Sistema de procesamiento OCR

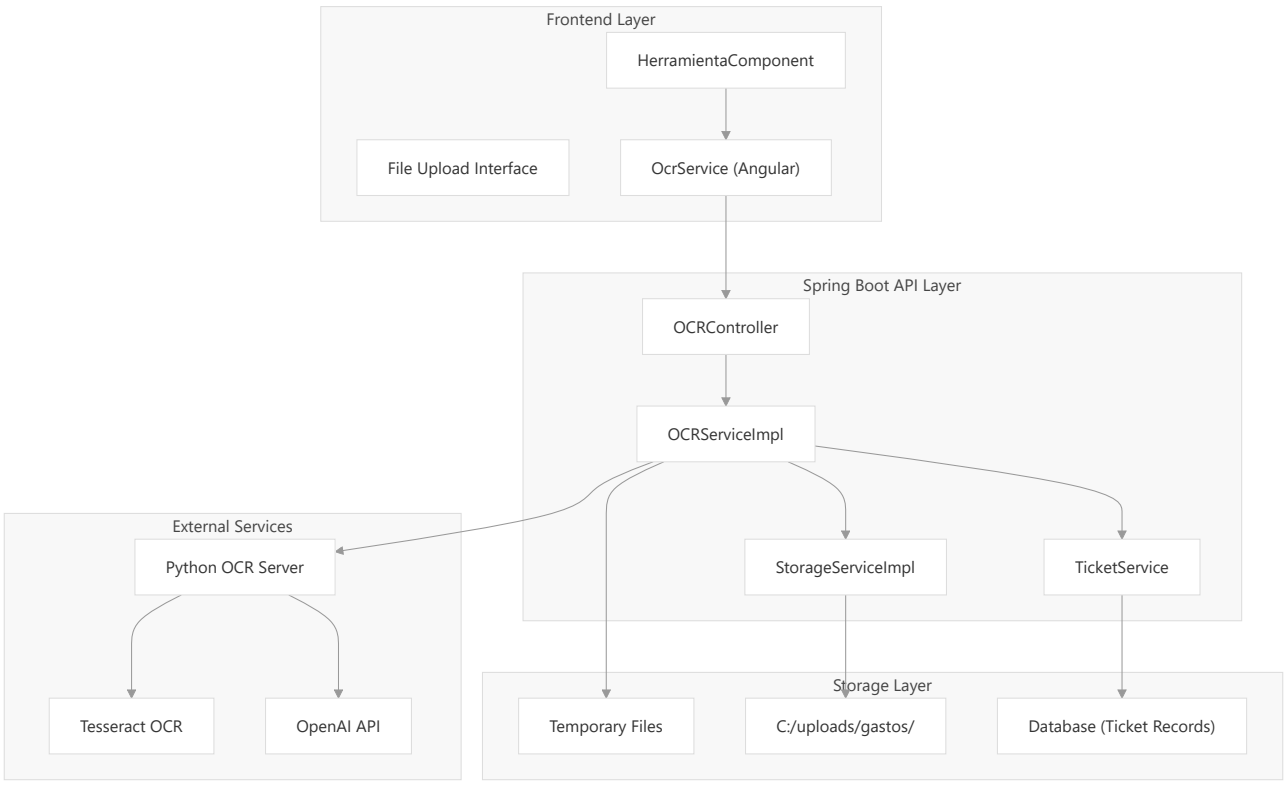
Propósito y alcance

El sistema de procesamiento OCR es la función principal de la aplicación de gestión de gastos, que extrae automáticamente datos estructurados de las imágenes de recibos y tickets. Este sistema permite a los usuarios cargar recibos físicos como imágenes o archivos digitales (PDF) y rellena automáticamente los registros de gastos con la información extraída, como nombres de tiendas, importes totales, valores de impuestos y detalles de cada producto.

Este documento abarca la capa de servicio de Spring Boot que coordina el procesamiento de OCR, los puntos finales de la API REST para la carga de archivos y la integración con el servicio externo de OCR de Python. Para obtener más información sobre la implementación del servidor de OCR de Python, consulte Servidor de OCR de Python . Para obtener información sobre los patrones de integración de servicios backend, consulte Servicios backend .

