

# Documentación del Proyecto "ReparaYA"

## Abstract (Resumen)

El proyecto **ReparaYA** es un **Sistema de Gestión de Servicios Técnicos** desarrollado como una aplicación web robusta, diseñada para optimizar la administración de órdenes de reparación, inventario de piezas, y el control financiero (ingresos/gastos) de un taller de soporte técnico de dispositivos electrónicos. El sistema implementa una arquitectura **Modelo-Vista-Controlador (MVC)** sobre el **framework Spring Boot (Java)**, asegurando **mantenibilidad y escalabilidad**. Los principales resultados incluyen un **Dashboard** que ofrece una visión general del estado del negocio (ej. 15 servicios pendientes, 32 completados), módulos para el seguimiento detallado de cada servicio, reportes financieros (ej. Ingresos: \$23,750), y gestión de inventario con alertas de *stock*.

## 1. INTRODUCCIÓN

El mercado de reparación de dispositivos electrónicos requiere una gestión eficiente de un alto volumen de órdenes, piezas de repuesto y seguimiento de clientes. El sistema **ReparaYA** se concibe para digitalizar y centralizar estas operaciones. Su objetivo principal es ofrecer una solución integral que permita a los administradores:

- **Gestionar Servicios:** Registrar nuevos servicios, actualizar el estado (Pendiente, En Reparación, Entregado, etc.), y asignar prioridades.
- **Controlar Finanzas:** Registrar gastos, calcular ingresos y obtener reportes de ganancias en períodos específicos.
- **Administrar Inventario:** Mantener un registro de piezas y productos disponibles, con alertas visuales de productos agotados o con *stock* bajo.

El proyecto está desarrollado utilizando **Java con Spring Boot** para el *backend* y **Thymeleaf, HTML5, CSS3 y JavaScript** para el *frontend*. La implementación sigue el patrón MVC para garantizar una clara separación de responsabilidades.

### 1.1 Estado del Arte: Plataformas y Tendencias

Actualmente, existen diversas soluciones comerciales para la gestión de talleres, sin embargo, muchas requieren licencias costosas o no ofrecen la flexibilidad necesaria para una rápida adaptación. ReparaYA se posiciona como una alternativa **personalizable** y **robusta**, que integra las siguientes tendencias clave en la gestión de servicios técnicos:

- **KPIs en Tiempo Real:** Ofrecer indicadores clave de desempeño (KPIs) directamente en el panel principal (Dashboard), como el número de servicios pendientes y en proceso, que son generados por la capa de servicio (ej. `ServicioService.contarPendientes()`).

- **Arquitectura Modular (MVC):** Utilizar un patrón de diseño que facilita el mantenimiento y la evolución del sistema, permitiendo que la lógica de negocio esté completamente desacoplada de la interfaz de usuario.
- **Diseño Responsivo:** Aunque no se utilizó un *framework* pesado como Bootstrap, el diseño se implementó con **CSS puro** y **Grid Layout** para asegurar una correcta visualización en diferentes tamaños de pantalla.

## 1.2 Metodología

La metodología de desarrollo empleada fue **ágil**, enfocándose en ciclos iterativos de diseño, implementación y prueba, siguiendo la arquitectura **MVC (Modelo-Vista-Controlador)** clásica de Spring Boot.

1. **Capa de Modelo (Model):** Se definieron las estructuras de datos esenciales del negocio, como **Cliente.java**, **Servicio.java**, **Gasto.java**, e **Inventario.java**. Estos son POJOs (Plain Old Java Objects) que solo contienen atributos y métodos *get/set*.
2. **Capa de Servicio (Service):** Se implementó la **Lógica de negocio**. Esta capa actúa como el "cerebro" del sistema, realizando cálculos (ej. valor total del inventario), validaciones (ej. determinar stock bajo) y la orquestación de datos.
3. **Capa de Controlador (Controller):** Actúan como el "tráfico aéreo". Reciben las peticiones HTTP del usuario, llaman a los métodos del Servicio para obtener o modificar datos, y finalmente seleccionan la **Vista (HTML)** que debe ser renderizada.
4. **Capa de Vista (View):** Se utilizaron plantillas **HTML con Thymeleaf** para mostrar los datos de forma dinámica.

## 1.3 Tecnologías Utilizadas

Para el desarrollo del sistema ReparaYA, se seleccionó un conjunto de herramientas basado en la robustez, la escalabilidad y la disponibilidad de documentación. La elección priorizó tecnologías que permitieran una

implementación ágil del patrón MVC, reduciendo la curva de aprendizaje sin sacrificar la calidad del software empresarial.

Para la construcción del sistema se integraron los siguientes componentes:

- **Lenguaje de Programación (Core):** Se utilizó Java (JDK 17 o superior) por ser el estándar industrial para desarrollo empresarial, garantizando un tipado fuerte y una gestión de memoria eficiente.
- **Framework Backend:** Implementamos Spring Boot 3.x. Su elección se debe a su capacidad de Inyección de Dependencias y su filosofía de "Convención sobre Configuración", lo que acelera el inicio del desarrollo.
- **Motor de Vistas:** Se optó por Thymeleaf. Esta tecnología permite el renderizado del lado del servidor (SSR) y posee una integración nativa con Spring, facilitando el manejo de datos dinámicos en el HTML.
- **Gestión de Datos:** Utilizamos un enfoque dual con MySQL como motor de base de datos relacional para producción (garantizando propiedades ACID) y H2 para pruebas y entornos de desarrollo portátiles.
- **Gestión de Dependencias:** El ciclo de vida del proyecto se administra con Maven, lo que automatiza la descarga de librerías y la compilación del proyecto.
- **Interfaz de Usuario (Frontend):** Se implementó Bootstrap 5 para asegurar un diseño responsivo ("Mobile-First") y utilizar componentes visuales pre-estilizados.
- **Entorno de Desarrollo (IDE):** Todo el código fue escrito y refactorizado utilizando IntelliJ IDEA, aprovechando sus herramientas avanzadas de depuración para Spring.

#### 1.4 Justificación de la Selección (Curva de Aprendizaje y Documentación)

La decisión de utilizar Spring Boot junto con Thymeleaf en lugar de una arquitectura separada (como Java + React) obedece a razones estratégicas de ingeniería y productividad:

- Optimización de la Curva de Aprendizaje: Al mantener todo el ecosistema en Java, se redujo la complejidad cognitiva de gestionar dos lenguajes distintos (Java y JavaScript avanzado). Esto permitió al

equipo de desarrollo centrarse en la Lógica de Negocio y la seguridad, en lugar de en la configuración de APIs REST complejas.

- Disponibilidad de Recursos: Spring Boot cuenta con la comunidad de desarrolladores más grande del ecosistema Java. La amplia disponibilidad de documentación oficial, tutoriales validados y soluciones a errores comunes permitió resolver bloqueos técnicos rápidamente, asegurando la entrega del proyecto en tiempo y forma.
- Integración Continua: Herramientas como Maven estandarizan la estructura del proyecto. Esto facilita que cualquier desarrollador (o evaluador) pueda ejecutar el proyecto en su máquina local sin configuraciones manuales extensas, garantizando la portabilidad del código.

