Imagen que contiene botiquín de primeros auxilios, objeto, dibujo, señal

El contenido generado por IA puede ser incorrecto.

**“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**

**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 | * 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 | * 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 | * 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 | * Propietario de Hamburguesería Piccola |
| Autores involucrados | * **Jean Pier David Vega Choque** • 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**

|  |  |  |
| --- | --- | --- |
| * Nº | * 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**

|  |  |  |
| --- | --- | --- |
| * Nº | * 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)

└── develop (integración)

├── feature/autenticacion-usuarios

├── feature/gestion-pedidos

├── feature/control-inventario

└── feature/reportes-estadisticas

**ESTRUCTURA DEL REPOSITORIO**

hamburgueseria-piccola/

├── .github/

│ └── workflows/

│ └── ci.yml

├── src/

│ ├── main/java/com/piccola/

│ │ ├── controller/

│ │ ├── 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