Instituto Politécnico Nacional - IPN

Escuela Superior de Cómputo - ESCOM

PRÁCTICA – Categorías y Productos

Alumno/Boleta: José Carlos Espinosa Martínez / 2023630045

Maestro: José Asunción Enríquez Zárate

Materia: Tecnologías para el Desarrollo de Aplicaciones Web

Carrera: Ingeniería en Inteligencia Artificial / 4to Semestre

Grupo: 4BM1

Link al repo: ***GitHub****(Ctrl+Click :D)*

31/06/2024

Índice

Objetivo

Desarrollar el BackEnd en Spring Boot para gestionar las operaciones relacionadas con Alumnos, el FrontEnd en Angular utilizando Bootstrap para gestionar las operaciones relacionadas con Categorías y Productos, e integrarás el BackEnd con el FrontEnd.

Introducción

En el presente reporte se describe el desarrollo de una aplicación web cuyo objetivo es la gestión de categorías y productos, en el contexto del ejercicio práctico de desarrollo web. Este ejercicio ha sido diseñado para que los estudiantes adquieran y consoliden habilidades en la creación de aplicaciones web completas, integrando tanto el Backend como el Frontend, utilizando herramientas y tecnologías modernas.

En la sección del ***Backend*** de la aplicación ha sido desarrollado utilizando Spring Boot, un marco de trabajo basado en Java. Spring Boot es conocido por simplificar el desarrollo de aplicaciones empresariales, proporcionando una configuración automática y una estructura predeterminada que permite a los desarrolladores concentrarse en la lógica de negocio sin preocuparse demasiado por la configuración subyacente. Este marco facilita la creación de servicios web RESTful, que son esenciales para la comunicación entre el servidor y el cliente en aplicaciones web modernas.

La parte del ***Frontend*** ha sido desarrollado utilizando Angular, un framework de desarrollo de aplicaciones web basado en TypeScript, creado por Google. Angular permite construir interfaces de usuario dinámicas y receptivas, ofreciendo una arquitectura robusta para manejar el flujo de datos y la sincronización con el

Backend. Para la estilización y el diseño visual de la aplicación, se ha utilizado Bootstrap, una biblioteca de código abierto que proporciona estilos CSS y componentes JavaScript predefinidos, facilitando la creación de interfaces atractivas y consistentes.

El propósito principal de este ejercicio es que los estudiantes puedan implementar y conectar de manera efectiva ambas partes de la aplicación, logrando una integración fluida entre el Backend y el Frontend. Este proceso de integración es fundamental en el desarrollo de aplicaciones web, ya que asegura que el cliente (Frontend) y el servidor (Backend) puedan comunicarse y funcionar como una unidad cohesiva.

A lo largo del reporte, se detallarán los pasos seguidos en el desarrollo de la aplicación, desde la configuración inicial del entorno de desarrollo, pasando por la creación de los modelos de datos, controladores y servicios en Spring Boot, hasta la implementación de los componentes de usuario, servicios y rutas en Angular. Además, se discutirán los desafíos encontrados durante el desarrollo y las soluciones implementadas para superarlos.

Marco teórico

Las operaciones clave en el desarrollo de este proyecto son la gestión de Alumnos, la administración de Categorías y Productos, y la integración entre el BackEnd y el FrontEnd. Estas operaciones se basan en conceptos fundamentales de desarrollo de software, arquitectura de sistemas y diseño de interfaces. A continuación, se describen de manera más específica estas operaciones:

Introducción a la Arquitectura de Software: La arquitectura de software es la base sobre la cual se construye cualquier aplicación. Esta define la estructura general, los componentes principales y las relaciones entre ellos. En el contexto de aplicaciones web modernas, una arquitectura comúnmente adoptada es la arquitectura de tres capas, que se compone de la capa de presentación (FrontEnd), la capa de lógica de negocio (BackEnd) y la capa de datos (Base de Datos).

Spring Boot: Spring Boot es un framework basado en Java que facilita la creación de aplicaciones independientes y productivas basadas en Spring. Proporciona una manera rápida de configurar y lanzar proyectos con configuraciones predeterminadas y un conjunto de dependencias listas para usar.

- Gestión de Dependencias: Spring Boot simplifica la gestión de dependencias con su gestor de dependencias y sus starters.

- Configuración Automática: Ofrece configuración automática para muchas bibliotecas y frameworks populares.

- Microservicios: Facilita la creación de aplicaciones en arquitecturas de microservicios.

Spring Boot es ideal para el desarrollo de BackEnds robustos y escalables, ofreciendo herramientas y características que soportan el desarrollo ágil y la gestión eficiente de operaciones de negocio.

Angular: Angular es un framework de desarrollo de aplicaciones web desarrollado y mantenido por Google. Es conocido por su capacidad para crear aplicaciones de una sola página (SPA) y proporciona una arquitectura modular que facilita el mantenimiento y la escalabilidad.

- Componentes: Angular utiliza componentes como bloques de construcción reutilizables.

- Data Binding: Permite la sincronización automática de datos entre el modelo y la vista.

- Directivas y Servicios: Ofrece directivas para manipular el DOM y servicios para gestionar la lógica de negocio.

