Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"

Proyecto de desarrollo de aplicaciones web
MacJava Definitivo

CICLO FORMATIVO DE GRADO SUPERIOR
Desarrollo de Aplicaciones Web (IFCS03)

Curso 2023-24

Autor/a/es:
Jaime Lozano Carbonell y Diego Torres Mijarra

Tutor/a:
Jose Luis y Rafa Manjavacas
Departamento de Informática y Comunicaciones
I.E.S. Luis Vives

1 INTRODUCCIÓN

La API MacJava es una solución integral para la administración segura y eficiente de bases de datos, especialmente diseñada para tiendas online. Este proyecto se basa en un enfoque robusto y moderno, utilizando SpringBoot para la capa de back-end y Angular junto con Ionic para la capa de front-end. La combinación de estas tecnologías permite una configuración sencilla y un desarrollo ágil, facilitando la creación y gestión de aplicaciones web y móviles de alto rendimiento.

Para asegurar un flujo de trabajo ordenado y colaborativo, se adopta la metodología GitFlow para la gestión de ramas. Este enfoque organiza las ramas de características en una rama de desarrollo, la cual, tras pasar por un riguroso proceso de pruebas, se fusiona en la rama principal. Esta estructura no solo optimiza la integración y el despliegue continuo, sino que también minimiza los riesgos y mejora la calidad del software.

Con la API MacJava, las tiendas online pueden beneficiarse de una plataforma confiable y escalable, diseñada para manejar grandes volúmenes de datos y proporcionar una experiencia de usuario excepcional.

1.1 OBJETIVO

El objetivo principal de este trabajo de fin de grado es diseñar, desarrollar y documentar una aplicación web y móvil utilizando la API MacJava, basada en las tecnologías SpringBoot, Angular y Ionic. Este proyecto tiene como propósito demostrar la viabilidad y eficacia de esta solución para la administración de bases de datos en entornos de tiendas online.

Específicamente, los objetivos son los siguientes:

1. **Diseñar la arquitectura del sistema**: Definir la estructura y los componentes del sistema, incluyendo la organización de las capas de back-end y front-end, así como la interacción entre ellas.
2. **Desarrollar la aplicación**: Implementar la funcionalidad necesaria para la gestión segura de datos en una tienda online, utilizando las tecnologías SpringBoot, Angular y Ionic.
3. **Realizar pruebas y validaciones**: Verificar el correcto funcionamiento del sistema mediante pruebas unitarias, de integración y funcionales, asegurando su estabilidad y calidad.
4. **Documentar el proceso**: Elaborar una documentación detallada que describa el proceso de desarrollo, desde la planificación inicial hasta la implementación final, incluyendo decisiones de diseño, problemas encontrados y soluciones propuestas.

Al alcanzar estos objetivos, se espera obtener una aplicación robusta y eficiente que pueda servir como ejemplo de buenas prácticas en el desarrollo de aplicaciones web y móviles para tiendas online, utilizando la API MacJava como base tecnológica.

1.2 ALCANCE

El alcance de este proyecto abarca el diseño, desarrollo, pruebas y documentación de una aplicación web y móvil para la gestión de una tienda online, utilizando la API MacJava y las tecnologías asociadas (SpringBoot, Angular y Ionic).

El proyecto incluirá los siguientes elementos:

1. **Diseño de la arquitectura**: Definición de la estructura del sistema, incluyendo la división en capas (back-end y front-end), la configuración de la base de datos y la integración de las tecnologías seleccionadas.
2. **Desarrollo de funcionalidades**: Implementación de las características esenciales de una tienda online, como la gestión de productos, categorías, usuarios y pedidos, asegurando la seguridad y la integridad de los datos.
3. **Interfaz de usuario**: Diseño y desarrollo de interfaces intuitivas y responsivas para la web y dispositivos móviles, utilizando Angular para la parte web y Ionic para la aplicación móvil.
4. **Pruebas**: Realización de pruebas exhaustivas para validar el funcionamiento del sistema en diferentes escenarios, incluyendo pruebas unitarias, de integración y funcionales.
5. **Documentación**: Elaboración de documentación técnica y de usuario, que incluya detalles sobre la arquitectura, la instalación, el uso y la configuración del sistema, así como cualquier otra información relevante para su comprensión y mantenimiento.

El alcance del proyecto no incluirá:

1. Integración con sistemas externos o de terceros, a menos que sea estrictamente necesario para cumplir con los objetivos establecidos.
2. Desarrollo de funcionalidades avanzadas que no sean directamente relevantes para una tienda online básica, como sistemas de recomendación o análisis predictivo.

El proyecto se desarrollará siguiendo un cronograma establecido, con entregas periódicas para revisión y retroalimentación, con el objetivo de cumplir con los objetivos y requisitos establecidos en el tiempo asignado.

