

Actualizar un Producto con Angular



Actualización de un producto en Angular (Parte 5 de la edición de un producto)

En esta quinta parte, agregaremos la funcionalidad para **actualizar un producto en Angular**. Modificaremos el **servicio**, el **formulario de edición**, y el **componente** para permitir que los cambios realizados en el producto se envíen al backend y se reflejen correctamente en la lista de productos.

Paso 1: Modificar `producto.service.ts` para agregar el método `editarProducto()`

Código actualizado

```
import { HttpClient } from '@angular/common/http';
import { inject, Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Producto } from './producto';

@Injectable({
  providedIn: 'root'
```

```
)  
export class ProductoService {  
  
    private urlBase = "http://localhost:8080/inventario-app/productos";  
  
    private clienteHttp = inject(HttpClient);  
  
    obtenerProductosLista(): Observable<Producto[]> {  
        return this.clienteHttp.get<Producto[]>(this.urlBase);  
    }  
  
    agregarProducto(producto: Producto): Observable<Object> {  
        return this.clienteHttp.post(this.urlBase, producto);  
    }  
  
    obtenerProductoPorId(id: number) {  
        return this.clienteHttp.get<Producto>(`${this.urlBase}/${id}`);  
    }  
  
    editarProducto(id: number, producto: Producto): Observable<Object>{  
        return this.clienteHttp.put(` ${this.urlBase}/${id}` , producto);  
    }  
}
```

Explicación de los cambios

1. Se agregó el método `editarProducto(id, producto)`:

```
editarProducto(id: number, producto: Producto): Observable<Object>{  
    return this.clienteHttp.put(` ${this.urlBase}/${id}` , producto);  
}
```

- Usa `HttpClient.put()` para realizar una **petición PUT** al backend.
- Envía el **ID** en la **URL** y el objeto `producto` en el cuerpo de la solicitud.
- Retorna un `Observable<Object>`, lo que permite manejar la respuesta del servidor.

Paso 2: Modificar `editar-producto.component.html` para actualizar el formulario

Código actualizado

```

<div class="container">

    <div class="container text-center" style="margin: 30px">
        <h3>Editar Producto</h3>
    </div>

    <form (ngSubmit)="onSubmit()">
        <div class="mb-3">
            <label for="descripcion" class="form-label">Descripción del Producto</label>
            <input type="text" class="form-control" id="descripcion" name="descripcion" required="true" [(ngModel)]="producto.descripcion">
        </div>
        <div class="mb-3">
            <label for="precio" class="form-label">Precio</label>
            <input type="number" step="any" class="form-control" id="precio" name="precio" [(ngModel)]="producto.precio">
        </div>
        <div class="mb-3">
            <label for="existencia" class="form-label">Existencia</label>
            <input type="number" class="form-control" id="existencia" name="existencia" [(ngModel)]="producto.existencia">
        </div>
        <div class="text-center">
            <button type="submit" class="btn btn-warning btn-sm me-3">Guardar</button>
            <a href="/" class="btn btn-danger btn-sm">Regresar</a>
        </div>
    </form>

</div>

```

Explicación de los cambios

1. Se cambió la etiqueta del botón de **Editar** a **Guardar**:

```

<button type="submit" class="btn btn-warning btn-sm me-3">Guardar</button>

```

- Hace más claro que la acción del botón es **guardar cambios** en el producto editado.

2. Los campos del formulario utilizan **ngModel**:

```

[(ngModel)]="producto.descripcion"

```

```
[ (ngModel) ]="producto.precio"
[ (ngModel) ]="producto.existencia"
```

- Permiten la **vinculación bidireccional** (Two-Way Binding), para que los cambios en el formulario actualicen automáticamente el objeto producto.
-

Paso 3: Modificar `editar-producto.component.ts` para implementar la actualización del producto

Código actualizado

```
import { Component, inject } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { Producto } from '../producto';
import { ProductoService } from '../producto.service';
import { FormsModule } from '@angular/forms';

@Component({
  selector: 'app-editar-producto',
  imports: [FormsModule],
  templateUrl: './editar-producto.component.html'
})
export class EditarProductoComponent {
  producto: Producto = new Producto();
  id: number;

  private productoServicio = inject(ProductoService);
  private ruta = inject(ActivatedRoute);
  private enrutador= inject(Router);

  ngOnInit() {
    this.id = this.ruta.snapshot.params['id'];
    this.productoServicio.obtenerProductoPorId(this.id).subscribe(
      {
        next: (datos) => this.producto = datos
        ,
        error: (errores: any) => console.log(errores)
      }
    );
  }

  onSubmit(){
    this.guardarProducto();
  }
}
```

```

    }

guardarProducto(){
    this.productoServicio.editarProducto(this.id, this.producto).subscribe(
        {
            next: (datos) => this.irProductoLista(),
            error: (errores) => console.log(errores)
        }
    );
}

irProductoLista(){
    this.enrutador.navigate(['/productos']);
}
}

```

Explicación de los cambios

1. Se inyectó Router para redirigir al usuario:

```
private enrutador = inject(Router);
```

- Permite redirigir a la lista de productos después de actualizar un producto.

