

## Reutilización de Software

### 3.1. Introducción y Análisis Teórico

La reutilización de software es una estrategia fundamental en la ingeniería de software que busca maximizar el uso de componentes y activos existentes para construir nuevos sistemas. En lugar de desarrollar cada parte desde cero, la reutilización implica la incorporación de elementos preexistentes, como código, diseños, especificaciones, arquitecturas, documentación y hasta patrones de diseño. Su formalización se remonta a 1968 con Doug McIlroy, quien visualizó la producción de componentes de software de manera similar a los componentes electrónicos.

#### **Panorama de la Reutilización de Software**

El panorama actual de la reutilización de software es amplio y dinámico, impulsado por la creciente complejidad de los sistemas y la necesidad de agilizar los ciclos de desarrollo. Va más allá del simple "copiar y pegar código", abarcando desde la reutilización de código fuente hasta la adopción de componentes de software disponibles comercialmente (COTS - Commercial Off-The-Shelf) y el desarrollo de familias de aplicaciones.

#### **Tipos de Reutilización:**

- **Reutilización de código:** El enfoque más básico, donde se reutilizan fragmentos de código, funciones o módulos.
- **Reutilización de componentes:** Implica el uso de componentes de software autónomos y bien definidos, con interfaces claras.
- **Reutilización de frameworks y bibliotecas:** Colecciones de clases y funcionalidades predefinidas que proporcionan una estructura para el desarrollo de aplicaciones.
- **Reutilización de patrones de diseño:** Soluciones probadas a problemas comunes en el diseño de software, que se adaptan a diferentes contextos.
- **Reutilización de arquitecturas:** Reutilización de estructuras de alto nivel que definen la organización general de un sistema.
- **Reutilización de requisitos y diseños:** La reutilización de artefactos de fases tempranas del ciclo de vida del software, como especificaciones de requisitos, casos de uso y modelos de diseño.

#### **Beneficios de la Reutilización de Software**

La reutilización ofrece una serie de ventajas significativas que impactan positivamente el desarrollo de software:

- **Reducción de costos:** Al evitar la recreación de funcionalidades, se disminuyen los gastos de desarrollo y, a menudo, los de mantenimiento.
- **Aceleración del tiempo de desarrollo:** El uso de componentes probados permite construir sistemas más rápidamente, acortando los ciclos de vida del proyecto.
- **Mejora de la calidad y fiabilidad:** Los componentes reutilizados, al haber sido probados en sistemas anteriores, tienden a tener menos errores y fallos de implementación.
- **Incremento de la productividad:** Los equipos de desarrollo pueden concentrarse en funcionalidades nuevas y específicas, en lugar de "reinventar la rueda".

- **Mayor consistencia y familiaridad:** Al utilizar patrones y componentes estandarizados, se facilita el mantenimiento y la comprensión del software.
- **Uso efectivo de especialistas:** Permite a los especialistas desarrollar software reutilizable que encapsula su conocimiento, en lugar de replicar el mismo trabajo una y otra vez.
- **Mayor resiliencia y adaptabilidad:** Los sistemas construidos con componentes reutilizables suelen ser más fáciles de adaptar a cambios en los requisitos o tecnologías.

### Ejemplos Reales de Reutilización Exitosa en la Industria

La reutilización de software es una práctica común y exitosa en muchas industrias. Aquí algunos ejemplos notables:

- **Frameworks de desarrollo web (React, Angular, Vue.js, Django, Ruby on Rails):** Estos frameworks proporcionan una estructura y un conjunto de herramientas predefinidas para construir aplicaciones web. Los desarrolladores reutilizan componentes, patrones de diseño y lógica de negocio común, acelerando drásticamente el desarrollo y asegurando una mayor consistencia.
- **Sistemas de gestión de bases de datos (PostgreSQL, MySQL, Oracle):** Las empresas reutilizan estos sistemas para almacenar y gestionar sus datos. Los desarrolladores interactúan con ellos a través de lenguajes de consulta estándar (SQL), sin tener que construir su propio motor de base de datos.
- **Plataformas de comercio electrónico (Shopify, Magento, WooCommerce):** Estas plataformas ofrecen funcionalidades preconstruidas para tiendas online, como gestión de productos, carritos de compra, pasarelas de pago y sistemas de envío. Las empresas reutilizan estas plataformas y las adaptan a sus necesidades específicas, en lugar de desarrollar un sistema de comercio electrónico desde cero.
- **Servicios en la nube (AWS Lambda, Azure Functions, Google Cloud Functions):** Los servicios "sin servidor" permiten a los desarrolladores reutilizar funcionalidades a través de pequeñas funciones de código que se ejecutan bajo demanda, sin preocuparse por la infraestructura subyacente.
- **APIs de terceros (Google Maps API, Stripe API, Twilio API):** Muchas empresas integran funcionalidades de terceros en sus aplicaciones a través de APIs. Por ejemplo, una aplicación de entrega de alimentos puede reutilizar la API de Google Maps para mostrar ubicaciones y calcular rutas, o la API de Stripe para procesar pagos.

