



Universidad Politécnica  
de Madrid

**Escuela Técnica Superior de  
Ingenieros Informáticos**



Grado en Ingeniería Informática

Trabajo Fin de Grado

**Gestión de Planes Financieros**

Autor: Cristina Perulero Cabrera

Tutor(a): Tomás San Feliu Gilabert

Madrid, enero 2025

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

*Trabajo Fin de Grado*  
*Grado en Ingeniería Informática*  
*Título: GESTIÓN DE PLANES FINANCIEROS*  
Madrid 2025

*Autor:* Cristina Perulero Cabrera

*Tutor:*  
Tomás San Feliu Gilabert  
Lenguajes y Sistemas Informáticos e Ingeniería de Software  
ETSI Informáticos  
Universidad Politécnica de Madrid

# Resumen

Este informe expone el proceso de creación de una aplicación web destinada a la gestión de finanzas personales. El propósito central es proporcionar a los usuarios una herramienta efectiva que les permita gestionar de manera más eficiente sus ingresos, gastos y presupuestos, favoreciendo la toma de decisiones financieras informadas y promoviendo una mejor administración de sus recursos.

El desarrollo de esta aplicación web se va a llevar a cabo en varias fases. En primer lugar, se realiza un análisis de los requisitos fundamentales que un usuario va a requerir, y una vez realizado el análisis, se definen las funcionalidades clave.

En la segunda fase se va a diseñar una interfaz de usuario haciendo uso de HTML y CSS, enfocándose en proporcionar una interacción sencilla y eficiente para el usuario.

En la tercera fase, se va a implementar un servidor con Node.js que gestionará los endpoints y se integrará con una base de datos para almacenar de manera segura la información del usuario utilizando SQLite3.

Y, por último, una vez completadas todas las fases anteriores, se van a realizar pruebas exhaustivas para verificar que todas las funcionalidades funcionan a la perfección. Además, se llevarán a cabo pruebas con usuarios externos para asegurar que el sistema respondiera de manera adecuada a diferentes escenarios, evitando así errores que pudieran afectar la seguridad o experiencia del usuario.

Una vez completado y probado el sistema en su totalidad, se realizará el despliegue de la aplicación, optando por su ejecución en un entorno local, lo que garantiza su accesibilidad y correcto funcionamiento en equipos individuales. Esta opción asegura que los usuarios puedan instalar y utilizar la aplicación de manera sencilla y sin depender de plataformas externas.

El desarrollo de esta aplicación web puede generar un gran impacto en la gestión de finanzas personales, ya que se proporciona una plataforma en la que los usuarios pueden registrar y analizar sus movimientos. Su uso promueve una mayor comprensión de los hábitos de las transacciones, ayudando a identificar oportunidades de mejora. Además, la capacidad de generar reportes visuales y establecer metas de ahorro contribuirá a que los usuarios adopten prácticas financieras más responsables. En esencia, esta herramienta no solo simplifica el manejo de las finanzas personales, sino que también educa a los usuarios sobre la importancia de gestionar adecuadamente sus recursos, contribuyendo a un bienestar financiero duradero.

# Abstract

This report outlines the process of creating a web application for personal finance management. The central purpose is to provide users with an effective tool that will enable them to manage their income, expenses and budgets more efficiently, thus enabling them to make informed financial decisions and promoting better management of their resources.

The development of this web application will be carried out in several phases. Firstly, an analysis of the fundamental requirements that a user will require is carried out, and once the analysis has been carried out, the key functionalities are defined.

In the second phase, a user interface will be designed using HTML and CSS, focusing on providing a simple and efficient interaction for the user.

In the third phase, a Node.js server will be implemented to manage the endpoints and integrate with a database to securely store user information using SQLite3.

And finally, once all the previous phases have been completed, exhaustive tests will be carried out to verify that all the functionalities work perfectly. In addition, tests will be carried out with external users to ensure that the system responds appropriately to different scenarios, thus avoiding errors that could affect security or user experience.

Once the system has been completed and tested in its entirety, the application will be deployed, opting for its execution in a local environment, which guarantees its accessibility and correct functioning on individual computers. This option ensures that users can install and use the application easily and without depending on external platforms.

The development of this web application can have a great impact on personal finance management, as it provides a platform where users can record and analyse their movements. Its use promotes a greater understanding of transaction habits, helping to identify opportunities for improvement. In addition, the ability to generate visual reports and set savings goals will help users adopt more responsible financial practices. In essence, this tool not only simplifies the management of personal finances, but also educates users on the importance of properly managing their resources, contributing to lasting financial well-being.

# Tabla de Ilustraciones

|                                                                      |    |
|----------------------------------------------------------------------|----|
| Ilustración 1: Diagrama de Gantt.....                                | 3  |
| Ilustración 2: Uso de Express en el servidor .....                   | 5  |
| Ilustración 3: Diseño de interfaz con Bootstrap .....                | 6  |
| Ilustración 4: Visualización de Datos con Chart.js.....              | 6  |
| Ilustración 5: Gestión de contraseñas con Bcrypt.....                | 6  |
| Ilustración 6: Conexión a la base de datos con SQLite3.....          | 6  |
| Ilustración 7:Análisis de solicitudes con Body-parser .....          | 7  |
| Ilustración 8:Gestión de sesiones con Express-session.....           | 7  |
| Ilustración 9: Almacenamiento de sesiones con Connect-sqlite3.....   | 7  |
| Ilustración 10: Base de datos de usuarios .....                      | 11 |
| Ilustración 11: Base de datos de transacciones .....                 | 11 |
| Ilustración 12: Base de datos de sesiones .....                      | 12 |
| Ilustración 13: Base de datos de presupuestos.....                   | 12 |
| Ilustración 14: Endpoint de registro de usuarios .....               | 13 |
| Ilustración 15: Endpoint para el inicio de sesión .....              | 14 |
| Ilustración 16: Endpoint para agregar transacciones .....            | 14 |
| Ilustración 17: Endpoint para agregar presupuestos .....             | 14 |
| Ilustración 18: Endpoint para ver todas las transacciones.....       | 15 |
| Ilustración 19: Endpoint para obtener el saldo del usuario.....      | 15 |
| Ilustración 20: Endpoint para ver la tendencia de saldo .....        | 16 |
| Ilustración 21: Endpoint para gestionar la configuración .....       | 16 |
| Ilustración 22: Endpoint para cerrar sesión .....                    | 17 |
| Ilustración 23: Endpoint para eliminar perfil.....                   | 17 |
| Ilustración 24: Prueba de registro .....                             | 18 |
| Ilustración 25: Prueba de inicio de sesión .....                     | 19 |
| Ilustración 26: Prueba inicio sesión con contraseña incorrecta.....  | 19 |
| Ilustración 27: Prueba inicio sesión con usuario no registrado ..... | 20 |
| Ilustración 28: Prueba obtener usuario actual .....                  | 20 |
| Ilustración 29: Prueba agregar una transacción .....                 | 21 |
| Ilustración 30: Prueba ver transacciones .....                       | 21 |
| Ilustración 31: Prueba agregar presupuesto .....                     | 22 |
| Ilustración 32: Prueba ver los presupuestos.....                     | 22 |
| Ilustración 33: Prueba obtener configuración .....                   | 23 |
| Ilustración 34: Prueba cambiar configuración .....                   | 23 |
| Ilustración 35: Prueba volver a ver configuración usuario .....      | 24 |
| Ilustración 36: Página de registro.....                              | 25 |
| Ilustración 37: Página de inicio sesión .....                        | 26 |
| Ilustración 38: Página principal I .....                             | 26 |
| Ilustración 39: Página principal II .....                            | 27 |
| Ilustración 40: Página para agregar transacciones .....              | 27 |
| Ilustración 41: Historial de Transacciones I .....                   | 28 |
| Ilustración 42: Historial de transacciones II .....                  | 28 |
| Ilustración 43: Agregar presupuestos.....                            | 28 |
| Ilustración 44: Historial de presupuestos I.....                     | 29 |
| Ilustración 45: Historial de presupuestos II.....                    | 29 |
| Ilustración 46: Configuración.....                                   | 30 |

# Tabla de contenidos

|                                                    |            |
|----------------------------------------------------|------------|
| <b>Tabla de Ilustraciones .....</b>                | <b>iii</b> |
| <b>1 Introducción .....</b>                        | <b>1</b>   |
| 1.1 Motivación .....                               | 1          |
| 1.2 Objetivos .....                                | 1          |
| 1.2.1 Objetivo General .....                       | 1          |
| 1.2.2 Objetivos Técnicos .....                     | 1          |
| 1.2.3 Objetivos personales .....                   | 2          |
| 1.3 Planificación del proyecto .....               | 2          |
| <b>2 Historia del arte .....</b>                   | <b>4</b>   |
| 2.1 Introducción .....                             | 4          |
| 2.2 Plataforma del desarrollo .....                | 4          |
| 2.3 Lenguaje de programación .....                 | 4          |
| 2.4 Librerías y Frameworks externos .....          | 5          |
| <b>3 Desarrollo .....</b>                          | <b>8</b>   |
| 3.1 Introducción .....                             | 8          |
| 3.1.1 Objetivos del desarrollo .....               | 8          |
| 3.1.2 Descripción general del sistema .....        | 8          |
| 3.1.3 Metodología de desarrollo .....              | 8          |
| 3.2 Arquitectura del Sistema .....                 | 8          |
| 3.2.1 Elección de tecnologías y herramientas ..... | 8          |
| 3.2.2 Diseño general del sistema .....             | 9          |
| 3.3 Requisitos .....                               | 9          |
| 3.3.1 Requisitos funcionales .....                 | 9          |
| 3.3.2 Requisitos no funcionales .....              | 10         |
| 3.4 Diseño de la aplicación .....                  | 10         |
| 3.4.1 Diseño de la interfaz de usuario .....       | 10         |
| 3.4.2 Estructura de la base de datos .....         | 11         |
| 3.5 Desarrollo de Funcionalidades .....            | 13         |
| 3.5.1 Registro y autenticación de usuarios .....   | 13         |
| 3.5.2 Gestión de transacciones .....               | 14         |
| 3.5.3 Creación y gestión de presupuestos .....     | 14         |
| 3.5.4 Visualización de informes financieros .....  | 15         |
| 3.5.5 Obtener el saldo .....                       | 15         |
| 3.5.6 Obtención tendencia de saldo .....           | 15         |
| 3.5.7 Gestión de la configuración .....            | 16         |
| 3.5.8 Cierre de sesión y eliminación perfil .....  | 17         |
| 3.6 Seguridad .....                                | 17         |
| 3.7 Pruebas y Validación .....                     | 18         |

|          |                                                  |           |
|----------|--------------------------------------------------|-----------|
| 3.7.1    | Pruebas manuales .....                           | 18        |
| 3.7.2    | Pruebas con usuarios externos .....              | 24        |
| <b>4</b> | <b>Resultados y conclusiones .....</b>           | <b>25</b> |
| 4.1      | Resumen de los resultados obtenidos .....        | 25        |
| 4.2      | Conclusiones personales .....                    | 30        |
| <b>5</b> | <b>Análisis de Impacto.....</b>                  | <b>31</b> |
| 5.1      | Impacto Personal .....                           | 31        |
| 5.2      | Impacto Empresarial.....                         | 31        |
| 5.3      | Impacto Social .....                             | 31        |
| 5.4      | Impacto Económico.....                           | 31        |
| 5.5      | Impacto Medioambiental .....                     | 31        |
| 5.6      | Impacto Cultural.....                            | 31        |
| 5.7      | Decisiones tomadas considerando el impacto ..... | 31        |
| <b>6</b> | <b>Bibliografía .....</b>                        | <b>32</b> |
| <b>7</b> | <b>Informe de Originalidad .....</b>             | <b>33</b> |

# 1 Introducción

## 1.1 Motivación

Gestionar bien las finanzas personales es fundamental para poder tener una estabilidad económica. Muchas personas buscan herramientas que les ayuden a organizar sus ingresos y gastos, a definir presupuestos y a cumplir ciertas metas de ahorro. A pesar de la gran cantidad de aplicaciones disponibles en el mercado, muchas de ellas no cubren estas necesidades, como la falta de funcionalidades específicas, interfaces poco intuitivas o muy complicadas de comprender, la ausencia de garantías de seguridad y privacidad de los datos, o la obligación de suscribirse a un servicio de pago [1].

