Documentación del Proyecto Backend: `prueba-tickets`

Felipe Carvajal Parra

13 de enero de 2026

Índice

1. Introducción

El proyecto `prueba-tickets` es un backend desarrollado en Java con Spring Boot que permite la gestión de tickets y asigna prioridades automáticamente según una serie de reglas predefinidas. El sistema está diseñado para funcionar en memoria, lo que permite pruebas rápidas y un despliegue sencillo sin necesidad de base de datos relacional.

2. Estructura del Proyecto

La estructura principal del proyecto es la siguiente:

- **controllers:** Contiene los controladores REST que exponen los endpoints de la API y el manejador global de excepciones.
- **models:** Define los modelos de datos, incluyendo `Ticket` y `TicketType`.
- **repositories:** Almacena los tickets en memoria y gestiona operaciones básicas de CRUD.
- **services:** Contiene la lógica de negocio, incluyendo el servicio de tickets y el motor de priorización.
- **rules:** Implementa reglas independientes para calcular la prioridad de cada ticket.

3. Modelo de Datos

3.1. Ticket

El modelo `Ticket` representa cada solicitud en el sistema y contiene los siguientes atributos:

- **id:** Identificador único generado automáticamente.
- **type:** Tipo de ticket, definido por el enum `TicketType`, que puede ser `INCIDENTE`, `REQUERIMIENTO` o `CONSULTA`.
- **manualPriority:** Prioridad ingresada manualmente por el usuario (1 a 5).
- **creationDate:** Fecha de creación del ticket, asignada automáticamente por el servicio.
- **user:** Nombre del usuario que crea el ticket.

3.2. TicketType

`TicketType` es un enum que representa los tipos de tickets y permite que los valores se muestren en español al consumir la API. Esto asegura consistencia en la presentación y en la validación de los datos.

4. Repositorios

`TicketRepository` es responsable del almacenamiento en memoria de los tickets. Permite operaciones básicas como crear, listar y limpiar tickets, sin necesidad de conectarse a una base de datos externa.

5. Servicios

`TicketService` contiene la lógica principal del sistema:

- Creación de tickets, asignando automáticamente la fecha de creación.
- Listado de todos los tickets.
- Listado de tickets ordenados por la prioridad calculada mediante el motor de priorización.

6. Motor de Priorización y Reglas de Filtrado

Para gestionar la prioridad de los tickets, se implementó un **motor de priorización desacoplado** (`TicketPriorityEngine`). Este motor permite calcular la prioridad final de un ticket aplicando un conjunto de reglas independientes, de manera modular y flexible.

6.1. Arquitectura del Motor

El motor de priorización está diseñado para:

- Recibir un ticket y calcular su prioridad final.
- Aplicar múltiples reglas de forma secuencial o acumulativa.
- Permitir agregar, modificar o quitar reglas sin afectar el resto del sistema.

Cada regla implementa un contrato común, asegurando consistencia y facilidad de integración en el motor. Esto permite que cada regla se mantenga independiente y reutilizable.

6.2. Reglas de Prioridad Implementadas

Actualmente, el motor utiliza las siguientes reglas independientes:

1. **ManualPriorityRule:** Considera la prioridad ingresada por el usuario al crear el ticket.
2. **TypePriorityRule:** Asigna un peso adicional según el tipo de ticket (`INCIDENTE`, `REQUERIMIENTO`, `CONSULTA`), permitiendo que ciertos tipos sean tratados con mayor urgencia.
3. **AgePriorityRule:** Ajusta la prioridad del ticket según su antigüedad, incrementando la prioridad de tickets más antiguos para asegurar que no se queden sin atención.

El motor combina estas reglas para calcular un valor de prioridad final que determina el orden en que los tickets deben ser atendidos.

7. Controladores y Endpoints

`TicketController` expone la API REST con los siguientes endpoints:

- **POST /tickets:** Crear un ticket. Recibe los datos del ticket y devuelve el objeto creado con la fecha de creación automática.
- **GET /tickets:** Listar todos los tickets.
- **GET /tickets/prioridad:** Listar todos los tickets ordenados por la prioridad calculada por el motor.

`ValidationExceptionHandler` maneja los errores de validación de forma centralizada, devolviendo mensajes claros en español cuando los campos no cumplen las reglas definidas.

8. Flujo de Creación y Prioridad

1. El usuario envía un ticket con tipo, prioridad y usuario.
2. El servicio asigna automáticamente la fecha de creación.
3. El motor de priorización aplica todas las reglas configuradas para calcular la prioridad final.
4. Los tickets se almacenan en memoria y pueden listarse en cualquier momento, ya sea en orden de creación o por prioridad calculada.

9. Mensajes de Validación

El sistema devuelve mensajes claros y amigables en español en caso de errores de validación, por ejemplo:

- Prioridad fuera de rango.
- Usuario no especificado.
- Tipo de ticket inválido.

10. Conclusión

El proyecto `prueba-tickets` proporciona un backend completo para la gestión de tickets con priorización automática. El uso de un motor desacoplado con reglas independientes garantiza modularidad y flexibilidad, permitiendo extender o modificar la lógica de priorización sin afectar el resto del sistema. Gracias al almacenamiento en memoria y a la API documentada con mensajes claros, es posible desplegar y probar rápidamente el sistema.