



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

INGENIERÍA DE COMPUTADORES

TRABAJO FIN DE GRADO

CYBERWEB

WEB PARA EL APRENDIZAJE DE LA CIBERSEGURIDAD

Realizado por

Álvaro Romero Cruz

Dirigido por

Rafael Martínez Gasca

Departamento

Lenguajes y Sistemas Informáticos

Sevilla, junio de 2024

Resumen

En el presente Trabajo se describe el desarrollo e implementación de una plataforma web para la realización de ejercicios de ciberseguridad, con el objetivo de mejorar las habilidades prácticas de los estudiantes en esta área. Este proyecto se enmarca dentro del Grado de Ingeniería Informática, en la Escuela Técnica Superior de Ingeniería Informática de Sevilla (España).

La plataforma ha sido diseñada para proporcionar un entorno seguro en el que los estudiantes puedan aprender y practicar diversos aspectos de la ciberseguridad a través de una serie de ejercicios y desafíos interactivos. Esta plataforma también ofrece la posibilidad de utilizar una máquina virtual para realizar los respectivos ejercicios con el fin de llegar a todas las personas, tengan o no ordenadores potentes capaces de realizar los ejercicios en poco tiempo y con fluidez.

Se ha utilizado una metodología propia para la creación y evaluación de los ejercicios para garantizar su calidad y relevancia educativa. Esta metodología también incluye retroalimentación continua de los usuarios, los cuales pueden subir sus propios ejercicios a la web, facilitando así la continua ampliación de los materiales educativos. La plataforma se basa en tecnologías web modernas, lo que garantiza su accesibilidad y facilidad de uso.

En conclusión, esta plataforma no sólo sirve como una herramienta práctica de aprendizaje, sino que también promueve la colaboración y la mejora continua de los materiales educativos, lo que repercute positivamente en la formación de los futuros profesionales de la ciberseguridad al tener una amplia gama de ejercicios con los que poner en marcha su aprendizaje en este sector.

Palabras clave: ciberseguridad, criptografía, hash, esteganografía, phishing, máquina virtual, entorno controlado, Kali-Linux, vulnerabilidad.

Abstract

This paper describes the development and implementation of a web platform for carrying out cybersecurity exercises, with the aim of improving students' practical skills in this area. This project is part of the Degree in Computer Engineering at the Escuela Técnica Superior de Ingeniería Informática of Seville (Spain).

The platform has been designed to provide a secure environment in which students can learn and practice various aspects of cybersecurity through a series of interactive exercises and challenges. This platform also offers the possibility to use a virtual machine to perform the respective exercises in order to reach everyone, whether or not they have powerful computers capable of performing the exercises in a short time and with fluency.

A proprietary methodology was used for the creation and evaluation of the exercises to ensure their quality and educational relevance. This methodology also includes continuous feedback from users, who can upload their own exercises, facilitating the continuous expansion of the educational materials. The platform is based on modern web technologies, which ensure its accessibility and ease of use.

In conclusion, this platform not only serves as a practical learning tool, but also promotes collaboration and continuous improvement of the educational materials, which has a positive impact on the training of future cybersecurity professionals by providing them with a wide range of exercises with which to implement their learning in this sector.

Keywords: cybersecurity, cryptography, hash, steganography, phishing, virtual machine, controlled environment, Kali-Linux, vulnerability.

Agradecimientos

Índice

Resumen	2
Abstract	3
Agradecimientos	4
Índice	5
Índice de Tablas	7
Índice de Figuras.....	8
1. Introducción	10
1.1. Objetivos	11
1.2. Alcance	12
1.3. Estructura del Documento	13
2. Estado del Arte	14
2.1. Hack The Box	15
2.2. TryHackMe.....	16
2.3. VulnHub	17
3. Pliego de Requisitos	18
4. Tecnologías empleadas.....	21
4.1. Node.js	22
4.2. React	24
4.3. Express	26
4.4. MySQL	28
4.5. Bootstrap. FontAwesome	30
5. Arquitectura del Sistema.....	32
5.1. Interacción entre React y Node.js.....	33
5.2. Interacción entre Node.js y MySQL	35
6. Desarrollo	36
6.1. Gestión de Usuarios	37
6.2. Gestión de Puntuaciones.....	39

6.3. Gestión de Archivos.....	41
6.4. Despliegue de Dockers	43
7. Pruebas del Sistema.....	45
7.1. Pruebas Unitarias y de Integración	47
7.2. Pruebas Funcionales y de Rendimiento.....	73
7.3. Pruebas de Usuario.....	74
8. Planificación Temporal.....	75
9. Costes.....	79
10. Conclusiones.....	81
10.1. Conclusiones Personales	83
11. Trabajo Futuro	84
12. Webgrafía.....	86
13. Anexos	87
A. Glosario	87
B. Manual de Usuario de la Aplicación.....	89
C. Manual de Instalación de la Aplicación	90
D. Manual de Montaje del Sistema	91

Índice de Tablas

Tabla 1 - Consultas a la Web	73
Tabla 2 - Consultas a la Base de Datos	73
Tabla 3 - Consultas al Sistema de Almacenamiento	73
Tabla 4 - Pruebas de Usuario	74
Tabla 5 - Coste de los Materiales	79
Tabla 6 - Coste del Personal	79
Tabla 7 - Coste Total del Proyecto	80

Índice de Figuras

Ilustración 1 - Node.js.....	22
Ilustración 2 - Arquitectura de Node.js	22
Ilustración 3 - React	24
Ilustración 4 - Express.js	26
Ilustración 5 - MySQL.....	28
Ilustración 6 - Bootstrap y FontAwesome	30
Ilustración 7 - Arquitectura del Sistema.....	32
Ilustración 8 - getFiles() (RepositoryPage).....	33
Ilustración 9 - getFiles() (getFiles.js)	33
Ilustración 10 - getFiles() (config.js)	34
Ilustración 11 - Diagrama de Flujo de la Gestión de Usuarios	38
Ilustración 12- Diagrama de Flujo de la Gestión de Puntuaciones.....	40
Ilustración 13- Diagrama de Flujo de la Gestión del Archivos I.....	42
Ilustración 14- Diagrama de Flujo de la Gestión de Archivos II.....	42
Ilustración 15- Diagrama de Flujo del Despliegue de Dockers	44
Ilustración 16 - addPoints.js	47
Ilustración 17 - addPoints.js (Thunder Client)	47
Ilustración 18 - approveAdminFiles.js	48
Ilustración 19 - approveAdminFiles.js (Thunder Client).....	48
Ilustración 20 - deleteAdminFiles.js	49
Ilustración 21 - deleteAdminFiles.js (Thunder Client)	49
Ilustración 22 - deleteFiles.js.....	50
Ilustración 23 - deleteFiles.js (Thunder Client)	50
Ilustración 24 - deleteUsers.js	51
Ilustración 25 - deleteUsers.js (Thunder Client)	51
Ilustración 26 - downloadAdminFiles.js.....	52
Ilustración 27 - downloadAdminFiles.js (Thunder Client)	52
Ilustración 28 - downloadFiles.js	53

Ilustración 29 - downloadFiles.js (Thunder Client).....	53
Ilustración 30 - getAdminFiles.js.....	54
Ilustración 31 - getAdminFiles.js (Thunder Client)	54
Ilustración 32 - getAvatars.js	55
Ilustración 33 - getAvatars.js (Thunder Client).....	55
Ilustración 34 - getFiles.js	56
Ilustración 35 - getFiles.js (Thunder Client).....	56
Ilustración 36 - getLogin.js	57
Ilustración 37 - getLogin.js (Thunder Client).....	58
Ilustración 38 - getOwnFiles.js	59
Ilustración 39 - getOwnFiles.js (Thunder Client).....	59
Ilustración 40 - getPoints.js	60
Ilustración 41 - getPoints.js (Thunder Client)	60
Ilustración 42 - getRanking.js	61
Ilustración 43 - getRanking.js (Thunder Client).....	61
Ilustración 44 - getRegister.js	62
Ilustración 45 - getRegister.js (Thunder Client).....	62
Ilustración 46 - getRenewToken.js.....	63
Ilustración 47 - getRenewToken.js (Thunder Client)	63
Ilustración 48 - getUpdateTasks.js	64
Ilustración 49 - getUpdateTask.js (Thunder Client)	64
Ilustración 50 - getUpdateTraceability.js.....	65
Ilustración 51 - getUpdateTraceability.js (Thunder Client)	65
Ilustración 52 - getUpdateUser.js.....	66
Ilustración 53 - getUpdateUser.js (Thunder Client)	66
Ilustración 54 - updatePassword.js	67
Ilustración 55 - updatePassword.js (Thunder Client)	67
Ilustración 56 - uploadAvatars.js	68
Ilustración 57 - uploadAvatars.js (Thunder Client).....	69
Ilustración 58 - uploadFiles.js.....	70
Ilustración 59 - uploadFiles.js (Thunder Client)	71
Ilustración 60 - Resultados Generales de Jest.....	72
Ilustración 61 - Diagrama de Gantt	78

1. Introducción

El propósito de este documento es describir los requerimientos y especificaciones, de los servicios de diseño y desarrollo, de la plataforma interactiva que la Universidad de Sevilla requiere para la creación y realización de ejercicios de ciberseguridad.

Nuestra empresa, especializada en el desarrollo web, tiene como misión principal brindar a nuestros clientes herramientas efectivas para fortalecer sus capacidades en el ámbito de la ciberseguridad, como uno de los desafíos más recurrentes. Esto, obedece a la necesidad de entrenar al personal en la identificación y respuesta a amenazas digitales. La plataforma de ejercicios de ciberseguridad se centra en abordar este problema proporcionando un entorno interactivo donde los usuarios pueden practicar y perfeccionar sus habilidades de detección y respuesta a incidentes cibernéticos, así como en el ataque, la defensa y el análisis forense.

Además de fortalecer las habilidades de los usuarios, se abunda en el compromiso de mejorar la calidad de las prácticas de ciberseguridad. Esto implica hacer que sea más sencillo para las organizaciones proteger sus activos digitales y salvaguardar la privacidad de sus usuarios. Reconociendo la importancia de la vigilancia y detección temprana de amenazas en el entorno digital. Paralelo anterior, se integran tecnologías avanzadas de análisis de datos y detección de patrones en nuestras soluciones para identificar posibles vulnerabilidades y amenazas de seguridad de manera proactiva.

En consecuencia, la plataforma está equipada con una amplia variedad de herramientas educativas y recursos técnicos, incluyendo laboratorios virtuales y módulos de aprendizaje que se ajustan al nivel de cada usuario. También cuenta con un sistema de trazabilidad para que el cliente pueda seguir con detalle la habilidad y capacidad de resolución de sus usuarios, teniendo en cuenta a los más brillantes para futuras posibles ofertas de trabajo por parte de empresas del sector de la ciberseguridad.

1.1. Objetivos

Este Trabajo se enfoca en alcanzar múltiples objetivos que se integran de manera sinérgica para ofrecer una solución integral en el ámbito de la ciberseguridad.

Este proyecto tiene como objetivo principal el desarrollo de plataformas interactivas donde el usuario puede poner a prueba sus conocimientos sobre ciberseguridad, ofreciendo tanto la plataforma final como ejemplos y ejercicios relacionados con la seguridad digital.

Como objetivo secundario que complementa al principal, se desarrollará la página web donde estos ejercicios aparecen, así el cliente no tiene que preocuparse de contratar un servicio a parte del que nosotros proporcionamos.

Además, proporciona también un sistema para que el usuario sea capaz de subir ejercicios creados por ellos mismos. Una vez que suban el ejercicio a la plataforma, se comprobará que es apto y no maligno, para que otros usuarios lo tengan visible.

En resumen, este proyecto representa una inversión clave para futuros desarrollos que elevarán el estándar de ciberseguridad en el entorno universitario. Desde la protección de datos hasta la formación de expertos, esta plataforma servirá como pilar fundamental en la misión de ofrecer un ambiente digital seguro y de alta calidad para toda su comunidad educativa.

1.2. Alcance

Se analizará el alcance en diferentes ámbitos:

Organizativo

Este proyecto, está enfocado en el desarrollo de una plataforma de ciberseguridad diseñada para el entorno universitario. Inicialmente, se dirige a satisfacer las necesidades del departamento de Lenguaje y Sistemas Informáticos (LSI) de la Universidad de Sevilla.

Esta plataforma sirve como base sólida para una amplia gama de proyectos destinados a mejorar el conocimiento sobre seguridad y ciberprotección en el ámbito académico. Futuros desarrollos permitirán extender el alcance de esta solución a prácticamente todos los aspectos de la vida universitaria, desde protección de la investigación y la integridad de los sistemas hasta la prevención de amenazas cibernéticas.

Nos comprometemos a proporcionar un sólido sistema de aprendizaje, asegurando que el usuario pueda trabajar en un entorno digital seguro. Permitimos a la Universidad de Sevilla preparar a la próxima generación de expertos en seguridad digital y defensa en el entorno web.

Funcional

Como ya ha sido expuesto en los objetivos, nuestra intención es elaborar un producto de cara al usuario final. Por tanto, la propuesta se basa en la creación de una web de resolución de ciberejercicios, así como de un repositorio dentro de la misma para albergar los ciberejercicios de la comunidad.

Desarrollando los puntos anteriores tenemos:

- Infraestructura de servidores, donde correrán los dockers y bases de datos.
- El software necesario para albergar la web, así como una base de datos donde se almacenarán los usuarios y sus puntuaciones. También necesitaremos de otra independiente para almacenar los ciberejercicios de la comunidad.

Temporal

Este proyecto ha de estar acabado para mediados del próximo año fiscal, es decir, Mayo/Junio de 2024, con el fin de cumplir con la fecha límite impuesta.

1.3. Estructura del Documento

- **Introducción:** En esta sección se presenta una introducción al proyecto, donde también se recoge su objetivo y su alcance.

2. Estado del Arte

A la hora de desarrollar una plataforma interactiva para la realización de ejercicios de ciberseguridad, es necesario revisar y analizar las soluciones disponibles en el mercado. A continuación, se presentan tres de las plataformas más famosas en este campo y se evalúan sus características, ventajas y desventajas, así como el aporte que brindará este proyecto frente a estas opciones.

2.1. Hack The Box

Hack The Box es una plataforma en línea que proporciona un laboratorio virtual donde los usuarios pueden practicar hacking ético en un entorno seguro. Se centra en la gamificación del aprendizaje, ofreciendo desafíos de distintos niveles de dificultad y cubriendo una amplia gama de tecnologías y herramientas de ciberseguridad.

Ventajas

- **Variedad de Retos**: Posee una amplia variedad de desafíos que abordan diferentes aspectos de la ciberseguridad, desde la explotación de vulnerabilidades hasta la ingeniería inversa.
- **Gamificación**: Usa elementos de juego como puntuaciones, insignias y tablas de clasificación para motivar a los usuarios.
- **Comunidad Activa**: Formado por una gran comunidad de usuarios que comparte soluciones, sugerencias y soporte.

Inconvenientes

- **Curva de Aprendizaje**: Puede resultar abrumador para principiantes debido a la falta de orientación y la dificultad de algunos desafíos.
- **Coste**: Aunque existe una versión gratuita, muchas funciones avanzadas y laboratorios premium requieren una suscripción de pago.
- **Enfoque Competitivo**: El gran enfoque competitivo puede no ser adecuado para todos los estilos de aprendizaje.

