Proyecto Final Tercer Parcial

Equipo:

- **Freda Maria Perez Novelo (200300593)**
- **Brisa Jael Lima Arenas (200300606)**

Carrera: Ingeniería en Datos e Inteligencia Organizacional.

Asignatura: Tópicos selectos en ingeniería de datos.

Profesor: Ismael Domínguez Jiménez.

Fecha de entrega: 28 de Noviembre.

En el contexto de un negocio de venta de discos, ya sea en formato CD, vinil u otros productos relacionados, la gestión eficiente del inventario, la consulta de productos y la posibilidad de insertar nuevos productos a la tienda son aspectos esenciales para mantener la operatividad del sistema. Sin una solución automatizada, estos procesos requieren mucho tiempo y esfuerzo, además de estar propensos a errores humanos.

Base de Datos:

La base de datos fue creada en la herramienta MongoDB con el nombre de Datos y la colección de la base se llama Datos1, está configurada para almacenar información sobre una tienda de discos, donde cada documento contiene detalles como el nombre del álbum, el artista, el formato (vinil o CD), el año de lanzamiento, el precio, la descripción y la cantidad en stock.

app.py

Este archivo es la entrada de la API, utilizando el framework Flask para definir las rutas y manejar las solicitudes HTTP. Las funcionalidades principales incluyen:
Librerías:

```
from flask import Flask, jsonify, request
from flask_cors import CORS
from services.data_service import DataService
```

Mensaje de bienvenida:

```
@app.route('/')
def home():
  return "Bienvenido a la API de la Tienda de Discos"
```

Método GET: Recupera todos los datos almacenados en la base de datos y los devuelve en formato JSON.

```
@app.route('/data', methods=['GET'])
def get_data():
        data = data_service.get_all_data()
        return jsonify(data)
```

Método POST: Inserta una lista de productos en la base de datos. Los productos incluyen información como el nombre, categoría, precio, stock, entre otros.

```
@app.route('/insert_data', methods=['POST'])
def insert_data():
        data = [...]
        data_service.insert_data(data)
        return jsonify({"message": "Datos insertados correctamente"}), 201
```

data_service.py

Maneja la interacción con MongoDB, proporcionando métodos para obtener, insertar y procesar los datos.

Librerias:

```
from pymongo import MongoClient

from bson.json_util import dumps

import json

import os

from dotenv import load_dotenv
```

Conexión a MongoDB: Se establece mediante el cliente MongoClient, utilizando variables de entorno cargadas desde un archivo .env.

```
self.client = MongoClient(os.getenv('MONGO_URI'))
self.db = self.client[os.getenv('DB_NAME')]
self.collection = self.db[os.getenv('COLLECTION_NAME')]
```

Métodos:

get_all_data(): Obtiene todos los datos almacenados en la colección y los convierte a formato JSON.

```
def get_all_data(self):
        data = self.collection.find({})
        return json.loads(dumps(data))
```

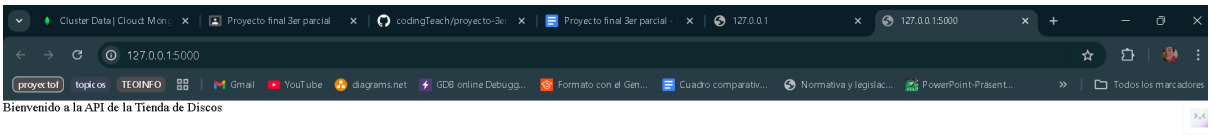
Filtra los datos por categoría (CD, Vinil, etc.) y los devuelve en formato JSON.

```
def get_data_by_category(self, category):
        data = self.collection.find({"categoria": category})
        return json.loads(dumps(data))
```

