

**EEP
iGroup**

SISTEMA DE EXÁMENES

**CFGS
201X**

ESCUELA DE ESTUDIOS PROFESIONALES

CICLO FORMATIVO DE GRADO SUPERIOR EN 2025



TRABAJO DE FIN DE GRADO

Título del trabajo de fin de Grado:

Sistema de exámenes

NOMBRE Y APELLIDOS DEL ESTUDIANTE:

Steven Simbaña Merchán

(MES PRESENTACION de 201x)



EEP iGroup Arturo Soria
TÉCNICO SUPERIOR EN DESARROLLO DE APLICACIONES WEB

SISTEMA DE EXAMENES

ALUMNO: STEVEN SIMBAÑA MERCHÁN

TUTOR ACADÉMICO: PEDRO hernández

FECHA DE PRESENTACIÓN: 30/05/2025

BREVE DESCRIPCIÓN:

El proyecto consiste en el desarrollo de una **aplicación web de gestión de exámenes en línea**, diseñada para facilitar tanto la **creación, administración y evaluación de pruebas** por parte de administradores (docentes), como la **realización de dichas evaluaciones por los usuarios** (estudiantes). La plataforma se construyó utilizando tecnologías modernas como **Java con Spring Boot para el backend** y **Angular para el frontend**, implementando una arquitectura basada en el patrón **Modelo-Vista-Controlador (MVC)** y un **modelo de base de datos relacional** con MySQL.

Entre sus funcionalidades principales se incluyen: registro y autenticación de usuarios con roles diferenciados, gestión de categorías y exámenes, temporización de pruebas, evaluación automática de resultados y visualización inmediata del desempeño. El proyecto destaca por su enfoque en la **seguridad, la escalabilidad, la experiencia de usuario (UX/UI)** y la organización modular del código, lo que permite una alta mantenibilidad y expansión futura.

FIRMA DEL TUTOR

FIRMA DEL ALUMNO

Resumen

Introducción

En la actualidad, la tecnología desempeña un papel crucial en la transformación de los procesos educativos. Este Trabajo de Fin de Grado (TFG) presenta el desarrollo de una aplicación web orientada a la gestión de exámenes académicos, que incorpora funcionalidades como la creación de preguntas, la evaluación automática de respuestas y la generación de reportes detallados. El propósito de este estudio es proporcionar una solución ágil, eficiente y escalable que atienda las demandas de instituciones educativas y estudiantes.

Metodología

Para la implementación del proyecto se adoptaron metodologías ágiles de desarrollo de software. En el frontend se utilizó Angular, combinando Bootstrap y Angular Material para lograr una interfaz moderna y funcional, mientras que en el backend se empleó Java con Spring Boot, siguiendo el patrón de diseño REST. La persistencia de datos se gestionó mediante MySQL, y se utilizaron herramientas como Visual Studio Code, Eclipse, Postman y GitHub para el desarrollo y control de versiones. Asimismo, se integraron diversas librerías en el frontend (por ejemplo, MatProgressSpinner, MatSelect, MatSlideToggle, entre otras), y en el backend se incorporaron dependencias esenciales como Spring Data JPA, Spring Security, JWT y Lombok.

Resultados

El sistema, desplegado en entorno local, permite a los usuarios (estudiantes y profesores) realizar exámenes y ver el resultado global en una pantalla específica, con opción a imprimir o guardar en PDF. Aunque la aplicación no presenta diseño responsive y las pruebas de rendimiento han sido limitadas, ha demostrado gestionar de manera funcional varios exámenes completos. El feedback recibido de compañeros y del tutor confirma su potencial en la optimización de procesos evaluativos.

Conclusiones

Este proyecto contribuye a la digitalización educativa y sienta las bases para futuras investigaciones en tecnología educativa, destacándose por su capacidad de mejorar la calidad y accesibilidad de los procesos de evaluación.

Palabras clave

Educación Digital, Evaluación Automatizada, Desarrollo de Aplicaciones Web, Tecnología Educativa.

Índice

Introducción	7
Presentación del Tema y su Relevancia.....	7
Objetivos del TFG.....	7
Justificación del Proyecto	7
Marco teórico	8
Tecnologías en la Educación Superior.....	8
Evaluación Automatizada y Gestión de Exámenes	8
Desafíos y Tendencias en la Digitalización Educativa	9
Evaluación Formativa y Sumativa en Entornos Digitales	9
Plataformas de Gestión de Aprendizaje (LMS)	9
Automatización en la Evaluación	10
Conceptos Clave y Marco Conceptual.....	10
Justificación de Teorías y Modelos.....	11
Identificación de Brechas en la Literatura	12
Planificación y análisis de requisitos.	14
Conclusión del Marco Teórico.....	15
Metodología	15
Enfoque Metodológico.....	15
Herramientas y Tecnologías Utilizadas	17
Reflexión sobre la Elección de Herramientas.	19

Diseño y Arquitectura de la Aplicación.....	19
Decisiones de Diseño Significativas en Aplicaciones Web.....	22
Conclusión metodología.	25
Desarrollo e implementación	26
Metodología de Desarrollo	26
Fases del desarrollo.....	28
Evidencia visual.....	49
Resultados	57
Presentación de la Aplicación Desarrollada	57
Resultados de las Pruebas y Evaluaciones.....	58
Comparación con Otras Soluciones Similares	59
Discusión.....	60
Análisis de los Resultados Obtenidos	60
Limitaciones del Proyecto y Áreas de Mejora	61
Implicaciones Prácticas y Teóricas de los Resultados	61
Conclusiones	62
Resumen de los Hallazgos Clave del Proyecto.....	62
Respuestas a las Preguntas de Investigación Planteadas	62
Lecciones Aprendidas Durante el Desarrollo del Proyecto	63
Referencias.....	64
Anexos	66

Introducción

Presentación del Tema y su Relevancia

La transformación digital en el ámbito educativo es uno de los motores que impulsan la innovación en los procesos de enseñanza y evaluación. En este contexto, el desarrollo de aplicaciones web que faciliten la gestión de exámenes y la evaluación automática se convierte en una necesidad imperante. Este TFG se centra en la creación de una plataforma diseñada para la administración de exámenes académicos, la generación de reportes y la optimización de procesos evaluativos. La relevancia de este proyecto radica en la capacidad de la tecnología para mejorar la eficiencia de los docentes y la experiencia de aprendizaje de los estudiantes, en un entorno marcado por desafíos modernos como la creciente digitalización y la necesidad de respuestas rápidas y adaptativas en el sector educativo.

Objetivos del TFG

El objetivo principal de este trabajo es desarrollar una aplicación web funcional y escalable, que permita la creación, administración y evaluación de exámenes de manera automatizada. Entre los objetivos específicos se destacan:

Implementar una interfaz de usuario intuitiva y moderna utilizando Angular, Bootstrap y Angular Material.

Desarrollar un backend robusto con Java y Spring Boot, integrando un sistema de autenticación basado en JWT y persistencia de datos en MySQL.

Automatizar el proceso de evaluación y la generación de reportes de resultados.

Optimizar la gestión de exámenes, facilitando la administración tanto para usuarios (estudiantes) como para administradores (profesores).

Justificación del Proyecto

La importancia de este proyecto se fundamenta en la creciente demanda de soluciones tecnológicas que impulsen la digitalización en educación. La implementación de este sistema

no solo responde a la necesidad de modernizar el proceso de evaluación académica, sino que también busca aportar al conocimiento existente mediante el desarrollo de una herramienta innovadora y adaptable. Al abordar problemas comunes en la gestión de exámenes –como la ineficiencia en la evaluación manual y la falta de reportes detallados–, este trabajo tiene el potencial de generar un impacto significativo en la calidad de la educación, beneficiando tanto a instituciones educativas como a sus estudiantes.

Marco teórico

La integración de tecnologías digitales ha transformado significativamente los procesos de enseñanza y aprendizaje en la educación superior. La evaluación académica, como componente esencial, se ha beneficiado de herramientas digitales que automatizan y optimizan la creación, distribución y corrección de pruebas, mejorando la eficiencia y la experiencia del estudiante. Este marco teórico analiza los conceptos y enfoques relevantes relacionados con la gestión digital de exámenes, explorando las bases teóricas del uso de tecnologías en la evaluación educativa, las metodologías existentes y las tendencias actuales en el diseño e implementación de sistemas de evaluación en línea.

Tecnologías en la Educación Superior

La transformación digital en las instituciones de educación superior implica la integración de herramientas digitales tanto en la enseñanza como en la gestión institucional, promoviendo una educación más accesible y personalizada. La adopción de plataformas digitales facilita la administración, documentación, seguimiento y entrega de programas educativos, permitiendo a los docentes organizar cursos, gestionar contenidos y evaluar el progreso de los estudiantes de manera más eficiente.

Evaluación Automatizada y Gestión de Exámenes

La implementación de sistemas automatizados de evaluación en entornos virtuales de aprendizaje mejora la eficiencia y la retroalimentación en el proceso educativo. La

automatización permite calificar y proporcionar retroalimentación a los estudiantes sin intervención humana directa, aplicándose tanto a respuestas de opción múltiple como a respuestas escritas más complejas. Las ventajas incluyen rapidez en la corrección, objetividad en las calificaciones y la posibilidad de personalizar el aprendizaje al identificar áreas de mejora para cada estudiante.

Desafíos y Tendencias en la Digitalización Educativa

La adopción de tecnologías educativas enfrenta retos institucionales como la resistencia al cambio y la falta de formación docente. Además, es crucial adaptar las metodologías de investigación a los contextos culturales locales, considerando la diversidad y las realidades específicas de cada entorno educativo. La integración de tecnologías emergentes, como la inteligencia artificial, ofrece oportunidades para personalizar las experiencias de aprendizaje y optimizar la gestión educativa, siempre que se implementen de manera ética y contextualizada [.bancomundial.org](https://www.bancomundial.org)

Evaluación Formativa y Sumativa en Entornos Digitales

En entornos digitales, la evaluación formativa se implementa mediante herramientas como cuestionarios en línea, foros de discusión y actividades interactivas, proporcionando retroalimentación constante para mejorar el proceso educativo. La evaluación sumativa, por su parte, se realiza al final de un período educativo para calificar el rendimiento de los estudiantes, utilizando exámenes finales, proyectos o trabajos de investigación. La aplicación adecuada de ambos tipos de evaluación es fundamental para diseñar sistemas de gestión de exámenes eficaces y adaptados a las necesidades actuales.

Plataformas de Gestión de Aprendizaje (LMS)

Los Learning Management Systems (LMS) son plataformas digitales que facilitan la administración y entrega de programas educativos. Permiten a los docentes organizar cursos, gestionar contenidos, evaluar el progreso de los estudiantes y fomentar la interacción en

línea. Entre sus características clave se encuentran la gestión centralizada de la formación, la personalización del aprendizaje y la posibilidad de realizar evaluaciones automatizadas.

Automatización en la Evaluación

La automatización en la evaluación implica el uso de tecnologías, como la inteligencia artificial, para calificar y proporcionar retroalimentación a los estudiantes sin intervención humana directa. Esta tecnología puede aplicarse tanto a respuestas de opción múltiple como a respuestas escritas más complejas. Las ventajas de la evaluación automatizada incluyen eficiencia y rapidez en la corrección, objetividad y consistencia en las calificaciones, y la posibilidad de personalizar el aprendizaje al identificar áreas de mejora para cada estudiante. Sin embargo, también presenta desafíos, como la precisión en la evaluación de respuestas subjetivas y la privacidad y seguridad de los datos de los estudiantes.

Conceptos Clave y Marco Conceptual

Evaluación Formativa y Sumativa en Entornos Digitales. La evaluación es esencial en el proceso educativo, permitiendo medir el progreso de los estudiantes y la eficacia de los métodos pedagógicos. En entornos digitales, se distinguen dos enfoques principales: la evaluación formativa, que se realiza durante el proceso de aprendizaje para proporcionar retroalimentación continua, y la evaluación sumativa, que se aplica al final de un período para calificar el rendimiento del estudiante. Herramientas digitales como cuestionarios en línea, foros de discusión y actividades interactivas facilitan la implementación de ambos tipos de evaluación, adaptándose a las necesidades actuales del entorno educativo.

Plataformas de Gestión de Aprendizaje (LMS). Los Learning Management Systems (LMS) son plataformas digitales que facilitan la administración, documentación, seguimiento y entrega de programas educativos. Permiten a los docentes organizar cursos,

gestionar contenidos, evaluar el progreso de los estudiantes y fomentar la interacción en línea. Entre sus características clave se encuentran la gestión centralizada de la formación, la personalización del aprendizaje, el acceso a materiales desde cualquier lugar y la posibilidad de realizar evaluaciones automatizadas. El uso de LMS en la educación moderna ha transformado la manera en que se imparten y gestionan los cursos, ofreciendo flexibilidad y eficiencia tanto para docentes como para estudiantes.

Automatización en la Evaluación. La automatización en la evaluación implica el uso de tecnologías, como la inteligencia artificial (IA), para calificar y proporcionar retroalimentación a los estudiantes sin intervención humana directa. Esta tecnología puede aplicarse tanto a respuestas de opción múltiple como a respuestas escritas más complejas. Las ventajas de la evaluación automatizada incluyen eficiencia y rapidez en la corrección, objetividad y consistencia en las calificaciones, y la posibilidad de personalizar el aprendizaje al identificar áreas de mejora para cada estudiante. Sin embargo, también presenta desafíos, como la precisión en la evaluación de respuestas subjetivas, la privacidad y seguridad de los datos de los estudiantes, y la dependencia de la tecnología. Integrar la automatización en la evaluación requiere una planificación cuidadosa y una comprensión profunda de sus implicaciones para garantizar una implementación ética y efectiva.

Justificación de Teorías y Modelos

Teoría del Aprendizaje Constructivista. La teoría del aprendizaje constructivista sostiene que el conocimiento se construye activamente por el estudiante a través de la interacción con su entorno y la reflexión sobre sus experiencias. Este enfoque promueve un aprendizaje significativo, donde el estudiante es el protagonista de su proceso educativo, desarrollando habilidades de pensamiento crítico y resolución de problemas (Chérrez & Quevedo, 2018). En el contexto de la educación digital, el constructivismo se adapta eficazmente, ya que las tecnologías de la información y la comunicación (TIC) ofrecen

entornos interactivos que facilitan la construcción del conocimiento. Herramientas como plataformas de aprendizaje en línea permiten a los estudiantes explorar, experimentar y colaborar, alineándose con los principios constructivistas. La aplicación de esta teoría en tu proyecto se refleja en el diseño de una herramienta que fomenta la autonomía del estudiante, permitiéndole gestionar su aprendizaje y evaluación de manera activa y personalizada.

Modelos de Evaluación Automatizada en Entornos Virtuales. La evaluación automatizada en entornos virtuales se basa en modelos que utilizan tecnologías como la inteligencia artificial (IA) para calificar y proporcionar retroalimentación a los estudiantes sin intervención humana directa. Estos modelos permiten una evaluación eficiente, objetiva y personalizada, adaptándose a las necesidades individuales de los estudiantes (Magistrum University, 2024). La implementación de estos modelos en tu proyecto justifica la elección de herramientas que automatizan el proceso de evaluación, ofreciendo retroalimentación inmediata y adaptativa. Esto no solo mejora la eficiencia del proceso educativo, sino que también empodera al estudiante al brindarle información oportuna sobre su desempeño, alineándose con los principios del aprendizaje constructivista.

