

## CRUD mySQL Angular

Vamos a desarrollar paso a paso tanto el frontend en Angular como el backend necesario para conectarnos a MySQL.

Comenzaré creando un proyecto completo que incluya:

### 1. Frontend en Angular 16.

#### 1.1 Crear el proyecto Angular.

En la terminal:

```
[jsersan@iMac-de-Jose angular16 % ng new crud-angular-mysql --routing --style=scss --skip-tests]
CREATE crud-angular-mysql/README.md (1070 bytes)
CREATE crud-angular-mysql/.editorconfig (274 bytes)
CREATE crud-angular-mysql/.gitignore (548 bytes)
CREATE crud-angular-mysql/angular.json (3523 bytes)
CREATE crud-angular-mysql/package.json (1049 bytes)
CREATE crud-angular-mysql/tsconfig.json (901 bytes)
CREATE crud-angular-mysql/tsconfig.app.json (263 bytes)
CREATE crud-angular-mysql/tsconfig.spec.json (273 bytes)
CREATE crud-angular-mysql/.vscode/extensions.json (130 bytes)
CREATE crud-angular-mysql/.vscode/launch.json (474 bytes)
CREATE crud-angular-mysql/.vscode/tasks.json (938 bytes)
CREATE crud-angular-mysql/src/favicon.ico (948 bytes)
CREATE crud-angular-mysql/src/index.html (302 bytes)
CREATE crud-angular-mysql/src/main.ts (214 bytes)
CREATE crud-angular-mysql/src/styles.scss (80 bytes)
CREATE crud-angular-mysql/src/assets/.gitkeep (0 bytes)
CREATE crud-angular-mysql/src/app/app-routing.module.ts (245 bytes)
CREATE crud-angular-mysql/src/app/app.module.ts (393 bytes)
CREATE crud-angular-mysql/src/app/app.component.scss (0 bytes)
CREATE crud-angular-mysql/src/app/app.component.html (23115 bytes)
CREATE crud-angular-mysql/src/app/app.component.ts (223 bytes)
✓ Packages installed successfully.
  Successfully initialized git.
jsersan@iMac-de-Jose angular16 %
```

#### 1.2 Entramos en el proyecto Angular:

```
✓ Packages installed successfully.
  Successfully initialized git.
[jsersan@iMac-de-Jose angular16 % cd crud-angular-mysql]
jsersan@iMac-de-Jose crud-angular-mysql %
```

#### 1.3 Instalamos las dependencias necesarias: bootstrap y material.

```
[jsersan@iMac-de-Jose crud-angular-mysql % npm i bootstrap]

added 2 packages, and audited 957 packages in 4s

120 packages are looking for funding
  run `npm fund` for details

6 vulnerabilities (4 moderate, 2 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

```
jsersan@iMac-de-Jose crud-angular-mysql % npm install @angular/material@15.2.9 @angular/cdk@15.2.9 @angular/animations@15.2.9 --legacy-peer-deps

added 55 packages, removed 1 package, changed 1 package, and audited 1011 packages in 13s

119 packages are looking for funding
  run `npm fund` for details

6 vulnerabilities (4 moderate, 2 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
jsersan@iMac-de-Jose crud-angular-mysql %
```

## 1.4 Modelo de datos:

```
src > app > models > TS producto.model.ts > Producto
1  export interface Producto {
2      id?: number;
3      nombre: string;
4      descripcion: string;
5      precio: number;
6      stock: number;
7      createdAt?: Date;
8  }
```

## 1.5 Servicio:



Colocamos las funciones del servicio:

```
src > app > services > producto.service.ts > ProductoService > getAll
1  // src/app/services/producto.service.ts
2
3  // Importación de Angular Injectable para crear un servicio
4  import { Injectable } from '@angular/core';
5  // HttpClient para realizar peticiones HTTP a nuestra API
6  import { HttpClient } from '@angular/common/http';
7  // Observable para manejar operaciones asíncronas
8  import { Observable } from 'rxjs';
9  // Importación del modelo Producto
10 import { Producto } from '../models/producto.model';
11
12 /**
13  * Servicio para manejar todas las operaciones CRUD relacionadas con productos.
14  * Este servicio se comunica con la API REST del backend.
15  * @Injectable({providedIn: 'root'}) - Decorador que permite que este servicio
16  * sea inyectado en otros componentes y que esté disponible en toda la aplicación
17  */
18 @Injectable({
19     providedIn: 'root'
20 })
21 // Disponible en toda la aplicación sin necesidad de incluirlo en 'providers' del módulo
22 }
```

Continuamos con el servicio:

```
22 export class ProductoService {
23     // URL base de la API que se utilizará para todas las peticiones
24     private apiUrl = 'http://localhost:3000/api/productos';
25
26     /**
27      * Constructor del servicio
28      * @param http - Inyección del servicio HttpClient para realizar peticiones HTTP
29      */
30     constructor(private http: HttpClient) { }
31
32     /**
33      * Obtiene todos los productos desde la API
34      * @returns Observable con un array de productos
35      */
36     getAll(): Observable<Producto[]> {
37         // Realiza una petición GET a la URL de la API y devuelve un observable
38         // El tipo <Producto[]> indica que la respuesta será un array de objetos Producto
39         return this.http.get<Producto[]>(this.apiUrl);
40     }
41
42     /**
43      * Obtiene un producto específico por su ID
44      * @param id - ID del producto a obtener
45      * @returns Observable con un único producto
46      */
47     get(id: number): Observable<Producto> {
48         // Construye la URL con el ID y realiza una petición GET
49         return this.http.get<Producto>(`${this.apiUrl}/${id}`);
50     }
51
52     /**
53      * Crea un nuevo producto
54      * @param data - Datos del producto a crear
55      * @returns Observable con la respuesta del servidor
56      */
57     create(data: Producto): Observable<any> {
58         // Realiza una petición POST enviando los datos del producto
59         return this.http.post(this.apiUrl, data);
60     }
61
62     /**
63      * Actualiza un producto existente
64      * @param id - ID del producto a actualizar
65      * @param data - Nuevos datos del producto
66      * @returns Observable con la respuesta del servidor
67      */
68     update(id: number, data: Producto): Observable<any> {
69         // Construye la URL con el ID y realiza una petición PUT con los nuevos datos
70         return this.http.put(`${this.apiUrl}/${id}`, data);
71     }
72
73     /**
74      * Elimina un producto por su ID
75      * @param id - ID del producto a eliminar
76      * @returns Observable con la respuesta del servidor
77      */
78     delete(id: number): Observable<any> {
79         // Construye la URL con el ID y realiza una petición DELETE
80         return this.http.delete(`${this.apiUrl}/${id}`);
81     }
82 }
```

Terminamos el servicio:

```

83  /**
84  * Busca productos por nombre
85  * @param nombre - Nombre o parte del nombre a buscar
86  * @returns Observable con un array de productos que coinciden con la búsqueda
87  */
88  findByNombre(nombre: string): Observable<Producto[]> {
89      // Realiza una petición GET con un parámetro de consulta para el nombre
90      // La API backend debe manejar este parámetro para filtrar los resultados
91      return this.http.get<Producto[]>(`${this.apiUrl}?nombre=${nombre}`);
92  }
93  }

```

## 1.6 Creamos los componentes:

- productos-list.component.
- producto-form.component.
- producto.detalle.component.

```

● jrsersan@iMac-de-Jose crud-angular-mysql % ng generate component components/productos-list --skip-tests
CREATE src/app/components/productos-list/productos-list.component.scss (0 bytes)
CREATE src/app/components/productos-list/productos-list.component.html (29 bytes)
CREATE src/app/components/productos-list/productos-list.component.ts (234 bytes)
UPDATE src/app/app.module.ts (516 bytes)
● jrsersan@iMac-de-Jose crud-angular-mysql % ng generate component components/producto-form --skip-tests
CREATE src/app/components/producto-form/producto-form.component.scss (0 bytes)
CREATE src/app/components/producto-form/producto-form.component.html (28 bytes)
CREATE src/app/components/producto-form/producto-form.component.ts (230 bytes)
UPDATE src/app/app.module.ts (635 bytes)
● jrsersan@iMac-de-Jose crud-angular-mysql % ng generate component components/producto-detalle --skip-tests
CREATE src/app/components/producto-detalle/producto-detalle.component.scss (0 bytes)
CREATE src/app/components/producto-detalle/producto-detalle.component.html (31 bytes)
CREATE src/app/components/producto-detalle/producto-detalle.component.ts (242 bytes)
UPDATE src/app/app.module.ts (766 bytes)
○ jrsersan@iMac-de-Jose crud-angular-mysql %

```

### a) producto-list.component.ts:

```

src > app > components > productos-list >  productos-list.component.ts >  ProductosListComponent
1  // src/app/components/productos-list/productos-list.component.ts
2
3  // Importaciones necesarias de Angular
4  import { Component, OnInit, ViewChild } from '@angular/core';
5  // Importaciones de Angular Material
6  import { MatPaginator } from '@angular/material/paginator'; // Para la paginación
7  import { MatTableDataSource } from '@angular/material/table'; // Para las tablas de datos
8  import { MatDialog } from '@angular/material/dialog'; // Para modales/diálogos
9  import { MatSnackBar } from '@angular/material/snack-bar'; // Para notificaciones tipo toast
10 // Importación del modelo Producto
11 import { Producto } from '../models/producto.model';
12 // Importación del servicio para operaciones CRUD
13 import { ProductoService } from '../services/producto.service';
14 // Importación del componente de formulario que se usará en el diálogo
15 import { ProductoFormComponent } from '../producto-form/producto-form.component';
16
17 /**
18 * Componente para mostrar y gestionar la lista de productos
19 * @Component - Decorador que define los metadatos del componente
20 */
21 @Component({
22     selector: 'app-productos-list', // Selector HTML del componente
23     templateUrl: './productos-list.component.html', // Ruta al archivo de la plantilla HTML
24     styleUrls: []
25 })

```

Continuamos con el componente:

```

26 export class ProductosListComponent implements OnInit {
27     // Columnas que se mostrarán en la tabla
28     displayedColumns: string[] = ['id', 'nombre', 'descripcion', 'precio', 'stock', 'acciones'];
29
30     // Fuente de datos para la tabla de Material, inicializada como un array vacío
31     // MatTableDataSource proporciona funcionalidades como ordenamiento, filtrado y paginación
32     dataSource = new MatTableDataSource<Producto>([]);
33
34     // Variable para almacenar el término de búsqueda
35     nombreBusqueda: string = '';
36
37     // Decorador ViewChild para acceder al componente de paginación en el DOM
38     // El signo de exclamación (!) es un operador de aserción de no-nulidad en TypeScript
39     @ViewChild(MatPaginator) paginator!: MatPaginator;
40
41     /**
42     * Constructor del componente
43     * @param productoService - Servicio para operaciones CRUD de productos
44     * @param dialog - Servicio de Angular Material para mostrar diálogos/modales
45     * @param snackBar - Servicio de Angular Material para mostrar notificaciones
46     */
47     constructor(
48         private productoService: ProductoService,
49         private dialog: MatDialog,
50         private snackBar: MatSnackBar
51     ) { }
52
53     /**
54     * Método del ciclo de vida que se ejecuta cuando se inicializa el componente
55     * Aquí realizamos operaciones iniciales como cargar los datos
56     */
57     ngOnInit(): void {
58         // Llama al método para obtener todos los productos al iniciar
59         this.obtenerProductos();
60     }
61
62     /**
63     * Método del ciclo de vida que se ejecuta después de que la vista del componente se ha inicializado
64     * Perfecto para configurar elementos que requieren que el DOM esté listo
65     */
66     ngAfterViewInit() {
67         // Asigna el paginador a la fuente de datos de la tabla
68         this.dataSource.paginator = this.paginator;
69     }
70
71     /**
72     * Método para obtener todos los productos desde el servicio
73     * Se suscribe al observable que retorna el servicio
74     */
75     obtenerProductos(): void {
76         this.productoService.getAll()
77             .subscribe({
78                 // Callback que se ejecuta cuando la petición es exitosa
79                 next: (data) => {
80                     // Asigna los datos recibidos a la fuente de datos de la tabla
81                     this.dataSource.data = data;
82                 },
83                 // Callback que se ejecuta si hay un error
84                 error: (e) => console.error(e)
85             });
86     }

```

El resto de las funciones:

```

88  /**
89  * Método para buscar productos por nombre
90  * Utiliza el valor de la variable nombreBusqueda
91  */
92  buscarPorNombre(): void {
93    // Verifica si hay un término de búsqueda válido (no vacío)
94    if (this.nombreBusqueda.trim()) {
95      // Llama al servicio para buscar productos por nombre
96      this.productoService.findByNombre(this.nombreBusqueda)
97        .subscribe({
98          // Actualiza la tabla con los resultados de la búsqueda
99          next: (data) => {
100            this.dataSource.data = data;
101          },
102          error: (e) => console.error(e)
103        });
104    } else {
105      // Si el campo de búsqueda está vacío, carga todos los productos
106      this.obtenerProductos();
107    }
108  }
109
110  /**
111  * Método para abrir el formulario de producto (para crear o editar)
112  * @param producto - Producto a editar (opcional). Si no se proporciona, crea uno nuevo.
113  */
114  abrirFormulario(producto?: Producto): void {
115    // Abre un diálogo con el componente ProductoFormComponent
116    const dialogRef = this.dialog.open(ProductoFormComponent, {
117      width: '500px', // Ancho del diálogo
118      data: producto || {} // Pasa el producto a editar o un objeto vacío para crear uno nuevo
119    });
120
121    // Se suscribe al evento de cierre del diálogo
122    dialogRef.afterClosed().subscribe(result => {
123      // Si hay un resultado (no se canceló la operación), recarga los productos
124      if (result) {
125        this.obtenerProductos();
126      }
127    });
128  }
129
130  /**
131  * Método para eliminar un producto
132  * @param id - ID del producto a eliminar
133  */
134  eliminarProducto(id: number): void {
135    // Muestra un diálogo de confirmación
136    if (confirm('¿Está seguro de eliminar este producto?')) {
137      // Si el usuario confirma, llama al servicio para eliminar el producto
138      this.productoService.delete(id)
139        .subscribe({
140          next: () => {
141            // Muestra una notificación de éxito
142            this.snackBar.open('Producto eliminado correctamente', 'Cerrar', {
143              duration: 3000 // Duración de la notificación en milisegundos
144            });
145            // Recarga la lista de productos
146            this.obtenerProductos();
147          },
148          error: (e) => console.error(e)
149        });
150    }
151  }
152  }

```



El template:

```

1 <!-- src/app/components/productos-list/productos-list.component.html -->
2
3 <!-- Contenedor principal con margen superior -->
4 <div class="container mt-4">
5   <!-- Título de la página -->
6   <h2>Lista de Productos</h2>
7
8   <!-- Fila para la barra de búsqueda y el botón de nuevo producto -->
9   <div class="row mb-3">
10    <!-- Columna para el campo de búsqueda (ocupa la mitad del ancho en pantallas medianas) -->
11    <div class="col-md-6">
12      <!-- Campo de búsqueda usando Material Form Field -->
13      <mat-form-field appearance="outline" class="w-100">
14        <!-- Etiqueta flotante del campo -->
15        <mat-label>Buscar por nombre</mat-label>
16      </mat-form-field>
17    </div>
18  </div>
19 </div>

```

Importamos en app.module.ts las librerías de material necesarias:

```

src > app > app.module.ts > AppModule
1 // src/app/app.module.ts
2 import { NgModule } from '@angular/core';
3 import { BrowserModule } from '@angular/platform-browser';
4 import { FormsModule, ReactiveFormsModule } from '@angular/forms';
5 import { HttpClientModule } from '@angular/common/http';
6 import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
7
8 // Importaciones de Angular Material
9 import { MatFormFieldModule } from '@angular/material/form-field';
10 import { MatInputModule } from '@angular/material/input'; // Importante para los inputs dentro de form-field
11 import { MatButtonModule } from '@angular/material/button';
12 import { MatTableModule } from '@angular/material/table';
13 import { MatPaginatorModule } from '@angular/material/paginator';
14 import { MatIconModule } from '@angular/material/icon';
15 import { MatDialogModule } from '@angular/material/dialog';
16 import { MatSnackBarModule } from '@angular/material/snack-bar';
17
18 import { AppRoutingModuleModule } from './app-routing.module';
19 import { AppComponent } from './app.component';
20 import { ProductosListComponent } from './components/productos-list/productos-list.component';
21 import { ProductoDetalleComponent } from './components/producto-detalle/producto-detalle.component';
22 import { ProductoFormComponent } from './components/producto-form/producto-form.component';
23
24 @NgModule({
25   declarations: [
26     AppComponent,
27     ProductosListComponent,
28     ProductoDetalleComponent,
29     ProductoFormComponent
30   ],
31   imports: [
32     BrowserModule,
33     AppRoutingModuleModule,
34     FormsModule,
35     ReactiveFormsModule,
36     HttpClientModule,
37     BrowserAnimationsModule,
38     // Módulos de Angular Material
39     MatFormFieldModule,
40     MatInputModule, // Este es crucial para mat-form-field
41     MatButtonModule,
42     MatTableModule,
43     MatPaginatorModule,
44     MatIconModule,
45     MatDialogModule,
46     MatSnackBarModule
47   ],
48   providers: [],
49   bootstrap: [AppComponent]
50 })
51 export class AppModule { }

```

Ahora el template:

```

1  <!-- src/app/components/productos-list/productos-list.component.html -->
2
3  <!-- Contenedor principal con margen superior -->
4  <div class="container mt-4">
5      <!-- Título de la página -->
6      <h2>Lista de Productos</h2>
7
8      <!-- Fila para la barra de búsqueda y el botón de nuevo producto -->
9      <div class="row mb-3">
10         <!-- Columna para el campo de búsqueda (ocupa la mitad del ancho en pantallas medianas) -->
11         <div class="col-md-6">
12             <!-- Campo de búsqueda usando Material Form Field -->
13             <mat-form-field appearance="outline" class="w-100">
14                 <!-- Etiqueta flotante del campo -->
15                 <mat-label>Buscar por nombre</mat-label>
16
17                 <!-- Input vinculado a la variable nombreBusqueda con ngModel (requiere FormsModule) -->
18                 <!-- El evento keyup.enter ejecuta la búsqueda al presionar Enter -->
19                 <input matInput [(ngModel)]="nombreBusqueda" (keyup.enter)="buscarPorNombre()">
20
21                 <!-- Botón de búsqueda ubicado en el sufijo del campo -->
22                 <button mat-button matSuffix mat-icon-button (click)="buscarPorNombre()">
23                     <!-- Ícono de búsqueda (requiere MatIconModule) -->
24                     <mat-icon>search</mat-icon>
25                 </button>
26             </mat-form-field>
27         </div>
28
29         <!-- Columna para el botón de nuevo producto (ocupa la mitad del ancho y alinea a la derecha) -->
30         <div class="col-md-6 text-end">
31             <!-- Botón para abrir el formulario de nuevo producto -->
32             <button mat-raised-button color="primary" (click)="abrirFormulario()">
33                 <!-- Ícono de añadir -->
34                 <mat-icon>add</mat-icon> Nuevo Producto
35             </button>
36         </div>
37     </div>
38
39     <!-- Contenedor de la tabla con elevación (sombra) -->
40     <div class="mat-elevation-z8">
41         <!-- Tabla Material vinculada a dataSource (MatTableDataSource) -->
42         <table mat-table [dataSource]="dataSource" class="w-100">
43
44             <!-- Definición de la columna ID -->
45             <ng-container matColumnDef="id">
46                 <!-- Encabezado de la columna -->
47                 <th mat-header-cell *matHeaderCellDef>ID</th>
48                 <!-- Celda con datos del producto -->
49                 <td mat-cell *matCellDef="let producto">{{ producto.id }}</td>
50             </ng-container>
51
52             <!-- Definición de la columna Nombre -->
53             <ng-container matColumnDef="nombre">
54                 <th mat-header-cell *matHeaderCellDef>Nombre</th>
55                 <td mat-cell *matCellDef="let producto">{{ producto.nombre }}</td>
56             </ng-container>
57
58             <!-- Definición de la columna Descripción -->
59             <ng-container matColumnDef="descripcion">
60                 <th mat-header-cell *matHeaderCellDef>Descripción</th>
61                 <td mat-cell *matCellDef="let producto">{{ producto.descripcion }}</td>
62             </ng-container>

```



El resto del template:

```

64     <!-- Definición de la columna Precio (con pipe currency para formatear como moneda) -->
65     <ng-container matColumnDef="precio">
66         <th mat-header-cell *matHeaderCellDef>Precio</th>
67         <td mat-cell *matCellDef="let producto">{{ producto.precio | currency }}</td>
68     </ng-container>
69
70     <!-- Definición de la columna Stock -->
71     <ng-container matColumnDef="stock">
72         <th mat-header-cell *matHeaderCellDef>Stock</th>
73         <td mat-cell *matCellDef="let producto">{{ producto.stock }}</td>
74     </ng-container>
75
76     <!-- Definición de la columna Acciones (botones de editar y eliminar) -->
77     <ng-container matColumnDef="acciones">
78         <th mat-header-cell *matHeaderCellDef>Acciones</th>
79         <td mat-cell *matCellDef="let producto">
80             <!-- Botón de editar que abre el formulario pasando el producto actual -->
81             <button mat-icon-button color="primary" (click)="abrirFormulario(producto)">
82                 <mat-icon>edit</mat-icon>
83             </button>
84             <!-- Botón de eliminar que ejecuta el método eliminarProducto -->
85             <button mat-icon-button color="warn" (click)="eliminarProducto(producto.id)">
86                 <mat-icon>delete</mat-icon>
87             </button>
88         </td>
89     </ng-container>
90
91     <!-- Definición de la fila de encabezado referenciando todas las columnas definidas -->
92     <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
93
94     <!-- Definición de las filas de datos -->
95     <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
96
97     <!-- Fila que se muestra cuando no hay datos -->
98     <tr class="mat-row" *matNoDataRow>
99         <td class="mat-cell text-center" colspan="6">No se encontraron productos</td>
100     </tr>
101 </table>
102
103 <!-- Componente de paginación referenciado por @ViewChild en el componente TS -->
104 <mat-paginator [pageSizeOptions]="[5, 10, 25]" showFirstLastButtons></mat-paginator>
105 </div>
106 </div>

```

b) producto-form.component.ts:

```

src > app > components > producto-form > producto-form.component.ts > ProductoFormComponent >
1  // src/app/components/producto-form/producto-form.component.ts
2
3  // Importaciones necesarias de Angular
4  import { Component, Inject, OnInit } from '@angular/core';
5  // Importaciones para formularios reactivos
6  import { FormBuilder, FormGroup, Validators } from '@angular/forms';
7  // Importaciones para manejo de diálogos de Material
8  import { MatDialogRef, MAT_DIALOG_DATA } from '@angular/material/dialog';
9  // Importación para mostrar notificaciones
10 import { MatSnackBar } from '@angular/material/snack-bar';
11 // Importación del modelo de Producto
12 import { Producto } from '../../models/producto.model';
13 // Importación del servicio para operaciones CRUD
14 import { ProductoService } from '../../services/producto.service';
15
16 /**
17  * Componente para el formulario de creación y edición de productos
18  * Este componente se muestra dentro de un diálogo de Angular Material
19  */

```

El resto del componente:

```

20 @Component({
21   selector: 'app-producto-form',
22   templateUrl: './producto-form.component.html',
23   styleUrls: ['./producto-form.component.scss']
24 })
25 export class ProductoFormComponent implements OnInit {
26   // Declaración del formulario reactivo
27   productoForm!: FormGroup;
28
29   // Bandera para determinar si estamos editando o creando un producto
30   isEditing = false;
31
32   /**
33    * Constructor del componente
34    * @param fb - FormBuilder para crear formularios reactivos
35    * @param productoService - Servicio para operaciones CRUD de productos
36    * @param snackBar - Servicio para mostrar notificaciones
37    * @param dialogRef - Referencia al diálogo actual para poder cerrarlo
38    * @param data - Datos pasados al diálogo (puede ser un producto existente o un objeto vacío)
39    */
40   constructor(
41     private fb: FormBuilder,
42     private productoService: ProductoService,
43     private snackBar: MatSnackBar,
44     // dialogRef permite interactuar con el diálogo, por ejemplo, para cerrarlo
45     public dialogRef: MatDialogRef<ProductoFormComponent>,
46     // MAT_DIALOG_DATA es un token de inyección para recibir datos del componente que abrió el diálogo
47     @Inject(MAT_DIALOG_DATA) public data: Producto
48   ) { }
49
50   /**
51    * Método del ciclo de vida que se ejecuta cuando se inicializa el componente
52    * Aquí creamos y configuramos el formulario
53    */
54   ngOnInit(): void {
55     // Determinamos si estamos en modo edición verificando si el producto tiene ID
56     this.isEditing = !!this.data.id;
57
58     // Creamos el formulario reactivo con FormBuilder
59     this.productoForm = this.fb.group({
60       // Para cada campo, establecemos el valor inicial (desde data) y los validadores
61
62       // nombre: obligatorio y mínimo 3 caracteres
63       nombre: [this.data.nombre || '', [Validators.required, Validators.minLength(3)]],
64
65       // descripción: obligatorio
66       descripcion: [this.data.descripcion || '', [Validators.required]],
67
68       // precio: obligatorio y mayor o igual a 0
69       precio: [this.data.precio || 0, [Validators.required, Validators.min(0)]],
70
71       // stock: obligatorio y mayor o igual a 0
72       stock: [this.data.stock || 0, [Validators.required, Validators.min(0)]]
73     });
74   }
75
76   /**
77    * Método que se ejecuta al enviar el formulario
78    * Maneja tanto la creación como la actualización de productos
79    */



```

Para finalizar el componente:

```
80   onSubmit(): void {
81     // Verificamos si el formulario es válido antes de proceder
82     if (this.productoForm.invalid) {
83       return;
84     }
85
86     // Creamos un objeto producto con los valores del formulario
87     const producto: Producto = {
88       ...this.productoForm.value
89       // Al usar el operador spread (...) estamos copiando todas las propiedades
90     };
91
92     // Determinamos si estamos editando o creando un producto
93     if (this.isEditing) {
94       // Actualización: llamamos al método update del servicio
95       this.productoService.update(this.data.id!, producto)
96         .subscribe({
97           next: () => {
98             // Notificamos el éxito con un snackbar
99             this.snackBar.open('Producto actualizado correctamente', 'Cerrar', {
100               duration: 3000 // Duración en milisegundos
101             });
102             // Cerramos el diálogo y pasamos true para indicar que se realizó una acción
103             this.dialogRef.close(true);
104           },
105           error: (e) => console.error(e)
106         });
107     } else {
108       // Creación: llamamos al método create del servicio
109       this.productoService.create(producto)
110         .subscribe({
111           next: () => {
112             // Notificamos el éxito con un snackbar
113             this.snackBar.open('Producto creado correctamente', 'Cerrar', {
114               duration: 3000
115             });
116             // Cerramos el diálogo y pasamos true para indicar que se realizó una acción
117             this.dialogRef.close(true);
118           },
119           error: (e) => console.error(e)
120         });
121     }
122   }
123
124   /**
125    * Método para cancelar la operación y cerrar el diálogo
126    * No pasa ningún valor al cerrar, lo que indica que no se realizó ninguna acción
127    */
128   cancelar(): void {
129     this.dialogRef.close();
130   }
131 }
```

## producto-form.component.html

```

irc > app > components > producto-form >  producto-form.component.html >  form
Go to component
1 <!-- src/app/components/producto-form/producto-form.component.html -->
2
3 <!-- Título del diálogo que cambia según estemos editando o creando -->
4 <!-- Este elemento es reconocido por Angular Material como el título del diálogo -->
5 <h2 mat-dialog-title>{{ isEditing ? 'Editar' : 'Nuevo' }} Producto</h2>
6
7 <!-- Formulario con evento submit vinculado al método onSubmit() -->
8 <form [formGroup]="productoForm" (ngSubmit)="onSubmit()">
9   <!-- Contenido principal del diálogo -->
10  <!-- Este elemento es reconocido por Angular Material como el contenido del diálogo -->
11  <div mat-dialog-content>
12    <!-- Campo de formulario para el nombre -->
13    <mat-form-field appearance="outline" class="w-100">
14      <!-- Etiqueta del campo -->
15      <mat-label>Nombre</mat-label>
16
17      <!-- Input vinculado al control 'nombre' del formulario -->
18      <!-- 'required' añade un asterisco visual pero la validación real está en el TS -->
19      <input matInput formControlName="nombre" required>
20
21      <!-- Mensaje de error si el nombre es requerido -->
22      <!-- Solo se muestra si el control ha sido tocado y tiene el error 'required' -->
23      <mat-error *ngIf="productoForm.get('nombre')?.hasError('required')">
24        El nombre es obligatorio
25      </mat-error>
26
27      <!-- Mensaje de error si el nombre no cumple con la longitud mínima -->
28      <mat-error *ngIf="productoForm.get('nombre')?.hasError('minlength')">
29        El nombre debe tener al menos 3 caracteres
30      </mat-error>
31    </mat-form-field>
32
33    <!-- Campo de formulario para la descripción -->
34    <mat-form-field appearance="outline" class="w-100">
35      <mat-label>Descripción</mat-label>
36
37      <!-- Textarea para descripciones más largas, con 3 filas iniciales -->
38      <textarea matInput formControlName="descripcion" rows="3" required></textarea>
39
40      <!-- Mensaje de error si la descripción es requerida -->
41      <mat-error *ngIf="productoForm.get('descripcion')?.hasError('required')">
42        La descripción es obligatoria
43      </mat-error>
44    </mat-form-field>
45
46    <!-- Fila para precio y stock (usando layout de Bootstrap) -->
47    <div class="row">
48      <!-- Columna para el precio (ocupa la mitad del ancho) -->
49      <div class="col-md-6">
50        <mat-form-field appearance="outline" class="w-100">
51          <mat-label>Precio</mat-label>
52
53          <!-- Input numérico para el precio -->
54          <input matInput type="number" formControlName="precio" min="0" required>
55
56          <!-- Prefijo de símbolo de moneda antes del valor -->
57          <span matPrefix>$</span>
58
59          <!-- Mensajes de error para el precio -->
60          <mat-error *ngIf="productoForm.get('precio')?.hasError('required')">
61            El precio es obligatorio
62          </mat-error>
63          <mat-error *ngIf="productoForm.get('precio')?.hasError('min')">
64            El precio debe ser mayor o igual a 0
65          </mat-error>
66        </mat-form-field>
67      </div>

```

El resto del template:

```

69     <!-- Columna para el stock (ocupa la mitad del ancho) -->
70     <div class="col-md-6">
71         <mat-form-field appearance="outline" class="w-100">
72             <mat-label>Stock</mat-label>
73
74             <!-- Input numérico para el stock -->
75             <input matInput type="number" formControlName="stock" min="0" required>
76
77             <!-- Mensajes de error para el stock -->
78             <mat-error *ngIf="productoForm.get('stock')?.hasError('required')">
79                 El stock es obligatorio
80             </mat-error>
81             <mat-error *ngIf="productoForm.get('stock')?.hasError('min')">
82                 El stock debe ser mayor o igual a 0
83             </mat-error>
84         </mat-form-field>
85     </div>
86 </div>
87 </div>
88
89 <!-- Pie del diálogo con botones de acción -->
90 <!-- Este elemento es reconocido por Angular Material como las acciones del diálogo -->
91 <div mat-dialog-actions class="justify-content-end">
92     <!-- Botón para cancelar la operación -->
93     <!-- Tipo "button" para que no envíe el formulario -->
94     <button mat-button type="button" (click)="cancelar()">Cancelar</button>
95
96     <!-- Botón para enviar el formulario -->
97     <!-- Se deshabilita si el formulario no es válido -->
98     <!-- El texto cambia según estemos editando o creando -->
99     <button mat-raised-button color="primary" type="submit" [disabled]="productoForm.invalid">
100         {{ isEditing ? 'Actualizar' : 'Guardar' }}
101     </button>
102 </div>
103 </form>

```

c) producto-detalle.component.ts:

```

1  // src/app/components/producto-detalle/producto-detalle.component.ts
2
3  // Importaciones necesarias de Angular
4  import { Component, OnInit } from '@angular/core';
5  import { ActivatedRoute, Router } from '@angular/router';
6  // Importación del modelo Producto
7  import { Producto } from '../../models/producto.model';
8  // Importación del servicio para operaciones CRUD
9  import { ProductoService } from '../../services/producto.service';
10 // Importación para mostrar notificaciones
11 import { MatSnackBar } from '@angular/material/snack-bar';
12
13 /**
14  * Componente para mostrar y modificar los detalles de un producto específico
15  * Este componente se muestra en una página dedicada (no en un diálogo)
16  */
17 @Component({
18     selector: 'app-producto-detalle',
19     templateUrl: './producto-detalle.component.html',
20     styleUrls: []
21 })
22 export class ProductoDetalleComponent implements OnInit {
23     // Objeto para almacenar el producto actual
24     // Se inicializa con valores por defecto para evitar errores de undefined
25     currentProducto: Producto = {
26         nombre: '',
27         descripcion: '',
28         precio: 0,
29         stock: 0
30     };
31
32     // Variable para mensajes de estado/error
33     message = '';
34
35     /**
36     * Constructor del componente
37     * @param productoService - Servicio para operaciones CRUD de productos
38     * @param route - Servicio que proporciona acceso a la información de la ruta actual
39     * @param router - Servicio para navegar entre rutas
40     * @param snackBar - Servicio de Angular Material para mostrar notificaciones
41     */

```

El resto del componente:

```

42     constructor(
43         private productoService: ProductoService,
44         private route: ActivatedRoute,
45         private router: Router,
46         private snackBar: MatSnackBar
47     ) { }
48
49     /**
50      * Método del ciclo de vida que se ejecuta cuando se inicializa el componente
51      * Aquí obtenemos el ID de la ruta y cargamos el producto
52      */
53     ngOnInit(): void {
54         // Obtiene el parámetro 'id' de la URL y lo usa para cargar el producto
55         this.obtenerProducto(this.route.snapshot.params['id']);
56     }
57
58     /**
59      * Método para obtener un producto específico por su ID
60      * @param id - ID del producto a obtener
61      */
62     obtenerProducto(id: number): void {
63         this.productoService.get(id)
64             .subscribe({
65                 next: (data) => {
66                     // Almacena el producto obtenido en la variable currentProducto
67                     this.currentProducto = data;
68                 },
69                 error: (e) => console.error(e)
70             });
71     }
72
73     /**
74      * Método para actualizar el producto actual
75      * Utiliza los datos actuales del modelo currentProducto
76      */
77     actualizarProducto(): void {
78         this.productoService.update(this.currentProducto.id!, this.currentProducto)
79             .subscribe({
80                 next: (res) => {
81                     // Muestra una notificación de éxito
82                     this.snackBar.open('Producto actualizado correctamente', 'Cerrar', {
83                         duration: 3000 // Duración en milisegundos
84                     });
85                 },
86                 error: (e) => console.error(e)
87             });
88     }
89
90     /**
91      * Método para eliminar el producto actual
92      * Incluye una confirmación antes de proceder
93      */
94     eliminarProducto(): void {
95         // Solicita confirmación al usuario
96         if (confirm('¿Está seguro de eliminar este producto?')) {
97             this.productoService.delete(this.currentProducto.id!)
98                 .subscribe({
99                     next: (res) => {
100                         // Navega de vuelta a la lista de productos
101                         this.router.navigate(['/productos']);
102                         // Muestra una notificación de éxito
103                         this.snackBar.open('Producto eliminado correctamente', 'Cerrar', {
104                             duration: 3000
105                         });
106                     },
107                     error: (e) => console.error(e)
108                 });
109         }
110     }
111

```



El template producto-detalle.component.html:

