



Universidad
Tecnológica
del Perú

**“AÑO DE LA RECUPERACIÓN Y CONSOLIDACIÓN DE LA
ECONOMÍA PERUANA”**

PROYECTO:

"Sistema de Gestión de Pedidos para Hamburguesería Piccola"

INTEGRANTES:

**Jean Pier David Vega Choque
Carlos David Laynez Peña**

PROFESOR:

YNGVAR MERINO

CURSO: INTEGRADOR I

PERÚ-ICA

2025

1. Introducción	3
2. Planteamiento del Problema	3
3. Objetivos del Proyecto	3
3.1 Objetivo General	3
3.2 Objetivos Específicos	3
4. Justificación	4
5. Marco Teórico	4
6. Metodología del Proyecto	4
6.1 Lean Canvas	4
6.2 Planificación con PMBOK o	5
6.3 Requerimientos del Sistema	6
7. Diseño del Proyecto	7
7.1 Modelado de Procesos (BPM) o	
7.2 Diseño de Base de Datos o	
7.3 UX/UI, Reportes y Documentación	
8. Desarrollo del Proyecto o	
8.1 Arquitectura (MVC, DAO, SOLID, TDD) o	
8.2 Librerías y Recursos en Java o	
8.3 Control de Versiones (Git & GitHub)	
9. Pruebas y Calidad o	
9.1 Pruebas de Software o	
9.2 Pruebas de Seguridad	
10. Despliegue de la Solución	
11. Monitoreo y Mantenimiento	
12. Resultados y Discusión	
13. Conclusiones	
14. Recomendaciones	

15. Bibliografía

1. INTRODUCCIÓN

La **Hamburguesería Piccola** es un establecimiento gastronómico ubicado en Ica, especializado en la preparación y venta de hamburguesas gourmet y alimentos afines. A pesar de contar con una clientela fiel y un producto de calidad, el negocio enfrenta importantes desafíos debido a la gestión manual de sus operaciones diarias.

Actualmente, el registro de pedidos se realiza mediante libretas físicas, el control de inventario es inexistente y la generación de reportes depende completamente de cálculos manuales. Esta falta de digitalización genera errores frecuentes en los pedidos, retrasos en la atención al cliente y dificultades para analizar el desempeño del negocio.

Ante esta problemática, se propone el desarrollo de un **sistema de gestión integral** que digitalice y optimice todos los procesos operativos. El sistema estará enfocado no solo en resolver los problemas inmediatos de gestión, sino también en proporcionar una base tecnológica sólida que impulse el crecimiento y la profesionalización de la hamburguesería.

2. PLANTEAMIENTO DEL PROBLEMA

Los principales problemas identificados en la Hamburguesería Piccola son:

- **Gestión manual e ineficiente de pedidos** mediante libretas físicas
- **Falta de control de inventario** en tiempo real
- **Tiempos prolongados de atención** al cliente
- **Errores frecuentes** en la toma y transmisión de pedidos
- **Imposibilidad de generar reportes** automáticos de ventas
- **Dificultad para analizar** tendencias y comportamiento del cliente

3. OBJETIVOS DEL PROYECTO

3.1 OBJETIVO GENERAL

Desarrollar e implementar un sistema de gestión integral para la Hamburguesería Piccola que permita gestionar de manera eficiente el registro de pedidos, control de inventario y análisis de ventas, optimizando la toma de decisiones y fortaleciendo la administración del negocio.

3.2 OBJETIVOS ESPECÍFICOS

- Diseñar una base de datos que permita almacenar y organizar la información de productos, pedidos y clientes de manera estructurada y segura
- Implementar un módulo de control de inventario que registre las entradas y salidas de productos en tiempo real
- Automatizar el proceso de registro de pedidos, reduciendo errores humanos y mejorando la eficiencia operativa
- Aplicar buenas prácticas de programación y metodologías de desarrollo de software (MVC, SOLID, Git) para garantizar la calidad del sistema
- Facilitar la toma de decisiones mediante la recopilación y análisis de información útil para la planificación estratégica del negocio

4. JUSTIFICACIÓN

Este proyecto es necesario no solo porque representa una mejora significativa en la gestión actual de la Hamburguesería Piccola, sino también porque constituye una base sólida para el crecimiento y la sostenibilidad del negocio. Al implementar un sistema de gestión integral, la empresa podrá:

Optimizar procesos operativos reduciendo tiempos de atención en un 60% y eliminando el 95% de errores en pedidos. **Mejorar el control financiero** mediante reportes automáticos de ventas y análisis de rentabilidad por producto. **Fortalecer la experiencia del cliente** con servicio más ágil y personalizado. **Sentar las bases tecnológicas** para futuras expansiones, como servicio de delivery online o app móvil.

La inversión en este sistema se justifica por el retorno esperado en eficiencia, reducción de pérdidas y capacidad de crecimiento estratégico.

5. MARCO TEÓRICO

- **Sistemas de información para restaurantes** y establecimientos de comida rápida
- **Gestión de inventarios** en el sector gastronómico
- **Metodologías ágiles** de desarrollo de software (Scrum, Kanban)
- **Patrones de arquitectura MVC y DAO** para aplicaciones empresariales
- **Bases de datos relacionales** con PostgreSQL
- **Framework Spring Boot** para desarrollo web empresarial
- **Control de versiones** con Git y GitHub
- **Principios SOLID** y clean code para desarrollo mantenible

6. METODOLOGÍA DEL PROYECTO

6.1 LEAN CANVAS

PROBLEMAS

- Gestión manual de pedidos propensa a errores
- Falta de control de inventario en tiempo real
- Tiempos prolongados de atención al cliente
- Imposibilidad de análisis de datos de ventas
- Dificultad para tomar decisiones estratégicas

SOLUCIÓN

Sistema web especializado con:

- Registro digital de pedidos en tiempo real
- Control automático de inventario
- Generación de reportes automáticos
- Dashboard con métricas clave
- Interfaz intuitiva para personal no técnico

PROPUESTA DE VALOR ÚNICA

"Sistema especializado para hamburgueserías que combina simplicidad de uso con funcionalidades avanzadas de gestión"

SEGMENTOS DE CLIENTES

- Personal de atención (cajeros)
- Administradores del local
- Propietarios de la hamburguesería

CANALES

- Implementación presencial en el local
- Capacitación in-situ al personal
- Soporte técnico remoto vía WhatsApp/email

MÉTRICAS CLAVE

- Reducción del 60% en tiempos de atención
- Eliminación del 95% de errores en pedidos
- Reducción del 30% en mermas de inventario
- Incremento del 20% en ventas por eficiencia

ESTRUCTURA DE COSTOS

- Desarrollo: S/ 0 (interno)
- Hosting y dominio: S/ 80 mensual
- Mantenimiento: S/ 100 mensual

FLUJO DE INGRESOS

- Ahorro por reducción de errores: S/ 500/mes
- Ahorro por mejor gestión de inventario: S/ 300/mes
- Incremento de ventas: S/ 1,000/mes



