Help Desk Mini - Sistema de Gestión de Tickets

Sistema web para registro y seguimiento de tickets de soporte técnico, desarrollado con Java 17 + Spring Boot (Backend) y HTML + Bootstrap + JavaScript (Frontend).

Tabla de Contenido

Características

El sistema implementa los siguientes requisitos funcionales:

- **RF-01**: Crear tickets con título, descripción, solicitante, categoría y prioridad
- **RF-02**: Listar y filtrar tickets con paginación (10 por página)
- **RF-03**: Cambiar estado de tickets con validación de transiciones
- **RF-04**: Agregar comentarios a tickets con trazabilidad
- **RF-05**: Resumen diario con estadísticas y top 3 categorías

Validaciones Implementadas

- Título: 5-120 caracteres
- Descripción: 10-2000 caracteres
- Prioridad: Baja, Media, Alta
- Estados: Abierto, En Progreso, Resuelto, Cerrado
- Transiciones de estado controladas por lógica de negocio

Tecnologías Utilizadas

Backend

- Java 17
- Spring Boot 3.2.1

- Spring Data JPA
- Oracle Database (con procedimientos PL/SQL para bonificación)
- Maven
- Lombok

Frontend

- HTML5
- CSS3
- Bootstrap 5.3.2
- JavaScript (Vanilla)
- Bootstrap Icons

Requisitos Previos

Antes de comenzar, asegúrate de tener instalado:

1. **Java Development Kit (JDK) 17 o superior**

bash

```
java -version
```

2. **Maven 3.6 o superior**

bash

```
mvn -version
```

3. **Oracle Database 11g o superior** (o cualquier RDBMS compatible)
    - Nota: Oracle otorga 5 puntos de bonificación
    - Alternativa: PostgreSQL, MySQL, H2
4. **Git** (para clonar el repositorio)

bash

```
git --version
```

5. **Navegador Web Moderno** (Chrome, Firefox, Edge, Safari)

Instalación

1. Clonar o Descargar el Proyecto

bash

```
git clone <url-repositorio>
cd helpdesk-mini
```

O descomprimir el archivo ZIP en una carpeta de tu elección.

2. Estructura de Carpetas

```
helpdesk-mini/
├── backend/
│   ├── src/
│   │   ├── main/
│   │   │   ├── java/com/soporteagil/helpdesk/
│   │   │   │   ├── config/
│   │   │   │   ├── controller/
│   │   │   │   ├── converter/
│   │   │   │   ├── dto/
│   │   │   │   ├── entity/
│   │   │   │   ├── exception/
│   │   │   │   ├── repository/
│   │   │   │   ├── service/
│   │   │   │   └── HelpDeskMiniApplication.java
│   │   │   └── resources/
│   │   │       ├── application.properties
│   │   │       └── (archivos estáticos)
│   │   │       └── db/
│   │   │           ├── 00_CrearUsuario.sql
│   │   │           ├── 01_crear_objetos_db.sql
│   │   │           └── 02_datos_de_prueba.sql
│   │   └── test/
│   │   │   ├── java/com/
│   │   │       ├── helpdesk/tickets/
│   │   │       └── soporteagil/helpdesk/
│   │   │   └── resources/
│   ├── .env
│   ├── pom.xml
├── frontend/
│   ├── index.html
│   ├── app.js
│   └── README.p
└── README.pdf (este archivo)
```

Configuración de Base de Datos

Paso 1: Conectar a Oracle Database

bash

```
sqlplus usuario/contraseña@localhost:1521/XE
```

Paso 2: Crear Usuario (opcional ver siguiente punto)

sql

```
CREATE USER helpdesk IDENTIFIED BY H3lpD3sk;
GRANT CONNECT, RESOURCE, CREATE SESSION, CREATE TABLE, CREATE SEQUENCE, CREATE
PROCEDURE TO helpdesk;
ALTER USER helpdesk QUOTA UNLIMITED ON USERS;
```

Paso 3: Ejecutar Scripts SQL en Orden

IMPORTANTE: Ejecutar los scripts en el siguiente orden:

bash

```
# 1. Crear usuario
sqlplus helpdesk_user/helpdesk_pass@localhost:1521/XE 00_CrearUsuario.sql

# 2. Crear secuencias, tablas y procedimientos
sqlplus helpdesk_user/helpdesk_pass@localhost:1521/XE 01_crear_objetos_db.sql

# 3. Insertar datos de prueba
sqlplus helpdesk_user/helpdesk_pass@localhost:1521/XE 02_datos_de_prueba.sql
```