Identificación de Brechas en la Literatura

La revisión de literatura sobre evaluación automatizada en entornos educativos evidencia avances relevantes en la incorporación de tecnologías como la inteligencia artificial (IA), especialmente en cuanto a la eficiencia y objetividad de las evaluaciones. No obstante, se identifican diversas brechas que requieren mayor atención:

Falta de generación de informes detallados para docentes. Aunque la automatización ha mejorado la rapidez en la corrección, la mayoría de los estudios se enfocan en la eficiencia del proceso evaluativo, dejando de lado la generación de informes pedagógicos detallados. Esta omisión limita la capacidad de los docentes para identificar

debilidades específicas en sus estudiantes y ajustar sus estrategias de enseñanza de forma personalizada.

Limitaciones en la evaluación de respuestas abiertas. Los sistemas actuales se orientan principalmente a ítems de opción múltiple o respuestas cerradas, presentando dificultades para interpretar respuestas abiertas que demandan análisis crítico. Esta limitación impide evaluar de forma integral el aprendizaje, especialmente en áreas que valoran la argumentación escrita y el razonamiento complejo.

Consideraciones éticas y de privacidad. La aplicación de IA en contextos educativos plantea preocupaciones relacionadas con la privacidad de los datos, la transparencia de los algoritmos y el consentimiento informado. La escasez de marcos éticos sólidos puede generar desconfianza y dificultar la adopción generalizada de estas tecnologías.

Escasa personalización del aprendizaje. Pese al potencial de la IA para adaptar contenidos y retroalimentación a las necesidades individuales, muchos sistemas aún no aprovechan plenamente esta capacidad. Esto limita la posibilidad de generar experiencias de aprendizaje diferenciadas, reduciendo el impacto formativo de las herramientas digitales.

Tecnologías, Herramientas y Metodologías Relevantes

Tecnologías empleadas. Para cumplir con los objetivos del proyecto, se seleccionaron herramientas tecnológicas que aseguran eficiencia, escalabilidad y mantenibilidad.

- **Lenguajes de programación.** Se eligieron Java para el desarrollo backend, debido a su robustez, soporte para programación orientada a objetos y amplia disponibilidad de bibliotecas; y JavaScript para el frontend, dada su versatilidad en la creación de interfaces interactivas.
- **Frameworks y bibliotecas.** En el frontend se utilizó Angular, que, gracias a su arquitectura basada en componentes, permite desarrollar interfaces dinámicas y

coherentes. Se integraron librerías como *Angular Material* y *NgxUiLoader* para mejorar la experiencia de usuario. En el backend, se contempló el uso de Spring Boot como soporte estructural para la implementación de servicios RESTful, facilitando la arquitectura orientada a servicios.

- **Base de datos.** Se empleó MySQL como sistema de gestión de bases de datos relacional, por su fiabilidad, eficiencia y amplio respaldo comunitario. Esta elección garantiza una gestión segura y estructurada de exámenes, usuarios y resultados.
- **Entornos y herramientas de desarrollo.** Se utilizó Eclipse como entorno de desarrollo para Java, destacando por sus capacidades de depuración y refactorización. Postman se implementó para la prueba y verificación de APIs, asegurando la correcta comunicación entre frontend y backend. GitHub cumplió funciones de control de versiones y documentación del proyecto.

Planificación y análisis de requisitos.

Se adoptó una metodología incremental, con sesiones de trabajo de dos horas diarias durante los fines de semana. El desarrollo comenzó con la implementación del sistema de autenticación (registro y login), una funcionalidad crítica para garantizar el acceso seguro. Primero se desarrolló y probó el backend usando Postman, asegurando la correcta funcionalidad de las APIs.

Posteriormente, se avanzó con el desarrollo del frontend en Angular, integrando interfaces que permitieran una interacción fluida. Se implementaron operaciones CRUD para las entidades clave del sistema, lo que facilitó la manipulación dinámica de los datos. La interfaz fue diseñada considerando criterios de usabilidad, estética y funcionalidad.

Aplicación práctica en el proyecto. Tras la validación del sistema de autenticación, se integraron los módulos del frontend y backend mediante APIs REST. Esta integración fue verificada mediante pruebas manuales y automatizadas en Postman, asegurando la

interoperabilidad y coherencia entre componentes. Durante todo el proceso, GitHub sirvió para documentar avances y mantener el historial de cambios, facilitando el mantenimiento futuro del sistema.

También se elaboraron guías de usuario y se documentaron las iteraciones, consolidando un producto funcional, accesible y fácilmente escalable.

Conclusión del Marco Teórico.

La revisión teórica respalda la pertinencia de digitalizar los procesos de enseñanza y evaluación mediante herramientas tecnológicas que mejoran la eficiencia, accesibilidad y objetividad. Tecnologías como Java, Angular, MySQL y Postman, combinadas con metodologías de desarrollo estructurado, permiten construir sistemas automatizados robustos y adaptables.

Sin embargo, la literatura también señala desafíos persistentes, como la falta de generación de informes analíticos, la dificultad para evaluar respuestas abiertas y las implicaciones éticas del uso de IA. Estos vacíos justifican la necesidad de desarrollar soluciones integrales y éticamente responsables que optimicen tanto la experiencia del estudiante como la labor docente.

Metodología

Enfoque Metodológico

El proyecto fue desarrollado de manera individual mediante un enfoque metodológico personalizado. Se estableció una rutina de trabajo enfocada en los fines de semana, dedicando entre tres y cuatro horas por jornada, con descansos de 20 minutos por cada hora de trabajo continuo. Esta estrategia fue elegida por su flexibilidad y comodidad, permitiendo un equilibrio entre productividad y bienestar, así como la posibilidad de realizar pruebas y ajustes conforme al ritmo de avance personal.

Fases del Proceso de Desarrollo. El desarrollo de la aplicación se dividió en ocho etapas secuenciales, permitiendo avanzar progresivamente en la construcción y validación de funcionalidades clave:

- **Primera etapa.** Backend de login y registro

Se implementaron las funcionalidades básicas de autenticación. Se desarrollaron clases para las entidades Usuario y Rol, se configuró la conexión con MySQL y se probaron los servicios mediante Postman.

- **Segunda etapa.** Frontend de login y registro

Se crearon los componentes en Angular para el ingreso y registro de usuarios. Se validó su integración con el backend mediante pruebas desde consola.

- **Tercera etapa.** Pantallas inicial y detalle de usuario

Se implementaron vistas adaptadas tanto para usuarios administradores como para usuarios finales, mostrando la información correspondiente a cada perfil.

- **Cuarta etapa.** Backend de categorías, exámenes y preguntas

Se añadieron servicios y clases para manejar estas entidades. Se desarrollaron operaciones CRUD y se validaron mediante Postman.

- **Quinta etapa.** Frontend del módulo administrativo

Se programaron funciones para gestionar la creación, edición y eliminación de categorías, exámenes y preguntas desde la interfaz administrativa.

- **Sexta etapa.** Frontend del módulo de usuario

Se diseñaron vistas para que los estudiantes pudieran consultar categorías, exámenes y preguntas disponibles.

- **Séptima etapa.** Funcionalidad de examen

Se construyó la interfaz para realizar exámenes, incorporando un temporizador, la generación de resultados y la opción de imprimirlos.

- **Octava etapa.** Mejora del diseño

Se realizaron ajustes visuales para mejorar la experiencia del usuario, modificando elementos de la interfaz según principios de diseño responsivo y accesible.

Organización y Gestión del Proyecto. Se utilizó Git como sistema de control de versiones, y GitHub como plataforma de alojamiento y seguimiento del progreso. Dado que el proyecto fue realizado de forma individual, la planificación de tareas se mantuvo en un formato informal, sin herramientas adicionales de gestión.

Herramientas y Tecnologías Utilizadas

Lenguajes de Programación

- **Java.** Lenguaje elegido para el backend por su enfoque orientado a objetos y su robustez. Se aplicaron principios como encapsulación, herencia y polimorfismo, facilitando un desarrollo modular y mantenible.
- **TypeScript.** Utilizado en el frontend con Angular, gracias a su tipado estático y capacidad para detectar errores en etapas tempranas del desarrollo.

Frameworks y Bibliotecas

- **Spring Boot.** Framework principal del backend que permitió simplificar la creación de aplicaciones RESTful. Se utilizaron módulos como:
 - **spring-boot-starter-data-jpa.** facilitó la conexión con MySQL mediante JPA.
 - **spring-boot-starter-web.** habilitó la creación de servicios REST.
 - **spring-boot-starter-security.** integró mecanismos de autenticación y autorización.
- **MySQL Connector/J.** Driver JDBC que permitió la persistencia de datos.
- **Lombok.** Redujo la escritura de código repetitivo mediante anotaciones automáticas para métodos comunes.

- ***JSON Web Token (JWT)***. Utilizado para implementar autenticación basada en tokens seguros.
- ***Jakarta Validation API***. Facilitó la validación de datos mediante anotaciones.
- ***Angular***. Framework del frontend que permitió una arquitectura modular y escalable, usando componentes reutilizables.
- ***Angular Material***. Biblioteca UI que implementa Material Design. Se usaron múltiples módulos como:

MatButtonModule, MatToolbarModule, MatInputModule, MatSnackBarModule, MatSidenavModule, entre otros, para construir una interfaz atractiva y funcional.
- ***NgxUiLoaderModule***. Implementó indicadores de carga visuales para mejorar la experiencia del usuario.
- ***HttpClient de Angular***. Servicio HTTP que permitió la conexión eficiente entre frontend y backend.

Control de Versiones

- ***Git***. sistema distribuido de control de versiones.
- ***GitHub***. repositorio centralizado para gestión del código.

Herramientas de Pruebas y Depuración

- ***Postman***. Utilizado para verificar el correcto funcionamiento de las API REST, permitiendo validar las respuestas de los servicios del backend.

Entornos de Desarrollo

- ***Eclipse***. IDE utilizado para programar en Java.
- ***Visual Studio Code***. Editor de código para el desarrollo frontend en Angular, con extensiones útiles para aumentar la productividad.

Reflexión sobre la Elección de Herramientas.

La selección de herramientas y tecnologías para este proyecto se basó en criterios de robustez, eficiencia y compatibilidad. Java y Spring Boot ofrecieron una base sólida para el desarrollo del backend, facilitando la implementación de funcionalidades complejas y garantizando un alto rendimiento. La integración de Spring Security mediante spring-boot-starter-security proporcionó mecanismos de autenticación y autorización esenciales para proteger la aplicación.

En el frontend, Angular y TypeScript permitieron construir una interfaz de usuario dinámica y responsiva, mejorando la experiencia del usuario. El uso de Angular Material y NgxUiLoaderModule enriqueció la interfaz con componentes visuales coherentes y funcionalidades adicionales. La incorporación de HttpClient facilitó la comunicación eficiente con el backend, permitiendo realizar operaciones CRUD de manera sencilla y optimizada.

Herramientas como Git, GitHub y Postman facilitaron el control de versiones, la colaboración y las pruebas, asegurando un desarrollo ágil y eficiente. En conjunto, estas elecciones tecnológicas contribuyeron al éxito del proyecto, permitiendo desarrollar una aplicación web robusta, escalable y mantenible.

Diseño y Arquitectura de la Aplicación

Introducción General. La aplicación está diseñada para facilitar la realización y calificación de exámenes de manera eficiente. Se optó por una arquitectura monolítica basada en un modelo cliente-servidor local, adecuada para entornos que no requieren alta escalabilidad y que pueden operar sin conexión a Internet.

Importancia de una Buena Arquitectura y Diseño. Una arquitectura bien estructurada garantiza la funcionalidad, escalabilidad y mantenibilidad de la aplicación. La elección de una arquitectura monolítica permite un entorno de ejecución controlado y

autónomo, optimizando la experiencia del usuario y asegurando una ejecución eficiente en la captura y calificación de exámenes.

Aspectos a Detallar en el Desarrollo

- **Estructura de la base de datos.** Se describirá el modelo entidad-relación y las estrategias de normalización empleadas para garantizar la integridad y eficiencia en el manejo de los datos.
- **Decisiones de diseño de la arquitectura del software.** Se incluirán diagramas que ilustren la distribución de responsabilidades entre la lógica de negocio, la presentación y el manejo de datos.
- **Consideraciones de usabilidad, UX y UI.** Se explicarán las estrategias implementadas para asegurar que la aplicación sea intuitiva, fácil de usar y eficiente para los usuarios finales.

Arquitectura del Software

Patrón Arquitectónico: Modelo-Vista-Controlador (MVC). Se adoptó el patrón arquitectónico MVC, que separa las responsabilidades de la aplicación en tres componentes principales: modelo, vista y controlador. Esta separación facilita la organización del código, mejora la mantenibilidad y permite una escalabilidad eficiente del sistema.

Justificación de la Elección del Patrón MVC

- **Separación de responsabilidades.** Cada componente tiene funciones bien definidas, lo que facilita el desarrollo y la organización del código.
- **Mantenibilidad y escalabilidad.** La estructura modular del MVC permite modificar o ampliar funcionalidades sin afectar al resto del sistema.
- **Reutilización del código.** La lógica de negocio centralizada en el modelo puede ser reutilizada por diferentes vistas, evitando la duplicación de código.

- **Facilidad de pruebas y depuración.** La estructura modular del MVC facilita el desarrollo de pruebas unitarias y de integración.
- **Flexibilidad en el desarrollo.** El patrón MVC permite que diferentes equipos trabajen en paralelo en distintos componentes de la aplicación.

Implementación del Patrón MVC en la Aplicación

- **Modelo.** Encapsula la lógica de negocio y gestiona el acceso a los datos relacionados con los exámenes, preguntas, respuestas y usuarios.
- **Vista.** Compuesta por las interfaces de usuario para administradores y usuarios, presenta la información de manera clara y accesible.
- **Controlador.** Maneja las solicitudes del usuario, procesa las entradas y coordina las respuestas adecuadas entre el modelo y la vista.

Diagramas de Diseño del Sistema

Duagrama de Flujo. Es una representación gráfica de un proceso, sistema o algoritmo, que muestra las acciones o pasos a seguir y el orden lógico en que ocurren.

Diagrama Entidad-Relación (ER). Modela las entidades de la base de datos, sus atributos y las relaciones entre ellas. Las entidades principales son Usuarios, Exámenes, Preguntas, Respuestas y Resultados.

Estructura de la Base de Datos

Modelo Relacional

- Se eligió el modelo relacional por su organización, integridad y escalabilidad.
- Permite representar datos en tablas interrelacionadas mediante claves primarias y foráneas.
- Facilita la realización de consultas complejas y asegura la integridad referencial.

Esquema de Tablas y Relaciones. Las principales tablas son:

- **Users.** id, apellido, email, nombre, password, teléfono, user_name, role_id.

- **Roles:** id, name.
- **Questions:** question_id, content, option1, option2, option3, option4, response, exam_id.
- **Exams:** exam_id, active, description, number_questions, points_max, title, category_id.
- **Categories:** category_id, description, title.

Las relaciones están definidas con claves foráneas, lo que:

- Asegura la integridad de los datos.
- Facilita consultas y mantenimiento.

