Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Matemáticas e Informática

Trabajo Fin de Grado

Desarrollo de una aplicación web para realizar investigación online en educción de requisitos

Autor: PABLO PALENZUELA BERMEJO
Tutor(a): OSCAR DIESTE TUBIO

Madrid, Enero 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Matemáticas e Informática

Título: Desarrollo de una aplicación web para realizar investigación online en educción de requisitos

Enero 2025

Autor: PABLO PALENZUELA BERMEJO
Tutor: OSCAR DIESTE TUBIO
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

La educción de requisitos es una de las áreas más importantes dentro de la Ingeniería del Software, ya que su propósito principal es obtener información precisa y detallada sobre las necesidades de clientes y usuarios. Entre las diversas técnicas existentes, la entrevista es, sin lugar a dudas, la más utilizada. Sin embargo, a pesar de su relevancia, aún persisten dificultades para llevar a cabo entrevistas que resulten realmente efectivas, ya que dependen en gran medida de la organización del entrevistador y de la correcta interpretación de los datos obtenidos.

El objetivo de este proyecto es desarrollar una aplicación web orientada a investigar y comprender cómo una persona interpreta y entiende los requisitos definidos, así como el grado de precisión con el que lo hace. Para ello, la aplicación facilita la realización de entrevistas basadas en contenido multimedia, automatizando tareas repetitivas y tediosas como la conducción de entrevistas y la anotación manual de los eventos ocurridos durante estas.

Para llevar a cabo esta automatización, se han aplicado técnicas de ingeniería de software, tales como la definición de requisitos, el diseño de alto nivel y la especificación seguida de la ejecución de pruebas de software.

Abstract

The elicitation of requirements is one of the most important areas within Software Engineering, as its main purpose is to obtain accurate and detailed information about the needs of clients and users. Among the various existing techniques, the interview is undoubtedly the most commonly used. However, despite its relevance, challenges persist in conducting interviews that are truly effective, as they heavily depend on the interviewer's organization and the correct interpretation of the data obtained.

The objective of this project is to create a web application to study and understand how a person interprets and understands given requirements and how accurately they do so. The application will help by making it easier to conduct interviews using multimedia content and will automate repetitive tasks like leading the interviews and taking notes about the events that happen during them.

Software engineering techniques have been applied to develop this automation, including requirements definition, high-level design creation, and software testing specification and execution.

Tabla de contenidos

Capítulo 1

Introducción

La educción de requisitos es una de las áreas más importantes dentro de la Ingeniería del Software, ya que su propósito principal es obtener información precisa y detallada sobre las necesidades de clientes y usuarios. Entre las diversas técnicas existentes, la entrevista es, sin lugar a dudas, la más utilizada. Sin embargo, a pesar de su relevancia, aún persisten dificultades para llevar a cabo entrevistas que resulten realmente efectivas, ya que dependen en gran medida de la organización del entrevistador y de la correcta interpretación de los datos obtenidos. Finalmente, la investigación realizada respecto a cómo mejorar la efectividad de las entrevistas es muy limitada.

El objetivo de este proyecto es el desarrollo de una aplicación web diseñada para facilitar la investigación acerca de la realización de entrevistas, en particular, respecto a los procesos de comprensión de información y documentación de los hallazgos realizados.

La aplicación permitirá definir y llevar a cabo tareas como mostrar texto o audio a los sujetos experimenales, realizar pruebas escritas y responder a cuestionarios. Los sujetos experimentales acceden mediante credenciales personalizadas. De esta forma, cada sujeto podrá ser asignado a un grupo experimental específico, lo cual depende de ciertas características que interesen a los investigadores como, por ejemplo, características específicas de los sujetos experimentales o el tipo de prueba a realizar.

Toda la información se almacenará en una base de datos para su posterior análisis cuantitativo y cualitativo. El análisis de la información no es parte del presente TFG.

El desarrollo de esta aplicación se ha llevado a cabo siguiendo una metodología incremental basada en la creación de múltiples versiones del producto. Esta estrategia ha permitido implementar mejoras continuas y realizar cambios de forma ágil, asegurando la adaptabilidad del sistema a las necesidades del proyecto.

En cada iteración del desarrollo, se establecieron objetivos que guiaron la implementación de funcionalidades clave, como la gestión de usuarios y grupos, la

asignación de contenido multimedia, la creación y evaluación de pruebas, y la generación automática de resultados y estadísticas. El uso de una metodología incremental facilitó no solo el seguimiento detallado del progreso, sino también la incorporación de retroalimentación temprana, permitiendo ajustar aspectos técnicos y funcionales según fuera necesario.

En el backend, Spring Boot fue la herramienta principal, brindando una estructura sólida y simplificada para la gestión de la lógica del negocio y la comunicación con la base de datos relacional, que fue implementada con MySQL. En el frontend, Angular permitió construir una interfaz de usuario interactiva y dinámica, que asegura una experiencia intuitiva para los usuarios finales.

Este enfoque tecnológico combinado con una estrategia incremental garantizó que el desarrollo se mantuviera alineado con los objetivos establecidos, a la vez que se ofrecía un sistema robusto, escalable y preparado para futuras ampliaciones.

Capítulo 2

Estado de la Cuestión

En este capítulo se abordan los aspectos fundamentales relacionados con el desarrollo del proyecto, detallando las tecnologías empleadas y el papel que desempeñan en la consecución de los objetivos planteados. Además, se analiza cómo estas herramientas han contribuido al diseño, implementación y despliegue de la solución propuesta, destacando sus ventajas y razones para su elección en el contexto del proyecto.

2.1. SPRING-BOOT

Java Spring Boot es una extensión del *Spring Framework*, un marco de trabajo empresarial ampliamente utilizado en el desarrollo de aplicaciones basadas en la *Java Virtual Machine (JVM)*. Spring Framework, reconocido por su versatilidad y robustez, ha sido durante años la base para la construcción de aplicaciones Java empresariales. Sin embargo, una de las mayores barreras de entrada para los desarrolladores era la complejidad asociada con su configuración inicial y la gestión de dependencias. Aquí es donde **Spring Boot** entra en juego, proporcionando un enfoque simplificado y eficiente para crear aplicaciones modernas.

Spring Boot es una herramienta diseñada específicamente para acelerar y simplificar el desarrollo de aplicaciones web y microservicios, aprovechando al máximo el ecosistema de Spring Framework. Combina una serie de características avanzadas que permiten a los desarrolladores centrarse en la lógica de negocio sin preocuparse excesivamente por configuraciones complejas. Estas características clave incluyen:[1]

2.1.1. Características de Spring Boot

- **Configuración Automática :** Una de las mayores ventajas de Spring Boot es su capacidad para realizar configuraciones automáticas. Basándose en las dependencias incluidas en el proyecto y las propiedades definidas, Spring Boot configura automáticamente los componentes necesarios para el correcto funcionamiento de la aplicación. Por ejemplo, si se incluye una dependencia de una base de datos como *Spring Data JPA*, Spring Boot con-

figurará automáticamente un gestor de entidades, una conexión a la base de datos y otros componentes relacionados.

- **Enfoque de Configuración Obstinada :** Spring Boot adopta un enfoque de configuración predeterminado. Esto significa que proporciona configuraciones por defecto que funcionan bien en la mayoría de los casos, permitiendo a los desarrolladores empezar rápidamente. Sin embargo, estas configuraciones son completamente personalizables según las necesidades específicas del proyecto.

  Por ejemplo, al desarrollar una API REST, Spring Boot incluye por defecto una implementación de servidor web embebido (como Tomcat, Jetty o Undertow), configuraciones básicas de seguridad y soporte para serialización/deserialización JSON mediante Jackson.

- **Aplicaciones Autónomas :** Spring Boot permite crear aplicaciones completamente autónomas que no requieren un servidor de aplicaciones externo para ejecutarse. Esto se logra mediante la inclusión de un servidor embebido, (en este trabajo *Apache Tomcat*), en el propio paquete de la aplicación. Las aplicaciones pueden ejecutarse directamente como archivos `.jar` o `.war`, lo que simplifica enormemente el despliegue.

2.1.2. Ventajas de Spring Boot

La elección de Spring Boot como tecnología principal para el desarrollo del backend se fundamenta en una serie de ventajas significativas. A continuación, se detallan los motivos que respaldan esta decisión:

- **Inicio Rápido:** Gracias a su configuración automática y predeterminada, los desarrolladores pueden crear aplicaciones funcionales en minutos. Esto es especialmente útil para prototipos rápidos o proyectos con plazos ajustados.

- **Ecosistema Unificado:** Spring Boot integra herramientas del ecosistema de Spring, como *Spring Data* (para la gestión de datos), *Spring Security* (para la seguridad) y *Spring Cloud* (para arquitecturas de microservicios). Esta integración asegura que todas las herramientas trabajen de manera fluida y consistente.

- **Facilidad de Pruebas:** Spring Boot incluye soporte avanzado para pruebas unitarias e integradas. Por ejemplo, permite simular servidores embebidos para probar servicios REST sin necesidad de desplegarlos realmente.

- **Documentación y Comunidad Activa:** Spring Boot cuenta con una extensa documentación oficial y una comunidad global activa, lo que facilita la resolución de problemas y el aprendizaje para nuevos desarrolladores.

- **Listo para la Producción:** Las aplicaciones construidas con Spring Boot están diseñadas para ser robustas, escalables y listas para su uso en entornos de producción.

En este proyecto,Spring Boot fue la tecnología elegida para el *Back-End* debido en gran parte a lo mencionado anteriormente y a que se contaba con experiencia previa en esta tecnología. Lo que posiblemente lo convertía en la mejor opción.

2.2. MAVEN

Maven es un sistema de gestión de proyectos y construcción ampliamente utilizado en el desarrollo de software Java. Fue diseñado para simplificar y estandarizar las tareas repetitivas y complejas asociadas con la construcción, prueba y despliegue de aplicaciones. En el contexto de este proyecto, Maven ha sido seleccionado como el gestor de dependencias y herramienta de construcción principal debido a sus numerosas ventajas y características clave.[2]

2.2.1. Características de Maven

1. **Gestión de dependencias automatizada**: Maven permite definir las bibliotecas necesarias para el proyecto en un archivo de configuración centralizado llamado `pom.xml` (Project Object Model). Una vez especificadas, Maven se encarga de descargar automáticamente las dependencias desde un repositorio remoto y asegurar que estén disponibles en las versiones adecuadas. Esto evita conflictos entre librerías y garantiza consistencia en el entorno de desarrollo.[3]
2. **Construcción y compilación simplificadas**: Gracias a un ciclo de construcción estándar, Maven puede compilar el código fuente, empaquetarlo en formatos como JAR o WAR, ejecutar pruebas automatizadas y desplegar la aplicación en diferentes entornos. Esto permite a los desarrolladores enfocarse en el código, dejando las tareas repetitivas a la herramienta.
3. **Soporte para integración continua**: Maven se integra fácilmente con herramientas de integración continua (CI) como Jenkins o GitHub Actions, facilitando la automatización del proceso de compilación, prueba y despliegue. Esto contribuye a detectar errores temprano en el ciclo de desarrollo y mejora la calidad del software.
4. **Gestión de entornos múltiples**: Maven permite configurar perfiles para diferentes entornos (por ejemplo, desarrollo, pruebas y producción). Esto asegura que la aplicación pueda ejecutarse con configuraciones específicas para cada entorno sin necesidad de cambios manuales en el código.

2.2.2. Ventajas de Maven

- **Consistencia**: Garantiza que se utilicen las mismas versiones de dependencias, eliminando problemas relacionados con configuraciones locales inconsistentes.
- **Ahorro de tiempo**: Automatiza tareas como la descarga de dependencias, la compilación y la ejecución de pruebas, reduciendo la carga de trabajo manual.

- **Escalabilidad**: Maneja proyectos grandes con múltiples módulos de manera eficiente, facilitando la organización y el crecimiento del proyecto.
- **Flexibilidad**: Ofrece soporte para una amplia variedad de plugins que extienden sus capacidades, desde la generación de documentación hasta el análisis estático de código.

En este proyecto, Maven se utilizó para gestionar dependencias clave como Spring Boot, así como otras bibliotecas necesarias para la implementación de funcionalidades específicas.Su integración con Github aseguró que el proyecto evolucionara de manera consistente y controlada.

2.3. MySQL

MySQL es un sistema de gestión de bases de datos relacionales ampliamente utilizado, conocido por su alto rendimiento, estabilidad y facilidad de uso. Es de código abierto y compatible con una amplia variedad de aplicaciones, desde proyectos pequeños hasta sistemas empresariales de gran escala.[4]

2.3.1. Características de MySQL

- **Estabilidad y rendimiento:** MySQL tiene capacidad para manejar grandes volúmenes de datos y múltiples transacciones simultáneas con un rendimiento consistente.
- **Compatibilidad:** Totalmente integrado con Java y con *Spring Boot*, además de herramientas como Hibernate ORM, utilizadas en este proyecto.
- **Escalabilidad:** Su diseño permite trabajar con bases de datos pequeñas y grandes sin afectar su desempeño.
- **Seguridad avanzada:** Incluye autenticación de usuarios, encriptación y control de acceso, asegurando la protección de los datos.

2.3.2. Ventajas de MySQK

- **Facilidad de uso:** MySQL permite implementar bases de datos relacionales de forma rápida y eficiente.
- **Interoperabilidad:** Funciona bien con múltiples sistemas operativos y lenguajes de programación, proporcionando flexibilidad al desarrollo.
- **Optimización de consultas:** Permite realizar consultas complejas de manera eficiente, asegurando tiempos de respuesta rápidos.

En este proyecto, MySQL fue la tecnología elegida para la base de datos debido a las características mencionadas, a la integración con *Java* y *Spring Boot*, y también a la necesidad de crear una base de datos relacional.

