test-18.
- Figura 8.18: muestra el resultado de realizar el Test-19.
- Figura 8.19: muestra el resultado de realizar el Test-20.
- Figura 8.20: muestra el resultado de realizar el Test-21.
- Figura 8.21: muestra el resultado de realizar el Test-22.

Figura 8.11: Acceso con rol de profesor

Figura 8.12: Acceso con rol de alumno

Figura 8.13: Creación de ejercicios

Figura 8.14: Publicación de ejercicios

Figura 8.15: Acceso a ejercicios

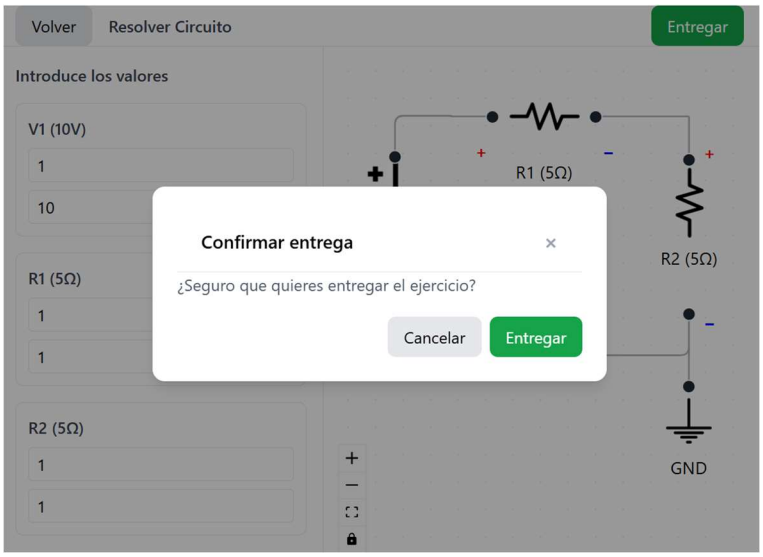


Figura 8.16: Resolución de ejercicios

Figura 8.17: Evaluación automática

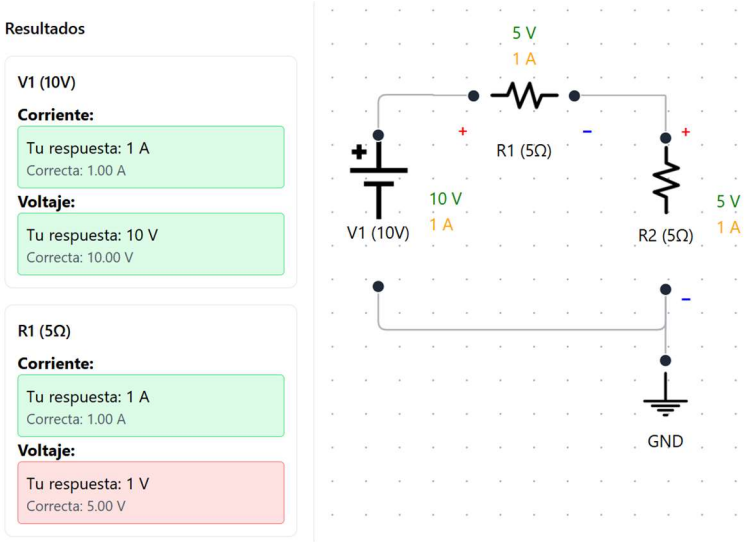


Figura 8.18: Visualización de la solución

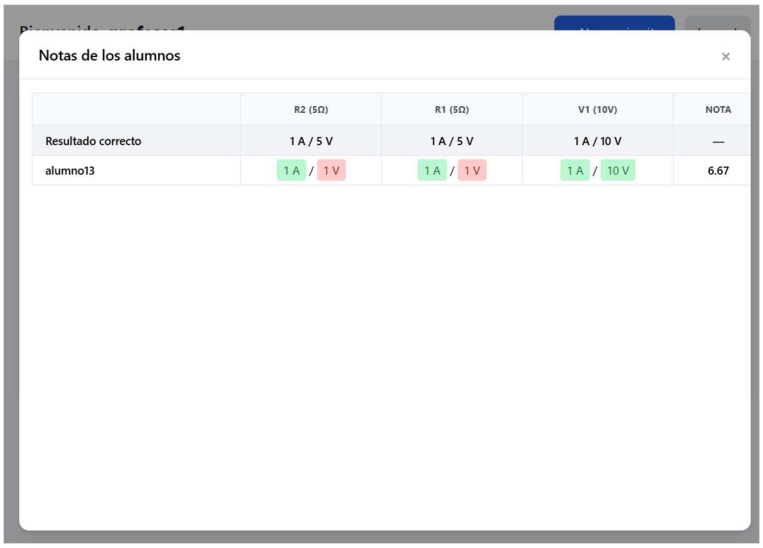


Figura 8.19: Consulta de respuestas

Figura 8.20: Cierre de ejercicio

Figura 8.21: Eliminación de ejercicio

8.2.4 Pruebas de persistencia y gestión de datos

Las pruebas de persistencia y gestión de datos permiten verificar que el sistema almacena de forma permanente la información generada y que esta puede ser recuperada correctamente en distintos momentos de uso.

Cabe destacar que estos aspectos ya han sido validados de forma indirecta en secciones anteriores, como en las pruebas de resolución de circuitos y en la plataforma educativa. No obstante, en esta sección se recogen de forma específica para garantizar el cumplimiento de los requisitos de persistencia definidos.

Los casos de prueba realizados se recogen en la Tabla 8.4.

Tabla 8.4: Casos de prueba de persistencia y gestión de datos

ID	Descripción	Pasos a realizar	Resultado esperado	Estado
Test-25	Almacenamiento de circuitos	Diseñar un circuito y guardarlo	El circuito se almacena correctamente en la base de datos	OK
Test-26	Recuperación de circuitos	Acceder a un circuito previamente guardado	El circuito se restaura con todos sus elementos y conexiones	OK
Test-27	Persistencia de resultados	Resolver un circuito y guardar los resultados	Los resultados se almacenan y pueden consultarse posteriormente	OK

8.3 Matriz de trazabilidad

La matriz de trazabilidad permite establecer la relación entre los requisitos funcionales definidos en el capítulo de análisis y los casos de prueba realizados durante la fase de validación. De este modo, se garantiza que todos los requisitos han sido verificados mediante pruebas específicas, asegurando la cobertura completa del sistema. En la Tabla 8.5 se presenta la correspondencia entre los requisitos funcionales y los casos de prueba asociados.