Configuración del Backend

1. Editar `application.properties`

Ubicación: `backend/src/main/resources/application.properties`

properties

```
# Configuración del servidor
server.port=8080
server.servlet.context-path=/api
```

```
# Configuración de Oracle Database
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:XE
spring.datasource.username=helpdesk_user ←-Tomado del achivo .env
spring.datasource.password=helpdesk_pass ←-Tomado del achivo .env
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

# Configuración de JPA/Hibernate
spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.OracleDialect
```

IMPORTANTE: Ajusta los valores según tu configuración:

- `localhost:1521` → tu host y puerto de Oracle
- `XE` → tu SID o Service Name
- `helpdesk_user/helpdesk_pass` → tus credenciales

2. Compilar el Proyecto

bash

```
cd backend
mvn clean install
```

Si todo está correcto, verás:

```
[INFO] BUILD SUCCESS
```

Ejecutar el Frontend

Opción 1: Usar Extensión Live Server (VS Code)

1. Abre la carpeta `frontend/` en VS Code
2. Click derecho en `index.html`
3. Selecciona "Open with Live Server"
4. Se abrirá en `http://localhost:5500` o similar

Opción 2: Servidor HTTP Simple con Python

bash

```
cd frontend
python -m http.server 8000
```

Abre: `http://localhost:8000`

Opción 3: Abrir Directamente

Simplemente abre `frontend/index.html` en tu navegador.

IMPORTANTE: Asegúrate de que `app.js` tenga la URL correcta del backend:

javascript

```
const API_URL = 'http://localhost:8080/api';
```

Uso de la Aplicación

1. Dashboard (Pantalla Principal)

- **Tickets Creados Hoy**: Contador en tiempo real
- **Tickets Resueltos Hoy**: Tickets que cambiaron a estado "Resuelto" hoy
- **Top 3 Categorías**: Categorías con más tickets abiertos o en progreso

2. Listado de Tickets

- **Filtros**: Estado y Categoría (opcionales)
- **Paginación**: 10 tickets por página
- **Orden**: Por fecha de creación descendente (más recientes primero)
- **Click en un ticket**: Ver detalle completo

3. Crear Nuevo Ticket

Formulario con validaciones:

- **Título** (obligatorio): 5-120 caracteres
- **Descripción** (obligatoria): 10-2000 caracteres
- **Solicitante** (obligatorio): Nombre del usuario
- **Categoría** (obligatoria): Hardware, Software, Red, Acceso
- **Prioridad** (obligatoria): Baja, Media, Alta

4. Detalle de Ticket

- Ver información completa
- **Cambiar Estado**: Con validación de transiciones
    - Abierto → En Progreso o Resuelto
    - En Progreso → Resuelto
    - Resuelto → Cerrado
- **Agregar Comentarios**: Con autor y texto
- **Ver Historial**: Todos los comentarios ordenados por fecha

Estructura del Proyecto

Backend (Spring Boot)

src/main/java/com/soporteagil/helpdesk/

```
├── controller/
│   ├── TicketController.java        # REST API para tickets
│   ├── ComentarioController.java    # REST API para comentarios
│   └── ResumenController.java       # REST API para resumen
├── converter/
│   ├── EstadoConverter.java         # Conversor de valores de Estado
│   └── PrioridadConverter.java      # Conversor de valores de Prioridad
├── service/
│   ├── TicketService.java           # Lógica de negocio - tickets
│   ├── ComentarioService.java       # Lógica de negocio - comentarios
│   └── ResumenService.java          # Lógica de negocio - resumen
├── repository/
│   ├── TicketRepository.java        # Acceso a datos - tickets
│   ├── ComentarioRepository.java    # Acceso a datos - comentarios
│   └── CambioEstadoRepository.java  # Acceso a datos - cambios
├── entity/
│   ├── Ticket.java                  # Entidad JPA - tickets
│   ├── Comentario.java              # Entidad JPA - comentarios
│   └── CambioEstado.java            # Entidad JPA - cambios de estado
├── dto/
│   ├── TicketDTO.java               # Data Transfer Object
│   ├── ComentarioDTO.java
│   ├── ResumenDiarioDTO.java
│   ├── ApiResponse.java             # Respuesta estándar API
│   ├── CambioEstadoRequest.java     # Respuesta estándar API
│   └── PageResponse.java            # Respuesta paginada
├── exception/
│   ├── BusinessException.java       # Excepción personalizada
│   └── GlobalExceptionHandler.java  # Manejo global de errores
└── config/
│   ├──CorsConfig.java               # Configuración CORS
    └── DotenvInitializer.java       # Logica para cargar archivos de configuración de Entorno
```