2.4. Hibernate ORM

Hibernate ORM (Object-Relational Mapping) es una herramienta de mapeo objeto-relacional para Java que facilita la interacción entre las aplicaciones y las bases de datos relacionales. Permite a los desarrolladores trabajar con objetos Java en lugar de con tablas SQL, simplificando el acceso y manejo de los datos.[5]

2.4.1. Características de Hibernate ORM

- **Simplificación del acceso a datos:** Hibernate convierte automáticamente las operaciones de Java en sentencias SQL, eliminando la necesidad de escribir consultas manuales.
- **Independencia del motor de base de datos:** Ofrece una capa de abstracción que permite trabajar con diferentes motores de base de datos sin modificar el código.
- **Manejo eficiente de relaciones:** Administra relaciones entre entidades (uno a muchos, muchos a muchos, etc.) mediante anotaciones o configuraciones XML.
- **Caché integrado:** Incorpora un sistema de caché que optimiza las consultas y mejora el rendimiento.
- **Generación automática de esquemas:** Hibernate puede crear automáticamente las tablas y relaciones en la base de datos a partir de las entidades definidas en Java.

2.4.2. Ventajas de Hibernate ORM

- **Productividad:** Al reducir la cantidad de código necesario para interactuar con la base de datos, permite a los desarrolladores centrarse en la lógica de negocio.
- **Consistencia:** Garantiza que el modelo de datos en Java esté sincronizado con la estructura de la base de datos, evitando inconsistencias.
- **Flexibilidad:** Facilita la migración entre bases de datos sin necesidad de realizar grandes cambios en el código fuente.
- **Validación de datos:** Incluye mecanismos para validar datos antes de almacenarlos, mejorando la calidad de los mismos.
- **Optimización de tiempo:** El manejo automático de relaciones y consultas reduce el esfuerzo requerido para desarrollar funcionalidades relacionadas con la base de datos.

2.5. Bibliotecas Relevantes en el Back

Además de las bibliotecas predeterminadas proporcionadas por Spring Boot, el proyecto incorpora una selección de bibliotecas adicionales clave. Estas herra-

mientas complementan las funcionalidades base de Spring Boot, optimizando diversos aspectos del desarrollo, como la autenticación y la gestión del código. A continuación, se describen las bibliotecas más relevantes empleadas en este proyecto.

2.5.1. JJWT (Java JSON Web Token)

JJWT es una biblioteca utilizada para trabajar con JSON Web Tokens (JWT), un estándar que permite la transmisión segura de información entre dos partes como un objeto JSON. En este proyecto, JJWT gestiona la autenticación de usuarios al eliminar la necesidad de mantener sesiones en el servidor. La biblioteca permite la creación, firma y validación de tokens, lo que garantiza una autenticación eficiente y segura. Su integración es sencilla y cumple con el estándar JWT, lo que facilita su uso en sistemas distribuidos.

2.5.2. Lombok

Lombok es una biblioteca que mejora la productividad y simplifica el desarrollo en Java. A través de anotaciones, Lombok genera automáticamente métodos comunes como getters, setters, constructores y `toString`, reduciendo significativamente el código repetitivo. Esto contribuye a una mayor claridad y enfoque en la lógica de la ap licación, haciendo el código más limpio y fácil de mantener.[6]

2.6. Angular

Angular es un marco de desarrollo front-end de código abierto, ampliamente utilizado para construir aplicaciones web dinámicas, interactivas y modernas. Creado y mantenido por Google, Angular ha evolucionado significativamente desde su versión inicial (AngularJS), adoptando una arquitectura completamente reestructurada en sus versiones posteriores, lo que lo convierte en uno de los frameworks más robustos y avanzados disponibles en la actualidad.[7]

Angular está diseñado para facilitar el desarrollo de aplicaciones de una sola página (*Single Page Applications, SPA*) y otras soluciones web complejas, proporcionando un enfoque basado en componentes. Este framework utiliza **TypeScript**, un superconjunto de JavaScript que incorpora tipado estático, mejorando la calidad del código, reduciendo errores y facilitando el mantenimiento.

Además, Angular promueve el desarrollo estructurado al separar la lógica de negocio, la presentación y los datos, lo que mejora la organización del proyecto. Su extensa documentación y su activa comunidad de desarrolladores proporcionan soporte continuo, recursos y ejemplos, facilitando la adopción y el aprendizaje del framework.

2.6.1. Características principales

- **Arquitectura basada en componentes**: Permite dividir la aplicación en pequeñas piezas de funcionalidad reutilizables, lo que facilita la organización

del proyecto y promueve el desarrollo ágil.

- **Data Binding bidireccional**: Proporciona una sincronización automática entre el modelo de datos y la vista, asegurando que los cambios realizados en cualquiera de ellos se reflejen instantáneamente en el otro.
- **Inyección de dependencias (Dependency Injection)**: Permite gestionar las dependencias de los componentes y servicios de manera eficiente, fomentando la modularidad y la reutilización del código.
- **TypeScript como lenguaje principal**: Mejora la legibilidad, la calidad y la escalabilidad del código al proporcionar tipado estático, interfaces y otras características avanzadas que no están disponibles en JavaScript puro. [8]
- **Sistema de enrutamiento avanzado**: Permite gestionar la navegación en aplicaciones de una sola página de manera dinámica y eficiente, soportando parámetros, rutas protegidas y *lazy loading* (carga diferida).
- **Angular CLI (Command Line Interface)**: Una herramienta que acelera el desarrollo al proporcionar comandos para crear componentes, servicios, módulos, ejecutar pruebas, generar compilaciones optimizadas y más.
- **Soporte para SSR (Server-Side Rendering)**: Angular Universal permite renderizar las aplicaciones del lado del servidor, mejorando el SEO y el rendimiento inicial de la página.
- **Soporte para pruebas**: Angular incluye herramientas integradas para realizar pruebas unitarias y de extremo a extremo, asegurando la calidad del código durante todo el ciclo de desarrollo.
- **Interoperabilidad con APIs y servicios externos**: Angular facilita el consumo de servicios REST, lo que lo hace ideal para aplicaciones como la de este trabajo que requieren comunicación constante con el *back-end*.

2.6.2. Ventajas de Angular

- **Desarrollo estructurado y escalable**: Su arquitectura modular permite manejar proyectos grandes y complejos, facilitando la integración y el mantenimiento de nuevas funcionalidades.
- **Mejora de la calidad del código**: Gracias a TypeScript, se pueden detectar errores durante el desarrollo, lo que reduce problemas en tiempo de ejecución.
- **Rendimiento optimizado**: Con características como la carga diferida de módulos (*lazy loading*) y el soporte para SSR, Angular asegura tiempos de carga rápidos y una experiencia de usuario fluida.
- **Interfaces de usuario modernas**: Angular facilita la creación de interfaces atractivas por medio de HTML y SCSS utilizados en este proyecto.

2.7. GitHub

GitHub es una plataforma de desarrollo colaborativo basada en Git, el sistema de control de versiones más popular en el ámbito del desarrollo de software. Lanzada en 2008, GitHub combina las funcionalidades de Git con características adicionales como herramientas de colaboración, gestión de proyectos y automatización de flujos de trabajo.

GitHub proporciona un entorno centralizado donde los desarrolladores pueden almacenar, versionar y gestionar sus proyectos, ya sea de forma pública o privada. Su integración con Git permite realizar un seguimiento de los cambios realizados en el código fuente, facilitando el trabajo en equipo y asegurando la integridad del proyecto a través de versiones históricas. Además, incluye herramientas útiles para revisiones de código, documentación en formato Markdown, y la capacidad de automatizar procesos mediante GitHub Actions.

En este proyecto, GitHub ha sido empleado para documentar y gestionar las distintas versiones del desarrollo. A través del control de versiones proporcionado por GitHub, se ha podido realizar un seguimiento detallado de los cambios realizados en el código fuente, identificar mejoras y solucionar problemas de manera colaborativa. Asimismo, la funcionalidad de ramas (*branches*) ha permitido desarrollar nuevas características de forma independiente antes de integrarlas en el proyecto principal. Esto ha facilitado un enfoque incremental y organizado para el desarrollo, garantizando la trazabilidad de todas las modificaciones realizadas a lo largo del ciclo de vida del proyecto.[9]

2.8. Lenguajes de Programación Utilizados

En el desarrollo del proyecto se han empleado diversos lenguajes de programación, cada uno con un propósito específico dentro de la arquitectura del sistema. A continuación, se describen brevemente los lenguajes utilizados y su papel en el proyecto:

2.8.1. Java

Java es un lenguaje de programación orientado a objetos ampliamente utilizado en el desarrollo de aplicaciones empresariales y web. Su robustez, portabilidad y rica biblioteca estándar lo convierten en una opción ideal para proyectos de gran escala.

En este proyecto, Java ha sido el lenguaje principal para el desarrollo del *backend*. Con la ayuda del marco de trabajo Spring Boot, se implementaron funcionalidades como la gestión de usuarios, la autenticación o la comunicación con la base de datos.

2.8.2. TypeScript

TypeScript es un superconjunto de JavaScript que introduce un sistema de tipos estáticos y otras características avanzadas que mejoran la experiencia de

desarrollo. Este lenguaje es especialmente adecuado para aplicaciones complejas y de gran escala debido a su capacidad de detectar errores en tiempo de compilación y su compatibilidad con las herramientas modernas de desarrollo.

En este proyecto, TypeScript fue utilizado para el desarrollo del *front-end* mediante el marco de trabajo Angular. Con TypeScript, se implementaron componentes interactivos, servicios para la comunicación con el *back-end*, y otras funcionalidades de la interfaz de usuario como el tiempo de examen.

2.8.3. HTML y CSS

HTML (HyperText Markup Language) y CSS (Cascading Style Sheets) son lenguajes fundamentales en el desarrollo web, utilizados para estructurar y estilizar páginas web.

En el proyecto, se utilizaron para definir todas las páginas y estructurar la web. Además, se emplearon extensiones como SCSS (Sassy CSS) para facilitar el manejo de estilos mediante características avanzadas como variables, anidamiento y reutilización de código.

2.8.4. SQL

SQL (Structured Query Language) es el lenguaje estándar para la gestión y consulta de bases de datos relacionales.

En el proyecto, SQL se utilizó para interactuar con la base de datos MySQL. Aunque muchas operaciones fueron gestionadas mediante Hibernate ORM, algunas consultas personalizadas fueron escritas directamente en SQL para optimizar el rendimiento.

Capítulo 3

Objetivos

La educción de requisitos es un área de la Ingeniería de Software que tiene como objetivo adquirir información acerca de las necesidades de clientes y usuarios. La técnica de educción más utilizada es la entrevista. Sin embargo, no se entiende todavía como realizar entrevistas realmente efectivas.

El objetivo de este proyecto es desarrollar una aplicación web orientada a investigar y comprender cómo una persona interpreta y entiende los requisitos definidos, así como el grado de precisión con el que lo hace. Para ello, la aplicación facilita la realización de entrevistas basadas en contenido multimedia, automatizando tareas repetitivas y tediosas como la conducción de entrevistas y la anotación manual de los eventos ocurridos durante estas.

El desarrollo de este sistema ha seguido una planificación estructurada en cinco etapas principales, con el objetivo de asegurar que el proyecto cumpla con lo establecido:

3.1. Definición de los Requisitos del Sistema

El primer paso consistió en definir y especificar de manera clara los requisitos del sistema, tanto funcionales como no funcionales. Entre los principales requisitos destacan:

- Gestión de usuarios y autenticación.
- Asignación de grupos y contenido multimedia según categorías.
- Creación y evaluación de pruebas basadas en contenido multimedia.
- Almacenamiento y análisis de resultados.

Esta fase es esencial para establecer una base sólida para el desarrollo, garantizando que la aplicación cumpla con los objetivos propuestos y responda a las necesidades del usuario final.

3.2. Desarrollo de la Aplicación Web

El desarrollo del sistema se llevó a cabo de forma incremental, dividiendo el trabajo en las siguientes fases:

Diseño de la Arquitectura: Se definió una arquitectura modular, donde cada componente de la aplicación (gestión de usuarios, contenido, pruebas y análisis de resultados) está claramente estructurado. Esto asegura escalabilidad, facilidad de mantenimiento y una separación clara de responsabilidades.

Implementación: Se utilizó *Angular* como *framework* principal para el *front-end* y *Spring Boot* para el *back-end*, incluyendo la gestión de la base de datos. Durante esta etapa, se siguieron prácticas recomendadas de desarrollo de software para garantizar la calidad del código.

Pruebas del Sistema: Se realizaron pruebas para asegurar que la aplicación en su conjunto cumpla con los requisitos definidos.

3.3. Implementación del Sistema de Grupos y Contenido Multimedia

Una de las características clave del sistema es la personalización de las pruebas. Esto se logra mediante la asignación de usuarios a grupos específicos y la presentación de contenido multimedia relevante para cada grupo. Los objetivos principales de esta funcionalidad incluyen:

- Definir los criterios de agrupación de usuarios.
- Asignar contenido (documentos, audios, videos) relevante a cada grupo.
- Garantizar que el contenido esté disponible solo durante un tiempo limitado antes de que los usuarios respondan a las preguntas.

Esta funcionalidad es crucial para personalizar las entrevistas y garantizar que los estudios sean lo más realistas posibles.

3.4. Automatización del Proceso de Evaluación y Almacenamiento de Resultados

Otro objetivo clave del proyecto fue automatizar la evaluación de respuestas y el almacenamiento de resultados. Esto se logró implementando las siguientes características:

- Almacenar las respuestas de los usuarios en una base de datos para su posterior análisis.
- Permitir que los resultados sean accesibles a los usuarios de manera individual, brindándoles la posibilidad de revisar su desempeño.

Este proceso automatizado no solo mejora la eficiencia del sistema, sino que también reduce la posibilidad de errores humanos en la evaluación.