2. Se modificó el método onSubmit() para guardar los cambios:

```
onSubmit() {
    this.guardarProducto();
}
```

- Llama a guardarProducto(), que se encarga de actualizar el producto en la base de datos.

3. Se agregó el método guardarProducto() para enviar la petición PUT:

```
guardarProducto() {
    this.productoServicio.editarProducto(this.id,
    this.producto).subscribe(
        {
            next: (datos) => this.irProductoLista(),
            error: (errores) => console.log(errores)
        }
    );
}
```

- Llama al método editarProducto() del servicio.
- Si la actualización es exitosa, redirecciona a la lista de productos.

4. Método irProductoLista() para redirigir después de guardar:

```
irProductoLista() {  
    this.enrutador.navigate(['/productos']);  
}
```

- Después de actualizar un producto, el usuario es **redirigido automáticamente** a la lista de productos.
-

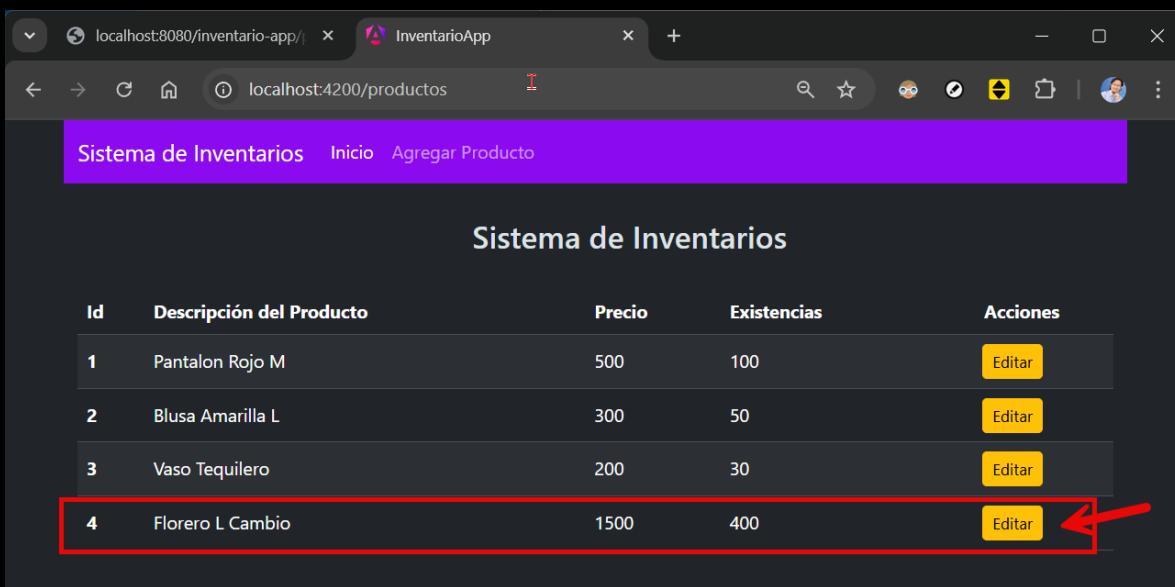
Paso 4: Probar la funcionalidad en el navegador

Pasos para probar

1. Ejecutar la aplicación Angular:

```
ng serve -o
```

2. Abrir el navegador en <http://localhost:4200/productos>



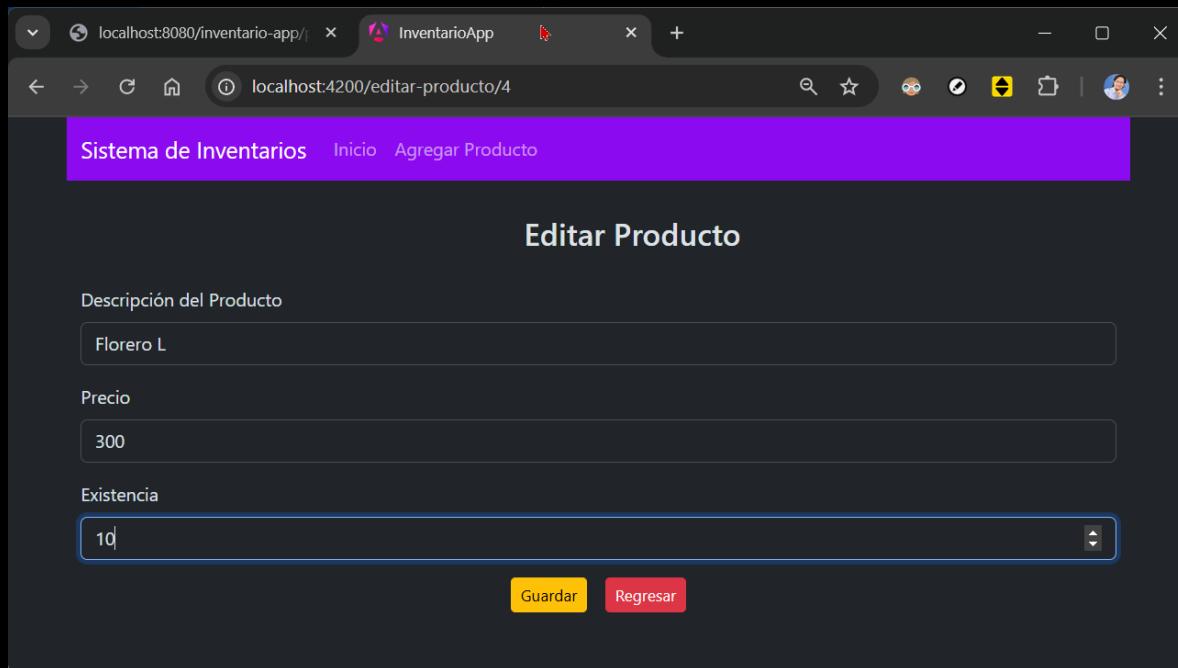
3. Seleccionar un producto y hacer clic en "Editar":