Diagrama Entidad-Relación (ER)

- Proporciona una representación visual de entidades y relaciones.
- Es útil para entender la estructura y planificar futuras modificaciones.

Normalización

La base de datos cumple con las tres primeras formas normales (1NF, 2NF, 3NF):

- **1NF.** Cada campo contiene valores atómicos, sin grupos repetitivos.
- **2NF.** Todos los atributos no clave dependen completamente de la clave primaria.
- **3NF.** No hay dependencias transitivas entre atributos no clave.

Esto reduce la redundancia, mejora la integridad de datos y facilita el mantenimiento.

Decisiones de Diseño Significativas en Aplicaciones Web

Patrón de Arquitectura: Modelo-Vista-Controlador (MVC)

- Permite una separación clara de responsabilidades:
 - **Modelo.** lógica de datos.
 - **Vista.** interfaz de usuario.
 - **Controlador.** gestión de interacciones.
- Facilita el mantenimiento, escalabilidad y trabajo en equipo.

Modelo de Base de Datos: Relacional

- Organiza datos en tablas relacionadas mediante claves primarias y foráneas.
- Asegura la consistencia, precisión y facilita consultas complejas.
- Ideal para entidades como usuarios, exámenes, preguntas y categorías.

Gestión de Roles y Permisos

- Se implementó un sistema de roles para controlar el acceso a funcionalidades.
- Cada usuario tiene un rol con privilegios específicos.
- Mejora la seguridad y la administración del sistema.

Selección de Tecnologías

Tecnologías utilizadas:

- ***Spring Boot***. Framework en Java para crear aplicaciones listas para producción, con configuración automática y menor código repetitivo.
- ***Node.js***. Entorno de ejecución de JavaScript eficiente para aplicaciones escalables y en tiempo real.
- ***HTML, CSS y JavaScript***. Para construir interfaces de usuario responsivas y dinámicas.

Beneficios:

- Arquitectura modular y eficiente.
- Integración entre frontend y backend mediante APIs RESTful.
- Buena escalabilidad y rendimiento.
- Amplia comunidad de soporte.

Estrategia de Pruebas




- Se utilizó Postman para probar las APIs.
- Permitió verificar funcionalidades y detectar errores.
- Aunque no hubo pruebas unitarias, se logró una buena estabilidad general.

Implementación de Seguridad y Autenticación

- Se implementaron mecanismos de autenticación y autorización.
- Verificación de identidad y control de accesos por roles.
- Garantiza que solo usuarios autorizados accedan a funcionalidades sensibles, protegiendo datos e integridad del sistema.

Interfaz de Usuario (UI)

Paleta de Colores. La aplicación implementa una paleta de colores cuidadosamente seleccionada para garantizar una experiencia visual coherente y agradable. Los colores utilizados son los siguientes:

#3f51b5: Utilizado en la barra de navegación, botones primarios e iconos.	
#ffffff: Aplicado al texto sobre fondos oscuros y como color de fondo principal.	
#000000: Empleado en subtítulos (etiquetas h2, h3, etc.) y texto sobre fondos claros.	
#035096: Asignado a los títulos principales (etiqueta h1), hover.	
#ff4081: Designado para botones de acción específicos.	
#f44336: Reservado para botones de advertencia o acciones destructivas.	

Esta selección cromática sigue principios de diseño que promueven la claridad y la jerarquía visual en la interfaz de usuario [.kitdigital.uc.cl](http://kitdigital.uc.cl)

Tipografía. Se ha elegido la tipografía **Roboto** por su diseño limpio y legible, características que la hacen ideal para interfaces digitales . Para enfatizar títulos y subtítulos, se utiliza la variante en negrita de esta fuente, lo que contribuye a una mejor organización visual del contenido.aguayo.co

Organización de Elementos. Los elementos de la interfaz están dispuestos de manera organizada y atractiva, utilizando contenedores con diseños de cajas que facilitan la separación y agrupación de componentes relacionados. Además, se ha prestado especial atención a la selección de tamaños de texto adecuados para cada nivel de información, lo que mejora la legibilidad y la experiencia del usuario.

Retroalimentación de Usuarios. Se ha incorporado la retroalimentación de usuarios para mejorar la accesibilidad de la aplicación. Este enfoque participativo ha permitido identificar áreas de mejora y adaptar la interfaz a las necesidades y preferencias de los usuarios, promoviendo una experiencia más inclusiva y satisfactoria.

Conclusión metodología.

El proceso metodológico aplicado en el desarrollo de la aplicación web ha seguido una estructura coherente basada en buenas prácticas de ingeniería de software y diseño centrado en el usuario. La adopción del patrón arquitectónico Modelo-Vista-Controlador (MVC) fue fundamental para organizar el proyecto en módulos independientes, promoviendo así la escalabilidad y el mantenimiento del sistema (Vargas, 2023; Frexus, 2023).

La utilización de una base de datos relacional permitió una estructuración clara y eficiente de la información, garantizando la integridad referencial mediante claves primarias y foráneas (Oracle, s.f.; Google Cloud, s.f.). Esto resultó esencial en un entorno donde el registro correcto de exámenes y respuestas era crítico para la funcionalidad del sistema.

La gestión de roles y permisos se implementó como una estrategia de seguridad fundamental, asegurando que los usuarios tuvieran acceso únicamente a las funciones correspondientes a su perfil. Asimismo, la elección de tecnologías como Spring Framework para el backend y Node.js para ciertos componentes del frontend proporcionó robustez, modularidad y compatibilidad en el entorno de desarrollo (Spring.io, s.f.; Node.js, s.f.).

Respecto a las pruebas, se utilizó Postman como herramienta de verificación de endpoints, permitiendo testear la correcta funcionalidad del sistema aunque no se desarrollaron pruebas unitarias. Esto permitió comprobar la estabilidad general y la interacción efectiva entre el cliente y el servidor (Postman, s.f.).

Desde el punto de vista de diseño, se optó por una paleta de colores bien definida y una tipografía moderna (Roboto), aplicadas con criterios de jerarquía visual, coherencia y contraste, contribuyendo así a una experiencia de usuario clara y estética. La disposición de los elementos en contenedores, la selección adecuada de tamaños de texto, y la retroalimentación continua por parte de los usuarios permitieron perfeccionar la usabilidad y accesibilidad de la aplicación (Google Fonts, s.f.; Ambit Iberia, 2023).

En conclusión, esta metodología permitió desarrollar una aplicación funcional, segura, estética y preparada para su escalamiento. Las decisiones tomadas durante el diseño, implementación y validación han establecido una base sólida para el crecimiento futuro del sistema.

Desarrollo e implementación

Metodología de Desarrollo

Enfoque Iterativo. En el desarrollo de la aplicación se adoptó una metodología iterativa, caracterizada por ciclos repetitivos de planificación, desarrollo y evaluación. Este enfoque se eligió debido a la necesidad de incorporar cambios y mejoras de forma continua,

lo cual permitió que la aplicación evolucionara permanentemente conforme se iban validando nuevas funcionalidades.

Ventajas del Enfoque Iterativo. La metodología iterativa facilitó la organización del proyecto al dividirlo en fases enfocadas —planificación, desarrollo y prueba—, lo cual agilizó la gestión de tareas y permitió concentrarse en objetivos concretos en cada ciclo. Además, esta forma de trabajo favoreció la detección temprana de errores y la incorporación de mejoras progresivas, incrementando la calidad del producto final.

Aplicación Práctica de la Metodología

Planificación

- Se elaboraron diagramas de flujo y diagramas entidad-relación para definir la estructura y los procesos clave de la aplicación.
- Se especificaron las herramientas a emplear en cada módulo (por ejemplo, Spring Security y JWT para la autenticación en el backend).

Desarrollo Backend

- Se implementó primero el módulo de login y registro de usuarios.
- Se crearon las clases Usuario y Rol, así como los servicios REST que exponen sus operaciones.
- Se configuró la seguridad mediante Spring Security y la generación/validación de tokens JWT.
- Se preparó la base de datos MySQL y se implementaron las entidades JPA correspondientes.

Desarrollo Frontend

- Se diseñó la interfaz de login y registro en Angular, cuidando la usabilidad y conectividad con el backend.

- Se construyeron servicios y componentes que consumen las APIs de autenticación.
- Se probó la comunicación mediante la consola del navegador y se ajustaron los errores de integración.

Pruebas y Depuración

- Se utilizó Postman para testear cada endpoint y validar que las respuestas cumplieran los requisitos funcionales.
- Se corrigieron los errores detectados en el backend y, simultáneamente, se ajustaron los componentes frontales según la retroalimentación de las pruebas.

Control de Versiones. Para apoyar este enfoque iterativo se empleó GitHub como herramienta de control de versiones. Esta plataforma permitió mantener un historial detallado de todos los cambios, facilitando la recuperación ante errores y proporcionando una visión clara de la evolución del proyecto.

Fases del desarrollo

Desarrollo del Módulo de Autenticación

Backend

Configuración de Persistencia. Application.properties contiene los parámetros de conexión a MySQL (URL, credenciales, driver). Esta clase de configuración centraliza toda la información de datasource, de forma que Spring Boot levanta automáticamente la conexión al arrancar la aplicación.

Entidades JPA

User. representa al usuario de la aplicación con atributos como id, userName, password, email, telefono, y la referencia a su rol.

Role:.define los distintos roles (ADMIN, USER) con un identificador y nombre.

Ambas clases están anotadas con `@Entity` y enlazadas mediante `@ManyToOne/@OneToMany`, modelando la relación entre usuarios y roles.

Inicialización del Administrador. En la clase de arranque (por ejemplo, `SecurityApplication`), se instancia un objeto `User` con rol `ADMIN` y se persiste. Así se gana control total sobre el primer administrador sin exponer endpoint alguno.

Repositorios JPA

RoleRepository. interfaz análoga para rol.

Estas clases abstraen completamente el manejo de la base de datos y ofrecen un punto único de acceso.

Servicios de Negocio

UserService. clase anotada con `@Service` que agrupa la lógica relacionada con usuarios y roles. Inyecta los repositorios y actúa como capa intermedia entre controladores y repositorios.

AuthService. gestiona la orquestación del proceso de login y registro, delegando en `UserService` y en las utilidades de JWT.

Seguridad y Filtros JWT

JwtUtil. clase utilitaria que genera y valida tokens JWT.

JwtAuthenticationFilter* y *JwtEntryPoint. filtros que interceptan las peticiones HTTP, extraen el token de la cabecera y validan su autenticidad antes de permitir el acceso.

SecurityConfig. configuración central de seguridad, define qué rutas son públicas y añade los filtros JWT a la cadena de seguridad de Spring.

Controlador REST

AuthController: clase anotada con `@RestController` que expone los endpoints de autenticación (`/auth/login`, `/auth/register`, `/auth/current-user`). Se conecta con `AuthService` y devuelve respuestas en formato JSON sin exponer la implementación interna.

Pruebas en Postman. Aunque no es una clase, la colección de Postman documenta cada endpoint. Estas pruebas confirman el correcto flujo de datos a través de las clases anteriores.

Frontend

RegisterComponent. Clase componente que agrupa el formulario de registro. Define un modelo de datos (correspondiente a `NewUserDto`) y conecta con `AuthService` para enviar la información al backend. La plantilla HTML y el archivo de estilos CSS se asocian a esta clase.

LoginComponent. Clase componente responsable de la pantalla de inicio de sesión. Mantiene el modelo de credenciales (`LoginUserDto`), invoca `AuthService` para obtener el token y, según el rol, navega a la vista de administrador o usuario.

AuthService. Servicio inyectable (`@Injectable`) que centraliza todas las llamadas HTTP (`HttpClient`) a los endpoints de autenticación. Gestiona el almacenamiento del token en `localStorage` y la extracción de información de usuario/rol del token.

AppRoutingModule. Clase de configuración de rutas que define el enrutamiento de la aplicación. Usa guards (por ejemplo, `AuthGuard`) para proteger las rutas que requieren autenticación, basándose en la presencia y validez del token.

AuthGuard. Servicio que implementa `CanActivate`, comprobando antes de cargar una ruta si el token existe y es válido (consultando `AuthService`). Si no, redirige al componente de login.

TokenInterceptor. Clase que implementa `HttpInterceptor` y se registra en el módulo principal. Añade automáticamente el token JWT a la cabecera de cada petición HTTP saliente, garantizando que solo las solicitudes autorizadas alcancen los endpoints protegidos.

Diseño. Se utiliza varias librerías para ayudar al diseño al de los formularios de Login y Registro, dichas librerías serían Angular material, Bootstrap

Desarrollo del Módulo de Navegación y Menús

Frontend

NavbarComponent. Se creó el componente NavbarComponent, encargado de la barra de navegación superior de la aplicación. Este componente incluye:

- Un botón para desplegar el menú lateral (sidebar).
- El nombre y logotipo de la aplicación.
- Una opción para cerrar sesión.

Este componente proporciona una interfaz consistente en la parte superior de la aplicación, facilitando la navegación y el acceso a funciones comunes.

Estructura de Módulos por Rol. Se crearon dos módulos principales para organizar los componentes según el rol del usuario:

- ***AdminModul.*** contiene los componentes y vistas específicas para usuarios con rol de administrador.
- ***UserModule.*** incluye los componentes y vistas destinadas a los usuarios estándar.

Esta separación modular permite una gestión más clara y escalable de las funcionalidades según los diferentes perfiles de usuario.

SidebarComponent para Administrador. Dentro del AdminModule, se desarrolló el componente SidebarComponent, que representa el menú lateral de navegación para administradores. Este componente incluye enlaces a las siguientes secciones:

- Inicio
- Perfil
- Categorías
- Agregar Categoría
- Cuestionarios

- Agregar Cuestionario
- Salir

El SidebarComponent utiliza el sistema de enrutamiento de Angular para navegar entre las diferentes vistas del panel de administración.

SidebarComponent para Usuario. De manera similar, en el UserModule, se implementó un SidebarComponent adaptado a las necesidades de los usuarios estándar. Este componente proporciona acceso a:

- Categorías disponibles
- Cuestionarios asignadoses.

Al igual que en el caso del administrador, este componente facilita la navegación mediante el enrutamiento de Angular, asegurando una experiencia de usuario coherente y eficiente.

Integración y Navegación. Los componentes NavbarComponent y SidebarComponent se integran en las vistas principales de la aplicación, proporcionando una estructura de navegación clara y accesible. La combinación de estos componentes permite a los usuarios interactuar con la aplicación de manera intuitiva, accediendo rápidamente a las diferentes funcionalidades según su rol.

Desarrollo de la Página de Detalles de Usuario

Frontend

ProfileComponent. Se creó el componente ProfileComponent, diseñado para ser utilizado por ambos roles de usuario (administrador y usuario estándar). Este enfoque unificado evita duplicación de código y facilita el mantenimiento.

Obtención de Datos del Usuario Actual. En el archivo TypeScript (profile.component.ts), se implementó la lógica necesaria para obtener la información del usuario actualmente autenticado. Esto se logró mediante la inyección de un servicio de

autenticación que proporciona los datos del usuario, como nombre, apellido, nombre de usuario, correo electrónico, teléfono y rol.

Visualización de Detalles en la Plantilla HTML La plantilla HTML (profile.component.html) se diseñó para mostrar de manera clara y organizada los detalles del usuario. Se utilizaron componentes y clases de Angular Material y Bootstrap para estructurar y estilizar la información, asegurando una presentación visual atractiva y coherente con el resto de la aplicación.