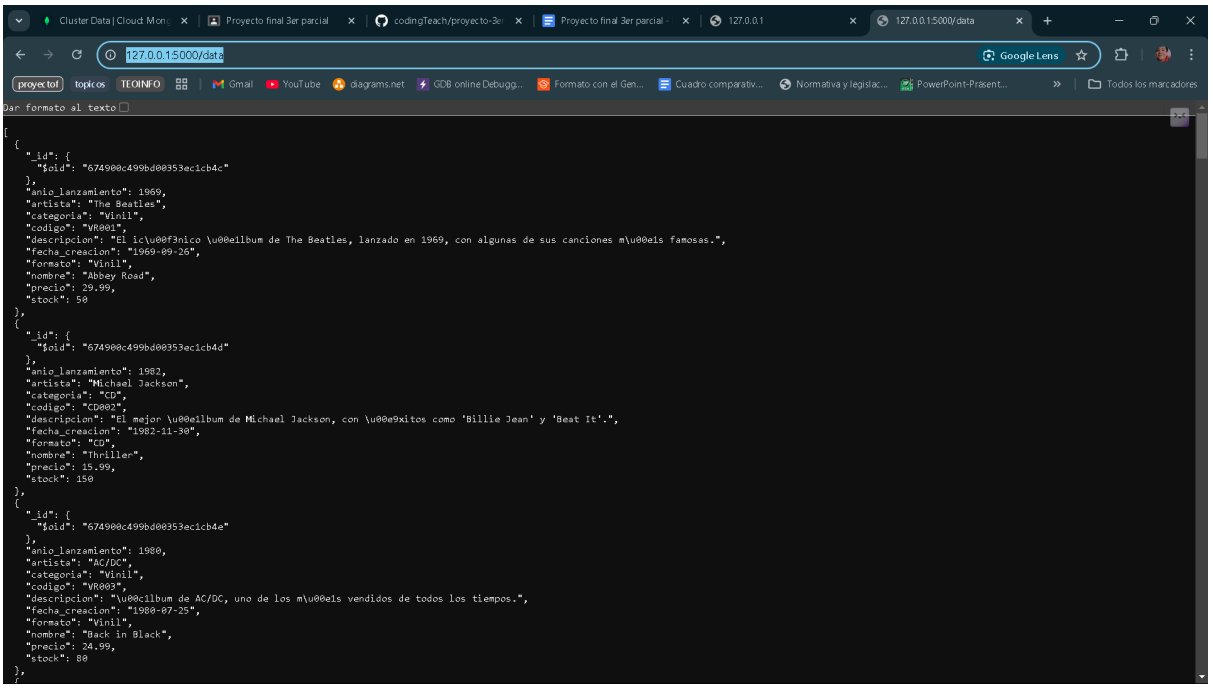
Inserta un listado de productos en la base de datos utilizando el método insert_many de MongoDB.

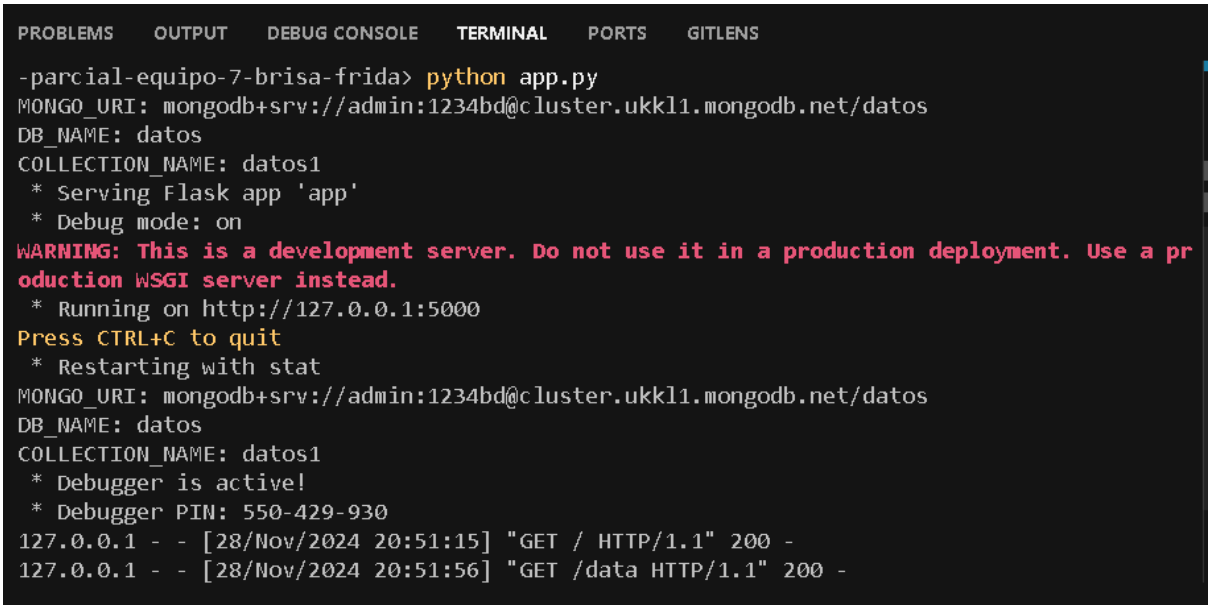
```
def insert_data(self, data):
    if isinstance(data, list):
        self.collection.insert_many(data)
    else:
        self.collection.insert_one(data)
        print("Datos insertados correctamente.")
```