Angular, combinado con Bootstrap, permite desarrollar interfaces de usuario atractivas y responsivas que mejoran la experiencia del usuario.

Bootstrap: Bootstrap es una biblioteca de CSS y JavaScript para el desarrollo de interfaces de usuario responsivas. Proporciona una colección de componentes y utilidades que facilitan el diseño de interfaces atractivas y coherentes.

- Grid System: Ofrece un sistema de rejilla para crear diseños responsivos.

- Componentes de UI: Incluye una variedad de componentes predefinidos como botones, formularios y modales.

- Customización: Permite personalizar los estilos para ajustarse a las necesidades del proyecto.

Gestión de Categorías y Productos: La gestión de Categorías y Productos es una parte fundamental de muchas aplicaciones web, particularmente en entornos de comercio electrónico. Esta funcionalidad implica la capacidad de crear, leer, actualizar y eliminar (CRUD) categorías y productos.

- Categorías: Permiten organizar y agrupar productos similares, facilitando la navegación y búsqueda para los usuarios.

- Productos: Incluyen detalles como nombre, descripción, precio, e imágenes,

proporcionando la información necesaria para que los usuarios tomen decisiones de compra informadas.

La implementación de estas funcionalidades en el FrontEnd con Angular y Bootstrap asegura una interfaz de usuario intuitiva y atractiva.

Integración FrontEnd-BackEnd: La integración entre el FrontEnd y el BackEnd es crucial para el funcionamiento de aplicaciones web modernas. Esta integración se realiza generalmente a través de APIs RESTful.

- RESTful APIs: Proporcionan una manera estándar de comunicar entre el cliente (FrontEnd) y el servidor (BackEnd) utilizando HTTP.
- JSON: Es el formato de datos comúnmente utilizado para la transferencia de datos entre el servidor y el cliente.
- Seguridad y Autenticación: Es esencial implementar medidas de seguridad para proteger los datos y asegurar la autenticación de los usuarios.

Esta integración permite una comunicación fluida y eficiente entre el FrontEnd desarrollado en Angular y el BackEnd en Spring Boot, asegurando que los datos de Alumnos, Categorías y Productos se gestionen de manera coherente y segura.

Desarrollo

0. Bases de datos | MySQL

Antes de iniciar con el backend se modelo y definió una base de datos utilizando MySQL, para que el mismo pueda acceder y modificarlas según las peticiones del servidor, las tablas que se utilizaron son:

De las cuales, la definición de las 2 más importantes es...