El propósito de este proyecto es proporcionar una aplicación web accesible y segura que asista a los usuarios en la gestión efectiva de sus finanzas personales. Esta herramienta facilitará el registro de ingresos y gastos, la elaboración de presupuestos, la visualización gráfica de las transacciones, las comparaciones entre el presupuesto y los gastos reales, y la generación de informes financieros. A diferencia de las aplicaciones existentes, esta herramienta se enfoca en la simplicidad y adaptabilidad a las necesidades financieras particulares de cada usuario.

Además de su impacto en la vida financiera de los usuarios, este proyecto contribuye al campo de la tecnología financiera al incorporar métodos visuales y claros para analizar datos financieros, junto con medidas que refuercen la seguridad y privacidad de la información. Estudios previos han demostrado que una gestión adecuada de las finanzas personales depende en gran medida de la capacidad de los usuarios para visualizar y analizar sus ingresos y gastos de forma estructurada [2]. Asimismo, enfoques como el método 50/30/20, que divide los ingresos en necesidades, deseos y ahorros, han demostrado ser eficaces en la promoción de una planificación financiera eficiente [3]. Estas ideas han inspirado y guiado la estructura funcional de esta aplicación.

## 1.2 Objetivos

### 1.2.1 Objetivo General

Este proyecto tiene como propósito principal crear y estructurar una aplicación web destinada a facilitar la administración de las finanzas personales. La herramienta incluirá funcionalidades clave como el registro y seguimiento de transacciones y presupuestos, además de la representación visual de estos datos mediante gráficos interactivos, lo que ayudará a los usuarios a obtener una perspectiva detallada de su situación económica. Adicionalmente, se incorporará la posibilidad de personalizar el perfil del usuario.

### 1.2.2 Objetivos Técnicos

Los objetivos técnicos se centran en ofrecer herramientas que simplifiquen la administración eficiente de las finanzas personales, ayudando a los usuarios a supervisar, organizar y evaluar sus ingresos y gastos de manera intuitiva. Entre los principales objetivos se encuentran:

- Control de transacciones: Implementar funcionalidades que permitan agregar transacciones, visualizarlas y categorizarlas de manera sencilla.
- Gestión de presupuestos: Proporcionar herramientas para crear presupuestos, tanto mensuales como anuales, ayudando así a los usuarios a organizar sus finanzas.
- Visualización gráfica: Incorporar gráficos interactivos, como diagrama de barras, circulares y lineales, que hagan la comprensión más sencilla.

- Historiables filtrables: Implementar filtros en los historiales, tanto de transacciones como de presupuestos, obteniendo más flexibilidad a la hora de analizar los datos en función de las necesidades de cada usuario.
- Personalización y seguridad: Desarrollar opciones para que los usuarios gestionen sus perfiles, como es el nombre, correo y contraseña, de forma segura, incluyendo cifrado en las contraseñas, y funcionalidades como cerrar sesión o eliminar el perfil permanentemente.
- Interfaz accesible: Diseñar una interfaz de usuario intuitiva, visual y llamativa de forma que puedan hacer uso de esta herramienta, aún sin tener conocimientos técnicos avanzados.

### **1.2.3 Objetivos personales**

En cuanto a objetivos personales, quiero destacar los siguientes:

- Aplicar conocimientos requeridos: poner en práctica los conocimientos aprendidos durante el grado, especialmente en el ámbito del desarrollo web.
- Desarrollar experiencia: familiarizarme y aprender nuevas herramientas y tecnologías actuales como Node.js, HTML, SCC y SQLite3.
- Gestionar un proyecto completo: Adquirir experiencia en el ciclo de vida de un proyecto, desde el análisis y el diseño hasta la implementación y pruebas.
- Resolución de problemas: Enfrentarme y superar retos técnicos durante la realización del desarrollo de la aplicación
- Generar un proyecto que aparte de cumplir con mis objetivos, pueda servir en el día a día, para uso personal.

## **1.3 Planificación del proyecto**

He seguido este diagrama de Gantt para planificar el desarrollo de este proyecto, organizando las tareas por semanas desde el inicio del trabajo. A lo largo de la implementación de la aplicación, el diagrama ha sido modificado para reflejar los cambios en el cronograma, ya que he tenido más dificultades de las esperadas en el desarrollo de la aplicación.

En particular, las áreas que presentaron mayores desafíos fueron las visualizaciones de las transacciones y de los presupuestos. Estas funcionalidades requerían un enfoque más detallado y un mayor tiempo de desarrollo del que inicialmente había previsto.

A pesar de estos contratiempos, la planificación inicial me ha permitido tener una idea más clara sobre las tareas a realizar. La flexibilidad para ajustar el cronograma ha sido crucial para adaptarme a los imprevistos y asegurar que cada funcionalidad se desarrollara de manera efectiva.

A continuación, se presenta el diagrama de Gantt actualizado, que refleja el progreso y las modificaciones realizadas durante el desarrollo del proyecto.

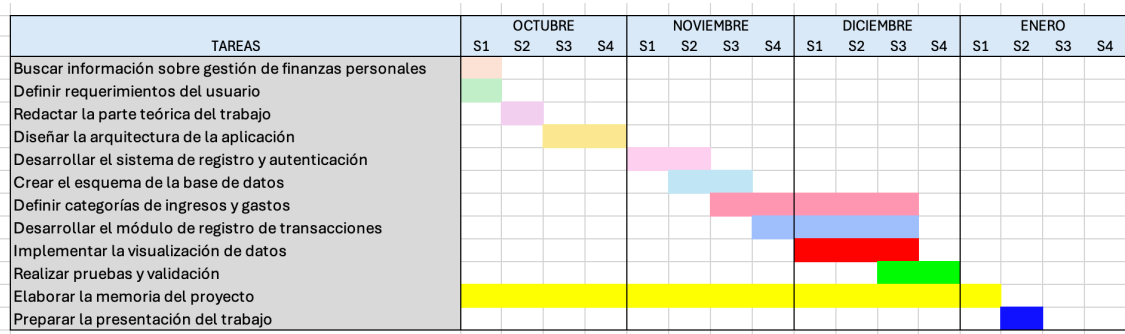


*Ilustración 1: Diagrama de Gantt*

## **2 Historia del arte**

### **2.1 Introducción**

En esta sección se detallarán las razones que llevaron a la elección de la plataforma de desarrollo, así como los lenguajes de programación y las bibliotecas clave empleadas durante el proceso de creación de la aplicación. Cada herramienta ha sido seleccionada para asegurarnos de que la experiencia del usuario sea lo más segura, eficiente posible y se adapte a las necesidades de organización.

### **2.2 Plataforma del desarrollo**

Se ha optado por desarrollar la aplicación como una aplicación web multiplataforma para maximizar su accesibilidad y compatibilidad. Este enfoque permite a los usuarios acceder a la aplicación desde distintos sistemas operativos, como Windows, macOS y Linux, utilizando simplemente un navegador web. Al eliminar la necesidad de instalaciones específicas en cada dispositivo, se facilita el acceso a una mayor cantidad de usuarios, lo que es fundamental para esta herramienta. Además, este enfoque garantiza escalabilidad y flexibilidad, permitiendo que la aplicación se adapte a las necesidades cambiantes de los usuarios a medida que evolucionan sus hábitos financieros y requerimientos de gestión [4].

Durante el proceso de selección del entorno de desarrollo, se consideraron varios editores de texto, incluyendo Atom y Sublime Text. Sin embargo, se ha decidido hacer uso de Visual Studio Code debido a su amplia gama de extensiones que optimizan el flujo de trabajo. Este editor de código fuente no solo proporciona herramientas de edición de código avanzadas, sino que también permite la integración de sistemas de control de versiones, depuración y pruebas.

Visual Studio Code es compatible con múltiples lenguajes y tecnologías web, como HTML, CSS y JavaScript, lo que ha facilitado la creación de un código consistente y organizado [5]. Esta capacidad de trabajar con diferentes tecnologías en un entorno unificado contribuye a una mayor eficiencia en el desarrollo y mantenimiento de la aplicación. La elección de Visual Studio Code se basa en su versatilidad y en la comunidad activa que respalda su desarrollo, lo que garantiza que siempre haya recursos y soporte disponibles para resolver cualquier desafío que surja durante el proceso de desarrollo.

### **2.3 Lenguaje de programación**

Para llevar a cabo el desarrollo de esta aplicación web destinada a la gestión de finanzas personales, se han elegido tecnologías específicas que aseguran un funcionamiento eficaz y ofrecen una experiencia de usuario de alta calidad:

- **JavaScript y Node.js:** JavaScript es el lenguaje estándar para el desarrollo de aplicaciones web [6], y en esta aplicación se ha utilizado junto con Node.js para gestionar el backend. Node.js es una plataforma que permite ejecutar JavaScript en el servidor, fuera del entorno del navegador, facilitando la creación de aplicaciones web escalables y eficientes. Esto lo convierte en una herramienta ideal para desarrollar aplicaciones web escalables y con un rendimiento elevado. Este enfoque

permite manejar múltiples solicitudes simultáneas, crucial para una aplicación que debe responder rápidamente a las interacciones de los usuarios, como en la gestión de finanzas personales. Node.js también proporciona un ecosistema robusto con muchas librerías y herramientas que agilizan el desarrollo.

- **HTML y CSS:** HTML y CSS se emplearon para diseñar la interfaz de usuario, proporcionando la estructura básica y los estilos visuales necesarios para garantizar una experiencia atractiva y funcional. HTML se utiliza para definir los elementos de la página, como formularios, tablas y botones interactivos, que facilitan la interacción del usuario con la plataforma [7]. En cambio, CSS, se ha utilizado para configurar el diseño y garantizar una experiencia visual atractiva y fácil de usar, lo que ayuda a que los usuarios puedan navegar sin complicaciones [8].
- **SQLite** fue elegido sistema de gestión de bases de datos debido a su alta velocidad, ligereza y facilidad de integración, lo que lo convierte en una opción ideal para aplicaciones que no requieren un servidor de bases de datos completo. Este sistema es especialmente adecuado para aplicaciones web que manejan datos localmente y no necesitan un servidor de base de datos dedicado [9]. Esta elección permite que la aplicación maneje los datos de manera eficiente, asegurando que la información del usuario esté siempre disponible y protegida, sin la necesidad de una infraestructura compleja.

## 2.4 Librerías y Frameworks externos

Para el desarrollo de la aplicación de gestión de finanzas personales, se han utilizado diversas librerías y frameworks externos que complementan las tecnologías principales y mejoran la funcionalidad y la experiencia del usuario.

- **Express:** Este framework de Node.js se ha utilizado para crear el servidor y gestionar las rutas de la aplicación. Simplifica el manejo de solicitudes HTTP y permite la creación de APIs RESTful de manera eficiente [10].

```
const express = require('express');
const app = express();
const port = 3000;
// Iniciar el servidor
app.listen(port, () => {
  console.log(`Servidor corriendo en http://localhost:${port}`);
});
```

*Ilustración 2: Uso de Express en el servidor*

- **Bootstrap:** Se ha integrado Bootstrap como framework CSS para el diseño de la interfaz de usuario. Bootstrap proporciona una serie de componentes predefinidos y un sistema de grid responsivo que facilita la creación de una interfaz atractiva y accesible en diferentes dispositivos. Esto asegura que la aplicación tenga un diseño moderno y profesional, mejorando la experiencia del usuario [11].

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Presupuestos</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
```

---

*Ilustración 3: Diseño de interfaz con Bootstrap*

- Chart.js: Esta librería se ha utilizado para la visualización de datos en forma de gráficos. Permite crear gráficos interactivos y personalizables. En la aplicación, se utiliza para representar visualmente los presupuestos y los gastos, facilitando a los usuarios el análisis de su situación financiera [12].

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

*Ilustración 4: Visualización de Datos con Chart.js*

- Bcrypt: Esta librería se ha implementado para la gestión de contraseñas y la seguridad de las cuentas de usuario. Bcrypt se utiliza para generar hashes de contraseñas, lo que proporciona una capa adicional de seguridad al almacenar información sensible en la base de datos. Esta funcionalidad resulta esencial para garantizar la protección y confidencialidad de los datos financieros de los usuarios [13].