Mensaje bienvenida: http://127.0.0.1:5000/



Datos: http://127.0.0.1:5000/data





backed:

data_service.py

```
from config.db_config import get_db

db = get_db()

def get_all_data():
    collection = db['datos']
    return list(collection.find({}, {'_id': 0}))

def get_data_by_category(category):
    collection = db['datos']
    return list(collection.find({"categoria": category}, {'_id': 0}))
```

Importación de dependencias:
El archivo importa la función get_db desde config.db_config, que se encarga de obtener la conexión a la base de datos.

Conexión a la base de datos:
La variable db obtiene la instancia de la base de datos utilizando la función get_db().

Función get_all_data():

Esta función accede a la colección 'datos' dentro de la base de datos y retorna todos los documentos de la colección sin incluir el campo _id (gracias al filtro {'_id': 0}).
Utiliza el método find() para recuperar todos los registros de la colección y los convierte en una lista utilizando list(), que se devuelve como resultado.
Función get_data_by_category(category):

Esta función permite obtener los documentos de la colección 'datos' que pertenezcan a una categoría específica.
El parámetro category se utiliza en la consulta para filtrar los documentos donde el campo "categoria" coincida con el valor proporcionado.
Al igual que en la función anterior, el filtro {'_id': 0} se utiliza para omitir el campo _id de los resultados, y los documentos se devuelven como una lista.

fronted:

app.js

```
function fetchData() {
    const category = document.getElementById('category').value;
    const endpoint = category ? /data/${category} : '/data';

    fetch(endpoint)
        .then(response => response.json())
        .then(data => displayData(data))
        .catch(error => console.error('Error:', error));
}

function displayData(data) {
    const display = document.getElementById('data-display');
    display.innerHTML = '';

    if (data.length === 0) {
        display.innerHTML = '<p>No hay datos para mostrar.</p>';
        return;
    }

    data.forEach(item => {
        const entry = document.createElement('div');
        entry.textContent = JSON.stringify(item);
        display.appendChild(entry);
    });
}
```

