Configuración e implementación

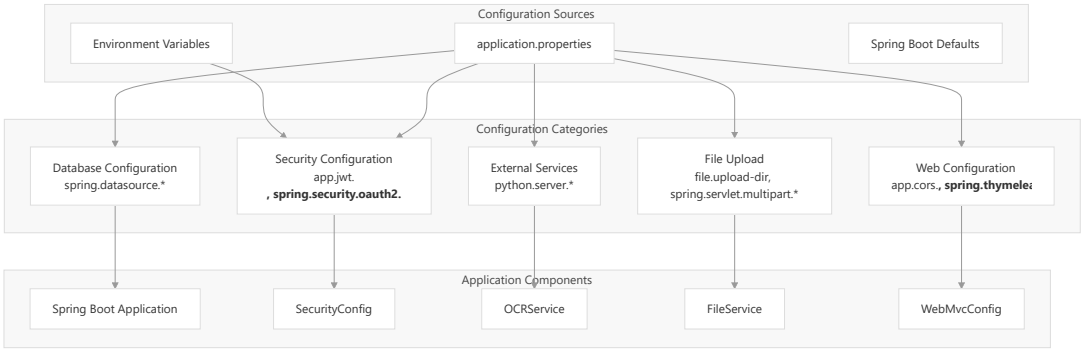
Este documento abarca la configuración de la aplicación, la configuración del entorno y las consideraciones de implementación para el sistema de gestión de gastos. Detalla las propiedades de configuración, la integración de servicios externos y los requisitos de infraestructura necesarios para implementar y ejecutar el conjunto completo del sistema.

Para configurar el entorno de desarrollo y las configuraciones de prueba, consulte Entorno de desarrollo . Para obtener información sobre la configuración de seguridad, consulte Seguridad y autenticación .

Descripción general de la configuración

El sistema utiliza la configuración basada en propiedades de Spring Boot `application.properties` para administrar todas las configuraciones de la aplicación, las conexiones de servicios externos y los parámetros de implementación.

Arquitectura de configuración



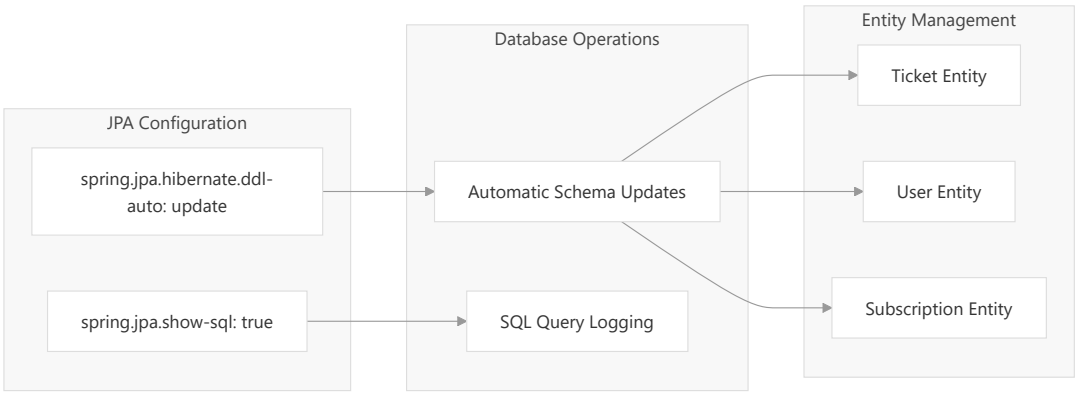
Configuración de la base de datos

La aplicación se conecta a una base de datos MySQL utilizando Spring Data JPA con Hibernate como proveedor ORM.

Propiedades de conexión de la base de datos

Propiedad	Valor	Descripción
`spring.datasource.url`	`jdbc:mysql://localhost:3306/gestor_bd`	URL de conexión MySQL
`spring.datasource.username`	`root`	Nombre de usuario de la base de datos
`spring.datasource.password`	`abc123.`	Contraseña de la base de datos
`spring.datasource.driver-class-name`	`com.mysql.cj.jdbc.Driver`	Controlador JDBC de MySQL

Configuración de JPA/Hibernate



La `ddl-auto=update` configuración crea o actualiza automáticamente el esquema de la base de datos según las definiciones de entidades JPA, lo que permite una evolución perfecta de la base de datos durante el desarrollo.