```
const bcrypt = require('bcrypt');
// Endpoint para registrar usuario
app.post('/registro', async (req, res) => {
  const { nombre, email, password } = req.body;
  try {
    const hashedPassword = await bcrypt.hash(password, 10);
  }
}
```

*Ilustración 5: Gestión de contraseñas con Bcrypt*

- SQLite3: Aunque no es una librería, se ha empleado SQLite como sistema de gestión de bases de datos para almacenar y manejar la información financiera de los usuarios.

---

```
const sqlite3 = require('sqlite3').verbose();

// Conectar a la base de datos SQLite
const db = new sqlite3.Database('./app.db', (err) => {
  if (err) {
    console.error('Error al conectar con la base de datos:', err.message);
  } else {
    console.log('Conectado a la base de datos SQLite');
  }
});
```

*Ilustración 6: Conexión a la base de datos con SQLite3*

- Body-parser: Esta librería se ha utilizado para analizar el cuerpo de las solicitudes HTTP, permitiendo que el servidor interprete los datos enviados desde el frontend. Esto es esencial para manejar formularios y datos JSON de manera efectiva [14].

```
const bodyParser = require('body-parser');
app.use(bodyParser.json());
```

*Ilustración 7: Análisis de solicitudes con Body-parser*

- Express-session: Esta librería se utiliza para gestionar sesiones de usuario en la aplicación. Permite almacenar información sobre el estado de la sesión del usuario, lo que es fundamental para mantener la autenticación y la personalización de la experiencia del usuario [12].

```
const session = require('express-session');

app.use(session({
  secret: 'tu_secreto_aqui',
  resave: false,
  saveUninitialized: true,
  cookie: { secure: false }
}));
```

*Ilustración 8: Gestión de sesiones con Express-session*

- Connect-sqlite3: Esta es una extensión para express-session que permite almacenar sesiones en una base de datos SQLite. Esto es útil para mantener la persistencia de las sesiones de usuario, incluso después de que el servidor se reinicie [15].

```
const sqlite3 = require('sqlite3').verbose();
const SQLiteStore = require('connect-sqlite3')(session);
```

*Ilustración 9: Almacenamiento de sesiones con Connect-sqlite3*

## 3 Desarrollo

### 3.1 Introducción

En esta sección se detalla el proceso de desarrollo de una aplicación destinada a la gestión de finanzas personales. Su propósito principal es proporcionar a los usuarios una herramienta fácil de usar y eficiente para el monitoreo y administración de sus recursos financieros.

#### 3.1.1 Objetivos del desarrollo

El desarrollo de la aplicación se centra en permitir a los usuarios gestionar sus finanzas de manera más eficiente, incluyendo:

- Registro de transacciones: Permitir agregar, editar y eliminar transacciones fácilmente.
- Gestión de presupuestos: Ofrecer herramientas para crear y seguir presupuestos mensuales y anuales.
- Visualización de datos: Incluir gráficos interactivos para facilitar la comprensión de la situación financiera.
- Protección de datos: Implementar medidas de seguridad para proteger la información sensible.

#### 3.1.2 Descripción general del sistema

La aplicación de gestión de finanzas personales es accesible desde cualquier dispositivo con un navegador web. Permite a los usuarios registrar ingresos y gastos, crear presupuestos y visualizar su situación financiera mediante gráficos interactivos, con un enfoque en la simplicidad y facilidad de uso.

#### 3.1.3 Metodología de desarrollo

Se han utilizado metodologías ágiles para el desarrollo, permitiendo una adaptación continua a los cambios y necesidades del proyecto. Las iteraciones regulares han permitido evaluar el progreso y ajustar las funcionalidades según los comentarios de los usuarios y las pruebas realizadas.

## 3.2 Arquitectura del Sistema

Esta sección describe la arquitectura general del sistema, incluyendo las tecnologías y herramientas elegidas para garantizar una aplicación escalable, mantenible y eficiente.

### 3.2.1 Elección de tecnologías y herramientas

Las tecnologías seleccionadas son clave para cumplir los objetivos del proyecto y asegurar un rendimiento óptimo:

- Node.js: Elegido para el backend por su capacidad de manejar múltiples conexiones simultáneas eficientemente [16].
- Express: Framework de Node.js utilizado para crear el servidor y gestionar las rutas de la aplicación, facilitando la creación de APIs RESTful.
- SQLite: Sistema de gestión de bases de datos ligero y fácil de implementar, ideal para almacenar información localmente.
- HTML, CSS y Bootstrap: Utilizados para el frontend, asegurando una interfaz de usuario atractiva y responsiva.
- Chart.js: Librería para la visualización de datos financieros a través de gráficos interactivos.
- Postman: Herramienta utilizada para probar y verificar la API, mejorando la calidad y fiabilidad antes de la integración con el frontend [17].

Estas tecnologías — Node.js, Express, SQLite, HTML, CSS, Bootstrap, Chart.js y Postman — ofrecen una estructura robusta y flexible para el desarrollo de la aplicación de gestión financiera personal. Juntas, permiten un rendimiento excelente y una interfaz atractiva, facilitando la integración de funcionalidades avanzadas y asegurando un manejo eficiente y seguro de los datos financieros.

### **3.2.2 Diseño general del sistema**

El sistema sigue un enfoque de arquitectura cliente-servidor, donde la interacción entre el frontend y el backend se realiza mediante APIs RESTful. A continuación, se describen los componentes principales de esta arquitectura:

- **Frontend:** La interfaz de usuario se desarrolla utilizando HTML, CSS y Bootstrap. Los usuarios interactúan con la aplicación a través de un navegador web, donde pueden registrar transacciones, crear presupuestos y visualizar informes financieros. Las solicitudes del usuario se envían al backend a través de llamadas a la API.
- **Backend:** El servidor, desarrollado con Node.js y Express, recibe las solicitudes del frontend y las procesa. El servidor es responsable de manejar la lógica del negocio, comunicarse con la BBDD y enviar las respuestas correspondientes al cliente.
- **Base de datos:** SQLite se utiliza para guardar la información relacionada con los usuarios, las transacciones y los presupuestos. El backend se encarga de interactuar con la base de datos para llevar a cabo operaciones de lectura y escritura, garantizando que los datos estén siempre actualizados y accesibles.

Esta arquitectura permite que la aplicación sea escalable y fácil de mantener, ya que cada componente se puede actualizar o modificar de manera independiente sin afectar al resto del sistema.

## **3.3 Requisitos**

Esta sección detalla los requisitos necesarios para el funcionamiento de la aplicación de gestión de finanzas personales.

### **3.3.1 Requisitos funcionales**

Los requisitos funcionales definen las características clave que la aplicación debe proporcionar a los usuarios. Algunos de estos incluyen:

- **Registro y autenticación de usuarios:** La aplicación debe permitir a los usuarios crear una cuenta y acceder a la plataforma mediante un proceso de inicio de sesión seguro. Esto incluye la validación de credenciales y la gestión de sesiones.
- **Gestión de transacciones:** Los usuarios deben poder agregar, editar y eliminar transacciones de ingresos y gastos. Esto incluye la capacidad de categorizar las transacciones para facilitar su seguimiento y análisis.
- **Creación y gestión de presupuestos:** La aplicación debe permitir a los usuarios crear presupuestos mensuales y anuales, así como realizar un seguimiento de su cumplimiento. Los usuarios deben poder ajustar sus presupuestos según sea necesario.
- **Visualización de informes financieros:** La aplicación debe ofrecer gráficos interactivos que representen visualmente la situación financiera del usuario. Esto incluye gráficos de barras, circulares y lineales que muestren ingresos, gastos y presupuestos.

### **3.3.2 Requisitos no funcionales**

Los requisitos no funcionales especifican las características relacionadas con el rendimiento y la calidad que la aplicación debe cumplir. Estos incluyen:

- **Accesibilidad:** La aplicación se debe poder acceder desde diferentes dispositivos (ordenadores, tablets y teléfonos móviles) y navegadores web (Chrome, Firefox, Safari, etc.), asegurando una experiencia de usuario consistente.
- **Seguridad de los datos:** La aplicación debe garantizar la protección de la información del usuario mediante el uso de cifrado para las contraseñas y la implementación de medidas de seguridad adecuadas para salvaguardar la información sensible.
- **Usabilidad:** La interfaz de usuario debe ser clara e intuitiva, permitiendo que los usuarios, sin importar su nivel de conocimiento técnico, puedan interactuar con la aplicación sin dificultades.
- **Rendimiento:** La aplicación debe ser capaz de procesar varias solicitudes al mismo tiempo sin que esto degrade la experiencia del usuario. Esto implica tiempos de respuesta ágiles y un rendimiento eficiente en la gestión de la información.

## **3.4 Diseño de la aplicación**

En esta parte se describirá el diseño de la aplicación, incluyendo tanto su interfaz de usuario como la organización de los datos. Un diseño bien estructurado es clave para asegurar una experiencia de usuario eficiente y sin interrupciones, así como para mantener la integridad y accesibilidad de la información.

### **3.4.1 Diseño de la interfaz de usuario**

La interfaz de usuario fue diseñada empleando Bootstrap, un framework CSS que facilita la creación de aplicaciones web responsivas y estéticamente atractivas. A continuación, se presentan los aspectos clave del diseño de la interfaz:

- **Usabilidad:** Se dio prioridad a la creación de una interfaz fácil de usar e intuitiva, permitiendo que los usuarios naveguen y realicen sus tareas de manera sencilla. Esto incluye menús y botones organizados lógicamente, accesibles sin importar el nivel de conocimiento técnico del usuario.
- **Navegación clara:** La aplicación cuenta con un menú de navegación que facilita el acceso a las diferentes secciones. Este menú está siempre visible para que los usuarios puedan moverse fácilmente entre secciones sin perder el contexto.
- **Estética visual:** Se utilizó una paleta de colores coherente y atractiva, junto con tipografías legibles, para mejorar la experiencia visual del usuario. Los colores no solo mejoran la estética, sino que también ayudan a distinguir visualmente las diferentes secciones y elementos interactivos.
- **Interactividad:** La aplicación incluye elementos interactivos, como formularios, botones, y gráficos que responden a las acciones del usuario. Los formularios permiten a los usuarios agregar y editar transacciones de manera fluida, mientras que los gráficos interactivos proporcionan una representación visual clara y dinámica de los datos financieros.
- **Responsividad:** Utilizando Bootstrap, la interfaz de usuario se ajusta automáticamente a diversas resoluciones de pantalla, garantizando una experiencia de usuario óptima en dispositivos móviles, tablets y ordenadores.

- Retroalimentación visual: Los usuarios reciben retroalimentación inmediata al interactuar con la aplicación. Esto incluye de confirmación al agregar o editar transacciones, y alertas en caso de errores. Esta retroalimentación ayuda a los usuarios a entender mejor el estado de sus acciones y a corregir errores fácilmente.
- Consistencia: Se mantiene una consistencia en el diseño a lo largo de toda la aplicación, utilizando patrones de diseño coherentes para botones, formularios, y otros elementos interactivos. Esta consistencia en el diseño permite a los usuarios adaptarse rápidamente a la interfaz y navegar de manera más fluida.
- Modularidad: La interfaz se diseñó de manera modular, permitiendo actualizar o modificar secciones individuales sin afectar el resto de la aplicación. Esto facilita la implementación de nuevas funcionalidades y la mejora continua de la interfaz.

### 3.4.2 Estructura de la base de datos

La estructura de la base de datos SQLite se diseñó para almacenar de manera eficiente los datos relacionados con los usuarios, las transacciones y los presupuestos. A continuación, se detallan las tablas clave y sus respectivas relaciones:

- Tabla de Usuarios: Esta tabla contiene la información básica de los usuarios:

```
CREATE TABLE usuarios (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nombre TEXT NOT NULL,
    correo TEXT UNIQUE NOT NULL,
    contraseña TEXT NOT NULL);
```

*Ilustración 10: Base de datos de usuarios*

- Id: identificador único del usuario
  - Nombre: Nombre del usuario
  - Correo: Dirección de correo electrónico del usuario
  - Contraseña: Contraseña cifrada del usuario
- Tabla de Transacciones: Esta tabla almacena la información de las transacciones:

```
CREATE TABLE transacciones (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    usuario_id INTEGER,
    tipo TEXT NOT NULL,
    cantidad REAL NOT NULL,
    categoria TEXT NOT NULL,
    fecha TEXT NOT NULL,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id)
);
```

*Ilustración 11: Base de datos de transacciones*

- Id: identificador único de cada transacción

- Usuario\_id: almacena el identificador del usuario
  - Tipo: indica si es gasto o ingreso
  - Cantidad: cantidad monetaria de la transacción
  - Categoría: indica la categoría a la que pertenecería dicha transacción
  - Fecha: fecha en la que se realiza la transacción
- Tabla de sesiones: Esta tabla almacena las sesiones de cada usuario:

```
CREATE TABLE sesiones (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    usuario_id INTEGER,
    session_id TEXT,
    fecha_inicio DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id)
);
```

*Ilustración 12: Base de datos de sesiones*

- Id: identificador de cada sesión
  - Usuario\_id: almacena el identificador del usuario
  - Sesión\_id: almacena el identificador de la sesión
  - Fecha\_inicio: Se pone por defecto la fecha y hora del momento en el que se inicia una sesión
- Tabla de presupuestos: En esta tabla se almacenan los gastos del usuario

```
CREATE TABLE presupuestos (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    usuario_id INTEGER,
    categoria TEXT NOT NULL,
    cantidad REAL NOT NULL,
    frecuencia TEXT NOT NULL,
    duracion INTEGER NOT NULL,
    fecha_inicio DATE,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id)
);
```

*Ilustración 13: Base de datos de presupuestos*

- Id: identificador único de cada gasto
  - Usuario\_id: almacena el identificador del usuario
  - Categoría: indica la categoría a la que pertenece el presupuesto
  - Frecuencia: almacena si es mensual o anual
  - Duración: almacena la duración del presupuesto, en años o meses
  - Fecha\_inicio: almacena el día inicial del presupuesto, que se tomará de referencia para próximos meses o años.

Las relaciones entre estas tablas garantizan la integridad de los datos, permitiendo que cada usuario tenga acceso a sus propias transacciones y presupuestos sin interferir con la información de otros usuarios.

## 3.5 Desarrollo de Funcionalidades

En esta sección se describirán las funcionalidades implementadas en la aplicación de gestión de finanzas personales, junto con ejemplos de código y explicaciones sobre su funcionamiento.

### 3.5.1 Registro y autenticación de usuarios

La funcionalidad de registro posibilita que los nuevos usuarios creen una cuenta en la aplicación. Este proceso incluye la validación de datos y el cifrado de contraseñas utilizando bcrypt para garantizar la seguridad de la información. Una vez registrado, el usuario se almacena en la base de datos y se inicia una sesión utilizando express-session.

```
app.post('/registro', async (req, res) => {
  const { nombre, email, password } = req.body;
  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    db.run(
      `INSERT INTO usuarios (nombre, correo, contraseña) VALUES (?, ?, ?)`,
      [nombre, email, hashedPassword],
      function (err) {
        if (err)
          return res.status(400).json({ error: "Error al registrar usuario o el usuario ya existe" });

        const userId = this.lastID;
        req.session.user = { id: userId, nombre, email };
        const sessionId = req.sessionID;
        db.run(`INSERT INTO sesiones (usuario_id, session_id) VALUES (?, ?)`, [userId, sessionId], (err) => {
          if (err)
            return res.status(500).json({ error: "Error al registrar sesión" });

          res.status(201).json({ message: "Usuario registrado con éxito", userId });
        });
      }
    );
  } catch (error) {
    res.status(500).json({ error: "Error en el servidor" });
  }
});
```

*Ilustración 14: Endpoint de registro de usuarios*

La función de inicio de sesión permite a los usuarios registrados acceder de manera segura a la aplicación. Para ello, se valida la contraseña ingresada al compararla con la almacenada en la base de datos. Si ninguna credencial es errónea, el usuario inicia sesión, garantizando su autenticación y acceso al sistema.

```

app.post('/iniciosesion', (req, res) => {
  const { email, password } = req.body;

  db.get(`SELECT * FROM usuarios WHERE correo = ?`, [email], async (err, user) => {
    if (err)
      return res.status(500).json({ error: "Error en el servidor" });

    if (!user)
      return res.status(404).json({ error: "Usuario no encontrado" });

    try {
      const isMatch = await bcrypt.compare(password, user.contraseña);
      if (!isMatch)
        return res.status(401).json({ error: "Contraseña incorrecta" });
      req.session.user = { id: user.id, nombre: user.nombre, email: user.correo };
      const sessionId = req.sessionID;
      await db.run(`INSERT INTO sesiones (usuario_id, session_id) VALUES (?, ?)`, [user.id, sessionId]);

      res.status(200).json({ message: "Inicio de sesión exitoso", success: true });
    } catch (error) {
      return res.status(500).json({ error: "Error en el servidor" });
    }
  });
});

```

*Ilustración 15: Endpoint para el inicio de sesión*

### 3.5.2 Gestión de transacciones

Los usuarios pueden agregar, editar y eliminar transacciones, lo que les permite llevar un seguimiento detallado de sus ingresos y gastos. Esta funcionalidad es esencial para una gestión financiera efectiva.

```

app.post('/transaccion', (req, res) => {
  const { tipo, cantidad, categoria, fecha } = req.body;
  const usuarioId = req.session.user.id;
  if (tipo !== 'ingreso' && tipo !== 'gasto')
    return res.status(400).json({ error: "Tipo de transacción inválido" });
  const sql = `INSERT INTO transacciones (usuario_id, tipo, cantidad, categoria, fecha) VALUES (?, ?, ?, ?, ?)`;
  db.run(sql, [usuarioId, tipo, cantidad, categoria, fecha], function(err) {
    if (err)
      return res.status(500).json({ error: "Error al agregar la transacción" });

    res.json({ success: true, message: "Transacción agregada correctamente" });
  });
});

```

*Ilustración 16: Endpoint para agregar transacciones*

### 3.5.3 Creación y gestión de presupuestos

Los usuarios pueden crear y gestionar presupuestos, lo que les permite planificar sus gastos y ahorros. Esta funcionalidad ayuda a los usuarios a establecer límites en sus gastos y a seguir su progreso a lo largo del tiempo.

```

app.post('/presupuesto', async (req, res) => {
  const { categoria, cantidad, frecuencia, duracion, diaEspecifico, fechaInicio } = req.body;
  const usuarioId = req.session.user.id;
  if (!usuarioId)
    return res.status(401).json({ message: 'No estás autenticado' });
  try {
    console.log('Datos recibidos:', { usuarioId, categoria, cantidad, frecuencia, duracion, diaEspecifico, fechaInicio });
    const query = `INSERT INTO presupuestos (usuario_id, categoria, cantidad, frecuencia, duracion, dia_especifico, fecha_inicio) VALUES (?, ?, ?, ?, ?, ?, ?)`;
    await db.run(query, [usuarioId, categoria, cantidad, frecuencia, duracion, diaEspecifico, fechaInicio], function (err) {
      if (err) {
        return res.status(500).json({ message: 'Error al agregar presupuesto' });
      }
      res.status(201).json({ message: 'Presupuesto agregado con éxito' });
    });
  } catch (error) {
    res.status(500).json({ message: 'Error al agregar presupuesto' });
  }
});

```

*Ilustración 17: Endpoint para agregar presupuestos*

### 3.5.4 Visualización de informes financieros

Los usuarios pueden acceder a informes financieros que les permiten visualizar sus ingresos, gastos y presupuestos de manera clara y concisa. Esta funcionalidad es crucial para que los usuarios comprendan su situación financiera y tomen decisiones informadas.

```
app.get('/transacciones', (req, res) => {
  if (!req.session.user)
    return res.status(401).json({ error: "No autenticado" });
  const usuarioId = req.session.user.id;

  const sql = `
    SELECT tipo, cantidad, categoria, fecha
    FROM transacciones
    WHERE usuario_id = ?
    ORDER BY fecha DESC
  `;
  db.all(sql, [usuarioId], (err, rows) => {
    if (err)
      return res.status(500).json({ error: "Error al obtener las transacciones" });
    res.json(rows);
  });
});
```

*Ilustración 18: Endpoint para ver todas las transacciones*

### 3.5.5 Obtener el saldo

Los usuarios pueden consultar su saldo actual, que se calcula a partir de las transacciones registradas en el mes actual. Esta funcionalidad permite a los usuarios tener una visión clara de su situación financiera en tiempo real.

```
app.get('/saldo', (req, res) => {
  if (!req.session.user)
    return res.status(401).json({ error: "No autenticado" });
  const usuarioId = req.session.user.id;
  const sql = `
    SELECT
      SUM(CASE WHEN tipo = 'ingreso' THEN cantidad ELSE 0 END) AS total_ingresos,
      SUM(CASE WHEN tipo = 'gasto' THEN cantidad ELSE 0 END) AS total_gastos
    FROM transacciones
    WHERE usuario_id = ?
    AND strftime('%Y-%m', fecha) = strftime('%Y-%m', 'now')
  `;

  db.get(sql, [usuarioId], (err, row) => {
    if (err) {
      return res.status(500).json({ error: "Error al calcular el saldo" });
    }
    const saldo = (row.total_ingresos || 0) - (row.total_gastos || 0);
    res.json({ saldo });
  });
});
```

*Ilustración 19: Endpoint para obtener el saldo del usuario*

### 3.5.6 Obtención tendencia de saldo

Los usuarios pueden visualizar la tendencia de su saldo a lo largo del tiempo, lo que les permite identificar patrones en sus ingresos y gastos. Esta funcionalidad es útil para la planificación financiera a largo plazo.

```

app.get('/tendencia-saldo', (req, res) => {
  if (!req.session.user)
    return res.status(401).json({ error: "No autenticado" });
  const usuarioId = req.session.user.id;

  const sql = `
    SELECT
      fecha,
      SUM(CASE WHEN tipo = 'ingreso' THEN cantidad ELSE 0 END) AS total_ingresos,
      SUM(CASE WHEN tipo = 'gasto' THEN cantidad ELSE 0 END) AS total_gastos
    FROM transacciones
    WHERE usuario_id = ?
    GROUP BY fecha
    ORDER BY fecha
  `;

  db.all(sql, [usuarioId], (err, rows) => {
    if (err) {
      return res.status(500).json({ error: "Error al obtener la tendencia de saldo" });
    }
    const fechas = rows.map(row => row.fecha);
    const ingresos = rows.map(row => row.total_ingresos);
    const gastos = rows.map(row => row.total_gastos);
    const saldoAcumulado = [];
    let saldo = 0;
    for (let i = 0; i < fechas.length; i++) {
      saldo += ingresos[i] - gastos[i];
      saldoAcumulado.push(saldo);
    }
    res.json({ fechas, saldoAcumulado });
  });
});

```

*Ilustración 20: Endpoint para ver la tendencia de saldo*

### 3.5.7 Gestión de la configuración

Los usuarios pueden gestionar su configuración personal, incluyendo la actualización de su información de perfil y la contraseña. Esta funcionalidad permite a los usuarios mantener su información actualizada y segura.

```

app.put('/configuracion', (req, res) => {
  const { nombre, correo } = req.body;
  const usuarioId = req.session.user.id;
  db.get(`SELECT nombre, correo FROM usuarios WHERE id = ?`, [usuarioId], (err, row) => {
    if (err)
      return res.status(500).json({ error: "Error al obtener los datos del usuario" });
    else {
      db.run(
        `UPDATE usuarios SET nombre = ?, correo = ? WHERE id = ?`,
        [nombre, correo, usuarioId],
        function (err) {
          if (err)
            return res.status(500).json({ error: "Error al actualizar los datos del usuario" });
          req.session.user.nombre = nombre;
          req.session.user.email = correo;
          res.json({ message: "Datos actualizados correctamente" });
        }
      );
    }
  });
});

```

*Ilustración 21: Endpoint para gestionar la configuración*

### 3.5.8 Cierre de sesión y eliminación perfil

La funcionalidad de cierre de sesión permite a los usuarios finalizar su sesión de manera segura. Esto es importante para proteger la información personal y financiera del usuario, especialmente en dispositivos compartidos o públicos. Al cerrar sesión, se destruye la sesión activa.

```
app.post('/cerrar-sesion', (req, res) => {
  const sessionId = req.sessionID;
  req.session.destroy(err => {
    if (err)
      return res.status(500).json({ error: "Error al cerrar sesión" });

    db.run(`DELETE FROM sesiones WHERE session_id = ?`, [sessionId], (err) => {
      if (err)
        console.error("Error al eliminar sesión:", err);
    });
    res.json({ message: "Sesión cerrada con éxito" });
  });
});
```

*Ilustración 22: Endpoint para cerrar sesión*

La funcionalidad de eliminación de perfil permite a los usuarios borrar su cuenta de la aplicación de forma permanente. Esta acción elimina todos los datos asociados al usuario, incluyendo su información personal, transacciones y presupuestos. Es importante que los usuarios sean conscientes de que esta acción es irreversible.