charts.js

```
function createCategoryChart() {
    fetch('/data')
        .then(response => response.json())
        .then(data => {
            const categories = ['CD', 'Vinil'];
            const stockData = categories.map(category =>
                data.filter(item => item.formato === category)
                    .reduce((sum, item) => sum + item.stock, 0)
```

```
        );
        const ctx = document.getElementById('categoryChart').getContext('2d');
        new Chart(ctx, {
            type: 'pie',
            data: {
                labels: categories,
                datasets: [{
                    data: stockData,
                    backgroundColor: ['#4CAF50', '#FF5722']
                }]
            },
            options: {
                plugins: {
                    legend: {
                        display: true
                    }
                }
            }
        });
    })
    .catch(error => console.error('Error:', error));
}

function createTimeChart() {
    fetch('/data')
        .then(response => response.json())
        .then(data => {
            const years = [...new Set(data.map(item => item.anio_lanzamiento))].sort();
            const stockByYear = years.map(year =>
                data.filter(item => item.anio_lanzamiento === year)
                    .reduce((sum, item) => sum + item.stock, 0)
            );
            const ctx = document.getElementById('timeChart').getContext('2d');
            new Chart(ctx, {
                type: 'bar',
                data: {
                    labels: years,
                    datasets: [{
                        label: 'Stock Total',
                        data: stockByYear,
                        backgroundColor: '#2196F3'
                    }]
                },
                options: {
                    scales: {
                        y: {
                            beginAtZero: true
                        }
```

```
                }
            }
        });
    })
    .catch(error => console.error('Error:', error));
}

function createPriceChart() {
    fetch('/data')
    .then(response => response.json())
    .then(data => {
        const categories = ['CD', 'Vinil'];
        const avgPrice = categories.map(category => {
            const filtered = data.filter(item => item.formato === category);
            return filtered.reduce((sum, item) => sum + item.precio, 0) / filtered.length;
        });
        const ctx = document.getElementById('priceChart').getContext('2d');
        new Chart(ctx, {
            type: 'bar',
            data: {
                labels: categories,
                datasets: [{
                    label: 'Precio Promedio',
                    data: avgPrice,
                    backgroundColor: ['#673AB7', '#FFC107']
                }]
            },
            options: {
                scales: {
                    y: {
                        beginAtZero: true
                    }
                }
            }
        });
    })
    .catch(error => console.error('Error:', error));
}

function createPriceDistributionChart() {
    fetch('/data')
    .then(response => response.json())
    .then(data => {
        const categories = ['CD', 'Vinil'];
        const avgPrice = categories.map(category => {
            const filtered = data.filter(item => item.formato === category);
            return filtered.reduce((sum, item) => sum + item.precio, 0) / filtered.length;
        });
```

```
        const ctx =
document.getElementById('priceDistributionChart').getContext('2d');
        new Chart(ctx, {
            type: 'pie',
            data: {
                labels: categories,
                datasets: [{
                    data: avgPrice,
                    backgroundColor: ['#FF9800', '#9C27B0']
                }]
            },
            options: {
                plugins: {
                    legend: {
                        display: true
                    }
                }
            }
        });
    })
    .catch(error => console.error('Error:', error));
}

function createStockEvolutionChart() {
    fetch('/data')
        .then(response => response.json())
        .then(data => {
            const years = [...new Set(data.map(item => item.anio_lanzamiento))].sort();
            const stockByYear = years.map(year =>
                data.filter(item => item.anio_lanzamiento === year)
                    .reduce((sum, item) => sum + item.stock, 0)
            );
            const ctx = document.getElementById('stockEvolutionChart').getContext('2d');
            new Chart(ctx, {
                type: 'line',
                data: {
                    labels: years,
                    datasets: [{
                        label: 'Stock Total',
                        data: stockByYear,
                        borderColor: '#4CAF50',
                        fill: false
                    }]
                },
                options: {
                    scales: {
                        y: {
                            beginAtZero: true
```

```
                }
            }
        }
    });
})
.catch(error => console.error('Error:', error));
}

function createTotalStockByFormatChart() {
    fetch('/data')
        .then(response => response.json())
        .then(data => {
            const formats = ['CD', 'Vinil'];
            const stockByFormat = formats.map(format =>
                data.filter(item => item.formato === format)
                    .reduce((sum, item) => sum + item.stock, 0)
            );
            const ctx =
document.getElementById('totalStockByFormatChart').getContext('2d');
            new Chart(ctx, {
                type: 'bar',
                data: {
                    labels: formats,
                    datasets: [{
                        label: 'Total de Stock',
                        data: stockByFormat,
                        backgroundColor: ['#FF5722', '#8BC34A']
                    }]
                },
                options: {
                    scales: {
                        y: {
                            beginAtZero: true
                        }
                    }
                }
            });
        })
        .catch(error => console.error('Error:', error));
}

document.addEventListener('DOMContentLoaded', () => {
    createCategoryChart();
    createTimeChart();
    createPriceChart();
    createPriceDistributionChart();
    createStockEvolutionChart();
    createTotalStockByFormatChart();
```

```
});
```