Frontend

```
frontend/
├── index.html      # UI principal con Bootstrap
└── app.js          # Lógica JavaScript (API calls, eventos)
```

API REST Endpoints

Tickets

Método	Endpoint	Descripción
POST	`/api/tickets`	Crear nuevo ticket
GET	`/api/tickets`	Listar tickets (con filtros y paginación)
GET	`/api/tickets/{id}`	Obtener ticket por ID
PUT	`/api/tickets/{id}/estado`	Cambiar estado de ticket

Comentarios

Método	Endpoint	Descripción
POST	`/api/tickets/{ticketId}/comentarios`	Agregar comentario
GET	`/api/tickets/{ticketId}/comentarios`	Listar comentarios

Resumen

Método	Endpoint	Descripción
GET	`/api/resumen/diario`	Obtener resumen diario

Ejemplos de Uso con cURL

Crear Ticket:

bash

```
curl -X POST http://localhost:8080/api/tickets \
  -H "Content-Type: application/json" \
  -d '{
    "titulo": "Problema con impresora",
    "descripcion": "La impresora no responde a los comandos de impresión",
    "solicitante": "Juan Pérez",
    "categoria": "Hardware",
    "prioridad": "Media"
  }'
```

Listar Tickets:

bash

```
curl "http://localhost:8080/api/tickets?estado=Abierto&page=0&size=10"
```

Cambiar Estado:

bash

```
curl -X PUT http://localhost:8080/api/tickets/1/estado \
  -H "Content-Type: application/json" \
  -d '{"nuevoEstado": "En Progreso"}'
```

Plan de Pruebas

Pruebas Funcionales Manuales

RF-01: Crear Ticket

Caso 1: Creación exitosa

1. Navegar a "Nuevo Ticket"
2. Llenar formulario con datos válidos
3. Click en "Crear Ticket"
4. **Resultado esperado**: Mensaje de éxito, ticket creado con ID

Caso 2: Validación de título corto

1. Ingresar título con menos de 5 caracteres
2. **Resultado esperado**: Error "El título debe tener entre 5 y 120 caracteres"

Caso 3: Validación de descripción corta

1. Ingresar descripción con menos de 10 caracteres
2. **Resultado esperado**: Error de validación

RF-02: Listar y Filtrar

Caso 1: Listar todos

1. Navegar a "Tickets"
2. **Resultado esperado**: Lista paginada de 10 tickets, ordenados por fecha desc

Caso 2: Filtrar por estado

1. Seleccionar estado "Abierto"
2. Click en "Filtrar"
3. **Resultado esperado**: Solo tickets con estado "Abierto"

Caso 3: Paginación

1. Si hay más de 10 tickets, click en "Página 2"
2. **Resultado esperado**: Siguientes 10 tickets

RF-03: Cambiar Estado

Caso 1: Transición válida (Abierto → En Progreso)

1. Abrir ticket en estado "Abierto"
2. Cambiar a "En Progreso"
3. **Resultado esperado**: Estado actualizado exitosamente

Caso 2: Transición inválida (Abierto → Cerrado)

1. Intentar cambiar de "Abierto" a "Cerrado"
2. **Resultado esperado**: Error "No se puede cerrar un ticket que no ha sido resuelto"

RF-04: Comentarios

Caso 1: Agregar comentario

1. Abrir un ticket
2. Escribir autor y texto
3. Click en "Enviar Comentario"
4. **Resultado esperado**: Comentario agregado y visible

RF-05: Resumen Diario

Caso 1: Ver dashboard

1. Navegar a Dashboard
2. **Resultado esperado**:
    - Contador de tickets creados hoy
    - Contador de tickets resueltos hoy
    - Top 3 categorías con tickets abiertos