### 3.2. Exploración de Frameworks de Aplicación

**Framework seleccionado: Spring Boot (Java)**

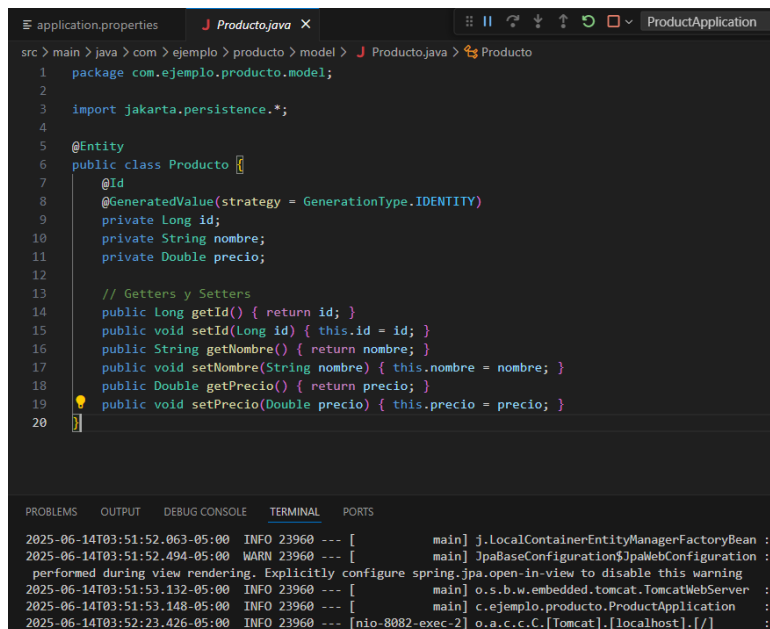
**Ventajas de Spring Boot para su reutilización:**

- Modularidad (uso de dependencias específicas).
- Inversión de control (IoC) y contenedores de inyección de dependencias.
- Estandarización de patrones como MVC, acceso a datos, etc.
- Integración sencilla de bibliotecas y COTS.

## Aplicación desarrollada:

Tenemos una API REST para gestionar productos (Producto) reutilizando controladores, servicios y repositorios.

Ejemplo de código reutilizable:



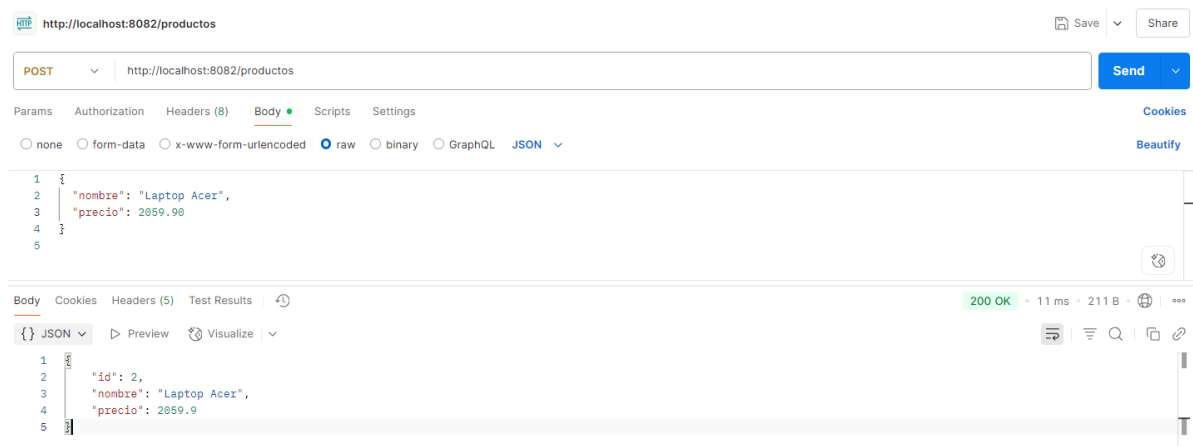
```
src > main > java > com > ejemplo > producto > model > J Producto.java > Producto
1 package com.ejemplo.producto.model;
2
3 import jakarta.persistence.*;
4
5 @Entity
6 public class Producto {
7     @Id
8     @GeneratedValue(strategy = GenerationType.IDENTITY)
9     private Long id;
10    private String nombre;
11    private Double precio;
12
13    // Getters y Setters
14    public Long getId() { return id; }
15    public void setId(Long id) { this.id = id; }
16    public String getNombre() { return nombre; }
17    public void setNombre(String nombre) { this.nombre = nombre; }
18    public Double getPrecio() { return precio; }
19    public void setPrecio(Double precio) { this.precio = precio; }
20 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
2025-06-14T03:51:52.063-05:00 INFO 23960 --- [main] j.LocalContainerEntityManagerFactoryBean :
2025-06-14T03:51:52.494-05:00 WARN 23960 --- [main] JpaBaseConfiguration$JpaWebConfiguration :
performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-06-14T03:51:53.132-05:00 INFO 23960 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer :
2025-06-14T03:51:53.148-05:00 INFO 23960 --- [main] c.ejemplo.producto.ProductApplication :
2025-06-14T03:52:23.426-05:00 INFO 23960 --- [nio-8082-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/] :
```

- Creamos una pequeña app CRUD de productos.
- Se puede separar el código en capas reutilizables (controlador, servicio, entidad, repositorio).