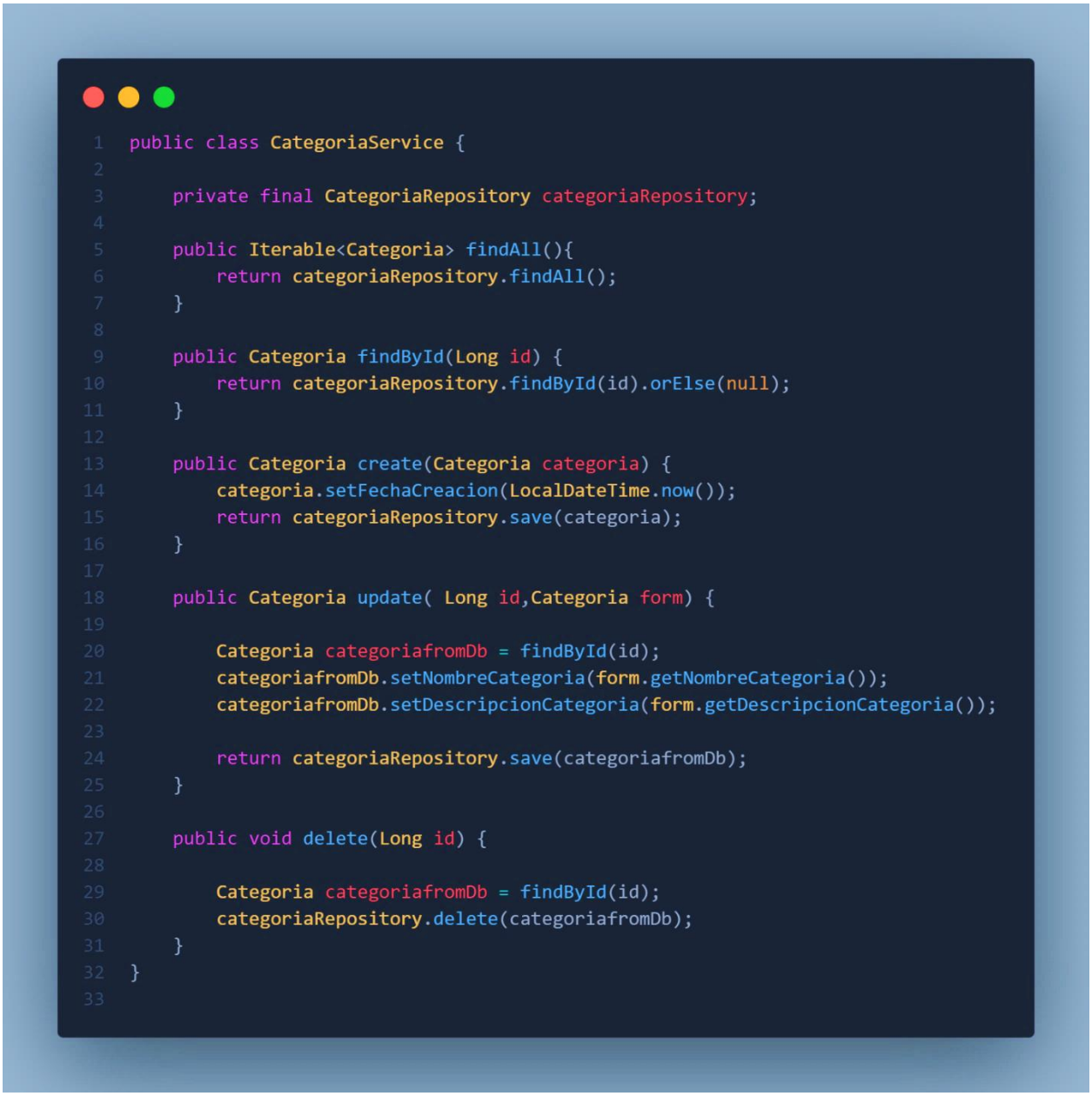
```
mysql> describe categoria;
+----------------------+--------------+------+-----+---------+-------+
| Field                | Type         | Null | Key | Default | Extra |
+----------------------+--------------+------+-----+---------+-------+
| idCategoria          | bigint       | NO   | PRI | NULL    |       |
| descripcionCategoria | varchar(255) | YES  |     | NULL    |       |
| fechaCreacion        | datetime(6)  | YES  |     | NULL    |       |
| nombreCategoria      | varchar(255) | YES  |     | NULL    |       |
+----------------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> describe producto;
+----------------------+--------------+------+-----+---------+-------+
| Field                | Type         | Null | Key | Default | Extra |
+----------------------+--------------+------+-----+---------+-------+
| idProducto           | bigint       | NO   | PRI | NULL    |       |
| descripcionProducto  | varchar(255) | YES  |     | NULL    |       |
| existenciaProducto   | bigint       | YES  |     | NULL    |       |
| fechaCreacion        | datetime(6)  | YES  |     | NULL    |       |
| idCategoria          | bigint       | YES  |     | NULL    |       |
| nombreProducto       | varchar(255) | YES  |     | NULL    |       |
| precioProducto       | double       | NO   |     | NULL    |       |
+----------------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

1. Servicios en el BackEnd con Spring Boot

En el BackEnd, los servicios son responsables de contener la lógica de negocio y de interactuar con la capa de datos. Utilizando Spring Boot, se crean servicios para manejar las operaciones CRUD de Categorías y Productos.

Servicio de Categoría (CategoriaService)

El servicio de Categoría maneja todas las operaciones relacionadas con las categorías. Esto incluye crear nuevas categorías, obtener listas de categorías, actualizar detalles de categorías y eliminar categorías.



```java
public class CategoriaService {

    private final CategoriaRepository categoriaRepository;

    public Iterable<Categoria> findAll(){
        return categoriaRepository.findAll();
    }

    public Categoria findById(Long id) {
        return categoriaRepository.findById(id).orElse(null);
    }

    public Categoria create(Categoria categoria) {
        categoria.setFechaCreacion(LocalDateTime.now());
        return categoriaRepository.save(categoria);
    }

    public Categoria update( Long id,Categoria form) {

        Categoria categoriafromDb = findById(id);
        categoriafromDb.setNombreCategoria(form.getNombreCategoria());
        categoriafromDb.setDescripcionCategoria(form.getDescripcionCategoria());

        return categoriaRepository.save(categoriafromDb);
    }

    public void delete(Long id) {

        Categoria categoriafromDb = findById(id);
        categoriaRepository.delete(categoriafromDb);
    }
}
```

Servicio de Producto (ProductoService)

El servicio de Producto maneja todas las operaciones relacionadas con los productos. Incluye funcionalidades para crear, leer, actualizar y eliminar productos.

```java
public class ProductoService {

    private final ProductoRepository productoRepository;

    public Iterable<Producto> findAll(){
        return productoRepository.findAll();
    }

    public Producto findById(Long id) {
        return productoRepository.findById(id).orElse(null);
    }

    public Producto create(Producto producto) {
        producto.setFechaCreacion(LocalDateTime.now());
        return productoRepository.save(producto);
    }

    public Producto update(Long id, Producto form) {
        Producto productoFromDb = findById(id);
        productoFromDb.setNombreProducto(form.getNombreProducto());
        productoFromDb.setDescripcionProducto(form.getDescripcionProducto());
        productoFromDb.setPrecioProducto(form.getPrecioProducto());
        productoFromDb.setExistenciaProducto(form.getExistenciaProducto());
        productoFromDb.setIdCategoria(form.getIdCategoria());

        return productoRepository.save(productoFromDb);
    }

    public void delete(Long id) {
        Producto productoFromDb = findById(id);
        productoRepository.delete(productoFromDb);
    }

    public Iterable<Producto> findProductosByCategoria(Long idCategoria) {
        return productoRepository.findByidCategoria(idCategoria);
    }