Capítulo 4

Plan de Trabajo

Este capítulo describe el plan de trabajo llevado a cabo durante el desarrollo del proyecto, dividido en las fases inicial, intermedia y final. Además, se presentan las tareas correspondientes a cada fase junto con los diagramas de Gantt respectivos. Estos planes han sido diseñados para garantizar una gestión adecuada del tiempo y los recursos, asegurando una ejecución eficiente y estructurada del proyecto.

4.1. Plan de Trabajo Inicial

El plan de trabajo inicial establece las tareas y el cronograma preliminar para la organización y ejecución del proyecto, abarcando desde la fase de estudio y análisis hasta la implementación y la documentación final.

4.1.1. Lista de Tareas

En el Plan de Trabajo Inicial se propusieron las siguientes tareas a realizar:

- Análisis de herramientas y tecnologías disponibles para seleccionar aquellas que mejor se adapten a los requisitos del proyecto.
- Identificación y especificación de las funcionalidades esenciales y los criterios de calidad que debe cumplir el sistema.
- Elaboración de una estructura modular que facilite la escalabilidad, el mantenimiento y la integración de los diferentes componentes del sistema.
- Desarrollo de funcionalidades que permitan la creación, autenticación y asignación de usuarios a grupos específicos.
- Creación de un sistema para gestionar y presentar contenido multimedia personalizado según las características de cada grupo.
- Desarrollo de un sistema que permita la creación de pruebas, la corrección automática y el almacenamiento de los resultados en la base de datos.

- Diseño de interfaces que permitan a los usuarios consultar sus resultados y estadísticas de manera clara y sencilla.
- Realización de pruebas unitarias, de integración y funcionales para garantizar que todos los módulos funcionen correctamente y cumplan con los requisitos establecidos.
- Redacción de documentación que facilite el uso y mantenimiento del sistema, incluyendo guías para desarrolladores y manuales para usuarios finales.
- Elaboración de la memoria final que detalla el proceso de desarrollo, las decisiones tomadas y los resultados obtenidos.

4.1.2. Diagrama de Gantt

El diagrama de Gantt del Plan de Trabajo Inicial muestra la distribución temporal de las tareas planificadas, proporcionando una visión clara de la secuencia y la duración de cada actividad.

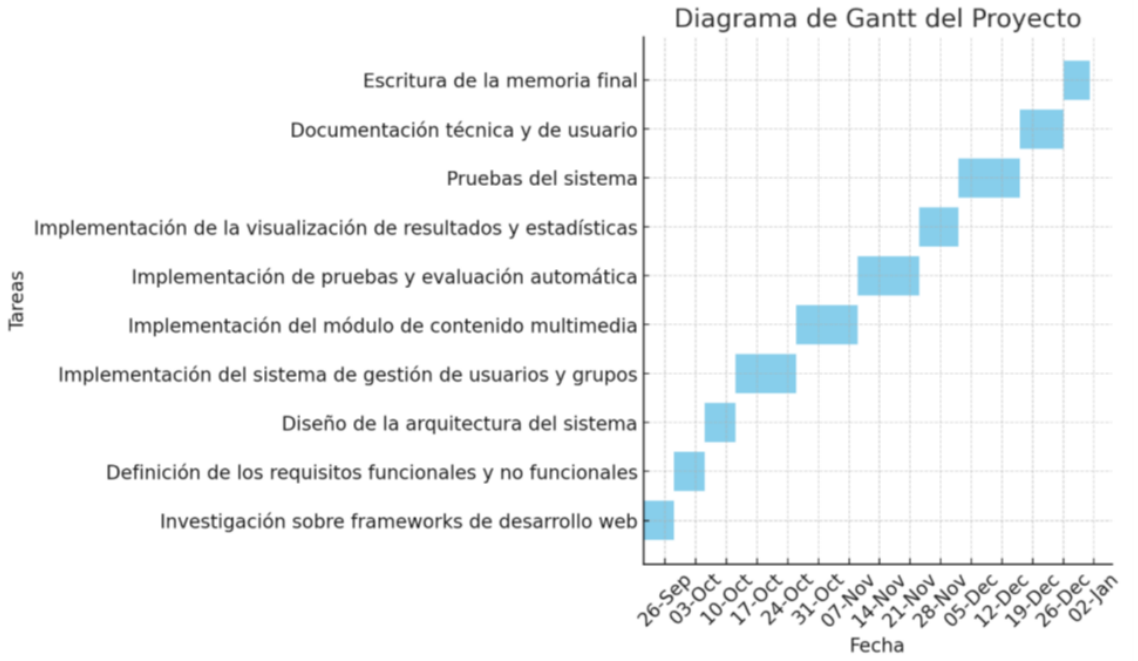

Figura 4.1: Diagrama de Gantt del Plan de Trabajo Inicial

4.2. Plan de Trabajo Intermedio

El plan de trabajo intermedio revisa y actualiza el plan inicial, tomando en cuenta el progreso real y las posibles incidencias surgidas durante la ejecución del proyecto.

4.2.1. Lista de Tareas

La lista de tareas se mantiene sin modificaciones con respecto a las definidas inicialmente, ya que continúan siendo pertinentes y esenciales para la culminación exitosa del proyecto.

4.2.2. Diagrama de Gantt

Este diagrama de Gantt representa los ajustes temporales realizados en cada tarea, proporcionando una visión clara de las modificaciones aplicadas a la planificación original.

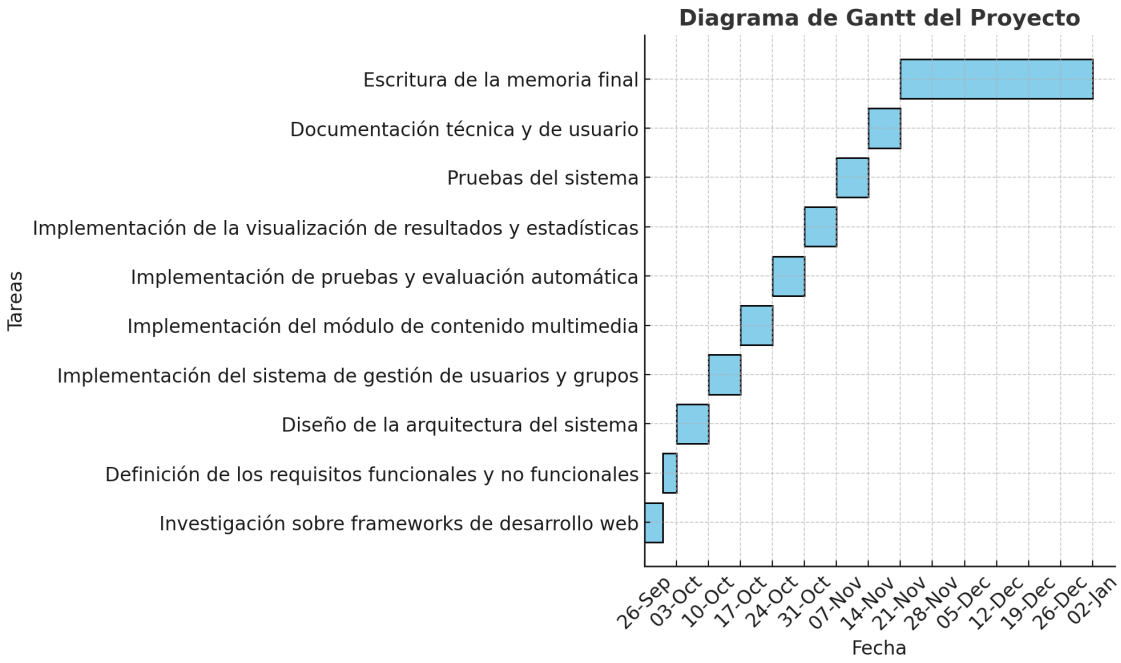

Figura 4.2: Diagrama de Gantt del Plan de Trabajo Intermedio

4.3. Plan de Trabajo Final

El plan de trabajo final presenta un resumen de las tareas completadas y las pendientes, ofreciendo una visión integral del progreso del proyecto y los pasos restantes para su conclusión.

4.3.1. Lista de Tareas

La lista de tareas permanece sin cambios respecto a las definidas inicialmente.

4.3.2. Diagrama de Gantt

El diagrama de Gantt del Plan de Trabajo Final muestra los plazos finales en los que se acabó realizando la aplicación.

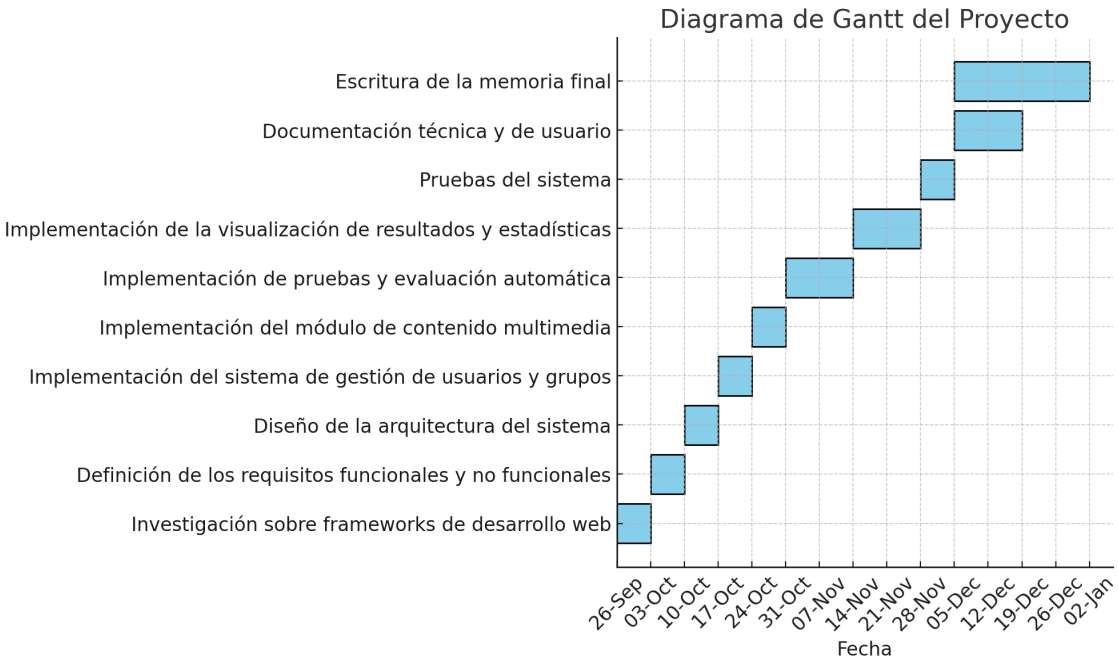

Figura 4.3: Diagrama de Gantt del Plan de Trabajo Final

Capítulo 5

Desarrollo

Durante la realización de este trabajo de fin de grado, se han seguido técnicas de ingeniería del software con el objetivo de desarrollar un producto robusto y mantenible. De esta manera, en este capítulo se tratan los procedimientos seguidos utilizando un modelo de proceso incremental.

Para el desarrollo de esta aplicación se ha seguido una estrategia de desarrollo evolutivo. La estrategia de desarrollo evolutivo mediante versiones implica crear, lanzar y mejorar la aplicación en ciclos iterativos. Este enfoque permite realizar pruebas, obtener retroalimentación y corregir errores o realizar mejoras en etapas tempranas del desarrollo, en lugar de esperar hasta tener una versión finalizada.

Al trabajar por medio de versiones se facilita la implementación de nuevas funcionalidades gradualmente, permitiendo que cada versión añada valor y funciones adicionales a la aplicación. Esto facilita la identificación y resolución de problemas, ya que los cambios se introducen en pequeños incrementos, haciendo más sencillo detectar y solucionar errores. Además, esta estrategia permite comenzar a interactuar con una versión básica de la aplicación, generando comentarios que guiarán el desarrollo de las siguientes versiones. Los beneficios de este enfoque son variados: en primer lugar, mejora la gestión de riesgos, ya que los errores o problemas se identifican en etapas tempranas. También mejora la calidad final de la aplicación, pues las funcionalidades se implementan y prueban de forma controlada y escalonada. Finalmente, el desarrollo evolutivo favorece la flexibilidad, permitiendo adaptarse a nuevos requisitos o ajustes sin afectar significativamente el avance general del proyecto.

El proyecto final tendrá una estructura dividida en tres partes: *back-end*, *front-end* y, finalmente, la base de datos.