Estilización con Angular Material y Bootstrap. Para mejorar la apariencia y la experiencia del usuario, se integraron estilos y componentes de Angular Material y Bootstrap. Esta combinación permitió una interfaz moderna y responsiva, facilitando la lectura y navegación en diferentes dispositivos.

Diseño de Entidades, Servicios, Repositorios y Controladores en el Backend

Entidades

Entidad Exam. La clase Exam representa un examen dentro del sistema y está mapeada a una tabla en la base de datos mediante la anotación @Entity. Esta entidad incluye atributos como examId, title, description, pointsMax, numberQuestions y active, que definen las propiedades básicas de un examen.

Además, establece relaciones con otras entidades:

- ***Relación con Category.*** Cada examen pertenece a una categoría específica, representada mediante una relación @ManyToOne con la entidad Category.
- ***Relación con Question.*** Un examen puede contener múltiples preguntas, lo que se modela con una relación @OneToMany hacia la entidad Question.

Estas relaciones permiten una navegación eficiente entre exámenes, sus categorías y preguntas asociadas, facilitando operaciones como la recuperación de todas las preguntas de un examen o la agrupación de exámenes por categoría.

Entidad Category. La clase Category define las categorías a las que pueden pertenecer los exámenes. Está mapeada a una tabla en la base de datos mediante la anotación @Entity y contiene atributos como categoryId, title y description.

Establece una relación @OneToMany con la entidad Exam, indicando que una categoría puede tener múltiples exámenes asociados. Esta relación inversa permite acceder fácilmente a todos los exámenes que pertenecen a una categoría específica.

Entidad Question. La clase Question representa una pregunta individual dentro de un examen y está mapeada a una tabla en la base de datos mediante la anotación @Entity. Incluye atributos como questionId, content, option1 a option4, response y responseGiven, que definen el contenido de la pregunta, las opciones disponibles y las respuestas.

Establece una relación @ManyToOne con la entidad Exam, indicando que cada pregunta pertenece a un examen específico. Esta relación permite agrupar preguntas bajo un examen y facilita operaciones como la evaluación de respuestas y la generación de cuestionarios.

Repositorios y Servicios. Para cada una de las entidades (Exam, Category y Question), se crearon interfaces de repositorio que extienden JpaRepository, proporcionando métodos para realizar operaciones CRUD y consultas personalizadas.

Además, se implementaron servicios que encapsulan la lógica de negocio relacionada con cada entidad, facilitando la interacción entre los controladores y los repositorios. Estos servicios gestionan operaciones como la creación, actualización, recuperación y eliminación de registros, así como la ejecución de consultas específicas basadas en las relaciones entre entidades.

Estructura y Funcionamiento de los Controladores

Controlador de Exámenes (ExamController). La clase ExamController está anotada con @RestController, lo que indica que es un componente de Spring encargado de manejar

solicitudes HTTP y devolver respuestas en formato JSON. Esta clase actúa como intermediario entre el cliente (frontend) y la lógica de negocio relacionada con los exámenes.

El controlador se comunica con el servicio correspondiente para realizar operaciones como la creación, actualización, recuperación y eliminación de exámenes. Además, facilita la obtención de exámenes filtrados por categoría o estado (activos/inactivos), lo que permite al frontend presentar información relevante según el contexto del usuario.

Controlador de Categorías (CategoryController). Similar al controlador de exámenes, la clase CategoryController está anotada con @RestController y se encarga de manejar las solicitudes relacionadas con las categorías de exámenes. Su función principal es servir como punto de entrada para las operaciones CRUD sobre las categorías, delegando la lógica de negocio al servicio correspondiente.

Este diseño permite una gestión eficiente de las categorías, facilitando al frontend la presentación y organización de los exámenes según su clasificación.

Controlador de Preguntas (QuestionController). La clase QuestionController, también anotada con @RestController, maneja las solicitudes relacionadas con las preguntas de los exámenes. Además de las operaciones CRUD estándar, este controlador incluye funcionalidades específicas que son fundamentales para la evaluación de los exámenes.

Una de las funciones clave de este controlador es la evaluación de exámenes, que permite comparar las respuestas proporcionadas por los usuarios con las respuestas correctas almacenadas en el sistema. Esta funcionalidad es esencial para calcular puntuaciones y proporcionar retroalimentación inmediata a los usuarios, y será de gran utilidad en la implementación del frontend.

Gestión de Categorías en el Panel de Administración (Frontend)

Servicio de Categorías (CategoryService). Se implementó un servicio dedicado a la gestión de categorías, el cual actúa como intermediario entre el frontend y el backend. Este

servicio utiliza el módulo `HttpClient` de Angular para realizar solicitudes HTTP a los endpoints correspondientes del backend. Las principales responsabilidades de este servicio incluyen:

- Obtener la lista de categorías disponibles.
- Agregar una nueva categoría.
- Actualizar una categoría existente.
- Eliminar una categoría específica.

Este enfoque centraliza la lógica de comunicación con el backend, promoviendo la reutilización de código y facilitando el mantenimiento.

Componente de Visualización de Categorías (`ViewCategoriesComponent`). Este componente es responsable de presentar la lista de categorías al administrador y proporcionar opciones para editar o eliminar cada una de ellas. Su estructura y funcionamiento se detallan a continuación:

- ***Archivo TypeScript (`view-categories.component.ts`):***
 - Define propiedades para almacenar la lista de categorías y gestionar el estado de la interfaz.
 - Implementa métodos que, al ser invocados desde la plantilla HTML, permiten:
 - Obtener y mostrar la lista actualizada de categorías.
 - Eliminar una categoría específica, mostrando una ventana de confirmación utilizando la biblioteca `SweetAlert2` (`Swal`).
 - Editar una categoría existente, abriendo una ventana emergente para modificar sus detalles.
- ***Archivo HTML (`view-categories.component.html`):***
 - Utiliza directivas de Angular para iterar sobre la lista de categorías y mostrar cada una en una tarjeta o fila, incluyendo su nombre y descripción.

- Incluye botones para editar y eliminar cada categoría, los cuales están vinculados a los métodos correspondientes en el archivo TypeScript.
- Aplica estilos y componentes de Angular Material y Bootstrap para lograr una interfaz atractiva y responsiva.

Componente de Agregado de Categorías (*AddCategoryComponent*). Este componente proporciona un formulario para que el administrador pueda agregar nuevas categorías al sistema. Su estructura y funcionamiento se describen a continuación:

- **Archivo TypeScript (*add-category.component.ts*):**
 - Define un modelo para representar los datos de la nueva categoría.
 - Implementa un método que, al ser invocado desde la plantilla HTML, envía los datos del formulario al servicio de categorías para agregar la nueva categoría.
 - Utiliza Swal para mostrar una ventana emergente que confirma la adición exitosa de la categoría o informa sobre posibles errores.
- **Archivo HTML (*add-category.component.html*):**
 - Contiene un formulario con campos para ingresar el título y la descripción de la nueva categoría.
 - Incluye un botón de "Guardar" que, al ser presionado, invoca el método correspondiente en el archivo TypeScript para procesar la adición.
 - Aplica estilos y componentes de Angular Material y Bootstrap para mantener la coherencia visual con el resto de la aplicación.

Gestión de Exámenes en el Panel de Administración (Frontend)

Servicio de Exámenes (*ExamService*). Se implementó un servicio dedicado a la gestión de exámenes, el cual actúa como intermediario entre el frontend y el backend. Este servicio utiliza el módulo HttpClient de Angular para realizar solicitudes HTTP a los

endpoints correspondientes del backend. Las principales responsabilidades de este servicio incluyen:

- Obtener la lista de exámenes disponibles.
- Agregar un nuevo examen.
- Actualizar un examen existente.
- Eliminar un examen específico. [YouTube+2Stack Overflow+2YouTube+2](#)

Este enfoque centraliza la lógica de comunicación con el backend, promoviendo la reutilización de código y facilitando el mantenimiento.

Componente de Visualización de Exámenes (ViewExamsComponent). Este componente es responsable de presentar la lista de exámenes al administrador y proporcionar opciones para editar o eliminar cada uno de ellos. Su estructura y funcionamiento se detallan a continuación:

- ***Archivo TypeScript (view-exams.component.ts):***
 - Define propiedades para almacenar la lista de exámenes y gestionar el estado de la interfaz.
 - Implementa métodos que, al ser invocados desde la plantilla HTML, permiten:
 - Obtener y mostrar la lista actualizada de exámenes.
 - Eliminar un examen específico, mostrando una ventana de confirmación utilizando la biblioteca SweetAlert2 (Swal).
- ***Archivo HTML (view-exams.component.html):***
 - Utiliza directivas de Angular para iterar sobre la lista de exámenes y mostrar cada uno en una tarjeta o fila, incluyendo su título, categoría asociada, número de preguntas y puntos máximos.
 - Incluye botones para editar y eliminar cada examen, los cuales están vinculados a los métodos correspondientes en el archivo TypeScript.

- Aplica estilos y componentes de Angular Material y Bootstrap para lograr una interfaz atractiva y responsiva.

Componente de Agregado de Exámenes (AddExamComponent). Este componente proporciona un formulario para que el administrador pueda agregar nuevos exámenes al sistema. Su estructura y funcionamiento se describen a continuación:

- **Archivo TypeScript (add-exam.component.ts):**
 - Define un modelo para representar los datos del nuevo examen.
 - Implementa métodos que, al ser invocados desde la plantilla HTML, permiten: YouTube
 - Obtener la lista de categorías disponibles para asociar al examen.
 - Enviar los datos del formulario al servicio de exámenes para agregar el nuevo examen.
 - Utiliza Swal para mostrar una ventana emergente que confirma la adición exitosa del examen o informa sobre posibles errores.
- **Archivo HTML (add-exam.component.html):**
 - Contiene un formulario con campos para ingresar el título, descripción, puntos máximos, número de preguntas, estado de publicación y selección de categoría del nuevo examen.
 - Incluye un botón de "Guardar" que, al ser presionado, invoca el método correspondiente en el archivo TypeScript para procesar la adición.
 - Aplica estilos y componentes de Angular Material y Bootstrap para mantener la coherencia visual con el resto de la aplicación.

Componente de Actualización de Exámenes (UpdateExamComponent). Este componente permite al administrador modificar los detalles de un examen existente. Su estructura y funcionamiento se describen a continuación:

- **Archivo TypeScript (*update-exam.component.ts*):**
 - Define propiedades para almacenar los datos del examen a actualizar y la lista de categorías disponibles.
 - Implementa métodos que, al ser invocados desde la plantilla HTML, permiten:
 - Obtener los datos del examen seleccionado para prellenar el formulario.
 - Obtener la lista de categorías disponibles.
 - Enviar los datos actualizados al servicio de exámenes para modificar el examen existente.
 - Utiliza Swal para mostrar una ventana emergente que confirma la actualización exitosa del examen o informa sobre posibles errores.
- **Archivo HTML (*update-exam.component.html*):**
 - Contiene un formulario prellenado con los datos actuales del examen, permitiendo al administrador modificar el título, descripción, puntos máximos, número de preguntas, estado de publicación y categoría asociada.
 - Incluye un botón de "Guardar" que, al ser presionado, invoca el método correspondiente en el archivo TypeScript para procesar la actualización.
 - Aplica estilos y componentes de Angular Material y Bootstrap para mantener la coherencia visual con el resto de la aplicación.

Gestión de Preguntas en el Panel de Administración (Frontend)

Servicio de Preguntas (*QuestionService*). Se implementó un servicio dedicado a la gestión de preguntas, el cual actúa como intermediario entre el frontend y el backend. Este servicio utiliza el módulo HttpClient de Angular para realizar solicitudes HTTP a los endpoints correspondientes del backend. Las principales responsabilidades de este servicio incluyen:

- Obtener la lista de preguntas asociadas a un examen específico.
- Agregar una nueva pregunta.
- Actualizar una pregunta existente.
- Eliminar una pregunta específica.
- Evaluar un examen completo, enviando las respuestas proporcionadas por el usuario y recibiendo el puntaje obtenido.

Este enfoque centraliza la lógica de comunicación con el backend, promoviendo la reutilización de código y facilitando el mantenimiento.

Componente de Visualización de Preguntas (ViewExamQuestionsComponent). Este componente es responsable de presentar la lista de preguntas asociadas a un examen específico y proporcionar opciones para editar o eliminar cada una de ellas. Su estructura y funcionamiento se detallan a continuación:

- ***Archivo TypeScript (view-exam-questions.component.ts):***
 - Define propiedades para almacenar la lista de preguntas y gestionar el estado de la interfaz.
 - Implementa métodos que, al ser invocados desde la plantilla HTML, permiten:
 - Obtener y mostrar la lista actualizada de preguntas asociadas a un examen.
 - Eliminar una pregunta específica, mostrando una ventana de confirmación utilizando la biblioteca SweetAlert2 (Swal).
- ***Archivo HTML (view-exam-questions.component.html):***
 - Utiliza directivas de Angular para iterar sobre la lista de preguntas y mostrar cada una en una tarjeta o fila, incluyendo su contenido, opciones, respuesta correcta y botones para editar o eliminar.

- Aplica estilos y componentes de Angular Material y Bootstrap para lograr una interfaz atractiva y responsiva.

Componente de Agregado de Preguntas (AddQuestionComponent). Este componente proporciona un formulario para que el administrador pueda agregar nuevas preguntas a un examen específico. Su estructura y funcionamiento se describen a continuación:

- ***Archivo TypeScript (add-question.component.ts):***
 - Define un modelo para representar los datos de la nueva pregunta.
 - Implementa métodos que, al ser invocados desde la plantilla HTML, permiten:
 - Validar que todos los campos del formulario estén completos antes de enviar los datos.
 - Enviar los datos del formulario al servicio de preguntas para agregar la nueva pregunta.
 - Utiliza Swal para mostrar una ventana emergente que confirma la adición exitosa de la pregunta o informa sobre posibles errores.
- ***Archivo HTML (add-question.component.html):***
 - Contiene un formulario con campos para ingresar el contenido de la pregunta, las cuatro opciones de respuesta y la selección de la respuesta correcta.
 - Incluye un botón de "Guardar" que, al ser presionado, invoca el método correspondiente en el archivo TypeScript para procesar la adición.
 - Aplica estilos y componentes de Angular Material y Bootstrap para mantener la coherencia visual con el resto de la aplicación.

Componente de Actualización de Preguntas (UpdateQuestionComponent). Este componente permite al administrador modificar los detalles de una pregunta existente. Su estructura y funcionamiento se describen a continuación:

- **Archivo TypeScript (*update-question.component.ts*):**
 - Define propiedades para almacenar los datos de la pregunta a actualizar.
 - Implementa métodos que, al ser invocados desde la plantilla HTML, permiten:
 - Obtener los datos de la pregunta seleccionada para prellenar el formulario.
 - Enviar los datos actualizados al servicio de preguntas para modificar la pregunta existente.
 - Utiliza Swal para mostrar una ventana emergente que confirma la actualización exitosa de la pregunta o informa sobre posibles errores.
- **Archivo HTML (*update-question.component.html*):**
 - Contiene un formulario prellenado con los datos actuales de la pregunta, permitiendo al administrador modificar el contenido, las opciones de respuesta y la respuesta correcta.
 - Incluye un botón de "Guardar" que, al ser presionado, invoca el método correspondiente en el archivo TypeScript para procesar la actualización.
 - Aplica estilos y componentes de Angular Material y Bootstrap para mantener la coherencia visual con el resto de la aplicación.