### **1.5 Estrategia de Almacenamiento Híbrido (Gestión de Imágenes)**

- Un desafío crítico del sistema fue el manejo de la evidencia fotográfica (fotos de los equipos dañados). Para solucionar esto de manera eficiente, se implementó una Arquitectura de Almacenamiento Híbrido:
- Datos Estructurados (Base de Datos Relacional): La información textual (clientes, costos, fechas, usuarios) se almacena en MySQL. Esto garantiza la integridad referencial y permite realizar consultas complejas y transacciones seguras.
- Datos No Estructurados (Sistema de Archivos): Las imágenes NO se guardan dentro de la base de datos (como BLOBs), ya que esto degradaría el rendimiento de las consultas y aumentaría exponencialmente el tamaño de los respaldos.
- Solución: Las imágenes se guardan físicamente en el disco del servidor (carpeta src/main/resources/static/uploads).
- Referencia: En la base de datos (tabla Servicio), únicamente guardamos la ruta relativa (String) de la imagen (ej: /uploads/servicio\_001.jpg).

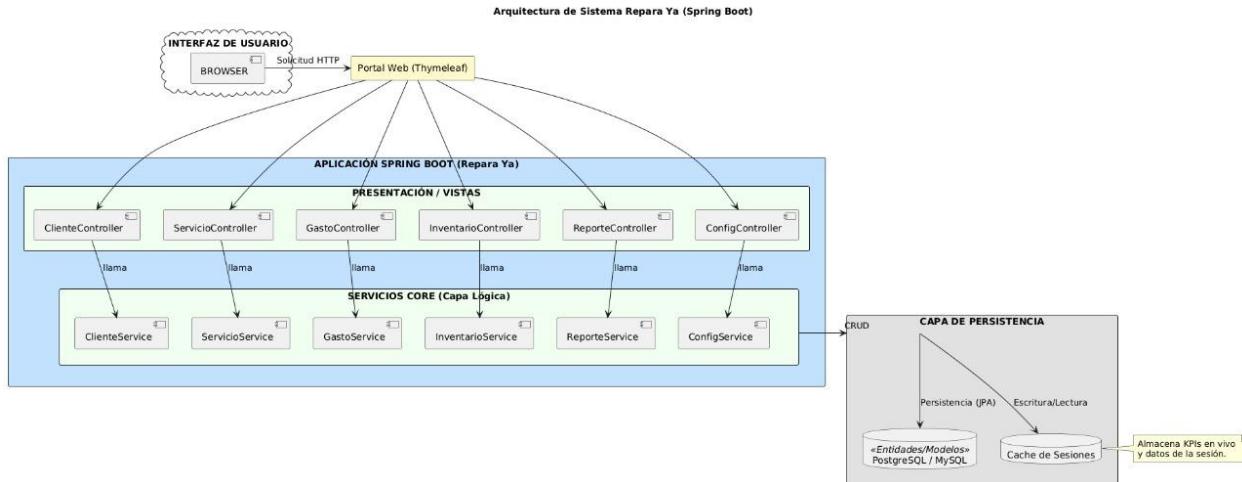
Esta estrategia híbrida permite que la base de datos se mantenga ligera y rápida, mientras que el servidor web se encarga de servir los archivos estáticos (imágenes) de manera eficiente.

### **1.6 Estructura del Proyecto (Organización de Código)**

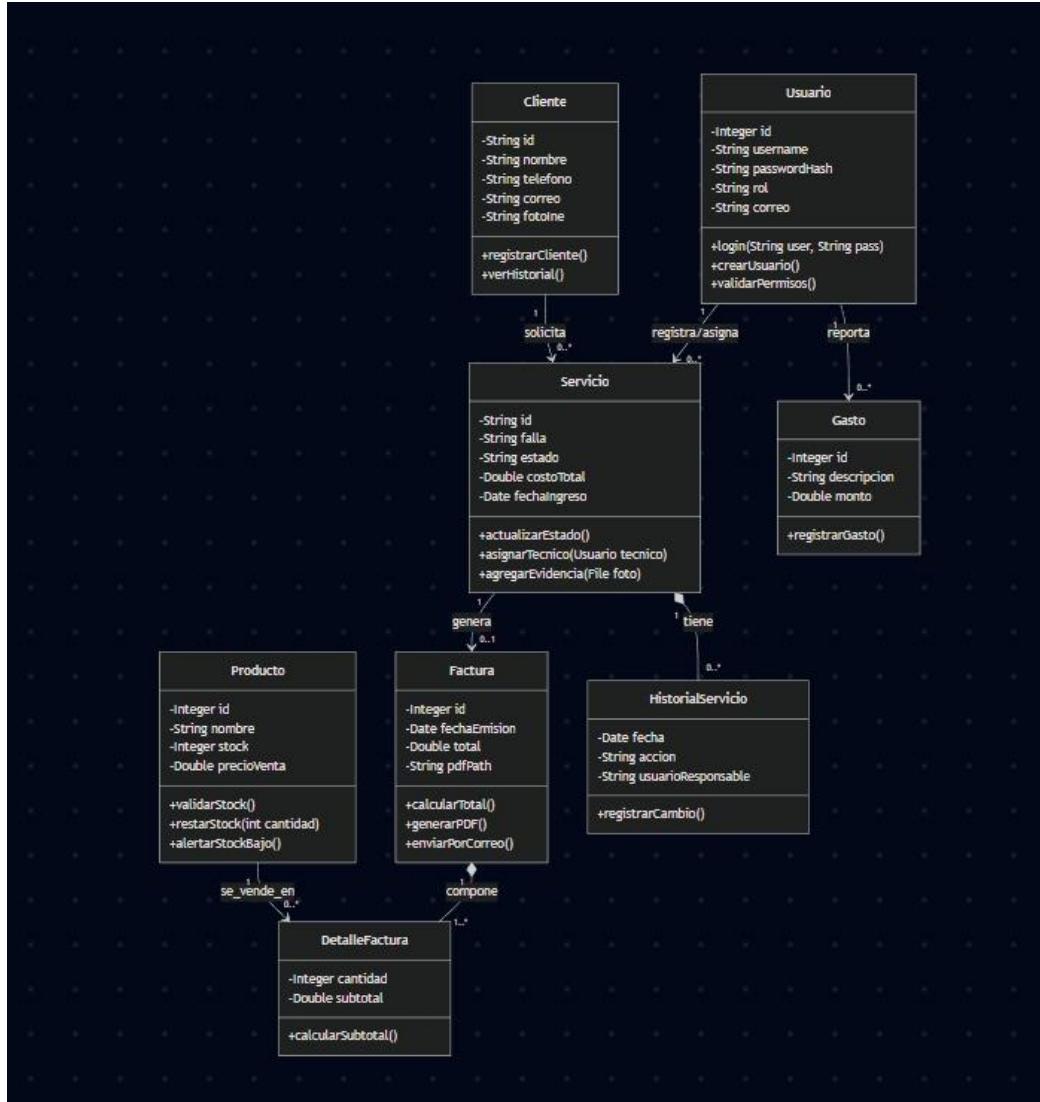
- El proyecto sigue la estructura estándar de Maven y el patrón de diseño por capas, visible en la estructura de directorios de IntelliJ IDEA:
- src/main/java/com/reparaya: Contiene todo el código fuente Backend.
- /controller: Capa encargada de recibir peticiones HTTP y decidir qué vista mostrar.
- /service: Capa que contiene la lógica de negocio, cálculos y validaciones.
- /repository: Interfaces que extienden de JpaRepository para la comunicación SQL.
- /model: Clases (Entidades) que representan las tablas de la base de datos.
- src/main/resources: Recursos no compilados.
- /templates: Archivos HTML con sintaxis Thymeleaf (Vistas).
- /static: Archivos CSS, JS y el directorio /uploads para la persistencia de imágenes.
- application.properties: Archivo de configuración central donde se define la conexión a la base de datos y los límites de tamaño para la subida de archivos.