El código fuente de este proyecto se encuentra disponible en el siguiente repositorio de **Github.**

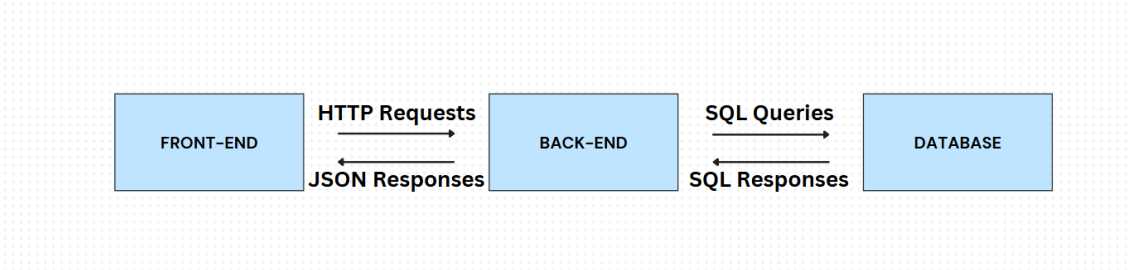

Figura 5.1: Estuctura Proyecto

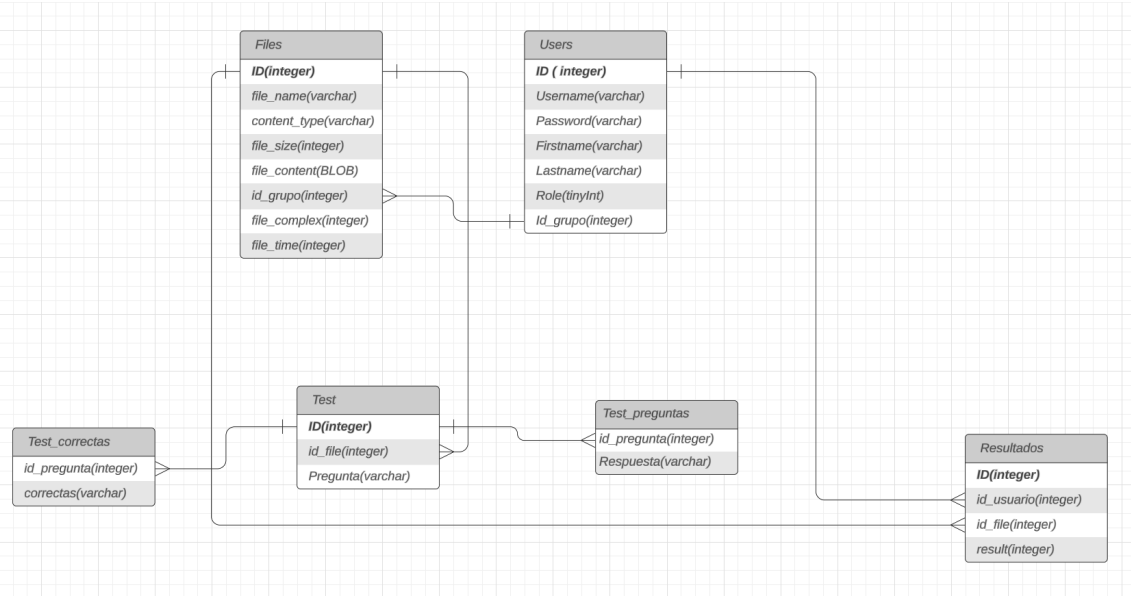

Figura 5.2: Diagrama Base de Datos

5.1. Primer Incremento

La primera versión del proyecto vino tras la selección de la arquitectura y la generación de la aplicación. Se conectó en primer lugar el back con el front para poder asegurar que funcionaba correctamente. La primera versión estuvo enteramente enfocada en el desarrollo front o visual de la aplicación, para poder así entender de forma correcta lo que necesitaba la aplicación y las cualidades de esta. Se crearon las páginas más importantes y se les dio un diseño atractivo y moderno adecuado a los colores universitarios. A continuación se detallan por medio visual lo realizado en esta versión:

La primera página que se creó fue la del inicio de sesión, que en este caso fue una página sencilla que incorporaba los elementos propios de esta, un placeholder para usuario y contraseña, junto con la posibilidad de acceder al registro y a la recuperación de contraseña.

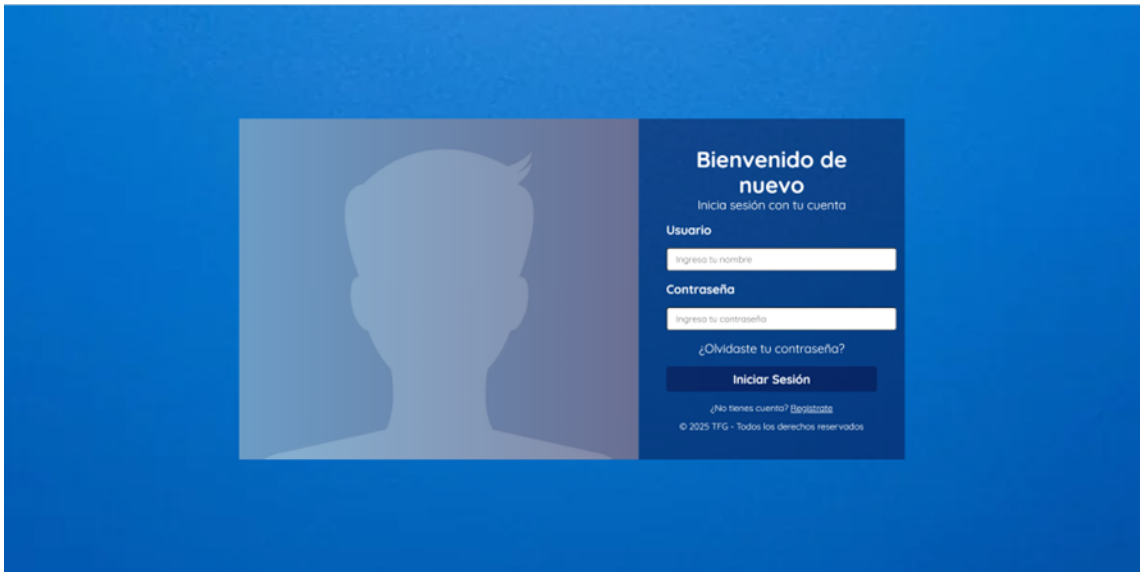

Figura 5.3: Login Primer Incremento

La página de registro, similar a esta, añade más campos para introducir y mantiene el diseño unificado.

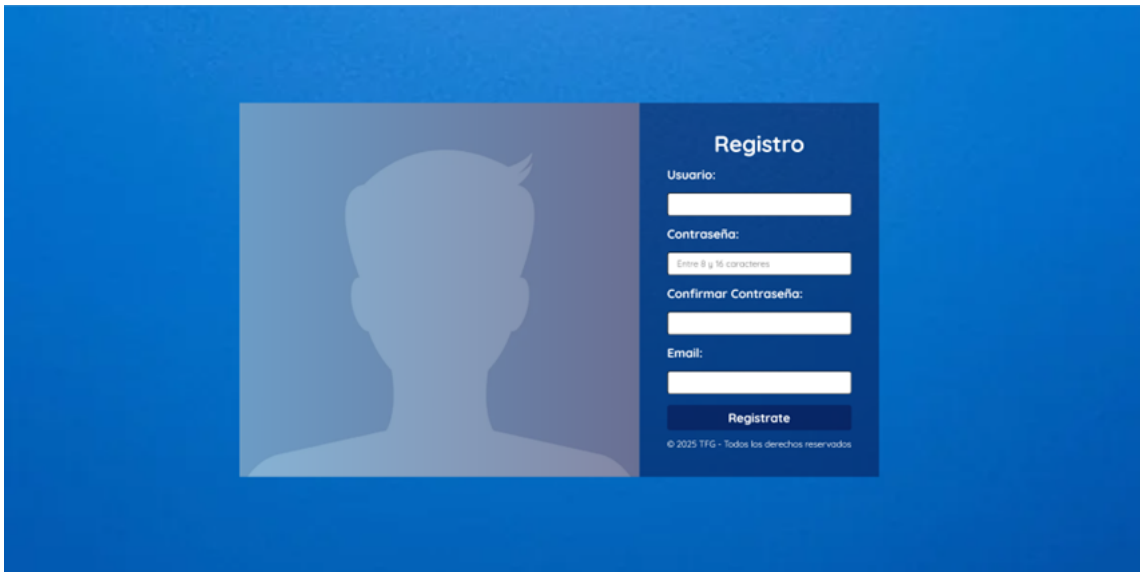

Figura 5.4: Registro Primer Incremento

Una vez iniciamos sesión iríamos a la página inicial o panel de usuario. Donde el usuario podría visualizar los test realizados junto con los resultados en ellos de forma visual y además podría acceder a la página de realizar tests. También nos encontramos un menú en la parte superior, invariante en toda la web, donde el usuario puede cerrar sesión, volver al inicio o ir a su perfil, página aún no implementada en esta versión. En los botones y contenedores de contenido nos encontramos con animaciones unificadas a lo largo de la página.

Figura 5.5: Página Inicial Primer Incremento

Una vez accedemos a ver los test, nos encontramos con una pantalla que muestra los test disponibles y la dificultad que suponen de forma colorida y visual. Mantenemos el diseño de la página de la anterior, que a su vez se mantendrá constante en toda la app.

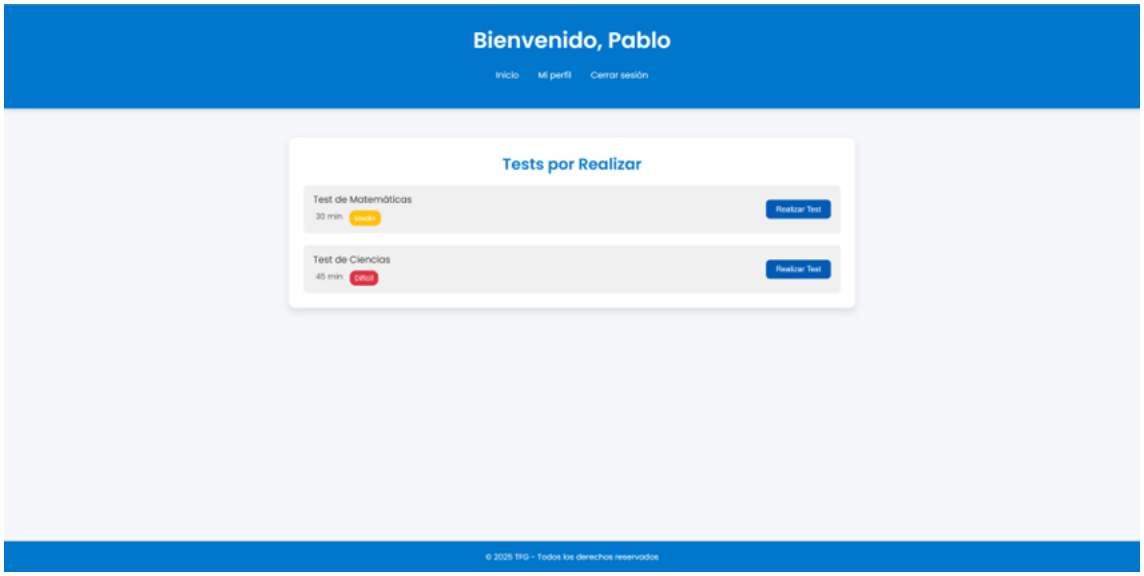

Figura 5.6: Tests Disponibles Primer Incremento

Podemos pulsar realizar test que nos redirigirá a la página del test, donde tendremos un tiempo limitado indicado en la parte superior para leer o visualizar el contenido del que realizaremos posteriormente preguntas.

Figura 5.7: Documento Primer Incremento

Una vez finalice el tiempo, o quiera el usuario(no implementado en esta versión), se procederá a cambiar a la página del test donde el usuario tendrá también tiempo para contestar las preguntas. En esta página aún no se ha implementado el tiempo.

Figura 5.8: Examen Primer Incremento

5.2. Segundo Incremento

Esta iteración del desarrollo del producto software se centró principalmente en las funciones de la parte del *back-end* y el desarrollo de la base de datos.

Se comenzó con la implementación de un sistema de gestión de usuarios, donde se creó inicialmente una tabla üsers.[en] la base de datos que incluía inicialmente los campos: id, nombre usuario, contraseña, nombre y apellido. Se decidió no guardar el email del usuario, pues al ser una aplicación destinada a entrevistas no se consideró necesario, ya que se mandarían unas credenciales predeterminadas para cada usuario que se deseara.

	id	username	password	firstname	lastname	role	id_grupo
▶	402	Pablo1	$2a$10$g3UgHUqEvjTxLBRqTn1/WeTvO/.64w6...	Pablo	Palenzuela	1	2
	552	Pablo2	$2a$10$2kfKZzr0Ma.Uh1RjuB3n8e.3YPEiYb3OH...	Pablo	Palenzuela	1	1

Figura 5.9: Tabla Users Segundo Incremento

Una vez creado la base de datos y la entidad en el *back-end* junto con los getters y setters, se procedió a realizar todos los temas relacionados con la seguridad en la aplicación. Para ello, se utilizó JJWT (Java JSON Web Token) comentado anteriormente en 2.5.1. Al iniciar sesión, se genera un token para el usuario con tiempo de expiración; este token se utiliza para validar la sesión en todas las páginas, además de asegurarse si la sesión está o no caducada. De este token además, se extrae toda la información necesaria del usuario.

En Spring, las contraseñas se encriptan utilizando un componente llamado `PasswordEncoder`. Este componente proporciona diferentes algoritmos de hash y herramientas para encriptar contraseñas de forma segura. La implementación usada es `BCryptPasswordEncoder`, que utiliza el algoritmo `BCrypt`, diseñado específicamente para proteger contraseñas.

- `BCrypt` genera un hash único para cada contraseña.
- Utiliza un *salt* (valor aleatorio) interno que se añade automáticamente. Esto asegura que incluso si dos usuarios tienen la misma contraseña, los hashes generados serán diferentes.
- `BCrypt` incluye un factor de coste configurable que controla cuántas iteraciones se realizan en el proceso de hash, lo que afecta al tiempo necesario para calcular el hash y aumenta la resistencia a ataques de fuerza bruta.
- Durante la autenticación, `BCryptPasswordEncoder` compara el hash almacenado con la contraseña ingresada por el usuario (después de hashearla nuevamente).

Tras realizar todos los aspectos relacionados con el usuario, los esfuerzos se centraron en la implementación del sistema de grupos.

Para ello, se añadió un nuevo campo a cada usuario que se denominaba *idgrupo*. Esta id se decidió que se asignaría de manera aleatoria, preparándolo ya para un posible futuro uso en el que, según ciertas características, el propietario de la aplicación pueda asignarlo como prefiera. Una vez asignados los grupos, se creó la tabla de documentos, que guarda el id, el nombre, el tamaño, el tipo de contenido, el grupo, la complejidad, el tiempo necesario para su lectura y el propio documento.

	id	file_name	content_type	file_size	file_content	id_grupo	file_complex	file_time
▶	1	FVE870KWBZ8W.pdf	Test Ciencias	248832	BLOB	2	1	600
	2	Evaluacion Usabilidad	Test Usabilidad	1075523	BLOB	2	2	1000

Figura 5.10: Tabla Documentos Segundo Incremento

El documento, como se puede ver en la imagen, se guarda como archivo binario o blob para posteriormente ser decodificado en el front. El tiempo se guarda en segundos, aunque luego es convertido a minutos en el propio front y poder configurar así el cronómetro. Para visualizar los documentos asociados al usuario se tiene en cuenta el idgrupo.