Tabla 8.5: Matriz de trazabilidad

ID	Descripción	Pruebas asociadas	Estado
Diseño de circuitos			
[RF-DIS-01]	Área de edición de circuitos	Test-01	
[RF-DIS-02]	Inserción de componentes	Test-02	OK
[RF-DIS-03]	Conexión de componentes	Test-03	OK
[RF-DIS-04]	Movimiento, rotación y eliminación	Test-04, Test-05, Test-06	OK
[RF-DIS-05]	Visualización interactiva	Test-01, Test-02, Test-03	OK
Resolución y almacenamiento de circuitos			
[RF-RES-01]	Validación del circuito	Test-07	OK
[RF-RES-02]	Cálculo de tensiones y corrientes	Test-08	OK
[RF-RES-03]	Visualización de resultados	Test-09	OK
[RF-RES-04]	Almacenamiento de circuitos	Test-10, Test-25	OK
Plataforma educativa			
[RF-EDU-01]	Acceso con roles diferenciados	Test-12, Test-13	OK
[RF-EDU-02]	Publicar, cerrar y eliminar ejercicios	Test-14, Test-15, Test-21, Test-22	OK

ID	Descripción	Pruebas asociadas	Estado
[RF-EDU-03]	Acceso y resolución de ejercicios	Test-16, Test-17	OK
[RF-EDU-04]	Evaluación automática	Test-18	OK
[RF-EDU-05]	Consulta de respuestas	Test-20	OK
Persistencia y gestión de datos			
[RF-PER-01]	Almacenamiento en base de datos	Test-25, Test-27	OK
[RF-PER-02]	Recuperación de circuitos y resultados	Test-11, Test-26, Test-27	OK

8.4 Validación de requisitos no funcionales

En esta sección se analiza el grado de cumplimiento de los requisitos no funcionales definidos en la Sección 4.2. A diferencia de los requisitos funcionales, estos no se validan mediante casos de prueba concretos, sino mediante la evaluación del comportamiento general del sistema.

8.4.1 Usabilidad

La aplicación presenta una interfaz intuitiva basada en interacción visual directa. Durante las pruebas funcionales se ha comprobado que:

- El usuario puede diseñar circuitos mediante acciones simples (clic, arrastrar, etc.).
- Las acciones realizadas tienen una respuesta visual inmediata.
- Los errores (por ejemplo, circuitos inválidos) se notifican claramente.

Esto confirma el cumplimiento del requisito de usabilidad.

8.4.2 Rendimiento

Durante las pruebas realizadas:

- El diseño de circuitos se realiza de forma fluida, sin retardos apreciables.
- La resolución de circuitos se ejecuta de forma inmediata para los casos probados.
- No se han detectado bloqueos de la interfaz.

Por tanto, el sistema cumple los requisitos de rendimiento establecidos.

8.4.3 Portabilidad

Al tratarse de una aplicación web desarrollada con tecnologías como React y Node.js, el sistema puede ejecutarse desde distintos navegadores y sistemas operativos sin necesidad de instalación adicional.

Las pruebas realizadas en diferentes entornos confirman su correcto funcionamiento, cumpliendo así el requisito de portabilidad.

8.4.4 Mantenibilidad

El sistema ha sido desarrollado siguiendo una arquitectura modular, separando frontend y backend.

- El código se organiza en componentes reutilizables.
- Se facilita la extensión futura del sistema.

Esto permite concluir que el sistema cumple con los requisitos de mantenibilidad.

8.4.5 Seguridad

El sistema implementa control de acceso mediante autenticación de usuarios y diferenciación de roles (profesor/alumno).

- Los usuarios solo pueden acceder a las funcionalidades correspondientes a su rol.
- Se restringe el acceso a recursos sensibles.

Por tanto, se cumplen los requisitos básicos de seguridad definidos.

8.5 Validación del solver

Debido a la complejidad del módulo encargado de la resolución de circuitos, su validación se realiza de forma independiente al resto del sistema mediante la resolución de distintos circuitos de prueba. Estos circuitos han sido diseñados para comprobar el correcto funcionamiento del solver ante diferentes configuraciones topológicas y verificar el cumplimiento de las principales leyes de análisis de circuitos eléctricos.

Para cada circuito se comprueba:

- El cumplimiento de la Ley de Corrientes de Kirchhoff (LCK) en todos los nodos.

- El cumplimiento de la Ley de Tensiones de Kirchhoff (LTK) en las distintas mallas.
- La correcta aplicación de la Ley de Ohm en todas las resistencias.

8.5.1 Circuito con supernodo

La Figura 8.22 muestra un circuito que contiene un supernodo. Un supernodo aparece cuando una fuente de tensión ideal conecta dos nodos que no corresponden al nodo de referencia. Este tipo de configuraciones resulta especialmente relevante para validar el método MNA, ya que el análisis nodal clásico no puede resolver directamente este tipo de circuitos sin introducir ecuaciones adicionales.

En esta validación se comprueba que el solver es capaz de:

- Resolver correctamente circuitos con fuentes de tensión flotantes.
- Calcular las corrientes asociadas a las fuentes de tensión.
- Mantener el cumplimiento simultáneo de LCK y LTK.
- Obtener tensiones y corrientes coherentes con la Ley de Ohm.

Para los cálculos de comprobación se utiliza el criterio de signos definido previamente. En las resistencias, la corriente se considera positiva cuando circula desde el terminal positivo al negativo. En las fuentes, tensión y corriente siguen el mismo sentido de referencia, ya que representan elementos que entregan energía al circuito.

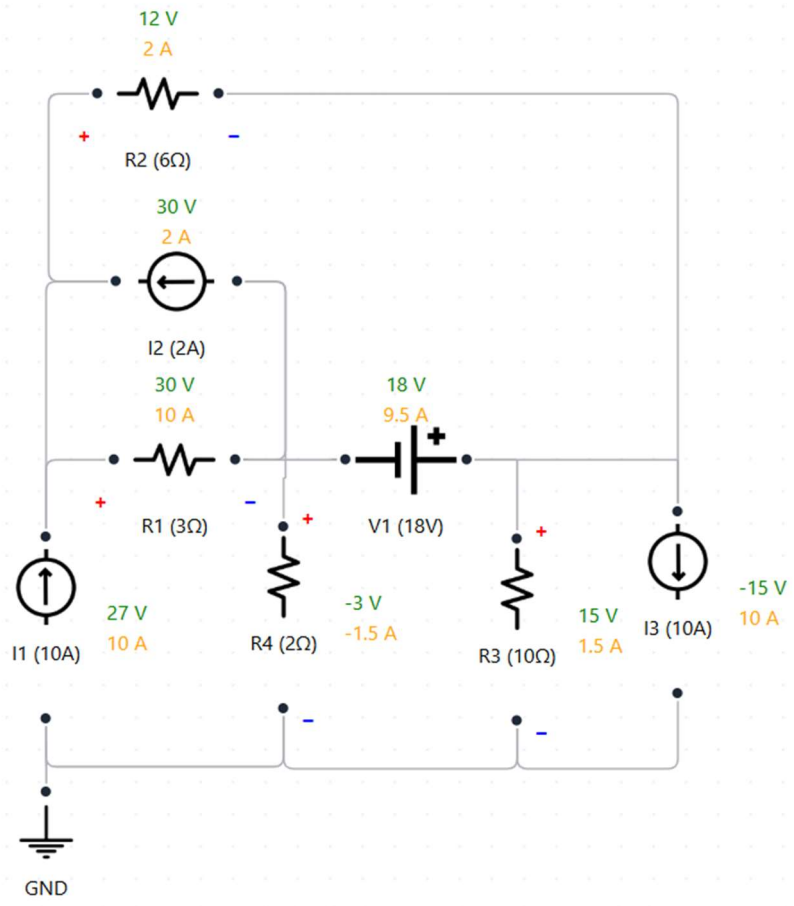


Figura 8.22: Circuito con supernodo

Verificación de la Ley de Corrientes de Kirchhoff

Se comprueba el cumplimiento de LCK en todos los nodos del circuito.