```
app.delete('/configuracion/eliminar', (req, res) => {
  if (!req.session.user)
    return res.status(401).json({ error: 'No autenticado' });

  const usuarioId = req.session.user.id;
  db.run(`DELETE FROM usuarios WHERE id = ?`, [usuarioId], function(err) {
    if (err)
      return res.status(500).json({ error: 'Error al eliminar perfil' });
    req.session.destroy(err => {
      if (err)
        return res.status(500).json({ error: 'Error al cerrar sesión después de eliminar perfil' });
      res.json({ message: 'Perfil eliminado correctamente' });
    });
  });
});
```

*Ilustración 23: Endpoint para eliminar perfil*

## 3.6 Seguridad

La seguridad es un elemento clave de la aplicación, por lo que se han adoptado diversas medidas para garantizar la protección de los datos de los usuarios:

- Hash de contraseñas: Las contraseñas de los usuarios se cifran al almacenarse en la BBDD, asegurando que las contraseñas no puedan ser leídas en texto plano, si surge algún acceso no autorizado.
- Gestión de sesiones: Para mantener la autenticación durante toda la interacción del usuario con la aplicación, se utiliza la librería express-session. Esto garantiza que solo los usuarios autenticados tengan acceso a sus datos.
- Protección la transmisión de datos: Se implementan procedimientos de validación y sanitización de entradas para evitar vulnerabilidades como inyecciones de SQL. Además, se emplea HTTPS para cifrar las

comunicaciones entre el cliente y el servidor, protegiendo así la información mientras se transmite.

### 3.7 Pruebas y Validación

Para asegurar el correcto funcionamiento de la aplicación, se han realizado diversas pruebas que incluyen pruebas manuales a través del navegador y Postman, así como pruebas con usuarios externos. Estas pruebas son esenciales para identificar y corregir errores, así como para validar que las funcionalidades cumplen con los requisitos establecidos.

#### 3.7.1 Pruebas manuales

Se realizaron pruebas manuales utilizando el navegador web y Postman, con el fin de interactuar directamente con la API de la aplicación. Este enfoque permite verificar el comportamiento de la aplicación en un entorno real y evaluar la experiencia del usuario.

- Navegador: Se accedió a la aplicación a través de localhost:3000 para probar las diferentes funcionalidades, como el registro de usuarios, inicio de sesión, gestión de transacciones y visualización de informes. Se verificó que cada funcionalidad funcionara como se esperaba y que la interfaz fuera intuitiva y fácil de usar. Queda reflejado en las ilustraciones que aparecerán posteriormente en Resultados.
- Postman: Se utilizaron solicitudes HTTP en Postman para probar los endpoints de la API. Adjunto varias de las pruebas realizadas:

#### Ejemplos de pruebas en Postman

Registro de usuario:

Método: POST

URL: <http://localhost:3000/registro>

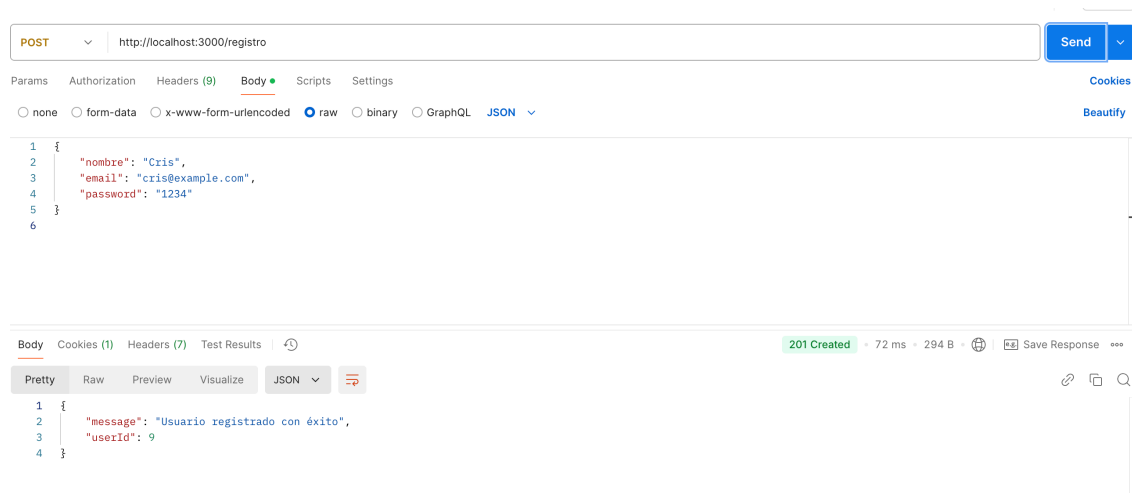


*Ilustración 24: Prueba de registro*

Verificación: Se espera un código de estado 201 y un mensaje de éxito.

Inicio de sesión:

Método: POST

URL: <http://localhost:3000/iniciosesion>

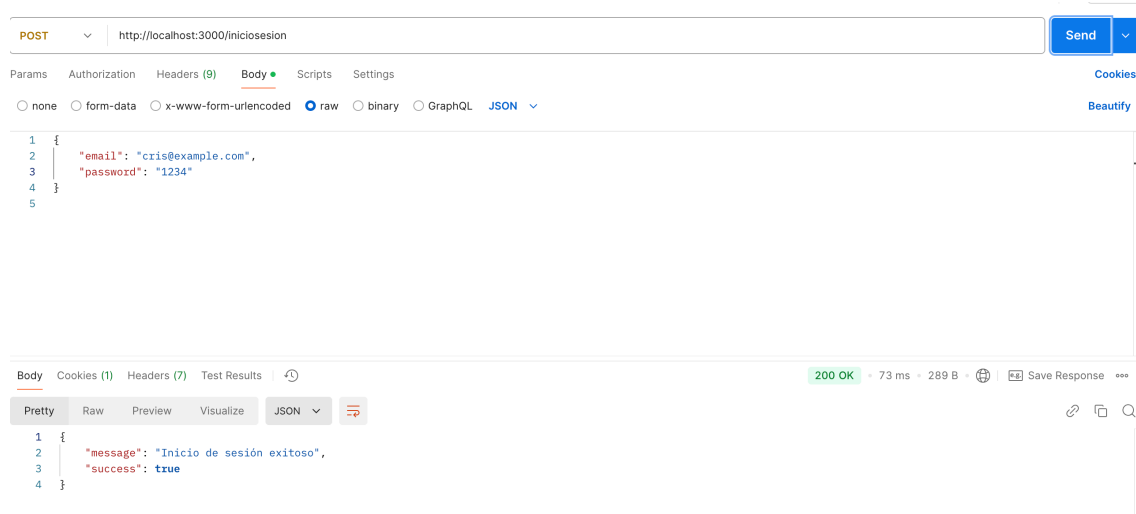


*Ilustración 25: Prueba de inicio de sesión*

Verificación: Se espera un código de estado 200 y un mensaje de inicio de sesión exitoso.

Inicio de sesión de un usuario con contraseña incorrecta:

Método: POST

URL: <http://localhost:3000/iniciosesion>

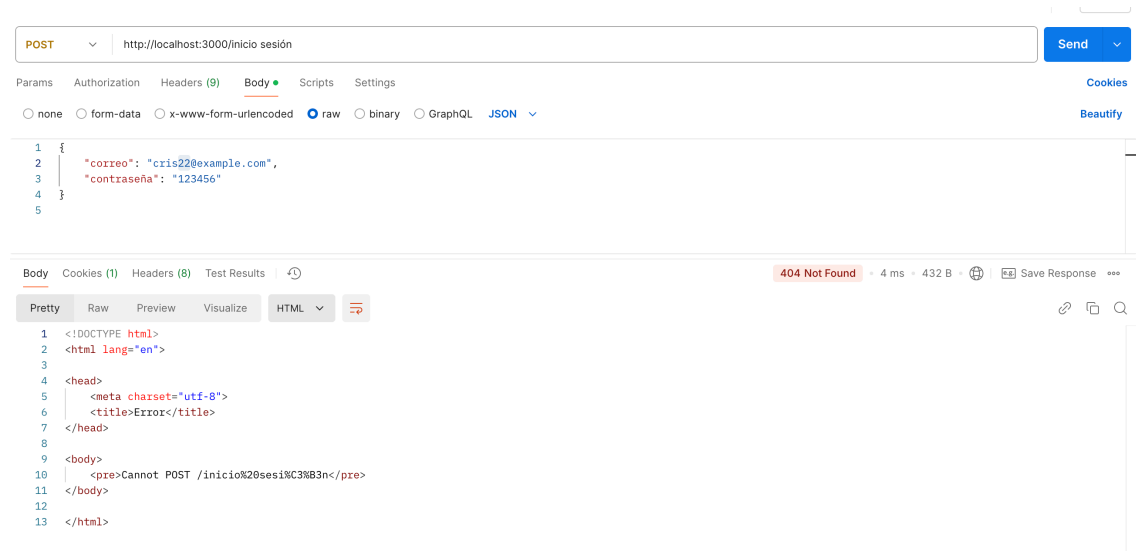


*Ilustración 26: Prueba inicio sesión con contraseña incorrecta*

Verificación: Error 404

Inicio de sesión de un usuario no registrado:

Método: POST

URL: <http://localhost:3000/iniciosesion>

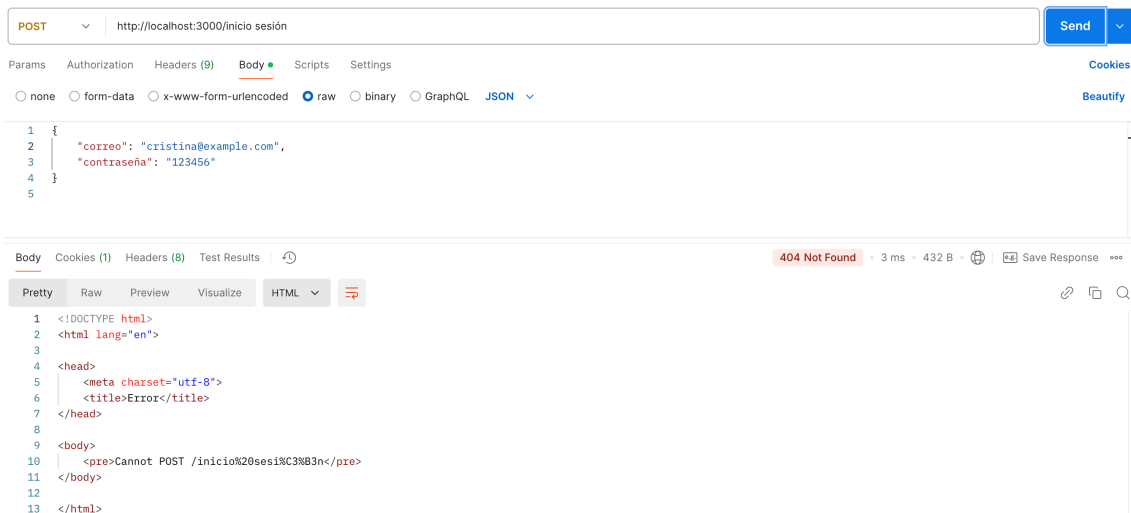


Ilustración 27: Prueba inicio sesión con usuario no registrado

Verificación: Error 404

Obtener el usuario actual:

Método: GET

URL: <http://localhost:3000/usuario>

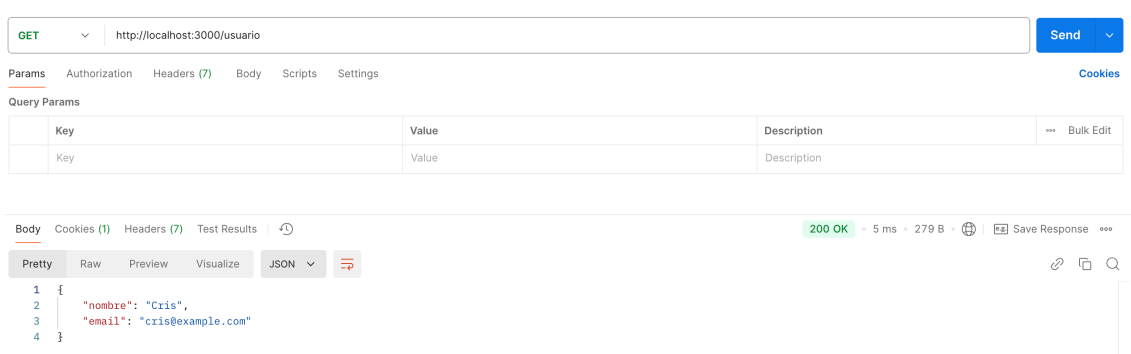


Ilustración 28: Prueba obtener usuario actual

Verificación: Se espera un código de estado 200 y los datos del usuario.

Agregar una transacción:

Método: POST

URL: <http://localhost:3000/transaccion>

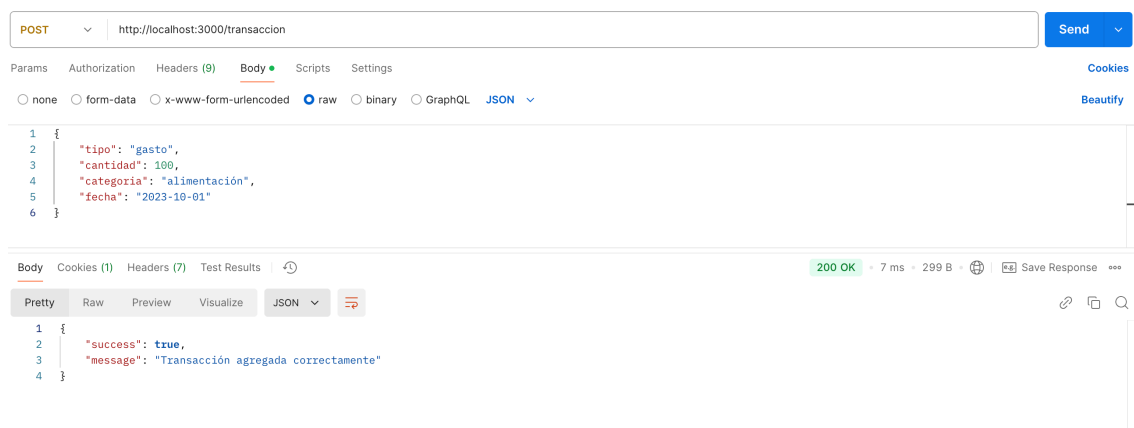


*Ilustración 29: Prueba agregar una transacción*

Verificación: Se espera un código de estado 200 y un mensaje de transacción agregada correctamente.

Obtener transacciones del usuario:

Método: GET

URL: <http://localhost:3000/transacciones>

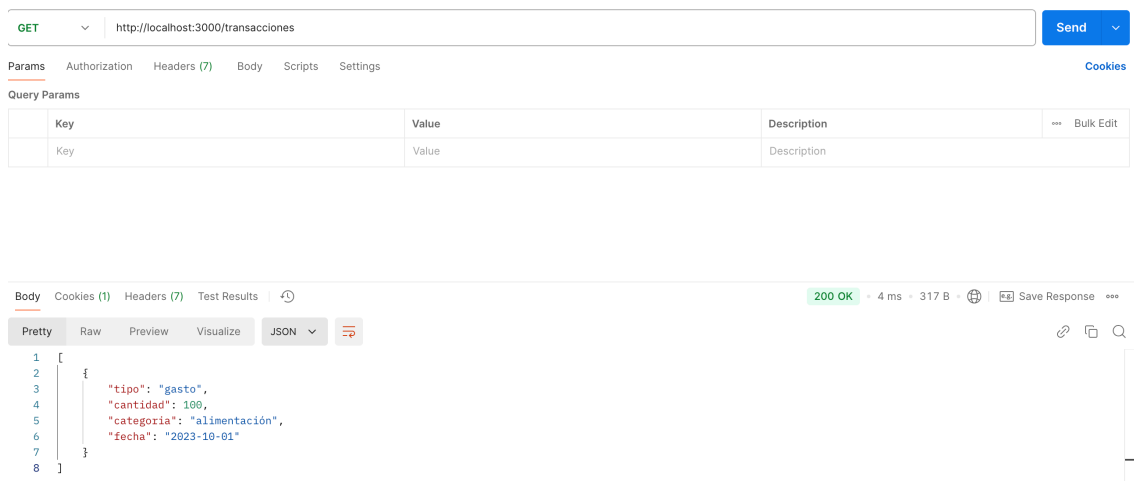


*Ilustración 30: Prueba ver transacciones*

Verificación: Se espera un código de estado 200 y los datos de las transacciones correspondientes.

Agregar presupuesto:

Método: POST

URL: <http://localhost:3000/presupuesto>

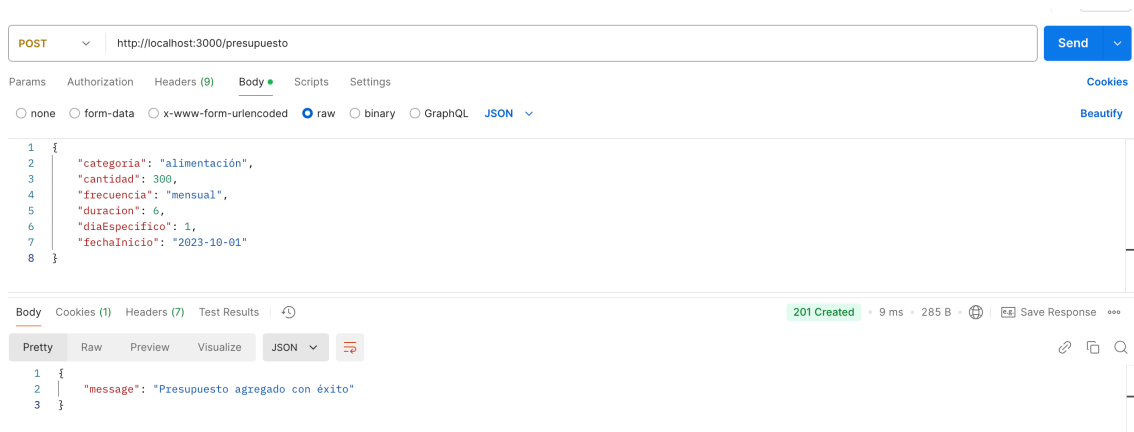


*Ilustración 31: Prueba agregar presupuesto*

Verificación: Se espera un código de estado 201 y un mensaje de transacción agregada con éxito.

Obtener presupuestos del usuario:

Método: GET

URL: <http://localhost:3000/presupuestos>

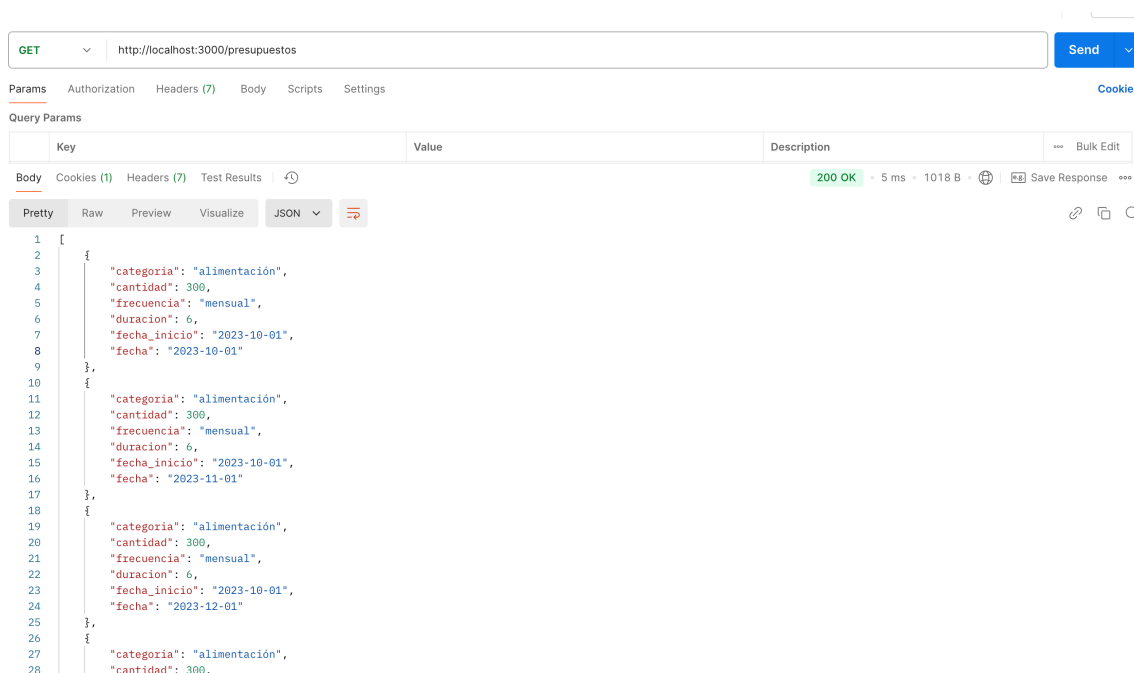


*Ilustración 32: Prueba ver los presupuestos*

Verificación: Se espera un código de estado 200 y los datos de los presupuestos correspondientes.

Obtener configuración del usuario:

Método: GET

URL: <http://localhost:3000/configuracion>

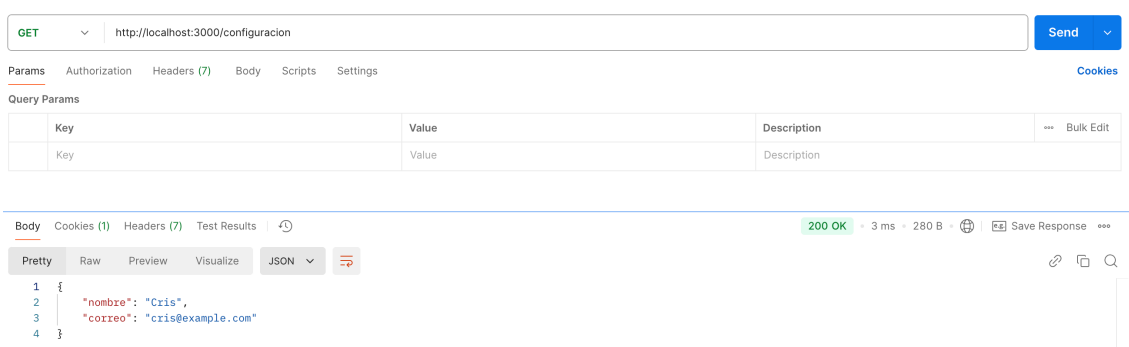


Ilustración 33: Prueba obtener configuración

Verificación: Se espera un código de estado 200 y los datos del usuario.

Modificar configuración del usuario:

Método: PUT

URL: <http://localhost:3000/configuracion>