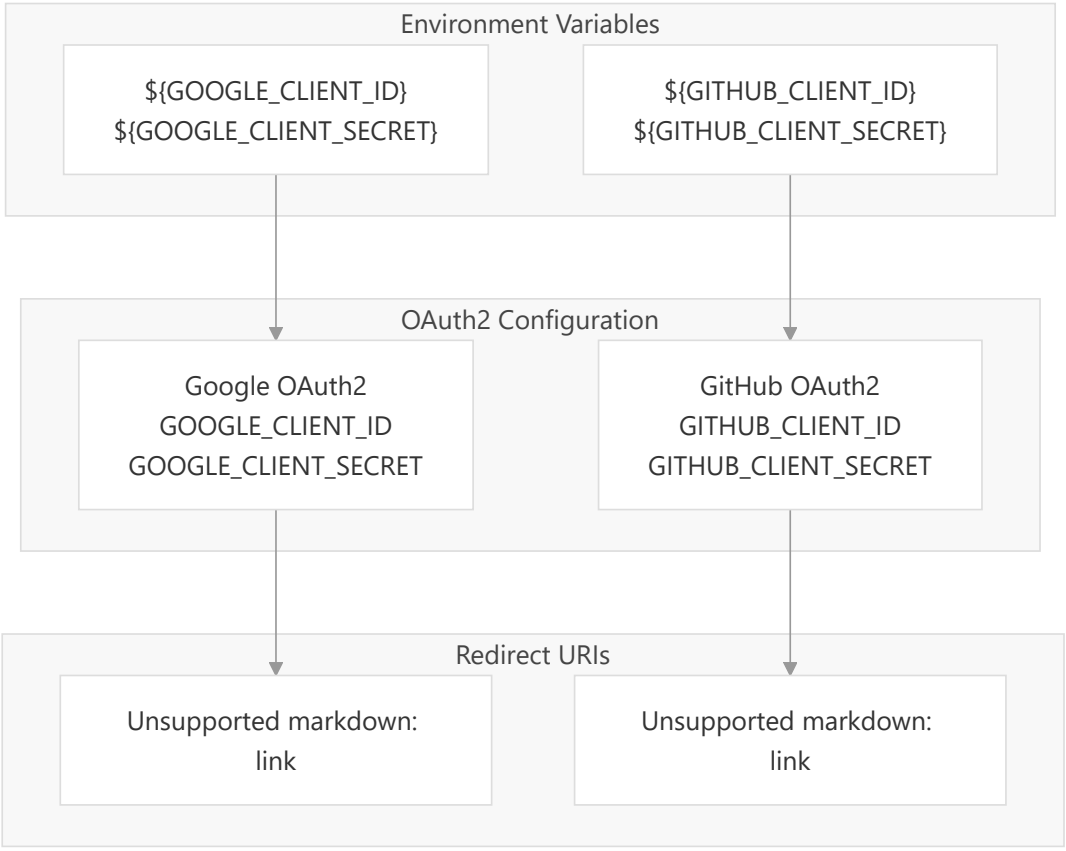
Configuración de seguridad y autenticación

El sistema implementa seguridad multicapa con tokens JWT e integración de inicio de sesión social OAuth2.

Configuración de JWT

Propiedad	Objetivo
`app.jwt.secret`	Clave secreta codificada en Base64 para la firma JWT
`app.jwt.expiration.minutes`	Tiempo de expiración del token (1440 minutos = 24 horas)

Configuración de proveedores de OAuth2



Configuración de seguridad de inicio de sesión

El sistema incluye protección contra fuerza bruta a través de límites de intentos de inicio de sesión configurables:

Propiedad	Por defecto	Descripción
`attemp.login.max.failed`	`5`	Máximo de intentos fallidos antes del bloqueo
`attemp.login.block.duration`	`30`	Duración del bloque en minutos
`attemp.login.delete.log`	`false`	Si se deben eliminar los registros de intentos de inicio de sesión