Aportación de Este Proyecto

Este proyecto destaca al proporcionar una plataforma más accesible para principiantes con instrucciones detalladas y tutoriales que acompañan a cada ejercicio. Además, dado que está diseñado para una institución educativa, no existirá costo de acceso para los estudiantes, brindando una opción más económica sin sacrificar la calidad de la formación.

2.2. TryHackMe

TryHackMe es una plataforma similar a Hack The Box, pero con un mayor enfoque en el aprendizaje guiado. Ofrece laboratorios virtuales y desafíos interactivos diseñados para brindar a los usuarios una comprensión paso a paso de diversos conceptos y habilidades de ciberseguridad.

Ventajas

- Aprendizaje Guiado: Módulos con instrucciones paso a paso, ideales para principiantes.
- Variedad de Contenido: Amplia gama de temas, desde conceptos básicos de la ciberseguridad hasta técnicas avanzadas.
- Accesibilidad: Interfaz intuitiva y recursos disponibles para todos los niveles de habilidad.

Inconvenientes

- Limitaciones en la Versión Gratuita: Determinadas funciones y laboratorios avanzados solo están disponibles para usuarios pagos.
- Menor Enfoque en la Competencia: Un menor énfasis en la competencia puede reducir la motivación de algunos usuarios.
- Actualización de Contenidos: Aunque se actualizan periódicamente, la velocidad a la que surgen nuevas amenazas puede hacer que algunos módulos queden obsoletos.

Aportación de Este Proyecto

Nuestra plataforma no solo proporciona aprendizaje guiado, sino que también integra tecnología avanzada de análisis de datos para personalizar desafíos y contenidos en función del progreso del usuario. Además, nos enfocamos en crear contenido actualizado que refleje las últimas tendencias y amenazas en ciberseguridad.

2.3. VulnHub

VulnHub es una plataforma que proporciona máquinas virtuales para practicar habilidades de hacking y penetración. Estas máquinas reproducen escenarios de vulnerabilidad reales que los usuarios pueden descargar y ejecutar en sus propios entornos para simular ataques y defensas.

Ventajas

- Realismo: Las máquinas virtuales están diseñadas para replicar vulnerabilidades y configuraciones reales, proporcionando una experiencia de aprendizaje práctica y relevante.
- Variedad de Máquinas: Existen muchos tipos de máquinas que cubren diferentes niveles de dificultad y tipos de vulnerabilidad.
- Acceso Gratuito: Todos los recursos en VulnHub son gratuitos, lo que los hace fácilmente accesibles para una amplia audiencia.

Inconvenientes

- Requiere Configuración Local: Los usuarios deben descargar y configurar máquinas virtuales en su propio entorno, lo que puede ser una barrera para los usuarios con menos conocimientos técnicos.
- Menor Interactividad: En comparación con otras plataformas como Hack The Box y TryHackMe, faltan funciones interactivas y gamificación.
- Actualización Irregular: La frecuencia de nuevas máquinas y actualizaciones puede no ser constante, lo que puede limitar la relevancia del contenido.

Aportación de Este Proyecto

Nuestra plataforma ofrece la comodidad de un entorno en línea sin configuraciones locales complejas y combina una experiencia interactiva y gamificada. Además, se proporciona un flujo continuo de actualizaciones y contenido personalizado para que los usuarios estén siempre al día sobre los últimos desafíos y tendencias en ciberseguridad.

3. Pliego de Requisitos

A continuación, se desarrollan los requisitos que se solicitan a los posibles ofertantes, en todos los ámbitos.

Entregables propuestos

Se proponen los entregables más cercanos al diseño del proyecto sin especificar su implementación:

- Diseño de aplicación genérica donde recibir y gestionar ejercicios.
- Diseño de diagrama de arquitectura.
- Diseño de base de datos.

Se proponen también los entregables referidos al desarrollo de todo lo diseñado anteriormente:

- Diseño de página web.
- Desarrollo de ejercicios de ciberseguridad.
- Creación de base de datos.
- Implantación de ejercicios en la página web.
- Recopilación de resultados de los usuarios de prueba.

Requisitos técnicos

En cuanto a los requisitos técnicos tenemos los siguientes:

- Adaptación a todos los sistemas operativos.
- Compatibilidad con todos los buscadores web.
- Regularización de privacidad y seguridad de datos.

Requisitos funcionales

En el punto de vista funcional nos encontraremos los especificados a continuación:

Cuenta y perfil

- El usuario podrá crear una cuenta con información personal.
- El usuario puede editar su perfil, añadiendo información personal.

Ejercicios

- Entorno seguro para realizar las pruebas sin afectar al sistema real.
- Los ejercicios estarán categorizados para hacer más fácil la búsqueda de estos.
- El usuario puede ver una lista con los ejercicios disponibles, con descripción y nivel de dificultad.
- Evaluación del resultado incluyendo puntaje y retroalimentación.
- El usuario tendrá un historial de los ejercicios completados.

Comunidad

- Los usuarios podrán subir sus ejercicios a la plataforma para que otros usuarios los realicen.

Administración

- Los administradores pueden agregar, editar y eliminar ejercicios y categorías.

Seguridad

- La página estará protegida contra ataques comunes, como ataques por inyección de SQL.

Requisitos temporales

Desde la perspectiva temporal, la organización y planificación de los diferentes hitos/entregables, así como sus correspondientes pruebas será propuesto por el oferente. Sin embargo, es crucial tener en cuenta que existen restricciones temporales que deben cumplirse en ciertos puntos del proyecto y que son de vital importancia para su desarrollo adecuado. Estas restricciones están relacionadas con el sistema financiero que se emplea, el cual se basa en la inversión. Por lo tanto, se establecen los siguientes plazos esenciales:

- Demostración de una fase temprana de la plataforma, donde podamos registrarnos y realizar al menos un ejercicio obteniendo puntuación en este para finales de este año fiscal (31 de diciembre de 2023).
- Proyecto finalizado completamente para mediados del próximo año fiscal, Mayo/Junio de 2024.

Requisitos de calidad

Para el correcto funcionamiento del sistema es necesario que todo el software relacionado con el protocolo de comunicación usuario-servidor tiene que cumplir el estándar ISO 17799.

4. Tecnologías empleadas

A continuación, se realiza un estudio de las diferentes tecnologías que se han contemplado para desarrollar la aplicación web.

En la primera sección se hablará acerca de Node.js, el lenguaje JavaScript que se ha empleado para la implementación del servidor.

En la segunda sección se explicará en qué consiste React, un framework de JavaScript desarrollado por Facebook, que se ha utilizado para la construcción de la interfaz de usuario.

En la tercera sección se explicará Express, un framework web minimalista para Node.js, que se ha utilizado para la creación del servidor y el manejo de rutas.

En la cuarta sección se hablará de la base de datos utilizada, la cual ha sido MySQL, una base de datos relacional ampliamente utilizada por su fiabilidad y eficiencia.

Para finalizar, se hablará de Bootstrap, una famosa librería CSS que incluye un diseño adaptable, y FontAwesome, una pequeña librería CSS conocida por su conjunto de iconos.

Cada una de estas tecnologías ha sido seleccionada y utilizada para asegurar que la aplicación sea robusta, escalable y fácil de mantener.

4.1. Node.js



Ilustración 1 - Node.js

Node.js es un entorno de ejecución para JavaScript creado con el motor V8 de Google Chrome. Permite a los desarrolladores utilizar JavaScript para el desarrollo del lado del servidor, lo que permite crear aplicaciones web altamente escalables y eficientes. Node.js presenta una arquitectura orientada a eventos y un modelo de I/O no bloqueante, lo que lo hace ideal para aplicaciones en tiempo real y aplicaciones que procesan grandes cantidades de datos.

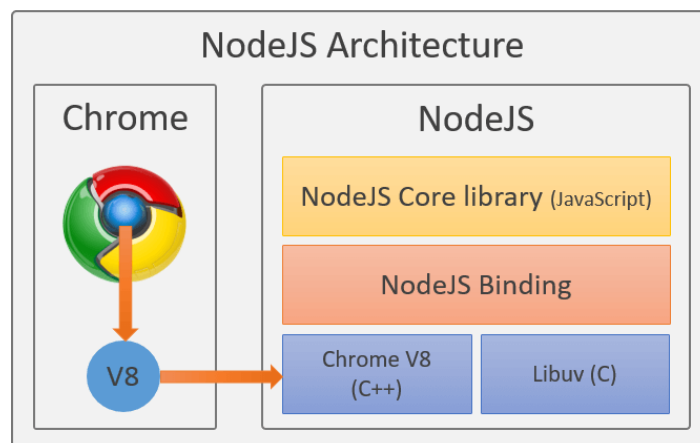


Ilustración 2 - Arquitectura de Node.js

Características Principales de Node.js

- **Orientación a Eventos y I/O No Bloqueante:** Node.js utiliza un modelo de I/O no bloqueante, basado en eventos. Esto significa que las operaciones de entrada/salida (como la lectura de archivos o la consulta a bases de datos) no bloquean la ejecución del hilo principal. En cambio, Node.js procesa estas operaciones de forma asíncrona, lo que permite que el servidor procese otras solicitudes mientras espera que se completen las operaciones de I/O.

- Motor V8 de Google: El motor V8 de Google, escrito en C++, compila código JavaScript directamente en código máquina para una ejecución extremadamente rápida. Esto es fundamental para el rendimiento de Node.js, especialmente para aplicaciones que requieren un procesamiento intensivo.
- Ecosistema de NPM: Node Package Manager (NPM) es el administrador de paquetes de Node.js, y es uno de los mayores repositorios de bibliotecas y herramientas de código abierto. NPM facilita la gestión de dependencias y la integración de módulos de terceros para acelerar el desarrollo y la implementación de aplicaciones.

Ventajas de Usar Node.js

- Escalabilidad: Debido a su modelo asíncrono y orientado a eventos, Node.js puede manejar una gran cantidad de conexiones simultáneas con un rendimiento óptimo. Esta característica es fundamental para aplicaciones que escalan horizontalmente y gestionan múltiples usuarios simultáneamente.
- Desarrollo Full Stack con JavaScript: Node.js permite a los desarrolladores usar JavaScript tanto en el lado del cliente como en el del servidor. Esto unifica el proceso de desarrollo, facilitando la reutilización de código y la colaboración entre los equipos de frontend y backend.
- Comunidad y Soporte: La comunidad de Node.js es extensa y activa, proporcionando una amplia gama de módulos y bibliotecas que se pueden integrar en proyectos. Además, existe una amplia documentación y el soporte comunitario es abundante, lo que facilita la resolución de problemas y la implementación de nuevas funcionalidades.

4.2. React



Ilustración 3 - React

React es una biblioteca de JavaScript desarrollada por Facebook para la construcción de interfaces de usuario. Su principal objetivo es facilitar el desarrollo de aplicaciones web dinámicas y eficientes, mediante la creación de componentes reutilizables. React sigue una arquitectura basada en componentes y aprovecha un DOM virtual para simplificar las actualizaciones de la interfaz de usuario, lo que resulta en una experiencia de usuario más fluida y rápida.

Características Principales de React

- Componentes Reutilizables: React introduce el concepto de componentes, que son fragmentos de código reutilizables que representan partes de una interfaz de usuario. Cada componente puede tener su propio estado y lógica, lo que permite construir interfaces de usuario complejas de manera modular y mantenible.
- Virtual DOM: React utiliza un DOM virtual para minimizar las actualizaciones directas al DOM real. El DOM virtual es una representación simplificada del DOM real que React puede actualizar de manera eficiente realizando solo los cambios necesarios. Esto mejora significativamente el rendimiento de las aplicaciones web.
- JSX: Extensión de JavaScript que permite escribir código similar a HTML en archivos JavaScript. JSX hace que el código sea más legible y expresivo, lo que facilita la creación de componentes de interfaz de usuario complejos.

Ventajas de Usar React

- **Eficiencia y Rendimiento**: Al utilizar Virtual DOM, React optimiza las actualizaciones de la interfaz de usuario, teniendo un rendimiento superior en comparación con las manipulaciones directas del DOM.
- **Desarrollo Basado en Componentes**: La arquitectura basada en componentes de React promueve la reutilización y la separación de preocupaciones, lo que facilita el desarrollo y mantenimiento de aplicaciones grandes y complejas.
- **Ecosistema y Herramientas**: React tiene un ecosistema rico y diverso que incluye herramientas como Create React App, React Developer Tools y una amplia variedad de bibliotecas y componentes de terceros que se pueden integrar fácilmente en los proyectos.

4.3. Express



Ilustración 4 - Express.js

Express es un framework web para Node.js que simplifica el desarrollo de aplicaciones web y APIs. Su diseño es minimalista y flexible, lo que permite a los desarrolladores crear servidores web de forma rápida y eficiente. Express proporciona una capa de abstracción sobre HTTP que facilita la gestión del enrutamiento, el middleware y otros aspectos comunes de las aplicaciones web.

Características Principales de Express

- **Routing**: Express facilita la definición de rutas y el manejo de solicitudes HTTP. Permite asociar funciones de controlador a rutas específicas, lo que facilita la organización del código y la gestión de la lógica de negocio.
- **Middleware**: El middleware en Express son funciones que se ejecutan antes de que se procese una solicitud. Esto permite realizar tareas como la autenticación, el registro de solicitudes, el análisis de datos, etc. Express proporciona un sistema de middleware flexible que permite encadenar múltiples funciones para manejar solicitudes de manera modular.
- **Motor de Plantillas**: Aunque no es necesario, Express puede integrarse fácilmente con diferentes motores de plantillas como Pug, EJS o Handlebars. Esto facilita la generación dinámica de contenido HTML en el servidor.

Ventajas de Usar Express

- **Simplicidad y Flexibilidad**: Express está diseñado para ser simple y flexible. No impone una estructura de archivos estricta ni convenciones de código, lo que permite a los desarrolladores crear aplicaciones web de acuerdo con sus necesidades y preferencias.
- **Gran Comunidad y Ecosistema**: Express tiene una gran comunidad de desarrolladores y una amplia variedad de middleware y complementos disponibles a través de npm. Esto facilita la integración de funcionalidades adicionales en las aplicaciones Express y acelera el proceso de desarrollo.
- **Escalabilidad**: Express es altamente escalable y puede manejar una gran cantidad de solicitudes simultáneas. Además, debido a que está construido sobre Node.js, se beneficia de la naturaleza asíncrona y no bloqueante de Node.js, lo que lo hace ideal para aplicaciones web de alto rendimiento en tiempo real.

4.4. MySQL



Ilustración 5 - MySQL

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto que se utiliza comúnmente en aplicaciones web y empresariales. Es conocido por su fiabilidad, rendimiento y escalabilidad, y es compatible con múltiples plataformas, incluyendo Linux, Windows y MacOS.

Características Principales de MySQL

- **Modelo Relacional**: MySQL utiliza un modelo relacional para almacenar datos en tablas organizadas en filas y columnas. Esto permite establecer relaciones entre diferentes conjuntos de datos y garantizar la integridad referencial.
- **Lenguaje SQL**: MySQL utiliza SQL (Structured Query Language) como lenguaje de consulta para interactuar con la base de datos. SQL proporciona una amplia gama de operaciones para crear, leer, actualizar y eliminar datos de manera eficiente.
- **Transacciones ACID**: MySQL garantiza la integridad y consistencia de los datos mediante el cumplimiento del estándar ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Esto garantiza que las transacciones sean atómicas, consistentes, aisladas y duraderas, incluso en entornos de alto rendimiento.