}
```

2. Controladores en el BackEnd con Spring Boot

Los controladores en el BackEnd son responsables de recibir las solicitudes HTTP y delegarlas a los servicios correspondientes. Utilizando Spring Boot, se crean controladores para gestionar las solicitudes relacionadas con Categorías y Productos.

Controlador de Categoría (CategoriaController)

```java
public class CategoriaController {

    private final CategoriaService categoriaService ;

    @GetMapping
    public Iterable<Categoria> list(){
        return categoriaService.findAll();
    }

    @GetMapping("{id}")
    public Categoria get(@PathVariable Long id) {
        return categoriaService.findById(id);
    }

    @ResponseStatus(HttpStatus.CREATED)
    @PostMapping
    public Categoria create(@RequestBody Categoria categoria) {
        return categoriaService.create(categoria);
    }

    @PutMapping("{id}")
    public Categoria update(@PathVariable Long id, @RequestBody Categoria form) {
        return categoriaService.update(id, form);
    }

    @ResponseStatus(HttpStatus.NO_CONTENT)
    @DeleteMapping("{id}")
    public void delete(@PathVariable Long id) {
       categoriaService.delete(id);
    }

}
```

Controlador de Producto (ProductoController)

```java
public class ProductoController {

    private final ProductoService productoService;

    @GetMapping
    public Iterable<Producto> list() {
        return productoService.findAll();
    }

    @GetMapping("{id}")
    public Producto get(@PathVariable Long id) {
        return productoService.findById(id);
    }

    @ResponseStatus(HttpStatus.CREATED)
    @PostMapping
    public Producto create(@RequestBody Producto producto) {
        return productoService.create(producto);
    }

    @PutMapping("{id}")
    public Producto update(@PathVariable Long id, @RequestBody Producto form) {
        return productoService.update(id, form);
    }

    @ResponseStatus(HttpStatus.NO_CONTENT)
    @DeleteMapping("{id}")
    public void delete(@PathVariable Long id) {
        productoService.delete(id);
    }

    @GetMapping("/categoria/{nombreCategoria}")
    public Iterable<Producto> findByCategoria(@PathVariable Long idCategoria) {
        return productoService.findProductosByCategoria(idCategoria);
    }
}
```

3. Servicios y Componentes en el FrontEnd con Angular

En el FrontEnd, los servicios son responsables de realizar las solicitudes HTTP al BackEnd y proporcionar los datos a los componentes que manejan la visualización.

Servicio de Categoría (categoria.service)

```
@Injectable({
  providedIn: 'root'
})
export class CategoriaService {
  private http=inject(HttpClient);

  list(){
    return this.http.get<Categoria[]>('http://localhost:8080/api/categorias');
  }

  get(id:number){
    return this.http.get<Categoria>('http://localhost:8080/api/categorias/'+id);
  }

  create(categoria: Categoria){
    return this.http.post<Categoria>('http://localhost:8080/api/categorias', categoria);
  }

  update(id:number,categoria: Categoria){
    return this.http.put<Categoria>('http://localhost:8080/api/categorias/'+id, categoria);
  }

  delete(id: number){
    return this.http.delete<void>('http://localhost:8080/api/categorias/'+id);
  }
}
```

Servicio de Producto (product.service)

```
@Injectable({
  providedIn: 'root'
})
export class ProductoService {
  private http=inject(HttpClient);

  list(){
    return this.http.get<Producto[]>('http://localhost:8080/api/productos');
  }

  get(id:number){
    return this.http.get<Producto>('http://localhost:8080/api/productos/'+id);
  }

  create(producto: Producto){
    return this.http.post<Producto>('http://localhost:8080/api/productos', producto);
  }

  update(id:number,producto: Producto){
    return this.http.put<Producto>('http://localhost:8080/api/productos/'+id, producto);
  }

  delete(id: number){
    return this.http.delete<void>('http://localhost:8080/api/productos/'+id);
  }

  findByCategoria(idCategoria: number) {
    return this.http.get<Producto>('http://localhost:8080/api/productos/categoria/' + idCategoria);
  }
}
```

Componentes en Angular

Los componentes de Angular manejan la presentación de los datos y las interacciones del usuario. Se crean componentes para listar, crear, editar y eliminar categorías y productos.

Componente de Listado de Categorías (categoria-list.component)

```typescript
@Component({
  selector: 'app-categoria-list',
  standalone: true,
  imports: [DatePipe,RouterModule],
  templateUrl: './categoria-list.component.html',
  styleUrl: './categoria-list.component.css'
})
export default class CategoriaListComponent implements OnInit{
  private CategoriaService = inject(CategoriaService);

  categorias:Categoria[] = [];

  ngOnInit(): void {
    this.loadAll();
  }

  loadAll(){
    this.CategoriaService.list()
    .subscribe(categorias => {
      this.categorias = categorias;
    });
  }

  deleteCategoria(categoria: Categoria){
    this.CategoriaService.delete(categoria.idCategoria)
    .subscribe(()=> {
      this.loadAll();
    })
  }

  buscarPorId(id: string): void {
    if (id.trim() !== '') {
      this.CategoriaService.get(Number(id)).subscribe((categoria) => {
        this.categorias = categoria ? [categoria] : [];
      });
    } else {
      this.loadAll();
    }
  }

}
```

Componente de Listado de Productos (producto-list.component)