- **Nodo entre GND, I1, R4, R3 e I3.**

$$I_1 = I_3 = I_{R3} + I_3$$
$$10 = -1{,}5 + 1{,}5 + 10$$

- **Nodo entre I1, R1, I2 y R2**

$$I_1 + I_2 = I_{R1} + I_{R2}$$
$$10 + 2 = 10 + 2$$

- **Nodo entre I2, V1, R1 y R4**

$$I_{R1} = I_2 + I_{R4} + I_{V1}$$
$$10 = -1{,}5 + 9{,}5 + 2$$

- **Nodo entre I3, V1, R3 y R2**

$$I_{V1} + I_{R2} = I_3 + I_{R3}$$
$$9{,}5 + 2 = 10 + 1{,}5$$

En todos los nodos se verifica correctamente la conservación de corriente

Verificación de la Ley de Tensiones de Kirchhoff

A continuación, se comprueba el cumplimiento de LTK en las distintas mallas independientes del circuito.

- **Malla formada por I1, R1 y R4**

$$V_{I1} - V_{R1} - V_{R4} = 0$$
$$27 - 30 - (-3) = 0$$

- **Malla formada por V1, R3 y R4**

$$V_{R4} + V_1 - V_{R3} = 0$$
$$-3 + 18 - 15 = 0$$

- **Malla formada por I3 y R3**

$$V_{R3} + V_{I3} = 0$$
$$15 - 15 = 0$$

- **Malla formada por I2 y R1**

$$V_{R1} - V_{I2} = 0$$
$$30 - 30 = 0$$

- **Malla formada por I2, V1 y R2**

$$V_{I2} - V_{R2} - V_1 = 0$$
$$30 - 12 - 18 = 0$$

En todas las mallas se verifica correctamente la conservación de energía.

Verificación de la Ley de Ohm

Finalmente, se comprueba que todas las resistencias del circuito cumplen la Ley de Ohm:

$$V = I \cdot R$$

Los valores obtenidos para tensiones y corrientes en cada resistencia coinciden con los valores esperados, verificando así la correcta resolución del circuito por parte del solver.

8.5.2 Red compleja mallada

La Figura 8.23 muestra un circuito eléctrico con múltiples mallas y caminos de corriente. Este tipo de configuración permite validar la capacidad del solver para resolver automáticamente sistemas nodales de mayor tamaño y complejidad, comprobando además la estabilidad del ensamblado matricial implementado.

Al igual que en el caso anterior, se verificará el cumplimiento de las leyes de Kirchhoff en nodos y mallas, así como la correcta aplicación de la Ley de Ohm en todas las resistencias.

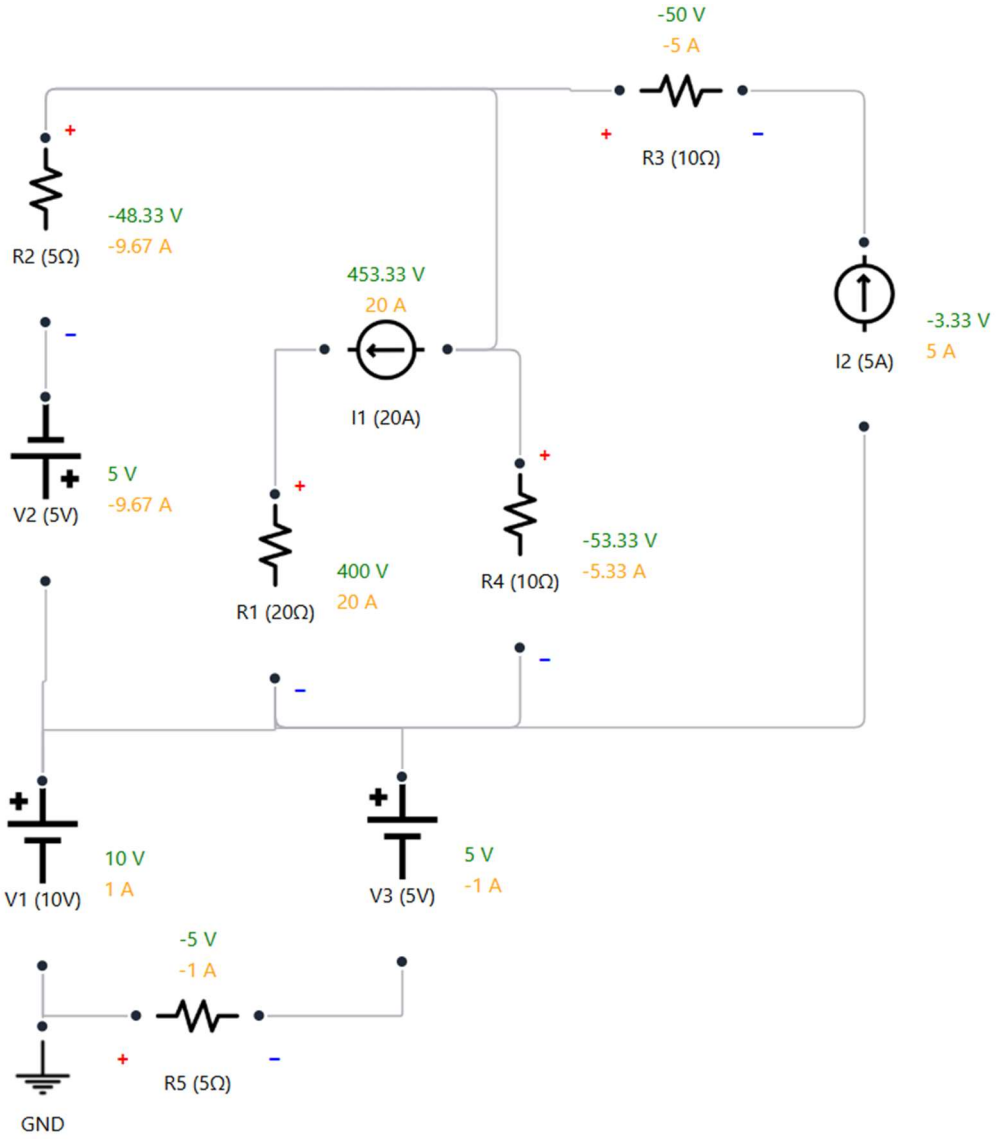


Figura 8.23: Red compleja aleatoria

Verificación de la Ley de Corrientes de Kirchhoff

Se comprueba el cumplimiento de LCK en todos los nodos del circuito.

