PROYECTO II – DOCUMENT ANALYZER 2.0

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos I (CE 1103)
Primer Semestre 2021
Valor 20%

OBJETIVO GENERAL

- ➔ Construir una aplicación web y un REST API aplicando los principios de arquitectura vistos en clase

OBJETIVOS ESPECÍFICOS

- ➔ Aplicar conceptos de branch strategies
- ➔ Implementar un pipeline de CI/CD
- ➔ Aprender sobre Cloud Storage
- ➔ Aprender sobre NLP
- ➔ Aplicar conceptos de Arquitectura de Software

DESCRIPCIÓN DEL PROBLEMA

Document Analyzer 1.0 fue un éxito. Sin embargo, dicha prueba de concepto introdujo algunas preocupaciones en los stakeholders y usted en su rol de líder de Arquitectura de Software, deberá encargarse de resolverlas:

SEGURIDAD

La forma en que se manejó la seguridad en la versión 1.0 fue insuficiente. Necesita tener un componente independiente que maneje la autenticación, autorización y que permita Single Sign-On. Para esto puede usarse un producto de Identity Manager Open Source como Apache Syncope o similar. Deberá investigar cómo integrar esto con sus componentes en el backend y con el portal Web.

Link relevante: Apache Syncope 3.0.0-SNAPSHOT - Getting Started Lea cuidadosamente la sección sobre Docker.

Este componente reemplaza la autenticación realizada en el API anteriormente.

ALTA DISPONIBILIDAD

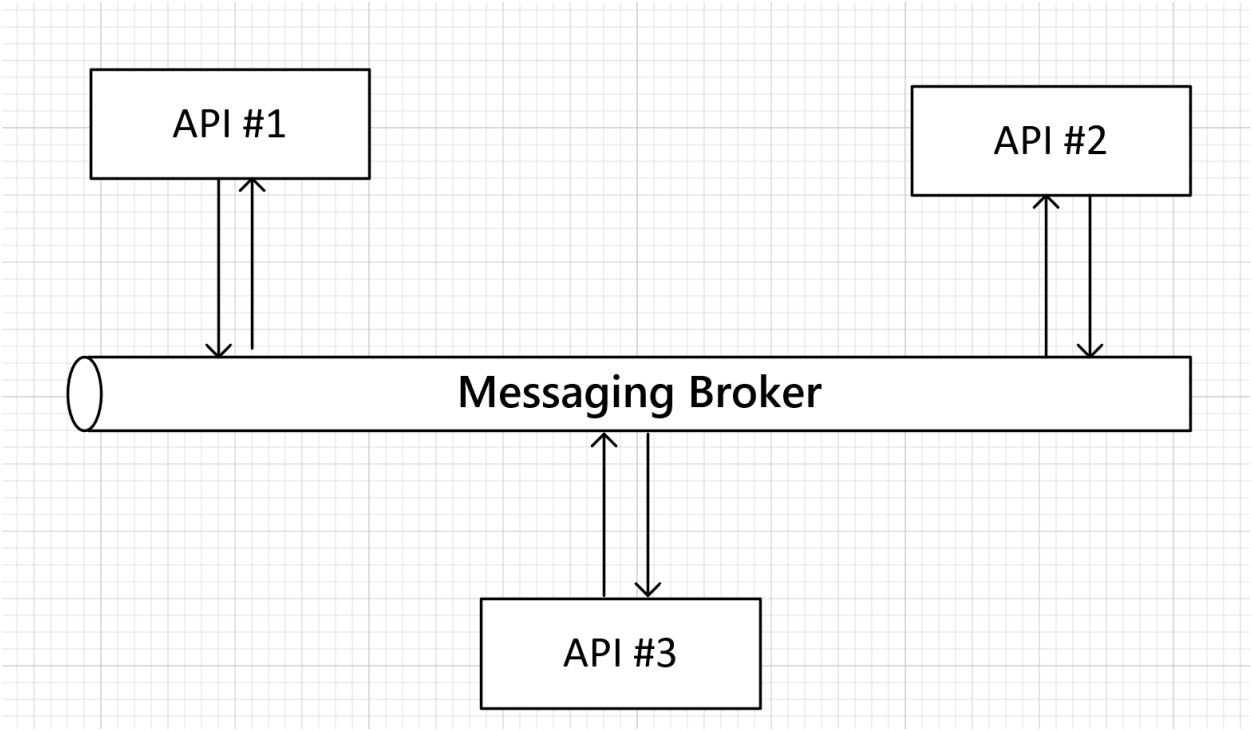
Todos los componentes de su solución necesitan tener alta disponibilidad, por lo que debe haber una réplica de cada componente. Es decir, dos web servers, dos instancias de cada API, entre otras cosas. Esto involucra el uso de algún mecanismo de Load Balancing por lo que deberá usar algún producto open source para este propósito. Puede usar un web

