Entorno de desarrollo

Este documento describe la configuración, el entorno de desarrollo y los procesos de compilación del sistema de gestión de gastos ProyectoGestorGastos. Incluye la configuración del entorno, la configuración del desarrollo local, los procedimientos de prueba y los requisitos de las herramientas de desarrollo en la arquitectura de tres niveles (frontend de Angular, backend de Spring Boot y servicio OCR de Python).

Para conocer las consideraciones sobre la implementación en producción, consulte Configuración e implementación . Para obtener una descripción general de la arquitectura del sistema, consulte Arquitectura del sistema .

Configuración del entorno

El sistema utiliza archivos de configuración específicos del entorno para administrar diferentes escenarios de implementación e integraciones de servicios externos.

Configuración del entorno de frontend angular

La aplicación Angular utiliza archivos de entorno para configurar los puntos finales de API y el comportamiento de la aplicación en entornos de desarrollo y producción.

Configuración del entorno de desarrollo

`ServidorWebFrontend/gesthorAngular/src/app/environments/environment.ts` `8-26` define la configuración del entorno de desarrollo:

```
export const environment = {
  production: false,
  apiUrl: 'http://localhost:8080',
  apiName: "GesThor-01"
};
```

Configuración del entorno de producción

`ServidorWebFrontend/gesthorAngular/src/app/environments/environment.prod.ts` `8-26` Contiene configuraciones específicas de producción:

```
export const environment = {
  production: true,
  apiUrl: 'http://localhost:8080',
  apiName: "GesThor-01"
};
```

Los componentes hacen referencia a la configuración del entorno para la comunicación API, como se ve en `ServidorWebFrontend/gesthorAngular/src/app/components/edit-form-profile/edit-form-profile.component.ts` 98 :

```
server: string = `${environment.apiUrl}/`;
```

Configuración del entorno de backend

El backend de Spring Boot utiliza archivos de propiedades de la aplicación para la gestión de la configuración. El archivo de configuración principal se excluye del control de versiones, como se indica en `.gitignore` 1 :

```
ServidorServicioAPI/GestorAPI/src/main/resources/application.properties
```

Configuración del entorno del servicio OCR de Python

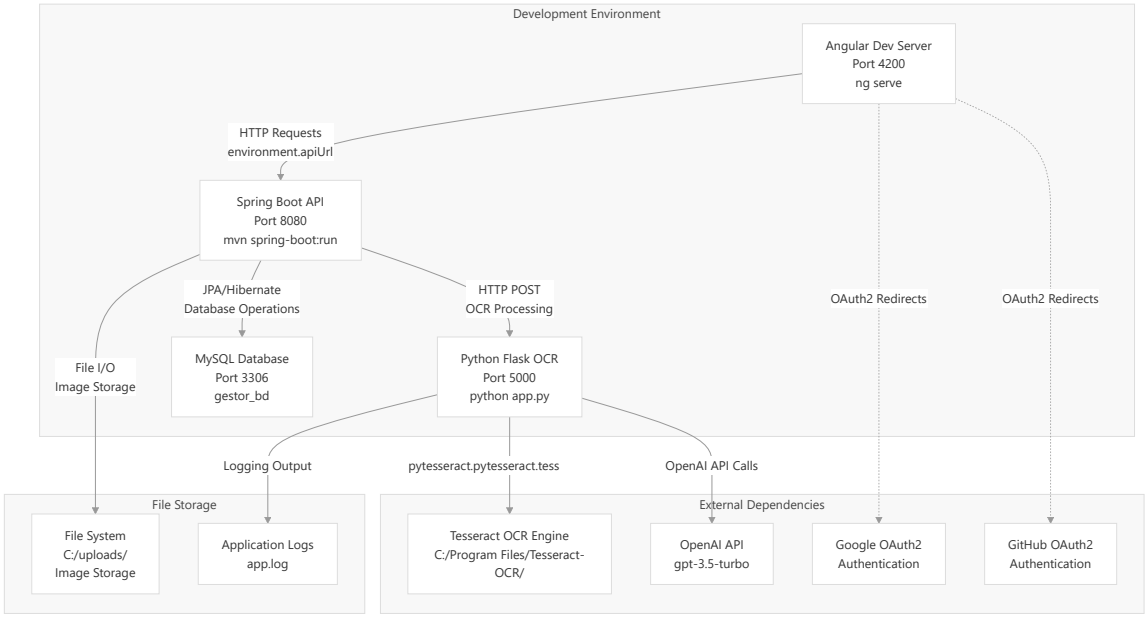
El servicio OCR de Python utiliza archivos de entorno para la configuración sensible. El archivo de entorno se excluye del control de versiones, como se muestra en `.gitignore` 2 :

```
ServidorPython\server\ApiRestOCR+AI\.env
```

Arquitectura de desarrollo local

El entorno de desarrollo consta de múltiples servicios que se ejecutan en diferentes puertos con dependencias y requisitos de configuración específicos.