styles.css

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    margin: 20px;
}

h1 {
    color: #333;
}

#data-display {
    margin-top: 20px;
    text-align: left;
    padding: 10px;
    border: 1px solid #ddd;
    background-color: #f9f9f9;
}
```

body:

- **font-family:** Arial, sans-serif: Establece la fuente principal de la página como Arial, con una fuente secundaria genérica sans-serif en caso de que Arial no esté disponible.
- t**ext-align: center:** Centra el texto de la página horizontalmente.
- **margin: 20px:** Aplica un margen de 20 píxeles alrededor de todo el contenido de la página.
- **h1:**
- **color: #333:** Establece el color del texto de los encabezados <h1> a un tono gris oscuro (#333), lo que lo hace legible pero menos intenso que el negro puro.
- **#data-display:**
- **margin-top: 20px:** Aplica un margen de 20 píxeles en la parte superior del contenedor con el id data-display, separándolo de otros elementos arriba de él.
- **text-align: left:** Alinea el texto dentro de #data-display a la izquierda.
- **padding: 10px:** Agrega un relleno de 10 píxeles dentro de #data-display, creando un espacio entre el contenido y los bordes del contenedor.
- **border: 1px solid #ddd**: Agrega un borde de 1 píxel de grosor con un color gris claro (#ddd) alrededor de #data-display.
- **background-color: #f9f9f9:** Establece un color de fondo gris muy claro (#f9f9f9) para el contenedor, lo que proporciona un contraste sutil con el borde.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
    <title>Dashboard de Gráficos</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .charts-container {
            display: grid;
            grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
            gap: 20px;
            justify-content: center;
        }
        .chart-container {
            text-align: center;
        }
        canvas {
            margin: 0 auto;
            max-width: 100%;
        }
    </style>
</head>
<body>
    <h1>Dashboard de Tienda de Discos</h1>

    <div class="charts-container">
        <div class="chart-container">
            <h2>Distribución de Stock por Categoría</h2>
            <canvas id="categoryChart"></canvas>
        </div>

        <div class="chart-container">
            <h2>Stock por Año de Lanzamiento</h2>
            <canvas id="timeChart"></canvas>
        </div>

        <div class="chart-container">
            <h2>Precio Promedio por Categoría</h2>
            <canvas id="priceChart"></canvas>
        </div>

        <div class="chart-container">
            <h2>Distribución de Precio por Categoría</h2>
            <canvas id="priceDistributionChart"></canvas>
        </div>

        <div class="chart-container">
```

```
        <h2>Evolución de Stock por Año</h2>
        <canvas id="stockEvolutionChart"></canvas>
      </div>

      <div class="chart-container">
        <h2>Total de Discos por Formato</h2>
        <canvas id="totalStockByFormatChart"></canvas>
      </div>
    </div>

    <script src="{{ url_for('send_static', path='js/charts.js') }}"></script>
</body>
</html>
```

Estructura General:

<html lang="en">: Define el idioma de la página como inglés.
<head>: Contiene los metadatos y enlaces a recursos externos (como la hoja de estilo CSS y la librería Chart.js).
<meta charset="UTF-8">: Establece el conjunto de caracteres como UTF-8 para soportar caracteres especiales.
<meta name="viewport" content="width=device-width, initial-scale=1.0">: Permite que la página sea responsiva y se ajuste correctamente en dispositivos móviles.
<link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">: Enlaza el archivo CSS para los estilos de la página. El url_for es utilizado en Flask para generar la ruta correcta hacia el archivo estático.
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>: Enlaza la biblioteca Chart.js, utilizada para crear los gráficos en la página.
<style>: Define estilos específicos para la página en línea, que personalizan el aspecto de los gráficos y el layout.