1.3 JUSTIFICACIÓN

La creación de una aplicación web y móvil para la gestión de una tienda online utilizando la API MacJava y las tecnologías asociadas (SpringBoot, Angular e Ionic) se justifica por varias razones:

1. **Demanda del mercado**: Con el crecimiento constante del comercio electrónico, existe una creciente demanda de soluciones tecnológicas eficientes para la gestión de tiendas online. Una aplicación bien diseñada y fácil de usar puede mejorar significativamente la experiencia del usuario y aumentar las ventas de la tienda.
2. **Eficiencia operativa**: La automatización de procesos como la gestión de inventario, pedidos y clientes puede ayudar a reducir los costos operativos y mejorar la eficiencia de la tienda online. Una aplicación bien diseñada puede agilizar estos procesos y minimizar los errores humanos.
3. **Accesibilidad**: Una aplicación móvil permite a los usuarios acceder a la tienda desde cualquier lugar y en cualquier momento, lo que amplía el alcance de la tienda y facilita las

compras impulsivas. Además, una interfaz móvil optimizada puede mejorar la experiencia del usuario en dispositivos móviles, aumentando la retención y la fidelidad del cliente.

4. **Seguridad**: La API MacJava ofrece una administración segura de bases de datos, lo que garantiza la integridad y la confidencialidad de los datos de la tienda y de los clientes. Esto es especialmente importante en un entorno donde la seguridad de la información es una preocupación constante.
5. **Escalabilidad**: La arquitectura basada en SpringBoot, Angular y Ionic permite una escalabilidad fácil y rápida, lo que significa que la aplicación puede crecer y adaptarse a medida que la tienda online expande su negocio y aumenta su base de usuarios.

En resumen, este proyecto proporcionará una solución tecnológica moderna y efectiva para la gestión de una tienda online, que mejorará la eficiencia operativa, aumentará la accesibilidad para los usuarios y garantizará la seguridad de la información. Además, servirá como una muestra práctica de las capacidades y ventajas de la API MacJava y las tecnologías asociadas en el desarrollo de aplicaciones web y móviles.

2 IMPLEMENTACIÓN

2.1 ANÁLISIS DE LA APLICACIÓN

La aplicación sigue una arquitectura de tres capas, compuesta por el back-end, el front-end y las bases de datos PostgreSQL y MongoDB. El back-end, desarrollado con Spring Boot, gestiona los endpoints y servicios que manipulan los datos almacenados en las bases de datos. Estas bases de datos se despliegan en contenedores Docker para facilitar su gestión y escalabilidad.

En cuanto a las bases de datos, PostgreSQL se utiliza para almacenar datos estructurados como usuarios, productos y la mayoría de los datos relacionados con la tienda en línea. Por otro lado, MongoDB se utiliza específicamente para almacenar datos relacionados con los pedidos de los usuarios, como detalles del pedido, historial de transacciones, etc. Esta elección se basa en las capacidades específicas de cada base de datos, adaptándose a las necesidades particulares de cada tipo de dato en la aplicación.

El front-end, desarrollado con Angular y Ionic, consume los endpoints proporcionados por el back-end para obtener los datos necesarios para rellenar las vistas de la página web y la aplicación móvil. Se implementan rutas en Angular e Ionic para facilitar la navegación entre las diferentes secciones de la aplicación y mostrar la información correspondiente.

En términos de seguridad, se implementan mecanismos de autenticación y autorización en el back-end utilizando JWT (JSON Web Tokens) para proteger los endpoints y los datos sensibles. Se realizan pruebas de rendimiento para evaluar la velocidad de respuesta de los endpoints y se optimizan las consultas a la base de datos para mejorar el rendimiento de las operaciones.

La aplicación es escalable gracias al despliegue de las bases de datos en contenedores Docker, lo que permite añadir o quitar recursos según sea necesario. Además, un enfoque modular en el diseño del back-end y el front-end facilita la escalabilidad al agregar nuevas funcionalidades o componentes.

En cuanto a la integración, el back-end se integra con las bases de datos PostgreSQL y MongoDB utilizando bibliotecas específicas para cada tecnología. El front-end se integra con el back-end mediante peticiones HTTP a los endpoints proporcionados por éste, utilizando servicios y controladores en Angular e Ionic.

Este análisis proporciona una visión general de la arquitectura, funcionamiento y características de la aplicación, destacando los aspectos clave que la hacen segura, escalable y eficiente.

2.2 DISEÑO

Consulte el Manual de Usuario

2.3 IMPLEMENTACIÓN

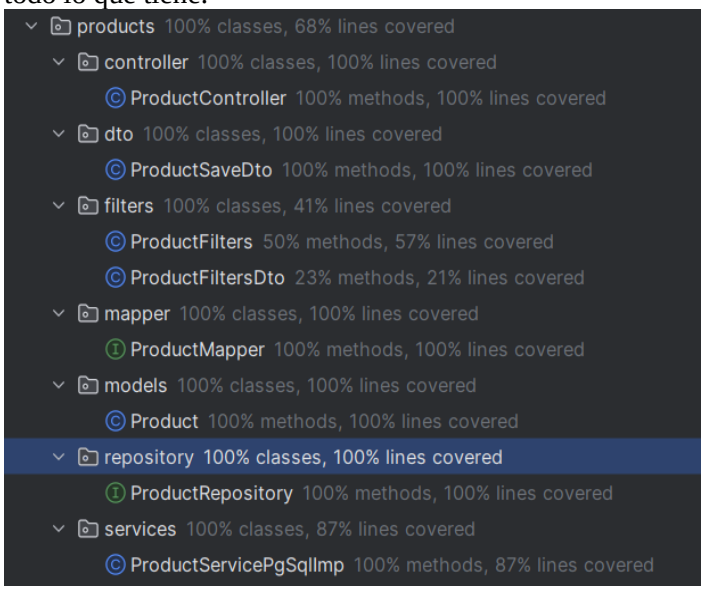
Para explicar esto primero vamos a ir a como lanzamos nuestros Docker asi que enseñare el script que tenemos para crear las imágenes y los contenedores:

Véase: docker-compose.yaml. Anexo 1.

