MANUAL TÉCNICO

Prueba técnica

Ronaldo García Aparicio

1. Configuración del entorno

1.1 Requisitos del sistema

Java 21
Eclipse IDE
Mysql
Node 21.7.3
Git

1.2 Clonar repositorios

Para el proyecto de spring

git clone https://github.com/ronalcode/api-rest.git

Para el frontend

git clone https://github.com/ronalcode/frontend-prueba.git

1.3 Configuración de base de datos

1. Creamos la base de datos en mysql llamada *empresa_db*
2. Creamos las tablas proveedores y productos

```
CREATE TABLE `proveedores` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `nombre` varchar(150) NOT NULL,
  `direccion` varchar(255) DEFAULT NULL,
  `telefono` varchar(30) NOT NULL,
  `email` varchar(200) NOT NULL,
  PRIMARY KEY (`id`)
)       ENGINE=InnoDB       AUTO_INCREMENT=10       DEFAULT       CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;


CREATE TABLE `productos` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `nombre` varchar(150) NOT NULL,
  `descripcion` varchar(255) NOT NULL,
  `precio` decimal(7,2) NOT NULL,
  `cantidad` int(11) NOT NULL DEFAULT 0,
  `proveedor_id` bigint(20) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `productos_proveedores_FK` (`proveedor_id`),
   CONSTRAINT `productos_proveedores_FK` FOREIGN KEY (`proveedor_id`) REFERENCES
`proveedores` (`id`) ON DELETE SET NULL
)       ENGINE=InnoDB       AUTO_INCREMENT=18       DEFAULT       CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

2. Configuración del proyecto en eclipse

2.1 Abrir el proyecto en eclipse

Una vez clonado el repositorio, abrimos el IDE eclipse y buscamos nuestro proyecto.
File>Open Projects from File System

2.2 Dependencias del proyecto

Las principales depencias del proyecto son:
`spring-boot-starter-data-jpa`
`spring-boot-starter-web`
`mysql-connector-j`
`lombok`
`spring-boot-starter-validation`

2.3 Configuración de propiedades de base de datos

Nombre de la aplicación
```
spring.application.name=api-rest
```

Conector a utilizar y la base de datos a la que se va a conectar
```
spring.datasource.url=jdbc:mysql://localhost:3306/empresa_db
```

Usuario de la base de datos
```
spring.datasource.username=root
```

Contraseña de la base de datos
```
spring.datasource.password=
```

Ver la salida de las consultas en la terminal (no es necesario pero es de ayuda)
```
spring.jpa.show-sql=true
```

Para gestionar los cambios en la base de datos, en este caso no es necesario realizar ninguna acción porque se crea la base de datos de manera manual.
```
spring.jpa.hibernate.ddl-auto=none
```

3. Despliegue de la aplicación

3.1 Lanzar la aplicación

Para lanzar la aplicación solamente es necesario Ejecutar la aplicación desde la barra de herramientas de eclipse.

3.2 Acceder desde el navegador

Para lanzar el frontend es necesario realizar los siguientes pasos.

1. Cambiarse al directorio donde proyecto clonado
2. Ejecutar **npm install**
3. Correr el servidor de desarrollo **npm run dev**
4. Ingresar a la url que indica en la terminal en este caso *http://localhost:5173/* desde el navegador.

4. Estructura del proyecto

-api-rest
  -src/main/java/   #Van todos los paquetes y clases creados para el proyecto
  -src/main/resources/ # Se coloca el archivo de configuración de propiedades de la base de datos

  -pom.xlm # Archivo de configuración de dependencias

4.2 Paquetes y clases

```
package com.api.api_rest;
```

Paquete donde se coloca la clase principal que va a ejecutar el proyecto.
Contiene la clase `ApiRestApplication`

```
package com.api.api_rest.controllers;
```

Paquete donde se colocan todos los controladores que dependen del proyecto.

Contiene las clases de `ProductoController` y `ProveedorController`
ProductoController contiene las rutas para las solicitudes y los métodos que se van a ejecutar en consecuencia para los Productos. Lo mismo para la clase ProveedorController.

```
package com.api.api_rest.cors;
```

Paquete donde se coloca la clase de configuración para las peticiones CORS.
Contiene la clase `WebConfig` la cual se encarga de permitir las peticiones desde el fronted, en este caso esta especificado que sea desde http://localhost:7173/ y que permita las solicitudes GET, POST, PUT, DELETE.

```
package com.api.api_rest.exceptions;
```

Paquete donde se encuentra una excepción personaliza la cual se engarga de lanzar un mensaje cuando no se encuentra algún recurso. Especificamente cuando no se encuentra un registro en la base de datos.

```
package com.api.api_rest.models;
```

Paquete que contiene las entidades o modelos. Contiene las clases Produtos y Proveedores. Cada clase contiene sus respectivas propiedades, así como sus métodos getters y setters, aunque no se especifica tal cual ya que se usa la dependencia de lombok.

```
package com.api.api_rest.repositories;
```

Contiene las interfaces del proyecto. Cada entidad cuenta con su propia interface, en este caso son las clases IProductosRepository e IProveedoresRepository.
Ambas interfaces extienden de `JpaRepository` que nos proporciona una implementación lista para realizar operaciones CRUD.

```
package com.api.api_rest.services;
```

Paquete donde se colocan los servicios utilizados para el proyecto. Se colocan las clases que nos ayudan a gestionar la lógica del proyecto, sirviendo de intermediaros entre el repositorio y el controlador. En cada clase se colocan las métodos básicos del CRUD, cada uno hace uso del repositorio para las operaciones en base de datos (insertar, leer, actualizar, eliminar).