Arquitectura de servicios de desarrollo



Configuración del desarrollo de OCR

El servicio de OCR requiere dependencias locales específicas para el desarrollo y las pruebas. La configuración del motor de OCR de Tesseract se muestra en `ServidorServicioOCR/ServidorPython/server/OldTestInfo/TesseractTest.py` `6` :

```
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"
```

Desarrollo de la línea de pruebas de OCR

El entorno de desarrollo incluye capacidades de prueba de OCR con preprocesamiento de imágenes, como se demuestra en `ServidorServicioOCR/ServidorPython/server/OldTestInfo/TesseractTest.py` `7-17` :

```
img = Image.open("Tests/ticket-zara-1.jpg")
img_gray = img.convert('L')
enhancer = ImageEnhance.Contrast(img_gray)
img_contrast = enhancer.enhance(2)
img_sharp = img_contrast.filter(ImageFilter.SHARPEN)
img_np = np.array(img_sharp)
img_bin = cv2.threshold(img_np, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]
img_bin_pil = Image.fromarray(img_bin)
custom_config = r'--oem 3 --psm 6'
texto = pytesseract.image_to_string(img_bin_pil, config=custom_config)
```

Flujo de configuración de desarrollo

La gestión de la configuración en el entorno de desarrollo sigue un enfoque jerárquico con anulaciones específicas del entorno y descubrimiento de servicios.

Flujo de dependencias de configuración

Herramientas de desarrollo y dependencias

El entorno de desarrollo requiere herramientas y bibliotecas específicas para cada nivel de servicio.

Pila de desarrollo de frontend angular

Componente	Objetivo	Configuración
Angular 18	Marco de interfaz de usuario	`ng serve` para servidor de desarrollo
CSS de Bootstrap	Estilo de la interfaz de usuario	Componentes de diseño responsivo
Chart.js/ng2-charts	Visualización de datos	Gráficos y análisis del panel de control
RxJS	Cliente HTTP	Comunicación API con backend

Pila de desarrollo backend de Spring Boot

Componente	Objetivo	Configuración
Bota de primavera 3	Marco de backend	`mvn spring-boot:run` para el desarrollo
Seguridad de primavera + JWT	Autenticación	Seguridad basada en tokens
Cliente OAuth2	Autenticación social	Integración de Google y GitHub
Datos de primavera JPA	Acceso a datos	Operaciones de base de datos MySQL
Hoja de tomillo	Plantillas del lado del servidor	Representación de la interfaz de administración

Pila de desarrollo de servicios de OCR de Python

Las dependencias del servicio OCR se muestran en las declaraciones de importación de

`ServidorServicioOCR/ServidorPython/server/OldTestInfo/TesseractTest.py` `1-5` :

Componente	Objetivo	Declaración de importación
pytesseract	Interfaz del motor OCR	`import pytesseract`
PIL (Almohada)	Procesamiento de imágenes	`from PIL import Image, ImageEnhance, ImageFilter`
OpenCV	Visión por computadora	`import cv2`
NumPy	Procesamiento numérico	`import numpy as np`
Matraz	Marco de API REST	Alojamiento de servicios web
OpenAI	Procesamiento de texto con IA	`openai==0.28.0`

Configuración de desarrollo de carga y almacenamiento de archivos

El entorno de desarrollo gestiona la carga de archivos a través del servicio backend con configuración de almacenamiento. La funcionalidad de carga de archivos se implementa en componentes como

`ServidorWebFrontend/gesthorAngular/src/app/components/edit-form-profile/edit-form-profile.component.ts` `156-188`

que demuestra el patrón de desarrollo para manejar cargas de archivos multiparte:

```
onFileSelected(event: Event): void {
    const input = event.target as HTMLInputElement;
    if (!input.files || input.files.length === 0) return;

    const file = input.files[0];
    const formData = new FormData();
    formData.append('image', file);
```

```
        if (this.spent == null) {
            // Profile image upload
            this.userService.subirFotoPerfil(formData).subscribe({
                next: (res) => {
                    this.profile = res.url;
                    this.save.emit({ field: this.field, value: this.profile });
                },
                error: () => {
                    alert('Error al subir la imagen.');
                }
            });
        } else {
            // Expense image upload
            this.spentService.subirFotoGasto(this.spent, file).subscribe({
                next: (res) => {
                    this.profile = res.url;
                    this.save.emit({ field: this.field, value: this.profile });
                },
                error: () => {
                    alert('Error al subir la imagen.');
                }
            });
        }
    }
```

Procesos de desarrollo, pruebas y compilación

El entorno de desarrollo incluye capacidades de prueba y procesos de compilación para cada nivel de servicio.

Configuración de pruebas del servicio OCR

La configuración de prueba de OCR muestra el flujo de trabajo de desarrollo para validar la funcionalidad de OCR con diferentes técnicas de procesamiento de imágenes. La configuración de prueba en `ServidorServicioOCR/ServidorPython/server/OldTestInfo/TesseractTest.py` | `15-16` muestra:

```
custom_config = r'--oem 3 --psm 6'
texto = pytesseract.image_to_string(img_bin_pil, config=custom_config)
```

Patrones de desarrollo de componentes

Los componentes angulares siguen patrones de desarrollo que facilitan las pruebas y el mantenimiento.

`ServidorWebFrontend/gesthorAngular/src/app/components/tool-group/tool-group.component.ts` | `53-61` demuestra la estructura de desarrollo para componentes reutilizables:

```
@Input() herramientas = [
    {
        nombre: 'Nombre',
        icono: '/icon/icons8-pencil.gif',
        descripcion: 'Descripcion',
        accion: 'Cargar',
        ruta: '/ruta'
    }
];
```

Gestión de activos de desarrollo

El entorno de desarrollo incluye activos estáticos para los componentes de la interfaz de usuario, como se hace referencia en `ServidorWebFrontend/gesthorAngular/src/app/components/tool-group/tool-group.component.ts:56` y almacenado en `ServidorWebFrontend/gesthorAngular/public/icon/icons8-pencil.gif:1-51`