Kevin Duvan Mariño Criollo

Grupo:2834922

gestion turistica

Solicitudes de Gestión Turistica

**Propósito del desarrollo**

Gestión Turística es una solución modular compuesta por un backend central que permite gestionar los distintos componentes turísticos (paquetes, hoteles, atracciones, personas, etc.). Su propósito es permitir que administradores y operadores puedan organizar, registrar y consultar toda la información relevante para la oferta de servicios turísticos, de forma segura y escalable.

**Arquitectura general**

* **Backend en Spring Boot**: Aplicación web construida en Java que expone servicios RESTful para gestionar datos turísticos.
* **Base de datos relacional (MySQL)**: Almacena información relacionada con personas, destinos, alojamientos, transporte, etc.
* **Postman**: Se utiliza para probar los endpoints y validar las funcionalidades del backend.
* **Maven**: Maneja la construcción y dependencias del proyecto.

Estructura modular orientada a la escalabilidad y separación de responsabilidades.

**Log a nivel de servidor**

Al ejecutar la aplicación, se generan logs automáticamente en consola y pueden ser redirigidos a archivos. Para mayor control:

logging.file.name=logs/gestion\_turistica.log

**Condiciones en las que se realiza la entrega**

* El backend se ejecuta localmente o en un servidor con **Java 17** y **MySQL** instalado.
* Se espera que el proyecto sea importado en una IDE con soporte para Maven.
* Se deja configurado para apuntar a una base de datos local o remota con credenciales proporcionadas.

**Importación del proyecto en desarrollo**

1. Descargar e instalar **Java JDK 17** o superior.
2. Instalar una IDE como **IntelliJ IDEA**, **Eclipse** o **Visual Studio Code**.
3. Descargar el código fuente.
4. Abrir la IDE y usar la opción Import > Maven > Existing Maven Projects.
5. Seleccionar el archivo pom.xml dentro del directorio:

/backend/gestion\_turistica

Nota: Se requiere conexión a Internet para descargar las dependencias del proyecto.

**Parámetros requeridos**

**#CONFIGURACIÓN DE LA BASE DE DATOS**

spring.datasource.url=jdbc:mysql://localhost:3306/gestion\_turistica

spring.datasource.username=usuario

spring.datasource.password=contraseña

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

server.port=8080

**#CONFIGURACIÓN DEL LOG DE LA APLICACIÓN**

logging.file.name=logs/gestion\_turistica.log

logging.pattern.console=%d{yy-MM-dd HH:mm:ss.SSS} %-5level %logger{36} - %msg%n

**Desplegar aplicación backend**

1. Abre una terminal en el directorio del backend.
2. Ejecuta:

mvn clean package

1. Luego ejecuta:

mvn spring-boot:run

1. Observa la consola para verificar errores.
2. El backend quedará disponible en:

http://localhost:8080

**Desplegar aplicación frontend**

**Consumo del servicio para guardar una entidad (ejemplo)**

**URL:**

POST http://localhost:8080/personas

Content-Type: application/json

**Body JSON:**

{

"tipoDocumento": "CC",

"numeroDocumento": "123456789",

"nombre": "Leonardo",

"apellido": "Aedo Jiménez",

"telefono": "3218363843",

"correo": "leonardo@example.com",

"direccion": "Carrera 14A # 5-46"

}

**Además, se recomienda:**

* Incluir un archivo .json para importar en Postman con los endpoints disponibles.
* Documentar los endpoints usando **Swagger**.

**Recomendaciones por el proveedor del servicio**

* Migrar los servicios a **contenedores Docker** y orquestarlos con **Kubernetes** para una arquitectura desacoplada y escalable.
* Usar **versionado semántico (ej. 1.0.0)**:
  + Primer dígito: cambios incompatibles.
  + Segundo dígito: nuevas funcionalidades.
  + Tercer dígito: parches o correcciones.
* Mantener estándares comunes para facilitar la portabilidad y mantenibilidad del proyecto.