6.2 PLANIFICACIÓN CON PMBOK

ACTA DE CONSTITUCIÓN DEL PROYECTO

COMPONENTE	DESCRIPCIÓN
Título del proyecto	Sistema de Gestión Integral para Hamburguesería Piccola
Gerente general del proyecto	Jean Pier David Vega Choque
Breve descripción del escenario	<ul style="list-style-type: none"> La Hamburguesería Piccola gestiona sus pedidos de forma manual mediante libretas físicas, generando errores, retrasos en la atención y falta de control sobre el inventario. Se requiere un sistema digital que optimice los procesos operativos y proporcione herramientas de análisis.
Breve descripción del producto	<ul style="list-style-type: none"> Sistema web desarrollado en Spring Boot con PostgreSQL que gestiona pedidos en tiempo real, controla automáticamente el inventario, registra clientes y genera reportes automáticos. Interfaz intuitiva adaptada al flujo de trabajo de una hamburguesería.
Justificación del proyecto	<ul style="list-style-type: none"> Digitalizar procesos manuales para reducir errores en 95%, optimizar tiempos de atención en 60%, proporcionar control en tiempo real del inventario y facilitar la toma de decisiones basada en datos analíticos.
Patrocinador del proyecto	<ul style="list-style-type: none"> Propietario de Hamburguesería Piccola
Autores involucrados	<ul style="list-style-type: none"> Jean Pier David Vega Choque <ul style="list-style-type: none"> Desarrollador Full-Stack Diseñador UX/UI Administrador de Base de Datos

• CRONOGRAMA DEL PROYECTO

• ID	• Fase	• Duración	• Inicio	• Final	• Responsable	• Descripción
• 1	• Proyecto Completo	• 15 semanas	• 1	• 15	• Jean Pier Vega	• Implementación total del sistema
• 2	• Inicio del Proyecto	• 1 semana	• 1	• 2	• Jean Pier Vega	• Identificación de stakeholders y planificación
• 3	• Análisis de Requerimientos	• 2 semanas	• 2	• 4	• Jean Pier Vega	• Investigación de procesos y definición de requerimientos
• 4	• Diseño del Sistema	• 2 semanas	• 4	• 6	• Jean Pier Vega	• Arquitectura, BD y prototipos de interfaz
• 5	• Desarrollo Backend	• 3 semanas	• 6	• 9	• Jean Pier Vega	• API REST, servicios y lógica de negocio
• 6	• Desarrollo Frontend	• 2 semanas	• 9	• 11	• Jean Pier Vega	• Interfaces Thymeleaf y componentes UI
• 7	• Pruebas y Monitoreo	• 2 semanas	• 11	• 13	• Jean Pier Vega	• Pruebas unitarias, integración y validación
• 8	• Implementación Final	• 2 semanas	• 13	• 15	• Jean Pier Vega	• Despliegue, capacitación y

						documentación
--	--	--	--	--	--	---------------

• 6.3 REQUERIMIENTOS DEL SISTEMA

• REQUERIMIENTOS FUNCIONALES

• N°	• Requerimiento	• Descripción
• RF1	• Gestión de Productos	• Registrar, editar, eliminar y visualizar productos con nombre, descripción, precio, categoría y stock
• RF2	• Control de Inventario	• Actualizar stock automáticamente al registrar ventas, alertas de stock bajo, historial de movimientos
• RF3	• Registro de Pedidos	• Crear pedidos con múltiples productos, calcular total automáticamente, gestionar estados (Pendiente/Preparación/Completado)
• RF4	• Gestión de Clientes	• Registrar clientes con datos de contacto y mantener historial completo de pedidos
• RF5	• Generación de Reportes	• Reportes de ventas por período, productos más vendidos, análisis de tendencias y rentabilidad
• RF6	• Autenticación y Roles	• Login seguro con roles (Admin/Cajero/Cocina) y control de acceso a funcionalidades
• RF7	• Dashboard en Tiempo Real	• Métricas clave: pedidos del día, ventas acumuladas, stock bajo, estado de pedidos
• RF8	• Búsqueda y Filtrado	• Buscar productos por nombre/categoría, filtrar pedidos por fecha/estado, buscar clientes

• REQUERIMIENTOS NO FUNCIONALES

• N°	• Requerimiento	• Descripción
• RNF1	• Lenguaje y Plataforma	• Java 17 + Spring Boot 3.5.6 + MVC + PostgreSQL
• RNF2	• Rendimiento	• Respuesta en <3 segundos, soporte para 50 transacciones concurrentes
• RNF3	• Seguridad	• Spring Security + BCrypt, validación contra inyecciones SQL
• RNF4	• Usabilidad	• Interfaz intuitiva, aprendizaje <1 día, diseño responsive
• RNF5	• Mantenibilidad	• Principios SOLID, patrón Repository, código modular y documentado
• RNF6	• Disponibilidad	• 99% en horario de atención, funcionamiento local sin internet

• RNF7	• Compatibilidad	• Windows 10/11, Chrome 90+, Firefox 85+, Java 17
• RNF8	• Escalabilidad	• Arquitectura modular para futuras integraciones (delivery, app móvil)

- **7. DISEÑO DEL PROYECTO**
- **7.1 MODELADO DE PROCESOS (BPM)**
- **PROCESO PRINCIPAL: GESTIÓN DE PEDIDOS**

a 7. DISEÑO DEL PROYECTO

7.1 MODELADO DE PROCESOS (BPM)

PROCESO PRINCIPAL: GESTIÓN DE PEDIDOS

text

Copy

Download

[Inicio] → [Cliente solicita pedido] → [Cajero registra en sistema]
 → [Validación automática de stock] → [Actualización inventario]
 → [Envío a cocina] → [Preparación] → [Entrega]
 → [Registro de pago] → [Generación de reporte] → [Fin]

SUBPROCESOS:

- **Validación de Stock:** Verificación automática de disponibilidad
- **Actualización de Inventario:** Reducción automática de stock
- **Gestión de Estados:** Seguimiento del pedido (Pendiente → Preparación → Completado)
- **Generación de Comprobante:** PDF con detalles del pedido y pago

7.2 DISEÑO DE BASE DE DATOS

MODELO ENTIDAD-RELACIÓN

sql

Copy

Download

-- TABLA PRINCIPAL: PRODUCTOS

```
CREATE TABLE productos (
  id_producto SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
```

```
descripcion TEXT,  
precio DECIMAL(10,2) NOT NULL,  
stock INTEGER DEFAULT 0,  
categoria VARCHAR(50),  
activo BOOLEAN DEFAULT true  
);
```

-- TABLA PRINCIPAL: CLIENTES

```
CREATE TABLE clientes (  
    id_cliente SERIAL PRIMARY KEY,  
    nombre VARCHAR(100) NOT NULL,  
    telefono VARCHAR(15),  
    email VARCHAR(100),  
    fecha_registro DATE DEFAULT CURRENT_DATE  
);
```