Integración de servicios externos

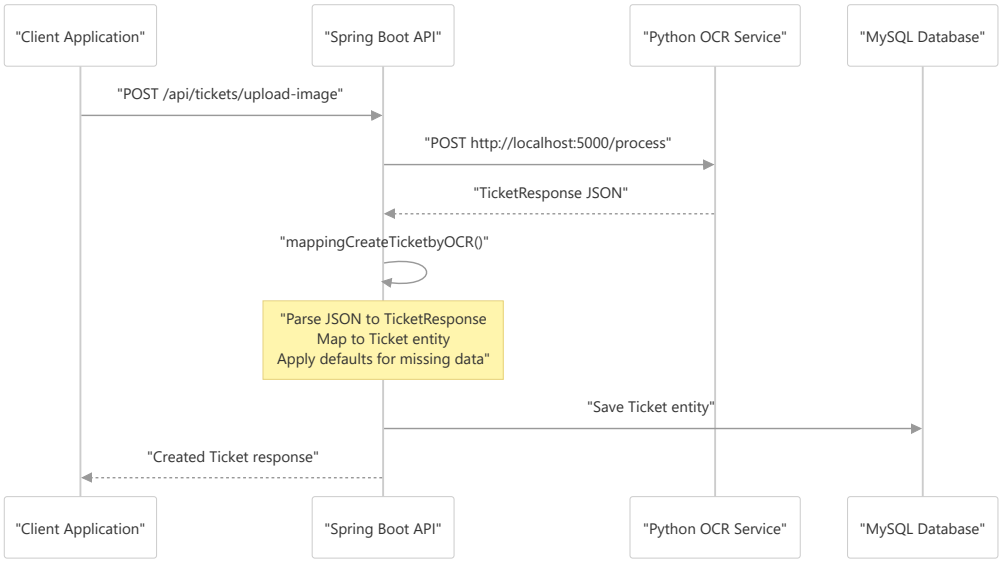
Configuración del servicio OCR de Python

La aplicación Spring Boot se integra con un servidor Python Flask externo para capacidades de procesamiento de OCR.

La `OCRService` interfaz define el contrato para el procesamiento de OCR externo, con implementaciones que manejan la carga de archivos al servicio Python y el análisis de la respuesta en `Ticket` entidades.

Procesamiento de respuestas de OCR

El `TicketServiceImpl.mappingCreateTicketbyOCR()` método procesa respuestas JSON del servicio OCR de Python:



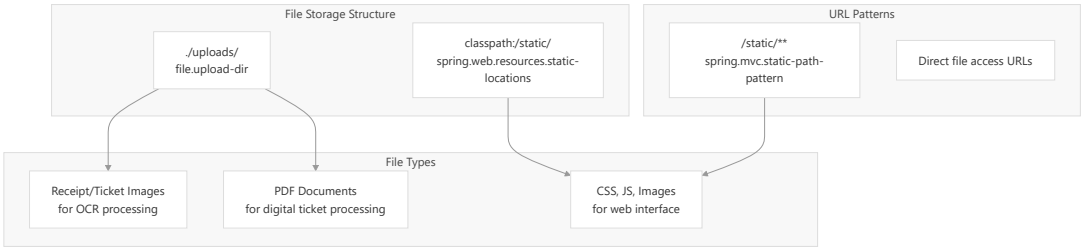
El proceso de mapeo maneja con elegancia los posibles errores de análisis, aplicando valores predeterminados cuando los datos de OCR están incompletos o mal formados.

Configuración de almacenamiento y carga de archivos

Configuración de carga de archivos

Propiedad	Valor	Descripción
`file.upload-dir`	`./uploads/`	Directorio para archivos cargados
`spring.servlet.multipart.enabled`	`true`	Habilitar la carga de archivos multiparte
`spring.servlet.multipart.max-file-size`	`5MB`	Tamaño máximo de archivo individual
`spring.servlet.multipart.max-request-size`	`5MB`	Tamaño máximo total de la solicitud

Configuración de recursos estáticos



El directorio de carga es relativo al directorio de trabajo de la aplicación, lo que permite configuraciones de implementación flexibles.