- **Nodo entre GND, V1 y R5.**

$$I_{V1} - I_{R5} = 0$$
$$1 - 1 = 0$$

- **Nodo entre R5 y V3.**

$$I_{R5} = I_{V3}$$
$$-1 = -1$$

- **Nodo entre V1, V2, R1, R4, I2 y V3.**

$$I_{V1} + I_{V2} + I_{V3} + I_{R1} + I_{R4} = I_{I2}$$
$$1 - 9{,}67 - 1 + 20 - 5{,}33 = 5$$

- **Nodo entre V2, R2.**

$$I_{R2} = I_{V2}$$
$$-9{,}67 = -9{,}67$$

- **Nodo entre R2, R3, R4 e I1.**

$$I_{R2} + I_{R3} + I_{R4} + I_{I1} = 0$$
$$-9{,}67 - 5 + 20 - 5{,}33 = 0$$

- **Nodo entre R1 e I1.**

$$I_{I1} = I_{R2}$$
$$20 = 20$$

- **Nodo entre R3 e I2.**

$$I_{I2} + I_{32} = 0$$
$$5 + (-5) = 0$$

Verificación de la Ley de Tensiones de Kirchhoff

Se comprueba el cumplimiento de LTK en las distintas mallas independientes del circuito.

- **Malla formada por V1, R5 y V3**

$$V_{V1} - V_{V3} + V_{R5} = 0$$
$$10 - 5 - 5 = 0$$

- **Malla formada por V2, R2, I1 y R1**

$$-V_2 + V_{R2} + V_{I1} - V_{R1} = 0$$
$$-5 - 48{,}33 + 453{,}33 - 400 = 0$$

- **Malla formada por R1, I1 Y R4**

$$V_{R1} - V_{I1} - V_{R4} = 0$$

$$400 - 453{,}33 - (-53{,}33) = 0$$

- **Malla formada por R4, R3 e I2**

$$V_{R4} - V_{R3} - V_{I2} = 0$$
$$-53{,}33 - (-50) - (-3{,}33) = 0$$

En todas las mallas se verifica correctamente la conservación de energía.

Verificación de la Ley de Ohm

Al igual que en el ejemplo anterior, se comprueba que todas las resistencias del circuito cumplen la Ley de Ohm:

$$V = I \cdot R$$

Los valores obtenidos son coherentes con las corrientes y resistencias definidas en el circuito, verificando así el correcto funcionamiento del solver en configuraciones de mayor complejidad.

8.6 Conclusiones

El proceso de pruebas y validación ha permitido comprobar que el sistema desarrollado cumple correctamente con los requisitos funcionales y no funcionales establecidos en fases previas del proyecto.

A través de las pruebas funcionales se ha verificado el correcto comportamiento de los principales módulos de la aplicación: el diseño de circuitos, la resolución y análisis de los mismos, la plataforma educativa y la gestión de persistencia de datos. En todos los casos, los resultados obtenidos han sido satisfactorios, validando tanto la interacción del usuario como la lógica interna del sistema.

Asimismo, la validación del sistema ha permitido confirmar que la aplicación responde adecuadamente a los requisitos de calidad definidos. Se ha comprobado que la interfaz es usable e intuitiva, que el rendimiento es adecuado para un uso educativo, que la aplicación es accesible desde distintos entornos al tratarse de una solución web, y que la arquitectura modular facilita su mantenimiento y posible evolución futura. También se ha verificado la existencia de un control básico de seguridad mediante la gestión de roles de usuario.

Por otro lado, la validación específica del solver ha permitido comprobar que el algoritmo implementado resuelve correctamente circuitos resistivos en corriente continua con fuentes independientes de tensión y corriente. Mediante los distintos circuitos de prueba se ha verificado el cumplimiento de las leyes fundamentales de análisis de circuitos eléctricos, incluyendo la Ley de Corrientes de Kirchhoff, la Ley de Tensiones de Kirchhoff y la Ley de Ohm.

Por último, las pruebas de persistencia han confirmado que el sistema es capaz de almacenar y recuperar correctamente circuitos, resultados y datos asociados, garantizando la continuidad del uso de la aplicación.

En conjunto, esta fase de validación demuestra que el sistema desarrollado es funcional, estable y coherente con los objetivos planteados inicialmente, cumpliendo con los requisitos definidos y proporcionando una herramienta útil en el ámbito educativo para el diseño y análisis de circuitos eléctricos.

Capítulo 9

Conclusiones y trabajos futuros

9.1 Introducción

En este capítulo se recogen las principales conclusiones derivadas del desarrollo del sistema propuesto, así como una reflexión sobre el grado de cumplimiento de los objetivos planteados inicialmente. Asimismo, se analizan las limitaciones actuales de la aplicación y se proponen posibles líneas de mejora que permitirían ampliar sus capacidades en futuros desarrollos.

9.2 Conclusiones generales

El presente trabajo ha dado lugar al desarrollo de una aplicación web orientada al apoyo en la enseñanza del análisis de circuitos eléctricos en corriente continua. La herramienta resultante integra en un único entorno la creación, resolución y gestión de ejercicios, facilitando tanto la labor del profesorado como el aprendizaje del alumnado.

Una de las principales aportaciones del sistema es la simplificación del proceso de diseño y análisis de circuitos, permitiendo a los usuarios interactuar con una interfaz visual e intuitiva, sin necesidad de recurrir a herramientas profesionales de mayor complejidad. De este modo, el alumnado puede centrarse en la comprensión de los conceptos fundamentales, mientras que el profesorado dispone de un medio ágil para la generación y evaluación de ejercicios.

El sistema implementa un motor de resolución basado en el método de Análisis Nodal Modificado, lo que permite calcular de forma automática las magnitudes eléctricas relevantes de los circuitos diseñados. Esta funcionalidad, integrada con la interfaz gráfica, proporciona retroalimentación inmediata al usuario, favoreciendo un aprendizaje más dinámico y autónomo.

Asimismo, la incorporación de una plataforma educativa con gestión de usuarios y roles diferenciados permite estructurar el proceso de enseñanza-aprendizaje, facilitando la publicación de ejercicios, la entrega de soluciones por parte del alumnado y la evaluación automática de las mismas.

Desde el punto de vista técnico, la aplicación se ha desarrollado siguiendo una arquitectura cliente-servidor basada en tecnologías web modernas, lo que garantiza su accesibilidad desde distintos dispositivos sin necesidad de instalación. Además, la organización modular del sistema contribuye a su mantenibilidad y a la posibilidad de ampliación futura.

En conjunto, el sistema desarrollado responde de forma satisfactoria a la problemática planteada inicialmente, proporcionando una herramienta funcional, accesible y orientada al ámbito educativo.

9.3 Cumplimiento de objetivos

En el Capítulo 1 se definieron los objetivos generales y específicos del proyecto. A partir del desarrollo realizado y de las pruebas llevadas a cabo, se puede afirmar que dichos objetivos han sido alcanzados de manera satisfactoria.

En primer lugar, se ha diseñado e implementado una interfaz web interactiva que permite tanto al profesorado como al alumnado interactuar con el sistema de forma intuitiva, cumpliendo con los criterios de usabilidad establecidos.

En relación con la gestión de circuitos, se ha desarrollado un editor gráfico que permite la creación de circuitos eléctricos en corriente continua mediante la inserción, conexión y manipulación de componentes. Asimismo, se ha implementado la lógica necesaria para su resolución automática, obteniendo las magnitudes eléctricas de forma precisa.