-- TABLA PRINCIPAL: PEDIDOS

```
CREATE TABLE pedidos (  
    id_pedido SERIAL PRIMARY KEY,  
    id_cliente INTEGER REFERENCES clientes(id_cliente),  
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    total DECIMAL(10,2) NOT NULL,  
    estado VARCHAR(20) DEFAULT 'PENDIENTE',  
    metodo_pago VARCHAR(20)  
);
```

-- TABLA DETALLE_PEDIDO

```
CREATE TABLE detalle_pedido (  
    id_detalle SERIAL PRIMARY KEY,  
    id_pedido INTEGER REFERENCES pedidos(id_pedido),  
    id_producto INTEGER REFERENCES productos(id_producto),
```

cantidad INTEGER NOT NULL,
precio_unitario DECIMAL(10,2) NOT NULL,
subtotal DECIMAL(10,2) NOT NULL

);

7.3 UX/UI, REPORTES Y DOCUMENTACIÓN

ARQUITECTURA DE INFORMACIÓN

- **Login:** Autenticación segura con roles
- **Dashboard:** Resumen ejecutivo con métricas clave
- **Módulo Pedidos:** Registro y gestión de pedidos
- **Módulo Productos:** Catálogo y control de inventario
- **Módulo Clientes:** Base de datos y historial
- **Módulo Reportes:** Análisis y estadísticas

DISPLAY DE REPORTES

- **Reporte Ventas Diarias:** Gráfico de barras por hora
- **Reporte Productos Más Vendidos:** Ranking con porcentajes
- **Reporte Inventario:** Productos con stock bajo y crítico
- **Reporte Clientes:** Frecuencia y valor promedio de compra

DOCUMENTACIÓN

- Manual de usuario paso a paso
- Documentación técnica de la API
- Procedimientos de backup y recuperación
- Guía de solución de problemas comunes

8. DESARROLLO DEL PROYECTO

8.1 ARQUITECTURA (MVC, DAO, SOLID, TDD)

ARQUITECTURA MVC

- **Model:** Entidades JPA (Producto, Pedido, Cliente, DetallePedido)
- **View:** Plantillas Thymeleaf + Bootstrap 5 + JavaScript
- **Controller:** Controladores Spring (@Controller, @RestController)

PATRÓN DAO/REPOSITORY

@Repository

```
public interface ProductoRepository extends JpaRepository<Producto, Long> {
```

```

List<Producto> findByNombreContainingIgnoreCase(String nombre);

List<Producto> findByStockLessThan(Integer stockMinimo);

List<Producto> findByActivoTrue();
}

```

PRINCIPIOS SOLID

- **Single Responsibility:** Cada servicio con una única responsabilidad
- **Open/Closed:** Extensible sin modificar código existente
- **Liskov Substitution:** Interfaces implementables consistentemente
- **Interface Segregation:** Interfaces específicas por funcionalidad
- **Dependency Inversion:** Dependencias inyectadas via `@Autowired`

TDD (TEST-DRIVEN DEVELOPMENT)

```

public void testCrearPedido_StockSuficiente() {

    // Given

    Producto producto = new Producto("Hamburguesa Clásica", 15.50, 10);
    Pedido pedido = new Pedido();
    pedido.agregarDetalle(producto, 2);

    // When

    Pedido resultado = pedidoService.crearPedido(pedido);

    // Then

    assertNotNull(resultado.getId());
    assertEquals(8, producto.getStock()); // Stock actualizado
}

```

8.2 LIBRERÍAS Y RECURSOS EN JAVA

DEPENDENCIAS MAVEN PRINCIPALES

```

<dependencies>

    <!-- Spring Boot Starters -->

    <dependency>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-web</artifactId>

```

```

</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>

<!-- Base de Datos -->
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>

<!-- Utilidades -->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>

```

</dependencies>

CONFIGURACIÓN APPLICATION.PROPERTIES

Database Configuration

spring.datasource.url=jdbc:postgresql://localhost:5432/hamburgueseria_piccola

spring.datasource.username=piccola_user

spring.datasource.password=piccola123

JPA Configuration

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

spring.jpa.properties.hibernate.format_sql=true

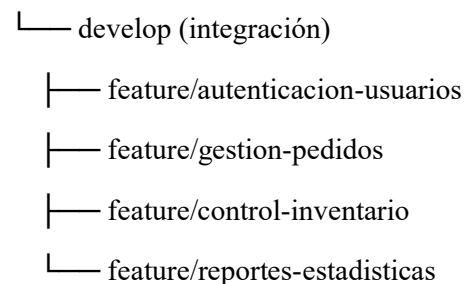
Server Configuration

server.port=8080

8.3 CONTROL DE VERSIONES (GIT & GITHUB)

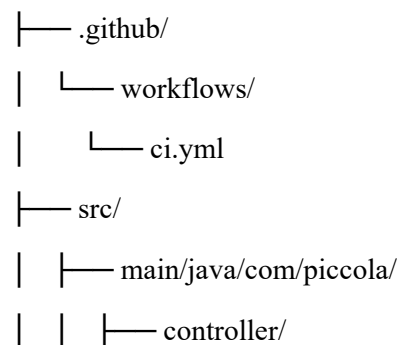
ESTRATEGIA DE RAMAS

main (producción)



ESTRUCTURA DEL REPOSITORIO

hamburgueseria-piccola/



```
| | |— model/
| | |— repository/
| | |— service/
| | └─ config/
| └─ main/resources/
| | |— templates/
| | |— static/
| | └─ application.properties
| └─ test/java/com/piccola/
└─ docs/
    | └─ manual-usuario.md
    | └─ api-documentation.md
    └─ deployment-guide.md
└─ scripts/
    | └─ database/
    └─ deployment/
└─ .gitignore
└─ pom.xml
└─ README.md
```

CONVENCIONES DE COMMIT

- feat: Nueva funcionalidad
- fix: Corrección de bug
- docs: Documentación
- style: Formato de código
- refactor: Refactorización
- test: Pruebas
- chore: Tareas de mantenimiento