En los Docker de las bbdd insertaremos los siguientes scripts:

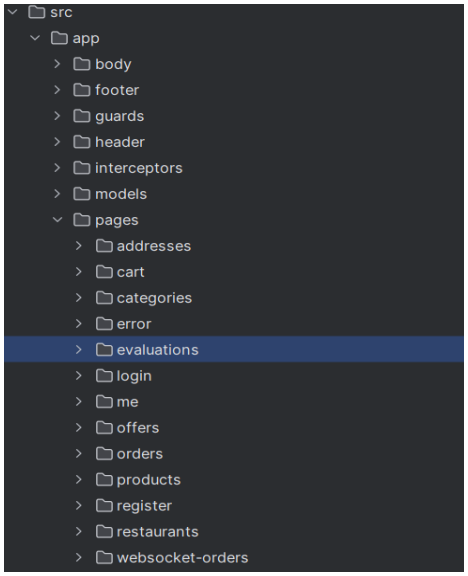
Véase: init.sql. Anexo 2. Véase: init.js. Anexo 3.

Luego voy a enseñar como es la creacion de un endpoint y como lo tenemos organizado en las carpetas con todo lo que tiene:

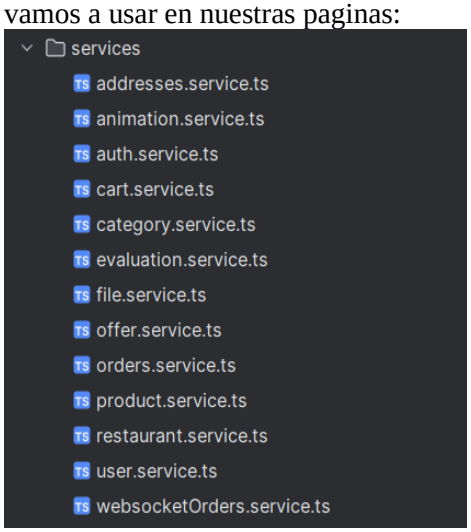


Tenemos un modelo con la información básica de nuestro endpoint un dto y mapper para mostrar en las respuestas lo que queremos, la implementacion del repositorio y el servicio para conectarnos a nuestra base de datos y operar con sus datos y un controlador que actúa de intermedio entre la información que sacamos de la base de datos y utiliza los HTTP para lanzar esos datos y que los pueda recoger nuestro Front-end.

En nuestra parte de Front-end tenemos un header, un footer y un body que se mantienen y en el body es donde vamos cambiando las paginas(vistas) que tenemos en nuestro proyecto:



Y unos servicios que a traves del HTTP se conectan con nuestro back-end y obtienen la información que vamos a usar en nuestras paginas:



2.4 DOCUMENTACIÓN

Consulte Manual Tecnico

3 RESULTADOS Y DISCUSIÓN

En esta sección, se presentan los resultados obtenidos durante el desarrollo de la aplicación, así como las dificultades encontradas y las estrategias empleadas para superarlas.

3.1 Objetivos Alcanzados

Durante el proceso de desarrollo de la aplicación, se lograron con éxito los siguientes objetivos:

- **Desarrollo Integral de Funcionalidades Clave**: Se implementaron todas las funcionalidades esenciales requeridas para una gestión efectiva de la tienda en línea. Esto incluyó la creación de endpoints y servicios en el back-end para la gestión de productos,

categorías, usuarios y pedidos, así como la implementación de la lógica necesaria para la autenticación de usuarios y la seguridad de la aplicación.

- **Integración Exitosa con Bases de Datos**: Se logró una integración sin problemas con las bases de datos PostgreSQL y MongoDB. Esto implicó configurar las conexiones adecuadas entre el back-end y las bases de datos, así como diseñar y ejecutar consultas eficientes para la recuperación y manipulación de datos. La elección de utilizar PostgreSQL para la mayoría de los datos y MongoDB específicamente para los pedidos demostró ser efectiva y proporcionó una estructura de datos óptima para cada tipo de información almacenada.
- **Despliegue en Entorno de Producción**: La aplicación se desplegó satisfactoriamente en un entorno de producción, utilizando contenedores Docker para garantizar la portabilidad y consistencia del entorno. Se realizaron pruebas adicionales en este entorno para verificar la estabilidad y el rendimiento de la aplicación en condiciones reales de uso.