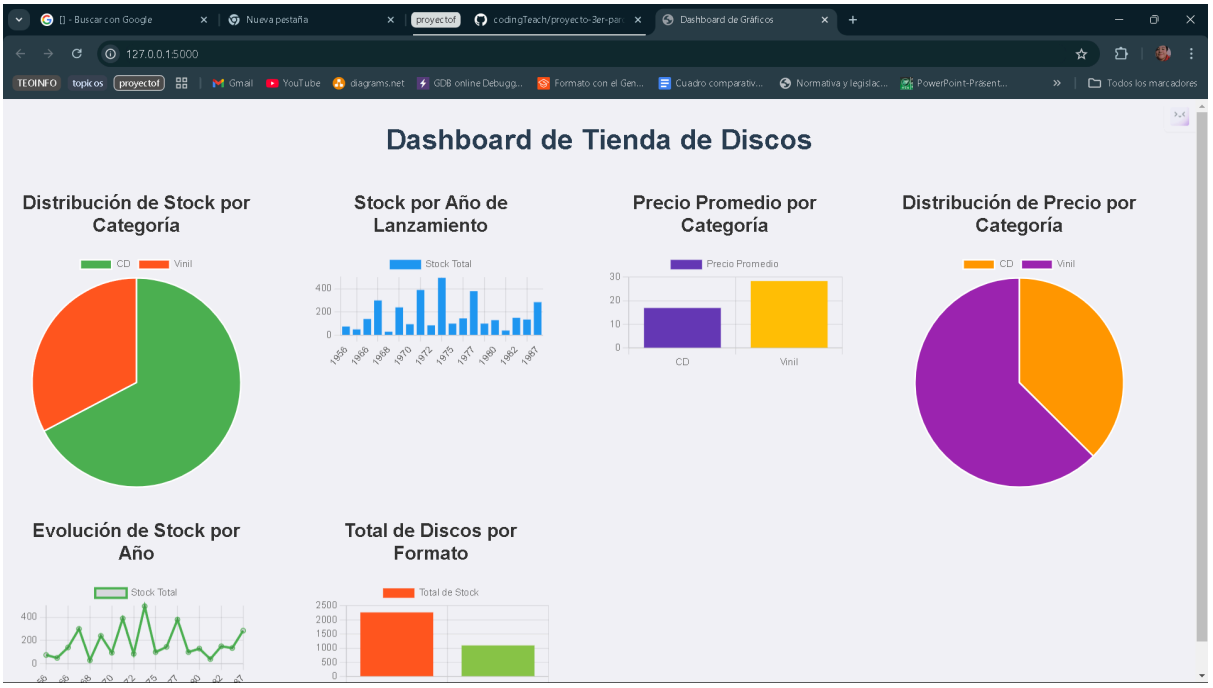
Cuerpo de la Página (<body>):

<h1>: Muestra el título principal del dashboard: "Dashboard de Tienda de Discos".
: Es un contenedor para los gráficos. Se utiliza un diseño de cuadrícula (grid) para organizar los gráficos de forma intercalada y flexible.
Cada contiene:
Un título (<h2>) con el nombre del gráfico.
Un <canvas> con un id único, como categoryChart, timeChart, etc., que es donde se renderizan los gráficos.

Gráficos:

- Gráfico de distribución de stock por categoría: Muestra la distribución del stock de discos por categorías (como 'CD' y 'Vinil').
- Gráfico de stock por año de lanzamiento: Muestra la cantidad de stock disponible en función del año de lanzamiento de los discos.
- Gráfico de precio promedio por categoría: Presenta el precio promedio de los discos según su categoría.
- Gráfico de distribución de precio por categoría: Muestra la distribución de precios entre las diferentes categorías.

- Gráfico de evolución de stock por año: Muestra cómo ha cambiado el stock de discos a lo largo de los años.
- Gráfico de total de discos por formato: Presenta la cantidad total de discos disponibles según su formato (por ejemplo, CD o Vinil).



Conclusión

A lo largo del desarrollo de este proyecto, se lograron implementar las funcionalidades propuestas mediante la integración de herramientas tecnológicas modernas, como MongoDB para la gestión de datos y React para el frontend. Esto permitió no solo construir una solución funcional y escalable, sino también fortalecer las competencias técnicas en el manejo de bases de datos, APIs y tecnologías frontend; el diseño adaptativo, basado en un sistema de cuadrícula, asegura que los gráficos se ajusten automáticamente a diferentes tamaños de pantalla, mejorando la experiencia de usuario tanto en dispositivos móviles como de escritorio. Esto permite que el dashboard sea accesible y fácil de usar.

El proceso destacó la importancia de la colaboración en equipo, el seguimiento de buenas prácticas de desarrollo y el uso de herramientas de control de versiones como Git, que facilitaron la organización y actualización del repositorio.

Si bien se enfrentaron desafíos técnicos, como la conexión inicial a la base de datos y la configuración de dependencias, estos problemas fueron resueltos mediante la investigación, el análisis y la implementación de soluciones efectivas. Esto subraya la importancia de la adaptabilidad y la resolución de problemas en el desarrollo de software.

En conclusión, este proyecto representa un avance significativo en el aprendizaje de tecnologías y metodologías de desarrollo, y sienta las bases para futuros proyectos más complejos, enfocados en la mejora continua y la integración de nuevas tecnologías.