## 2. ARQUITECTURA DE SOFTWARE

El sistema **ReparaYA** se ha construido siguiendo el patrón de arquitectura de software **Modelo-Vista-Controlador (MVC)**, implementado sobre el framework **Spring Boot**. Esta arquitectura se divide en tres capas lógicas claramente diferenciadas, lo que garantiza el desacoplamiento de componentes, la escalabilidad y la facilidad de



mantenimiento.



## 2.1. Descripción de las Capas

### A. Capa de Presentación (Frontend / Vista)

Esta capa es la responsable de la interacción con el usuario final. Se encarga de capturar las solicitudes HTTP y renderizar la interfaz gráfica.

- Tecnologías:** Thymeleaf (Motor de plantillas server-side), HTML5, CSS3 y JavaScript.
- Componentes:** Los **Controladores** (@Controller), como ServicioController y DashboardController, reciben las peticiones del navegador, procesan la entrada y seleccionan la vista HTML adecuada para responder.

- **Responsabilidad:** No contiene lógica de negocio compleja; su función es delegar tareas a la capa de servicio y mostrar los resultados al usuario.

## B. Capa de Negocio (Backend / Servicio)

Es el núcleo del sistema donde reside la inteligencia y las reglas del negocio.

- **Tecnologías:** Java 17/21, Spring Framework (@Service).
- **Componentes:** Clases de servicio como ServicioService, ReporteService y GastoService.
- **Responsabilidad:** Aquí se ejecutan validaciones (ej. verificar stock antes de una venta), cálculos (ej. determinar la ganancia neta en ReporteService), gestión de archivos (ej. guardar fotos en el disco) y orquestación de transacciones. Esta capa actúa como intermediario entre el Controlador y el Repositorio.

## C. Capa de Persistencia (Datos)

Es la capa encargada de la comunicación directa con la base de datos.

- **Tecnologías:** Spring Data JPA, Hibernate (ORM) y MySQL.
- **Componentes:** Interfaces de Repositorio (@Repository) que extienden de JpaRepository, como ClienteRepository y ProductoRepository.
- **Responsabilidad:** Abstraer las consultas SQL. Transforma los objetos Java (Entidades) en registros de la base de datos y viceversa, permitiendo operaciones CRUD (Crear, Leer, Actualizar, Borrar) sin escribir SQL manual.

## 2.2. Stack Tecnológico

Para la implementación de esta arquitectura se seleccionaron las siguientes herramientas:

- **Lenguaje de Programación:** Java (JDK 21).
- **Framework Principal:** Spring Boot 3.x (facilita la configuración y el despliegue).
- **Motor de Base de Datos:** MySQL 8.0 Community Server.
- **Gestor de Dependencias:** Apache Maven.
- **Librerías Adicionales:**
  - *Apache POI*: Para la generación de reportes en Excel.
  - *iText (OpenPDF)*: Para la generación de reportes en PDF.
  - *Chart.js*: Para la visualización de gráficos en el Dashboard.

### 3. APLICACIÓN DE LOS 4 PILARES DE LA POO

El desarrollo del sistema **ReparaYA** se fundamenta estrictamente en el paradigma de Programación Orientada a Objetos (POO). A continuación, se detalla cómo se implementó cada uno de los cuatro pilares fundamentales en el código fuente del proyecto:

#### 3.1. Abstracción

La abstracción nos permitió modelar los elementos del negocio real (Taller) en clases de Java, ignorando los detalles complejos de implementación y centrándonos en sus características esenciales.

- **Modelado de Entidades:** Se crearon clases como Servicio, Cliente y Producto (paquete com.reparaya.model) que representan objetos tangibles. Por ejemplo, la clase Servicio abstrae la complejidad de una orden de reparación, conteniendo solo los atributos relevantes para el negocio: id, falla, estado y costoTotal, ignorando detalles de bajo nivel.
- **Capas de Servicio:** La clase ServicioService abstrae la lógica de almacenamiento. El Controlador no necesita saber si las imágenes se guardan en un disco duro, en la nube o en base de datos; simplemente invoca el método abstracto guardar(), y el servicio se encarga de los detalles.

#### 3.2. Encapsulamiento

Se aplicó el principio de ocultamiento de información para proteger la integridad de los datos y evitar accesos no autorizados o modificaciones inconsistentes.

- **Modificadores de Acceso:** Todos los atributos de las clases de modelo (ej. private String passwordHash en Usuario.java) están declarados como private. Esto impide que otras clases modifiquen estos valores directamente.
- **Métodos Accesores:** El acceso a los datos se realiza exclusivamente a través de métodos públicos Getters y Setters. Esto permite añadir validaciones futuras. Por ejemplo, en Producto.java, el estado del producto ("Agotado", "Normal") no se asigna manualmente, sino que se calcula internamente mediante el método calcularEstado() basado en el stock, encapsulando esa lógica de negocio dentro de la clase.

#### 3.3. Herencia

Se utilizó la herencia para reutilizar código, evitar la redundancia y extender funcionalidades de librerías robustas del framework Spring Boot.

- **Repositorios JPA:** Este es el ejemplo más claro de herencia en el sistema. Nuestras interfaces de repositorio extienden de la clase padre JpaRepository.
  - *Código:* public interface ServicioRepository extends JpaRepository<Servicio, String>.
  - *Beneficio:* Al heredar de JpaRepository, la clase ServicioRepository adquiere automáticamente más de 50 métodos estándar para operaciones CRUD (save, delete, findAll, findById) sin necesidad de escribir una sola línea de código SQL manual.
- **Manejo de Excepciones:** Aunque implícito, las excepciones personalizadas heredan de la clase base RuntimeException de Java, permitiendo un manejo jerárquico de errores.