Ventajas de Usar MySQL

- **Fiabilidad y Escalabilidad:** MySQL es conocido por su fiabilidad y escalabilidad. Puede manejar grandes cantidades de datos y operaciones simultáneas sin afectar al rendimiento o la estabilidad del sistema.
- **Amplia Compatibilidad:** MySQL es compatible con una amplia variedad de plataformas, lo que lo hace adecuado para una variedad de entornos de desarrollo y despliegue. Además, es compatible con muchos lenguajes de programación y frameworks de desarrollo.
- **Comunidad y Soporte:** MySQL tiene una gran comunidad de usuarios y desarrolladores que brindan soporte técnico, documentación y recursos educativos. Además, hay disponible una amplia variedad de herramientas y bibliotecas disponibles para facilitar el desarrollo y administración de bases de datos MySQL.

4.5. Bootstrap. FontAwesome



Ilustración 6 - Bootstrap y FontAwesome

Bootstrap es un framework para el desarrollo de interfaces de usuario web de código abierto, mientras que FontAwesome es una biblioteca de iconos vectoriales. Ambas herramientas son comúnmente utilizadas en el desarrollo web para mejorar la apariencia y funcionalidad de las interfaces de usuario.

Características Principales de Bootstrap y FontAwesome

Bootstrap

- Proporciona un conjunto de componentes y estilos predefinidos para crear fácilmente interfaces web responsivas.
- Su diseño basado en rejillas permite una adaptación automática a diferentes tamaños de pantalla.
- Proporciona una documentación detallada y ejemplos de código para facilitar su uso.

FontAwesome

- Proporciona una amplia variedad de iconos vectoriales sobre diferentes estilos y categorías.
- Se integra fácilmente en proyectos web mediante el uso de clases CSS predefinidas.
- Permite personalizar los iconos mediante CSS para adaptarlos al diseño de la aplicación.

Ventajas de Usar Bootstrap y FontAwesome

- Ahorro de Tiempo: Bootstrap y FontAwesome permiten crear interfaces atractivas y funcionales rápidamente gracias a sus componentes e iconos.
- Consistencia Visual: Ambas herramientas garantizan una apariencia consistente en toda la aplicación, mejorando la experiencia del usuario.
- Flexibilidad y Escalabilidad: Bootstrap y FontAwesome ofrecen una amplia variedad de opciones y son compatibles con una gran variedad de dispositivos y resoluciones de pantalla.

5. Arquitectura del Sistema

En este apartado se comentará el uso que se le ha dado a cada una de las tecnologías citadas en el apartado anterior.

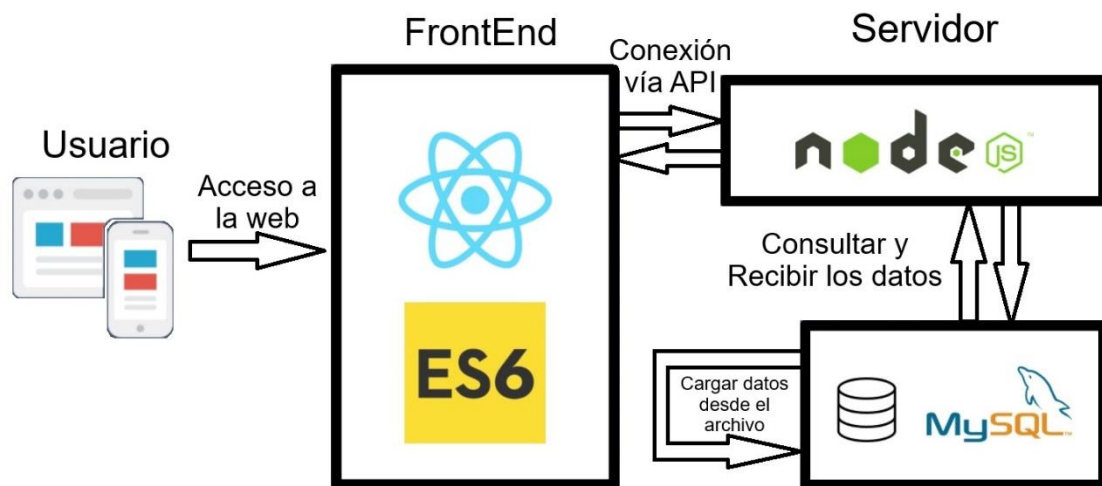


Ilustración 7 - Arquitectura del Sistema

En la imagen anterior se puede apreciar la arquitectura de la aplicación web. Cuando un usuario accede al sitio web, la interfaz de usuario, construida con React y Express, le muestra el contenido correspondiente. Dicho contenido se genera después de que el FrontEnd realice una solicitud al servidor a través de una API. El servidor, a su vez, consulta los datos necesarios en la base de datos y devuelve una respuesta al FrontEnd para que el usuario pueda visualizarla.

5.1. Interacción entre React y Node.js

La interacción entre React y Node.js es crucial para el buen funcionamiento de la aplicación web. React, como biblioteca de JavaScript del lado del cliente, se encarga de construir la interfaz de usuario dinámica y reactiva que los usuarios ven y con la que interactúan. Por otro lado, Node.js actúa como el entorno de ejecución del lado del servidor que ejecuta el BackEnd de la aplicación. React y Node.js se comunican entre sí mediante solicitudes HTTP, donde React envía solicitudes al servidor Node.js para acceder a datos o realizar acciones específicas. Node.js, a su vez, responde a estas solicitudes procesando la lógica de negocio, accediendo a la base de datos si es necesario y devolviendo los datos relevantes a React para su visualización en la interfaz de usuario. Esta interacción bidireccional entre React y Node.js permite una experiencia de usuario dinámica y receptiva, mediante la cual, los cambios realizados en el FrontEnd se reflejan inmediatamente en el BackEnd y viceversa, proporcionando una experiencia de usuario fluida y coherente.

A continuación, se muestra como ejemplo de esta interacción un fragmento de la vista “RepositoryPage.jsx” donde se realiza la llamada a la función “getFiles()” para obtener todos los ejercicios subidos por los usuarios al servidor.

```

17      useEffect(() => {
18          const fetchFiles = async () => {
19              try {
20                  const data = await getFiles();
21                  setFiles(data.files);
22              } catch (error) {
23                  console.error(error);
24              }
25          }
26
27          fetchFiles();
28      }, []);

```

Ilustración 8 - getFiles() (RepositoryPage)

Esta es una función asíncrona de la vista “getFiles.js” que utiliza la API Fetch para hacer una solicitud HTTP GET al servidor, como se muestra a continuación:

```

1  export const getFiles = async () => {
2      const response = await fetch('http://localhost:3001/getFiles', {
3          method: 'GET',
4          headers: {
5              'Content-Type': 'application/json',
6          },
7      });
8
9      const data = await response.json();
10     return data;
11 }

```

Ilustración 9 - getFiles() (getFiles.js)

Tras esto, la solicitud es recibida por el servidor, el cual se encarga de realizar la consulta y devolver la respuesta al FrontEnd, empleando para ello el siguiente código localizado en la vista “config.js”:

```

50
51 const getFiles = async (req, res) => {
52   try {
53     const baseDir = path.resolve(__dirname, '..', '..', 'repository');
54     let userDirs = await fs.promises.readdir(baseDir);
55
56     // Filtrar directorios que no son numéricos
57     userDirs = userDirs.filter(dir => !isNaN(dir));
58
59     const allFiles = [];
60
61     for (const uid of userDirs) {
62       const userDir = path.join(baseDir, uid);
63       const categories = await fs.promises.readdir(userDir);
64
65       for (const category of categories) {
66         const categoryDir = path.join(userDir, category);
67         const difficulties = await fs.promises.readdir(categoryDir);
68
69         for (const difficulty of difficulties) {
70           const difficultyDir = path.join(categoryDir, difficulty);
71           const files = await fs.promises.readdir(difficultyDir);
72
73           const userName = await getUserName(Number(uid));
74
75           const data = files.map(file => {
76             const stats = fs.statSync(path.join(difficultyDir, file));
77             return {
78               uid: uid,
79               name: file,
80               user: userName,
81               size: stats.size,
82               date: stats.birthtime,
83               category: category,
84               difficulty: difficulty
85             };
86           });
87
88           allFiles.push(...data);
89         }
90       }
91     }
92
93     res.json({ files: allFiles });
94
95   } catch (error) {
96     console.log(error);
97     res.status(500).json({
98       ok: false,
99       msg: 'Por favor hable con el administrador'
100     });
101   }
102 }
103

```

Ilustración 10 - getFiles() (config.js)

Si no ocurren errores en la petición, el servidor devolverá todos los ficheros que han sido almacenados en el repositorio.

5.2. Interacción entre Node.js y MySQL

La interacción entre Node.js y MySQL se lleva a cabo de manera eficiente y sincrónica durante la ejecución del servidor. Cuando Node.js recibe una solicitud de un cliente para acceder a la base de datos, inicia el proceso estableciendo una conexión con el servidor MySQL. Una vez establecida la conexión, Node.js ejecuta consultas SQL para realizar operaciones de lectura, escritura o modificación en la base de datos. Estas consultas se ejecutan de forma asíncrona para evitar bloquear el flujo principal de ejecución del servidor, permitiendo al servidor manejar múltiples solicitudes simultáneamente sin comprometer su rendimiento. Node.js espera las respuestas de MySQL y, una vez recibidas, procesa los datos devueltos según sea necesario para generar una respuesta adecuada a la solicitud del cliente. Esta ejecución eficiente y asincrónica garantiza un rendimiento óptimo del servidor, contribuyendo a una experiencia de usuario fluida y receptiva en la aplicación web.

6. Desarrollo

En este apartado se analiza en profundidad el proceso de desarrollo de la aplicación web, explorando las diversas partes del sistema y las tecnologías empleadas en su creación. Cada componente ha sido desarrollado de manera independiente, siguiendo un enfoque meticuloso para garantizar un funcionamiento óptimo y su integración armoniosa en el sistema general:

- Gestión de Usuarios: Aquí se analizan las funcionalidades de autenticación y autorización, los métodos de almacenamiento seguro de contraseñas y las estrategias para gestionar las sesiones de usuario.
- Gestión de Puntuaciones: En esta sección se describe cómo se registran y gestionan las puntuaciones de los usuarios cuando completan ejercicios, incluyendo las consultas a la base de datos para calcular y almacenar las puntuaciones.
- Gestión de Archivos: Esta sección se centra en los procesos de carga, almacenamiento y descarga de archivos, así como las validaciones necesarias para asegurar la integridad de los datos.
- Despliegue de Dockers: Aquí se describen las configuraciones necesarias para crear y administrar contenedores Docker, y se discuten las ventajas de usar Podman en lugar de otras herramientas de contenedorización.

A lo largo de cada parte, se analizan los problemas encontrados durante el desarrollo, las alternativas consideradas y la integración con otros componentes del sistema. Se proporcionan diagramas de flujo y otros recursos visuales para ilustrar el funcionamiento y la interacción de cada parte del sistema. Esta sección proporciona una visión completa y detallada del proceso de desarrollo, destacando el esfuerzo y la dedicación necesarios para la creación de una aplicación web robusta, escalable y de alta calidad.

6.1. Gestión de Usuarios

Funcionamiento

Este componente se encarga de gestionar la autenticación y autorización de los usuarios en el sistema. Utiliza Express para manejar las solicitudes de registro, inicio de sesión y cierre de sesión.

Modo de Operación

Cuando un usuario intenta registrarse o iniciar sesión, este componente valida las credenciales proporcionadas y consulta la base de datos MySQL para verificar la información del usuario.

Problemas Encontrados y Solucionados

Uno de los desafíos encontrados fue gestionar el almacenamiento de las contraseñas de los usuarios de manera eficiente y segura. Para ello, se implementó un algoritmo de hash para cifrar las contraseñas antes de almacenarlas en la base de datos.

Alternativas Desechadas

Inicialmente, se consideró el uso de un servicio de autenticación externo como Auth0, pero se descartó debido a limitaciones de costos y requisitos de personalización.

Integración con el Resto de Componentes

Este componente se integra con el FrontEnd mediante solicitudes HTTP enviadas desde React al servidor Node.js. La información del usuario autenticado se almacena en la sesión del usuario para su acceso posterior.

Diagrama de Flujo

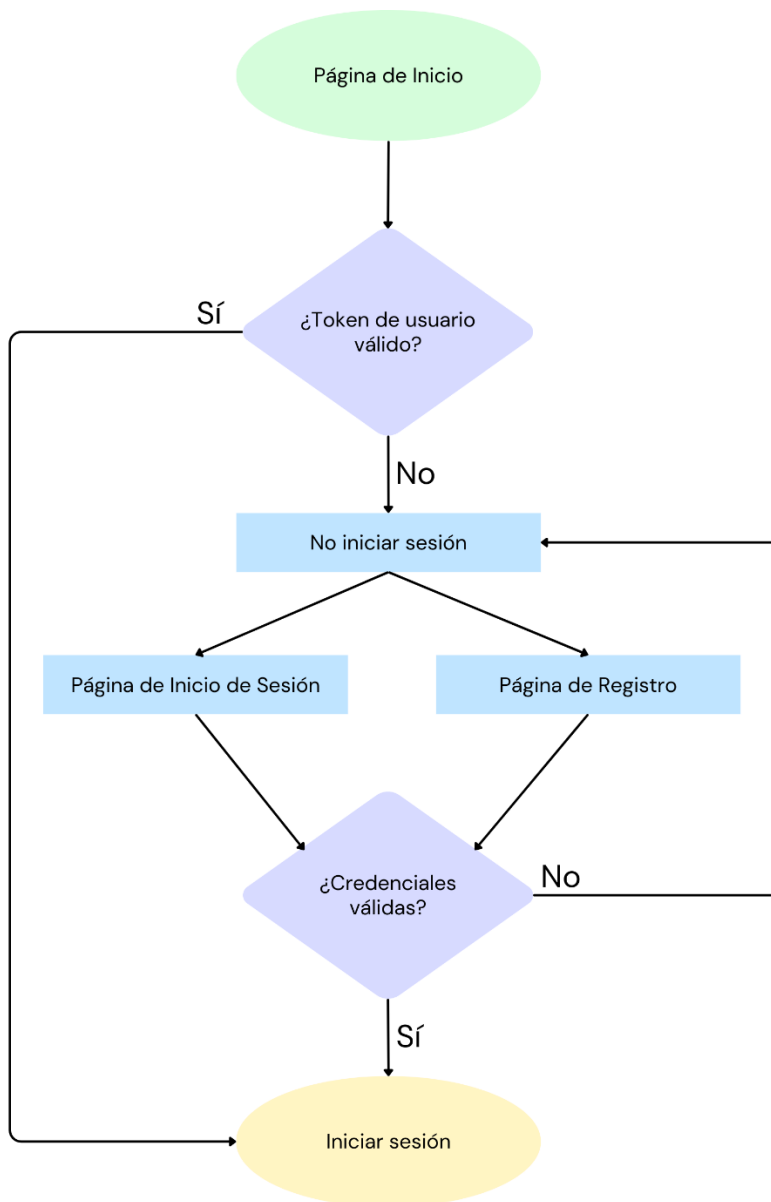


Ilustración 11 - Diagrama de Flujo de la Gestión de Usuarios

6.2. Gestión de Puntuaciones

Funcionamiento

Este componente se encarga de registrar y gestionar las puntuaciones de los usuarios cuando completan ejercicios o tareas dentro de la aplicación. Utiliza una base de datos MySQL para almacenar las puntuaciones.