## Ejemplo práctico:

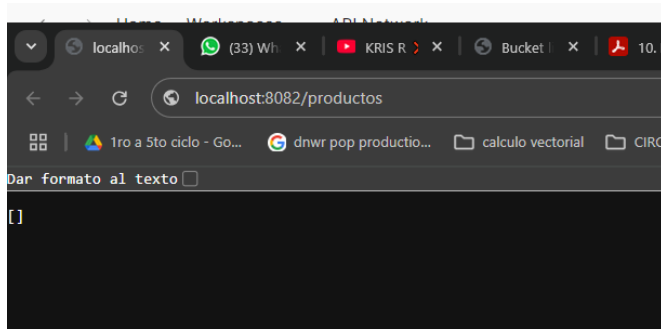


Este Producto se puede reutilizar en varias apps: tiendas, inventarios, etc.

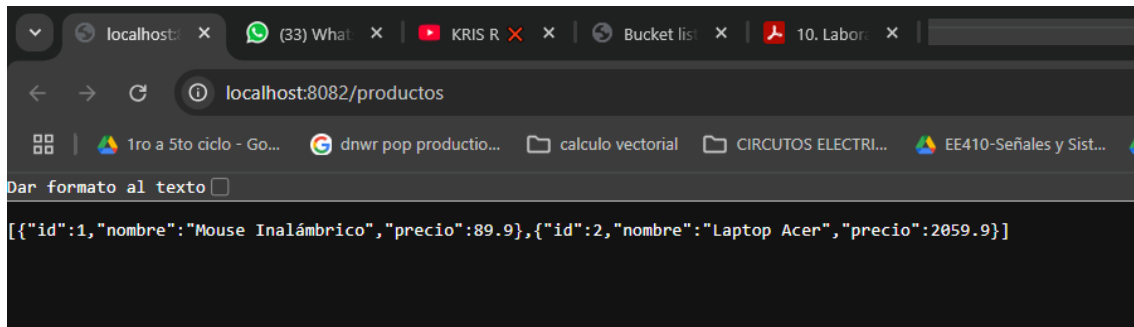
Donde el url es: <http://localhost:8082/productos>

Corriendo el código:

1) Está vacío.



2) Agregaremos 2 productos.



Creamos una API REST con Spring Boot que implementa un CRUD para la clase Producto. La app permite que otros sistemas (como frontends o apps móviles) se conecten para gestionar productos, todo reutilizando componentes del framework.

### **3.3. Diseño de Líneas de Productos de Software**

**Dominio:** Sistemas de Gestión Académica

**Componentes comunes identificados:**

- Gestión de usuarios
- Gestión de cursos
- Gestión de notas
- Reportes académicos

Estos componentes pueden ser reutilizados en productos como:

- Sistema para universidades
- Software para institutos técnicos

### **3.4. Reutilización de Productos COTS**

**Producto COTS:**

**Keycloak (gestión de usuarios y autenticación)**

**Ventajas del COTS:**

- 1) Ya está probado y documentado.
- 2) Ahorra semanas de trabajo.
- 3) Muchos tienen soporte comunitario o empresarial.

**Ventajas del Keycloak:**

- Soporte para OAuth2, JWT y SSO.
- Panel de administración listo para usar.
- Escalable y seguro.

**Proceso de integración con Spring Boot:**

1. Agregar dependencia de Keycloak.
2. Configurar el `application.properties` con la URL del servidor Keycloak.
3. Proteger rutas con anotaciones y configuración de seguridad.

**Resultado esperado:**

Usar el Spring Boot y configurar Keycloak para proteger los endpoints. Así se lograría proteger el endpoint `/productos` y manejar la autenticación de forma externa. Ya que solo se permitiría que usuarios autenticados vean los productos.

**3.5. Documentación y Retroalimentación****Beneficios observados:**

- Reducción significativa del código personalizado.
- Más tiempo dedicado a lógica de negocio que a configuración.
- Uso de patrones ya establecidos.

**Desafíos encontrados:**

- Curva de aprendizaje de frameworks complejos.
- Dificultad de configurar productos COTS.
- Ajuste de componentes a requisitos específicos.