```typescript
@Component({
  selector: 'app-producto-list',
  standalone: true,
  imports: [DatePipe,RouterModule],
  templateUrl: './producto-list.component.html',
  styleUrl: './producto-list.component.css'
})
export default class ProductoListComponent implements OnInit{
  private ProductoService = inject(ProductoService);

  productos:Producto[] = [];

  ngOnInit(): void {
    this.loadAll();
  }

  loadAll(){
    this.ProductoService.list()
    .subscribe(productos => {
      this.productos = productos;
    });
  }

  deleteProductos(producto: Producto){
    this.ProductoService.delete(producto.idProducto)
    .subscribe(()=> {
      this.loadAll();
    })
  }

  buscarPorId(id: string): void {
    if (id.trim() !== '') {
      this.ProductoService.get(Number(id)).subscribe((producto) => {
        this.productos = producto ? [producto] : [];
      });
    } else {
      this.loadAll();
    }
  }

  buscarPorCategoria(id: string): void {
    if (id.trim() !== '') {
      this.ProductoService.findByCategoria(Number(id)).subscribe((producto) => {
        this.productos = producto ? [producto]:[];
      });
    } else {
      this.loadAll();
    }
  }

}

```

Tablas en la base de datos

```
mysql> use inventario;
Database changed
mysql> show tables;
+----------------------+
| Tables_in_inventario |
+----------------------+
| categoria            |
| categoria_seq        |
| producto             |
| producto_seq         |
+----------------------+
4 rows in set (0.01 sec)

mysql>
```

```
mysql> SELECT * FROM categoria;
+-------------+-------------------------+-------------------------+-----------------------------------+
| idCategoria | descripcionCategoria    | fechaCreacion           | nombreCategoria                   |
+-------------+-------------------------+-------------------------+-----------------------------------+
|           1 | categoria prueba :D     | 2024-07-01 22:56:46.967601 | Ola2, la venganza del editar categoria |
|           2 | Celulares y cosas asi xd | 2024-07-01 22:58:01.241586 | Dispositivos moviles              |
|           4 | alimentos :D            | 2024-07-01 23:07:41.260790 | Comida                            |
+-------------+-------------------------+-------------------------+-----------------------------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM producto;
+------------+---------------------+--------------------+-------------------------+-------------+----------------+----------------+
| idProducto | descripcionProducto | existenciaProducto | fechaCreacion           | idCategoria | nombreProducto | precioProducto |
+------------+---------------------+--------------------+-------------------------+-------------+----------------+----------------+
|          2 | ave muerta          |                102 | 2024-07-01 23:13:14.639515 |           4 | Pollo          |             99 |
|          3 | Celular de samsu    |                  2 | 2024-07-01 23:13:46.289732 |           1 | S24            |          16599 |
|          4 | Comida rapida       |                  2 | 2024-07-01 23:14:13.081205 |           4 | Pizza          |              1 |
+------------+---------------------+--------------------+-------------------------+-------------+----------------+----------------+
3 rows in set (0.00 sec)
```

Definición de cada tabla

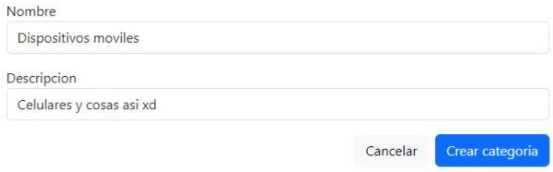
```
mysql> describe categoria;
+----------------------+--------------+------+-----+---------+-------+
| Field                | Type         | Null | Key | Default | Extra |
+----------------------+--------------+------+-----+---------+-------+
| idCategoria          | bigint       | NO   | PRI | NULL    |       |
| descripcionCategoria | varchar(255) | YES  |     | NULL    |       |
| fechaCreacion        | datetime(6)  | YES  |     | NULL    |       |
| nombreCategoria      | varchar(255) | YES  |     | NULL    |       |
+----------------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> describe producto;
+----------------------+--------------+------+-----+---------+-------+
| Field                | Type         | Null | Key | Default | Extra |
+----------------------+--------------+------+-----+---------+-------+
| idProducto           | bigint       | NO   | PRI | NULL    |       |
| descripcionProducto  | varchar(255) | YES  |     | NULL    |       |
| existenciaProducto   | bigint       | YES  |     | NULL    |       |
| fechaCreacion        | datetime(6)  | YES  |     | NULL    |       |
| idCategoria          | bigint       | YES  |     | NULL    |       |
| nombreProducto       | varchar(255) | YES  |     | NULL    |       |
| precioProducto       | double       | NO   |     | NULL    |       |
+----------------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Demostración de Resultados

Categorias:

Imagen 1

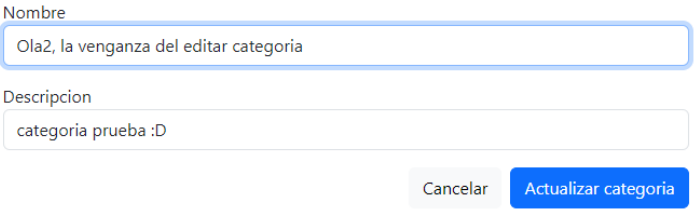
Nueva categoria

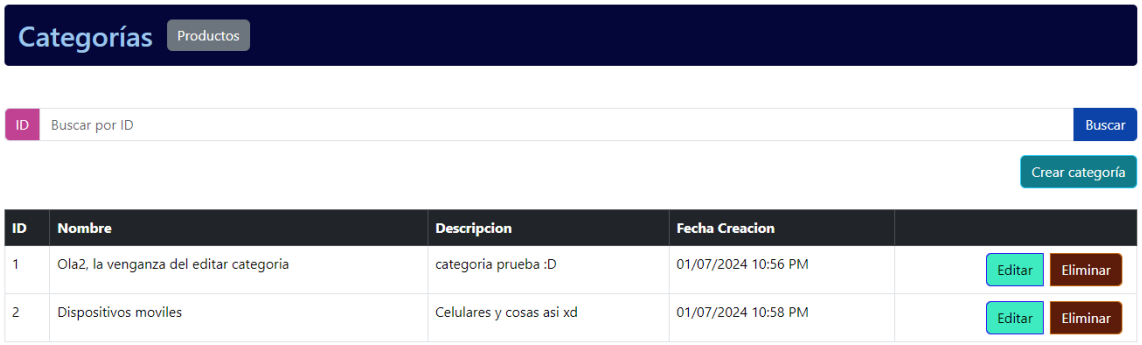


Categoría creada:D

Imagen 2

Editar categoria





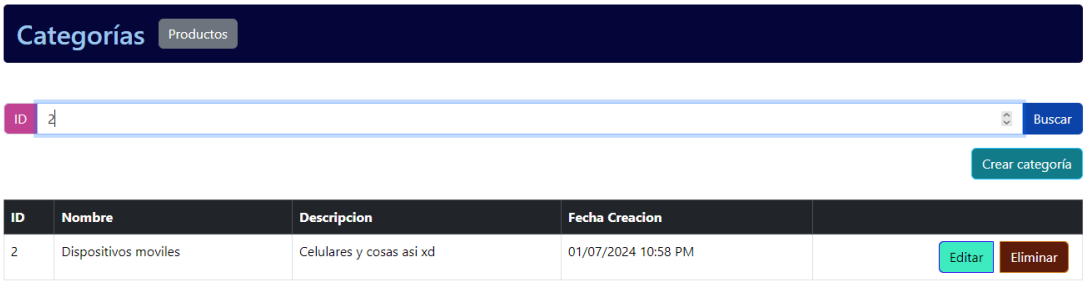
Eliminar categoría

Imagen 3

ID	Nombre	Descripcion	Fecha Creacion	
1	Ola2, la venganza del editar categoria	categoria prueba :D	01/07/2024 10:56 PM	Editar Eliminar
2	Dispositivos moviles	Celulares y cosas asi xd	01/07/2024 10:58 PM	Editar Eliminar
3	Comida	comida xd	01/07/2024 11:07 PM	Editar Eliminar
4	Comida	alimentos :D	01/07/2024 11:07 PM	Editar Eliminar

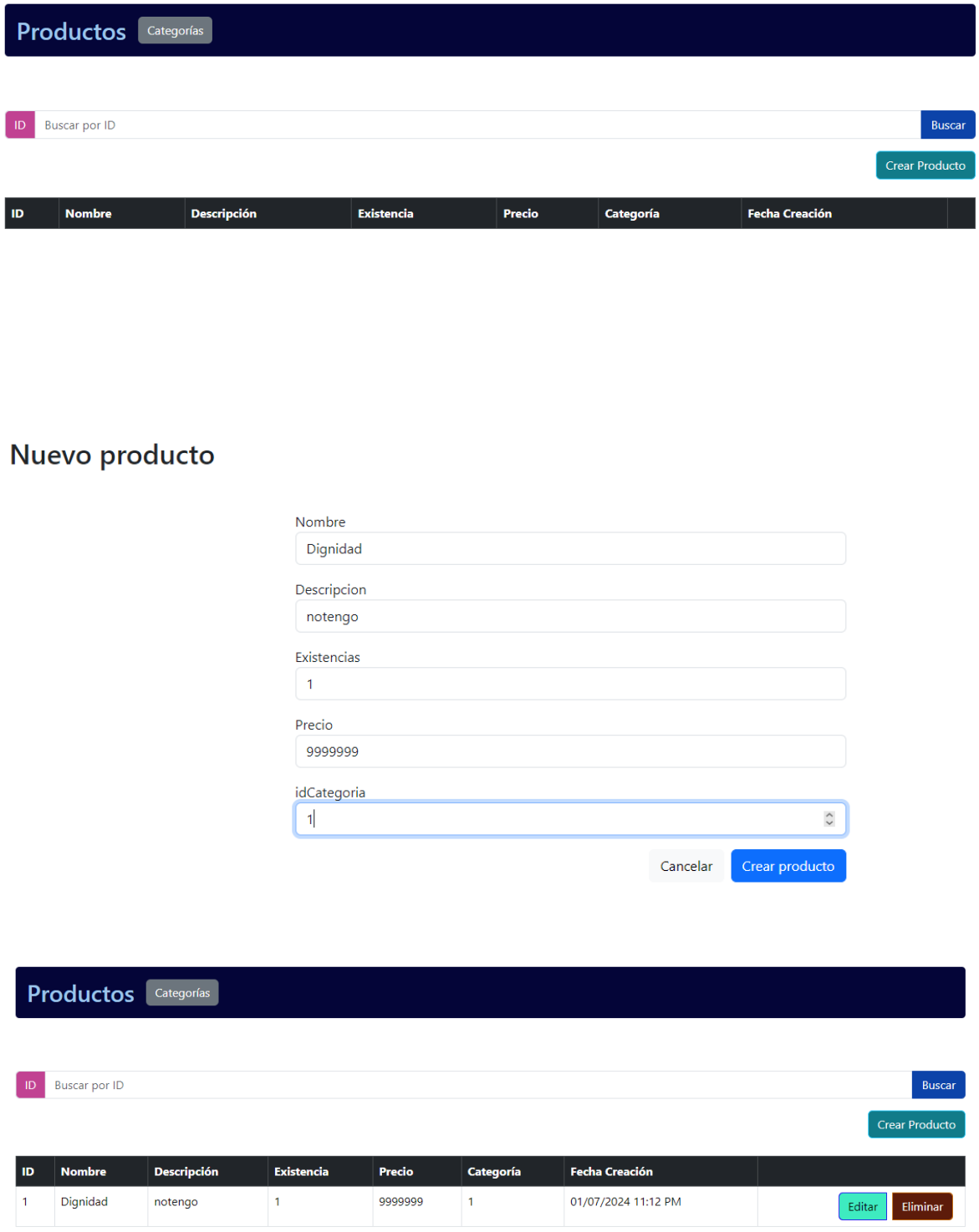
ID	Nombre	Descripcion	Fecha Creacion	
1	Ola2, la venganza del editar categoria	categoria prueba :D	01/07/2024 10:56 PM	Editar Eliminar
2	Dispositivos moviles	Celulares y cosas asi xd	01/07/2024 10:58 PM	Editar Eliminar
4	Comida	alimentos :D	01/07/2024 11:07 PM	Editar Eliminar