En el ámbito educativo, se ha conseguido integrar una plataforma que permite la gestión de ejercicios, diferenciando los roles de profesor y alumno. El profesorado puede crear, publicar y evaluar ejercicios, mientras que el alumnado puede acceder a ellos, resolverlos y recibir retroalimentación inmediata.

Además, se ha implementado un sistema de persistencia que permite almacenar y recuperar circuitos, resultados y datos asociados, garantizando la continuidad del trabajo del usuario.

Por último, se ha logrado que la aplicación sea accesible desde distintos dispositivos al tratarse de una solución basada en tecnologías web, y se ha priorizado una interfaz sencilla que reduce la curva de aprendizaje, en línea con los objetivos de accesibilidad y usabilidad.

9.4 Limitaciones actuales

A pesar de los resultados obtenidos, la aplicación presenta ciertas limitaciones que conviene tener en cuenta.

En primer lugar, el sistema se limita al análisis de circuitos en corriente continua en régimen permanente, considerando únicamente componentes básicos como resistencias y fuentes independientes. No se contemplan, por tanto, componentes reactivos como condensadores o inductores, ni elementos no lineales.

Asimismo, aunque el motor de resolución es adecuado para circuitos de complejidad moderada en un contexto educativo, su rendimiento podría verse afectado en circuitos de gran tamaño o con un número elevado de nodos.

En cuanto a la plataforma educativa, si bien permite la gestión básica de ejercicios y la evaluación automática, no incorpora funcionalidades más avanzadas de seguimiento del alumnado o comunicación entre usuarios.

Por último, el sistema de despliegue utilizado, basado en servicios externos, es adecuado para un entorno de desarrollo y pruebas, pero podría requerir una configuración más robusta en un entorno de producción real.

9.5 Líneas de trabajo futuro

El sistema desarrollado presenta un amplio margen de mejora y ampliación en distintas áreas.

En relación con el motor de resolución, una posible línea de evolución consiste en ampliar el soporte a circuitos de corriente alterna, incorporando el tratamiento de

números complejos y el análisis en régimen sinusoidal. Asimismo, sería de interés incluir el análisis en régimen transitorio, lo que permitiría estudiar el comportamiento temporal de los circuitos.

También se podría ampliar el conjunto de componentes disponibles, incorporando elementos como condensadores, inductores, diodos, transistores o amplificadores operacionales, lo que permitiría abordar circuitos de mayor complejidad y acercar la herramienta a escenarios más realistas.

En cuanto al editor de circuitos, podrían incorporarse funcionalidades avanzadas como la representación de formas de onda, la visualización de potencia o la integración de herramientas de medida virtuales, como un osciloscopio.

Desde el punto de vista de la plataforma educativa, existen diversas posibilidades de mejora, como la incorporación de sistemas de comunicación entre profesor y alumnado, la posibilidad de solicitar revisiones de calificaciones o la implementación de mecanismos de evaluación más flexibles, incluyendo la ponderación de ejercicios o la diferenciación entre ejercicios evaluables y de práctica.

Asimismo, se podría mejorar el sistema de seguimiento del progreso del alumnado, proporcionando estadísticas y métricas que faciliten al profesorado el análisis del rendimiento.

9.6 Conclusión final

Como cierre del proyecto, puede afirmarse que el sistema desarrollado constituye una herramienta válida y útil para el apoyo a la enseñanza del análisis de circuitos eléctricos en niveles introductorios. La integración de un editor gráfico, un motor de resolución automática y una plataforma educativa en un único entorno accesible representa una aportación significativa en este ámbito.

Si bien existen limitaciones inherentes al alcance del proyecto, la arquitectura y las decisiones de diseño adoptadas permiten sentar una base sólida sobre la que construir futuras mejoras. En este sentido, el trabajo realizado no solo cumple con los objetivos planteados, sino que abre la puerta a nuevas líneas de desarrollo orientadas a enriquecer la herramienta y ampliar su aplicación en el contexto educativo.

Bibliografía

[1] Moodle, [En línea]. Disponible: https://moodle.org/. [Último acceso: 05 05 2026].

[2] Every Circuit, [En línea]. Disponible: https://everycircuit.com/. [Último acceso: 05 05 2026].

[3] T. Circuits. [En línea]. Disponible: https://www.tinkercad.com/circuits. [Último acceso: 05 05 2026].

[4] LTspice. [En línea]. Disponible: https://ltspice.softmany.com/. [Último acceso: 05 05 2026].

[5] Multisim, [En línea]. Disponible: https://www.multisim.com/. [Último acceso: 05 05 2026].

[6] IEEE, "Electronic Circuits," IEEE Technology Navigator, [En línea]. Disponible: https://technav.ieee.org/topic/electronic-circuits. [Último acceso: 19 04 2026].

[7] Jack E. Kemmerly, Steven M. Durbin y William H. Hayt, Engineering Circuit Analysis, New York: McGraw-Hill, 2012.

[8] F. R. Quintela y. R. C. Redondo Melchor, Diccionario de Ingeniería Eléctrica, Universidad de Salamanca, [En línea]. Disponible: https://electricidad.usal.es/Diccionario. [Último acceso: 19 04 2026].

[9] P. A. Tipler, Física para la ciencia y la tecnología Vol.2, España: Editorial Reverté, 2010.

[10] Wikipedia, [En línea]. Disponible: https://es.wikipedia.org/wiki/Leyes_de_Kirchhoff. [Último acceso: 20 04 2026].

[11] A. Roldán y J.B. Roldán, Análisis simbólico de circuitos mediante técnicas de análisis nodal modificado, Departamento de Electrónica y Tecnología de los Computadores. Facultad de Ciencias. Universidad de Granada., 2010.

[12] Typescript, [En línea]. Disponible: https://www.typescriptlang.org/. [Último acceso: 22 04 2026].

[13] React, [En línea]. Disponible: https://es.react.dev/. [Último acceso: 21 04 2026].

[14] React Flow, [En línea]. Disponible: https://reactflow.dev/. [Último acceso: 21 04 2026].

[15] React Router, [En línea]. Disponible: https://reactrouter.com/. [Último acceso: 21 04 2026].

[16] Tailwindcss, [En línea]. Disponible: https://tailwindcss.com/. [Último acceso: 21 04 2026].

[17] NodeJS, [En línea]. Disponible: https://nodejs.org/es. [Último acceso: 21 04 2026].

[18] Express, [En línea]. Disponible: https://expressjs.com/. [Último acceso: 21 04 2026].

[19] Postgresql, [En línea]. Disponible: https://www.postgresql.org/. [Último acceso: 21 04 2026].

[20] Visual Studio Code, [En línea]. Disponible: https://code.visualstudio.com/. [Último acceso: 21 04 2026].

[21] Netlify, [En línea]. Disponible: https://www.netlify.com/. [Último acceso: 21 04 2026].

[22] Render, [En línea]. Disponible: https://render.com/. [Último acceso: 21 04 2026].

[23] Ken Schwaber y Jeff Schwaber, The Scrum Guide, 2020.