Tras esto, se conectó todo con el *front-end* y se realizaron pruebas para comprobar el correcto funcionamiento. Pudiendo ya visualizar con datos almacenados en la base de datos los tests disponibles del usuario según su grupo y el documento asociado a dicho test.

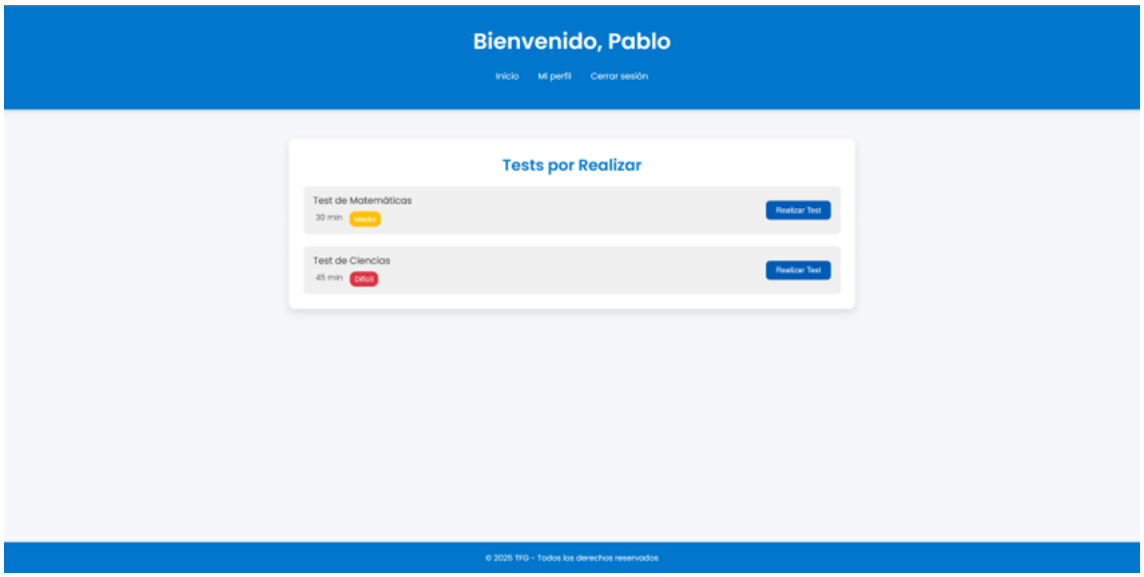

Figura 5.11: Lista Documentos Segundo Incremento

La parte visual no cambió, por lo que es igual al del primer incremento, salvo que los datos son reales.

Una vez todo lo referido a los documentos y al usuario estuvo correctamente configurado y conectado con el *front-end*, se comenzó con la creación de la tabla de **Preguntas**. Los campos de preguntas son id de la pregunta, id del fichero y la pregunta en si.

id_pregunta	id_file	pregunta
1	1	¿Cuáles son países europeos?
2	1	¿Cuál es un número primo?
3	2	¿Cuál es el objetivo principal del prototipo?

Figura 5.12: Tabla Preguntas Segundo Incremento

A su vez se creó la tabla **Respuestas**, donde se almacenan las respuestas asociadas a cada una de las preguntas. Esta tabla cuenta con id de la pregunta y la propia respuesta en si.

test_id_pregunta	respuestas
1	Francia, Brasil, Alemania, Japón
2	4, 7, 9, 12

Figura 5.13: Tabla Respuestas Segundo Incremento

En referencia a la tabla respuestas se creó una nueva tabla **Correctas**. En la que se almacenan las respuestas correctas de cada una de las preguntas. En esta tabla se almacena el id de la pregunta y la respuesta correcta.

test_id_pregunta	correctas
1	0, 2
2	1

Figura 5.14: Tabla Respuestas Segundo Incremento

El motivo principal para esta separación es cumplir con las mejores prácticas de diseño de bases de datos relacionales, enfocándose en evitar redundancias, mejorar el rendimiento y facilitar la escalabilidad futura. Una vez creadas todas las tablas y las entidades necesarias en el *back-end* se crearon los métodos necesarios para la obtención de las preguntas desde el front mediante las bibliotecas de Angular. Obteniendo así el test asociado a cada uno de los documentos, al igual que las respuestas correctas. El test implementado era de respuesta múltiple o única, con la posibilidad de añadir preguntas de completar. La parte visual no cambió y se mantuvo como en el primer incremento.

Ahora faltaba almacenar los resultados y la lógica detrás de la corrección. Para ello se creó una nueva tabla denominada **Resultados**. Esta tabla almacena id propio, el id del usuario, el id del documento y el resultado.

id	id_usuario	id_file	resultado
1	1	5	85.75
3	1	5	85.75

Figura 5.15: Tabla Resultados Segundo Incremento

Tras esto, se conectó todo con el *front-end* permitiendo ya almacenar resultados y poder ver en la página principal los resultados de los test. Pero era necesario también poder corregir los test. Para ello se realizó una función desde el *front-end* con Typescript que, mediante la obtención de las respuestas correctas y las propias respuestas enviadas por el usuario, procesa de manera automática ,a la hora de finalizar el test, el resultado obtenido en el test y lo manda al *back-end* añadiéndolo entonces a la tabla resultados.

Debido a esto, en la página principal ya podía verse los resultados obtenidos en los test anteriores, aunque era necesario cambiar la lógica de obtención de test, ya que no debían mostrarse los test ya realizados. Para ello se modificó el método *GET* del back y, mediante consulta a la base de datos, con el idusuario y el idgrupo se ve qué documento le corresponde al usuario según el grupo y si no está en la tabla resultados se pasa en dicho *GET*. Con esto finalizó el segundo incremento, donde quedó finalizada toda la parte del *back-end* y de la base de datos de la aplicación. Quedando solo ya retoques para la parte visual y alguna página más en el *front-end*.

5.3. Tercer Incremento

El tercer y último incremento de la aplicación se basó enteramente en retoques visuales para la página y en la creación de la página **Mi Perfil**. Además, se añadieron diversos mensajes de error y se realizaron cambios menores en diseño relacionados con botones o transiciones. Se incluyeron mensajes de error a la hora de iniciar sesión o registrarse si no estaban todos los campos rellenos.

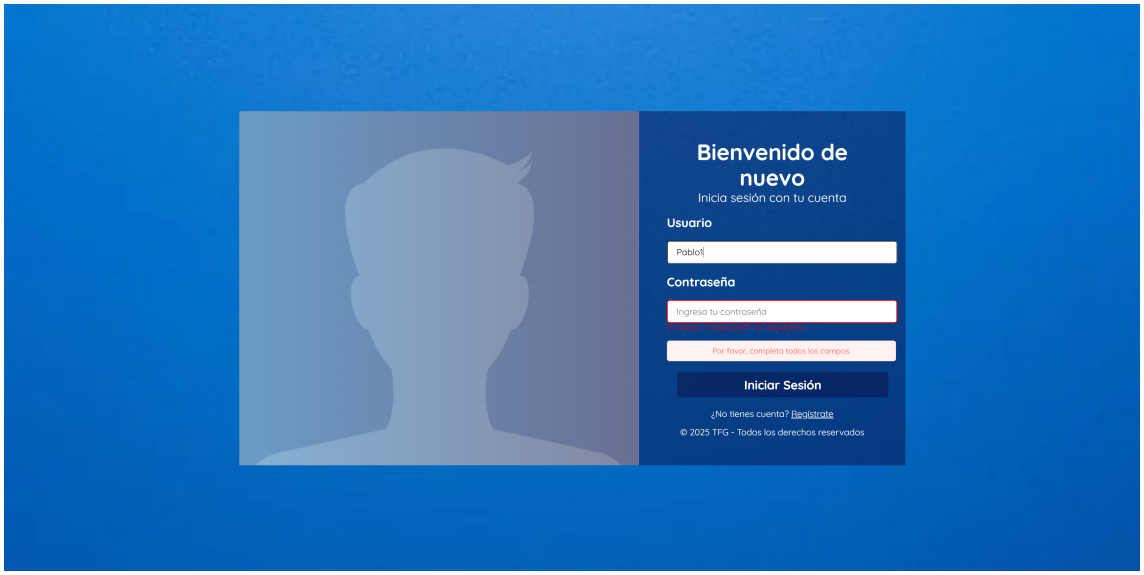

Figura 5.16: Iniciar Sesión con Campo No Completado Tercer Incremento

Además, se añadió un mensaje de error al intentar iniciar sesión con credenciales inválidas.

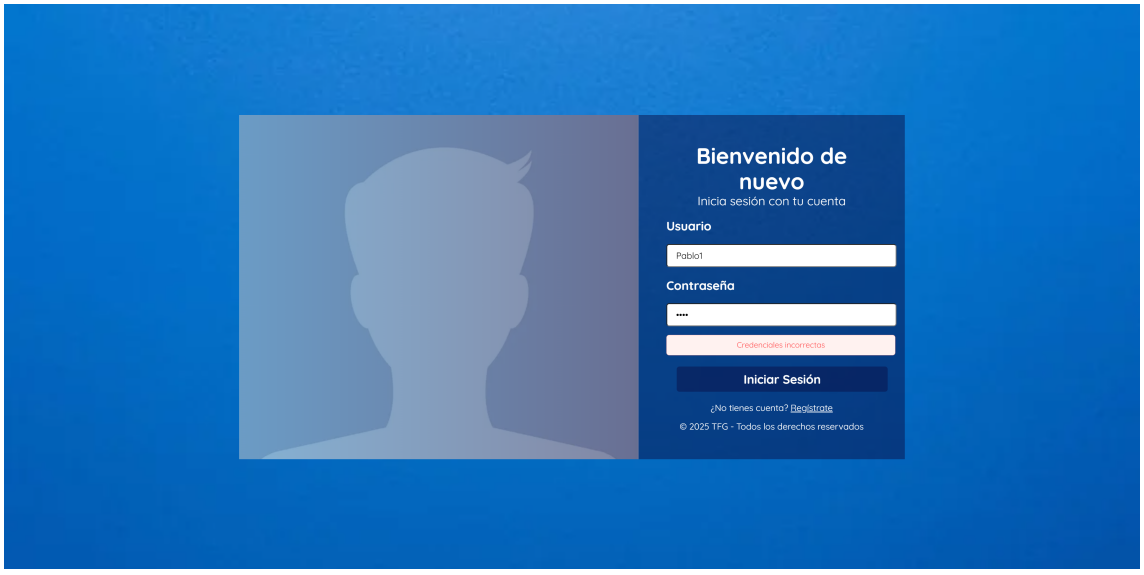

Figura 5.17: Iniciar Sesión con Credenciales Incorrectas Tercer Incremento

En lo referido a la página de **Mi Perfil**, nos encontramos con una página que muestra los datos personales de interés del usuario. Además, muestra el número de test realizados, al igual que un diagrama de sectores para visualizar de forma vistosa el porcentaje de preguntas acertadas y erróneas.

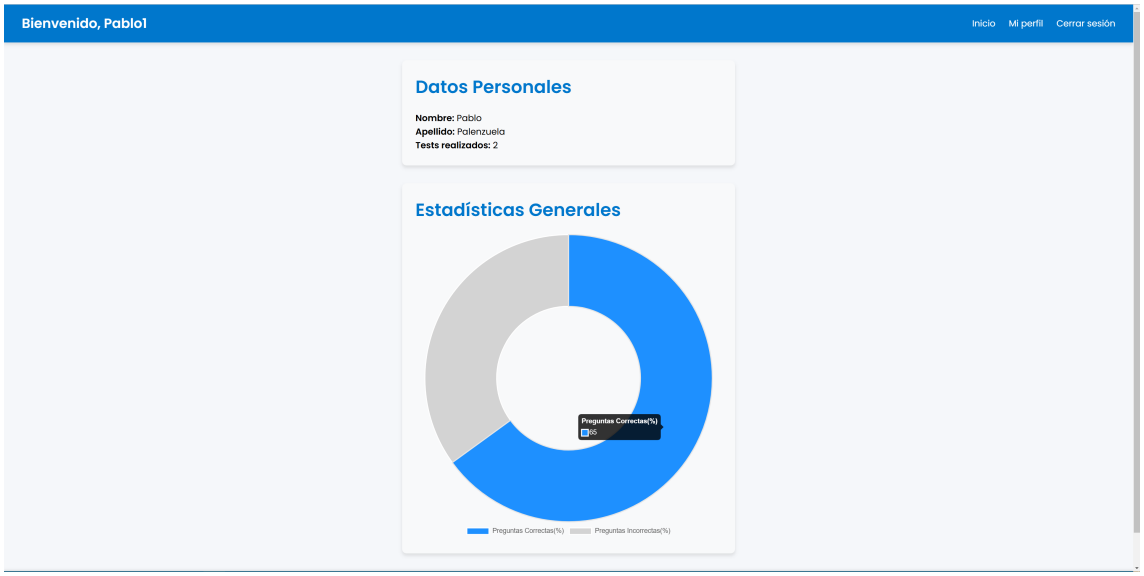

Figura 5.18: Mi Perfil Tercer Incremento

Capítulo 6

Demo

En este capítulo, se explica detalladamente la forma de uso de la herramienta desarrollada en este trabajo de fin de grado. Se muestra el proceso desde la pantalla inicial, la realización de un test y la obtención de los resultados,incluyendo imágenes para ilustrar cada paso del proceso.

6.1. Pantalla Principal

Al inicio de la aplicación, se accede a la pantalla principal, donde el usuario debe iniciar sesión con sus credenciales o registrarse con las credenciales que desee.

Pasos a seguir:

1. Completar el Formulario de Inicio de Sesión:
   a) Introducir el nombre de usuario en el campo correspondiente.
   b) Introducir la contraseña en el campo correspondiente.
2. Pulsar el botón “Iniciar Sesión” para iniciar el proceso de verificación.