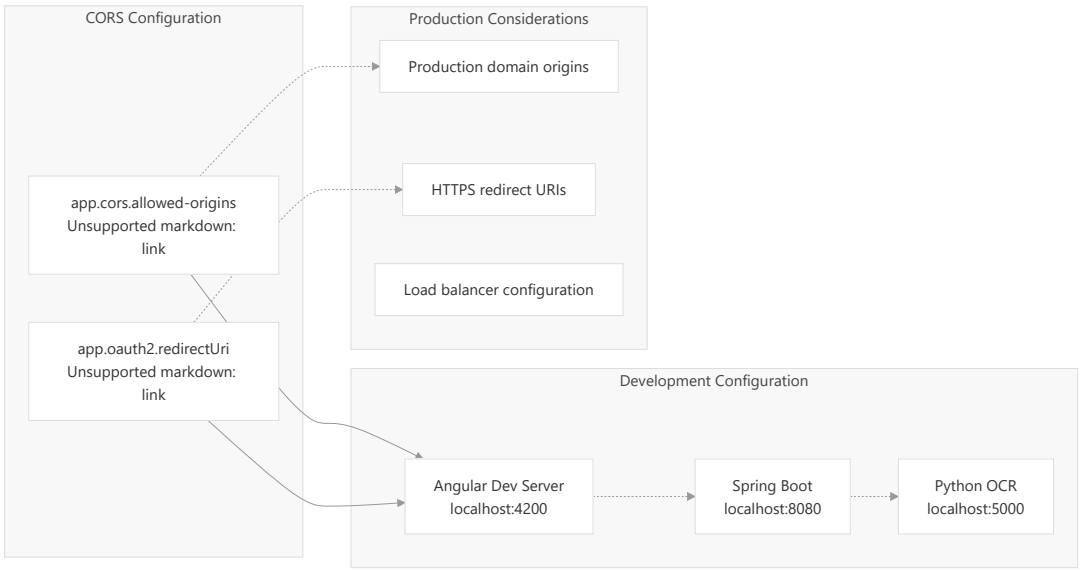
CORS y configuración de origen cruzado

Configuración de CORS

La aplicación configura el uso compartido de recursos de origen cruzado (CORS) para permitir la comunicación entre el frontend de Angular y el backend de Spring Boot que se ejecutan en diferentes puertos.

Propiedad	Valor	Objetivo
`app.cors.allowed-origins`	`http://localhost:3000`	Origen de frontend permitido
`app.oauth2.redirectUri`	`http://localhost:3000/oauth2/redirect`	URL de redireccionamiento de OAuth2

Configuración de desarrollo vs. configuración de producción



Nota: El origen configurado `localhost:3000` parece ser para una configuración de frontend diferente a la del servidor de desarrollo Angular (normalmente `localhost:4200`), lo que sugiere que puede estar configurado para una compilación de producción o una configuración de frontend alternativa.

Configuración de la plantilla de Thymeleaf

La aplicación incluye capacidades de renderizado del lado del servidor para interfaces administrativas utilizando plantillas de Thymeleaf.

Propiedad	Valor	Descripción
`spring.thymeleaf.prefix`	`classpath:/templates/`	Ubicación del archivo de plantilla
`spring.thymeleaf.suffix`	`.html`	Extensión de archivo de plantilla
`spring.thymeleaf.cache`	`false`	Deshabilitar el almacenamiento en caché de plantillas para el desarrollo

Esta configuración admite los controladores de interfaz web que proporcionan paneles administrativos renderizados del lado del servidor junto con la API REST principal.

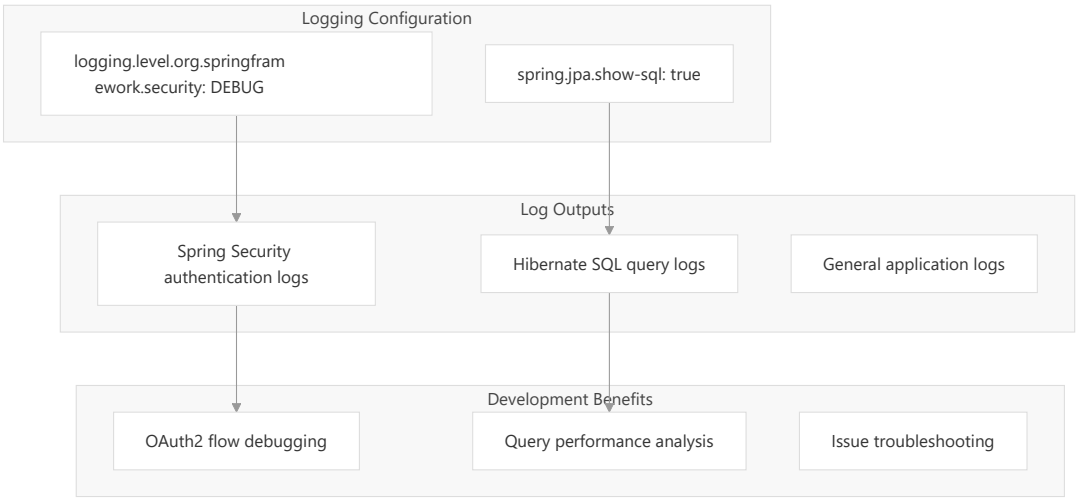
Marca y registro de aplicaciones

Banner de aplicación personalizado

La aplicación utiliza una configuración de banner personalizada para la visualización de inicio:

Propiedad	Objetivo
`spring.banner.location`	Ubicación del archivo del banner personalizado
`spring.banner.charset`	Codificación de caracteres para banner
`application.version`	Marcador de posición de la versión del proyecto Maven
`application.name`	Marcador de posición del nombre del proyecto Maven

Configuración de registro



El registro de nivel de depuración para Spring Security es particularmente útil para solucionar problemas en los flujos de autenticación OAuth2 y el procesamiento de tokens JWT.