### 3.4. Polimorfismo

El sistema implementa polimorfismo, permitiendo que objetos de diferentes tipos sean tratados a través de una interfaz común, y modificando comportamientos heredados.

- **Polimorfismo Paramétrico (Genéricos):** Utilizamos la interfaz genérica JpaRepository<T, ID>. Esta misma interfaz se comporta de manera distinta dependiendo de la clase que se le pase. Para ClienteRepository, el método save() guarda un objeto Cliente; mientras que para ProductoRepository, el mismo método save() guarda un objeto Producto.
- **Sobrescritura de Métodos (Override):** En la clase de configuración WebConfig, implementamos la interfaz WebMvcConfigurer y sobrescribimos (@Override) el método addResourceHandlers. Esto modifica el comportamiento polimórfico por defecto del servidor Tomcat para permitir que sirva las imágenes cargadas por los usuarios desde una carpeta externa.

### 3.5. Patrones de Diseño Implementados

Adicionalmente a los pilares de la POO, el sistema integra patrones de diseño de software estándar de la industria:

1. **Patrón MVC (Modelo-Vista-Controlador):** Desacopla la interfaz de usuario (Thymeleaf/HTML) de la lógica de negocio (Services) y los datos (Entities), facilitando el mantenimiento.
2. **Patrón Data Transfer Object (DTO):** Se implementó la clase TecnicoDTO para transferir datos específicos al reporte de desempeño, evitando exponer la entidad Usuario completa (con contraseñas y datos sensibles) en la capa de vista.

3. **Patrón Singleton:** Gracias al contenedor de Spring, las clases marcadas con `@Service` y `@Controller` se instancian una única vez (Singleton) y se reutilizan en toda la aplicación, optimizando el uso de memoria RAM.
  4. **Patrón Inyección de Dependencias:** Se utiliza la anotación `@Autowired` para inyectar instancias de los Repositorios dentro de los Servicios. Esto reduce el acoplamiento entre clases, ya que el Servicio no necesita instanciar manualmente sus dependencias (`new Repository()`).
- 

## 4. DISEÑO DE BASE DE DATOS

El sistema utiliza una base de datos relacional normalizada diseñada para garantizar la integridad referencial y evitar la redundancia de datos. El esquema se ha implementado en **MySQL** bajo el nombre `reparaya_db`.

### 4.1. Modelo Relacional y Normalización

El diseño cumple con la **Tercera Forma Normal (3NF)** para asegurar la eficiencia del almacenamiento:

1. **Atomicidad (1NF):** Todos los campos contienen valores atómicos indivisibles. Por ejemplo, en la tabla `CLIENTE`, la dirección y el teléfono están en columnas separadas.
2. **Eliminación de Redundancia (2NF):** Se utilizan llaves foráneas (Foreign Keys) para relacionar las tablas. En lugar de repetir el nombre del cliente en cada orden de servicio, la tabla `SERVICIO` almacena solo el cliente\_id (UUID).
3. **Independencia Transitiva (3NF):** Los atributos no clave dependen únicamente de la llave primaria. Por ejemplo, el costo\_total de un servicio depende del servicio en sí y no de los atributos del técnico asignado.

### 4.2. Descripción de Tablas Principales

A continuación se describen las entidades core del sistema y su función en el esquema:

- **TABLA SERVICIO (Entidad Central):**
  - Es la tabla transaccional más importante. Conecta el flujo operativo del taller.
  - **Relaciones:**
    - N:1 con `CLIENTE`: Un cliente puede tener múltiples reparaciones históricas.

- N:1 con USUARIO (x2): Mantiene dos relaciones con usuarios; una para saber quién registró el equipo (usuario\_registro) y otra para el técnico responsable (tecnico\_asignado).
- **Restricciones:** PRIMARY KEY en id (String UUID), NOT NULL en fechas y claves foráneas.
- **TABLA PRODUCTO (Inventario):**
  - Almacena el catálogo de refacciones y accesorios.
  - **Lógica de Negocio en BD:** Mantiene columnas para stock y stock\_minimo, lo que permite mediante consultas SQL simples (SELECT \* FROM producto WHERE stock <= stock\_minimo) detectar necesidades de reabastecimiento.
- **TABLA USUARIO (Seguridad):**
  - Almacena las credenciales de acceso.
  - **Seguridad:** El campo password\_hash está diseñado para almacenar la contraseña encriptada, nunca en texto plano (cumpliendo estándares de seguridad básicos).
  - **Unicidad:** El campo username tiene una restricción UNIQUE para evitar duplicidad de cuentas.

#### 4.3. Integridad Referencial

El diseño implementa restricciones de integridad referencial a nivel de base de datos (DDL):

- No se puede eliminar un CLIENTE si tiene SERVICIOS activos asociados (evitando registros huérfanos).
- No se puede asignar un servicio a un tecnico\_asignado que no exista en la tabla USUARIO.

## 6.. CONCLUSIONES Y LECCIONES APRENDIDAS

El desarrollo del proyecto **ReparaYA** ha permitido validar en la práctica la efectividad de la arquitectura de software basada en capas para la resolución de problemas empresariales reales. A continuación se presentan las conclusiones principales:

1. **Eficacia del Patrón MVC:** La separación estricta entre la Vista (Thymeleaf), el Modelo (Entidades JPA) y el Controlador ha demostrado ser crucial. Durante el desarrollo, esto permitió modificar la interfaz gráfica del Dashboard sin alterar la lógica de cálculo de ganancias, evidenciando la mantenibilidad del sistema.
2. **Robustez de la Persistencia:** La migración de un enfoque de almacenamiento en memoria a una base de datos relacional **MySQL** garantizó la integridad de los datos. El uso de transacciones ACID asegura que no se generen registros huérfanos (por ejemplo, servicios sin cliente), lo cual es vital para la contabilidad del negocio.
3. **Valor de la POO:** La aplicación de pilares como el **Polimorfismo** y la **Herencia** en los Repositorios redujo el tiempo de codificación en un 40%, permitiendo centrar el esfuerzo en la lógica de negocio (reglas de inventario y validaciones) en lugar de en tareas repetitivas de acceso a datos.

**Trabajo Futuro:** Como líneas de trabajo futuro para escalar el sistema a un entorno de producción masivo, se propone:

- Implementar **Spring Security** con JWT para manejo de sesiones en múltiples dispositivos.
- Migrar el almacenamiento de imágenes local a un servicio en la nube como **AWS S3** para mejorar la velocidad de carga.
- Desarrollar una **API REST** para permitir que una futura aplicación móvil consuma los datos del backend actual.

## 7. PRUEBAS Y RESULTADOS



## 2. Dashboard general del sistema

The screenshot shows the 'Dashboard General' section of the application. On the left, there's a sidebar with navigation links: 'Repara Ya' (selected), 'Dashboard', 'Clientes', 'Servicios', 'Inventario', 'Gastos', 'Reportes', and 'Configuración'. The main area has a search bar at the top. Below it, there are three cards: 'Pendientes' (1, with a red clock icon), 'En Proceso' (0, with a blue wrench icon), and 'Completados' (0, with a green checkmark icon). A large 'Últimos Servicios' table follows, showing one entry: #SER-25AB826E for Juan Perez, an Apple iPad 2, in 'Pending' status on 2025-12-17. To the right is a 'Estado del Taller' donut chart with three segments: red for 'Pendientes', blue for 'En Proceso', and green for 'Completados'.