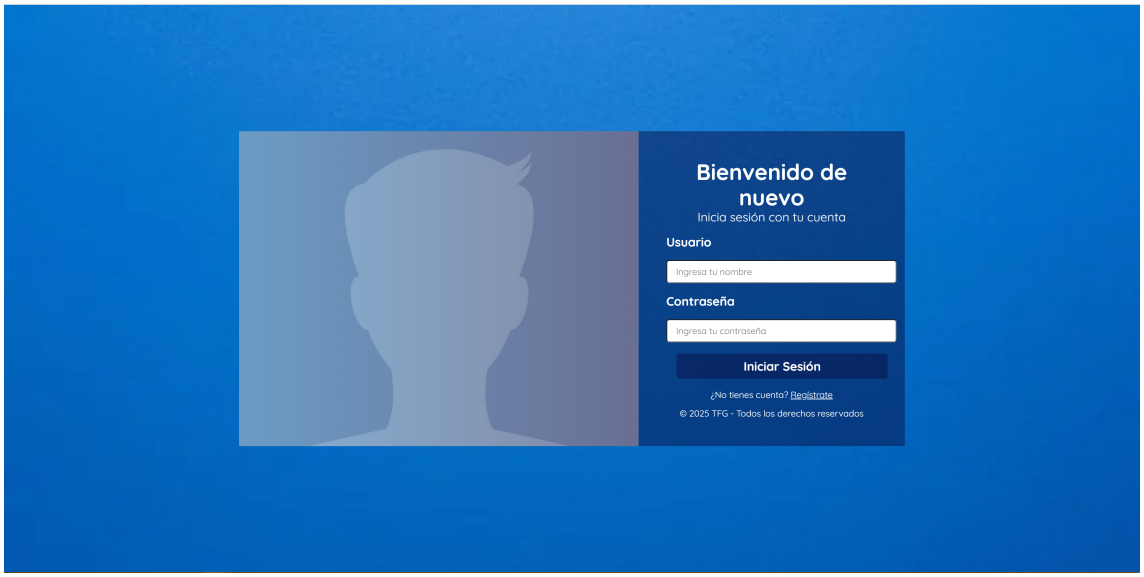

Figura 6.1: Pantalla Login

Pasos a seguir en caso de tener que registrarse:

1. Completar el Formulario de Inicio de Sesión:
   a) Introducir el nombre de usuario en el campo correspondiente.
   b) Introducir la contraseña en el campo correspondiente.
   c) Introducir la confirmación de la contraseña en el campo correspondiente.
   d) Introducir nombre y apellido.
2. Pulsar el botón “Regístrate” para iniciar el proceso de creación de usuario.

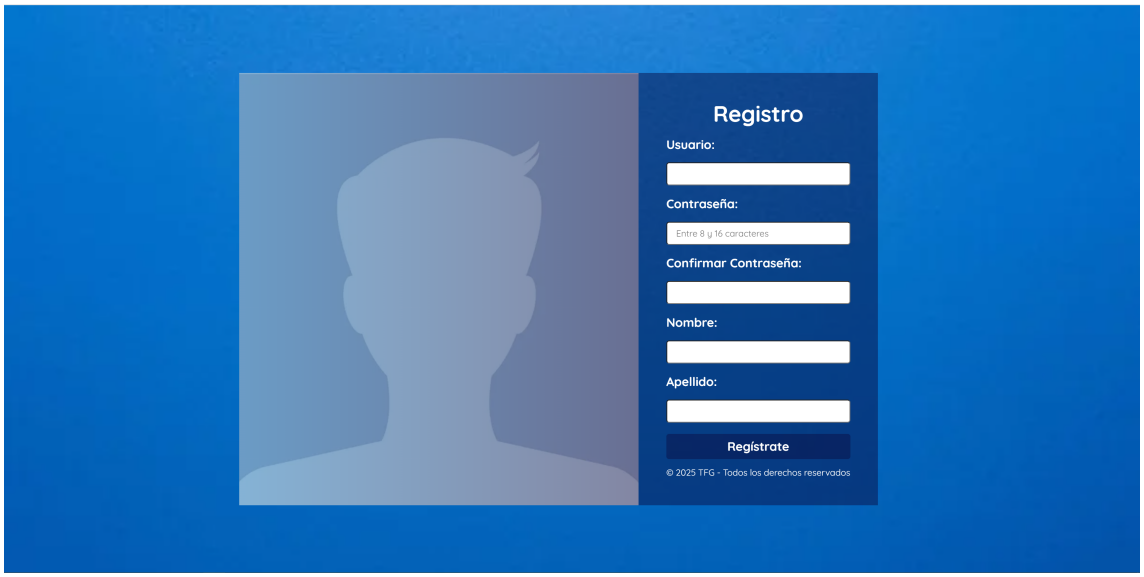

Figura 6.2: Pantalla de Registro

6.2. Pantalla de Inicio

Una vez realizado el proceso de verificación, se tiene acceso la pantalla de inicio. En ella, se muestra la siguiente información:

- Menú en la parte superior, que podemos encontrar en todas las páginas. Con accesos a “mi perfil”, a la pantalla de inicio (en la que estamos) y a “cerrar sesión”.
- Test por realizar: Muestra el numero de test que falta por realizar, junto con el botón de “Ver test” que nos llevaría a la siguiente página.
- Lista de Resultados: lista de resultados obtenidos por el usuario en los test. El botón de “Mostrar resultados” lleva a la misma página que “Mi Perfil” en la que encontramos estadísticas del usuario.
- Footer de la página: footer de la página que muestra el propietario de la aplicación.

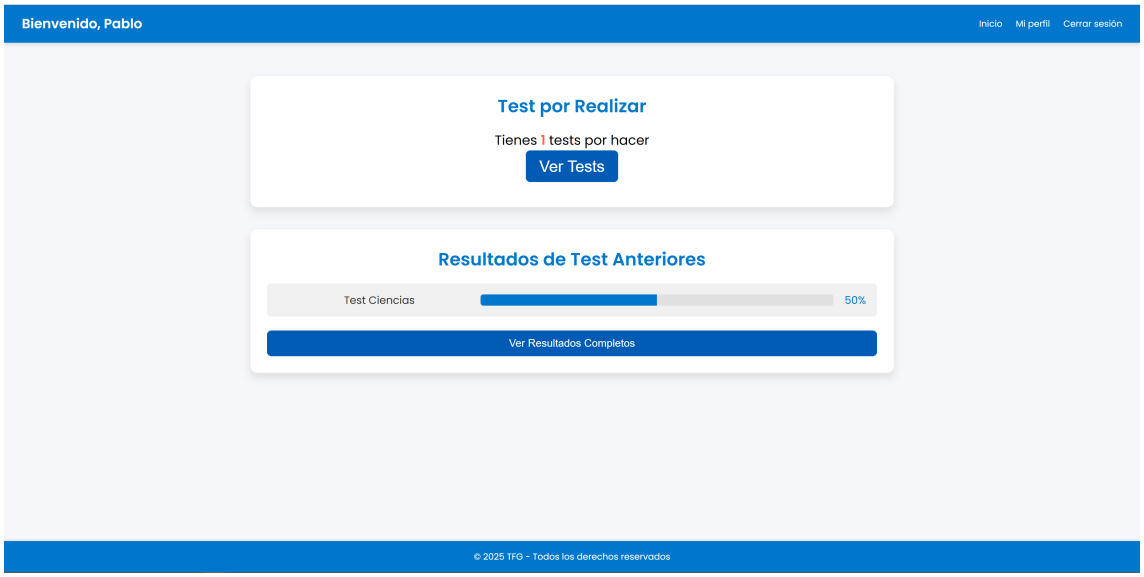

Figura 6.3: Tabla Inicio

6.3. Pantalla de Test

Tras pulsar el botón “Ver Test”, se tiene acceso a una pantalla en la que se muestran los tests disponibles. En ella, se muestra la siguiente información:

- Menú en la parte superior, que podemos encontrar en todas las páginas. Con accesos a “mi perfil”, a la pantalla de inicio (en la que estamos) y a “cerrar sesión”.
- Test por realizar: Muestra los tests que faltan por realizar, con información como su dificultad o el tiempo necesario para realizarlo, junto con el botón de “Realizar test” que nos llevaría a la siguiente página.
- Footer de la página: pie de página que muestra el propietario de la aplicación.

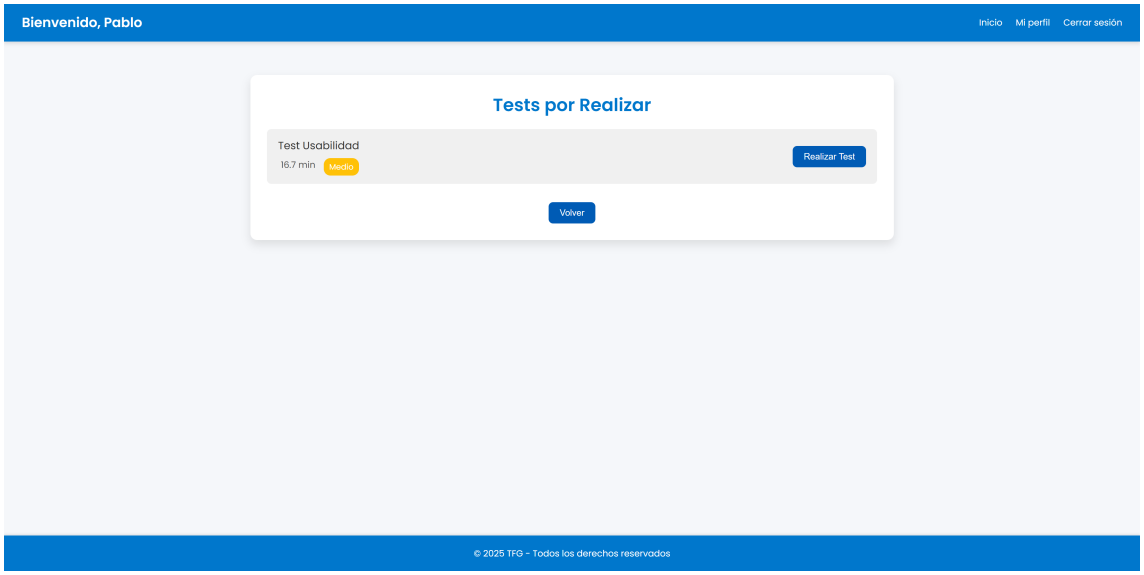

Figura 6.4: Página Tests

6.4. Pantalla de Documento

Tras pulsar el botón “Realizar Test”, se tiene acceso a una pantalla en la que se muestra el documento a leer. En ella, se muestra la siguiente información:

- Menú en la parte superior, que podemos encontrar en todas las páginas. Con accesos a “mi perfil”, a la pantalla de inicio (en la que estamos) y a “cerrar sesión”.
- Tiempo que falta para finalizar la lectura del documento.
- El documento en formato pdf para su lectura.
- Botón “Ir al examen” que tras pulsarlo nos pide un mensaje de confirmación para ir al test.
- Footer de la página: pie de página que muestra el propietario de la aplicación.

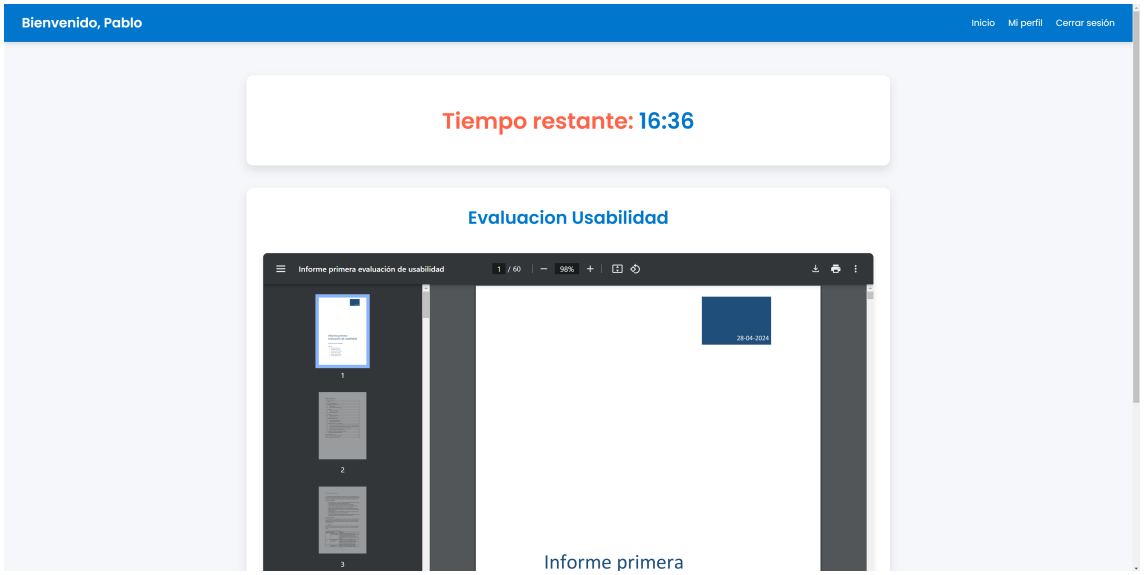

Figura 6.5: Página Documento

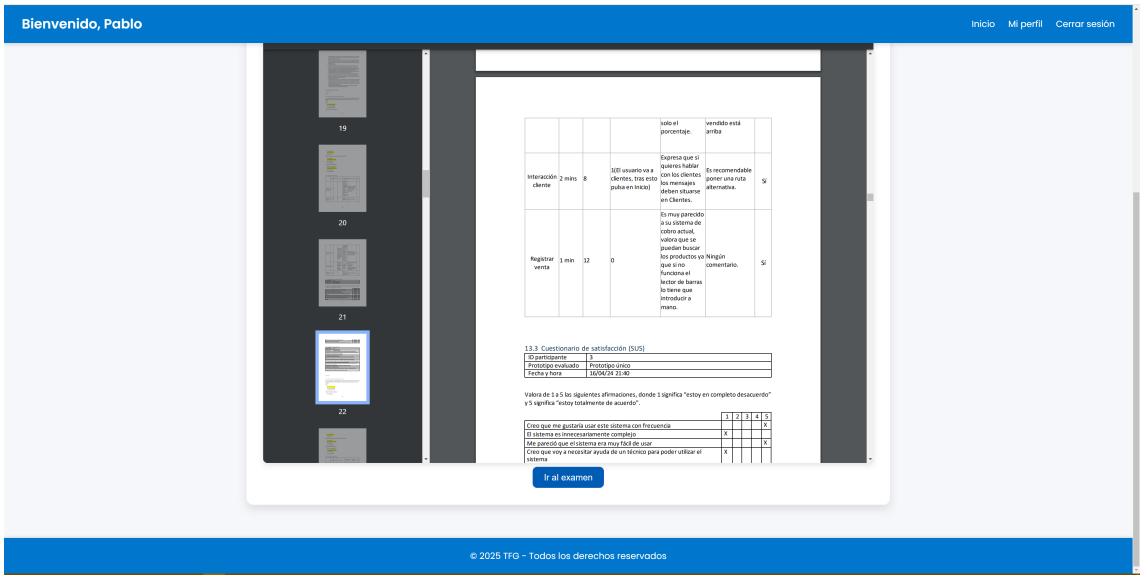

Figura 6.6: Página Documento

6.5. Pantalla de Examen

Tras pulsar el botón “Ir al Examen”, se tiene acceso a una pantalla en la que se muestra el examen a realizar. En ella, se muestra la siguiente información:

- Menú en la parte superior, que podemos encontrar en todas las páginas. Con accesos a “mi perfil”, a la pantalla de inicio (en la que estamos) y a “cerrar sesión”.

