Manuel Técnico IA_Practica1

Universidad de San Carlos de Guatemala
Inteligencia Artificial
Escuela de Sistemas
Ing. Luis Espino
Aux. Erick Sandoval.

Nombre	Carné
Danny Eduardo Cuxum Sanchez	201709528

Objetivos

- Comprender la aplicación de Sprin-boot para la generación de un server en Java
- Comprender el funcionamiento de la librería de google.
- Poner en practica las funcionalidades que ofrece la librería y el manejo de las respuestas del servicio para mostrarlo en un ambiente web.

¡Bienvenido al manual Técnico de la aplicación de análisis de imágenes!

Introducción

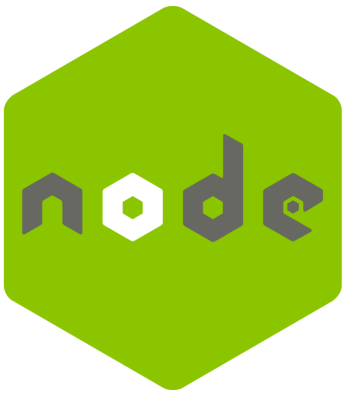
La aplicación de análisis de imágenes es una herramienta que te permite analizar imágenes cargadas para detectar rostros y evaluar el contenido de seguridad de la imagen. Utiliza la API de Cloud Vision de Google para llevar a cabo estas tareas. La aplicación está construida utilizando el framework Spring Boot en el backend y React en el frontend.

Requisitos del Sistema

Para utilizar la aplicación, asegúrate de tener instalado lo siguiente:

- Java Development Kit (JDK) 8 o superior

- Node.js y npm (Node Package Manager):



- Un navegador web moderno como Google Chrome, Mozilla Firefox, o Microsoft Edge

- React para la creación del Front.

Instalación y Configuración

1. Clona el repositorio de la aplicación desde GitHub.
2. Abre una terminal y navega hasta el directorio raíz del proyecto.
3. Ejecuta el comando `npm install` para instalar las dependencias del frontend.
4. Ejecuta el comando `./mvnw spring-boot:run` para ejecutar la aplicación Spring Boot.

Uso de la Aplicación

Una vez que la aplicación esté en funcionamiento, puedes acceder a ella a través de tu navegador web visitando la dirección `http://localhost:8080`.

1. **Cargar una Imagen**:
   - Haz clic en el botón "Seleccionar Archivo" para cargar una imagen desde tu computadora.
   - Una vez seleccionada la imagen, haz clic en el botón "Cargar" para cargarla en la aplicación.
2. **Analizar la Imagen**:
   - Después de cargar la imagen, haz clic en el botón "Procesar" para iniciar el análisis.
   - La aplicación enviará la imagen al servidor para su análisis utilizando la API de Cloud Vision de Google.

3. **Visualizar Resultados**:
    - Una vez completado el análisis, la aplicación mostrará la imagen analizada junto con los resultados del análisis.
    - Si la imagen contiene contenido inapropiado, se aplicará un filtro de desenfoque a la imagen y se mostrará una alerta.
    - Si la imagen es considerada segura, se mostrará una alerta indicando que la imagen es válida.

Componentes de la Aplicación

La aplicación consta de los siguientes componentes principales:

EjemploSpringApplication.java:

- Clase principal que inicia la aplicación Spring Boot.
- Configura la aplicación para permitir solicitudes CORS desde `http://localhost:3000`.

VisionController.java:

- Controlador que maneja las solicitudes relacionadas con el análisis de imágenes.
- Contiene el método `analyzeImage` que procesa las imágenes cargadas y las envía a la API de Cloud Vision de Google para su análisis.

Explicación más detallado del funcionamiento del Endpoint principal de la aplicación.

1. **RequestMapping**:
    - Este método está anotado con `@PostMapping("/analyze-image")`, lo que significa que responde a las solicitudes HTTP POST en la ruta `/analyze-image`.
2. **Parámetros del Método**:
    - El método recibe un parámetro `@RequestParam("file") MultipartFile file`, que representa el archivo de imagen enviado por el cliente. `MultipartFile` es

una clase de Spring que maneja los archivos adjuntos en las solicitudes HTTP.