Modo de Operación

Cuando un usuario completa un ejercicio, este componente registra la puntuación correspondiente en la base de datos y actualiza el perfil del usuario en consecuencia. También puede generar informes de rendimiento y estadísticas basadas en los resultados recopilados.

Problemas Encontrados y Solucionados

Un desafío fue garantizar la correcta actualización de las puntuaciones registradas y su trazabilidad, especialmente en entornos de alta concurrencia. Para ello, se implementaron diccionarios que relacionaban el ejercicio resuelto con su respectivo en la base de datos.

Alternativas Desechadas

Inicialmente, se consideró el uso de una solución de terceros para la gestión de puntuaciones, pero se optó por desarrollar una solución personalizada para satisfacer mejor los requisitos específicos del proyecto.

Integración con el Resto de Componentes

Este componente se integra con otros módulos del sistema, como la gestión de usuarios y la autenticación, para garantizar que las puntuaciones se registren y asocien correctamente a las cuentas de usuario correspondientes.

Diagrama de Flujo

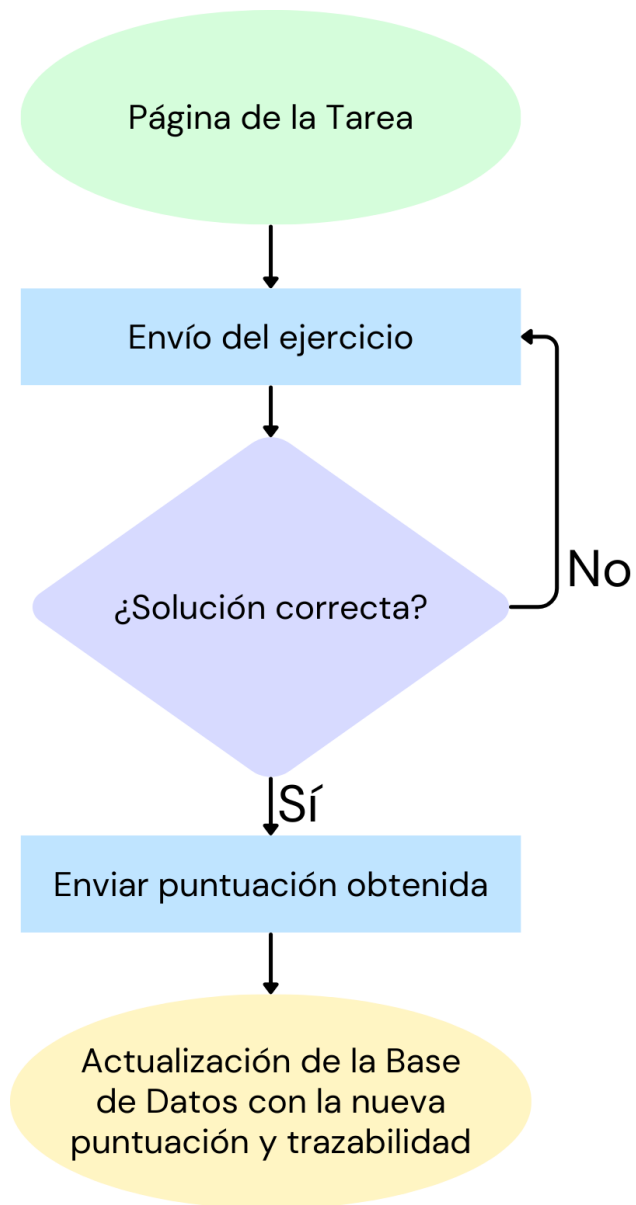


Ilustración 12- Diagrama de Flujo de la Gestión de Puntuaciones

6.3. Gestión de Archivos

Funcionamiento

Este componente se responsable de gestionar la carga, almacenamiento y manipulación de archivos en el sistema. Utiliza Express para manejar las solicitudes de carga y descarga de archivos, y se integra con un sistema de almacenamiento local.

Modo de Operación

Cuando un usuario carga un fichero, este componente lo procesa, realiza las validaciones necesarias, y lo almacena en el sistema de archivos local. Además, los usuarios pueden descargar archivos previamente cargados.

Problemas Encontrados y Solucionados

Uno de los desafíos fue mantener la seguridad de los usuarios frente a posibles subidas de contenido malicioso. Para ello, se creó un panel de administrador donde se suben los ficheros de los usuarios inicialmente. En este panel, los administradores podrán descargar el contenido en un entorno seguro para su revisión, y tras ello, aprobar el fichero para ser compartido con el resto de los usuarios.

Alternativas Desechadas

Inicialmente, se consideró la utilización de un servicio de almacenamiento en la nube, como AWS S3, pero se optó por un sistema de almacenamiento local debido a restricciones de costos, así como para tener un mayor control sobre los datos.

Integración con el Resto de Componentes

Este componente se integra con otros módulos del sistema, como la gestión de usuarios, para garantizar que solo los usuarios autorizados puedan acceder y manipular archivos.

Diagrama de Flujo

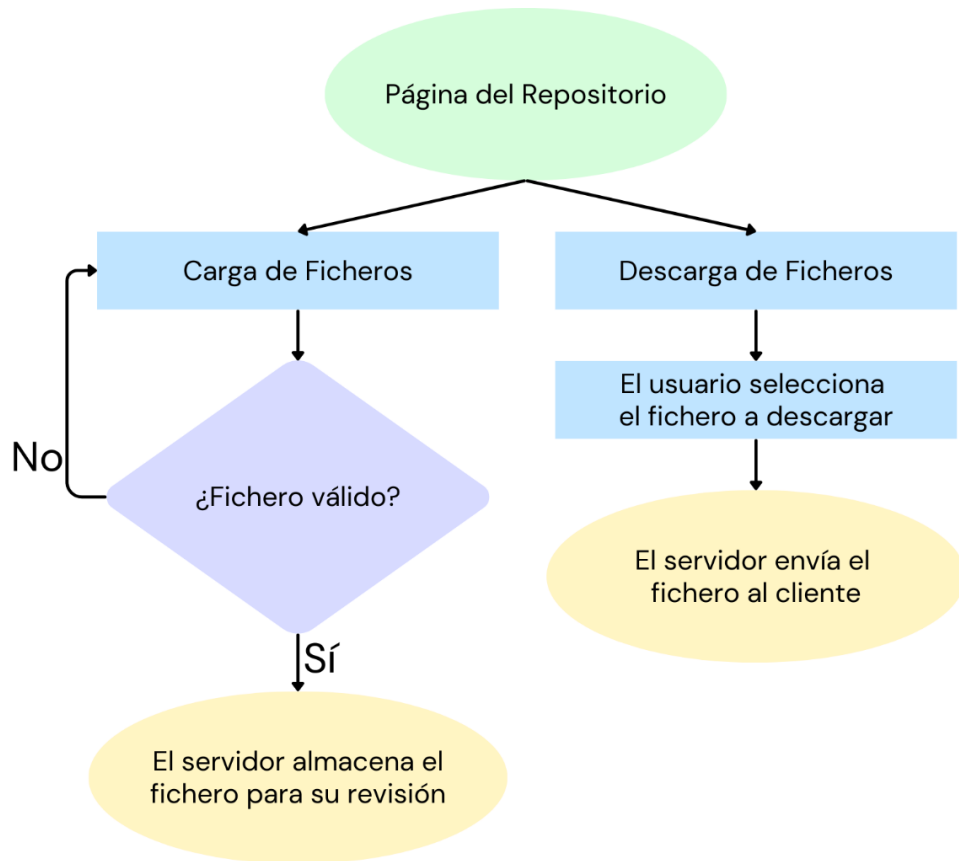


Ilustración 13- Diagrama de Flujo de la Gestión del Archivos I

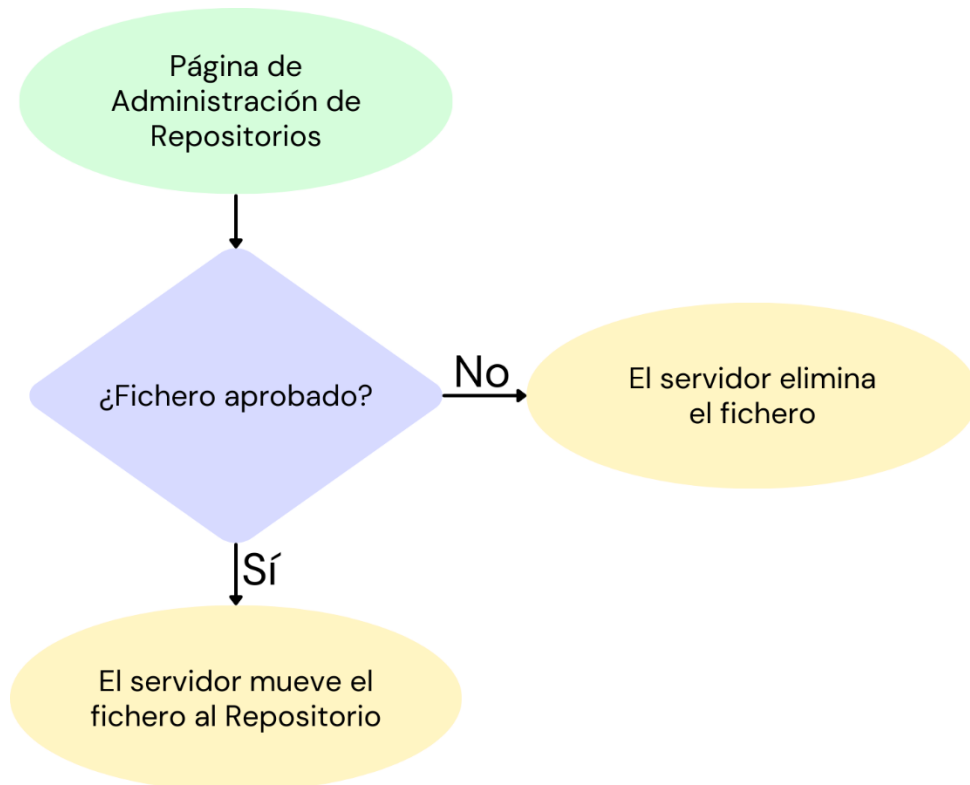


Ilustración 14- Diagrama de Flujo de la Gestión de Archivos II

6.4. Despliegue de Dockers

Funcionamiento

Este componente se encarga de implementar y gestionar contenedores Docker utilizando Podman. Utiliza scripts de automatización para crear, ejecutar y gestionar contenedores de aplicaciones de manera eficiente.

Modo de Operación

El componente utiliza Podman para gestionar la creación y ejecución de contenedores Docker que alojan los entornos seguros para la resolución de ejercicios.

Problemas Encontrados y Solucionados

Un desafío fue conseguir visualizar el GUI del sistema operativo que corre en el Docker. Para ello, **RELLENAR CON LA SOLUCION UTILIZADA**

Alternativas Desechadas

Inicialmente, se consideró el uso de Docker como herramienta de despliegue, pero se optó por Podman debido a sus características de seguridad.

Integración con el Resto de Componentes

Este componente se integra con el proceso de despliegue y automatización del sistema, garantizando que los contenedores Docker se creen y ejecuten correctamente según las especificaciones del proyecto.

Diagrama de Flujo

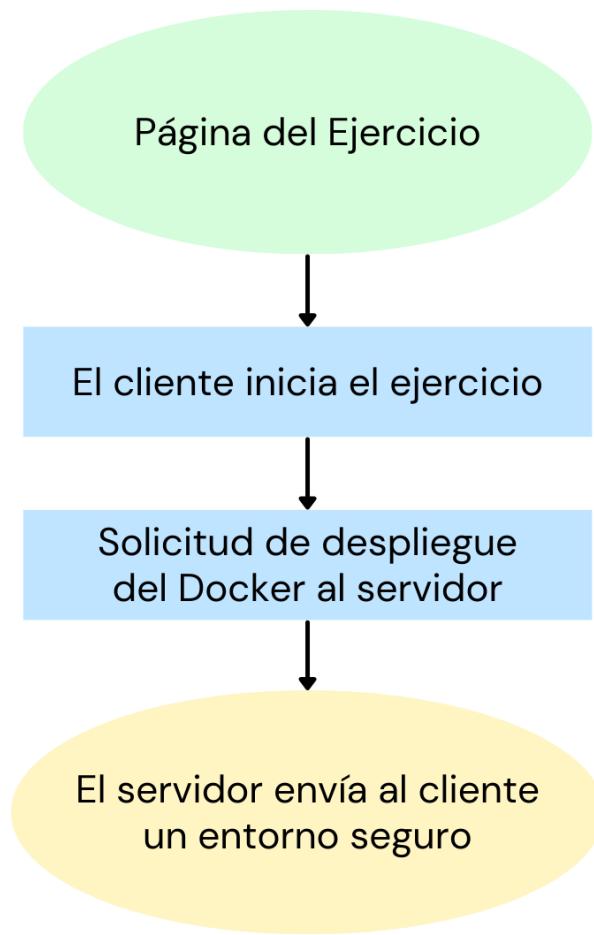


Ilustración 15- Diagrama de Flujo del Despliegue de Dockers

7. Pruebas del Sistema

Para asegurar el correcto funcionamiento del sistema y la integración de sus componentes, se ha seguido un plan de pruebas exhaustivo. Este plan incluye pruebas unitarias, pruebas de integración, pruebas funcionales, pruebas de rendimiento y pruebas de usuario. A continuación, se detallan las metodologías y enfoques utilizados para cada tipo de prueba:

Pruebas Unitarias

- Objetivo: Validar el correcto funcionamiento de las unidades individuales de código.
- Herramienta: Jest.
- Descripción: Se realizan pruebas para cada función o módulo del sistema, verificando que produzcan los resultados esperados.
- Cobertura: Todas las funciones críticas del sistema, incluyendo autenticación de usuarios, gestión de archivos, consultas a la base de datos y cálculo de puntuaciones.

Pruebas de Integración

- Objetivo: Verificar que los diferentes componentes del sistema funcionen correctamente juntos.
- Herramientas: Thunder Client.
- Descripción: Se realizan pruebas para las interacciones entre los componentes del sistema, como las solicitudes HTTP de React a Node.js y las consultas de Node.js a MySQL.
- Cobertura: Cubre casos de uso comunes, como el flujo de autenticación, la carga y descarga de archivos, y la actualización de las puntuaciones del usuario.

Pruebas Funcionales

- Objetivo: Validar que el sistema cumple con los requisitos funcionales especificados.
- Herramientas: Loadtest.
- Descripción: Se escriben pruebas que simulan la interacción del usuario con la aplicación, verificando que las funciones clave funcionen de acuerdo con lo esperado.
- Cobertura: La funcionalidad general de la aplicación desde el punto de vista del usuario final.

Pruebas de Rendimiento

- Objetivo: Evaluar el rendimiento del sistema bajo diferentes condiciones de carga.
- Herramientas: Loadtest.
- Descripción: Se simulan múltiples usuarios que acceden y utilizan el sistema simultáneamente para medir el tiempo de respuesta, el uso de recursos y la escalabilidad.
- Cobertura: Escenarios de alta concurrencia, operaciones de carga y descarga de archivos grandes. También se da la ejecución simultánea de múltiples consultas a la base de datos.