server como HTTPD o NGINX para este propósito

Puntos extra:

1. Tener un esquema activo-pasivo en las bases de datos
2. El identity manager con HA

El análisis de los archivos es un proceso que puede tardarse mucho tiempo. Por lo que su API 1.0 deberá separarse de una forma similar a microservicios. El API de análisis de archivos, debe ser un proceso aparte “que escucha” en una cola de mensajería.



La comunicación entre los API será asíncrona utilizando Rabbit MQ o similar. Estos componentes se comunican mediante un modelo de publicador subscriptor y punto a punto en otros.

Entonces, cuando el usuario sube el archivo satisfactoriamente, se llama un API que solicita el análisis del archivo. Dicho API publica el request de análisis en el broker. Los analizadores suscritos reciben el request y proceden a analizarlo. De igual forma, los analizadores publican el resultado del análisis para que el originador pueda actualizar sus registros internos e indicar en la UI que el archivo ha sido analizado.

USABILIDAD

Deberá mejorar la interfaz gráfica utilizando algún toolkit de UI como IBM Carbon Components, Google Material UI o Microsoft FluentUI. Debe darle una apariencia profesional a su portal web.

Actualmente la UI constantemente pregunta al API el estado del análisis del archivo. Para evitar esta situación, deberá investigar cómo usar WebSockets para notificar a la UI desde el servidor sobre cambios en el análisis del archivo.

NUEVOS FEATURES

No solo basta con identificar las entidades en el documento. Se le solicita implementar los siguientes analizadores:

1. Análisis de Sentimiento: utilizando un servicio de AI de algún proveedor de la nube, determina el sentimiento del documento
2. Análisis de Contenido ofensivo: determina si el contenido del archivo es ofensivo o no. Deberá investigar como hacer esto y puede usar una biblioteca externa

EN SU ROL DE ARQUITECTO O ARQUITECTA

Su frustración ha llegado a niveles máximos. No es claro como documentar la arquitectura y el equipo no se decide. Un modelo de architecture conocido como C4 se ve sencillo y prometedor aunque tiene claras carencias. Ha decidido explorar si este modelo es suficientemente bueno. Así que para esta iteración, seguirá dicho modelo para generar los diagramas.

The C4 model for visualising software architecture

EN SU ROL DE DEVOPS

Igual que en la iteración anterior. Sin embargo, dado que quiere contenerizar su aplicación, el CI/CD debe incluir pasos para construir los contenedores y para publicarlos en DockerHub.

EN SU ROL DE DEVELOPER

Debe utilizar los siguientes lenguajes/tecnologias de programación:

- Frontend: React + Typescript
- Backend: C# .NET Core o Java, Golang y Python

Dado su interés en aprender tecnologías modernas, deberá utilizar Docker como runtime de contenedores y contenerizar cada componente de su arquitectura.

ASPECTOS OPERATIVOS Y EVALUACIÓN

1. El proyecto es en los grupos de trabajo definidos previamente.
2. La fecha de entrega es según lo que se indique en el cronograma del curso

OTRA REGLAS

1. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
2. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la cita revisión oficial
3. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto.
4. Cada grupo tendrá como máximo 30 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
5. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.