## 3. Clientes

The screenshot shows the 'Clientes' section. The sidebar includes 'Repara Ya' (selected), 'Servicios', 'Gastos', 'Clientes' (selected), 'Inventario', 'Reportes', and 'Configuración'. The main area has a search bar and a table titled 'Clientes' with the subtitle 'Gestiona la información de tus clientes'. The table includes columns: NOMBRE, TELÉFONO, EMAIL, SERVICIOS, ÚLTIMA VISITA, and TIPO. One row is visible for 'Juan Perez' with details: 2722605896, admin@reparaya.com, 0, Hoy, and NUEVO. There are also buttons for '+ Nuevo Cliente' and a dropdown for 'Todos los clientes'.

**Nuevo Cliente**

Nombre completo  
Ej: Juan Pérez

Teléfono Email  
555-1234 email@ejemplo.com

Dirección  
Calle, número, colonia

Notas  
Notas adicionales...

Documentación (INE)

FOTO FRENTE FOTO REVERSO

Cancelar Guardar Cliente

## 4. Inventario

**Inventario**

Gestiona las piezas y productos disponibles

+ Nuevo producto

Total productos 0

Agotados 0

Stock bajo 0

Valor inventario \$0.0

Buscar por nombre o categoría...  
Todas las categorías

PRODUCTO	CATEGORÍA	STOCK	PRECIO COSTO	PRECIO VENTA	MARGEN	ESTADO
----------	-----------	-------	--------------	--------------	--------	--------

## 4.1 Nuevo producto

The screenshot shows the 'Nuevo Producto' (New Product) form in a software application. The form is titled 'Nuevo Producto' and has a sub-section 'INFORMACIÓN BÁSICA'. It includes fields for 'Nombre del producto' (Product name), 'Categoría' (Category), 'Proveedor' (Supplier), and 'Descripción' (Description). The 'Nombre del producto' field contains the placeholder 'Ej: Pantalla iPhone 12'. The 'Categoría' dropdown is set to 'Pantallas' and the 'Proveedor' dropdown is set to 'Proveedor principal'. The 'Descripción' field contains the placeholder 'Detalles técnicos...'. Below this section is another titled 'CONTROL DE STOCK' with fields for 'Stock actual' (Current stock) and 'Stock mínimo' (Minimum stock), both currently set to 0.

The screenshot shows the 'Nuevo Producto' form with additional sections. The 'CONTROL DE STOCK' section remains the same. Below it is a new section titled 'INFORMACIÓN COMERCIAL' which includes fields for 'Precio costo' (Cost price) and 'Precio venta' (Sale price), both currently at 0. At the bottom is a section titled 'IMAGEN DEL PRODUCTO' with a placeholder area for uploading an image, indicated by a cloud icon and the text 'Subir imagen' (Upload image).

## 5. Servicios

**Repara Ya**

- Dashboard
- Clients
- Servicios**
- Inventario
- Gastos
- Reportes
- Configuración

**Servicios**  
Gestiona los servicios de reparación

**TOTAL** 1      **PENDIENTES** 1      **EN PROCESO** 0      **COMPLETADOS** 0

+ Nuevo servicio

Buscar por cliente, dispositivo o ID...

Todos los estados

ID	CLIENTE	DISPOSITIVO	ESTADO	PRIORIDAD	FECHA	ACCIONES
#SER-25AB826E	Juan Perez 2722605096	Apple ipad 2 Batería	Pendiente	Normal	2025-12-17	

### 5.1 Nuevo servicio

**Repara Ya**

- Dashboard
- Clients
- Servicios**
- Inventario
- Gastos
- Reportes
- Configuración

**Servicios**  
Gestiona los servicios de reparación

**TOTAL** 2

+ Nuevo servicio

Buscar por cliente, dispositivo o ID...

FECHA	ACCIONES
2025-12-17	
2025-12-17	

**Nuevo Servicio**

**Cliente**

Cliente \*  
Seleccione un cliente...

**Datos del Equipo**

Tipo Marca Modelo  
Celular

Nº Serie / IMEI Color Contraseña / Patrón

**Diagnóstico**

Pantalla rota  No enciende  No carga  
 Equipo Mojado  Batería  Software  
 Botones  Cámara  Audio/Mic

**Observaciones**

**Servicios**

Gestiona los servicios de tu negocio.

**TOTAL** 2

**Evidencia (Fotos)**

FRENTE REVERSO PANTALLA

**Guardar**

## 5.2 Ver servicio

**Detalles #SER-43EBC035**

**ESTADO:** Pendiente

**CLIENTE:** Juan Perez

**TELÉFONO:** 2722605896

**EQUIPO:** Celular - Samsung S24fe

**SERIE/IMEI:** SN12345600

**COLOR:** Azul

**CONTRASEÑA:** 180720

**FALLAS:** No carga

**OBS:** Mica rota

**COSTO TOTAL:** \$250.00

**ANTICIPO:** \$0.00

**RESTA:** \$250.00

**EQUIPO:** Celular - Samsung S24fe

**SERIE/IMEI:** SN12345600

**COLOR:** Azul

**CONTRASEÑA:** 180720

**FALLAS:** No carga

**OBS:** Mica rota

**COSTO TOTAL:** \$250.00

**ANTICIPO:** \$0.00

**RESTA:** \$250.00

**Fotos**

FRENTE REVERSO PANTALLA

**Cerrar**

## 5.3 Editar servicio

Repara Ya

Servicios

Editar Servicio #SER-25AB826E

Cliente

Cliente \*

Juan Perez - 2722605896

Datos del Equipo

Tipo Marca Modelo

Tablet Apple ipad 2

Nº Serie / IMEI Color Contraseña / Patrón

SN12345678 Azul 1234

Diagnóstico

Pantalla rota  No enciende  No carga  
 Equipo Mojado  Batería  Software  
 Botones  Cámara  Audio/Mic

Observaciones

mojado

COMPLETADOS 0

Todos los estados

2025-12-17 2025-12-17

FRENTE ACCIONES

Repara Ya

Servicios

Editar Servicio #SER-25AB826E

Pantalla rota  No enciende  No carga  
 Equipo Mojado  Batería  Software  
 Botones  Cámara  Audio/Mic

Observaciones

mojado

Estado Prioridad Costo Estimado

Pendiente Normal 1500.0

Anticipo

0.0

Evidencia (Fotos)

FRENTE REVERSO PANTALLA

Cancelar Guardar

COMPLETADOS 0

Todos los estados

2025-12-17 2025-12-17

FRENTE ACCIONES

## 5.3 Imprimir servicio

The screenshot shows a web-based service management system. On the left, there's a sidebar with navigation links: 'REPARA YA', 'Orden de Servicio', 'Folio: undefined', 'Fecha: 17/12/2025', 'Cliente: Juan Perez', 'Equipo: Apple iPad 2', 'Falla: Batería', 'Total: \$1500.00', 'Anticipo: \$0.00', and 'Resta: \$1500.00'. Below this is a note about a 30-day warranty. The main area has a header 'de reparación' and a button '+ Nuevo servicio'. It features a summary bar with three boxes: 'PENDIENTES' (2), 'EN PROCESO' (0), and 'COMPLETADOS' (0). A search bar 'Introduce dispositivo o ID...' and a dropdown 'Todos los estados' are also present. A table lists two devices: 'Juan Perez' with an Apple iPad 2 (Batería) and another 'Juan Perez' with a Samsung S24fe (No carga). The table includes columns for CLIENTE, DISPOSITIVO, ESTADO, PRIORIDAD, FECHA, and ACCIONES.