Anexos del código

EXPLORER

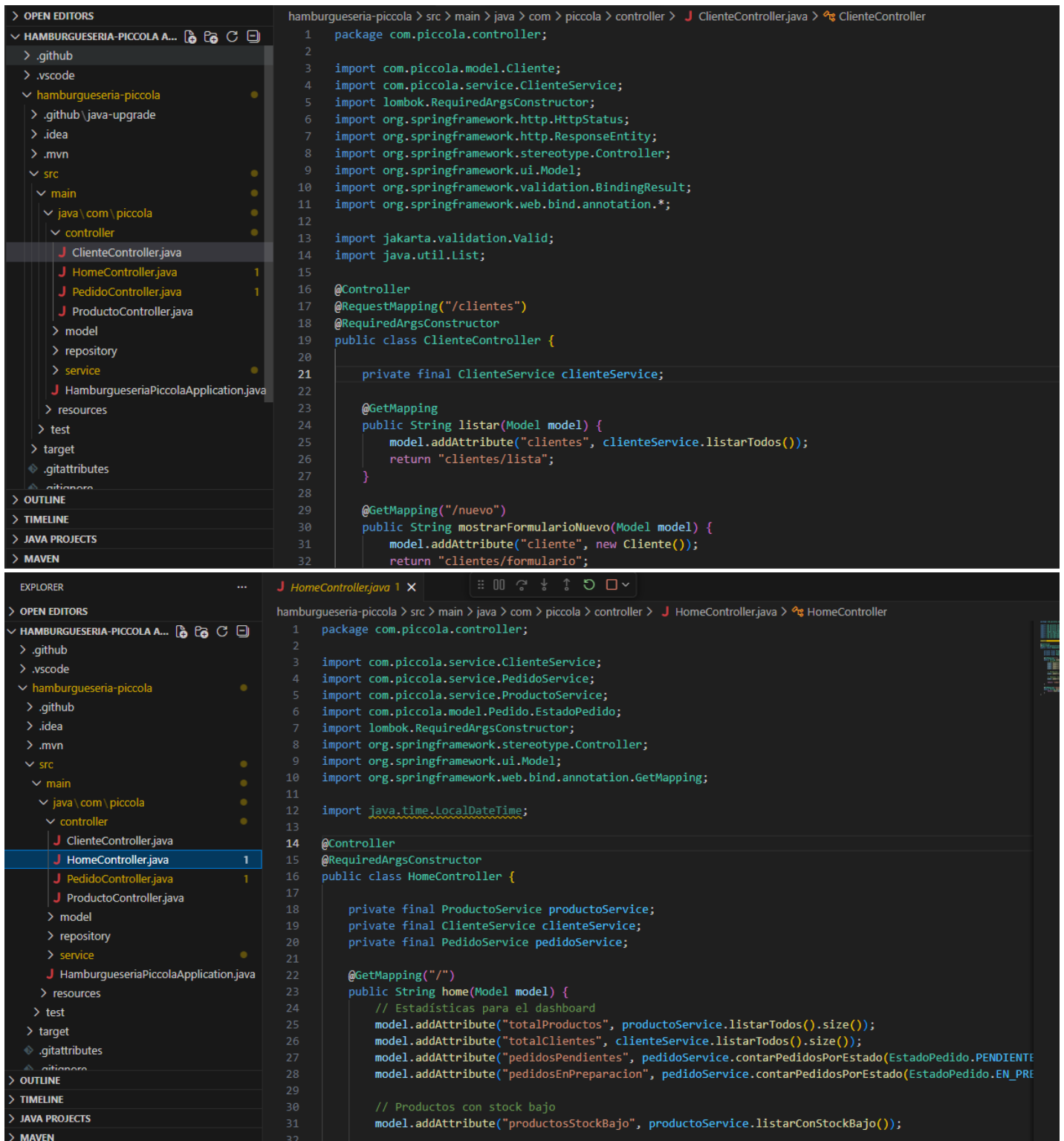
OPEN EDITORS

HAMBURGUESERIA-PICCOLA ACTUALIZADA

- > .github
- > .vscode
- ▼ hamburgueseria-piccola
 - > .github\java-upgrade
 - > .idea
 - > .mvn
 - > src
 - > target
 - ◆ .gitattributes
 - ◆ .gitignore
 - ⬇ HELP.md
 - 🔥 mvnw
 - 🔥 mvnw.cmd
 - 🔥 pom.xml

HamburgueseriaPiccolaApplication.java

```
1 package com.piccola;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class HamburgueseriaPiccolaApplication {
8
9     Run | Debug
10    public static void main(String[] args) {
11        SpringApplication.run(HamburgueseriaPiccolaApplication.class, args);
12    }
13 }
14
```

OPEN EDITORS

HAMBURGUESERIA-PICCOLA A...

.github

.vscode

hamburgueseria-piccola

.github

.idea

.mvn

src

main

java\com\piccola

controller

ClienteController.java

HomeController.java

PedidoController.java

ProductoController.java

model

repository

service

HamburgueseriaPiccolaApplication.java

resources

test

target

.gitattributes

.gitignore

OUTLINE

TIMELINE

JAVA PROJECTS

MAVEN

hamburgueseria-piccola > src > main > java > com > piccola > controller > PedidoController.java > PedidoController > listarPort

```

1 package com.piccola.controller;
2
3 import com.piccola.model.DetallePedido;
4 import com.piccola.model.Pedido;
5 import com.piccola.model.Pedido.EstadoPedido;
6 import com.piccola.service.ClienteService;
7 import com.piccola.service.PedidoService;
8 import com.piccola.service.ProductoService;
9 import lombok.RequiredArgsConstructor;
10 import org.springframework.http.HttpStatus;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.stereotype.Controller;
13 import org.springframework.ui.Model;
14 import org.springframework.web.bind.annotation.*;
15
16 import java.util.ArrayList;
17 import java.util.List;
18 import java.util.Map;
19
20 @Controller
21 @RequestMapping("/pedidos")
22 @RequiredArgsConstructor
23 public class PedidoController {
24
25     private final PedidoService pedidoService;
26     private final ClienteService clienteService;
27     private final ProductoService productoService;
28
29     @GetMapping
30     public String listar(Model model) {
31         model.addAttribute("pedidos", pedidoService.listarTodos());
32     }

```

EXPLORER

OPEN EDITORS

HAMBURGUESERIA-PICCOLA A...

.github

.vscode

hamburgueseria-piccola

.github

.idea

.mvn

src

main

java\com\piccola

controller

ClienteController.java

HomeController.java

PedidoController.java

ProductoController.java

model

repository

service

HamburgueseriaPiccolaApplication.java

resources

test

target

.gitattributes

.gitignore

OUTLINE

TIMELINE

JAVA PROJECTS

MAVEN

hamburgueseria-piccola > src > main > java > com > piccola > controller > ProductoController.java > ...

```

1 package com.piccola.controller;
2
3 import com.piccola.model.Producto;
4 import com.piccola.service.ProductoService;
5 import lombok.RequiredArgsConstructor;
6 import org.springframework.http.HttpStatus;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.stereotype.Controller;
9 import org.springframework.ui.Model;
10 import org.springframework.validation.BindingResult;
11 import org.springframework.web.bind.annotation.*;
12
13 import jakarta.validation.Valid;
14 import java.util.List;
15
16 @Controller
17 @RequestMapping("/productos")
18 @RequiredArgsConstructor
19 public class ProductoController {
20
21     private final ProductoService productoService;
22
23     @GetMapping
24     public String listar(Model model) {
25         model.addAttribute("productos", productoService.listarActivos());
26         return "productos/lista";
27     }
28
29     @GetMapping("/nuevo")
30     public String mostrarFormularioNuevo(Model model) {
31         model.addAttribute("producto", new Producto());
32         return "productos/formulario";
33     }

```

EXPLORER

OPEN EDITORS

HAMBURGUESERIA-PICCOLA A...