Pruebas de Usuario

- Objetivo: Recopilar feedback sobre la experiencia del usuario y detectar problemas de usabilidad.
- Herramientas: Sesiones de prueba con usuarios reales.
- Descripción: Se invita a usuarios reales a interactuar con la aplicación, donde interactuarán libremente para posteriormente dar unas valoraciones.
- Cobertura: Todo el flujo de uso de la aplicación, desde el registro hasta el uso avanzado de sus funciones.

7.1. Pruebas Unitarias y de Integración

addPoints.js

```
1 export const addPoints = async ({ username, points }) => {
2   const response = await fetch('http://localhost:3001/auth/addPoints', {
3     method: 'POST',
4     headers: {
5       'Content-Type': 'application/json',
6     },
7     body: JSON.stringify({ username, points }),
8   });
9
10  const data = await response.json();
11  return data;
12 }
```

Ilustración 16 - addPoints.js

Esta función se encarga de realizar una petición a la API para añadir puntos a un usuario específico.

Descripción de la prueba

- Prueba exitosa: Se ha mockeado la respuesta de la API para simular una respuesta exitosa que actualice correctamente los puntos del usuario.
- Prueba errónea: Se ha mockeado un error en la llamada a la API para verificar que la función maneja adecuadamente los errores.

Resultado Thunder Client

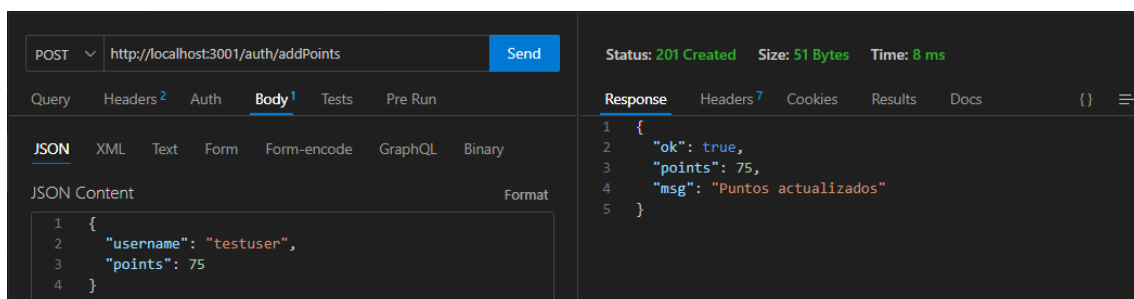


Ilustración 17 - addPoints.js (Thunder Client)

approveAdminFiles.js

```

1 export const approveAdminFiles = async ({ username, fileName, selectedCategory, selectedDifficulty }) => {
2   const response = await fetch('http://localhost:3001/approveAdminFiles', {
3     method: 'POST',
4     headers: {
5       'Content-Type': 'application/json',
6     },
7     body: JSON.stringify({ username, fileName, selectedCategory, selectedDifficulty }),
8   });
9
10  const data = await response.json();
11  return data;
12 }

```

Ilustración 18 - approveAdminFiles.js

Esta función permite a los administradores aprobar los ficheros subidos por los usuarios.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la aprobación exitosa de ficheros, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba de archivo no encontrado: Se ha aprobado el escenario donde el archivo solicitado no se encontraba en el servidor, asegurando que la función maneje correctamente esta situación.
- Prueba errónea: Se ha verificado que la función reaccione correctamente ante errores.

Resultado Thunder Client

The screenshot shows the Thunder Client interface. At the top, a POST request is configured to 'http://localhost:3001/approveAdminFiles'. The 'Body' tab is selected, showing a JSON payload: { "username": "testuser", "fileName": "test.zip", "selectedCategory": "hash", "selectedDifficulty": "easy" }. The status bar indicates a '200 OK' response with a size of 11 Bytes and a time of 4 ms. The 'Response' tab shows the returned JSON: { "ok": true }.

Ilustración 19 - approveAdminFiles.js (Thunder Client)

deleteAdminFiles.js

```

1  export const deleteAdminFiles = async ({ username, fileName }) => {
2    const response = await fetch('http://localhost:3001/deleteAdminFiles', {
3      method: 'POST',
4      headers: {
5        'Content-Type': 'application/json',
6      },
7      body: JSON.stringify({ username, fileName }),
8    });
9
10   const data = await response.json();
11   return data;
12 }

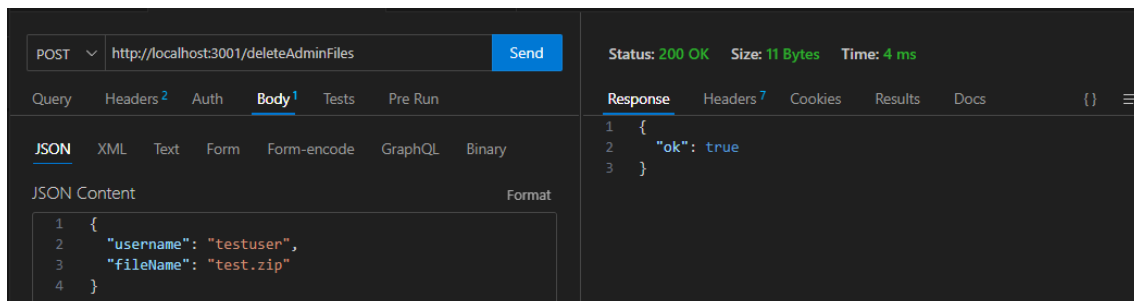
```

Ilustración 20 - deleteAdminFiles.js

Esta función permite a los administradores eliminar un fichero del sistema de archivos local administrativo.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la eliminación exitosa de un fichero, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client*Ilustración 21 - deleteAdminFiles.js (Thunder Client)*

deleteFiles.js

```

1  export const deleteFiles = async ({ uid, fileName, category, difficulty }) => {
2      const response = await fetch('http://localhost:3001/deleteFiles', {
3          method: 'POST',
4          headers: {
5              'Content-Type': 'application/json',
6          },
7          body: JSON.stringify({ uid, fileName, category, difficulty }),
8      });
9
10     const data = await response.json();
11     return data;
12 }

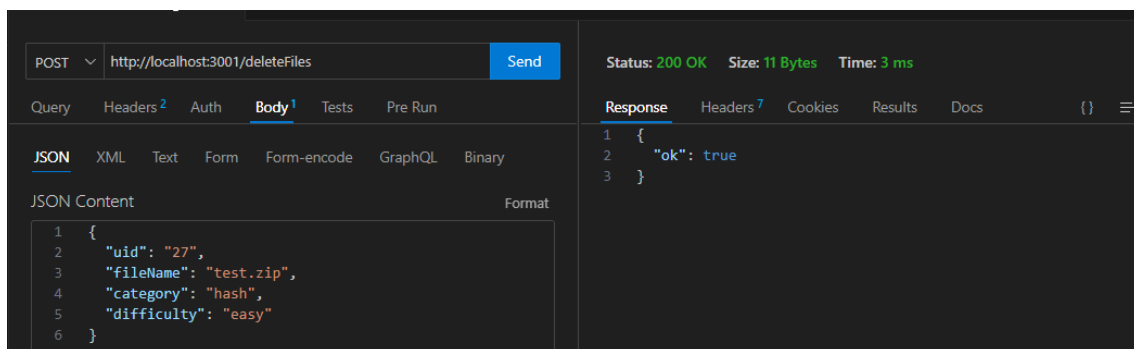
```

Ilustración 22 - deleteFiles.js

Esta función permite a los usuarios eliminar un fichero del sistema de archivos local.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la eliminación exitosa de un fichero, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client*Ilustración 23 - deleteFiles.js (Thunder Client)*

deleteUsers.js

```
1 export const deleteUser = async ({ uid }) => {
2   const response = await fetch('http://localhost:3001/auth/deleteUser', {
3     method: 'POST',
4     headers: {
5       'Content-Type': 'application/json',
6     },
7     body: JSON.stringify({ uid }),
8   });
9
10  const data = await response.json();
11  return data;
12 }
```

Ilustración 24 - deleteUsers.js

Esta función elimina un usuario de la base de datos y todas sus asociaciones.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la eliminación exitosa de un usuario de la base de datos, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

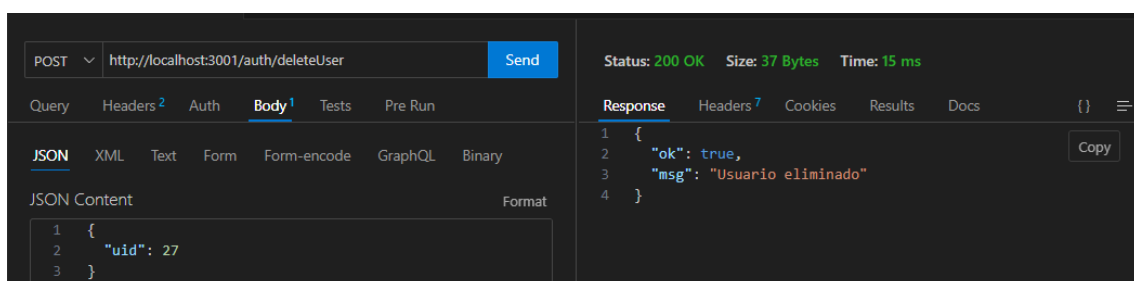


Ilustración 25 - deleteUsers.js (Thunder Client)

downloadAdminFiles.js

```

1  export const downloadAdminFiles = async ({ username, fileName }) => {
2      const response = await fetch('http://localhost:3001/downloadAdminFiles', {
3          method: 'POST',
4          headers: {
5              'Content-Type': 'application/json',
6          },
7          body: JSON.stringify({ username, fileName }),
8      });
9
10     const blobResponse = new Response(response.body);
11     const data = await blobResponse.blob();
12     return data;
13 }

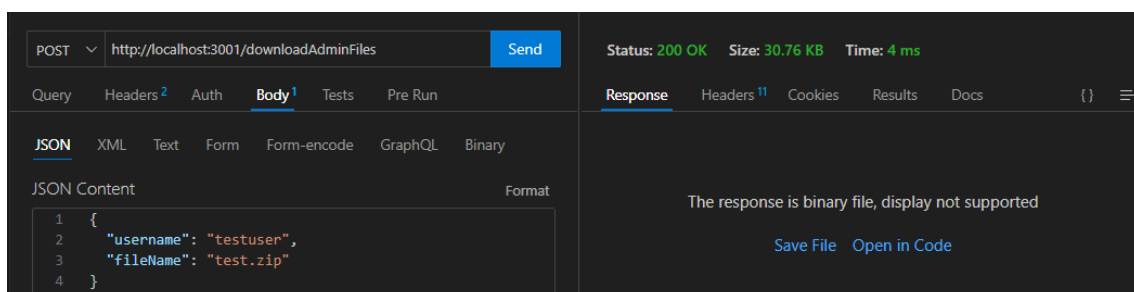
```

Ilustración 26 - downloadAdminFiles.js

Esta función permite a los administradores descargar ficheros asociados a un usuario específico.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la descarga exitosa de un fichero, validando que la función maneje adecuadamente la respuesta positiva del servidor y devuelva el fichero en formato “blob”.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client*Ilustración 27 - downloadAdminFiles.js (Thunder Client)*

downloadFiles.js

```

1  export const downloadFiles = async ({ username, fileName, category, difficulty }) => {
2      const response = await fetch('http://localhost:3001/downloadFiles', {
3          method: 'POST',
4          headers: {
5              'Content-Type': 'application/json',
6          },
7          body: JSON.stringify({ username, fileName, category, difficulty }),
8      });
9
10     const blobResponse = new Response(response.body);
11     const data = await blobResponse.blob();
12     return data;
13 }

```

Ilustración 28 - downloadFiles.js

Esta función permite a los usuarios descargar ficheros asociados a un usuario específico.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la descarga exitosa de un fichero, validando que la función maneje adecuadamente la respuesta positiva del servidor y devuelva el fichero en formato “blob”.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

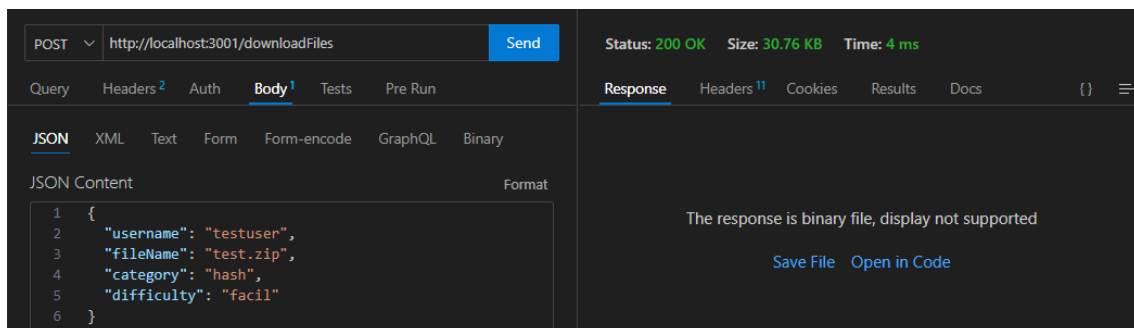


Ilustración 29 - downloadFiles.js (Thunder Client)

getAdminFiles.js

```

1  export const getAdminFiles = async () => {
2      const response = await fetch('http://localhost:3001/getAdminFiles', {
3          method: 'GET',
4          headers: {
5              'Content-Type': 'application/json',
6          },
7      });
8
9      const data = await response.json();
10     return data;
11 }

```

Ilustración 30 - getAdminFiles.js

Esta función devuelve los ficheros alojados en el sistema de archivos local administrativo.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa de la información, validando que la función maneje adecuadamente la respuesta positiva del servidor y formatee los datos correctamente.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

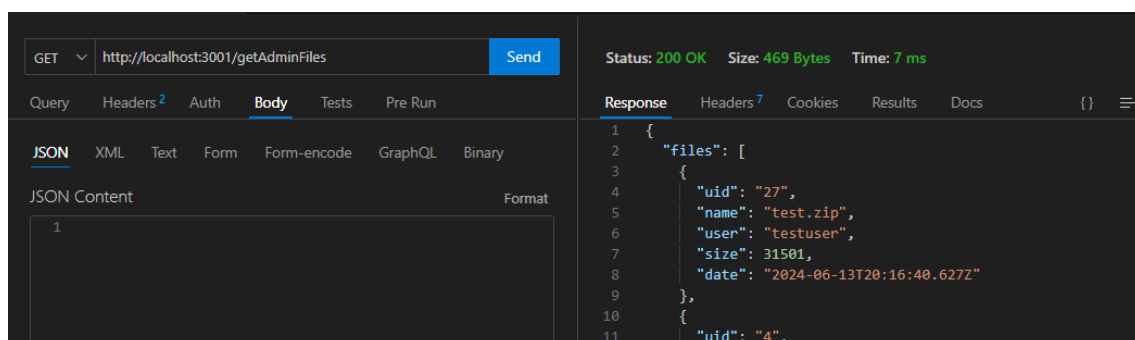


Ilustración 31 - getAdminFiles.js (Thunder Client)

getAvatars.js

```
1 export const getAvatars = async ({ uid }) => {
2   const response = await fetch('http://localhost:3001/getAvatars', {
3     method: 'POST',
4     headers: {
5       'Content-Type': 'application/json',
6     },
7     body: JSON.stringify({ uid }),
8   });
9
10  const data = await response.blob();
11  return data;
12 }
```