- La URL debe cambiar a:

```
http://localhost:4200/editar-producto/4
```

4. Modificar los valores en el formulario.

5. Presionar "Guardar".



Sistema de Inventarios Inicio Agregar Producto

Editar Producto

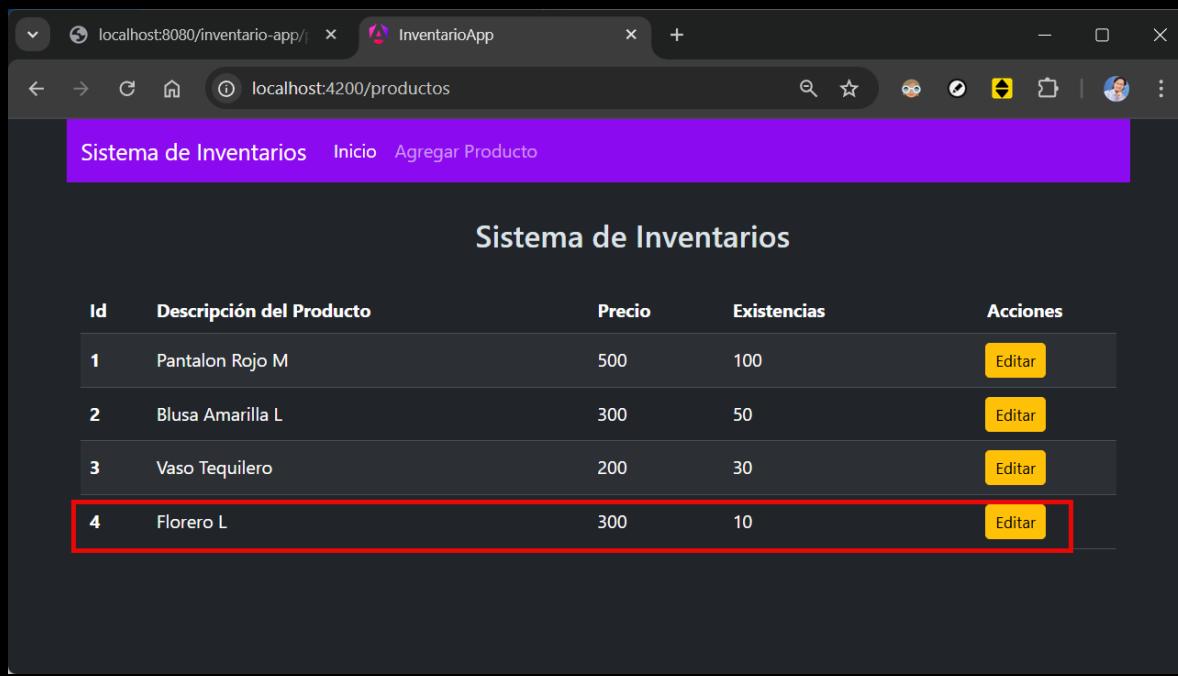
Descripción del Producto
Florero L

Precio
300

Existencia
10

Guardar Regresar

6. Serás redirigido automáticamente a la lista de productos y el producto debe aparecer actualizado.



Sistema de Inventarios Inicio Agregar Producto

Sistema de Inventarios

ID	Descripción del Producto	Precio	Existencias	Acciones
1	Pantalon Rojo M	500	100	<button>Editar</button>
2	Blusa Amarilla L	300	50	<button>Editar</button>
3	Vaso Tequilero	200	30	<button>Editar</button>
4	Florero L	300	10	<button>Editar</button>

Conclusión

- Se agregó el método `editarProducto()` en `producto.service.ts` para actualizar productos en el backend.
- Se modificó el formulario en `editar-producto.component.html` para reflejar la acción de "Guardar".
- Se implementó `onSubmit()`, `guardarProducto()`, y `irProductoLista()` en `editar-producto.component.ts` para actualizar y redirigir.
- Se probó la funcionalidad editando un producto y confirmando los cambios en la lista.

Con esto, hemos completado la funcionalidad de **editar productos** en Angular, integrando el backend de **Spring Boot**. 

Saludos!

Ing. Ubaldo Acosta

Fundador de [GlobalMentoring.com.mx](https://www.globalmentoring.com.mx)