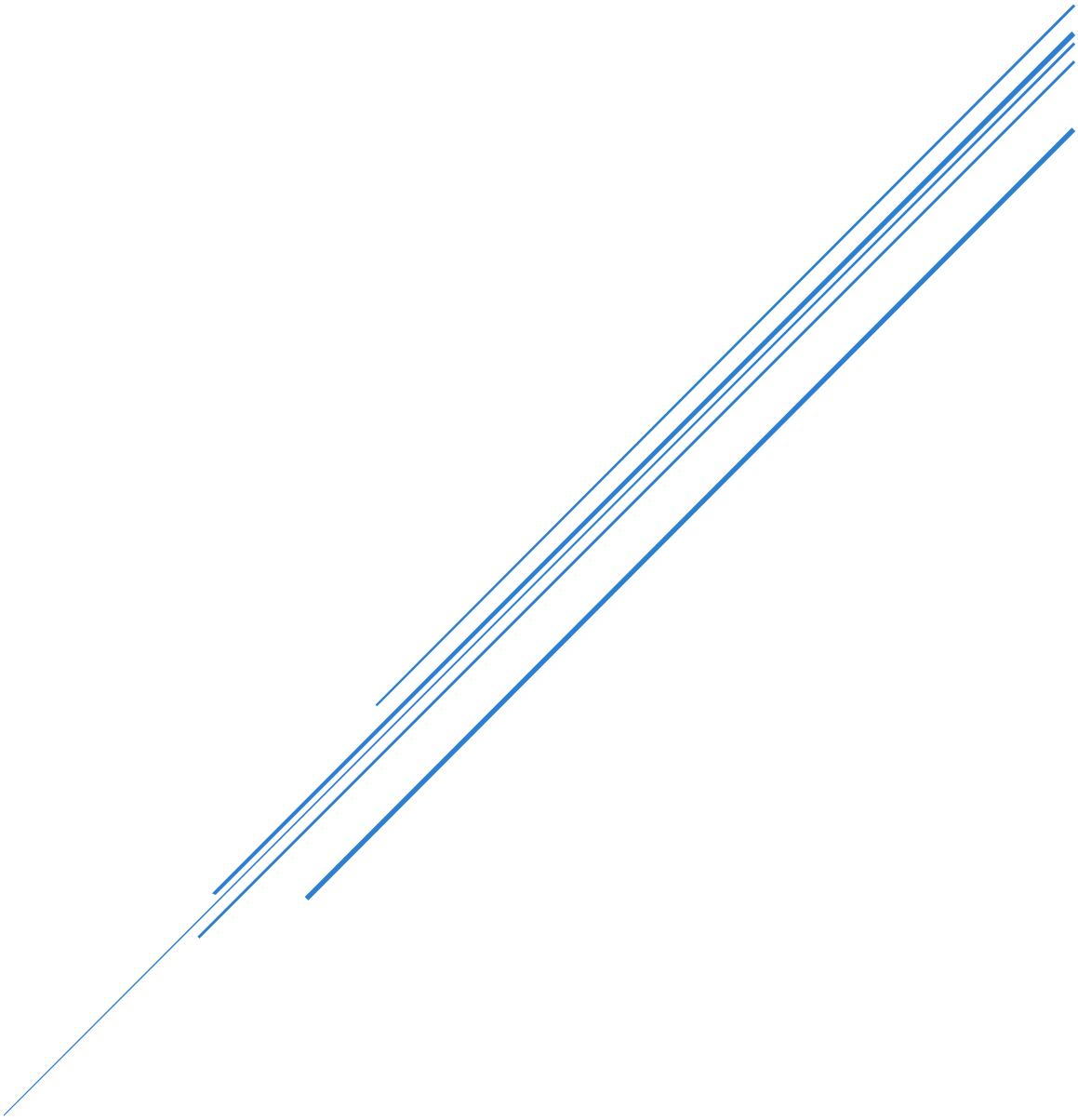
MANUAL TECNICO

De Configuración y Despliegue



Alejandro Flores Navidad

Tabla de contenido

Configuración del Entorno:

Requisitos del Sistema:

- Java JDK (Jakarta 8)
- MySQL Workbench 8.0 (o cualquier otro gestor de bases de datos relacionales)
- Eclipse IDE versión 2024-06
- SpringBoot versión 3.3.2

Instrucciones para clonar el repositorio del proyecto:

1. Ir al repositorio del usuario y descargar el archivo comprimido .ZIP
2. Descomprimirlo en la ruta de preferencia

Configuración de la base de datos:

1. Crear la base de datos con nombre empresa_db.
2. Las tablas proveedores y productos serán creadas por la propia aplicación y sus entidades correspondientes, los datos serán los siguientes:

Producto:

- ID (auto-generado)
- Nombre
- Descripción
- Precio
- Cantidad en stock
- ID del proveedor (relación con la entidad "Proveedor")

Proveedor:

- ID (auto-generado)
- Nombre
- Dirección
- Teléfono
- Email

Configuración del Proyecto en Eclipse:

Importar el proyecto a Eclipse IDE:

- De no haberse descomprimido el archivo .ZIP anteriormente, se debe descomprimir en la ruta deseada, posteriormente, la carpeta principal del proyecto la movemos al icono de nuestro entorno de desarrollo Eclipse IDE y así se abrirá el programa.