[24] IETF, IETF, 2015. [En línea]. Disponible: https://datatracker.ietf.org/doc/html/rfc7519 . [Último acceso: 2026 04 24].

ANEXO A

Manual de usuario

A.1 Introducción

En este anexo se presenta el manual de usuario de la aplicación desarrollada, cuyo objetivo es facilitar su uso tanto por parte del profesorado como del alumnado.

El manual se estructura en diferentes secciones que describen el acceso al sistema, los usuarios de prueba disponibles y el funcionamiento de las distintas funcionalidades según el rol del usuario.

Para facilitar la evaluación del sistema, se ha desplegado una versión de prueba accesible a través de la siguiente dirección web:

https://iatutxa.netlify.app

A.2 Usuarios de prueba

Con el fin de facilitar la evaluación de la aplicación, se han habilitado los usuarios de prueba que se muestran en las tablas A.1 (usuarios profesores) y A.2 (usuarios alumnos). Los alumnos están asociados a cada profesor, de modo que la Tabla A.2 muestra los alumnos correspondientes al usuario *profesor1*.

Tabla A.1: Tabla de profesores de prueba

Usuario	**Contraseña**
profesor1	p1
profesor2	p2
profesor3	p3
profesor4	p4

Tabla A.2: Tabla de alumnos de prueba

Usuario	**Contraseña**
alumno11	a11

Usuario	Contraseña
alumno12	a12
alumno13	a13

A.3 Acceso a la aplicación

El acceso al sistema se realiza mediante una pantalla de autenticación (Fig. A.1) común para profesorado y alumnado, en la que el usuario debe introducir sus credenciales.

Figura A.1: Página de login

A.3.1 Manual de usuario – Profesor

Una vez autenticado como profesor, el usuario accede al panel principal de gestión de ejercicios.

A.3.2 Barra superior

En la parte superior de la interfaz se encuentra la barra superior (Fig. A.2) que incluye las siguientes opciones:

- **“+ Nuevo circuito”**: permite crear un nuevo ejercicio.
- **Logout**: cierra la sesión actual.

Al pulsar el botón **“+ Nuevo circuito”**, se abre el editor de circuitos, donde el profesor puede diseñar el ejercicio.

Figura A.2: Barra superior del panel de profesor

A.3.3 Panel de gestión de ejercicios

Como se muestra en la Fig A.3, el sistema organiza los ejercicios en tres pestañas: **Borrador**, **Publicado** y **Cerrado**. Cada pestaña muestra, entre paréntesis, el número de ejercicios en dicho estado.

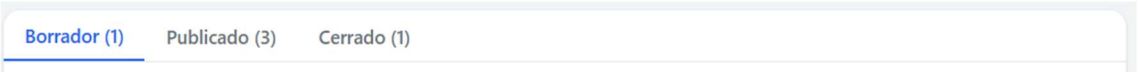


Figura A.3: Organización por pestañas

Pestaña "Borrador"

Contiene los ejercicios que han sido creados pero aún no están disponibles para el alumnado.

Se muestran las siguientes columnas (Fig. A.4): Nombre, Estado (Borrador), Fecha de creación, Fecha de publicación (no definida), Fecha límite (no definida) y Acciones.

Figura A.4: Panel de borrador

Las acciones disponibles son las siguientes:

- **Ver**: Abre un visor del ejercicio en modo lectura (Fig. A.5). No permite edición y muestra el circuito ya resuelto.

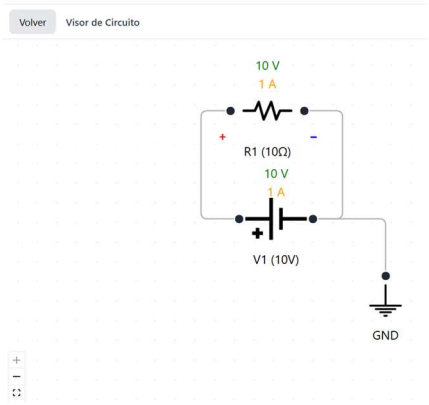


Figura A.5: Visor de circuito

- **Publicar**: Abre una ventana modal (Fig. A.6) donde el profesor debe seleccionar una **fecha límite de entrega**. Tras confirmar, el ejercicio pasa automáticamente a la pestaña *Publicado*.

Figura A.6: Modal de publicación

- **Eliminar**: Muestra un mensaje de confirmación. Si se confirma, el ejercicio se elimina de forma permanente del sistema.

Pestaña “Publicado”

Incluye los ejercicios disponibles para los alumnos.

Se muestran las siguientes columnas (Fig. A.7): Nombre, Estado, Fecha de creación, Fecha de publicación, Fecha límite y Acciones.

Figura A.7: Panel de publicado

Las acciones disponibles son las siguientes:

- **Ver notas**: Abre una ventana modal donde la primera fila muestra las respuestas correctas y, las siguientes, muestran las respuestas de los alumnos. Se utilizan indicadores visuales: verde para respuesta correcta y rojo para respuesta incorrecta. Además, muestra la calificación de cada alumno. En la Fig A.8 se muestra esta ventana modal

Figura A.8: Panel de notas

- **Ver**: Abre el ejercicio en modo visor.
- **Cerrar**: Cambia el estado del ejercicio a cerrado, impidiendo nuevas entregas por parte del alumnado
- **Eliminar**: Elimina el ejercicio del sistema de forma permanente.

Pestaña “Cerrado”

Contiene los ejercicios finalizados.

Las columnas disponibles son las mismas que en la pestaña Publicado.

Las acciones disponibles son**:** Ver notas, Ver y Eliminar. Todas ellas se comportan de la misma forma que en la pestaña de Publicado.

A.3.4 Editor de circuitos

El editor permite al profesor: insertar componentes eléctricos, conectarlos entre sí, diseñar el circuito que formará el ejercicio y resolver el circuito antes de su publicación. La Figura A.9 muestra el editor de circuitos.

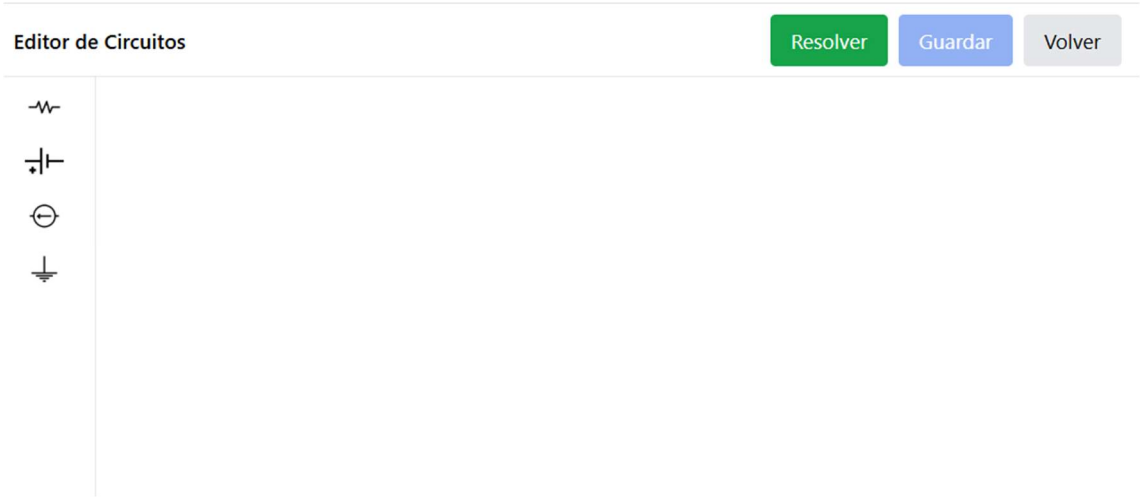


Figura A.9: Editor de circuitos

El editor, accesible desde el botón de "+ Nuevo circuito, se compone de tres áreas principales :

Barra lateral izquierda

Contiene componentes disponibles: resistencia, fuente de tensión, fuente de intensidad y nodo de tierra o GND.