3.2 Dificultades Encontradas y Estrategias de Superación

Durante el desarrollo de la aplicación, surgieron diversas dificultades que requirieron atención y resolución:

- **Complejidad de Integración Tecnológica**: La integración entre el back-end desarrollado en Spring Boot y el front-end desarrollado en Angular e Ionic presentó desafíos iniciales debido a diferencias en la estructura de datos y en los paradigmas de programación utilizados en cada tecnología. Para superar esta dificultad, se realizaron iteraciones adicionales en el diseño de la arquitectura y se implementaron estrategias de comunicación efectivas entre las distintas capas de la aplicación.
- **Optimización de Consultas a Bases de Datos**: Se identificó que algunas consultas a la base de datos, especialmente en MongoDB, no ofrecían el rendimiento esperado debido a la complejidad de las operaciones y la cantidad de datos involucrados. Para resolver este problema, se revisaron y refactorizaron las consultas, se implementaron índices adicionales y se consideró la denormalización de datos en algunos casos específicos para mejorar el rendimiento de la aplicación.
- **Gestión de Errores y Excepciones**: La gestión adecuada de errores y excepciones en el back-end y el front-end resultó ser un aspecto crítico durante el desarrollo de la aplicación. Se implementaron mecanismos robustos de manejo de errores en ambas capas, incluyendo el registro adecuado de excepciones, la comunicación efectiva de errores al usuario y la implementación de estrategias de recuperación en caso de fallos inesperados.

3.3 Reflexiones y Lecciones Aprendidas

A lo largo del proceso de desarrollo de la aplicación, se han obtenido diversas reflexiones y lecciones que pueden ser útiles para futuros proyectos:

- La planificación adecuada y la definición clara de los requisitos desde el principio son fundamentales para el éxito del proyecto.
- La comunicación y colaboración efectiva entre los miembros del equipo son esenciales para superar los desafíos técnicos y mantener un progreso constante.
- La adopción de prácticas de desarrollo ágiles y la realización de pruebas continuas son clave para identificar y solucionar problemas de manera oportuna.
- La flexibilidad y la capacidad de adaptación son importantes para enfrentar los cambios y desafíos inesperados que puedan surgir durante el desarrollo del proyecto.

En resumen, el proceso de desarrollo de la aplicación ha sido una experiencia enriquecedora que ha permitido identificar áreas de mejora y fortalecer las habilidades técnicas y de colaboración del equipo. Los resultados obtenidos son satisfactorios y proporcionan una base sólida para futuras iteraciones y mejoras en la aplicación.

4 TRABAJO FUTURO (OPCIONAL)

1. **Sistema de Gestión de Restaurantes**:
   - Ampliar la aplicación para incluir un sistema completo de gestión de restaurantes, permitiendo a los propietarios de restaurantes gestionar sus menús, horarios de apertura, ubicación, etc.
2. **Ofertas Personalizadas y Cupones de Descuento**:
   - Implementar un sistema de ofertas personalizadas, donde los usuarios puedan recibir descuentos específicos basados en su historial de compras o preferencias. También podrías incluir la posibilidad de generar y aplicar cupones de descuento durante el proceso de compra.
3. **Mejoras en el Carrito de Compra**:
   - Ampliar las funcionalidades del carrito de compra permitiendo a los usuarios guardar los productos para más tarde, compartir el carrito con otros usuarios, o recibir notificaciones sobre cambios en el stock o precios de los productos en el carrito.
4. **Roles de Usuario más Complejos**:
   - Reforzar el sistema de roles de usuario, permitiendo la creación de roles adicionales como "Gerente de Tienda" o "Empleado", con permisos específicos para gestionar inventario, procesar pedidos, etc.
5. **Mejoras en la Seguridad y Autenticación**:
   - Implementar medidas adicionales de seguridad como autenticación de dos factores, cifrado de extremo a extremo para datos sensibles, y seguimiento de actividad de usuario para detectar posibles actividades maliciosas.
6. **Sistema de Notificaciones**:
   - Integrar un sistema de notificaciones para informar a los usuarios sobre el estado de sus pedidos, cambios en el inventario de productos favoritos, ofertas personalizadas, etc.
7. **Optimización del Rendimiento y Escalabilidad**:
   - Realizar mejoras en el rendimiento y la escalabilidad de la aplicación, especialmente en áreas como la gestión de pedidos y la consulta de productos, para garantizar una experiencia de usuario fluida incluso en momentos de alta demanda.
8. **Internacionalización y Localización**:

- Implementar soporte para múltiples idiomas y monedas, permitiendo que la aplicación sea accesible para usuarios de diferentes regiones geográficas.

5 CONCLUSIONES

En conclusión, el desarrollo de este proyecto de aplicación web y móvil para la gestión de una tienda online ha sido un proceso enriquecedor que ha permitido alcanzar importantes logros y superar

diversos desafíos. A lo largo de este proceso, se han cumplido los objetivos establecidos inicialmente, logrando implementar funcionalidades clave, integrar con éxito bases de datos PostgreSQL y MongoDB, y desplegar la aplicación en un entorno de producción.

La aplicación desarrollada proporciona una experiencia completa para los usuarios, permitiéndoles navegar por el catálogo de productos, realizar pedidos, gestionar su perfil y direcciones de envío, entre otras funcionalidades. Además, se ha puesto un énfasis especial en la seguridad, la escalabilidad y el rendimiento de la aplicación, garantizando la confidencialidad de los datos de los usuarios y la eficiencia en el procesamiento de las operaciones.

Durante el proceso de desarrollo, se han identificado y superado diversas dificultades, como la complejidad de integración entre el back-end y el front-end, la optimización de consultas a la base de datos y la gestión adecuada de errores y excepciones. Sin embargo, gracias al trabajo en equipo, la dedicación y la colaboración efectiva, se han encontrado soluciones adecuadas para cada desafío, lo que ha permitido avanzar de manera constante hacia la consecución de los objetivos del proyecto.