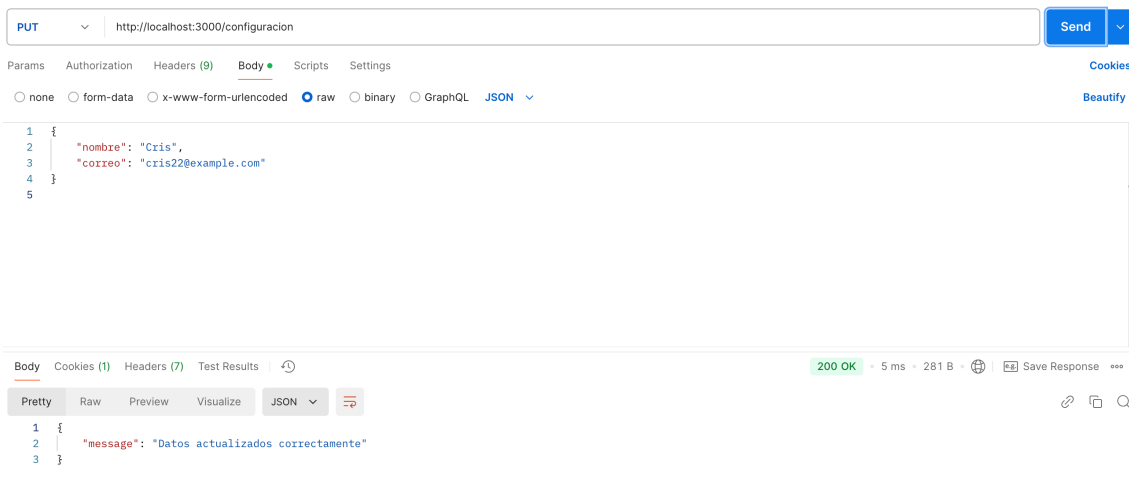


Ilustración 34: Prueba cambiar configuración

Verificación: Se espera un código de estado 200 y un mensaje de datos actualizados correctamente.

Obtener configuración del usuario:

Método: GET

URL: <http://localhost:3000/configuracion>

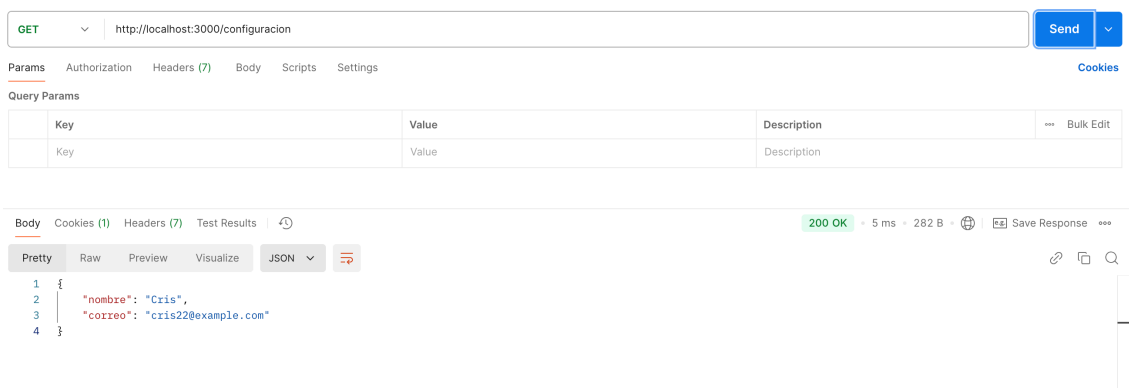


*Ilustración 35: Prueba volver a ver configuración usuario*

Verificación: Se espera un código de estado 200 y los datos del usuario, después de haber hecho el POST.

### 3.7.2 Pruebas con usuarios externos

Además, se llevaron a cabo varias sesiones de prueba con usuarios ajenos al equipo de desarrollo para evaluar la experiencia de usuario y la facilidad de uso de la aplicación. Durante estas pruebas, se recopilaron datos sobre la facilidad de uso, la navegación y la satisfacción general del usuario.

Los usuarios externos interactuaron con la aplicación y proporcionaron retroalimentación sobre las siguientes áreas:

- Usabilidad: Se evaluó si la interfaz era intuitiva y fácil de navegar.
- Funcionalidades: Se verificó que todas las funcionalidades, como el registro, inicio de sesión, gestión de transacciones y presupuestos, funcionaran correctamente.
- Satisfacción general: Se recopiló información sobre la experiencia general del usuario al utilizar la aplicación.

Con base en esta retroalimentación, se realizaron mejoras en la interfaz y en las funcionalidades de la aplicación, asegurando que se cumplieran las expectativas y necesidades de los usuarios finales.

## 4 Resultados y conclusiones

### 4.1 Resumen de los resultados obtenidos

A lo largo del desarrollo de la aplicación de gestión de finanzas personales, se han alcanzado varios resultados significativos que demuestran la efectividad y funcionalidad de la herramienta. Los principales resultados incluyen:

- **Funcionalidades implementadas:** Se han desarrollado y probado con éxito funcionalidades clave, como el registro y autenticación de usuarios, la gestión de transacciones, la creación y gestión de presupuestos, y la visualización de informes financieros.
- **Pruebas exhaustivas:** al realizarse exhaustivas pruebas manuales utilizando Postman y la navegación directa en el navegador, ha permitido poder identificar y corregir errores antes de la implementación final. Las pruebas con usuarios externos también proporcionaron valiosa retroalimentación sobre la usabilidad y la interfaz de la aplicación.
- **Interfaz de usuario:** La aplicación presenta una interfaz visualmente atractiva y diseñada para ser intuitiva, facilitando su uso por parte de los usuarios, diseñada con HTML, CSS y Bootstrap. Esto ha contribuido a una experiencia de usuario positiva, facilitando la navegación y el acceso a las diferentes funcionalidades.
- **Rendimiento y seguridad:** La aplicación ha sido diseñada para manejar múltiples solicitudes simultáneamente, gracias a la utilización de Node.js en el backend. Además, se han llevado a cabo diferentes medidas de seguridad para proteger la información de los usuarios.

### Imágenes de la aplicación

Registro:

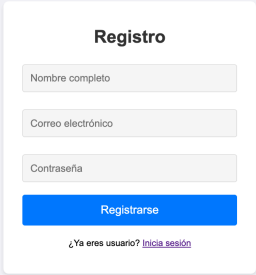


*Ilustración 36: Página de registro*

Inicio sesión:

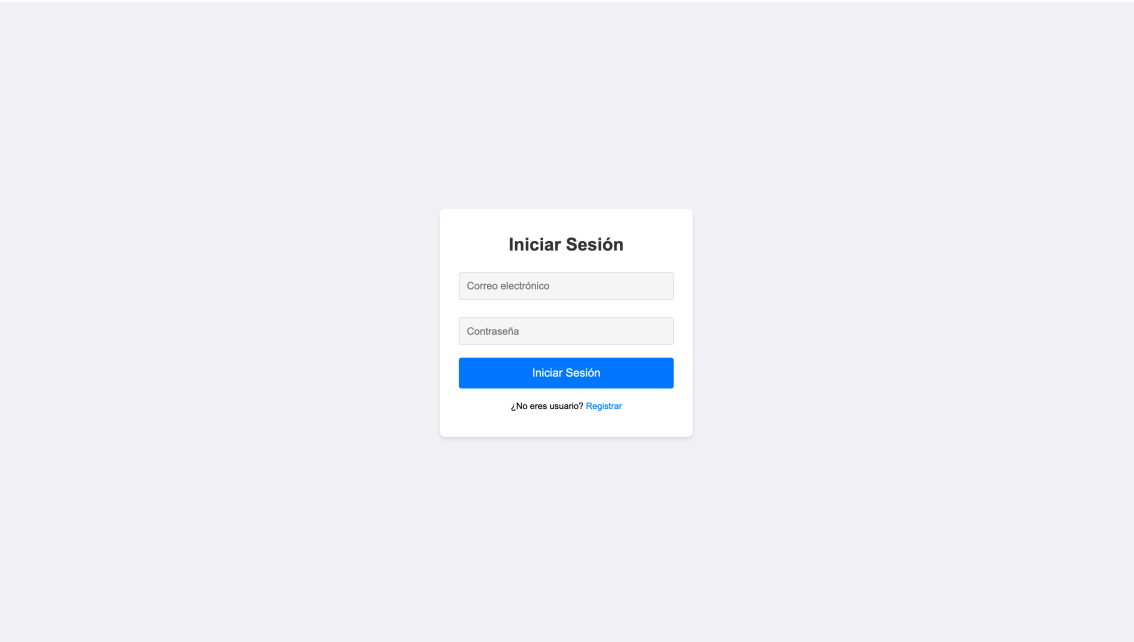


Ilustración 37: Página de inicio sesión

Página principal:

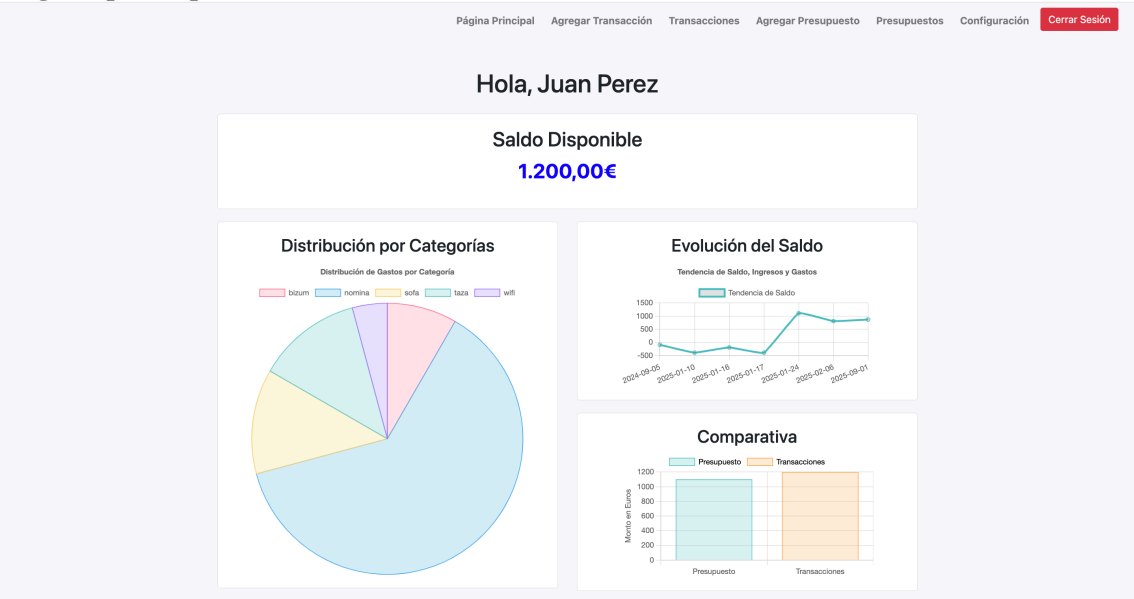


Ilustración 38: Página principal I

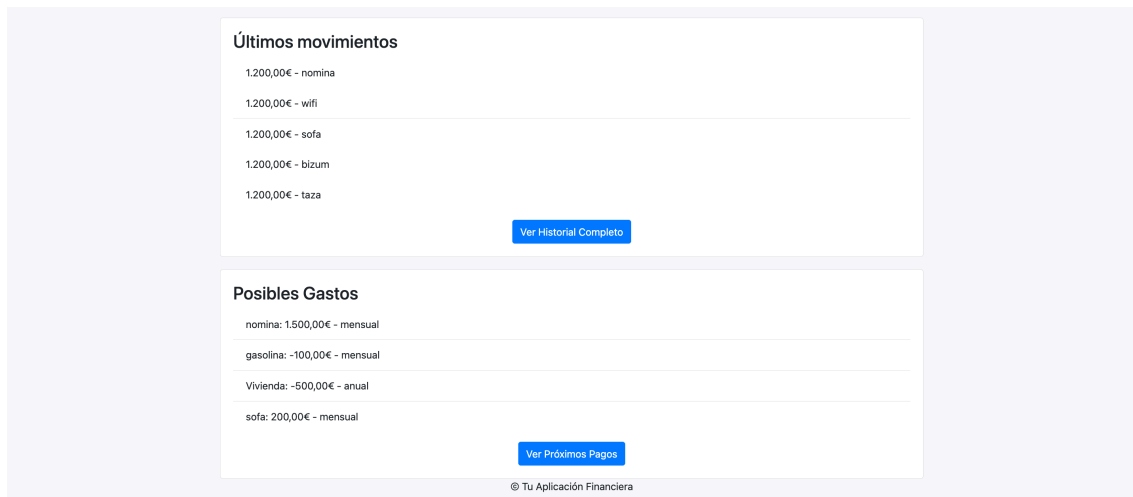


*Ilustración 39: Página principal II*

## Agregar Transacciones:

The screenshot shows a web application interface for adding a transaction. At the top, there is a navigation bar with links: 'Página Principal', 'Agregar Transacción', 'Transacciones', 'Agregar Presupuesto', 'Presupuestos', 'Configuración', and 'Cerrar Sesión'. The main heading is 'Agregar Transacción'. Below this, there is a form with the following fields: 'Tipo de Transacción:' with a dropdown menu showing 'Ingreso'; 'Cantidad:' with a text input field; 'Categoría:' with a text input field; and 'Fecha:' with a date picker showing 'dd/mm/aaaa'. At the bottom of the form is a blue button labeled 'Agregar Transacción'. At the bottom of the page, there is a small copyright notice: '© Tu Aplicación Financiera'.

*Ilustración 40: Página para agregar transacciones*

Historial de Transacciones:

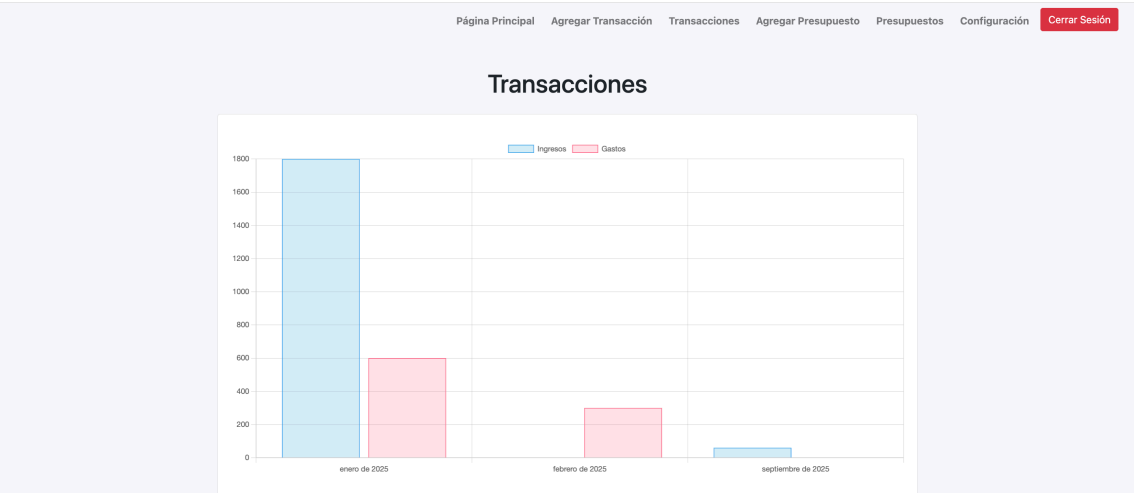


Ilustración 41: Historial de Transacciones I

|                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <div>enero de 2025</div> <div><div>taza: -300.00€ - undefined (Fecha: 10/1/2025)</div><div>bizum: 200.00€ - undefined (Fecha: 16/1/2025)</div><div>wifi: 100.00€ - undefined (Fecha: 17/1/2025)</div><div>sofa: -300.00€ - undefined (Fecha: 17/1/2025)</div><div>nomina: 1500.00€ - undefined (Fecha: 24/1/2025)</div></div> |
| <div>febrero de 2025</div> <div><div>comida: -300.00€ - undefined (Fecha: 6/2/2025)</div></div>                                                                                                                                                                                                                               |
| <div>septiembre de 2025</div> <div><div>gh: 60.00€ - undefined (Fecha: 1/9/2025)</div></div>                                                                                                                                                                                                                                  |

© Tu Aplicación Financiera