Ilustración 32 - getAvatars.js

Esta función devuelve el avatar asociado al usuario basándose en su uid.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la recepción exitosa del avatar del usuario, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba de avatar por defecto: Se ha simulado la solicitud de un avatar sin dar el uid, validando que la función devuelva el avatar por defecto.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

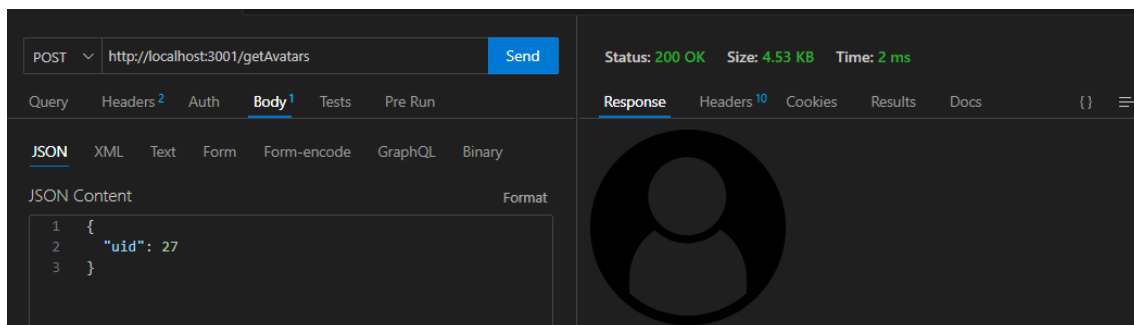


Ilustración 33 - getAvatars.js (Thunder Client)

getFiles.js

```

1  export const getFiles = async () => {
2      const response = await fetch('http://localhost:3001/getFiles', {
3          method: 'GET',
4          headers: {
5              'Content-Type': 'application/json',
6          },
7      });
8
9      const data = await response.json();
10     return data;
11 }

```

Ilustración 34 - getFiles.js

Esta función devuelve los ficheros alojados en el sistema de archivos local.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa de la información, validando que la función maneje adecuadamente la respuesta positiva del servidor y formatee los datos correctamente.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

The screenshot shows the Thunder Client interface. The top bar displays the method 'GET', the URL 'http://localhost:3001/getFiles', and a 'Send' button. Below this, the 'Body' tab is selected, showing the JSON response. The response is a JSON array with two objects. The first object has properties: 'uid': '27', 'name': 'test.zip', 'user': 'testuser', 'size': 31501, 'date': '2024-06-13T20:12:34.635Z', 'category': 'hash', and 'difficulty': 'facil'. The second object has 'uid': '7'.

```

1  {
2      "files": [
3          {
4              "uid": "27",
5              "name": "test.zip",
6              "user": "testuser",
7              "size": 31501,
8              "date": "2024-06-13T20:12:34.635Z",
9              "category": "hash",
10             "difficulty": "facil"
11         },
12         {
13             "uid": "7",

```

Ilustración 35 - getFiles.js (Thunder Client)

getLogin.js

```
1  import Swal from "sweetalert2";
2
3  export const getLogin = async ({ email, password }) => {
4    const response = await fetch('http://localhost:3001/auth/login', {
5      method: 'POST',
6      headers: {
7        'Content-Type': 'application/json',
8      },
9      body: JSON.stringify({ email, password }),
10   });
11
12   if (!response.ok) {
13     Swal.fire("Error", "Usuario o contraseña incorrectos", "error");
14   }
15
16   const data = await response.json();
17   return data;
18 }
```

Ilustración 36 - getLogin.js

Esta función envía las credenciales del usuario y maneja la respuesta del servidor si es satisfactorio el inicio de sesión o no.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa del inicio de sesión, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba de credenciales incorrectas: Se ha verificado que la función muestre un mensaje de error cuando se introducen credenciales incorrectas.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

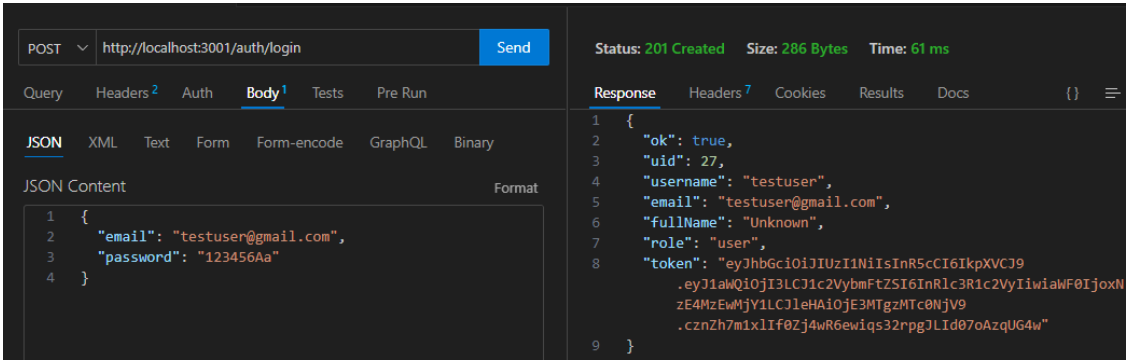


Ilustración 37 - getLogin.js (Thunder Client)

getOwnFiles.js

```

1  export const getOwnFiles = async ({ uid }) => {
2      const response = await fetch('http://localhost:3001/getOwnFiles', {
3          method: 'POST',
4          headers: {
5              'Content-Type': 'application/json',
6          },
7          body: JSON.stringify({ uid }),
8      });
9
10     const data = await response.json();
11     return data;
12 }

```

Ilustración 38 - getOwnFiles.js

Esta función devuelve los ficheros alojados en el sistema de archivos local en base al uid del usuario.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa de la información, validando que la función maneje adecuadamente la respuesta positiva del servidor y formatee los datos correctamente.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

The screenshot shows the Thunder Client interface. On the left, the request is configured as a POST to `http://localhost:3001/getOwnFiles` with a JSON body: `{ "uid": 27 }`. On the right, the response is displayed as a 200 OK status with a size of 135 Bytes and a time of 3 ms. The response body is a JSON array of file objects:

```

1  {
2    "files": [
3      {
4        "name": "test.zip",
5        "user": "testuser",
6        "size": 31501,
7        "date": "2024-06-13T20:12:34.635Z",
8        "category": "hash",
9        "difficulty": "facil"
10     }
11   ]
12 }

```

Ilustración 39 - getOwnFiles.js (Thunder Client)

getPoints.js

```

1  export const getPoints = async ({ uid }) => {
2      const response = await fetch('http://localhost:3001/auth/getPoints', {
3          method: 'POST',
4          headers: {
5              'Content-Type': 'application/json',
6          },
7          body: JSON.stringify({ uid }),
8      });
9
10     const data = await response.json();
11     return data;
12 }

```

Ilustración 40 - getPoints.js

Esta función devuelve los puntos y tareas resueltas en base al uid del usuario.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa de la puntuación, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba de usuario no encontrado: Se ha verificado si la función maneja correctamente escenarios en los que no se encuentra un usuario con ese uid.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

The screenshot shows the Thunder Client interface. On the left, the request is configured as a POST to `http://localhost:3001/auth/getPoints` with a body of `{ "uid": 27 }`. On the right, the response is displayed with status `201 Created`, size `212 Bytes`, and time `3 ms`. The response body is a JSON object:

```

{
  "ok": true,
  "points": 100,
  "hashEasyTask1": 0,
  "hashEasyTask2": 0,
  "hashEasyTask3": 0,
  "hashHardTask1": 0,
  "hashHardTask2": 0,
  "steganographyEasyTask1": 0,
  "steganographyHardTask1": 0,
  "phishingEasyTask1": 0,
  "phishingHardTask1": 0
}

```

Ilustración 41 - getPoints.js (Thunder Client)

getRanking.js

```

1  export const getRanking = async () => {
2      const response = await fetch('http://localhost:3001/auth/getRanking', {
3          method: 'GET',
4          headers: {
5              'Content-Type': 'application/json',
6          },
7      });
8
9      const data = await response.json();
10     return data;
11 }

```

Ilustración 42 - getRanking.js

Esta función devuelve los puntos de los 10 usuarios con más puntos ordenados en base a dichos puntos.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa de las puntuaciones, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

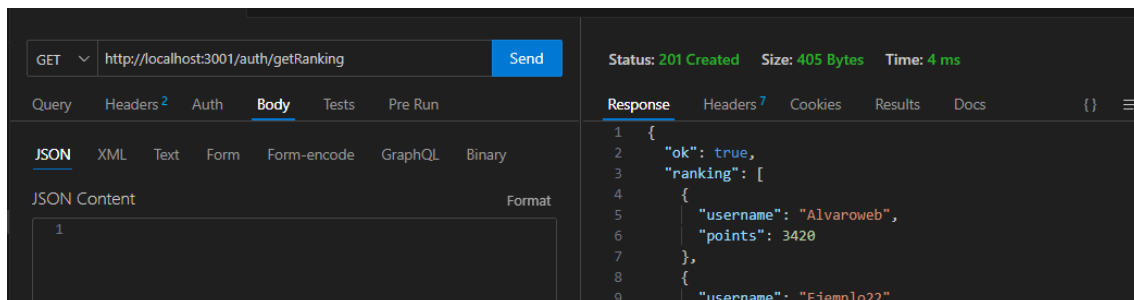


Ilustración 43 - getRanking.js (Thunder Client)

getRegister.js

```

1 import Swal from 'sweetalert2'
2
3 export const getRegister = async ({ username, email, password }) => {
4   const response = await fetch('http://localhost:3001/auth/register', {
5     method: 'POST',
6     headers: {
7       'Content-Type': 'application/json',
8     },
9     body: JSON.stringify({ username, email, password }),
10  });
11
12  if (!response.ok) {
13    Swal.fire('Error', 'Ya existe un usuario con ese Username o Email', 'error');
14  }
15
16  return response.json();
17 }

```

Ilustración 44 - getRegister.js

Esta función envía las credenciales del usuario y maneja la respuesta del servidor si es satisfactorio el inicio de sesión o no.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa del registro de un usuario, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

POST Send

Query Headers² Auth **Body¹** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "username": "testuser",
3   "email": "testuser@gmail.com",
4   "password": "123456Aa"
5 }
```

Status: **201 Created** Size: **251 Bytes** Time: **68 ms**

Response Headers⁷ Cookies Results Docs {} ≡

```
1 {
2   "ok": true,
3   "uid": 29,
4   "username": "testuser",
5   "email": "testuser@gmail.com",
6   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
    eyJ1IjAwQioiIjI5LjC1c2VybWZtZSI6InRlc3R1c2V5IiwiaWF0Ijox
    zE4MzEwZnIwLjC1eHA1OjE3MTgzMTc5MjB9
    .q8V7PpfH3AfrYwLDR6UoUofeAFo_Dx9M-dxaYDgwhaaak"
7 }
```

Copy

Ilustración 45 - getRegister.js (Thunder Client)

getRenewToken.js

```

1  export const getRenewToken = async ({ uid, token }) => {
2    const response = await fetch('http://localhost:3001/auth/renew', {
3      method: 'POST',
4      headers: {
5        'Content-Type': 'application/json',
6      },
7      body: JSON.stringify({ uid, token }),
8    });
9
10   const data = await response.json();
11   return data;
12 }

```

Ilustración 46 - getRenewToken.js

Esta función envía el uid y el token del usuario para comprobar si sigue siendo válido.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa del token del usuario, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

The screenshot shows the Thunder Client interface. On the left, the 'Body' tab is selected, displaying the JSON content of the request. The request is a POST to 'http://localhost:3001/auth/renew' with a body containing 'uid' and 'token'. On the right, the 'Response' tab is selected, showing the server's response. The status is '200 OK', the size is '286 Bytes', and the time is '4 ms'. The response body is a JSON object with the following fields: 'ok' (true), 'uid' (29), 'username' (testuser), 'email' (testuser@gmail.com), 'fullName' (Unknown), 'role' (user), and 'token' (a long alphanumeric string).

Ilustración 47 - getRenewToken.js (Thunder Client)

getUpdateTasks.js

```

1  export const getUpdateTasks = async (taskUpdates) => {
2      const response = await fetch('http://localhost:3001/auth/updatePoints', {
3          method: 'POST',
4          headers: {
5              'Content-Type': 'application/json',
6          },
7          body: JSON.stringify(taskUpdates),
8      });
9
10     const data = await response.json();
11     return data;
12 }

```

Ilustración 48 - getUpdateTasks.js

Esta función actualiza los puntos y las tareas de un usuario en la base de datos.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa de la actualización de las tareas del usuario, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

The screenshot shows the Thunder Client interface. The top bar indicates a POST request to `http://localhost:3001/auth/updatePoints` with a status of **201 Created**, size of **70 Bytes**, and time of **14 ms**. The 'Body' tab is selected, showing the request body in JSON format: `{ "uid": 29, "points": 175, "hashEasyTask1": 1 }`. The 'Response' tab is also visible, showing the response body in JSON format: `{ "ok": true, "points": 175, "hashEasyTask1": 1, "msg": "Puntos actualizados" }`.

Ilustración 49 - getUpdateTask.js (Thunder Client)

getUpdateTraceability.js

```

1  export const getUpdateTraceability = async (traceabilityUpdates) => {
2      const response = await fetch('http://localhost:3001/auth/updateTraceability', {
3          method: 'POST',
4          headers: {
5              'Content-Type': 'application/json',
6          },
7          body: JSON.stringify(traceabilityUpdates),
8      });
9
10     const data = await response.json();
11     return data;
12 }

```

Ilustración 50 - getUpdateTraceability.js

Esta función actualiza la trazabilidad de un usuario en la base de datos.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa de la trazabilidad de las tareas del usuario, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

The screenshot shows the Thunder Client interface. On the left, the request is configured as a POST to `http://localhost:3001/auth/updateTraceability`. The 'Body' tab is selected, showing a JSON object: `{ "uid": 29, "hashEasyTask1": 65 }`. On the right, the 'Response' tab shows the server's reply: `{ "ok": true, "msg": "Trazabilidad actualizada" }`. The status bar at the top indicates a successful `201 Created` response.

Ilustración 51 - getUpdateTraceability.js (Thunder Client)

getUpdateUser.js

```

1  import Swal from 'sweetalert2'
2
3  export const getUpdateUser = async ({ uid, username, email, fullName }) => {
4    const response = await fetch('http://localhost:3001/auth/updateUser', {
5      method: 'POST',
6      headers: {
7        'Content-Type': 'application/json',
8      },
9      body: JSON.stringify({ uid, username, email, fullName }),
10   });
11
12   if (!response.ok) {
13     Swal.fire('Error', 'Ya existe un usuario con ese Username o Email', 'error');
14   }
15
16   return response.json();
17 }

```

Ilustración 52 - getUpdateUser.js

Esta función actualiza los datos personales y de la cuenta del usuario en la base de datos.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa de la actualización de los datos del usuario, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

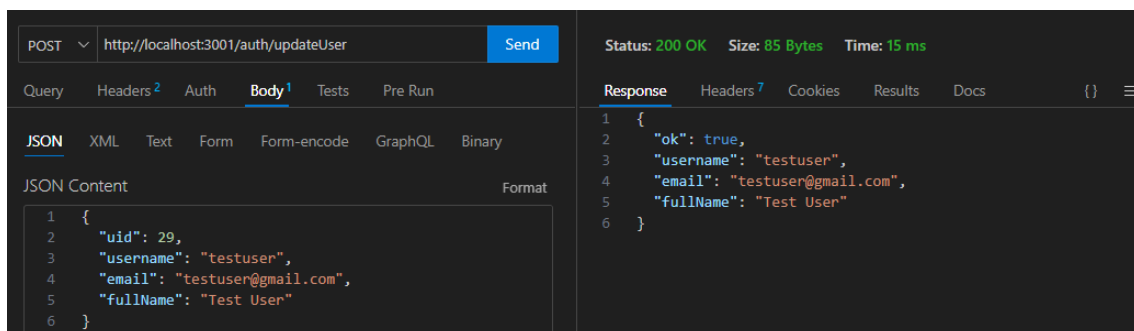


Ilustración 53 - getUpdateUser.js (Thunder Client)

updatePassword.js