En resumen, este proyecto ha sido una experiencia valiosa que ha permitido adquirir nuevos conocimientos y habilidades en el desarrollo de aplicaciones web y móviles, así como en el trabajo en equipo y la resolución de problemas. La aplicación desarrollada proporciona una base sólida para futuras iteraciones y mejoras, y representa un paso importante en la implementación de soluciones tecnológicas innovadoras para la gestión eficiente de tiendas online.

6 BIBLIOGRAFÍA

Baeldung: https://www.baeldung.com/spring-boot
Documentacion Springboot: https://docs.spring.io/spring-boot/index.html
Documentacion Angular: https://docs.angular.lat/docs
Documentacion Ionic: https://ionicframework.com/docs/
Chatgpt: https://chatgpt.com/

ANEXOS

…

I. ANEXO 1: DOCKER-COMPOSE.YAML

```yaml
version: '3.8'

services:
  backend:
    build:
      context: ./backApp
      dockerfile: Dockerfile
    container_name: macjava-api-rest
    restart: always
    env_file: ./backApp/.env
    ports:
      - "3000:3000"
    networks:
      - app-network
    depends_on:
      - postgres-db
      - mongo-db
```

```
  frontend:
    build:
      context:                                                    ./frontApp
      dockerfile:                                                 Dockerfile
    container_name:                                               macjava-frontend
    restart:                                                      always
    ports:
      -                                                           "4200:80"
    networks:
      -                                                           app-network
    depends_on:
      -                                                           backend

  postgres-db:
    container_name:                                               db_postgres
    image:                                                        postgres:12-alpine
    restart:                                                      always
    env_file:                                                     ./.env
    environment:
      POSTGRES_USER:                                              ${DATABASE_USER}
      POSTGRES_PASSWORD:                                          ${DATABASE_PASSWORD}
      POSTGRES_DB:                                                ${POSTGRES_DATABASE}
    ports:
      -                                                           "${POSTGRES_PORT}:5432"
    volumes:
      -                                                           postgres-data-volume:/var/lib/postgresql/data
      -                                                           ./database/init.sql:/docker-entrypoint-initdb.d/init.sql
    networks:
      -                                                           app-network

  mongo-db:
    container_name:                                               db_mongo
    image:                                                        mongo:5.0
    restart:                                                      always
    env_file:                                                     ./.env
    environment:
      MONGO_INITDB_ROOT_USERNAME:                                 ${DATABASE_USER}
      MONGO_INITDB_ROOT_PASSWORD:                                 ${DATABASE_PASSWORD}
      MONGO_INITDB_DATABASE:                                      ${MONGO_DATABASE}
    ports:
      -                                                           "${MONGO_PORT}:27017"
    volumes:
      -                                                           mongo-data-volume:/data/db
      -                                                           ./database/init.js:/docker-entrypoint-initdb.d/init.js:ro
    networks:
      -                                                           app-network

volumes:
  postgres-data-volume:
    name:                                                         postgres-data-volume
  mongo-data-volume:
    name:                                                         mongo-data-volume

networks:
  app-network:
    driver: bridge
```