Visualización de Exámenes para Usuarios Normales

Componente *load-examen*. Este componente permite a los usuarios visualizar los exámenes disponibles, filtrando únicamente aquellos que están activos. Además, ofrece la posibilidad de listar exámenes por categoría específica.

- **Archivo TypeScript (*load-examen.component.ts*):**
 - Implementa una función para obtener todos los exámenes activos desde el backend, asegurando que solo se muestren aquellos que están disponibles para los usuarios.

- Incluye una función adicional que permite listar exámenes activos pertenecientes a una categoría específica, facilitando la navegación por temas de interés.
- Estas funciones se ejecutan al inicializar el componente, garantizando que la información esté disponible al momento de renderizar la vista.
- **Archivo HTML (*load-examen.component.html*):**
 - Utiliza una directiva *ngIf para verificar la existencia de exámenes antes de mostrarlos, mejorando la experiencia del usuario al evitar listas vacías.
 - Estructura la información de cada examen en una tarjeta que incluye:
 - Título del examen.
 - Descripción breve.
 - Número de preguntas que contiene.
 - Puntuación máxima posible.
 - Un botón que redirige a las instrucciones del examen y permite comenzar su realización.
 - Aplica estilos utilizando Bootstrap, Angular Material y CSS personalizado para lograr una interfaz atractiva y responsiva.

Instrucciones del Examen para Usuarios Normales

Componente instrucciones. Este componente tiene como objetivo presentar al usuario las instrucciones y reglas específicas de un examen antes de su inicio, asegurando que el usuario esté informado y preparado.

- **Archivo TypeScript (*instrucciones.component.ts*):**
 - Implementa un método obtenerExamen() que recupera los detalles del examen seleccionado desde el backend, incluyendo título, descripción, número de intentos permitidos y otras configuraciones relevantes.

- Define un método `empezarExamen()` que, al ser invocado, muestra una ventana emergente de confirmación utilizando `SweetAlert2`. Esta ventana solicita al usuario confirmar si desea comenzar el examen. Si el usuario confirma, se le redirige a la vista de realización del examen.
- **Archivo HTML (*instrucciones.component.html*):**
 - Muestra la siguiente información del examen:
 - Título del examen.
 - Descripción detallada.
 - Instrucciones importantes y reglas que el usuario debe conocer antes de iniciar.
 - Número de intentos permitidos para realizar el examen.
 - Incluye un botón "Empezar" que, al ser clicado, invoca el método `empezarExamen()`.
 - Aplica estilos utilizando Bootstrap, Angular Material y CSS personalizado para lograr una interfaz atractiva y coherente con el resto de la aplicación.

Realización, Evaluación e Impresión de Exámenes para Usuarios Normales

Componente start. Este componente permite a los usuarios realizar un examen, evaluarlo al finalizar y obtener una opción para imprimir los resultados.

- **Archivo TypeScript (*start.component.ts*):**
 - **Prevención de navegación no deseada:** Se implementa una funcionalidad que impide al usuario retroceder o salir del examen accidentalmente, asegurando que el proceso de evaluación no se interrumpa.
 - **Carga de preguntas:** Al iniciar el componente, se cargan todas las preguntas asociadas al examen desde el backend, permitiendo su presentación secuencial al usuario.

- **Temporizador del examen:** Se incorpora un temporizador que limita el tiempo disponible para completar el examen. Este temporizador se visualiza mediante un componente de tipo "progress-spinner", proporcionando una representación gráfica del tiempo restante.
- **Evaluación del examen:** Al finalizar el examen, se recopilan las respuestas del usuario y se envían al backend para su evaluación. El resultado incluye la puntuación obtenida y el número de respuestas correctas.
- **Finalización del examen:** Se implementa una función que permite al usuario finalizar el examen manualmente. Al activarla, se muestra una ventana emergente de confirmación utilizando SweetAlert2. Si el usuario confirma, se procede a la evaluación y presentación de resultados.
- **Impresión de resultados:** Tras la evaluación, se ofrece la opción de imprimir o descargar los resultados del examen, facilitando su almacenamiento o revisión posterior.
- **Archivo HTML (*start.component.html*):**
 - Muestra el título del examen y una serie de instrucciones breves para el usuario.
 - Presenta cada pregunta numerada junto con sus opciones de respuesta.
 - Incluye un temporizador visual que indica el tiempo restante para completar el examen.
 - Proporciona un botón "Terminar examen" que, al ser clicado, invoca la función de finalización y evaluación del examen.
 - Tras la evaluación, se muestra una vista con los resultados obtenidos y dos botones: uno para imprimir los resultados y otro para regresar al inicio.

- Aplica estilos utilizando Bootstrap, Angular Material y CSS personalizado para lograr una interfaz atractiva y coherente con el resto de la aplicación.

Panel de Inicio del Administrador

Componente *welcome*. Este componente actúa como la página de inicio para los administradores, proporcionando accesos directos a las principales secciones de gestión: Categorías, Exámenes y Preguntas.

- **Archivo *TypeScript* (*welcome.component.ts*):**
 - Implementa una función `navegarA(ruta: string)` que utiliza el servicio de enrutamiento de Angular para redirigir al usuario a la sección correspondiente al hacer clic en una tarjeta.

Archivo *HTML* (*welcome.component.html*):

- Presenta una serie de tarjetas interactivas, cada una representando una sección del panel de administración.
- Cada tarjeta incluye:
 - Un título (por ejemplo, "Categorías", "Exámenes", "Preguntas").
 - Una breve descripción de la funcionalidad.
 - Un botón o área clicable que, al interactuar, invoca la función `navegarA` con la ruta correspondiente.

Se aplican estilos utilizando Bootstrap, Angular Material y CSS personalizado para lograr una interfaz atractiva y coherente con el resto de la aplicación.

Página de Inicio Pública

Componente *home*. Este componente sirve como la página de bienvenida para todos los usuarios que acceden a la aplicación sin haber iniciado sesión.

- **Archivo *HTML* (*home.component.html*):**

- Establece una imagen de fondo que cubre toda la pantalla, proporcionando una apariencia atractiva y profesional. SeedProd
- Centra una tarjeta que contiene:
 - El nombre de la aplicación.
 - Una breve descripción que destaca las funcionalidades principales.
 - Dos botones:
 - **Iniciar Sesión:** Redirige al formulario de inicio de sesión.
 - **Registrarse:** Redirige al formulario de registro de nuevos usuarios.
- Aplica estilos utilizando Bootstrap, Angular Material y CSS personalizado para lograr una interfaz atractiva y coherente con el resto de la aplicación.
- **Archivo TypeScript (*home.component.ts*):**
 - Define métodos para manejar las acciones de los botones, utilizando el servicio de enrutamiento de Angular para navegar a las rutas correspondientes.

Desafíos y Soluciones

Generación del Token JWT. Desafío: Durante la implementación de la autenticación basada en tokens JWT, se presentaron dificultades en la generación del token.

Solución: Se consultó la documentación oficial y se implementaron bloques de manejo de excepciones (try-catch) para identificar y capturar los errores específicos en el proceso de generación del token. Esta estrategia permitió aislar y corregir los problemas en cada etapa del proceso.

Conexión entre Login y Registro. Desafío: Inicialmente, se experimentaron problemas al conectar las funcionalidades de inicio de sesión y registro de usuarios, lo que impedía que los usuarios se registraran o iniciaran sesión correctamente.

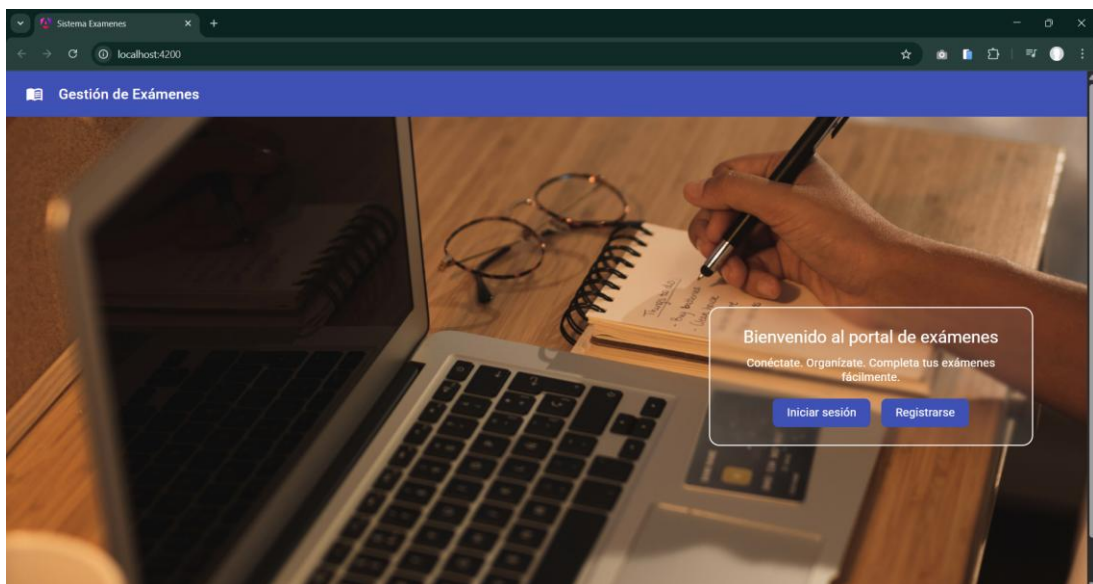
Solución: Se utilizó la herramienta de inspección del navegador para monitorear las solicitudes y respuestas HTTP. Esto permitió identificar errores en las rutas y en la configuración de los encabezados de las solicitudes, los cuales fueron corregidos para establecer una comunicación efectiva entre el frontend y el backend.

Carga de Datos al Iniciar la Aplicación. Desafío: Se detectó que ciertos datos no se cargaban automáticamente al iniciar la aplicación, requiriendo una recarga manual de la página para que aparecieran.

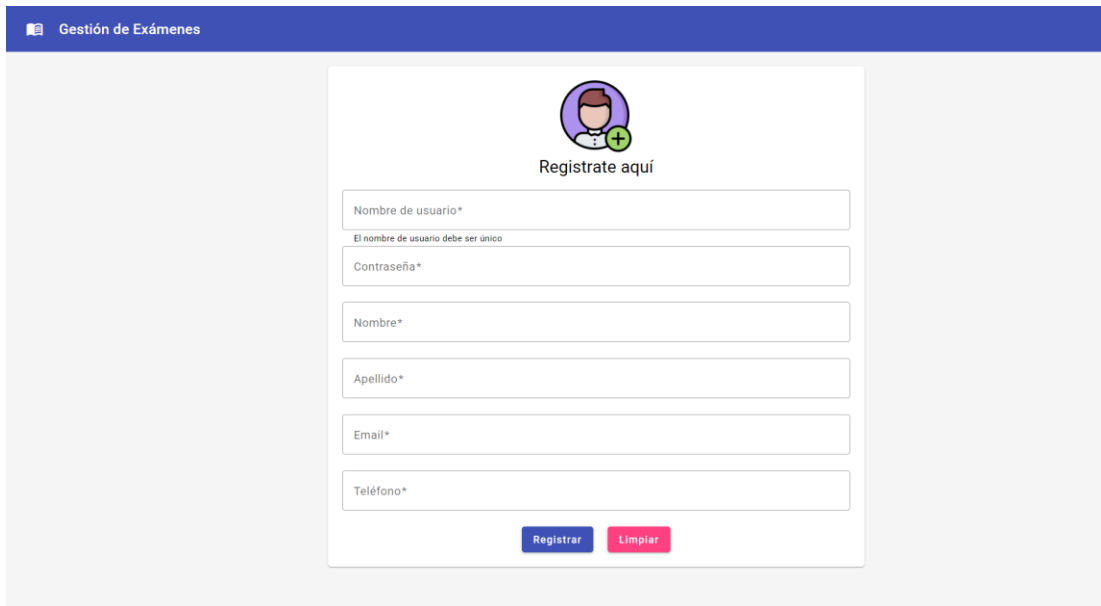
Solución: A través del panel de inspección del navegador, se identificaron errores relacionados con la carga de datos. Se implementaron soluciones como la verificación de la inicialización de los servicios y la correcta suscripción a los observables en los componentes para asegurar que los datos se cargaran adecuadamente al iniciar la aplicación.

Evidencia visual

Figura 1

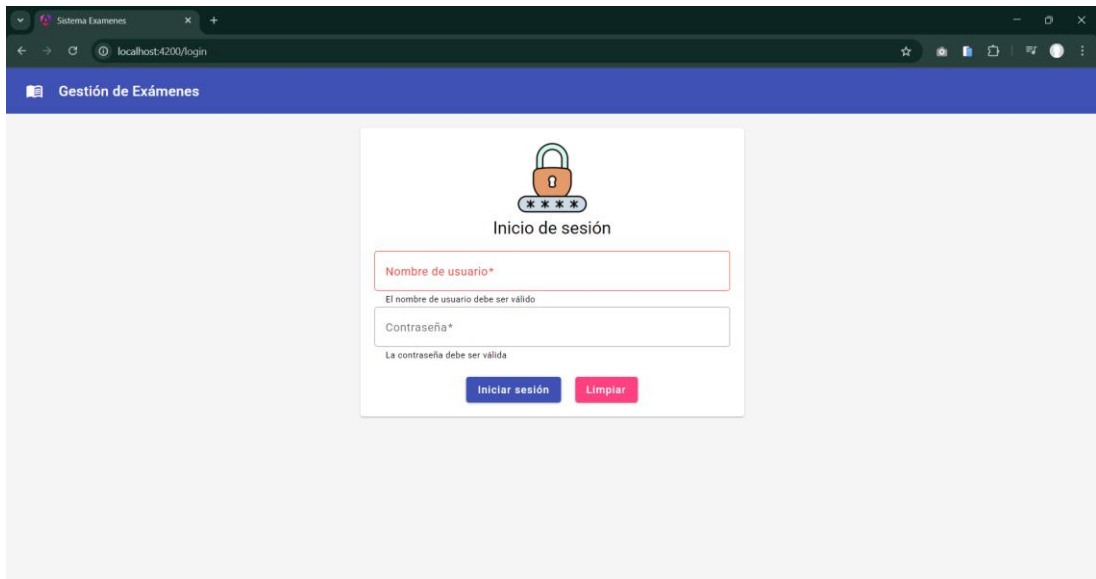


Pantalla principal del sistema de gestión de exámenes con opciones de inicio de sesión y registro.

Figura 2

The screenshot shows a web application titled 'Gestión de Exámenes' with a registration form. The form is titled 'Regístrate aquí' and features a user icon with a plus sign. It contains the following fields: 'Nombre de usuario*' (with a validation message 'El nombre de usuario debe ser único'), 'Contraseña*', 'Nombre*', 'Apellido*', 'Email*', and 'Teléfono*'. At the bottom are 'Registrar' and 'Limpiar' buttons.