```

1  import Swal from 'sweetalert2'
2
3  export const updatePassword = async ({ uid, oldPassword, newPassword }) => {
4    const response = await fetch('http://localhost:3001/auth/updatePassword', {
5      method: 'POST',
6      headers: {
7        'Content-Type': 'application/json',
8      },
9      body: JSON.stringify({ uid, oldPassword, newPassword }),
10   });
11
12   if (!response.ok) {
13     return Swal.fire('Error', 'Contraseña incorrecta', 'error');
14   }
15
16   return response.json();
17 }

```

Ilustración 54 - updatePassword.js

Esta función actualiza la contraseña del usuario en la base de datos.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa de la actualización de la contraseña del usuario, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

The screenshot shows the Thunder Client interface. On the left, the request is configured as a POST to `http://localhost:3001/auth/updatePassword`. The 'Body' tab is selected, showing a JSON payload: `{ "uid": 29, "oldPassword": "123456Aa", "newPassword": "12345678Aa" }`. On the right, the 'Response' tab shows a successful status of `200 OK` with a size of `43 Bytes` and a time of `119 ms`. The response body is a JSON object: `{ "ok": true, "msg": "Contraseña actualizada" }`.

Ilustración 55 - updatePassword.js (Thunder Client)

uploadAvatars.js

```

1  import Swal from 'sweetalert2'
2
3  export const uploadAvatars = async (uid, file) => {
4      const formData = new FormData();
5      formData.append('file', file);
6      formData.append('uid', uid);
7
8      const response = await fetch('http://localhost:3001/uploadAvatars', {
9          method: 'POST',
10         body: formData,
11     });
12
13     if (response.status === 412) {
14         Swal.fire('Error', 'Sólo se pueden subir archivos JPG y PNG', 'error');
15     } else if (!response.ok) {
16         Swal.fire('Error', 'Error al subir el archivo', 'error');
17     } else {
18         Swal.fire('Éxito', ' ', 'success');
19         Swal.fire({
20             html: `<p>Avatar subido correctamente.</p>`,
21             icon: 'success',
22             showConfirmButton: false,
23             timer: 3000
24         });
25     }
26
27     return response.blob();
28 }

```

Ilustración 56 - uploadAvatars.js

Esta función maneja el proceso de subida de imágenes al servidor para usarlas como foto de perfil.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa de la subida de una imagen como avatar, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba de no subida de archivo: Se ha verificado que la función maneja correctamente casos en los que no se suben archivos.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

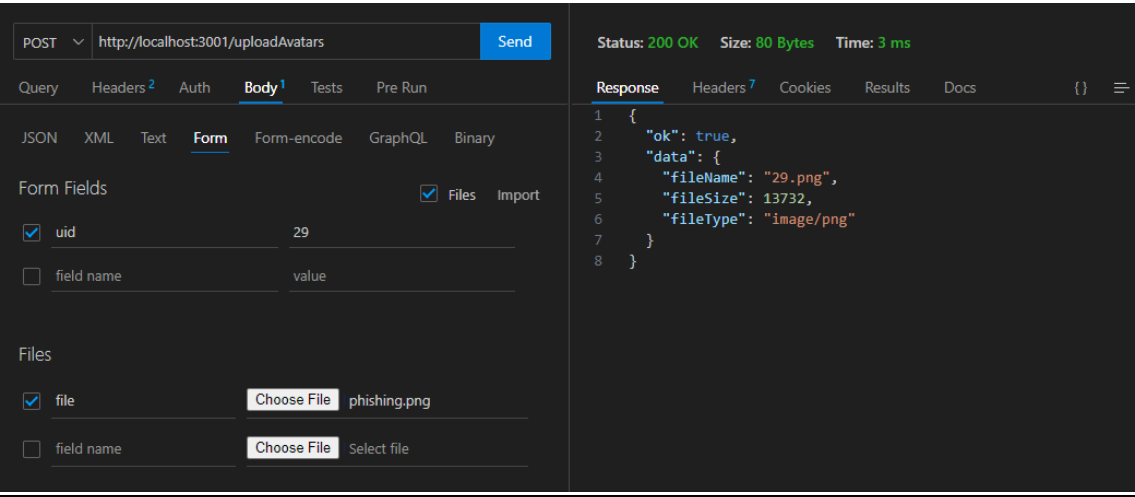


Ilustración 57 - uploadAvatars.js (Thunder Client)

uploadFiles.js

```

1  import Swal from 'sweetalert2'
2
3  export const uploadFiles = async (uid, file) => {
4      const formData = new FormData();
5      formData.append('file', file);
6      formData.append('uid', uid);
7
8      const response = await fetch('http://localhost:3001/uploadFiles', {
9          method: 'POST',
10         body: formData,
11     });
12
13     if (response.status === 412) {
14         Swal.fire('Error', 'Sólo se pueden subir archivos comprimidos', 'error');
15     } else if (!response.ok) {
16         Swal.fire('Error', 'Error al subir el archivo', 'error');
17     } else {
18         Swal.fire('Éxito', ' ', 'success');
19         Swal.fire({
20             html: `<p>Archivo subido correctamente.</p>
21                 <br>
22                 <p>Será revisado por nuestros administradores.</p>`,
23             icon: 'success',
24             showConfirmButton: false,
25             timer: 3000
26         });
27     }
28
29     return response.json();
30 }

```

Ilustración 58 - uploadFiles.js

Esta función maneja el proceso de subida de ficheros al servidor por parte de los usuarios.

Descripción de la prueba

- Prueba exitosa: Se ha simulado la solicitud y respuesta exitosa de la subida de un fichero, validando que la función maneje adecuadamente la respuesta positiva del servidor.
- Prueba de no subida de fichero: Se ha verificado que la función maneja correctamente casos en los que no se suben ficheros.
- Prueba errónea: Se ha verificado que la función reaccione adecuadamente ante errores.

Resultado Thunder Client

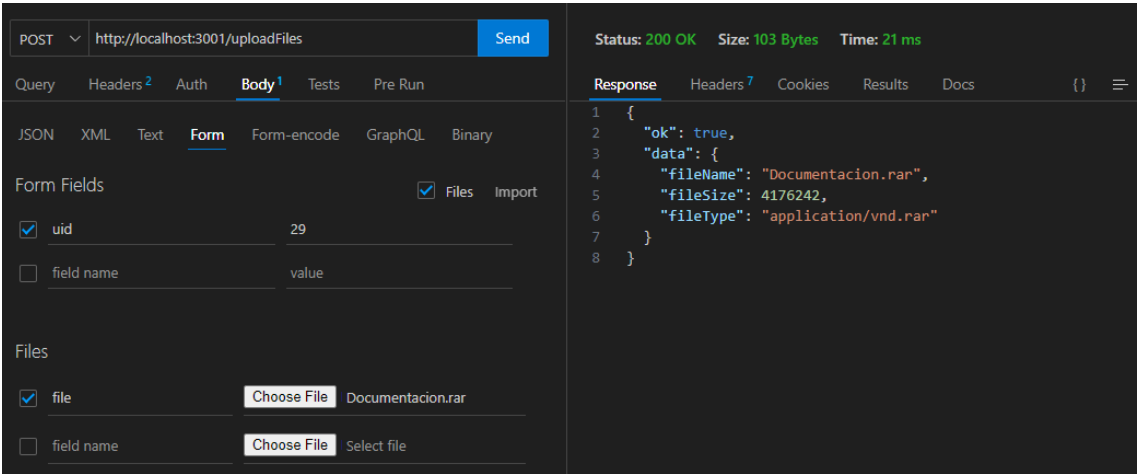
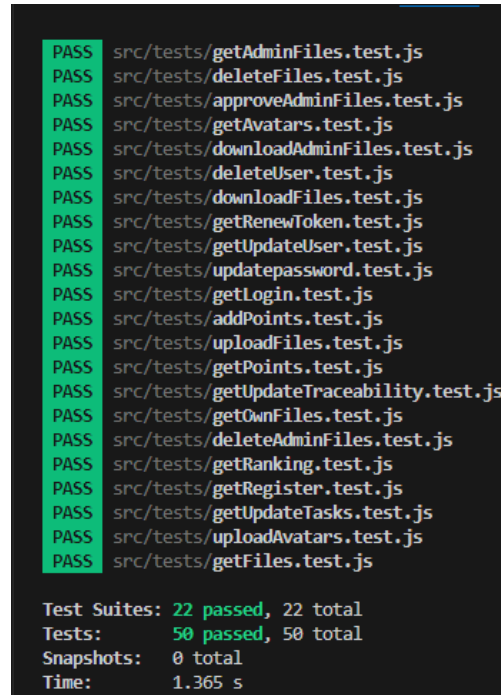


Ilustración 59 - uploadFiles.js (Thunder Client)

Resultados Generales de Jest

Como podemos observar en la siguiente imagen, todas las pruebas unitarias, siendo un total de 22 scripts con 50 tests, han sido superadas satisfactoriamente.



```
PASS src/tests/getAdminFiles.test.js
PASS src/tests/deleteFiles.test.js
PASS src/tests/approveAdminFiles.test.js
PASS src/tests/getAvatars.test.js
PASS src/tests/downloadAdminFiles.test.js
PASS src/tests/deleteUser.test.js
PASS src/tests/downloadFiles.test.js
PASS src/tests/getRenewToken.test.js
PASS src/tests/getUpdateUser.test.js
PASS src/tests/updatepassword.test.js
PASS src/tests/getLogin.test.js
PASS src/tests/addPoints.test.js
PASS src/tests/uploadFiles.test.js
PASS src/tests/getPoints.test.js
PASS src/tests/getUpdateTraceability.test.js
PASS src/tests/getOwnFiles.test.js
PASS src/tests/deleteAdminFiles.test.js
PASS src/tests/getRanking.test.js
PASS src/tests/getRegister.test.js
PASS src/tests/getUpdateTasks.test.js
PASS src/tests/uploadAvatars.test.js
PASS src/tests/getFiles.test.js

