Harmony Dealer

E-commerce instrumentos musicales

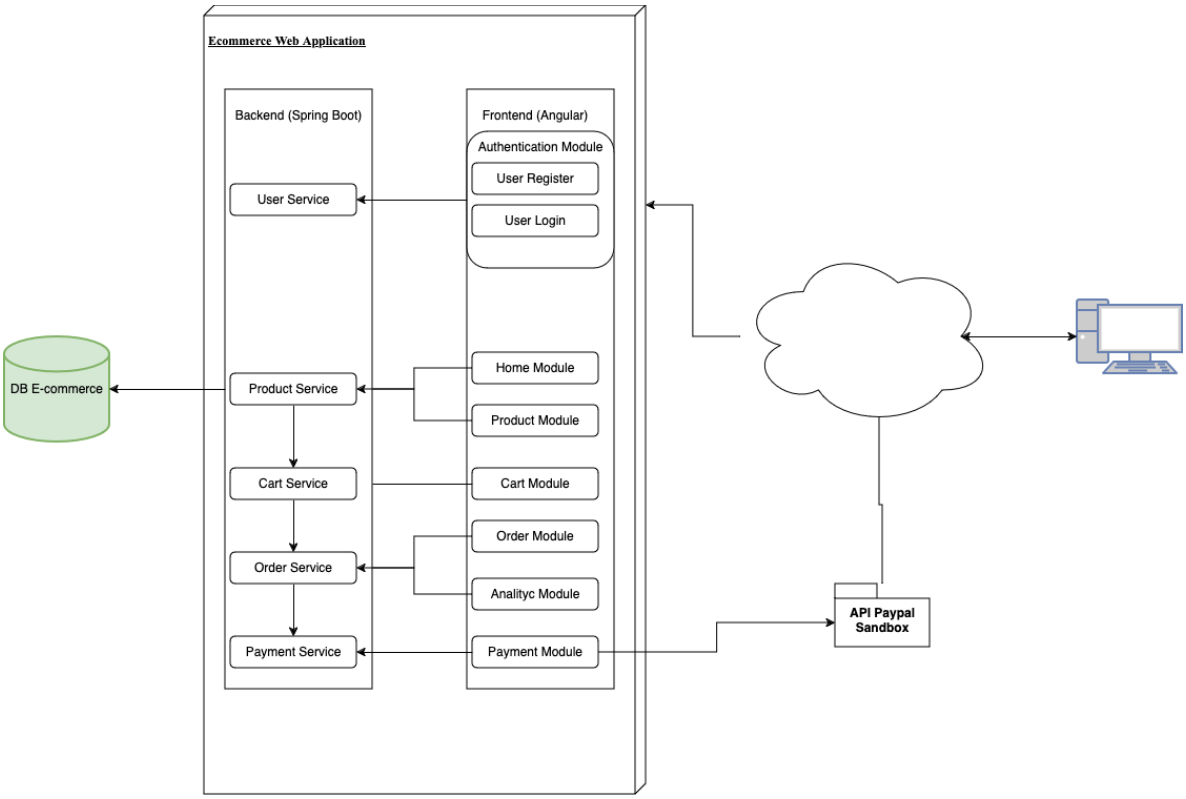
Propósito

Este documento tiene como objetivo guiar a los desarrolladores y administradores en el proceso de despliegue de una aplicación web. La aplicación está compuesta por un frontend desarrollado en Angular y un backend utilizando Spring Boot con PostgreSQL. El propósito principal es asegurar que el despliegue se realice de manera eficiente y cumpla con los parámetros establecidos por el proveedor de servicios.

Arquitectura General:

La aplicación sigue una arquitectura de cliente-servidor. A continuación se presenta un diagrama simplificado que detalla los componentes principales:

- **Frontend (Angular):** Interfaz de usuario que interactúa con los servicios del backend a través de peticiones HTTP.
- **Backend (Spring Boot):** Lógica de negocio y gestión de la base de datos.
- **Base de datos (PostgreSQL):** Almacenamiento de los datos de la aplicación.



a) Frontend (Angular)

- Proporciona la interfaz gráfica para que los usuarios interactúen con la aplicación.
- Consume los endpoints REST expuestos por el backend.

b) Backend (Spring Boot)

- Gestiona la lógica de negocio, el manejo de usuarios, productos, órdenes y pagos.
- Utiliza JPA para interactuar con la base de datos PostgreSQL.
- Exposición de APIs REST que permiten al frontend obtener y modificar datos.

c) Base de Datos (PostgreSQL)

- Almacena la información persistente, como usuarios, productos, órdenes, etc.
- Utiliza Hibernate como ORM para gestionar las operaciones sobre la base de datos.

Condiciones en las que se realiza la entrega:

- El proyecto esta en un repositorio Git para facilitar su distribución.
- Se entregan configurados los perfiles de desarrollo y producción con archivos .properties para diferenciar los entornos.
- Se deben cumplir los requisitos del servidor como tener instalados Java, Node.js y un servidor de base de datos compatible (PostgreSQL).
- Certificados SSL deben estar configurados para las peticiones seguras (HTTPS).

Importación del proyecto en desarrollo:

Clonar el repositorio del proyecto. https://github.com/Mapp9/HarmonyEcom.git

Frontend (Angular)

1. Navegar a la carpeta del frontend.
2. Instalar las dependencias de Node.js
3. Compilar el proyecto para desarrollo

Backend (Spring Boot)

1. Navegar a la carpeta del backend.
2. Configurar el archivo application.properties con las credenciales correctas de la base de datos.
3. Ejecutar Maven para compilar el proyecto
4. Ejecutar la aplicación

Parámetros requeridos:

Frontend

- Angular CLI instalado globalmente:
  npm install -g @angular/cli
- Configuración de entorno para conectar al backend (URL del API):
  export const environment = {
  production: false,
  apiUrl: 'http://localhost:8085/api'
  };

Backend

- **JDK 17** para compilar y ejecutar el backend.
- **PostgreSQL 14** configurado con las siguientes credenciales:
  - Host: localhost
  - Puerto: 5432
  - Usuario: ecommerce_user
  - Contraseña: password
  - Base de datos: ecommerce

Despliegue de la Aplicación Backend

- **Configurar Base de Datos**: Crear una base de datos llamada ecommerce en PostgreSQL e insertar las configuraciones en application.properties.

spring.datasource.url=jdbc:postgresql://localhost:5432/ecommerce
spring.datasource.username=ecommerce_user
spring.datasource.password=password

- **Desplegar el Backend** en un servidor con las herramientas necesarias:
    - Subir el archivo .jar al servidor
    - Configurar un servicio para ejecutar la aplicación con supervisión (usando systemd o similares).
    - Iniciar la aplicación:

        java -jar ecommerce-backend.jar

- **Configurar el Servidor Web** para el frontend (Nginx o Apache).
    - Subir la compilación del frontend (directorio *dist/)* al servidor.
    - Configurar las rutas para servir los archivos estáticos.

- **Configurar el Proxy Inverso** para redirigir las peticiones a la API:
    - Ejemplo en Nginx:

location /api/ { proxy_pass http://localhost:8085/; }

Recomendaciones por el Proveedor del Servicio

- **Seguridad**: Asegurarse de que todas las comunicaciones sean encriptadas utilizando HTTPS.
- **Escalabilidad**: Configurar un balanceador de carga si se espera un alto volumen de tráfico.
- **Monitoreo**: Implementar herramientas de monitoreo como Prometheus o Grafana para supervisar el estado de la aplicación.
- **Backup**: Realizar copias de seguridad periódicas de la base de datos y los archivos estáticos.