Al seleccionar un componente (excepto GND), se abre un modal (Fig. A.10) donde el usuario introduce su valor eléctrico (ohmios, voltios o amperios). Tras introducir el valor, el componente se añade al área de trabajo.

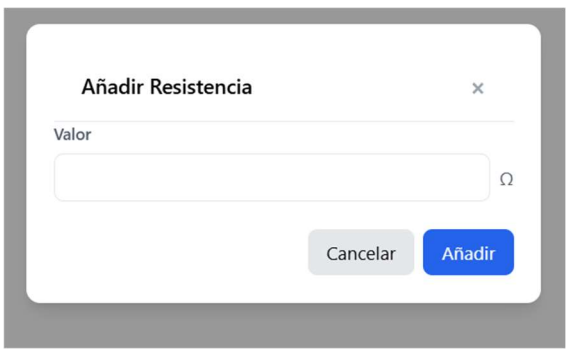


Figura A.10: Modal para introducir un valor de resistencia

Barra superior del editor

Incluye tres botones: resolver, guardar y volver.

El botón **Guardar** permanece deshabilitado hasta que el circuito ha sido resuelto, ya que es necesario disponer de los resultados para almacenarlo.

Al pulsar **Guardar**, se abre un modal (Fig. A.11) para introducir el nombre del ejercicio.

Figura A.11: Modal para nombrar ejercicio

Área de trabajo

En esta zona el usuario puede mover componentes, rotarlos y conectarlos entre si permitiendo la construcción del circuito de forma visual e interactiva.

A.4 Manual de usuario – Alumno

Una vez autenticado como alumno, el usuario accede al panel principal donde puede consultar y resolver los ejercicios propuestos por el profesorado.

A.4.1 Barra superior

En la Figura A.12 se muestra la barra superior del usuario alumno. En ella se encuentra la opción **Logout** que permite cerrar la sesión actual.

Figura A.12: Barra superior del panel de alumno

A.4.2 Panel de ejercicios

El sistema organiza los ejercicios en dos pestañas: publicado y cerrado.

Pestaña “Publicado”

Incluye los ejercicios disponibles para su resolución (Fig. A.13). Se muestran las siguientes columnas: nombre, estado (Publicado), fecha de publicación, fecha límite y acción.

Figura A.13: Panel de ejercicios publicados (alumno)

Las acciones disponibles son:

- **Resolver**: disponible cuando el ejercicio aún no ha sido entregado. Permite acceder al visor de resolución.
- **Ver respuesta**: disponible cuando el alumno ya ha enviado su solución. Permite visualizar la respuesta junto con la calificación obtenida. En este caso, también se muestra la **nota obtenida** por el alumno.

Pestaña “Cerrado”

Incluye los ejercicios cuyo plazo de entrega ha finalizado (Fig. A.14). Las columnas son las mismas que en la pestaña *Publicado*.

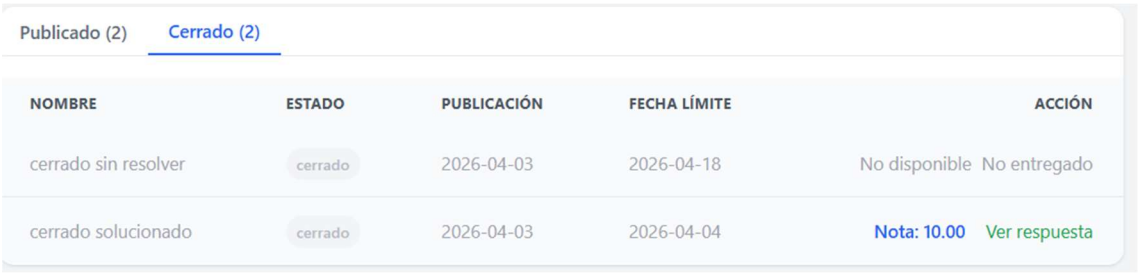


Figura A.14: Panel de ejercicios cerrados (alumno)

Si el ejercicio ha sido entregado la acción disponible es **ver respuesta**, mostrando la nota obtenida.

Si el ejercicio no ha sido entregado no hay acción disponible.

A.4.3 Visor de resolución de circuitos

Este visor (Fig. A.15) permite al alumno resolver un ejercicio.

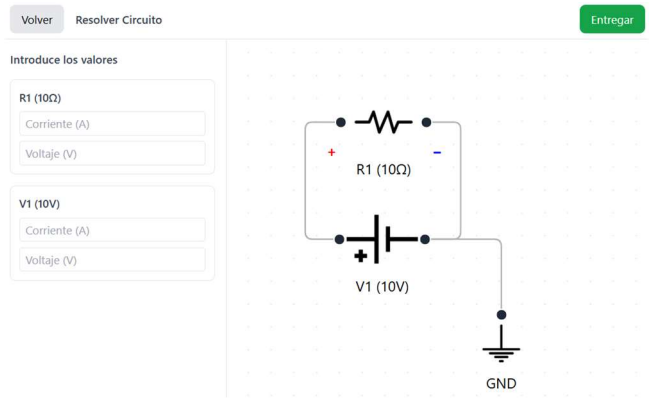


Figura A.15: Visor de resolución

El visor de resolución se compone de 3 áreas principales:

Barra superior

Incluye el botón **“Entregar”** que permite enviar la solución del ejercicio.

Barra lateral izquierda

Contiene los campos de entrada (textboxes) donde el alumno introduce la corriente y tensión para cada componente del circuito.

Área de trabajo:

Muestra el circuito sobre el que se está trabajando.

A.4.4 Visor de respuestas

Este visor (Fig. A.16) permite al alumno consultar su solución una vez entregada.