Formulario de registro del sistema de gestión de exámenes con campos obligatorios para el alta de nuevos usuarios.

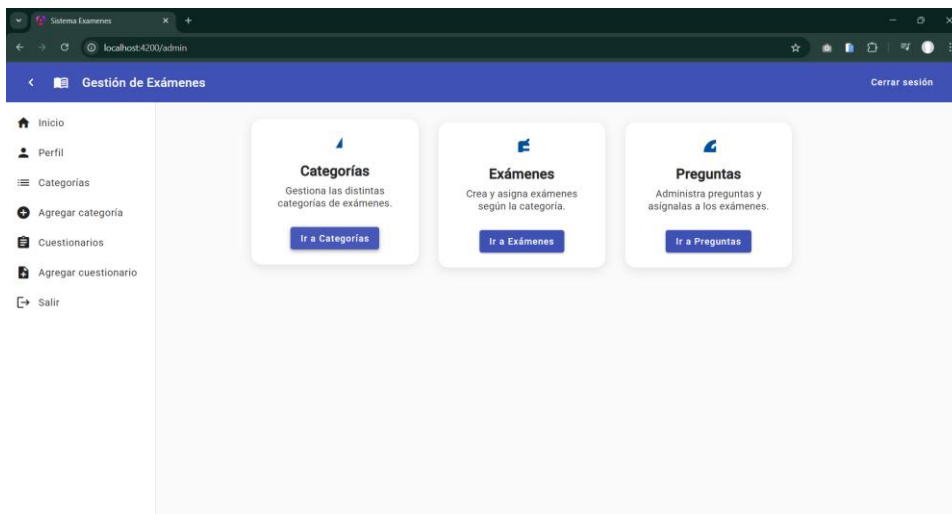
Figura 3

The screenshot shows the same web application with a login form titled 'Inicio de sesión'. It features a padlock icon and two input fields: 'Nombre de usuario*' (with a validation message 'El nombre de usuario debe ser válido') and 'Contraseña*' (with a validation message 'La contraseña debe ser válida'). At the bottom are 'Iniciar sesión' and 'Limpiar' buttons.

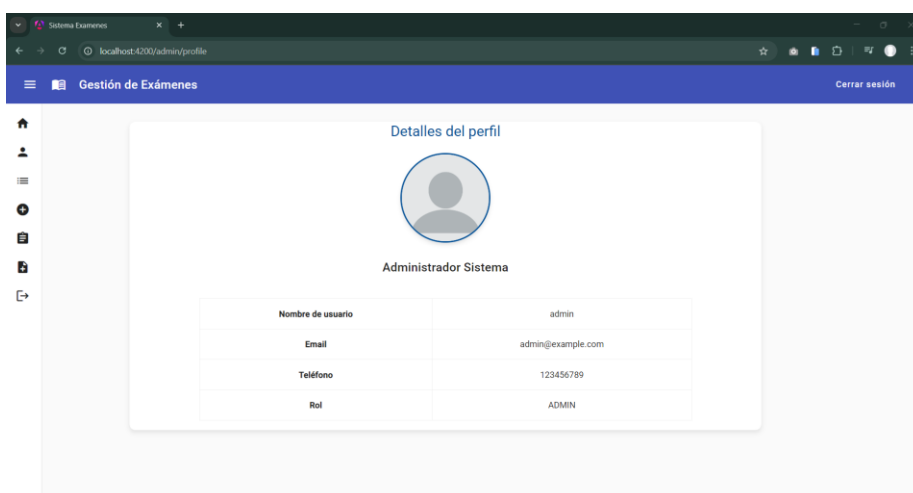
Formulario de inicio de sesión del sistema con validación de campos para nombre de usuario y contraseña.

Figura 4

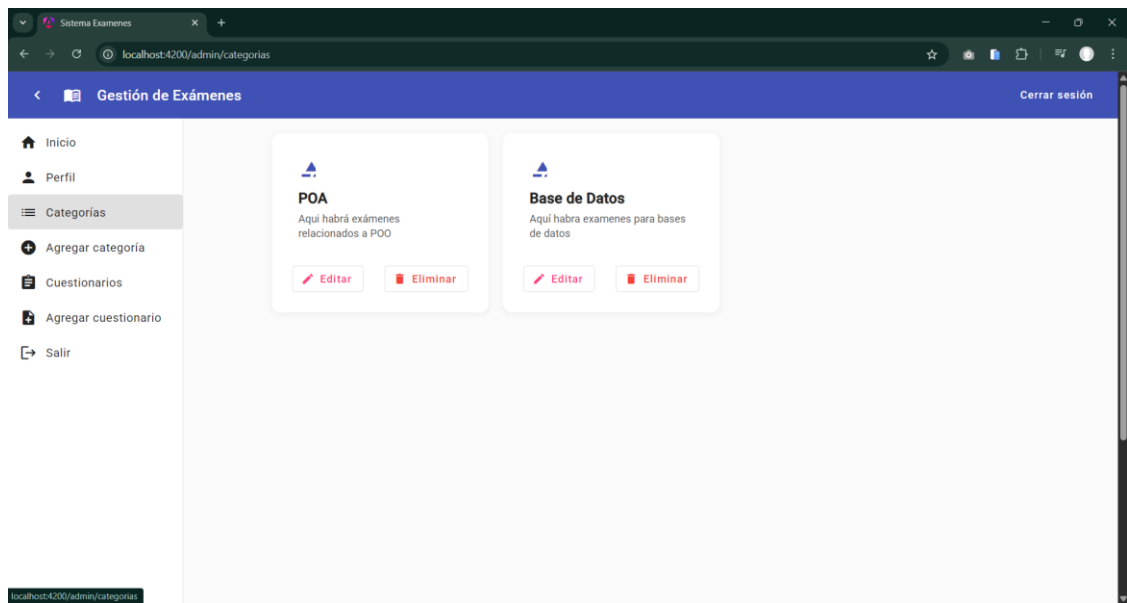
Barra superior del sistema con el título “Gestión de Exámenes”, menú de navegación y opción para cerrar sesión.

Figura 5

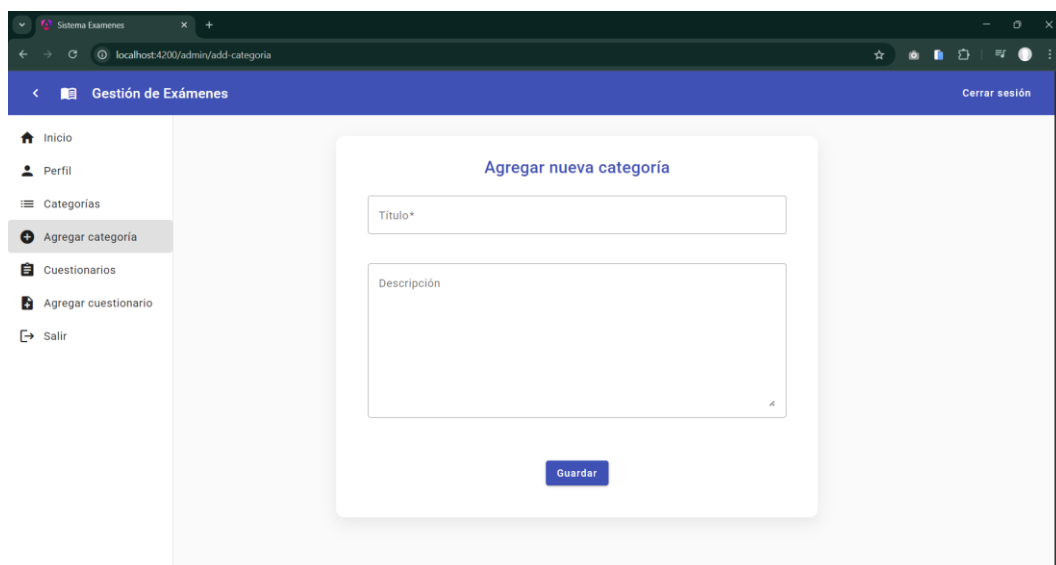
Panel de administración con accesos a categorías, exámenes y preguntas, junto a un menú lateral para navegación entre secciones.

Figura 6

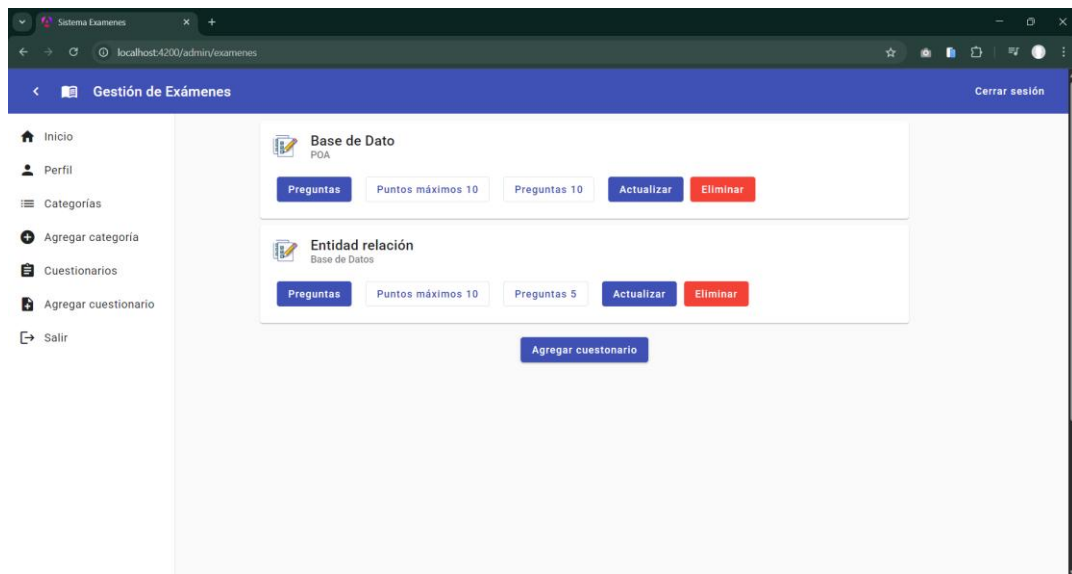
Vista del perfil del administrador con información de usuario, correo, teléfono y rol asignado.

Figura 7

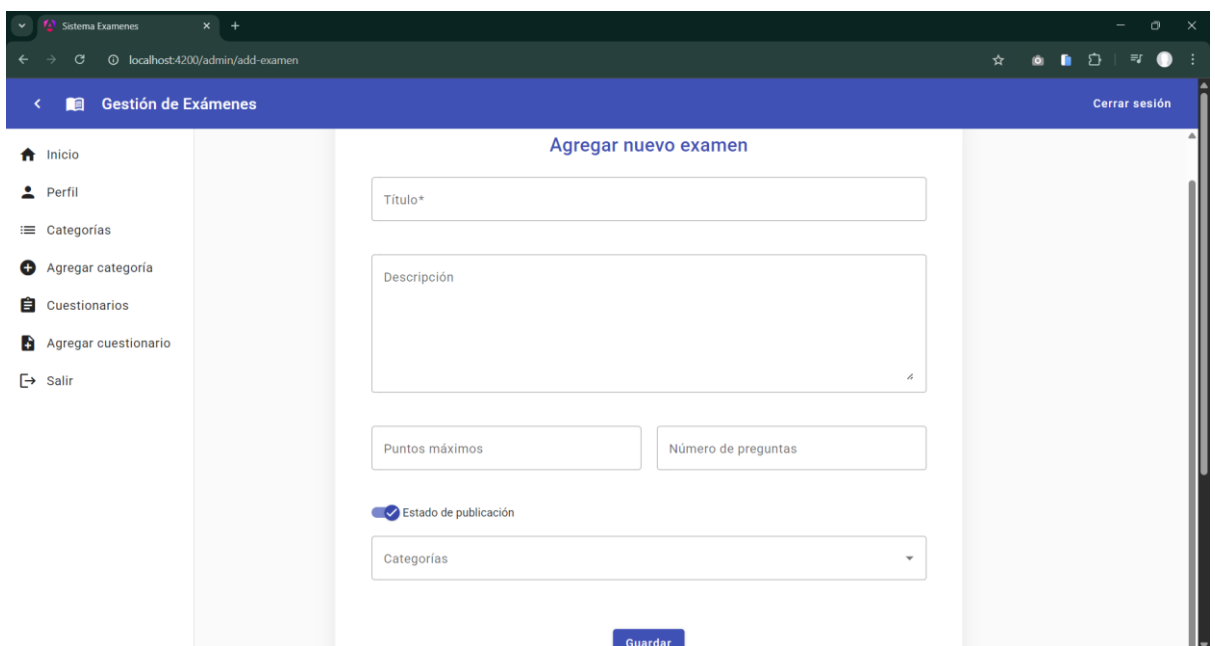
Vista de la sección de categorías del sistema, con opciones para editar o eliminar categorías existentes.

Figura 8

Formulario para agregar una nueva categoría con campos de título y descripción.

Figura 9

Vista de la sección de exámenes, con opciones para administrar preguntas, puntajes y editar o eliminar cuestionarios.

Figura 10

Formulario para agregar un nuevo examen, con campos para título, descripción, puntaje, número de preguntas y categoría.

Figura 11

Actualizar examen

Titulo*
Base de Dato

Descripción
MySQL

Puntos máximos
10

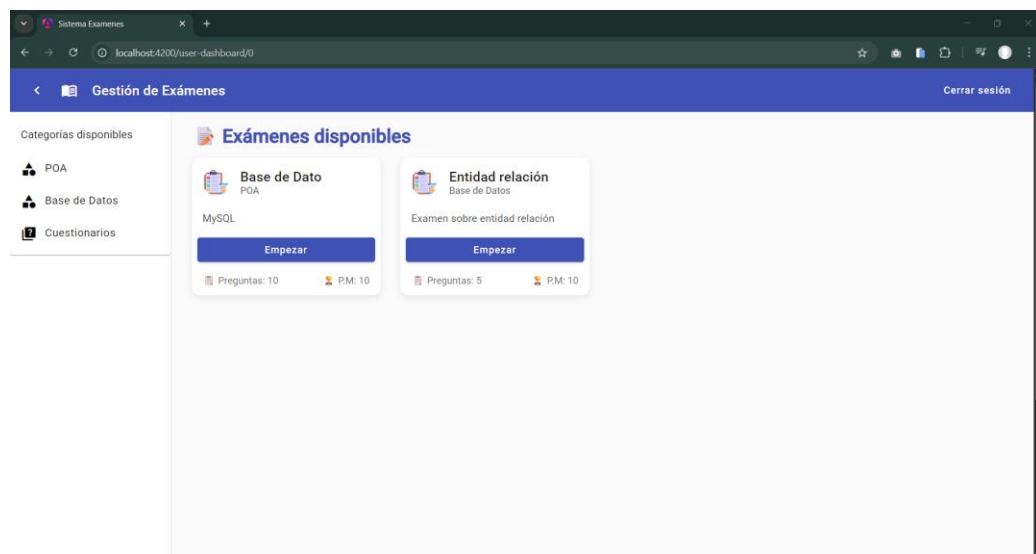
Número de preguntas
10

☒ Estado de publicación

Categorías
POA

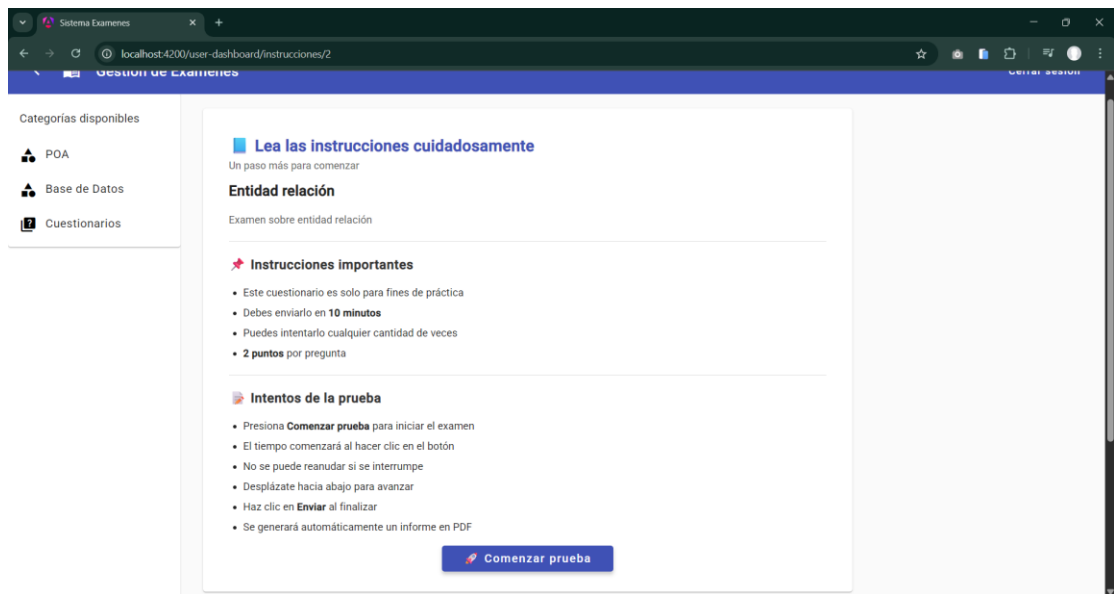
Guardar

Formulario para actualizar un examen, con campos editables como título, descripción, puntaje, número de preguntas y estado de publicación.