Buscar por id:



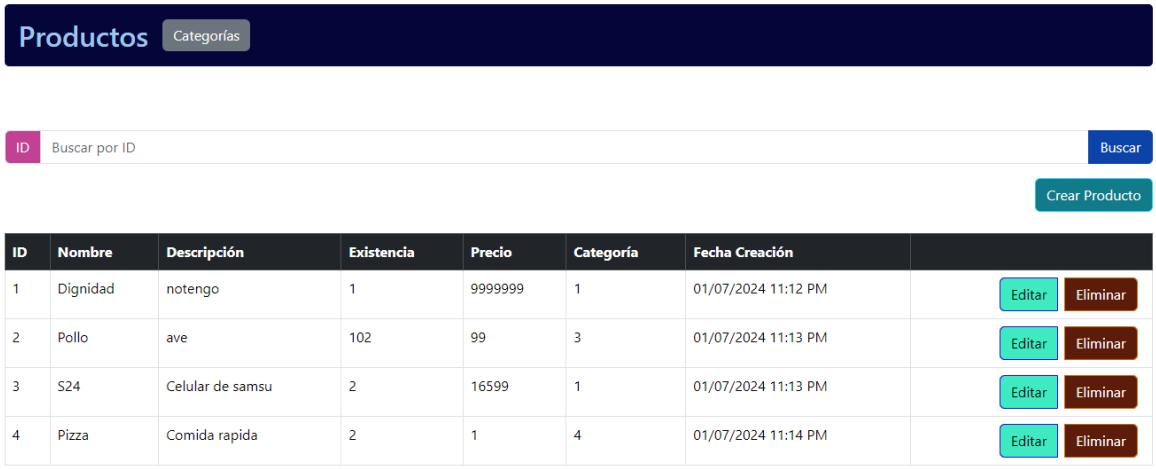
PRODUCTOS:

Imagen 4



Base de productos:

Imagen 5



ID	Nombre	Descripción	Existencia	Precio	Categoría	Fecha Creación	
1	Dignidad	notengo	1	9999999	1	01/07/2024 11:12 PM	Editar Eliminar
2	Pollo	ave	102	99	3	01/07/2024 11:13 PM	Editar Eliminar
3	S24	Celular de samsu	2	16599	1	01/07/2024 11:13 PM	Editar Eliminar
4	Pizza	Comida rapida	2	1	4	01/07/2024 11:14 PM	Editar Eliminar