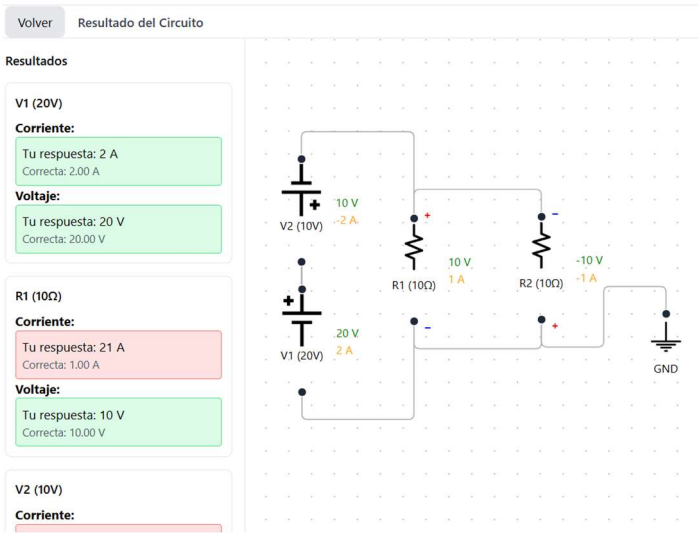


Figura A.16: Visor de respuestas

El visor de resolución de respuestas se compone de 3 áreas principales:

Área de trabajo:

Muestra el circuito con los valores de corriente y tensión.

Barra lateral izquierda:

Para cada componente se muestran la respuesta introducida por el alumno y la respuesta correcta, con los siguientes indicadores visuales: verde si la respuesta es correcta, rojo si la respuesta es incorrecta.

Este visor permite al alumno comparar sus resultados con los correctos, facilitando la retroalimentación y el aprendizaje.

ANEXO B

Código fuente

B.1 Introducción

En este anexo se describe la organización del código fuente del sistema desarrollado. Dado que la aplicación sigue una arquitectura cliente-servidor, el proyecto se encuentra dividido en dos grandes bloques funcionales:

- **Frontend**: interfaz de usuario desarrollada con React.
- **Backend**: API REST desarrollada con Node.js y Express.

Con el objetivo de facilitar la consulta, la ejecución y la reproducibilidad del sistema, el código fuente completo se encuentra disponible en repositorios públicos de GitHub, lo que permite acceder tanto al frontend como al backend de forma independiente.

B.2 Repositorios del proyecto

El código fuente del sistema está alojado en la plataforma GitHub en los siguientes repositorios:

- Backend: **https://github.com/inigo82/backend**
- Frontend: **https://github.com/inigo82/frontend**

B.3 Estructura del backend

El backend presenta una estructura modular basada en rutas, organizada de la siguiente forma que se muestra en la Figura B.1.

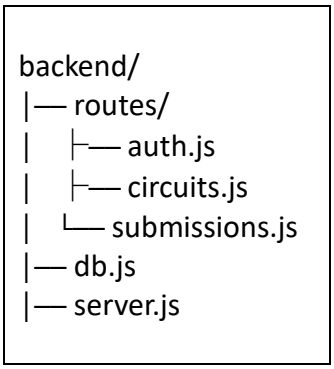


Figura B.1: Estructura de archivos del backend

- server.js: punto de entrada de la aplicación. Configura el servidor Express, registra las rutas y gestiona las peticiones HTTP.
- db.js: configuración de la conexión a la base de datos PostgreSQL mediante un pool de conexiones.
- routes/: contiene los distintos endpoints de la API REST, agrupados por funcionalidad (autenticación, circuitos y entregas).

Esta estructura permite una separación clara de responsabilidades y facilita la escalabilidad del sistema.

B.4 Estructura del frontend

El frontend sigue una arquitectura basada en componentes, propia de React, lo que permite una organización modular del sistema. La estructura de archivos del frontend se muestra en la Figura B.2.

```
frontend/
|— components/
|    ├— ComponentNode.tsx
|   └— ui/
|        ├— AddComponentModal.tsx
|        ├— ConfirmModal.tsx
|        ├— ErrorModal.tsx
|        ├— Modal.tsx
|        ├— NotesModal.tsx
|       └— SaveModal.tsx
|— pages/
|    ├— Login.tsx
|    ├— ProfesorDashboard.tsx
|    ├— AlumnoDashboard.tsx
|    ├— Editor.tsx
|    ├— ResolveCircuit.tsx
|    ├— ViewCircuit.tsx
|   └— ReviewCircuit.tsx
|— logic/
|   └— solver.ts
|— services/
|   └— authService.ts
|— App.tsx
```

Figura B.2: Estructura de archivos del frontend

- components/: componentes reutilizables de la interfaz.
- components/ui/: elementos de interfaz genéricos como modales o formularios.
- pages/: vistas principales de la aplicación.
- logic/: lógica de negocio en el cliente, incluyendo el solver de circuitos.
- services/: comunicación con la API backend.
- App.tsx: definición de rutas de la aplicación.

B.5 Licencia de uso

El código fuente de este proyecto ha sido desarrollado con fines exclusivamente académicos.

Se permite su uso, consulta y modificación con fines educativos, siempre que se cite adecuadamente la autoría original del proyecto.

ANEXO C

Estructura de la base de datos

C.1 Introducción

En este anexo se presenta la estructura de la base de datos utilizada en el sistema, implementada mediante PostgreSQL. Se incluyen las sentencias de definición de las principales tablas que conforman el sistema, reflejando la implementación física del modelo de datos.

El diseño conceptual de la base de datos, así como la descripción detallada de las entidades y sus relaciones, se ha desarrollado en la Sección 6.3, donde se muestra el correspondiente diagrama entidad–relación.

C.2 Definición de tablas

La Figura C.1 muestra las sentencias SQL correspondientes a la creación de las tablas del sistema.

```sql
-- Tabla de usuarios
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    usuario VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    role VARCHAR(20) NOT NULL
);

-- Tabla de circuitos
CREATE TABLE circuits (
    id SERIAL PRIMARY KEY,
    data JSONB,
    profesor_id INTEGER,
    status VARCHAR(20) DEFAULT 'borrador',
    created_at TIMESTAMP DEFAULT now(),
    published_at TIMESTAMP,
    due_date TIMESTAMP,
    name VARCHAR(255),
    result JSONB,
    FOREIGN KEY (profesor_id) REFERENCES users(id) ON DELETE CASCADE
);

-- Tabla de entregas de alumnos
CREATE TABLE submissions (
    id SERIAL PRIMARY KEY,
    student_id INTEGER NOT NULL,
    circuit_id INTEGER NOT NULL,
```

```
    solution JSONB,
    grade NUMERIC,
    submitted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE (student_id, circuit_id),
    FOREIGN KEY (student_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (circuit_id) REFERENCES circuits(id) ON DELETE CASCADE
);

-- Relación profesor-alumno
CREATE TABLE teacher_students (
    id SERIAL PRIMARY KEY,
    teacher_id INTEGER NOT NULL,
    student_id INTEGER NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE (teacher_id, student_id),
    FOREIGN KEY (teacher_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (student_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Figura C.1: SQL de creación de la BD

C.3 Consideraciones

Las tablas presentadas en este anexo corresponden a la implementación física del modelo de datos descrito en la Sección 6.3. En esta sección se ha optado por incluir exclusivamente la definición SQL de las tablas, evitando redundar en la explicación conceptual ya desarrollada en el capítulo de diseño.

El uso de claves primarias, claves foráneas y restricciones de unicidad permite garantizar la integridad de los datos y la coherencia de las relaciones entre las distintas entidades del sistema.