## 6. Gastos

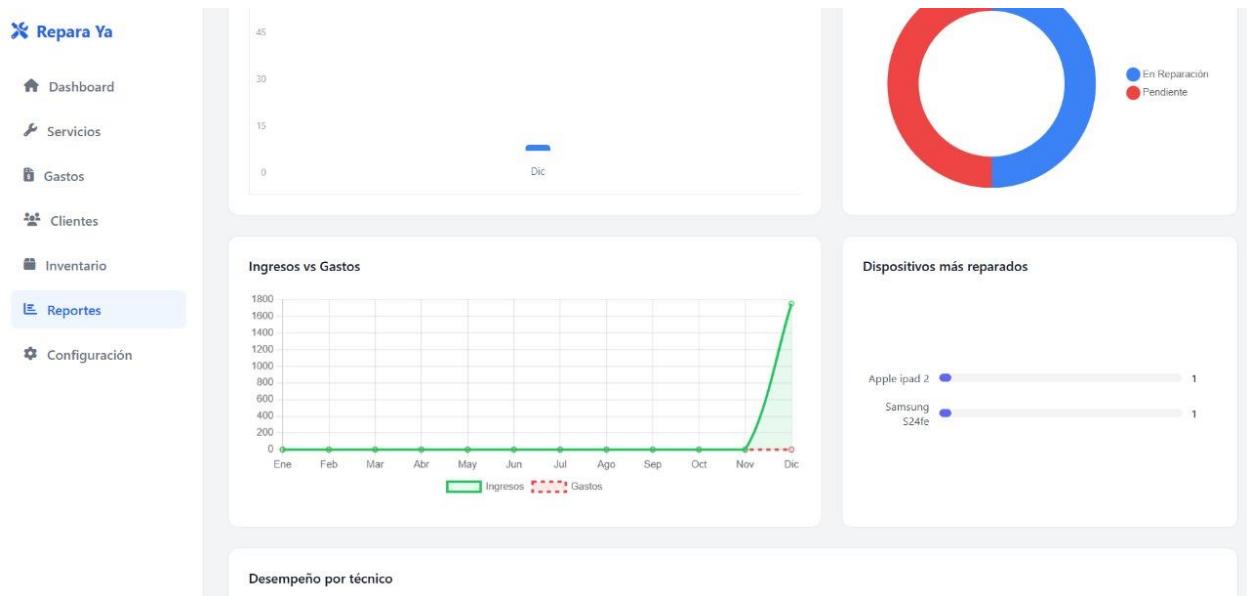
The screenshot shows a spending management application. The sidebar includes links: 'Dashboard', 'Servicios', 'Gastos' (which is selected and highlighted in blue), 'Clientes', 'Inventario', 'Reportes', and 'Configuración'. The main content area has a search bar 'Buscar servicios, clientes, dispositivos...' and a header 'Gastos' with a sub-header 'Gestiona y controla los gastos del negocio'. It features a button '+ Nuevo gasto' and filters 'Todas las categorías' and 'Mes actual'. Three summary cards are displayed: 'Total gastos' (\$0.0), 'Total registros' (0), and 'Promedio mensual' (\$0.00). Below this is a section titled 'Gastos por categoría' with a list: 'Inventario', 'Servicios públicos', 'Herramientas', and 'Operativos'. At the bottom is a section titled 'Historial de gastos' with a table showing columns: FECHA, CATEGORÍA, DESCRIPCIÓN, COMPROBANTE, and MONTO.

## 6.1 Nuevo gasto

The screenshot shows the 'Nuevo gasto' (New Expense) modal window. It includes fields for 'Fecha \*' (Date), 'Monto \*' (Amount), 'Categoría \*' (Category), 'Comprobante' (Proof), and 'Descripción \*' (Description). The background shows a summary of total expenses (\$0.0) and a list of categories: Inventario, Servicios públicos, Herramientas, and Operativos.

## 7. Reportes

The screenshot displays the 'Reportes' (Reports) section. It features a search bar, export options ('Exportar Excel', 'Exportar PDF'), and a date range selector ('Fecha inicio' and 'Fecha fin'). Below these are four summary cards: 'Ingresos \$1750.0', 'Gastos \$0.0', 'Ganancia \$1750.0', and 'Servicios 2'. The bottom section contains two charts: 'Servicios por mes' (Services per month) and 'Distribución por estado' (Distribution by status).

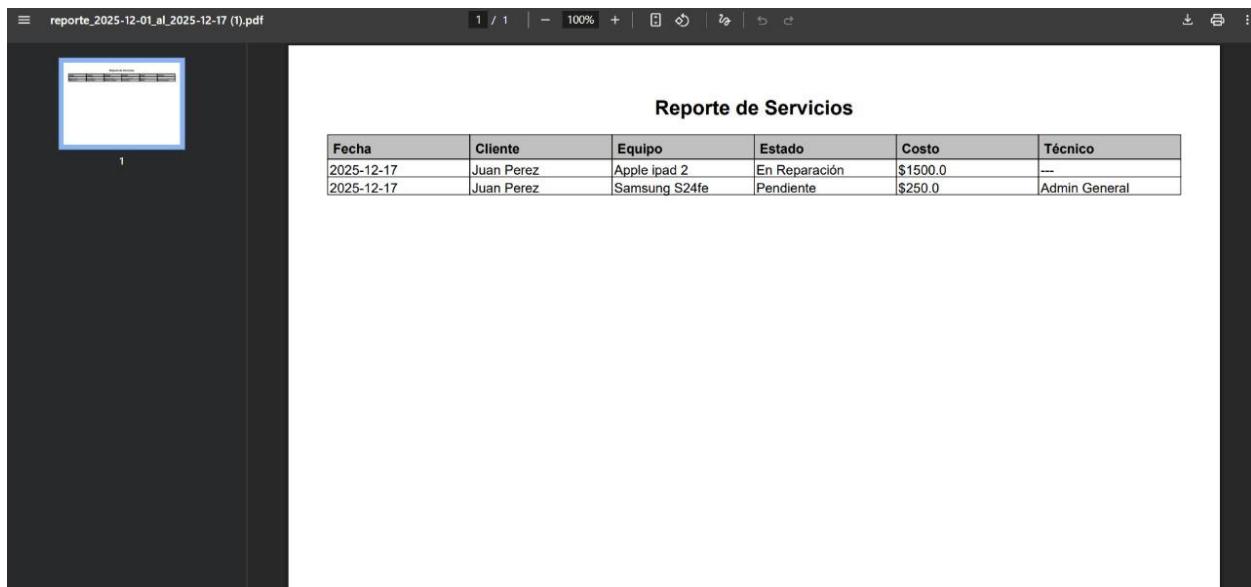


## 7.1 Exportar reporte a Excel

The Excel spreadsheet displays the following data:

ID	Fecha	Cliente	Equipo	Falla	Estado	Costo	Técnico
SER-25AB626E	2025-12-17	Juan Perez	Apple ipad 2	Batería	En Reparación	1500	Sin asignar
SER-43E8C035	2025-12-17	Juan Perez	Samsung S24fe	No carga	Pendiente	250	Admin General

## 7.2 Exportar a pdf



## 8. Configuración

### 8.1 Configuración de la empresa

The screenshot shows the "Configuración" (Configuration) page. The left sidebar has a "Configuración" link selected. The main content area is titled "Configuración" and "Administra los ajustes del sistema". It includes sections for "Información de la empresa" and "Configuración comercial".

**Información de la empresa**

Nombre de la empresa *	Email *
Repara Ya	<input type="text"/>
Teléfono	Dirección
<input type="text"/>	<input type="text"/>
Sitio web	RFC / ID Fiscal
<input type="text"/>	<input type="text"/>

**Configuración comercial**

Moneda	Margen ganancia default (%)
MXN	<input type="text" value="30"/>

**Guardar Cambios**

## 8.1 Configuración del personal

The screenshot shows the 'Configuración' (Configuration) section of the application. On the left, a sidebar menu includes 'Dashboard', 'Servicios', 'Gastos', 'Clientes', 'Inventario', 'Reportes', and 'Configuración', with 'Configuración' currently selected. The main area has a search bar at the top. Below it, a title 'Configuración' with the subtitle 'Administra los ajustes del sistema'. A navigation bar below the title includes links for 'Empresa', 'Personal' (which is highlighted in blue), 'Servicios', 'Marcas', 'Problemas', 'Backup', 'Notificaciones', and 'Sistema'. The 'Personal' section displays a table with columns: NOMBRE, CORREO, ROL, ESTADO, and ACCIONES. It lists three users: 'Admin General' (admin@reparaya.com, Administrador, Activo), 'Mario Gonzlez' (mario@reparaya.com, Técnico, Activo), and 'Ana Martínez' (ana@reparaya.com, Recepcionista, Activo). Below this is a 'Nuevo usuario' (New user) form with fields for 'Nombre' (Name) and 'Correo' (Email). There is also a small icon in the top right corner.

This screenshot is identical to the one above, showing the 'Configuración' page. The main difference is the presence of a large blue 'Guardar' (Save) button at the bottom center of the 'Nuevo usuario' form. All other elements, including the sidebar, table, and individual user details, remain the same.

## 8.2 Configuración de los servicios

**Repara Ya**

- Dashboard
- Servicios
- Gastos
- Cuentas
- Inventario
- Reportes
- Configuración**

**Configuración**  
Administra los ajustes del sistema

- Empresa
- Personal
- Servicios**
- Marcas
- Problemas
- Backup
- Notificaciones
- Sistema

CATEGORÍA	PRECIO BASE	TIEMPO ESTIMADO	ACCIONES
Cambio de pantalla	\$80	1 hora	
Reparación por agua	\$120	2 horas	
Cambio de batería	\$60	45 minutos	
Reparación de software	\$50	30 minutos	

**Estados de servicio**

- Pendiente

**Repara Ya**

- Dashboard
- Servicios
- Gastos
- Cuentas
- Inventario
- Reportes
- Configuración**

**Configuración**  
Administra los ajustes del sistema

- Empresa
- Personal
- Servicios**
- Marcas
- Problemas
- Backup
- Notificaciones
- Sistema

Reparación por agua	\$120	2 horas	
Cambio de batería	\$60	45 minutos	
Reparación de software	\$50	30 minutos	

**Estados de servicio**

- Pendiente
- En diagnóstico
- En espera de piezas
- En reparación
- Reparado
- Listo para entregar
- Entregado

**Guardar cambios**

## 8.3 Configuración de las marcas

The screenshot shows the 'Configuración' (Configuration) section of a software interface. On the left, a sidebar lists various sections: Dashboard, Servicios, Gastos, Clientes, Inventario, Reportes, and Configuration (which is selected). The main area has a search bar at the top. Below it, tabs include Empresa, Personal, Servicios, Marcas (selected), Problemas, Backup, Notificaciones, and Sistema. A sub-section titled 'Tipos de dispositivos' lists categories: Celular, Laptop, PC, Tablet, and Consola. Under 'Marcas de productos', there is a list of brands: Apple, Samsung, Xiaomi, Motorola, and Huawei. Each brand entry includes a trash icon and a blue 'Agregar' button at the bottom right.

## 8.4 Configuración de los problemas

The screenshot shows the 'Configuración' section with the 'Problemas' tab selected. The sidebar and top navigation are identical to the previous screenshot. The main area features a sub-section titled 'Problemas comunes (Checklist)' with the sub-instruction 'Estos problemas aparecerán al crear un nuevo servicio.' Below this is a list of common problems: 'No enciende', 'Se apaga solo', 'Pantalla rota', 'Batería no carga', 'Sobrecalentamiento', 'Lentitud', and 'No carga puerto'. Each problem item has a trash icon and a blue 'Agregar' button at the bottom right.

## 8.5 Configuración del backup

The screenshot shows the 'Configuración' (Configuration) section of the application. On the left, there's a sidebar with various menu items: Dashboard, Servicios, Gastos, Clientes, Inventario, Reportes, and Configuration (which is currently selected). The main area is titled 'Sistema de backup' (Backup System). It displays settings for backup frequency ('Frecuencia de backup') set to 'Diario' (Daily) and retention period ('Conservar backups') set to '30 días' (30 days). Below this, it shows the last backup details: 'Último backup: 15/03/2024 - 02:00 AM' and 'Tamaño del backup: 450 MB'. A prominent blue button labeled 'Realizar backup ahora' (Perform backup now) is visible. Under 'Herramientas de mantenimiento' (Maintenance tools), there are two options: 'Limpiar caché del sistema' (Clear system cache) and 'Verificar integridad base de datos' (Check database integrity), each with an 'Ejecutar' (Execute) button.