II. ANEXO 2: INIT.SQL

```
SELECT 'CREATE DATABASE postgres'
WHERE NOT EXISTS (SELECT FROM pg_database WHERE datname = 'postgres');

CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

DROP TABLE IF EXISTS "categories";
DROP TABLE IF EXISTS "evaluation";
DROP TABLE IF EXISTS "products";
DROP TABLE IF EXISTS "restaurants";
DROP TABLE IF EXISTS "offers";
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS "user_roles";

CREATE SEQUENCE products_id_seq INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775807
START 21 CACHE 1;
CREATE SEQUENCE categories_id_seq INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775807
START 9 CACHE 1;
CREATE SEQUENCE evaluation_id_seq INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775807
START 6 CACHE 1;
CREATE SEQUENCE offers_id_seq INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775807 START
3 CACHE 1;

-- Crear la tabla categories
CREATE TABLE "public"."categories"
(
  "id" bigint DEFAULT nextval('categories_id_seq') NOT NULL,
  "name" character varying(255),
  "created_at" timestamp,
  "updated_at" timestamp default CURRENT_TIMESTAMP,
  "deleted_at" timestamp default null,
  CONSTRAINT "categories_pkey" PRIMARY KEY ("id")
) WITH (oids = false);

-- Insertar la tabla categories
INSERT INTO "categories" ("id", "name", "created_at", "updated_at")
VALUES (1, 'Sin Categoria', '2023-01-01', '2023-01-01'),
    (2, 'Platos Principales', '2024-06-03', '2024-06-03'),
    (3, 'Postres', '2024-06-03', '2024-06-03'),
    (4, 'Bebidas', '2024-06-03', '2024-06-03'),
    (5, 'Aperitivos', '2024-06-03', '2024-06-03'),
    (6, 'Sopas y Ensaladas', '2024-06-03', '2024-06-03'),
    (7, 'Especialidades de la Casa', '2024-06-03', '2024-06-03'),
    (8, 'Platos Veganos', '2024-06-03', '2024-06-03')
;

-- Crear la tabla products
CREATE TABLE "public"."products"
(
  "id" bigint DEFAULT nextval('products_id_seq') NOT NULL,
  "name" character varying(255),
  "price" double precision,
  "stock" integer,
  "gluten" boolean,
  "image" bytea Default NULL,
  "image_extension" character varying(10) default null,
  "created_at" timestamp,
  "updated_at" timestamp default CURRENT_TIMESTAMP,
```

```
  "deleted_at" timestamp default null,
  "category_id" bigint,
  CONSTRAINT "products_pkey" PRIMARY KEY ("id"),
  CONSTRAINT "products_fk_categories" FOREIGN KEY ("category_id") REFERENCES "categories" ("id") NOT DEFERRABLE
) WITH (oids = false);
-- Insertar la tabla products
INSERT INTO "products" ("id", "name", "price", "stock", "gluten", "created_at", "updated_at", "category_id") VALUES
    (1, 'Bruschetta Byte', 5.50, 100, true, '2024-06-03', '2024-06-03', 1),  -- Entrantes
    (2, 'Calamares Compilados', 8.75, 50, false, '2024-06-03', '2024-06-03', 1),  -- Entrantes
    (3, 'Sopa de Código', 6.00, 75, false, '2024-06-03', '2024-06-03', 6),  -- Sopas y Ensaladas
    (4, 'Ensalada César en C++', 7.99, 120, false, '2024-06-03', '2024-06-03', 6),  -- Sopas y Ensaladas
    (5, 'Hamburguesa MacJava', 10.50, 100, true, '2024-06-03', '2024-06-03', 2),  -- Platos Principales
    (6, 'Pizza Python', 12.00, 80, false, '2024-06-03', '2024-06-03', 2),  -- Platos Principales
    (7, 'Taco TensorFlow', 9.75, 60, true, '2024-06-03', '2024-06-03', 2),  -- Platos Principales
    (8, 'Sushi SQL', 14.00, 40, false, '2024-06-03', '2024-06-03', 2),  -- Platos Principales
    (9, 'Postre JSON', 5.00, 50, true, '2024-06-03', '2024-06-03', 3),  -- Postres
    (10, 'Tarta de Chocolate HTTP', 6.50, 30, false, '2024-06-03', '2024-06-03', 3),  -- Postres
    (11, 'Batido Binary', 4.25, 70, true, '2024-06-03', '2024-06-03', 4),  -- Bebidas
    (12, 'Café Java', 2.50, 100, false, '2024-06-03', '2024-06-03', 4),  -- Bebidas
    (13, 'Aperitivo AJAX', 3.00, 90, true, '2024-06-03', '2024-06-03', 5),  -- Aperitivos
    (14, 'Tapas TCP/IP', 4.75, 80, false, '2024-06-03', '2024-06-03', 5),  -- Aperitivos
    (15, 'Falafel Frontend', 7.00, 60, true, '2024-06-03', '2024-06-03', 8),  -- Platos Vegetarianos
    (16, 'Burger Backend', 9.00, 50, false, '2024-06-03', '2024-06-03', 8),  -- Platos Vegetarianos
    (17, 'Wrap Web', 8.00, 70, true, '2024-06-03', '2024-06-03', 8),  -- Platos Veganos
    (18, 'Curry Cloud', 11.00, 40, false, '2024-06-03', '2024-06-03', 8),  -- Platos Veganos
    (19, 'Paella PHP', 13.50, 30, true, '2024-06-03', '2024-06-03', 7)  -- Especialidades de la Casa
;


CREATE SEQUENCE restaurants_id_seq INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775807 START 4 CACHE 1;
-- Crear la tabla restaurants
CREATE TABLE "public"."restaurants"
(
  "id" bigint DEFAULT nextval('restaurants_id_seq') NOT NULL,
  "name" character varying(255),
  "address" character varying(255),
  "phone" varchar(9),
  "created_at" timestamp,
  "updated_at" timestamp default CURRENT_TIMESTAMP,
  "deleted_at" timestamp default null,
  CONSTRAINT "restaurants_pkey" PRIMARY KEY ("id")
) WITH (oids = false);

-- Insertar datos en la tabla restaurants
INSERT INTO "restaurants" ("id", "name", "address", "phone", "created_at", "updated_at") VALUES
    (1, 'Restaurante1', 'Calle 1', 111111111, '2023-01-01', '2023-01-01'),
    (2, 'Restaurante2', 'Calle 2', 999999999, '2023-01-01', '2023-01-01'),
    (3, 'Restaurante3', 'Calle 3', 999999999, '2023-01-01', '2023-01-01')
;

-- Crear la tabla ofertas
CREATE TABLE "public"."offers"
(
  "id" bigint DEFAULT nextval('offers_id_seq') NOT NULL,
  "descuento" double precision NOT NULL,
  "fecha_desde" timestamp NOT NULL,
  "fecha_hasta" timestamp NOT NULL,
  "created_at" timestamp,
  "updated_at" timestamp default CURRENT_TIMESTAMP,
```

```
  "deleted_at" timestamp default null,
  "product_id" bigint,
  CONSTRAINT "offers_pkey" PRIMARY KEY ("id"),
  CONSTRAINT "offers_fk_products" FOREIGN KEY ("product_id") REFERENCES "products" ("id") NOT DEFERRABLE
) WITH (oids = false);

-- Insertar la tabla ofertas
INSERT INTO "offers" ("id", "descuento", "fecha_desde", "fecha_hasta", "created_at", "updated_at", "product_id") VALUES (1, 30.0, '2024-01-01', '2024-10-10', '2023-01-01', '2023-01-01', 1),
    (2, 30.0, '2023-01-01', '2023-10-10', '2023-01-01', '2023-01-01', 1)
;

-- Creación de la tabla usuarios
CREATE TABLE "public".users
(
  "id" uuid DEFAULT uuid_generate_v4() NOT NULL,