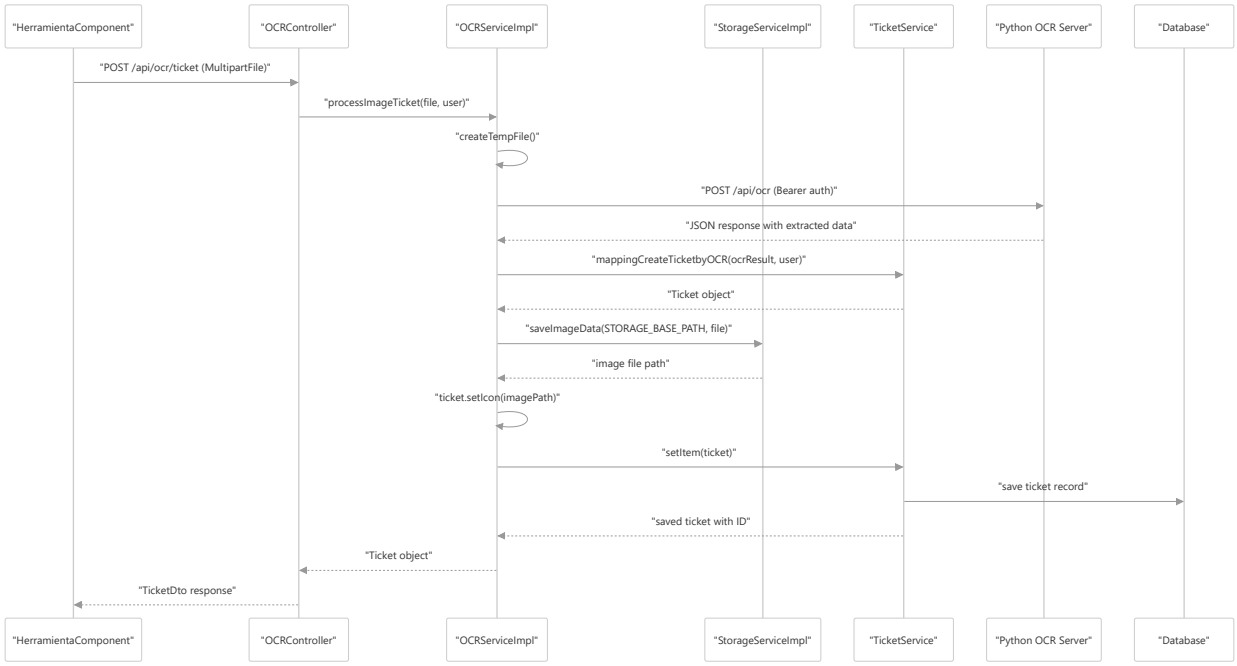
Descripción general de la arquitectura del sistema

Arquitectura del sistema de procesamiento de OCR



Flujo de procesamiento de OCR

Secuencia completa de procesamiento de OCR



Puntos finales y controladores de API

Expone `OCRController` dos puntos finales REST principales para el procesamiento de OCR:

Punto final de procesamiento de tickets de imagen

Método	Camino	Descripción
CORREO	`/api/ocr/ticket`	Procesar archivos de imagen (JPG, PNG) con creación completa de tickets
CORREO	`/api/ocr/ticketdigital`	Procesar archivos digitales (PDF, texto) solo con extracción JSON

Implementación del punto final del ticket de imagen:

- Acepta `multipart/form-data` con nombre de parámetro `archivo`
- Requiere autenticación de token de portador a través de `@AuthenticationPrincipal CustomUserDetails`
- Valida que el archivo no esté vacío antes de procesarlo

- Devuelve `TicketDto` un objeto o un mensaje de error
- Pide `OCRServiceImpl.processImageTicket()` tramitación completa

Implementación del punto final del ticket digital:

- Estructura similar pero llamadas `OCRServiceImpl.proccessDigitalTicket()`
- Devuelve solo datos JSON sin crear un registro de base de datos
- Se utiliza para flujos de trabajo de vista previa/validación

Detalles de implementación del servicio

La `OCRServiceImpl` clase coordina todo el flujo de trabajo de procesamiento de OCR:

Métodos básicos

Método	Objetivo	Tipo de retorno
`processImageTicket()`	Procesamiento completo de imágenes con persistencia de base de datos	`Ticket`
`proccessDigitalTicket()`	Procesamiento de archivos digitales sin persistencia	`Ticket`
`sendFileForOCR()`	Comunicación con el servicio OCR de Python	`String`
`getStatus()`	Comprobación del estado del servidor OCR de Python	`StatusServerResponse`

Flujo de trabajo de procesamiento de archivos

1. **Creación de archivo temporal** : crea un archivo temporal con el patrón `ticket_` + nombre de archivo original
2. **Llamada de servicio OCR** : envía un archivo al servidor Python a través de `sendFileForOCR()`
3. **Mapeo de datos** : se utiliza `TicketService.mappingCreateTicketbyOCR()` para convertir JSON a `Ticket`
4. **Almacenamiento de imágenes** : guarda una copia permanente mediante `StorageServiceImpl.saveImageData()`
5. **Persistencia de la base de datos** : guarda el registro del ticket mediante `TicketService.setItem()`

Propiedades de configuración

El servicio utiliza una configuración externalizada para la integración del servidor Python:

```
@Value("${python.server.url}")
private String pythonServerUrl;

@Value("${python.server.apiKey}")
private String pythonServerApiKey;
```

Integración con el servidor OCR de Python

Detalles de la comunicación HTTP

Configuración de la solicitud:

- Utiliza Spring `RestTemplate` para la comunicación HTTP
- Envía archivos como `multipart/form-data` con `FileSystemResource`
- Incluye autenticación de token de portador en `Authorization` el encabezado
- Rutas a diferentes puntos finales según el tipo de archivo:
  - `/api/ocr` para archivos de imagen
  - `/api/ocr-file` para documentos digitales

Manejo de errores:

- Capturas `HttpClientErrorException` y `HttpServerErrorException`
- Convierte errores HTTP en `IOException` códigos de estado
- Registra respuestas tanto de éxito como de error para la depuración.

Autenticación y seguridad

La integración utiliza autenticación basada en clave API:

```
headers.set("Authorization", "Bearer " + pythonServerApiKey);
```

Monitoreo del estado del servidor

El `getStatus()` método proporciona una comprobación del estado de salud:

- Devuelve `StatusServerResponse` indicadores booleanos para los componentes del servidor
- Maneja con elegancia la falta de disponibilidad del servidor al devolver un estado completamente falso
- Se utiliza para la monitorización y el diagnóstico del sistema.

Integración de frontend

Integración de componentes angulares

Maneja `HerramientaComponent` la interfaz de usuario para cargas de archivos OCR:

Características principales:

- Selección de archivos con generación de vista previa para imágenes
- Gestión del estado de carga durante el procesamiento
- Manejo de errores con mensajes fáciles de usar
- Redirección automática para editar formularios después del procesamiento

Flujo de procesamiento:

1. El usuario selecciona el archivo a través `onFileSelected()` del controlador de eventos
2. Vista previa de la imagen generada `FileReader` para obtener retroalimentación visual
3. `subirArchivoYEditar()` llama al `OcrService` método apropiado
4. El procesamiento exitoso redirecciona a `/protected/form/ticket/{ticketId}`

Mapeo de métodos de servicio:

- `ticketimagen` tipo → `ocrService.procesarTicketImagen()`
- `ticketdigital` tipo → `ocrService.procesarTicketDigital()`

Manejo de errores y monitoreo de estado

Estrategia de gestión de errores

Manejo de errores de nivel de servicio:

- Operaciones de limpieza: elimina las imágenes almacenadas si falla el procesamiento de OCR
- Envoltura de excepciones: convierte las excepciones marcadas en `RuntimeException`
- Registro: salida de depuración en etapas clave de procesamiento

Manejo de errores a nivel de controlador:

- Valida la presencia del archivo antes de procesarlo
- Devuelve los códigos de estado HTTP apropiados (400, 500)
- Proporciona mensajes de error descriptivos al frontend

Manejo de errores de frontend:

- Muestra mensajes de error fáciles de usar
- Restablece el estado de carga en caso de errores
- Registra información detallada de errores en la consola

Monitoreo de la salud del sistema

El sistema incluye capacidades de verificación de salud a través de `getStatus()`:

- Supervisa la disponibilidad del servidor OCR de Python
- Devuelve información de estado estructurada
- Permite la monitorización proactiva del sistema