Figura 12

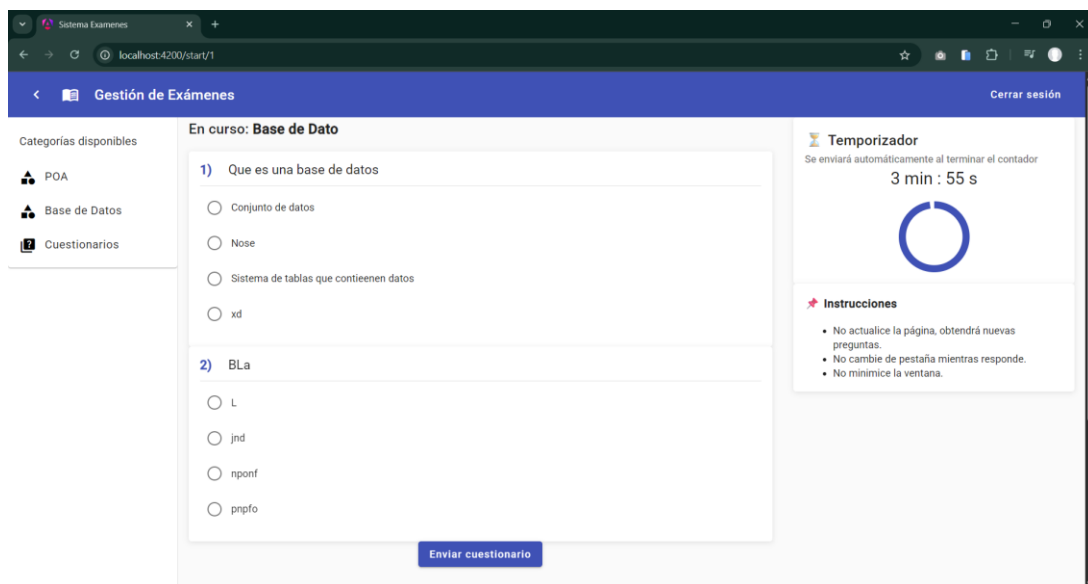
Panel de usuario con exámenes disponibles, clasificados por categoría y con opción para comenzar.

Figura 13

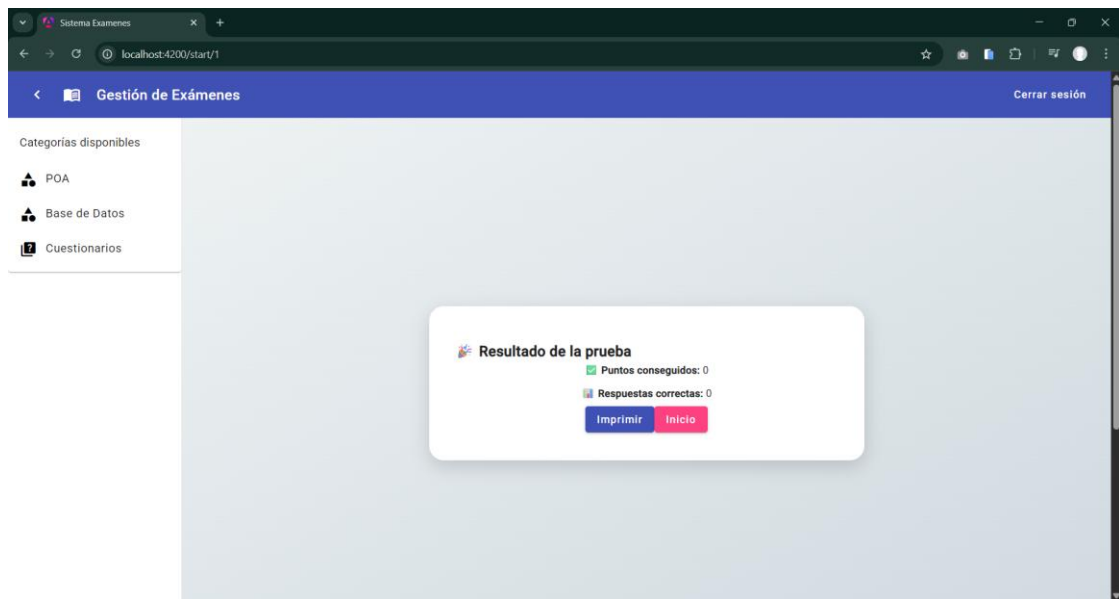


Pantalla con instrucciones previas al examen, incluyendo detalles sobre duración, intentos y condiciones de envío.

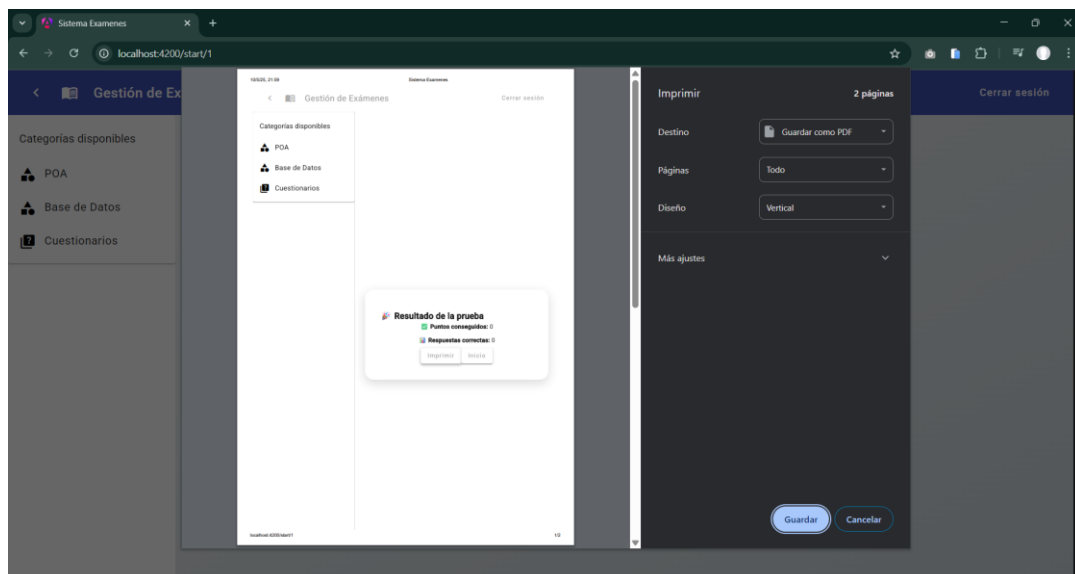
Figura 14



Pantalla del examen en curso con preguntas, temporizador e instrucciones para completar el cuestionario.

Figura 15

Resultados del examen con puntaje, respuestas correctas y opciones de imprimir o reiniciar.

Figura 16

Vista previa de impresión del resultado del examen con opción para guardar en PDF.

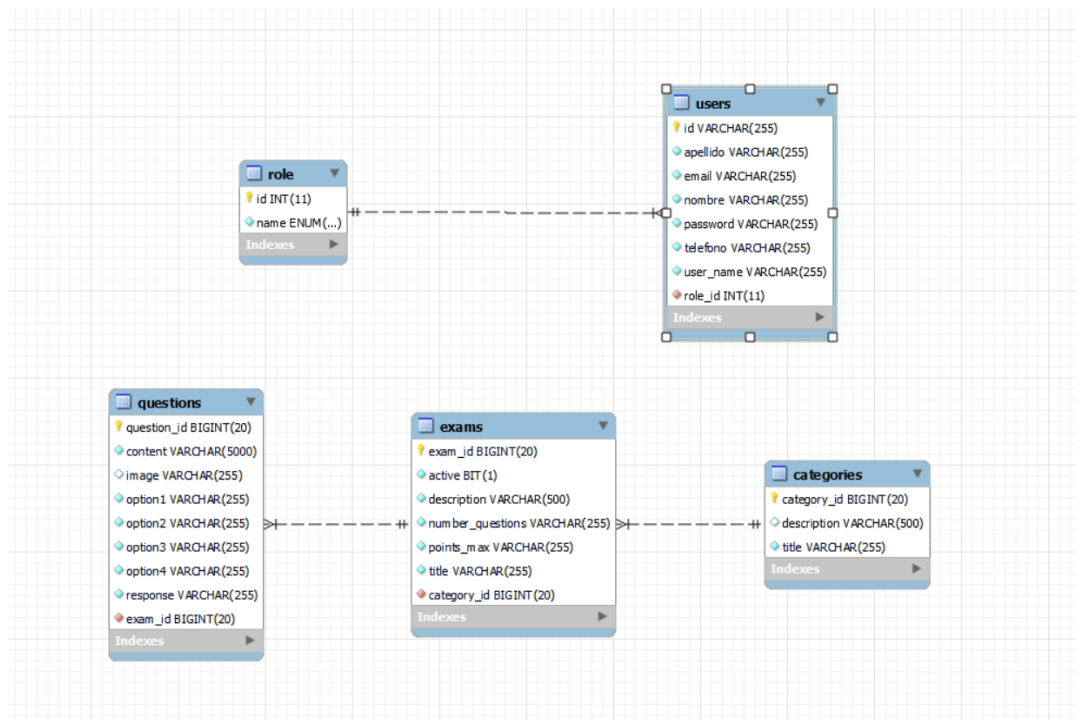
Figura 17

Diagrama entidad-relación del sistema con las tablas: usuarios, roles, exámenes, preguntas y categorías.

Diagrama de Flujo de Datos: Describe cómo fluye la información a través del sistema durante procesos clave, como la realización de un examen.

Resultados

Presentación de la Aplicación Desarrollada

La aplicación desarrollada es una plataforma de gestión de exámenes en línea orientada al ámbito educativo. Su propósito es facilitar tanto la creación como la realización de pruebas evaluativas, proporcionando una herramienta intuitiva, segura y eficaz para administradores (docentes) y usuarios (estudiantes).

Las funcionalidades principales incluyen:

- **Autenticación y Registro de Usuarios:** Mecanismo seguro para la creación de cuentas y la autenticación mediante JWT (JSON Web Tokens), permitiendo el acceso diferenciado según el rol (administrador o usuario).

- **Gestión de Categorías, Exámenes y Preguntas:** A través de un panel administrativo, se pueden crear, editar y eliminar categorías temáticas, configurar exámenes y añadir preguntas de manera estructurada.
- **Realización de Exámenes por Parte del Usuario:** Los estudiantes pueden acceder a un listado de exámenes activos, leer las instrucciones, iniciar una evaluación con temporizador, y recibir los resultados al finalizar.
- **Evaluación Automática:** El sistema corrige de forma automática las respuestas y muestra al usuario su puntaje y rendimiento.
- **Interfaz Adaptativa y Atractiva:** Gracias al uso de Angular Material, Bootstrap y CSS personalizado, la aplicación presenta una experiencia de usuario agradable, moderna y responsiva.

Estas características hacen de la aplicación una herramienta útil para instituciones educativas que buscan modernizar sus procesos de evaluación, reduciendo tiempos administrativos y errores humanos.

Resultados de las Pruebas y Evaluaciones

Durante el desarrollo se realizaron diversas pruebas funcionales, centradas en la verificación de operaciones clave como:

- **Pruebas de autenticación y seguridad:** Se comprobó que solo usuarios autenticados podían acceder a secciones protegidas, validando la generación y uso correcto de tokens JWT.
- **Pruebas de comunicación frontend-backend:** Usando Postman y la consola del navegador, se identificaron errores en los datos enviados o recibidos, permitiendo una corrección eficiente de endpoints.
- **Evaluación del rendimiento de la aplicación:** Se observó un tiempo de carga adecuado y respuesta ágil en operaciones CRUD. A pesar de ello, se identificó un reto

técnico en el refresco automático de datos tras ciertas operaciones, lo cual se solucionó parcialmente inspeccionando los eventos de carga.

- **Pruebas de la funcionalidad del examen:** Se probó la ejecución completa de un examen, desde el acceso, realización con temporizador, evaluación automática, hasta la descarga de resultados. Estas pruebas resultaron exitosas y demostraron la utilidad de la aplicación.

Principales Desafíos y Soluciones:

- **Generación del Token JWT:** Inicialmente se presentaron errores en la generación del token. La solución fue consultar la documentación y utilizar declaraciones throws para localizar fallos específicos.
- **Conexión Login/Register:** Se presentaron problemas de integración que fueron resueltos analizando los errores desde el panel del navegador.
- **Refresco de Datos:** Al insertar o actualizar información, los cambios no se reflejaban automáticamente. Se mitigó esta situación reestructurando eventos de recarga desde el frontend.

Comparación con Otras Soluciones Similares

La aplicación se puede comparar con plataformas como Google Forms, Kahoot o Socrative. Sin embargo, ofrece ventajas significativas:

- **Personalización de Roles:** A diferencia de las soluciones mencionadas, la presente aplicación ofrece una separación clara entre perfiles de administradores y usuarios.
- **Evaluación Automática Integrada:** Aunque Google Forms permite corrección automática, esta aplicación permite mayor control sobre las preguntas, la asignación de puntajes y la estructura del examen.
- **Interfaz Integrada y Responsiva:** El uso de Angular permite una SPA fluida sin recargas constantes, mejorando la experiencia del usuario.