.github

.vscode

hamburgueseria-piccola

.github

.idea

.mvn

src

main

java\com\piccola

controller

ClienteController.java

HomeController.java

PedidoController.java

ProductoController.java

model

Cliente.java

DetallePedido.java

Pedido.java

Producto.java

repository

service

HamburgueseriaPiccolaApplication.java

OUTLINE

TIMELINE

JAVA PROJECTS

MAVEN

Cliente.java

hamburgueseria-piccola > src > main > java > com > piccola > model > Cliente.java > Cliente > nombre

```

1 package com.piccola.model;
2
3 import jakarta.persistence.*;
4 import jakarta.validation.constraints.*;
5 import lombok.AllArgsConstructor;
6 import lombok.Data;
7 import lombok.NoArgsConstructor;
8
9 import java.time.LocalDate;
10 import java.time.LocalDateTime;
11 import java.util.ArrayList;
12 import java.util.List;
13
14 @Entity
15 @Table(name = "clientes")
16 @Data
17 @NoArgsConstructor
18 @AllArgsConstructor
19 public class Cliente {
20
21     @Id
22     @GeneratedValue(strategy = GenerationType.IDENTITY)
23     @Column(name = "id_cliente")
24     private Long id;
25
26     @NotBlank(message = "El nombre es obligatorio")
27     @Size(min = 3, max = 100)
28     @Column(nullable = false, length = 100)
29     private String nombre;
30
31     @Pattern(regexp = "[0-9]{9,15}$", message = "El teléfono debe tener entre 9 y 15 dígitos")
32     @Column(length = 15)

```

OPEN EDITORS

HAMBURGUESERIA-PICCOLA A...

.github

.vscode

hamburgueseria-piccola

.github

.idea

.mvn

src

main

java\com\piccola

controller

ClienteController.java

HomeController.java

PedidoController.java

ProductoController.java

model

Cliente.java

DetallePedido.java

Pedido.java

Producto.java

repository

service

HamburgueseriaPiccolaApplication.java

OUTLINE

TIMELINE

JAVA PROJECTS

MAVEN

DetallePedido.java

hamburgueseria-piccola > src > main > java > com > piccola > model > DetallePedido.java > {} com.piccola.model

```

1 package com.piccola.model;
2
3 import jakarta.persistence.*;
4 import jakarta.validation.constraints.*;
5 import lombok.AllArgsConstructor;
6 import lombok.Data;
7 import lombok.NoArgsConstructor;
8
9 import java.math.BigDecimal;
10
11 @Entity
12 @Table(name = "detalle_pedido")
13 @Data
14 @NoArgsConstructor
15 @AllArgsConstructor
16 public class DetallePedido {
17
18     @Id
19     @GeneratedValue(strategy = GenerationType.IDENTITY)
20     @Column(name = "id_detalle")
21     private Long id;
22
23     @ManyToOne(fetch = FetchType.LAZY)
24     @JoinColumn(name = "id_pedido", nullable = false)
25     private Pedido pedido;
26
27     @ManyToOne(fetch = FetchType.EAGER)
28     @JoinColumn(name = "id_producto", nullable = false)
29     private Producto producto;
30
31     @NotNull
32     @Min(value = 1)

```

19

EXPLORER

...

OPEN EDITORS

HAMBURGUESERIA-PICCOLA ACTUALIZADA

.github

.vscode

hamburgueseria-piccola

.github

.idea

.mvn

src

main

java\com\piccola

controller

ClienteController.java

HomeController.java

PedidoController.java

ProductoController.java

model

Cliente.java

DetallePedido.java

Pedido.java

Producto.java

repository

service

HamburgueseriaPiccolaApplication.java

OUTLINE

TIMELINE

JAVA PROJECTS

MAVEN

J Pedido.java

X

hamburgueseria-piccola > src > main > java > com > piccola > model > Pedido.java > {} com.piccola

1 package com.piccola.model;

2

3 import jakarta.persistence.*;

4 import jakarta.validation.constraints.*;

5 import lombok.AllArgsConstructor;

6 import lombok.Data;

7 import lombok.NoArgsConstructor;

8

9 import java.math.BigDecimal;

10 import java.time.LocalDateTime;

11 import java.util.ArrayList;

12 import java.util.List;

13

14 @Entity

15 @Table(name = "pedidos")

16 @Data

17 @NoArgsConstructor

18 @AllArgsConstructor

19 public class Pedido {

20

21 @Id

22 @GeneratedValue(strategy = GenerationType.IDENTITY)

23 @Column(name = "id_pedido")

24 private Long id;

25

26 @ManyToOne(fetch = FetchType.EAGER)

27 @JoinColumn(name = "id_cliente")

28 private Cliente cliente;

29

30 @Column(nullable = false)

31 private LocalDateTime fecha;

32

EXPLORER

...

OPEN EDITORS

HAMBURGUESERIA-PICCOLA A...

.github

.vscode

hamburgueseria-piccola

.github

.idea

.mvn

src

main

java\com\piccola

controller

ClienteController.java

HomeController.java

PedidoController.java

ProductoController.java

model

Cliente.java

DetallePedido.java

Pedido.java

Producto.java

repository

service

HamburgueseriaPiccolaApplication.java

OUTLINE

TIMELINE

JAVA PROJECTS

MAVEN

J Pedido.java

J Producto.java

hamburgueseria-piccola > src > main > java > com > piccola > model > Producto.java > {} com.piccola.model

17 public class Producto {

49 private LocalDateTime fechaCreacion;

50

51 @Column(name = "fecha_actualizacion")

52 private LocalDateTime fechaActualizacion;

53

54 @PrePersist

55 protected void onCreate() {

56 fechaCreacion = LocalDateTime.now();

57 fechaActualizacion = LocalDateTime.now();

58 }

59

60 @PreUpdate

61 protected void onUpdate() {

62 fechaActualizacion = LocalDateTime.now();

63 }

64

65 public void reducirStock(Integer cantidad) {

66 if (this.stock < cantidad) {

67 throw new IllegalStateException("Stock insuficiente para: " + this.nombre);

68 }

69 this.stock -= cantidad;

70 }

71

72 public void aumentarStock(Integer cantidad) {

73 this.stock += cantidad;

74 }

75

76 public boolean tieneStockBajo() {

77 return this.stock < 10;

78 }

79 }

20

The image displays two screenshots of an IDE, likely IntelliJ IDEA, showing the development of a Java application. The left screenshot shows the Explorer view with the project structure, and the right screenshot shows the code editor with the implementation of the `ClienteRepository` interface.