Test Suites: 22 passed, 22 total
Tests:       50 passed, 50 total
Snapshots:   0 total
Time:        1.365 s
```

Ilustración 60 - Resultados Generales de Jest

7.2. Pruebas Funcionales y de Rendimiento

En esta sección se muestra la realización de solicitudes individuales y en masa a diferentes elementos de la web, para así simular su funcionalidad y rendimiento de cara a su lanzamiento, cuando recibirá numerosas consultas simultáneas.

Consultas a la Web

Comando	Número de Peticiones	Número de Usuarios Concurrentes	Tiempo (ms)
<code>loadtest -n 1 -c 1 http://localhost:3000</code>	1	1	2'8
<code>loadtest -n 1000 -c 10 http://localhost:3000</code>	1000	10	921

Tabla 1 - Consultas a la Web

Consultas a la Base de Datos

Comando	Número de Peticiones	Número de Usuarios Concurrentes	Tiempo (ms)
<code>loadtest -n 1 -c 1 -m POST -T "application/json" -p '{"email": "testuser@gmail.com", "password": "123456Aa"}' http://localhost:3001/auth/login</code>	1	1	382'9
<code>loadtest -n 1000 -c 10 -m POST -T "application/json" -p '{"email": "testuser@gmail.com", "password": "123456Aa"}' http://localhost:3001/auth/login</code>	1000	10	55.409

Tabla 2 - Consultas a la Base de Datos

Consultas al Sistema de Almacenamiento

Comando	Número de Peticiones	Número de Usuarios Concurrentes	Tiempo (ms)
<code>loadtest -n 1 -c 1 -m POST -T "application/json" -P '{"username": "testuser", "fileName": "test.zip", "category": "hash", "difficulty": "facil"}' http://localhost:3001/downloadFiles</code>	1	1	7'8
<code>loadtest -n 1000 -c 10 -m POST -T "application/json" -P '{"username": "testuser", "fileName": "test.zip", "category": "hash", "difficulty": "facil"}' http://localhost:3001/downloadFiles</code>	1000	10	2.080'2

Tabla 3 - Consultas al Sistema de Almacenamiento

Como podemos observar, el rendimiento de la web con pocos usuarios y peticiones es excelente. Sin embargo, a medida que se aumentan los usuarios y peticiones, el rendimiento baja, aunque sigue siendo bastante bueno.

7.3. Pruebas de Usuario

En esta sección se muestran los resultados de las pruebas de usuario con las que se evalúa el diseño, la usabilidad, la funcionalidad y el desempeño de la aplicación web desde la perspectiva del usuario final. Estas pruebas ayudarán a identificar problemas de usabilidad y funcionalidad, y asegurarán que la aplicación cumpla con las expectativas del usuario.

Cada usuario ha evaluado la aplicación en su conjunto según los siguientes criterios: Diseño, Usabilidad, Funcionalidad, y Desempeño, en una escala del 1 al 5 (1 = Muy Insatisfecho, 5 = Muy Satisfecho).

Usuario	Diseño	Usabilidad	Funcionalidad	Desempeño
Jorge Casanovas	4	5	3	5
Juan Alberto García	3	4	4	4
Jessica Plack	5	5	5	5
Carlos Bruñas	4	2	3	3

Tabla 4 - Pruebas de Usuario

A rasgos generales, las valoraciones son positivas, con una valoración media de 4 puntos.

8. Planificación Temporal

La planificación temporal del desarrollo de la plataforma se ha dividido en varias fases, cada una de las cuales corresponde a un conjunto específico de tareas o hitos. A continuación, se presenta una descripción detallada de cada fase, el tiempo empleado en cada una y una representación gráfica de la planificación en un diagrama de Gantt.

Fases del Proyecto y Tiempos Empleados

Investigación y Análisis Inicial

- Descripción: Investigación de productos similares en el mercado, recopilación de requisitos, análisis de viabilidad.
- Duración: 3 semanas
- Resultado: Informe de análisis y requisitos.

Aprendizaje de Tecnologías

- Descripción: Aprendizaje y familiarización con las tecnologías empleadas en el proyecto.
- Duración: 4 semanas
- Resultado: Conocimiento práctico y preparación para el desarrollo.

Diseño de la Arquitectura del Sistema

- Descripción: Diseño detallado de la arquitectura del sistema y diagramas de bloques.
- Duración: 2 semanas
- Resultado: Documentación de la arquitectura del sistema.

Desarrollo del FrontEnd

- Descripción: Desarrollo de la interfaz de usuario utilizando React y otras tecnologías.
- Duración: 6 semanas
- Resultado: Interfaz de usuario funcional.

Desarrollo del BackEnd

- Descripción: Desarrollo de API RESTful, integración con la base de datos.
- Duración: 6 semanas
- Resultado: API funcional y BackEnd operativo.

Integración del Sistema

- Descripción: Integración del Frontend con el Backend y pruebas de integración.
- Duración: 3 semanas
- Resultado: Sistema integrado y funcional.

Pruebas y Corrección de Fallos

- Descripción: Pruebas exhaustivas del sistema, identificación y corrección de fallos.
- Duración: 4 semanas
- Resultado: Sistema probado y estable.

Despliegue y Hosting

- Descripción: Despliegue del sistema en el servidor y configuración del entorno de producción.
- Duración: 2 semanas
- Resultado: Sistema desplegado y accesible.

Documentación Final

- Descripción: Elaboración de la documentación final del proyecto.
- Duración: 2 semanas
- Resultado: Documentación completa del proyecto.

Diagrama de Gantt

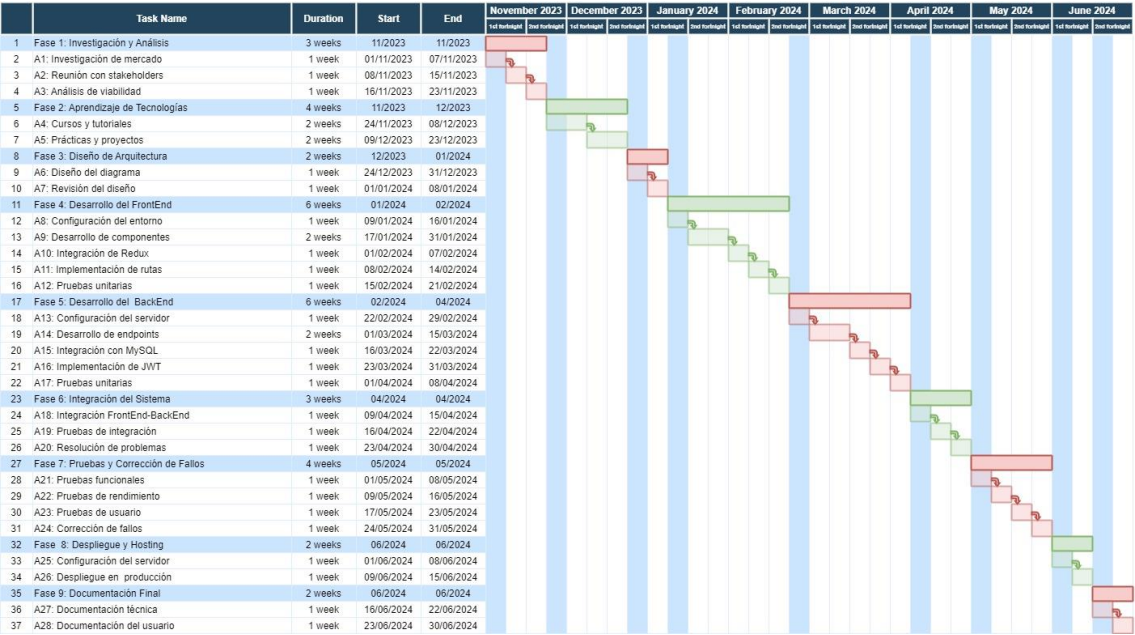


Ilustración 61 - Diagrama de Gantt

Comparación de Tiempos y Análisis

Tareas que Más Tiempo Tomaron

- Desarrollo del Frontend y Backend: Cada una tomó 6 semanas debido a la complejidad de implementar las funcionalidades y asegurar la correcta integración con otras partes del sistema.
- Pruebas y Corrección de Fallos: Tomó 4 semanas, ya que implicó pruebas exhaustivas para garantizar la estabilidad y seguridad del sistema.

Tareas que Menos Tiempo Tomaron

- Diseño de la Arquitectura: Tomó 2 semanas, ya que el diseño fue basado en principios y patrones de diseño ya conocidos.
- Despliegue y Hosting: Tomó 2 semanas, ya que se utilizó un servidor privado de la universidad, lo que simplificó algunos aspectos de la configuración.

9. Costes

En este apartado se presenta una aproximación de los costes materiales para el desarrollo de la plataforma. Los materiales incluyen el software y hardware necesarios para el desarrollo y despliegue del proyecto.

Materiales

Concepto	Cantidad	Precio Unitario (€)	Total (€)
Servidor privado de la universidad	1	Incluido	0
Licencia de software de desarrollo	5	200	1.000
Herramientas de colaboración	5 licencias	100/licencia/año	500
Equipos de desarrollo	5	800	4.000
Total			5.500

Tabla 5 - Coste de los Materiales

Costes de Personal

Para la estimación de los costes de personal, contamos con un equipo compuesto por desarrolladores frontend, backend, y un gestor de proyecto. Asignamos un sueldo mensual a cada trabajador y calculamos el coste total de personal en base al tiempo necesario para el desarrollo del proyecto.

Rol	Número de Personas	Salario Mensual (€)	Coste Total (€)
Desarrollador <u>FrontEnd</u>	2	Incluido	36.000
Desarrollador <u>BackEnd</u>	2	200	36.000
Gestor de Proyecto	1	100/licencia/año	24.000
Total	5		96.000

Tabla 6 - Coste del Personal

Coste Total del Proyecto

Concepto	Coste (€)
Material	5.500
Personal	96.000
Total	101.500

Tabla 7 - Coste Total del Proyecto

La estimación de costes para el desarrollo de la plataforma de la Universidad de Sevilla ha sido cuidadosamente analizada. La inversión total necesaria para la ejecución del proyecto es de aproximadamente 101.500 €. Este coste incluye los materiales necesarios como las licencias de software y equipos de desarrollo, así como los costes de personal involucrados en las diferentes fases del proyecto.

Este análisis de costes resalta la viabilidad económica del proyecto, asegurando que la Universidad de Sevilla puede proceder con el desarrollo de la plataforma sin enfrentar sobrecostes inesperados. Al mismo tiempo, garantiza el uso eficiente y efectivo en el uso de los recursos, permitiendo la creación de una herramienta robusta y funcional para la enseñanza y práctica de la ciberseguridad.

10. Conclusiones

Al inicio de este proyecto, se plantearon una serie de objetivos específicos que guiaron su desarrollo. A continuación, se detallan los aspectos que se han cumplido en relación con estos objetivos:

Gestión de Usuarios

- **Objetivo:** Implementar funcionalidades seguras de autenticación y autorización, así como asegurar el almacenamiento seguro de contraseñas.
- **Cumplimiento:** Se han desarrollado sistemas robustos de autenticación y autorización utilizando JSON Web Tokens (JWT), junto con métodos seguros de almacenamiento de contraseñas, asegurando la protección de los datos de los usuarios.

Gestión de Puntuaciones

- **Objetivo:** Registrar y gestionar las puntuaciones de los usuarios de manera precisa y eficiente, así como su trazabilidad.
- **Cumplimiento:** Se han implementado mecanismos efectivos para calcular y almacenar las puntuaciones en la base de datos, asegurando la integridad y precisión de los datos.

Gestión de Archivos

- **Objetivo:** Facilitar la carga, almacenamiento y descarga de archivos, garantizando la integridad de los datos.
- **Cumplimiento:** Se han desarrollado procesos seguros y eficientes para la gestión de archivos, incluyendo validaciones estrictas para asegurar la integridad de los datos y facilitar su acceso por parte de los usuarios.

Despliegue de Dockers

- **Objetivo:** Configurar y administrar contenedores Docker de manera eficiente.
- **Cumplimiento:** Se han creado y administrado contenedores Docker utilizando Podman, aprovechando las ventajas de seguridad y gestión de recursos que ofrece esta herramienta, lo que ha permitido un despliegue eficiente y seguro de la aplicación.

10.1. Conclusiones Personales

El desarrollo de este Trabajo ha sido una experiencia muy enriquecedora y educativa. A lo largo de cada fase del proyecto he podido profundizar en diferentes tecnologías y prácticas de desarrollo web y ciberseguridad.

Cabe a destacar el despliegue de Dockers utilizando Podman, el cual ha sido particularmente interesante, ya que me ha permitido descubrir una nueva herramienta y su aplicación en un entorno de producción, junto a enseñarme algo que desconocía como es el hecho de poder ejecutar sistemas operativos con sus propias interfaces de usuarios dentro de contenedores.

No obstante, el proyecto no estuvo exento de desafíos y dificultades. Uno de los mayores obstáculos fue la falta de experiencia inicial en ciertas tecnologías, lo que implicó una curva de aprendizaje pronunciada y la necesidad de dedicar tiempo adicional a la investigación y el aprendizaje.

En conclusión, el proyecto ha cumplido con creces con los objetivos propuestos, resultando en una plataforma web robusta y de alta calidad. A pesar de los desafíos enfrentados, la experiencia y el conocimiento adquiridos serán invaluable para mi futura carrera profesional, capacitándome para abordar nuevos desafíos en el campo del desarrollo software y de la ciberseguridad. La superación de dichos desafíos ha mejorado en gran medida mi capacidad para afrontar problemas y adaptarme a nuevas situaciones, cosa que también será esencial en mi desarrollo profesional.

11. Trabajo Futuro

Durante el desarrollo del proyecto se identificaron diversas áreas y aspectos que podrían ser considerados para futuras mejoras o ampliaciones. A continuación, se describen brevemente algunos de estos aspectos:

Mejoras en la Interfaz de Usuario

- Explorar diseños más intuitivos y accesibles para mejorar la experiencia del usuario.
- Implementar técnicas de diseño responsivo para optimizar la visualización en diferentes dispositivos.

Funcionalidades Adicionales de Seguridad

- Integrar análisis continuos de vulnerabilidades y auditorías automáticas de seguridad.
- Desarrollar funcionalidades avanzadas de detección y respuesta ante amenazas cibernéticas.

Optimización del Rendimiento

- Realizar pruebas exhaustivas de carga y rendimiento para identificar y mejorar puntos críticos.
- Implementar técnicas de optimización de código y configuración de servidores para mejorar los tiempos de carga y respuesta.

Integración con Herramientas de Aprendizaje Automático

- Investigar la integración de técnicas de aprendizaje automático para mejorar la detección de anomalías y comportamientos sospechosos.
- Desarrollar modelos predictivos para anticipar posibles ataques y vulnerabilidades.

Ampliación de Contenidos y Ejercicios

- Incluir nuevos tipos de ejercicios y escenarios de ciberseguridad que aborden nuevos desafíos.
- Colaborar con expertos en ciberseguridad para desarrollar contenidos más avanzados y realistas.

Implementación de Monitorización y Métricas

- Desarrollar paneles de control y métricas para monitorizar el uso de la plataforma y el rendimiento de los usuarios.
- Integrar sistemas de feedback para recopilar comentarios y sugerencias de mejora de los usuarios.

Mejoras en la Gestión de Proyectos y Colaboración

- Implementar herramientas de gestión de proyectos más avanzadas para facilitar la colaboración y el seguimiento de tareas.
- Introducir metodologías ágiles para mejorar la flexibilidad y la capacidad de respuesta a cambios.

Estos aspectos representan oportunidades significativas para ampliar y mejorar el proyecto inicial, abordando áreas clave como la usabilidad, la seguridad, el rendimiento y la expansión a nuevos usuarios. Cada uno de estos elementos ayudaría a hacer de la plataforma una herramienta aún más eficaz y relevante en el campo de la ciberseguridad y la educación tecnológica.

12. Webgrafía

Bootstrap

- <https://getbootstrap.com/>

FontAwesome

- <https://fontawesome.com/>

Express

- <https://expressjs.com/>

JWT

- <https://jwt.io/>

MySQL

- <https://www.mysql.com/>

Node.js

- <https://nodejs.org/en>

Podman

- <https://podman.io/>

React

- <https://react.dev/>

Redux

- <https://redux.js.org/>

13. Anexos

A. Glosario

- Autenticación: Proceso de verificar la identidad de un usuario o sistema.
- Autorización: Proceso de dar permisos a un usuario para acceder a ciertos recursos o funcionalidades.
- Axios: Librería de JavaScript utilizada para hacer solicitudes HTTP desde el navegador y Node.js.
- bcryptjs: Biblioteca para hashing de contraseñas en JavaScript.
- CORS: Mecanismo que permite solicitudes de recursos restringidos en una página web desde un dominio distinto.
- date-fns: Biblioteca de JavaScript para el manejo y manipulación de fechas.
- dotenv: Módulo para cargar variables de entorno desde un archivo .env.
- Express.js: Framework web para Node.js que proporciona un conjunto robusto de características para aplicaciones web y móviles.
- Express FileUpload: Middleware de Node.js que facilita la carga de archivos.
- Express Validator: Middleware para validación de datos en aplicaciones Express.js.
- JWT: Estándar abierto para la creación de tokens de acceso.
- Multer: Middleware de Node.js para el manejo de multipart/form-data, utilizado principalmente para subir archivos.

- Nodemon: Herramienta que ayuda a desarrollar aplicaciones basadas en Node.js reiniciando automáticamente la aplicación cuando se detectan cambios en los archivos.
- Podman: Herramienta para gestionar contenedores de manera similar a Docker, pero sin necesidad de un daemon central.
- Redux: Biblioteca de JavaScript para el manejo del estado de las aplicaciones.
- RESTful API: Conjunto de servicios web que utilizan HTTP para realizar operaciones CRUD sobre los recursos.
- SweetAlert2: Biblioteca para mostrar mensajes de alerta personalizados en aplicaciones web.

B. Manual de Usuario de la Aplicación

Acceso a la Aplicación

- Navega a la URL de la aplicación proporcionada por la Universidad.
- Inicia sesión utilizando tus credenciales de usuario.

Navegación Principal

- Dashboard: Vista general de tus actividades y puntuaciones.
- Ejercicios: Acceso a los ejercicios de ciberseguridad disponibles.
- Perfil: Configuración de tu perfil y ajustes de cuenta.

Realización de Ejercicios

- Selecciona un ejercicio de la lista.
- Sigue las instrucciones proporcionadas para completar el ejercicio.
- Envía tus respuestas y recibe retroalimentación inmediata.

Gestión de Puntuaciones

- Consulta tu historial de puntuaciones en la sección de puntuaciones.
- Comparte y analiza tus resultados con tus compañeros.

C. Manual de Instalación de la Aplicación

Requisitos Previos

- Node.js y npm instalados en el sistema.
- MySQL configurado y en funcionamiento.

Instalación del FrontEnd

- Navega al directorio del FrontEnd.
- Ejecuta npm install para instalar las dependencias.
- Configura las variables de entorno en un archivo .env.
- Ejecuta npm start para iniciar el servidor de desarrollo.

Instalación del BackEnd

- Navega al directorio del BackEnd.
- Ejecuta npm install para instalar las dependencias.
- Configura las variables de entorno en un archivo .env.
- Ejecuta npm run dev para iniciar el servidor de desarrollo.

D. Manual de Montaje del Sistema

Preparación del Entorno

- Asegúrate de que los servidores están configurados adecuadamente.
- Instala Docker y Podman según las guías oficiales.

Despliegue de Contenedores

- Navega al directorio de los archivos de configuración de Docker.
- Construye los contenedores usando docker build.
- Despliega los contenedores utilizando podman run.

Configuración de la Base de Datos

- Accede al servidor MySQL.
- Crea las bases de datos y tablas necesarias ejecutando los scripts SQL proporcionados.

Verificación y Pruebas

- Realiza pruebas de funcionamiento accediendo a la URL de la aplicación.
- Verifica que todos los componentes se integran y funcionan correctamente.