Ilustración 42: Historial de transacciones II

Agregar Presupuestos:

Página Principal

Agregar Transacción

Transacciones

Agregar Presupuesto

Presupuestos

Configuración

Cerrar Sesión

Agregar Presupuesto

Tipo:

Ingreso

Categoría:

Cantidad:

Frecuencia:

Mensual

Duración (en meses o años):

Fecha de Inicio:

dd/mm/aaaa

Agregar Presupuesto

© Tu Aplicación Financiera

Ilustración 43: Agregar presupuestos

Página Principal

Agregar Transacción

Transacciones

Agregar Presupuesto

Presupuestos

Configuración

Cerrar Sesión

## Presupuestos

Mensual

1 año

The bar chart displays the monthly budget for the year 2025. The y-axis represents the budget amount, ranging from 0 to 1600 in increments of 200. The x-axis lists the months from January to December. All bars are purple and have a height of approximately 1600.

| Mes        | Budget (Año 2025) |
|------------|-------------------|
| enero      | 1600              |
| febrero    | 1600              |
| marzo      | 1600              |
| abril      | 1600              |
| mayo       | 1600              |
| juno       | 1600              |
| julio      | 1600              |
| agosto     | 1600              |
| septiembre | 1600              |
| octubre    | 1600              |
| noviembre  | 1600              |
| diciembre  | 1600              |

## Configuración:

Página Principal   Agregar Transacción   Transacciones   Agregar Presupuesto   Presupuestos   Configuración

### Configuración

#### Perfil del Usuario

Nombre:

Correo:

[Guardar Cambios](#)

#### Configuración de Seguridad

Nueva Contraseña:

Confirmar Contraseña:

[Cambiar Contraseña](#)

[Cerrar Sesión](#)   [Eliminar Perfil](#)

© Tu Aplicación Financiera

*Ilustración 46: Configuración*

## 4.2 Conclusiones personales

El proceso de desarrollo de esta aplicación de gestión de finanzas personales ha resultado ser una experiencia valiosa y formativa. A través de este proyecto, he podido poner en práctica y consolidar mis conocimientos en desarrollo web, programación y la gestión de bases de datos. Algunas de las conclusiones personales que he extraído del trabajo realizado son:

- **Importancia de la planificación:** La planificación inicial del proyecto fue fundamental para establecer un enfoque claro y estructurado. Definir los objetivos y funcionalidades desde el principio facilitó el desarrollo y la implementación de la aplicación.
- **Valor de las pruebas:** La realización de pruebas exhaustivas ha sido crucial para garantizar la calidad del producto final. Aprendí que las pruebas no solo ayudan a identificar errores, sino que también proporcionan una mejor comprensión de la experiencia del usuario.
- **Adaptabilidad y aprendizaje continuo:** Durante el desarrollo, enfrenté varios desafíos técnicos que requirieron investigación y aprendizaje adicional.
- **Impacto en la gestión financiera:** Al desarrollar esta aplicación, he tomado conciencia de la importancia de la gestión financiera personal. La herramienta que he creado no solo es un proyecto académico, sino que también tiene el potencial de ayudar a otros a llevar un mejor control de sus finanzas.

En conclusión, este trabajo de fin de grado no solo ha sido una oportunidad para aplicar mis habilidades técnicas, sino también una experiencia valiosa que me ha permitido crecer como desarrolladora. Estoy satisfecho con los resultados obtenidos y espero que la aplicación sea útil para quienes deseen mejorar su gestión financiera personal.

## **5 Análisis de Impacto**

### **5.1 Impacto Personal**

La aplicación de gestión de finanzas personales permite a los usuarios controlar mejor sus ingresos y gastos, mejorando su salud financiera. Facilita decisiones informadas sobre ahorro e inversión, promoviendo hábitos financieros saludables. Sin embargo, podría generar una dependencia excesiva de la tecnología y descuidar la educación financiera básica.

### **5.2 Impacto Empresarial**

La aplicación puede ayudar a PYMES a gestionar sus finanzas, mejorando la eficiencia operativa y la planificación financiera, fomentando la transparencia y la responsabilidad, y mejorando la reputación empresarial. No obstante, puede requerir capacitación adicional para los empleados, generando costos iniciales.

### **5.3 Impacto Social**

Promueve la inclusión financiera al facilitar el acceso a herramientas de gestión para personas sin servicios financieros, contribuyendo a la reducción de la pobreza. Sin embargo, la falta de acceso a tecnología o internet en ciertas comunidades puede limitar su impacto positivo.

### **5.4 Impacto Económico**

Mejora la gestión financiera personal, estimulando el ahorro y la inversión, lo que puede contribuir al crecimiento económico y a una mayor estabilidad financiera de los usuarios. Sin embargo, un uso incorrecto puede llevar a decisiones financieras erróneas, afectando negativamente la situación económica de los usuarios.

### **5.5 Impacto Medioambiental**

La aplicación fomenta un consumo consciente y responsable, ayudando a los usuarios a tomar decisiones que consideren el impacto ambiental de sus gastos. Sin embargo, si promueve el consumo excesivo o innecesario, podría aumentar la producción de residuos y el uso de recursos.

### **5.6 Impacto Cultural**

Ayuda a incrementar la conciencia sobre la relevancia de la educación financiera y la gestión responsable de los recursos, impactando de manera positiva en la cultura financiera de la comunidad. No obstante, puede llevar a una homogeneización de prácticas financieras, dejando de lado enfoques culturales tradicionales.