- Las preguntas numeradas una a una con las respuestas posibles. Se deja una casilla para marcar.
- Botón “Enviar test” que tras pulsarlo nos pide un mensaje de confirmación.
- Footer de la página: pie de página que muestra el propietario de la aplicación.

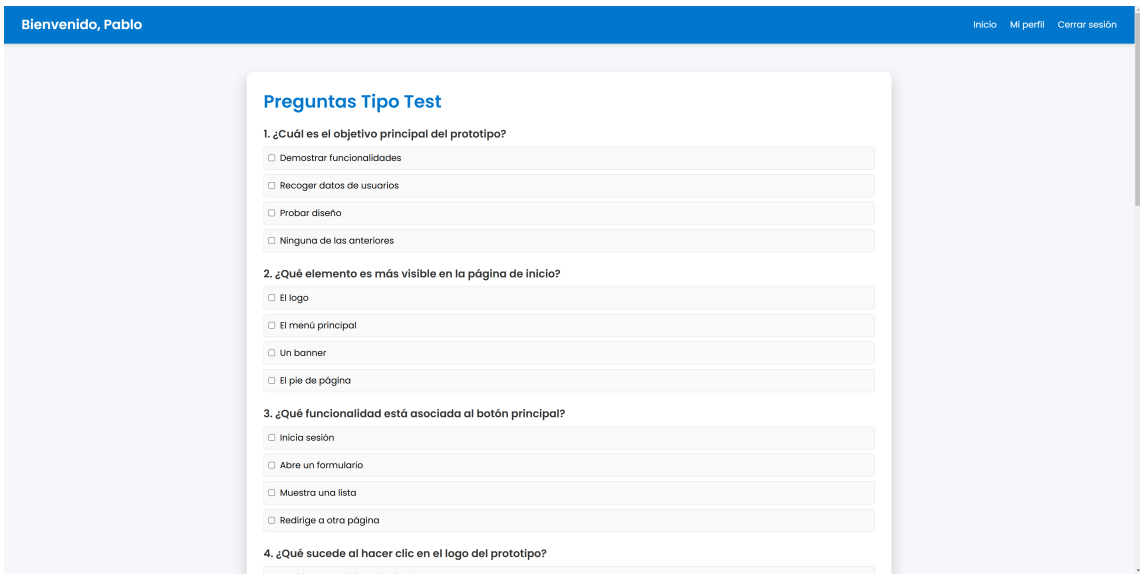

Figura 6.7: Página Examen

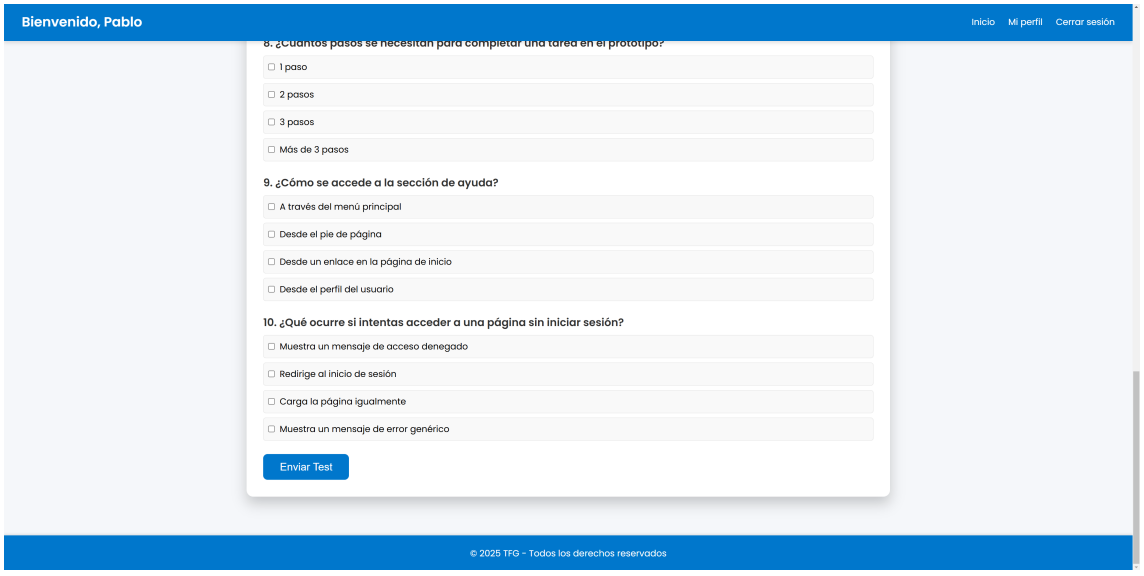

Figura 6.8: Página Examen

Tras pulsar el botón “Enviar Test” nos redirige a la página de inicio donde el resultado de nuestro test se habrá añadido a “Resultados de Test Anteriores” y ya no mostrará el test en ninguna pantalla.

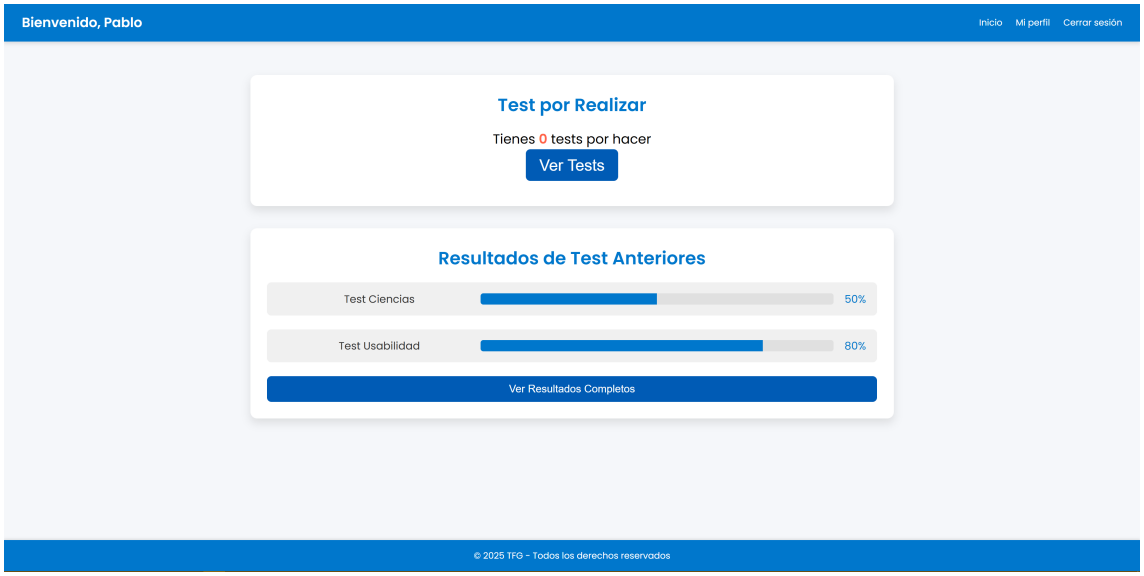

Figura 6.9: Página Inicio tras Test

En la página principal podríamos acceder entonces a **Mi Perfil**, ya sea desde la barra de navegación o pulsando en "Ver Resultados Completos", donde encontraremos una página con información y estadísticas generales del usuario.

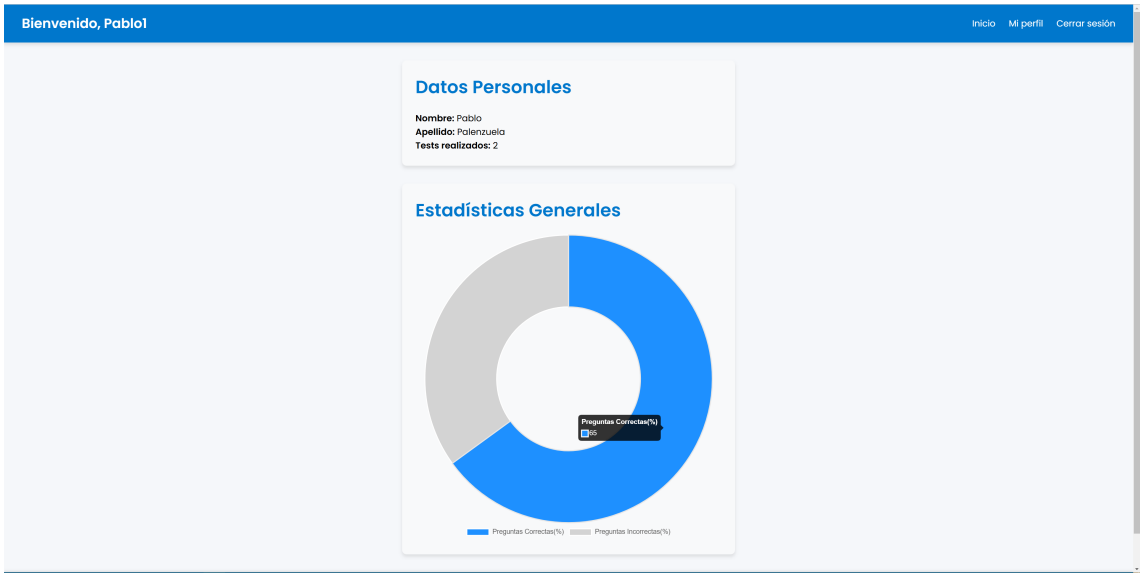

Figura 6.10: Página Mi Perfil tras Test

Capítulo 7

Análisis de Impacto

En este capítulo se analiza el impacto del desarrollo de la herramienta propuesta en este trabajo de fin de grado.

A continuación, se detallan los beneficios primordiales que genera la herramienta.

- **Eficiencia en la recopilación de datos**: La herramienta elimina la necesidad de realizar entrevistas manualmente, con lo que ahorra tiempo y esfuerzo. Además al estar la gestión centralizada, reduce errores humanos en la recopilación manual de datos.
- **Mejora en la Calidad de los Estudios:** La herramienta tiene resultados más precisos debido a la evaluación automática que evita sesgos subjetivos y asgura uniformidad en las correcciones.También al no depender de la disponibilidad física de los entrevistadores facilita el llegar a un mayor número de participantes.
- **Aplicación en múltiples sectores**: Aunque en un principio está pensada para evaluaciones de usabilidad y poder ver la percepción de los usuarios sobre productos o servicios e impulsar el desarrollo de soluciones, también podría ser utilizada para la investigación académica en estudios que requieran análisis empírico de datos, como investigaciones educativas o psicológicas, o incluso en formación y educación que podría llegar a usarse para evaluar el progreso de los estudiantes, adaptando pruebas según su nivel o necesidades.
- **Ahorro de Recursos**: Al automatizar procesos como la evaluación o almacenamiento de resultados se reducen recursos necesarios, como personal y materiales físicos.
- **Accesibilidad**: Gracias al acceso remoto podría realizarse pruebas a cualquier usuario y además, con el sistema de grupos, facilita la inclusión de grupos específicos con pruebas diseñadas para ellos, lo que ayuda a personas con necesidades específicas.
- **Mejora en la toma de decisiones**: Dado que los resultados se almacenan

en la base de datos permite generar informes y estadísticas que facilitan la toma de decisiones rápida y de forma eficaz basada en datos objetivos. Además, estos datos pueden usarse para ajustar o mejorar futuras pruebas.

7.1. Objetivos de Desarrollo Sostenible

7.1.1. Vinculación con el ODS 4: Educación de calidad

La herramienta contribuye a garantizar una educación inclusiva, equitativa y de calidad, al proporcionar una plataforma accesible y digitalizada que permite a los usuarios aprender y evaluarse de manera eficiente. Mediante el uso de contenido multimedia y evaluaciones automatizadas, se promueve una experiencia educativa enriquecedora y adaptada a las necesidades de los participantes. Esto fomenta el aprendizaje continuo y mejora las oportunidades de desarrollo personal y profesional.

7.1.2. Vinculación con el ODS 9: Industria, Innovación e Infraestructura

El desarrollo de esta herramienta fomenta la innovación tecnológica al aplicar enfoques modernos como la automatización de evaluaciones y el análisis de datos mediante plataformas digitales. Además, al integrar tecnologías como Angular, Spring Boot o Hibernate, se establece una infraestructura tecnológica robusta que puede ser adaptada y utilizada en diferentes sectores, desde la industria educativa hasta el análisis empresarial, contribuyendo a la modernización de los sistemas tradicionales, como mencionamos anteriormente.