  "email" character varying(255) NOT NULL,
  "name" character varying(255) NOT NULL,
  "surname" character varying(255) NOT NULL,
  "password" character varying(255) NOT NULL,
  "username" character varying(255) NOT NULL,
  "created_at" timestamp,
  "updated_at" timestamp default CURRENT_TIMESTAMP,
  "deleted_at" timestamp default null,
  CONSTRAINT "usuarios_email_unique_key" UNIQUE ("email"),
  CONSTRAINT "usuarios_pkey" PRIMARY KEY ("id"),
  CONSTRAINT "usuarios_username_unique_key" UNIQUE ("username")
) WITH (oids = false);
-- Creación de la tabla user_roles
CREATE TABLE "public"."user_roles"
(
  "user_id" uuid NOT NULL,
  "roles" character varying(255)
) WITH (oids = false);

-- Insert usuarios y roles
-- Contraseña: Admin1
INSERT INTO users (created_at, id, updated_at, surname, email, name, password, username) VALUES ('2023-11-02 11:43:24.724871', 'a8fd9bb7-62e1-41dc-9f57-338b17c5bcc0', '2023-11-02 11:43:24.724871', 'Admin Admin', 'admin@prueba.net', 'Admin', '$2a$10$Fjs4lFtGRtCgOsLURykTW.IGYfdqsFVXZN1jGhl3PlZAqMTKHK7S6', 'admin');

-- Asignar roles al administrador
INSERT INTO user_roles (user_id, roles) VALUES ('a8fd9bb7-62e1-41dc-9f57-338b17c5bcc0', 'USER');
INSERT INTO user_roles (user_id, roles) VALUES ('a8fd9bb7-62e1-41dc-9f57-338b17c5bcc0', 'ADMIN');
INSERT INTO user_roles (user_id, roles) VALUES ('a8fd9bb7-62e1-41dc-9f57-338b17c5bcc0', 'WORKER');

-- Contraseña: Worker1
INSERT INTO users (created_at, id, updated_at, surname, email, name, password, username) VALUES ('2023-11-02 11:43:24.730431', '455e6e87-5c32-454b-b658-e39330ceefa2', '2023-11-02 11:43:24.730431', 'Worker Worker', 'worker@prueba.net', 'Worker', '$2a$10$XYoJYUTLtLcQX9eytXQ/cuGk.OeUBNMWfqWHzoIFZJOMTgG11/zui', 'worker');

-- Asignar roles al trabajador
INSERT INTO user_roles (user_id, roles)
```

```
VALUES ('455e6e87-5c32-454b-b658-e39330ceefa2', 'WORKER');


-- Contraseña: User1
INSERT INTO users (created_at, id, updated_at, surname, email, name, password, username)
VALUES ('2023-11-02 11:43:24.730431', '24bee18d-920c-4f25-971f-99e91d0aa331', '2023-11-02 11:43:24.730431',
    'User User', 'user@prueba.net', 'User',
'$2a$10$co8cRNxqcwJvCoOUQD9freA/b.FcKGdlI3khs3FxqniJyo3LcpeHe', 'user');

-- Asignar roles al usuario
INSERT INTO user_roles (user_id, roles)
VALUES ('24bee18d-920c-4f25-971f-99e91d0aa331', 'USER');

-- Restricciones de clave externa
ALTER TABLE ONLY "public"."user_roles"
  ADD CONSTRAINT "user_roles_fk_user" FOREIGN KEY (user_id) REFERENCES users (id) NOT
DEFERRABLE;


CREATE SEQUENCE addresses_id_seq INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775807
START 1 CACHE 1;

-- Crear la tabla addresses
CREATE TABLE "public"."addresses"
(
  "id" uuid DEFAULT uuid_generate_v4() NOT NULL,
  "country" character varying(255) NOT NULL DEFAULT 'España',
  "province" character varying(255) NOT NULL DEFAULT 'Madrid',
  "city" character varying(255) NOT NULL,
  "street" character varying(255) NOT NULL,
  "number" character varying(255) NOT NULL,
  "apartment" character varying(255),
  "postal_code" character varying(255) NOT NULL,
  "extra_info" character varying(255),
  "name" character varying(255) NOT NULL,
  "created_at" timestamp,
  "updated_at" timestamp default CURRENT_TIMESTAMP,
  "deleted_at" timestamp default null,
  "user_id" uuid,
  CONSTRAINT "ADDRESSES_pkey" PRIMARY KEY ("id"),
  CONSTRAINT "FK_ADDRESSES_user_id" FOREIGN KEY ("user_id") REFERENCES "public"."users" ("id")
ON DELETE CASCADE
) WITH (oids = false);

-- Insertar datos en la tabla addresses para casas en Leganés, Madrid
INSERT INTO "addresses" ("id", "country", "province", "city", "street", "number", "apartment", "postal_code",
    "extra_info", "name",
    "created_at", "updated_at", "user_id")
VALUES ('37bf14f0-db5d-4f12-bf4a-7b780cfac071', 'España', 'Madrid', 'Leganés', 'Calle de la Luna', '45', 'Bajo B',
    '28915',
    'Cuidado con el Perro', 'Casa de la familia Luna', '2023-01-03 00:00:00', '2023-01-03 00:00:00',
    '24bee18d-920c-4f25-971f-99e91d0aa331'),
   ('64873a5d-5336-45ba-8907-09a8b74da391', 'España', 'Madrid', 'Leganés', 'Avenida de la Universidad', '10',
    'Bajo C',
    '28916', NULL, 'Casa de la familia Universidad', '2023-01-04 00:00:00', '2023-01-04 00:00:00',
    '24bee18d-920c-4f25-971f-99e91d0aa331'),
   (default, 'España', 'Madrid', 'Leganés', 'Calle del Sol', '23', NULL, '28917',
    NULL,
    'Casa de la familia Sol', '2023-01-05 00:00:00', '2023-01-05 00:00:00', '24bee18d-920c-4f25-971f-99e91d0aa331'),
   (default, 'España', 'Madrid', 'Leganés', 'Calle de la Estrella', '78', NULL,
    '28918',
```

```
    NULL, 'Casa de la familia Estrella', '2023-01-06 00:00:00', '2023-01-06 00:00:00',
    '24bee18d-920c-4f25-971f-99e91d0aa331'),
   (default, 'España', 'Madrid', 'Leganés', 'Calle del Universo', '11', 'Primero B',
    '28919',
    NULL, 'Casa de la familia Universo', '2023-01-07 00:00:00', '2023-01-07 00:00:00',
    '24bee18d-920c-4f25-971f-99e91d0aa331')
;

-- Crear la tabla valoraciones
CREATE TABLE "public"."evaluation"
(
  "id" bigint DEFAULT nextval('evaluation_id_seq') NOT NULL,
  "value" Integer NOT NULL,
  "comment" VARCHAR(255),
  "created_at" timestamp,
  "updated_at" timestamp default CURRENT_TIMESTAMP,
  "deleted_at" timestamp default null,
  "user_id" uuid not null,
  "product_id" bigint,
  CONSTRAINT "evaluation_pkey" PRIMARY KEY ("id"),
  CONSTRAINT "evaluation_fk_users" FOREIGN KEY ("user_id") REFERENCES "users" ("id") NOT
DEFERRABLE,
  CONSTRAINT "evaluation_fk_products" FOREIGN KEY ("product_id") REFERENCES "products" ("id") NOT
DEFERRABLE
) WITH (oids = false);

-- Insertar la tabla valoraciones
INSERT INTO "evaluation" ("id", "value", "comment", "created_at", "updated_at", "user_id", "product_id")
VALUES (1, 3, 'Muy buen producto', '2023-01-01', '2023-01-01', '24bee18d-920c-4f25-971f-99e91d0aa331', 1),
    (2, 2, 'Muy buen producto', '2023-01-02', '2023-01-02', '24bee18d-920c-4f25-971f-99e91d0aa331', 2),
    (3, 3, 'Muy buen producto', '2023-01-03', '2023-01-03', '24bee18d-920c-4f25-971f-99e91d0aa331', 3),
    (4, 3, 'Muy buen producto', '2023-01-03', '2023-01-03', '24bee18d-920c-4f25-971f-99e91d0aa331', 3),
    (5, 2, 'Muy buen producto', '2023-01-03', '2023-01-03', '24bee18d-920c-4f25-971f-99e91d0aa331', 3)
;
```