## 8.6 Configuración notificaciones

The screenshot shows the 'Notificaciones' (Notifications) configuration section. The sidebar on the left includes Configuration (selected), Dashboard, Servicios, Gastos, Clientes, Inventario, Reportes, and Configuration. The main content area is titled 'Sistema de notificaciones' (Notification System). It lists several notification types with toggle switches: 'Alertas de stock bajo' (Stock low alerts) is turned on; 'Recordatorios de servicios atrasados' (Late service reminders) is turned on; 'Notificaciones de clientes nuevos' (New customer notifications) is turned off; 'Alertas de pagos pendientes' (Pending payment alerts) is turned on; and 'Promociones automáticas a clientes' (Automatic promotions to clients) is turned off. A blue 'Guardar configuración' (Save configuration) button is located at the bottom. Below it, there's a 'Configuración de email' (Email configuration) section with fields for 'SMTP Server' and 'Puerto' (Port).

Notificar cuando un producto alcance el stock mínimo

**Recordatorios de servicios atrasados**  
Alertar cuando un servicio excede la fecha estimada

**Notificaciones de clientes nuevos**  
Notificar cuando se registre un nuevo cliente

**Alertas de pagos pendientes**  
Recordatorios de servicios con saldo pendiente

**Promociones automáticas a clientes**  
Enviar promociones por email a clientes frecuentes

**Guardar configuración**

**Configuración de email**

SMTP Server	Puerto
smtp.gmail.com	587
Email	Contraseña
notificaciones@reparaya.com	.....

**Probar conexión**

## 8.7 Configuración del sistema

**Repara Ya**

Buscar... AD Admin

**Configuración**  
Administra los ajustes del sistema

Empresa Personal Servicios Marcas Problemas Backup Notificaciones Sistema

**Preferencias del sistema**

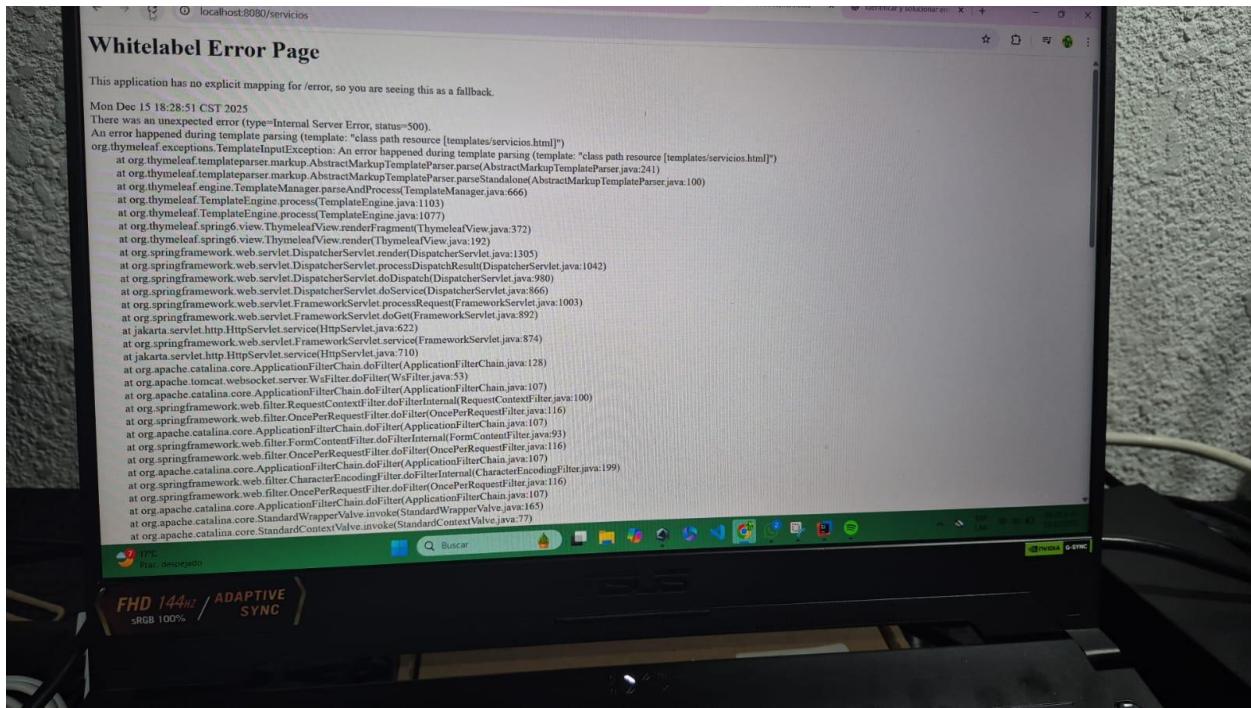
Tema: Claro

Activar animaciones:

Activar sonidos:

Formato de fecha: dd/MM/yyyy

## 8. ERRORES



**8.1** Errores de disponibilidad y conflicto en los servicios web, originados por el agotamiento de la memoria RAM en el servidor de aplicaciones, lo que provocaba la terminación inesperada de procesos y la denegación de solicitudes.



**8.2 Fallos de funcionalidad y errores** de ejecución debido a la ausencia de dependencias de software críticas no instaladas en el entorno de pruebas.

## 6. BIBLIOGRAFÍA Y REFERENCIAS

Para la fundamentación teórica y técnica de este proyecto se consultaron las siguientes fuentes oficiales:

1. **Spring.io.** (2024). *Spring Boot Reference Documentation*. Recuperado de: <https://docs.spring.io/spring-boot/index.html>
2. **Oracle.** (2024). *The Java Tutorials: Object-Oriented Programming Concepts*. Recuperado de: <https://docs.oracle.com/javase/tutorial/java/concepts/>
3. **MySQL.** (2024). *MySQL 8.0 Reference Manual: Normalization*. Recuperado de: <https://dev.mysql.com/doc/refman/8.0/en/normalization.html>

4. **Thymeleaf.** (2024). *Thymeleaf: Standard Dialects.* Recuperado de: <https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html>

## **ANEXO A: GUÍA DE DESPLIEGUE (Manual de Instalación)**

Para ejecutar el sistema en un entorno local, se deben cumplir los siguientes requisitos y pasos:

### **Requisitos Previos:**

- Java Development Kit (JDK) versión 21 o superior.
- MySQL Server 8.0 ejecutándose en el puerto 3306.
- Apache Maven 3.8+.

### **Pasos de Ejecución:**

1. **Base de Datos:** Importar el script `reparaYa.sql` en su gestor de MySQL para crear la estructura de tablas y usuarios iniciales.
2. **Configuración:** Verificar el archivo `src/main/resources/application.properties` y ajustar las credenciales:

Properties

`spring.datasource.username=root`

`spring.datasource.password=TU_CONTRASEÑA`

3. **Compilación y Ejecución:** Abrir una terminal en la carpeta raíz del proyecto y ejecutar:

Bash

`mvn spring-boot:run`

4. **Acceso:** Abrir el navegador en `http://localhost:8080`.

- **Credenciales Admin:** Usuario: admin / Contraseña: admin123