```

1  <!-- src/app/components/producto-detalle/producto-detalle.component.html -->
2
3  <!-- Contenedor principal con margen superior -->
4  <div class="container mt-4">
5      <!-- Fila con justificación centrada para el contenido -->
6      <div class="row justify-content-center">
7          <!-- Columna que ocupa 8 de 12 columnas en pantallas medianas y grandes -->
8          <div class="col-md-8">
9              <!-- Tarjeta para mostrar el detalle del producto -->
10             <div class="card">
11                 <!-- Encabezado de la tarjeta con fondo azul y texto blanco -->
12                 <div class="card-header bg-primary text-white">
13                     <h3>Detalle del Producto</h3>
14                 </div>
15
16                 <!-- Cuerpo de la tarjeta -->
17                 <div class="card-body">
18                     <!-- Contenido que se muestra solo si el producto tiene ID (existe) -->
19                     <div *ngIf="currentProducto.id" class="mb-3">
20                         <!-- Formulario para el nombre -->
21                         <div class="form-group">
22                             <label for="nombre">Nombre</label>
23                             <!-- Input vinculado al modelo currentProducto.nombre mediante [(ngModel)] -->
24                             <input
25                                 type="text"
26                                 class="form-control"
27                                 id="nombre"
28                                 [(ngModel)]="currentProducto.nombre"
29                                 name="nombre"
30                             />
31                         </div>
32
33                         <!-- Formulario para la descripción -->
34                         <div class="form-group">
35                             <label for="descripcion">Descripción</label>
36                             <!-- Textarea para texto más largo -->
37                             <textarea
38                                 class="form-control"
39                                 id="descripcion"
40                                 [(ngModel)]="currentProducto.descripcion"
41                                 name="descripcion"
42                             ></textarea>
43                         </div>
44
45                         <!-- Formulario para el precio -->
46                         <div class="form-group">
47                             <label for="precio">Precio</label>
48                             <!-- Input numérico para el precio -->
49                             <input
50                                 type="number"
51                                 class="form-control"
52                                 id="precio"
53                                 [(ngModel)]="currentProducto.precio"
54                                 name="precio"
55                             />
56                         </div>
57
58                         <!-- Formulario para el stock -->
59                         <div class="form-group">
60                             <label for="stock">Stock</label>
61                             <!-- Input numérico para el stock -->
62                             <input
63                                 type="number"
64                                 class="form-control"
65                                 id="stock"
66                                 [(ngModel)]="currentProducto.stock"
67                                 name="stock"
68                             />
69                         </div>

```

## 1.7 Archivo de rutas: src/app/app-routing.module.ts

```
src > app > app-routing.module.ts > AppRoutingModuleModule
1  import { NgModule } from '@angular/core';
2  import { RouterModule, Routes } from '@angular/router';
3  import { ProductosListComponent } from '../components/productos-list/productos-list.component';
4  import { ProductoDetalleComponent } from '../components/producto-detalle/producto-detalle.component';
5  import { ProductoFormComponent } from '../components/producto-form/producto-form.component';
6
7  const routes: Routes = [
8    { path: '', redirectTo: 'productos', pathMatch: 'full' },
9    { path: 'productos', component: ProductosListComponent },
10   { path: 'productos/:id', component: ProductoDetalleComponent },
11   { path: 'add', component: ProductoFormComponent }
12 ];
13
14 @NgModule({
15   imports: [RouterModule.forRoot(routes)],
16   exports: [RouterModule]
17 })
18 export class AppRoutingModule {}
```

## 1.8 Estilos globales: src/styles.css:

```
src > styles.scss > .mat-form-field
1  /* You can add global styles to this file, and also import other style files */
2  @import "~bootstrap/dist/css/bootstrap.min.css";
3  @import "~@angular/material/prebuilt-themes/indigo-pink.css";
4
5  html, body { height: 100%; }
6  body { margin: 0; font-family: Roboto, "Helvetica Neue", sans-serif; }
7
8  .mat-form-field {
9    margin-bottom: 15px;
10 }
```

## 1.9 Componente principal: app.component.html:

```
rc > app > app.component.html > ...
1  <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
2    <div class="container">
3      <a class="navbar-brand" routerLink="/">CRUD Angular-MySQL</a>
4      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">
5        <span class="navbar-toggler-icon"></span>
6      </button>
7      <div class="collapse navbar-collapse" id="navbarNav">
8        <ul class="navbar-nav">
9          <li class="nav-item">
10             <a class="nav-link" routerLink="/productos" routerLinkActive="active">Productos</a>
11           </li>
12         </ul>
13       </div>
14     </div>
15   </nav>
16
17   <div class="container-fluid">
18     <router-outlet></router-outlet>
19   </div>
20
```