III. ANEXO 2: INIT.JS

```
db.createUser({
  user: 'admin',
  pwd: 'admin',
  roles: [
    {
      role: 'readWrite',
      db: 'mongo',
    },
  ],
});

db = db.getSiblingDB('mongo');

db.createCollection('order');

// Insertamos los datos de la coleccion pedidos
db.order.insertMany([
  {
    userId: BinData(3, "JU8Mko3hviQxowod6Zkflw=="),
    restaurantId: 1,
    addressesId: '37bf14f0-db5d-4f12-bf4a-7b780cfac071',
    orderedProducts: [
      {
```

```
            quantity: 2,
            productId: 1,
            productPrice: 10.50,
            totalPrice: 21.00,
          },
          {
            quantity: 3,
            productId: 3,
            productPrice: 20.00,
            totalPrice: 60.00,
          },
        ],
        totalPrice: 81.00,
        totalQuantityProducts: 5,
        isPaid: false,
        state: 'ACCEPTED',
        createdAt: '2023-10-23T12:57:17.3411925',
        updatedAt: '2023-10-23T12:57:17.3411925',
        deletedAt: null,
      },
      {
        userId: BinData(3, "JU8Mko3hviQxowod6Zkflw=="),
        restaurantId: 2,
        addressesId: '37bf14f0-db5d-4f12-bf4a-7b780cfac071',
        orderedProducts: [
          {
            quantity: 4,
            productId: 1,
            productPrice: 10.5,
            totalPrice: 42.0
          }
        ],
        totalPrice: 42.0,
        totalQuantityProducts: 4,
        isPaid: false,
        state: 'DELIVERED',
        createdAt: '2022-10-23T12:57:17.3411925',
        updatedAt: '2022-10-23T12:57:17.3411925',
        deletedAt: null
      }
]);
```