### **5.7 Decisiones tomadas considerando el impacto**

Durante el desarrollo del TFG, se tomaron decisiones que evaluaron el impacto potencial de la aplicación. Algunos ejemplos incluyen:

- **Diseño inclusivo:** Se priorizó la creación de una interfaz intuitiva y accesible, lo que permite que personas con diferentes niveles de habilidad tecnológica puedan utilizar la aplicación sin dificultades.
- **Sostenibilidad:** Se consideró el impacto medioambiental al promover un consumo responsable y consciente, animando a los usuarios a reflexionar sobre sus decisiones de gasto.
- **Pruebas con usuarios diversos:** Se realizaron pruebas con un grupo diverso de usuarios para asegurar que la aplicación sea útil y relevante para diferentes contextos culturales y socioeconómicos.

## 6 Bibliografía

- [1] Gómez, R., & Castillo, P., "La importancia de una interfaz intuitiva en las aplicaciones financieras," *Journal of Financial Technology*, vol. 14, no. 3, pp. 19-25, 2020. Accessed: Oct. 22, 2024
- [2] Fernández, M., & López, J., "Principios de gestión financiera personal para el usuario promedio," *Revista de Educación Financiera*, vol. 9, no. 2, pp. 33-45, 2022. Accessed: Oct. 22, 2024
- [3] Martínez, A., González, L., & Sánchez, E., "Métodos para el ahorro y la gestión de finanzas personales," *Estudios de Economía*, vol. 23, no. 4, pp. 55-64, 2022. Accessed: Dec. 10, 2024
- [4] Google Developers, "Desarrollo de aplicaciones web progresivas," Google Developers. [Online]. Available: <https://developers.google.com/web/progressive-web-apps>. Accessed: Oct. 26, 2024
- [5] Microsoft, "Visual Studio Code: Documentación oficial," Microsoft Docs. [Online]. Available: <https://code.visualstudio.com/docs>. Accessed: Oct. 28, 2024
- [6] Ecma International. (2021). ECMAScript 2021 Language Specification. [Online]. Available: <https://www.ecma-international.org/publications/standards/Ecma-262.htm>. Accessed: Oct. 28, 2024
- [7] World Wide Web Consortium (W3C). (2019). HTML5 specification. [Online]. Available: <https://www.w3.org/TR/html5/>. Accessed: Oct. 20, 2024
- [8] World Wide Web Consortium (W3C). (2018). CSS: *Cascading Style Sheets*. [Online]. Available: <https://www.w3.org/Style/CSS/>. Accessed: Oct. 20, 2024
- [9] SQLite. (2025). SQLite Official Documentation. [Online]. Available: <https://www.sqlite.org/>. Accessed: Oct. 25, 2024
- [10] Express.js, "Guía oficial de Express.js," Express.js Documentation. [Online]. Available: <https://expressjs.com/>. Accessed: Oct. 27, 2024
- [11] Bootstrap. (2025). Bootstrap Official Site. [Online]. Available: <https://getbootstrap.com/>. Accessed: Oct. 28, 2024
- [12] Chart.js. (2025). Chart.js Official Documentation. [Online]. Available: <https://www.chartjs.org/>. Accessed: Oct. 27, 2024
- [13] bcrypt Contributors. (2021). bcrypt. [Online]. Available: <https://www.npmjs.com/package/bcrypt>. Accessed: Nov. 04, 2024
- [14] body-parser Contributors. (2021). body-parser [Online]. Available: <https://www.npmjs.com/package/body-parser>. Accessed: Oct. 28, 2024
- [15] express-session Contributors. (2021). express-session [Online]. Available: <https://www.npmjs.com/package/express-session>. Accessed: Nov. 04, 2024
- [16] connect-sqlite3 Contributors. (2021). connect-sqlite3. [Online]. Available: <https://www.npmjs.com/package/connect-sqlite3>. Accessed: Oct. 28, 2024
- [17] Node.js Foundation. (2021). Node.js. [Online]. Available: <https://nodejs.org/>. Accessed: Oct. 20, 2024
- [18] Postman Contributors. (2021). Postman. [Online]. Available: <https://www.postman.com/>. Accessed: Oct. 20, 2024

## 7 Informe de Originalidad



Turnitin ID: 2563626264

5%



### Recibo digital

Este recibo confirma que su trabajo ha sido recibido por **Turnitin**. A continuación podrá ver la información del recibo con respecto a su entrega.

La primera página de tus entregas se muestra abajo.

Autor de la entrega: CRISTINA PERULERO CABRERA  
Título del ejercicio: Turnitin Memoria Final  
Título de la entrega: TFG\_CristinaPeruleroCabrera.pdf  
Nombre del archivo: 31036\_CRISTINA\_PERULERO\_CABRERA\_TFG\_CristinaPerulero...  
Tamaño del archivo: 4.68M  
Total páginas: 39  
Total de palabras: 7,865  
Total de caracteres: 50,241  
Fecha de entrega: 13-ene.-2025 09:40p. m. (UTC+0100)  
Identificador de la entre... 2563626264



Derechos de autor 2025 Turnitin. Todos los derechos reservados.

Este documento esta firmado por



|                               |                                                                                                                     |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------|
| <b>Firmante</b>               | CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES                                               |
| <b>Fecha/Hora</b>             | Tue Jan 14 00:01:01 CET 2025                                                                                        |
| <b>Emisor del Certificado</b> | EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES |
| <b>Numero de Serie</b>        | 561                                                                                                                 |
| <b>Metodo</b>                 | urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)                                                       |