- **Autenticación y Seguridad:** Se implementó un sistema robusto de autenticación y autorización basado en JWT y Spring Security, algo no común en plataformas abiertas.

En conclusión, la aplicación desarrollada no solo cumple con los requerimientos básicos de una plataforma de exámenes en línea, sino que ofrece mejoras importantes en personalización, seguridad, usabilidad y eficiencia, posicionándose como una solución completa y escalable para entornos educativos.

Discusión

Análisis de los Resultados Obtenidos

Los resultados evidencian que la aplicación satisface los objetivos planteados: ofrecer una plataforma funcional, segura y eficiente para la gestión de exámenes en línea. La implementación del sistema de autenticación con JWT, la diferenciación de roles, la creación dinámica de pruebas y la evaluación automática permitieron consolidar un sistema completo y operativo.

Entre los logros más destacados se encuentra la evaluación automática, que optimizó los tiempos de corrección y brindó retroalimentación inmediata a los usuarios. Asimismo, el uso de tecnologías como Spring Boot para el backend y Angular en el frontend permitió una arquitectura robusta, organizada y responsiva. La interacción efectiva entre ambos entornos a través de APIs RESTful facilitó la integración total del sistema.

A lo largo del desarrollo surgieron diversos desafíos técnicos, como la generación del token JWT, la integración entre las funcionalidades de login y registro, y la actualización automática de la interfaz tras operaciones CRUD. No obstante, estas dificultades fueron abordadas eficazmente mediante el análisis en el panel de inspección del navegador, el uso de herramientas como Postman y la revisión de documentación técnica actualizada.

Limitaciones del Proyecto y Áreas de Mejora

A pesar de los buenos resultados, el proyecto presentó limitaciones técnicas y operativas. En primer lugar, no se incorporaron pruebas unitarias automatizadas, lo que limita la verificación exhaustiva del comportamiento del sistema ante cambios futuros. En segundo lugar, algunas funcionalidades del frontend requerían la recarga manual de la página para reflejar los cambios, afectando levemente la experiencia de usuario.

Otra limitación importante fue la ausencia de un entorno de despliegue en producción. La aplicación fue desarrollada y probada en un entorno local, lo que impide valorar su comportamiento bajo condiciones reales de uso masivo. Asimismo, si bien la interfaz cumple con estándares de diseño visual, aún es posible fortalecer aspectos relacionados con la accesibilidad web para personas con discapacidades.

Se identifican como áreas de mejora:

- La implementación de pruebas automatizadas (unitarias e integración).
- El despliegue en la nube o servidores remotos para pruebas de rendimiento.
- La mejora de la reactividad del frontend tras operaciones de modificación.
- La incorporación de estándares de accesibilidad (WCAG 2.1).

Implicaciones Prácticas y Teóricas de los Resultados

En términos prácticos, la plataforma propuesta representa una solución aplicable en instituciones educativas que buscan modernizar sus sistemas de evaluación. La automatización de procesos, la retroalimentación inmediata y la administración centralizada constituyen ventajas relevantes para docentes y estudiantes.

Desde una perspectiva teórica, el desarrollo de esta aplicación aporta al estudio del uso integrado de frameworks modernos como Spring Boot y Angular, y a la implementación de patrones arquitectónicos como MVC en contextos educativos. Además, demuestra cómo

una metodología iterativa puede adaptarse eficazmente a ciclos de mejora continua en proyectos de desarrollo web.

Este proyecto reafirma la importancia de la planificación estructurada, el diseño centrado en el usuario y la capacidad de resolver problemas técnicos en tiempo real como elementos clave en la construcción de soluciones tecnológicas eficaces.

Conclusiones

Resumen de los Hallazgos Clave del Proyecto

El desarrollo de una plataforma de gestión de exámenes en línea permitió validar la viabilidad técnica y pedagógica de una solución orientada al ámbito educativo. Se logró implementar con éxito un sistema que permite la autenticación de usuarios mediante JWT, la diferenciación de roles (administrador y estudiante), la creación y edición de exámenes y preguntas, así como la evaluación automática y generación de resultados. La aplicación ofrece una experiencia de usuario fluida, segura y adaptable, respaldada por tecnologías modernas como Spring Boot, Angular y JWT.

Las pruebas funcionales realizadas durante el desarrollo demostraron que la aplicación cumple con los requisitos planteados inicialmente, presentando una arquitectura escalable, una interfaz intuitiva y una correcta gestión de los procesos evaluativos.

Respuestas a las Preguntas de Investigación Planteadas

Si bien el proyecto no se estructuró en torno a hipótesis formales, se planteó la siguiente pregunta de investigación general: *¿Es posible desarrollar una aplicación web eficiente, segura y funcional para la gestión de exámenes en entornos educativos, utilizando tecnologías actuales y accesibles?*

Los resultados obtenidos permiten afirmar que sí. La aplicación cumple satisfactoriamente con los criterios de eficiencia (automatización y rendimiento), seguridad (autenticación con tokens y control de roles) y funcionalidad (gestión integral de exámenes).

Además, su implementación y estructura responden a principios sólidos de ingeniería de software y diseño centrado en el usuario.

Lecciones Aprendidas Durante el Desarrollo del Proyecto

El proceso de desarrollo implicó una curva de aprendizaje importante en múltiples áreas:

- **Dominio de herramientas y frameworks modernos**, como Spring Boot, Angular y JWT, lo que fortaleció la capacidad técnica para diseñar sistemas escalables y seguros.
- **Enfrentamiento de desafíos reales**, como problemas de integración backend-frontend, gestión de tokens o actualización dinámica de datos, que fueron resueltos mediante documentación, análisis en el navegador y herramientas de prueba como Postman.
- **Mejora de la planificación y la organización**, mediante una metodología iterativa que permitió avanzar fase por fase, corregir errores rápidamente y optimizar el diseño progresivamente.

Estas experiencias refuerzan no solo las habilidades técnicas, sino también las capacidades de análisis, resolución de problemas, toma de decisiones y autonomía profesional.

Referencias

Amazon Web Services. (s. f.). Características de Amazon RDS. Recuperado de <https://aws.amazon.com/es/rds/features/>

Angular. (s. f.). The modern web development platform. Recuperado el 3 de mayo de 2025, de <https://angular.io/>

Angular Material. (s. f.). Angular Material UI Component Library. Recuperado el 3 de mayo de 2025, de <https://material.angular.io/>

GitHub, Inc. (s. f.). GitHub: Where the world builds software. Recuperado el 3 de mayo de 2025, de <https://github.com/>

Google. (s. f.). Communicating with backend services using HTTP. Recuperado el 3 de mayo de 2025, de <https://angular.dev/guide/http>

JWT. (s. f.). JSON Web Token for Java (JJWT). Recuperado el 3 de mayo de 2025, de <https://github.com/jwt/jwt>

Jakarta Validation API. (s. f.). Jakarta Bean Validation. Recuperado el 3 de mayo de 2025, de <https://jakarta.ee/specifications/validation/2.0/>

JavaTechOnline. (2024). Advantages of Spring Boot in Java. Recuperado de <https://javatechonline.com/advantages-of-spring-boot-in-java/>

MindInventory. (2023). Pros and Cons of Node.js Web App Development. Recuperado de <https://www.mindinventory.com/blog/pros-and-cons-of-node-js-web-app-development/>

MySQL. (s. f.). The world's most popular open source database. Recuperado el 3 de mayo de 2025, de <https://www.mysql.com/>

Node.js. (s. f.). Node.js documentation. Recuperado de <https://nodejs.org/en/docs>

Oracle. (s. f.). ¿Qué es una base de datos relacional? Recuperado de <https://www.oracle.com/database/what-is-a-relational-database/>

Oracle Corporation. (s. f.). Java™. Recuperado el 3 de mayo de 2025, de <https://www.oracle.com/java/>

Postman. (s. f.). Postman API platform. Recuperado el 3 de mayo de 2025, de <https://www.postman.com/>

SAP. (s. f.). ¿Qué es el modelado de datos? Recuperado de <https://www.sap.com/products/data-cloud/datasphere/what-is-data-modeling.html>

Spring. (s. f.). Spring Boot Reference Documentation. Recuperado el 3 de mayo de 2025, de <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>

Spring. (s. f.). Spring Data JPA Reference Documentation. Recuperado el 3 de mayo de 2025, de <https://docs.spring.io/spring-data/jpa/reference/index.html>

Spring. (s. f.). Spring Boot Web Documentation. Recuperado el 3 de mayo de 2025, de <https://docs.spring.io/spring-boot/reference/web/index.html>

Spring. (s. f.). Spring Security Reference Documentation. Recuperado el 3 de mayo de 2025, de <https://docs.spring.io/spring-security/reference/index.html>

TDEV. (s. f.). NGX-UI-LOADER. Recuperado el 3 de mayo de 2025, de <https://tdev.app/ngx-ui-loader>

Visual Studio Code. (s. f.). Code Editing. Redefined. Recuperado el 3 de mayo de 2025, de <https://code.visualstudio.com/>

Vargas, B. (2023). ¿Por qué usar el patrón MVC en aplicaciones web? Recuperado de <https://www.byronvargas.com/web/por-que-usar-el-mvc/>

Anexos

Figura 19

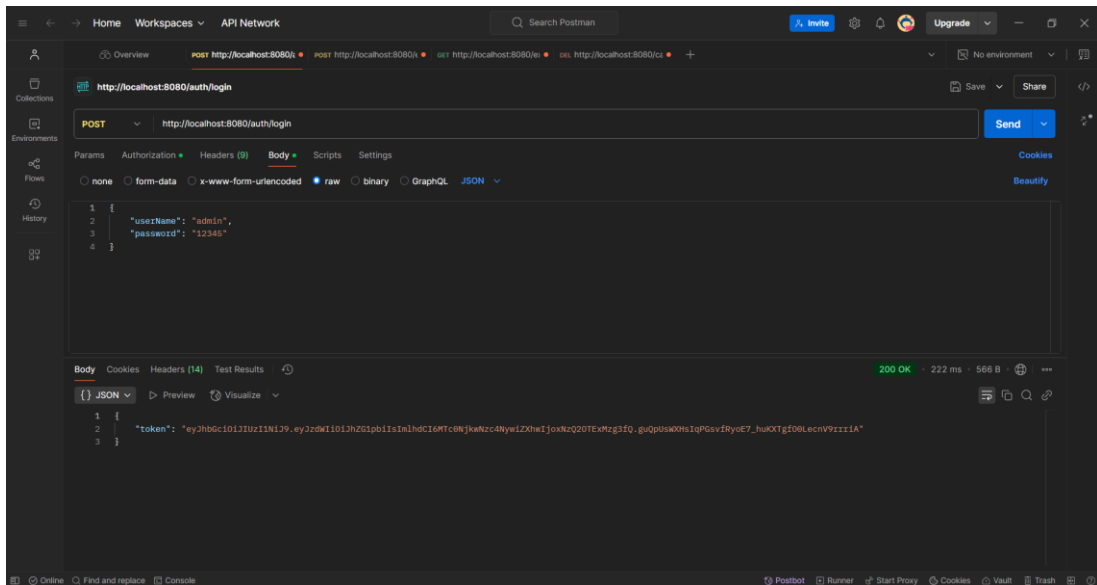
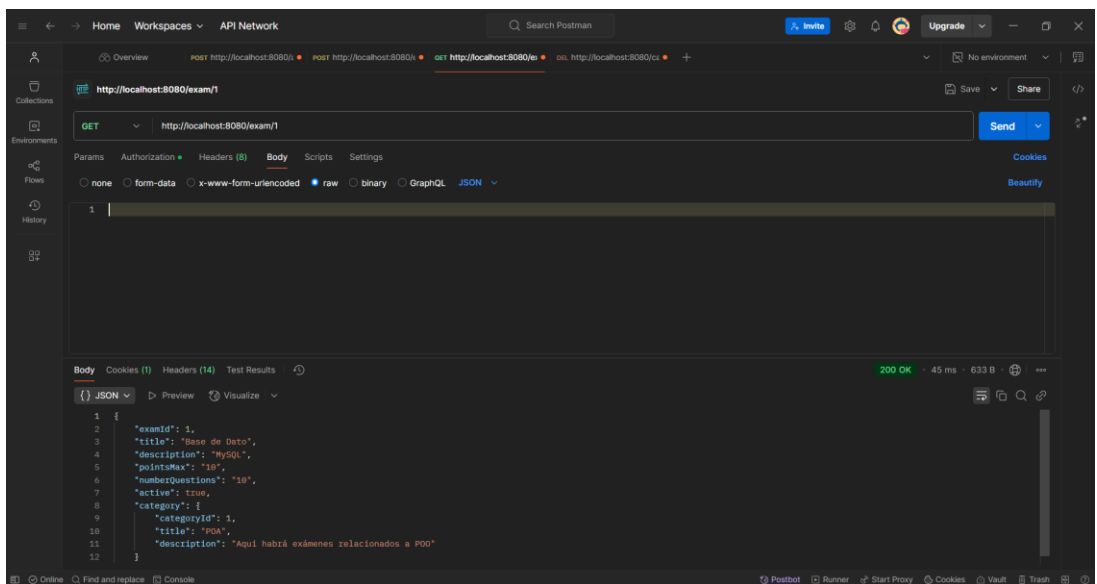
**Figura 20**

Figura 21

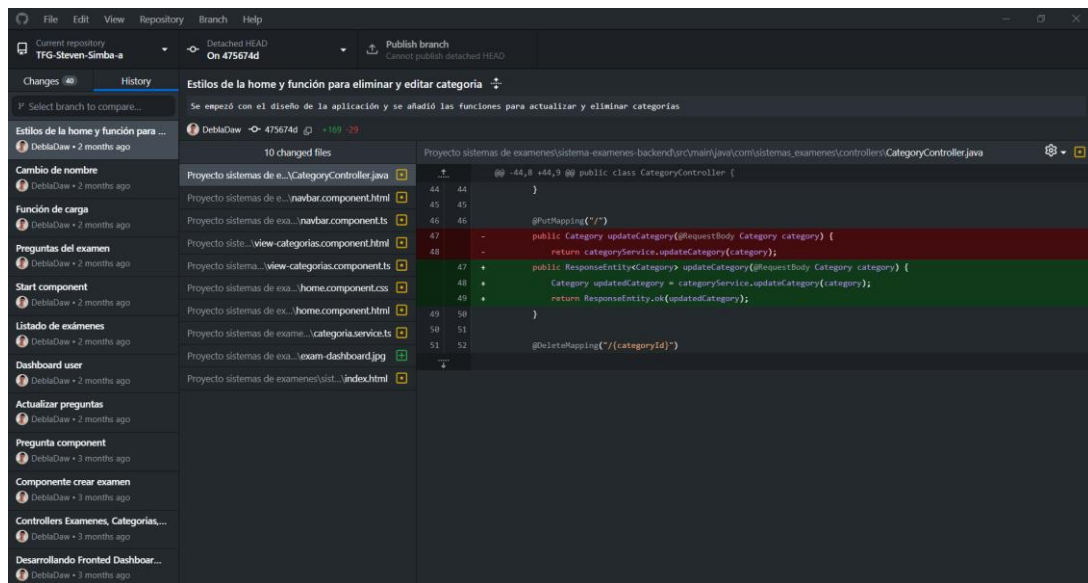


Figura 22