3. **Obtención del Contenido de la Imagen**:
    - Se lee el contenido del archivo de imagen y se convierte a una cadena base64 para poder enviarlo en el cuerpo de la solicitud HTTP.
4. **Construcción del Cuerpo de la Solicitud**:
    - Se construye el cuerpo de la solicitud en formato JSON. Este cuerpo incluye la imagen en formato base64 y especifica los tipos de características que se desean analizar, como detección de rostros (`FACE_DETECTION`) y detección de contenido inapropiado (`SAFE_SEARCH_DETECTION`).
5. **Establecimiento de Headers**:
    - Se configuran los encabezados de la solicitud, incluyendo el tipo de contenido (`application/json`) y el tipo de contenido que se aceptará en la respuesta.
6. **Creación de la Entidad de la Solicitud**:
    - Se crea una entidad `HttpEntity` que encapsula el cuerpo de la solicitud y los encabezados.
7. **Envío de la Solicitud**:
    - Se utiliza el `RestTemplate` para enviar una solicitud HTTP POST a la URL especificada, con la entidad de la solicitud. La respuesta se guarda en un objeto `ResponseEntity<String>`.
8. **Procesamiento de la Respuesta**:
    - Se maneja la respuesta recibida del servidor. Si la solicitud fue exitosa, se devuelve una respuesta HTTP 200 OK con el cuerpo de la respuesta de la API de Google Cloud Vision. Si ocurre algún error durante el proceso, se devuelve una respuesta HTTP 500 Internal Server Error con un mensaje de error.

RestTemplateConfig.java:

- Configuración de Spring para crear un bean de `RestTemplate`, que se utiliza para realizar solicitudes HTTP a la API de Cloud Vision.

Frontend de la Aplicación en React

El frontend de la aplicación está construido utilizando React, un framework de JavaScript para construir interfaces de usuario interactivas. El código proporcionado es la parte principal del frontend y se encarga de la interfaz de usuario, así como de la interacción con el usuario y el servidor backend.

Componentes Principales

El código del frontend incluye varios componentes principales:

1. **TableComponent**:
    - Este componente muestra una tabla con los resultados del análisis de la imagen. Recibe dos props: `Data` para los datos de la imagen y `Percentage` para el porcentaje de cada categoría.
2. **Principal**:
    - Este es el componente principal que contiene toda la lógica y la interfaz de usuario de la aplicación.
    - Utiliza estados (`useState`) para manejar la visibilidad de las alertas, el porcentaje de cada categoría, la información de la imagen, y el número de caras detectadas.
    - Utiliza `useEffect` para aplicar filtros a la imagen basados en los resultados del análisis y mostrar alertas según la adecuación de la imagen.
    - Implementa funciones para cargar y analizar imágenes, así como para procesar los resultados del análisis.

Funcionalidad Principal

El componente `Principal` permite al usuario cargar una imagen, analizarla y visualizar los resultados del análisis. La funcionalidad principal se describe a continuación:

1. **Cargar Imagen**:
    - El usuario puede seleccionar una imagen desde su dispositivo haciendo clic en el botón "Importar".
    - Una vez seleccionada la imagen, se muestra en pantalla y se dispara una alerta de éxito utilizando SweetAlert.
2. **Analizar Imagen**:
    - Después de cargar la imagen, el usuario puede hacer clic en el botón "Analizar" para enviarla al servidor backend para su análisis.
    - Se muestra una alerta de éxito cuando se completa el análisis.
    - Los resultados del análisis se procesan y se muestran en una tabla, incluyendo el número de caras detectadas y el porcentaje de cada categoría.
3. **Visualización de Resultados**:
    - Los resultados del análisis se muestran en una tabla junto con el número de caras detectadas.
    - Si la imagen no es adecuada para la institución (según los criterios establecidos), se muestra una alerta de error.
    - Si la imagen es válida, se muestra una alerta de éxito.

Solución de Problemas

Si encuentras algún problema mientras usas la aplicación, puedes intentar lo siguiente:

- Asegúrate de tener una conexión a Internet estable.
- Verifica que estés utilizando una versión compatible de Java y Node.js.
- Consulta los registros de la aplicación y del servidor para obtener información sobre errores específicos.