7.1.3. Vinculación con el ODS 10: Reducción de las desigualdades

La accesibilidad de la herramienta, que permite el acceso remoto desde cualquier lugar, ayuda a reducir las desigualdades al ofrecer la posibilidad de participación a personas que de otro modo podrían no tener acceso a evaluaciones o capacitaciones similares; además, con el sistema de grupos, podrían preparar entrevistas específicas para personas con necesidades especiales.

7.1.4. Vinculación con el ODS 12: Producción y consumo responsables

El uso de tecnologías digitales para gestionar pruebas y evaluaciones elimina la necesidad de materiales físicos como papel, reduciendo así el impacto ambiental asociado a estos procesos. Este enfoque sostenible promueve un consumo responsable de recursos y apoya prácticas que minimizan el impacto ambiental, alineándose con los objetivos de sostenibilidad.

7.1.5. Vinculación con el ODS 17: Alianzas para lograr los objetivos

La herramienta facilita la colaboración entre instituciones educativas, empresas y otras entidades, fomentando sinergias que impulsan la investigación, la innovación y el desarrollo de soluciones tecnológicas. Estas alianzas fortalecen la conexión entre el ámbito académico y el profesional, promoviendo el intercambio de conocimientos y recursos que son esenciales para alcanzar los objetivos de desarrollo sostenible.

Capítulo 8

Conclusiones y Futuras Líneas de Trabajo

8.1. Conclusiones

En este trabajo de fin de grado se ha desarrollado una herramienta innovadora diseñada para optimizar la realización de entrevistas y pruebas relacionadas con productos de software. La herramienta automatiza procesos clave, como la generación inmediata de resultados y la retroalimentación al usuario, lo que se traduce en la obtención de conclusiones más rápidas y precisas.

Un aspecto destacado del sistema desarrollado es la implementación de un sistema de gestión de grupos y contenido multimedia. Este sistema permite asignar a los usuarios a grupos específicos en función de los objetivos de las pruebas, lo que facilita las entrevistas. Además, garantiza que cada grupo tenga acceso al contenido multimedia adecuado y lo visualice durante un tiempo limitado antes de responder a las preguntas asociadas. Esta funcionalidad asegura que las pruebas sean más relevantes y ajustadas a los objetivos del análisis, mejorando la calidad y utilidad de los resultados obtenidos.

El propósito principal de esta solución ha sido facilitar y mejorar la ejecución de pruebas en el contexto de evaluación de requisitos de productos de software. Este proceso, que tradicionalmente resulta tedioso y requiere una inversión considerable de tiempo y recursos, se ve significativamente optimizado gracias a la automatización implementada. Además, también contribuye a que los desarrolladores identifiquen y corrijan errores en sus requisitos de forma más ágil, mejorando así la calidad final de sus proyectos.

En definitiva, este trabajo representa un paso hacia la modernización y optimización de los métodos tradicionales de evaluación, proporcionando una solución práctica que puede ser aplicada en diversos contextos tecnológicos. La combinación de automatización, gestión personalizada de grupos y análisis eficiente hace que esta herramienta sea una valiosa contribución al desarrollo de procesos de evaluación más eficientes y productivos.

8.2. Futuras Líneas de Trabajo

Pese a que la herramienta implementada es funcional y cumple con su cometido inicial, hay cierto margen de mejora.

En primer lugar, el despliegue de la herramienta en un servidor para su uso sería esencial para poder ser utilizada de forma correcta. Ya que, a la hora de realizar entrevistas, es trivial que los usuarios puedan acceder desde internet a dicha herramienta.

En lo referido a la herramienta en sí, cabría la posibilidad de implementar o ampliar el sistema de usuarios pudiendo otorgar derechos de administrador a ciertos usuarios. Que si bien, la posibilidad ya esta creada, no hay ninguna diferencia entre ellos. Esto podría facilitar al administrador a añadir contenido de forma más sencilla a la base de datos desde la propia aplicación o decidir de que forma dividir a los usuarios entrevistados sin necesidad de especialización o alterar de forma directa la base de datos.

Cabe destacar también, una posible implementación de IA que de su propia opinión de los resultados obtenidos en los test de tal forma que pueda ayudar al desarrollador a obtener conclusiones de forma más rápida. Sin embargo, esto podría acarrear también conclusiones imprecisas además de ser complicado de desarrollar.

Bibliografía

[1] (2021) Java spring-boot. [Online]. Available: https://www.ibm.com/es-es/topics/java-spring-boot

[2] (2024) What is maven? [Online]. Available: https://maven.apache.org/what-is-maven.html

[3] J. M. A. Aguín. (2022) Java: ¿qué es maven? ¿qué es el archivo pom.xml? [Online]. Available: https://www.campusmvp.es/recursos/post/java-que-es-maven-que-es-el-archivo-pom-xml.aspx

[4] ¿qué es mysql? [Online]. Available: https://www.oracle.com/es/mysql/what-is-mysql/

[5] Hibernate orm documentation. [Online]. Available: https://hibernate.org/orm/documentation/

[6] Lombok documentation. [Online]. Available: https://projectlombok.org

[7] Angular documentation. [Online]. Available: https://angular.io/docs

[8] (2023) Introduction to typescript. [Online]. Available: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html

[9] Github documentation. [Online]. Available: https://docs.github.com

Anexos

Capítulo 9

Descripción de Entidades y Métodos del Sistema

En este capítulo, se detalla la estructura y funcionalidad de las entidades principales utilizadas en el desarrollo del sistema, así como los métodos expuestos a través de la API REST para interactuar con dichas entidades. Esta información es fundamental para comprender la arquitectura y la lógica del sistema, ya que establece cómo se organizan los datos y cómo se gestionan las operaciones sobre ellos.

9.1. Entidad User

La entidad **User** representa a los usuarios del sistema. Esta entidad es fundamental para la autenticación y gestión de usuarios dentro de la aplicación. Además, permite la asignación de usuarios a grupos específicos y la realización de pruebas.

9.1.1. Estructura de la Entidad

La entidad **User** está compuesta por los siguientes campos:

- **id**: Identificador único del usuario (`Integer`).
- **username**: Nombre de usuario único utilizado para la autenticación (`String`).
- **password**: Contraseña cifrada del usuario (`String`).
- **firstname**: Nombre del usuario (`String`).
- **lastname**: Apellido del usuario (`String`).
- **id_grupo**: Identificador del grupo al que pertenece el usuario (`Integer`).
- **role**: Rol asignado al usuario (por ejemplo, `ADMIN` o `USER`).

9.1.2. Relaciones

- Relación con la entidad **Group**: El campo `id_grupo` permite asignar al usuario a un grupo específico.
- Relación con la entidad **Result**: Permite visualizar los resultados de las pruebas realizadas por el usuario.

9.1.3. Métodos User

El sistema expone varios métodos relacionados con la entidad **User** para realizar operaciones a través de la API REST:

Método POST: Registro de Usuarios

- **URL:** `/api/auth/register`
- **Descripción:** Permite registrar un nuevo usuario en el sistema.
- **Parámetros:**
  - `username`: Nombre de usuario.
  - `password`: Contraseña del usuario.
  - `firstname`: Nombre del usuario.
  - `lastname`: Apellido del usuario.
  - `id_grupo`: Grupo al que pertenece el usuario.
- **Respuesta:** Confirmación del registro exitoso o error en caso de conflicto (por ejemplo, si el nombre de usuario ya existe).

Método POST: Inicio de Sesión

- **URL:** `/api/auth/login`
- **Descripción:** Permite a un usuario autenticarse en el sistema.
- **Parámetros:**
  - `username`: Nombre de usuario.
  - `password`: Contraseña del usuario.
- **Respuesta:** Un token JWT en caso de autenticación exitosa o un mensaje de error en caso contrario.

Método GET: Obtener Información del Usuario Autenticado

- **URL:** `/api/auth/user`
- **Descripción:** Devuelve la información del usuario actualmente autenticado.

- **Parámetros:** Ninguno (requiere autenticación con un token JWT).
- **Respuesta:**
  - `id`: Identificador único del usuario.
  - `username`: Nombre de usuario.
  - `firstname`: Nombre del usuario.
  - `lastname`: Apellido del usuario.
  - `id_grupo`: Identificador del grupo al que pertenece.

9.2. Entidad Files

La entidad **Files** representa los archivos multimedia que son utilizados como contenido en las pruebas dentro del sistema. Estos archivos están asociados a un grupo específico y cuentan con propiedades que permiten gestionar su complejidad y tiempo asignado.

9.2.1. Campos principales de la entidad

- **id**: Identificador único del archivo.
- **fileName**: Nombre del archivo.
- **contentType**: Tipo de contenido.
- **fileSize**: Tamaño del archivo en bytes.
- **fileContent**: Contenido del archivo almacenado como un blob de datos.
- **idGrupo**: Identificador del grupo al que pertenece el archivo.
- **fileComplex**: Nivel de complejidad asociado al archivo.
- **fileTime**: Tiempo asignado al archivo en segundos.

Esta estructura permite que los archivos se almacenen, clasifiquen y utilicen en las pruebas de manera eficiente y organizada.

9.2.2. Métodos Files

El sistema expone varios métodos relacionados con la entidad **Files** para realizar operaciones a través de la API REST:

Obtener archivos no realizados por el usuario

- **Endpoint**: `GET /api/files/user/{idUsuario}/notInResults`
- **Descripción**: Devuelve una lista de archivos que aún no han sido completados por el usuario con el ID especificado.
- **Parámetros**:

  - `idUsuario`: ID del usuario que consulta los archivos.
- **Respuesta**: Lista de objetos `Files` que representan los archivos disponibles.

Obtener archivos realizados por el usuario

- **Endpoint**: `GET /api/files/user/{idUsuario}/inResults`
- **Descripción**: Devuelve una lista de archivos que ya han sido completados por el usuario con el ID especificado.
- **Parámetros**:
  - `idUsuario`: ID del usuario que consulta los archivos.
- **Respuesta**: Lista de objetos `Files` que representan los archivos realizados.

Obtener archivo por su ID

- **Endpoint**: `GET /api/files/{id}`
- **Descripción**: Devuelve un archivo específico identificado por su ID.
- **Parámetros**:
  - `id`: ID del archivo solicitado.
- **Respuesta**: Objeto `Files` correspondiente al archivo solicitado.

9.3. Entidad Test

La entidad **Test** representa las preguntas de los tests que se asignan a un archivo específico dentro del sistema. Esta entidad está diseñada para almacenar las preguntas, sus posibles respuestas y las respuestas correctas, asociándolas a un contenido multimedia previamente definido.

9.3.1. Campos principales de la entidad

- **idPregunta**: Identificador único de la pregunta.
- **idFile**: Identificador del archivo al que pertenece la pregunta.
- **pregunta**: Texto que contiene la pregunta formulada.
- **respuestas**: Lista de posibles respuestas para la pregunta.
- **correctas**: Lista de índices que identifican las respuestas correctas.

Esta estructura permite una gestión clara y flexible de las preguntas y sus respuestas, asegurando que cada pregunta esté vinculada a un contenido multimedia específico.

9.3.2. Métodos Test

El sistema expone varios métodos relacionados con la entidad **Test** para realizar operaciones a través de la API REST:

Obtener preguntas asociadas a un archivo

- **Endpoint**: `GET /api/tests/{id_File}`
- **Descripción**: Devuelve todas las preguntas asociadas al archivo identificado por su ID.
- **Parámetros**:
  - `id_File`: ID del archivo cuyas preguntas se desean obtener.
- **Respuesta**: Lista de objetos `Test` que representan las preguntas del test asociado al archivo.

9.4. Entidad Result

La entidad **Result** representa los resultados obtenidos por los usuarios en las pruebas asociadas a los archivos multimedia del sistema. Este modelo permite registrar y consultar los resultados individuales, relacionándolos tanto con los usuarios como con los archivos evaluados.

9.4.1. Campos principales de la entidad

- **id**: Identificador único del resultado.
- **idUsuario**: Identificador del usuario que realizó la prueba.
- **idFile**: Identificador del archivo asociado al test del que se obtiene el resultado.
- **resultado**: Nota o puntuación obtenida por el usuario en la prueba (valor numérico).

Esta entidad permite almacenar y gestionar los resultados de manera eficiente, posibilitando la evaluación y análisis de las pruebas realizadas.

9.4.2. Métodos Result

El sistema expone varios métodos relacionados con la entidad **Result** para realizar operaciones a través de la API REST:

Obtener resultados por usuario

- **Endpoint**: `GET /api/results/{idUsuario}`
- **Descripción**: Devuelve todos los resultados asociados al usuario identificado por su ID.

- **Parámetros**:
  - `idUsuario`: ID del usuario cuyos resultados se desean consultar.
- **Respuesta**: Lista de objetos `Result` que representan los resultados del usuario.

Añadir un nuevo resultado

- **Endpoint**: `POST /api/results`
- **Descripción**: Permite registrar un nuevo resultado para un usuario en un archivo específico.
- **Parámetros**:
  - **Body**: Objeto JSON con los datos del resultado, que incluye:
    - `idUsuario`: ID del usuario.
    - `idFile`: ID del archivo asociado.
    - `resultado`: Nota obtenida.
- **Respuesta**: Objeto `Result` que representa el resultado guardado.

Obtener un resultado específico por usuario y archivo

- **Endpoint**: `GET /api/results/user/{idUsuario}/file/{idFile}`
- **Descripción**: Recupera el resultado de un usuario específico en un archivo específico.
- **Parámetros**:
  - `idUsuario`: ID del usuario.
  - `idFile`: ID del archivo.
- **Respuesta**: Nota obtenida por el usuario en ese archivo, en formato numérico (`Double`).

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Tue Jan 14 12:21:22 CET 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)