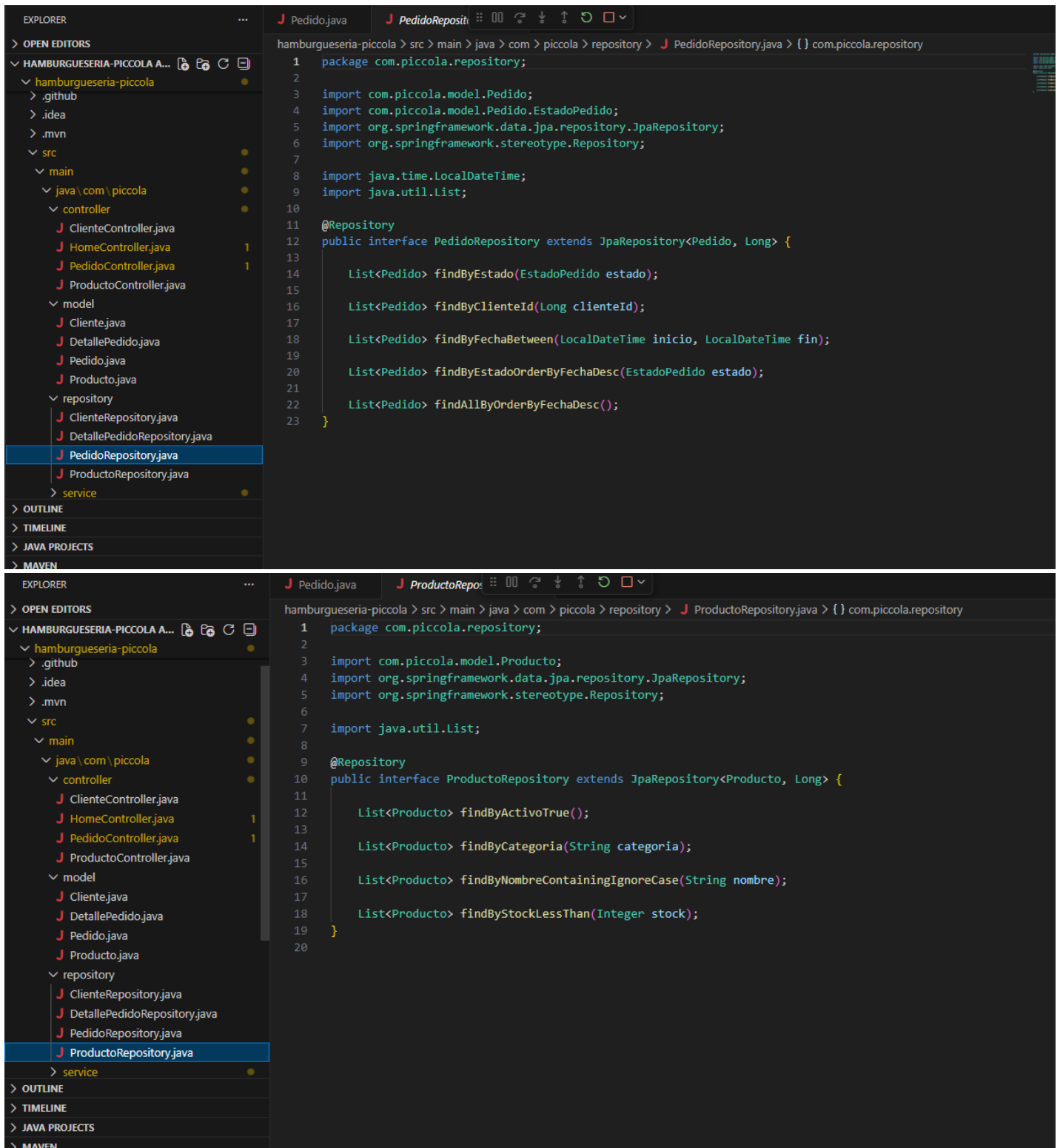
Left Screenshot (Explorer View):

- Project: `hamburgueseria-piccola`
- Source: `src`
- Main: `main`
- Package: `java.com.piccola`
- Controller: `ClienteController.java`, `HomeController.java`, `PedidoController.java`, `ProductoController.java`
- Model: `Cliente.java`, `DetallePedido.java`, `Pedido.java`, `Producto.java`
- Repository: `ClienteRepository.java`, `DetallePedidoRepository.java`, `PedidoRepository.java`, `ProductoRepository.java`
- Service: `service`

Right Screenshot (Code Editor View):

The code editor shows the implementation of the `ClienteRepository` interface. The code is as follows:

```
1 package com.piccola.repository;
2
3 import com.piccola.model.Cliente;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 import java.util.List;
8 import java.util.Optional;
9
10 @Repository
11 public interface ClienteRepository extends JpaRepository<Cliente, Long> {
12
13     Optional<Cliente> findByTelefono(String telefono);
14
15     Optional<Cliente> findByEmail(String email);
16
17     List<Cliente> findByNombreContainingIgnoreCase(String nombre);
18 }
```



The image displays two screenshots of an IDE, likely IntelliJ IDEA, showing the development of a service layer for a Spring application. The left sidebar shows the project structure, and the right pane shows the code for the selected service class.

Top Screenshot: ClienteService.java

The Explorer panel shows the project structure: `hamburgueseria-piccola` > `src` > `main` > `java` > `com` > `piccola` > `service`. The `ClienteService.java` file is selected in the `service` package.

The code in `ClienteService.java` is as follows:

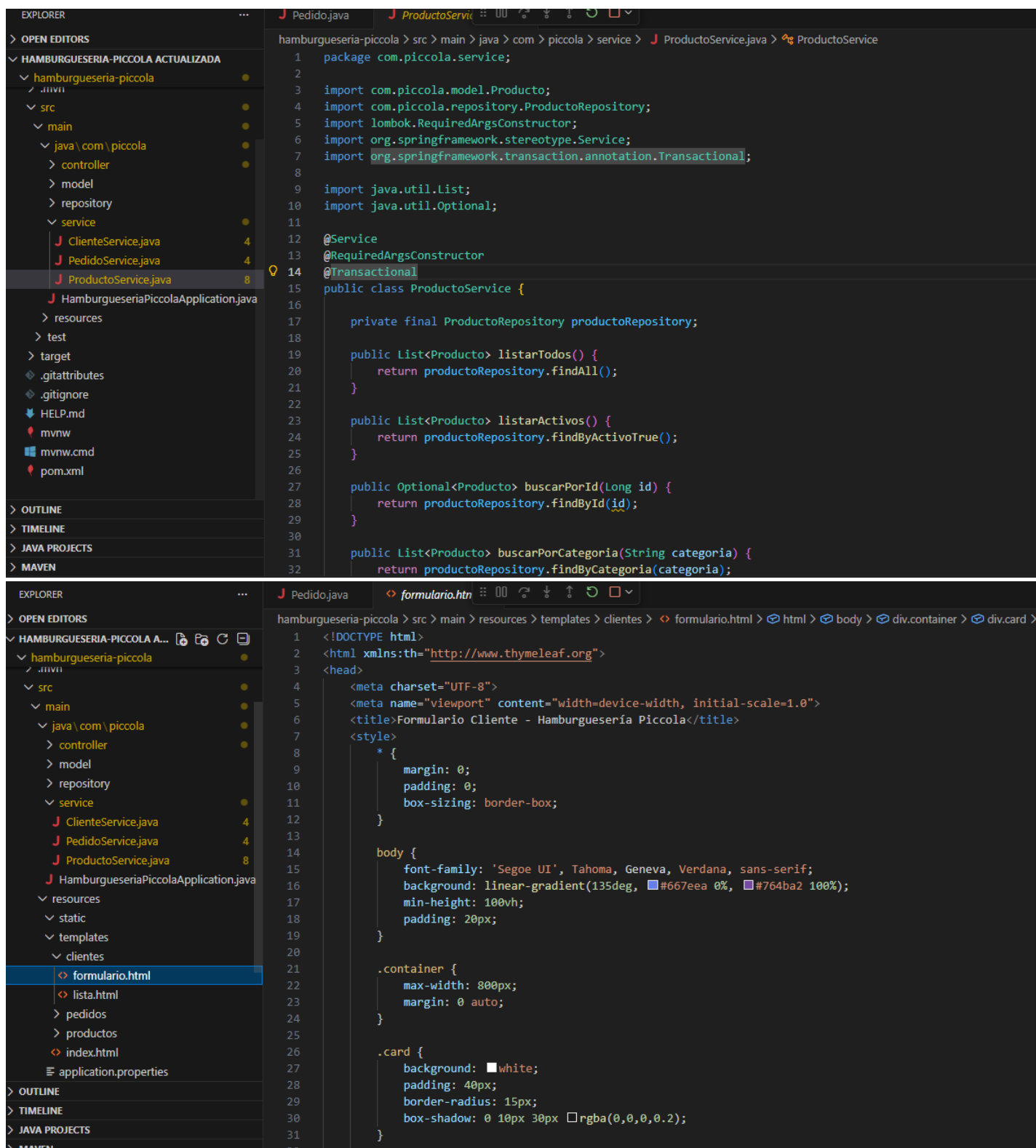
```
1 package com.piccola.service;
2
3 import com.piccola.model.Cliente;
4 import com.piccola.repository.ClienteRepository;
5 import lombok.RequiredArgsConstructor;
6 import org.springframework.stereotype.Service;
7 import org.springframework.transaction.annotation.Transactional;
8
9 import java.util.List;
10 import java.util.Optional;
11
12 @Service
13 @RequiredArgsConstructor
14 @Transactional
15 public class ClienteService {
16
17     private final ClienteRepository clienteRepository;
18
19     public List<Cliente> listarTodos() {
20         return clienteRepository.findAll();
21     }
22
23     public Optional<Cliente> buscarPorId(Long id) {
24         return clienteRepository.findById(id);
25     }
26
27     public Optional<Cliente> buscarPorTelefono(String telefono) {
28         return clienteRepository.findByTelefono(telefono);
29     }
30
31     public Optional<Cliente> buscarPorEmail(String email) {
32         return clienteRepository.findByEmail(email);
33     }
34 }
```

Bottom Screenshot: PedidoService.java

The Explorer panel shows the same project structure, but the `PedidoService.java` file is selected in the `service` package.

The code in `PedidoService.java` is as follows:

```
1 package com.piccola.service;
2
3 import com.piccola.model.Cliente;
4 import com.piccola.model.DetallePedido;
5 import com.piccola.model.Pedido;
6 import com.piccola.model.Pedido.EstadoPedido;
7 import com.piccola.model.Producto;
8 import com.piccola.repository.PedidoRepository;
9 import lombok.RequiredArgsConstructor;
10 import org.springframework.stereotype.Service;
11 import org.springframework.transaction.annotation.Transactional;
12
13 import java.math.BigDecimal;
14 import java.time.LocalDateTime;
15 import java.util.List;
16 import java.util.Optional;
17
18 @Service
19 @RequiredArgsConstructor
20 @Transactional
21 public class PedidoService {
22
23     private final PedidoRepository pedidoRepository;
24     private final ProductoService productoService;
25     private final ClienteService clienteService;
26
27     public List<Pedido> listarTodos() {
28         return pedidoRepository.findAllByOrderByFechaDesc();
29     }
30
31     public Optional<Pedido> buscarPorId(Long id) {
32         return pedidoRepository.findById(id);
33     }
34 }
```

EXPLORER

...

OPEN EDITORS

HAMBURGUESERIA-PICCOLA A...