Configuración de dependencias en pom.xml:

- Las dependencias en pom.xml ya están configuradas para poder correr sin problemas el programa, solo hace falta limpiar en terminal con mvn clean para evitar algún problema en la ejecución del programa, posteriormente ejecutar el comando mvn install para instalar las dependencias en pom.xml.

Configuración de Propiedades de la base de datos en aplication.properties:

- Las propiedades de la conexión a la base de datos así como las necesarias para el uso de devtools están habilitadas correctamente:

```
spring.datasource.url=jdbc:mysql://localhost:3306/empresa_db?serverTimeZone=UTC
spring.datasource.username=root
spring.datasource.password=12345
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

spring.devtools.livereload.enabled=true

```

- Hace falta cambiar el puerto del localhost al puerto en el que este alojada la base de datos en el lugar donde se ejecutara la aplicación, por defecto es 3306.
- Las propiedades username y password deberán ser modificadas por los datos del usuario de la base de datos donde se trabajara.

Despliegue de la Aplicación:

Pasos para ejecutar la aplicación localmente:

1. Dar click derecho (o izquierdo de ser zurdo) en la carpeta principal del proyecto'
2. Ubicar la pestaña Run As
3. Ubicar Spring Boot App y dar clck

Por defecto el localhosto abrirá en el puerto 8080, con la url "/listar"

Instrucciones para acceder a la aplicación a través del navegador:

1. Abrir el navegador de su preferencia
2. En el cuadro de texto de la url escribimos: localhost:8080/listar
3. Se abrirá la ventana principal del programa

Estructura del Proyecto:

Descripción de la estructura de directorios y archivos del proyecto:

Dentro de la carpeta src/main/java se encuentran todos los packages necesarios de la aplicación, esta misma esta diseñada con la arquitectura MVC, por ende, hay un package para controlador, otro para las interfaces otro para los interfacesService, otro para el modelo y uno final para service.

Todo el proceso empieza desde la creación de los modelos “proveedor” y “Producto” en el package “com.empresa.demo.modelo” y les damos sus respectivas anotaciones para entidades (@Entity) y la creación de la tabla (@Table), cada una de estas entidades tiene los atributos y los tipos de datos especificados en la prueba.

En el package “com.empresa.demo.interfaceService” se procede a implementar las funciones y colecciones que se van a requerir (listar, listarId, guardar y eliminar) y de las que luego se van a usar en otro package.

El el package “com.empresa.demo.services” desarrollamos las colecciones y funciones declaradas en “com.empresa.demo.interfaceService” y es donde ocurrirán las operaciones a realizar.

En la ruta src/main/resources, en la carpeta template, se crea el archivo html para la vista en el navegador.

Usamos un cdn de Bootstrap 5 para darle estilos de forma rápida a la pagina.

Dentro del pom.xml también esta la dependencia de thymeleaf, que nos ayudara con las vistas y la organización de los datos en la vista, en el html, en la etiqueta html escribimos “xmlns:th=*http://www.thymeleaf.org* para habilitarla, y también habilitamos Bootstrap5 con la siguiente cadena:

<**link** href=*https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css*

rel=*"stylesheet"*

integrity=*"sha384EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65Voh*

puuCOmLASjC" crossorigin=*"anonymous"*>

Con las cadenas y links ya implementados, procedemos a crear las etiquetas de tablas correspondientes para crear la vista.

Al final del div con la clase container se ubican los scripts de javascript necesarios para la alerta al tratar de eliminar un producto o proveedor:

<**script** src=*"https://code.jquery.com/jquery-3.7.1.js"* integrity=*"sha256-eKhayi8LEQwp4NKxN+CfCh+3qOVUtJn3QNZ0TciWLP4="*crossorigin=*"anonymous"*></**scrip**>

<**script** src=*"https://cdn.jsdelivr.net/npm/sweetalert2@11"*></**script**>

<**script** src=*"funciones.js"*></**script**>

Las alertas estan ubicadas en la misma ruta que templates, pero en la carpeta llamada statics.

Pruebas:

Las pruebas se realizaron cada que se implementaba una nueva funcionalidad, desde la actualización autmoatica con SpringBootDevTools hasta la conexión a la base de datos local.

No se utilizaron herramientas para debuguear.

No se tuvo que instalar herramientas externas para poder ejecutar el servidor local de otra forma.