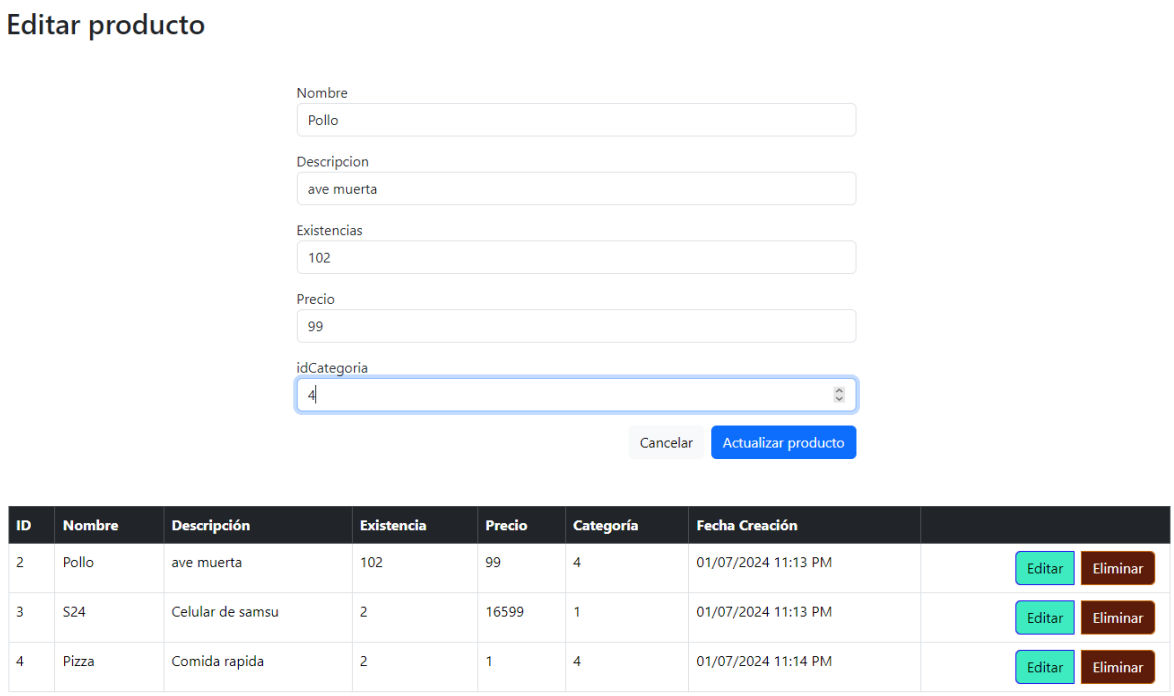
Eliminar:

ID	Nombre	Descripción	Existencia	Precio	Categoría	Fecha Creación	
1	Dignidad	notengo	1	9999999	1	01/07/2024 11:12 PM	Editar Eliminar
2	Pollo	ave	102	99	3	01/07/2024 11:13 PM	Editar Eliminar
3	S24	Celular de samsu	2	16599	1	01/07/2024 11:13 PM	Editar Eliminar
4	Pizza	Comida rapida	2	1	4	01/07/2024 11:14 PM	Editar Eliminar

ID	Nombre	Descripción	Existencia	Precio	Categoría	Fecha Creación	
2	Pollo	ave	102	99	3	01/07/2024 11:13 PM	Editar Eliminar
3	S24	Celular de samsu	2	16599	1	01/07/2024 11:13 PM	Editar Eliminar
4	Pizza	Comida rapida	2	1	4	01/07/2024 11:14 PM	Editar Eliminar

Imagen 6

Editar:



Buscar:

Conclusión

El desarrollo de la aplicación web para la gestión de categorías y productos ha sido una experiencia enriquecedora que permitió aplicar y consolidar conocimientos en diversas tecnologías y herramientas modernas de desarrollo web. A lo largo del proyecto, se implementaron y se integraron tanto el Backend, desarrollado con Spring Boot, como el Frontend, desarrollado con Angular y Bootstrap. Este proceso integral facilitó una comprensión completa de cómo los diferentes componentes de una aplicación web interactúan y funcionan juntos para crear una solución coherente

y funcional.

Durante el desarrollo del Backend con Spring Boot, se trabajó en la creación de servicios RESTful que permitieran manejar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) necesarias para la gestión de categorías y productos. Se implementaron modelos de datos utilizando JPA (Java Persistence API) para mapear las entidades a las tablas de la base de datos, y se desarrollaron controladores para manejar las solicitudes HTTP. Esta parte del proyecto subrayó la importancia de una arquitectura bien estructurada y de la utilización de patrones de diseño, como la inyección de dependencias, para mantener el código modular y fácil de mantener.

En la implementación del Frontend con Angular, se crearon componentes reutilizables y servicios que se encargaron de la comunicación con el Backend a través de HTTP. La utilización de Angular permitió desarrollar una interfaz de usuario dinámica y receptiva, donde los usuarios pueden interactuar con la aplicación de manera intuitiva. Bootstrap se utilizó para estilizar la aplicación, garantizando una apariencia profesional y consistente en todas las vistas. Este proceso destacó la relevancia de un diseño centrado en el usuario y de la implementación de buenas prácticas de desarrollo front-end, como el uso de servicios para la lógica de negocio y la separación de preocupaciones.

El proyecto presentó diversos desafíos, especialmente en la configuración y conexión de los distintos módulos de la aplicación. Problemas comunes como la gestión de CORS (Cross-Origin Resource Sharing), la validación de datos y la sincronización entre el Frontend y el Backend fueron abordados y resueltos mediante la consulta de documentación, la implementación de soluciones iterativas y el trabajo colaborativo. Estos desafíos sirvieron como valiosas oportunidades de aprendizaje, subrayando la importancia de la persistencia y la resolución de problemas en el desarrollo de software.

Link al GitHub #;

https://github.com/Jose2401/Api-Inventario