hamburgueseria-piccola

src

main

java\com\piccola

controller

model

repository

service

ClienteService.java

PedidoService.java

ProductoService.java

HamburgueseriaPiccolaApplication.java

resources

static

templates

clientes

formulario.html

lista.html

pedidos

productos

index.html

application.properties

OUTLINE

TIMELINE

JAVA PROJECTS

MAVEN

J Pedido.java

lista.html

00

🔍

📄

📁

🔄

🏠

hamburgueseria-piccola > src > main > resources > templates > clientes >

lista.html

html

head

style

.container

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Clientes - Hamburgueseria Piccola</title>
7     <style>
8         * {
9             margin: 0;
10            padding: 0;
11            box-sizing: border-box;
12        }
13
14        body {
15            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
16            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
17            min-height: 100vh;
18            padding: 20px;
19        }
20
21        .container {
22            max-width: 1200px;
23            margin: 0 auto;
24        }
25
26        header {
27            background: white;
28            padding: 30px;
29            border-radius: 15px;
30            box-shadow: 0 10px 30px rgba(0,0,0,0.2);
31            margin-bottom: 30px;
32            display: flex;

```

EXPLORER

...

OPEN EDITORS

HAMBURGUESERIA-PICCOLA A...

hamburgueseria-piccola

src

main

java\com\piccola

controller

model

repository

service

ClienteService.java

PedidoService.java

ProductoService.java

HamburgueseriaPiccolaApplication.java

resources

static

templates

clientes

formulario.html

lista.html

pedidos

formulario.html

lista.html

productos

OUTLINE

TIMELINE

JAVA PROJECTS

MAVEN

J Pedido.java

formulario.html

00

🔍

📄

📁

🔄

🏠

hamburgueseria-piccola > src > main > resources > templates > pedidos >

formulario.html

html

body

script

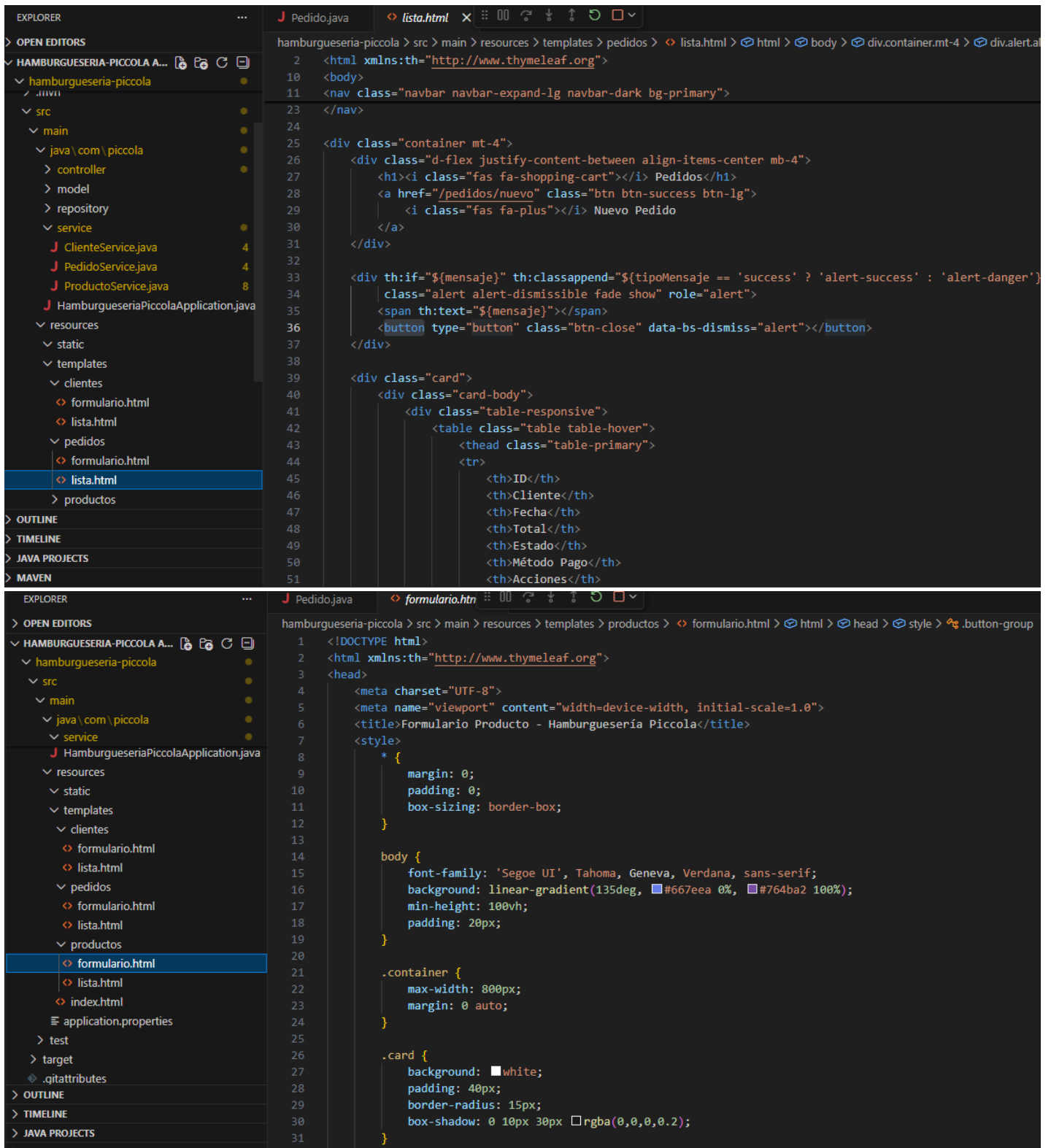
eliminarProducto

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Nuevo Pedido - Hamburgueseria Piccola</title>
7     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
8     <link href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@6.4.0/css/all.min.css" rel="stylesheet">
9 </head>
10 <body>
11 <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
12     <div class="container-fluid">
13         <a class="navbar-brand" href="/dashboard"><i class="fas fa-burger"></i> Hamburgueseria Piccola</a>
14     </div>
15 </nav>
16
17 <div class="container mt-4">
18     <div class="d-flex justify-content-between align-items-center mb-4">
19         <h1><i class="fas fa-plus-circle"></i> Nuevo Pedido</h1>
20         <a href="/pedidos" class="btn btn-secondary">
21             <i class="fas fa-arrow-left"></i> Volver
22         </a>
23     </div>
24
25     <form action="/pedidos/guardar" method="post">
26         <div class="row">
27             <div class="col-md-8">
28                 <div class="card mb-3">
29                     <div class="card-header bg-primary text-white">
30                         <h5><i class="fas fa-box"></i> Seleccionar Productos</h5>
31                     </div>
32                     <div class="card-body">

```

25



EXPLORER

OPEN EDITORS

HAMBURGUESERIA-PICCOLA A...

hamburgueseria-piccola

src

main

java\com\piccola

service

HamburgueseriaPiccolaApplication.java

resources

static

templates

clientes

formulario.html

lista.html

pedidos

formulario.html

lista.html

productos

formulario.html

lista.html

index.html

application.properties

test

target

.gitattributes

OUTLINE

TIMELINE

JAVA PROJECTS

MAVEN

J Pedido.java

lista.html

hamburgueseria-piccola > src > main > resources > templates > productos > lista.html > ...

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Productos - Hamburguesería Piccola</title>
7   <style>
8     * {
9       margin: 0;
10      padding: 0;
11      box-sizing: border-box;
12    }
13
14    body {
15      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
16      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
17      min-height: 100vh;
18      padding: 20px;
19    }
20
21    .container {
22      max-width: 1200px;
23      margin: 0 auto;
24    }
25
26    header {
27      background: white;
28      padding: 30px;
29      border-radius: 15px;
30      box-shadow: 0 10px 30px rgba(0,0,0,0.2);
31      margin-bottom: 30px;
32      display: flex;

```

OPEN EDITORS

HAMBURGUESERIA-PICCOLA A...

hamburgueseria-piccola

src

main

java\com\piccola

service

HamburgueseriaPiccolaApplication.java

resources

static

templates

clientes

formulario.html

lista.html

pedidos

formulario.html

lista.html

productos

formulario.html

lista.html

index.html

application.properties

test

target

.gitattributes

OUTLINE

TIMELINE

JAVA PROJECTS

MAVEN

hamburgueseria-piccola > src > main > resources > application.properties

```

1 # CONFIGURACION BASE DE DATOS
2 # =====
3 spring.datasource.url=jdbc:postgresql://localhost:5432/hamburgueseria_piccola
4 spring.datasource.username=piccola_user
5 spring.datasource.password=piccola123
6 spring.datasource.driver-class-name=org.postgresql.Driver
7
8 # =====
9 # CONFIGURACION JPA/HIBERNATE
10 # =====
11 spring.jpa.hibernate.ddl-auto=update
12 spring.jpa.show-sql=true
13 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
14 spring.jpa.properties.hibernate.format_sql=true
15
16 # =====
17 # CONFIGURACION SERVIDOR
18 # =====
19 server.port=8080
20
21 # =====
22 # CONFIGURACION THYMELEAF
23 # =====
24 spring.thymeleaf.cache=false
25 spring.thymeleaf.prefix=classpath:/templates/
26 spring.thymeleaf.suffix=.html
27
28 # =====
29 # CONFIGURACION LOGGING
30 # =====
31 logging.level.com.piccola=DEBUG
32 logging.level.org.springframework.web=INFO

```

27