Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

Programa de Licenciatura en Ingeniería en Computadores

soa41d: Arquitectura Orientada a Servicios Aplicada a Sistemas Emergentes - GR 1

Proyecto 2 - Kubernetes y Arquitectura orientada a eventos

Realizado por:

Daniel Sing - 2017124999

Adrián López - 2016108981

Profesor:
Alejandra Bolaños Murillo

Octubre 15, 2022

II Semestre 2022

Documentación y justificación de proceso de diseño

Principios SOLID

Single Responsibility Principle (SRP)

Se dividieron los servicios con el fin de que cada uno asumiera un rol en específico, esto permite que el código de los servicios sea más pequeño y fácil de manejar. Cada servicio va a tener una funcionalidad única y exclusiva lo que permite disminuir la dependencias.

Open/Closed Principle (OCP)

El código permite interactuar mediante una página web, en donde el usuario puede ingresar sus imágenes y recibir el resultado de la misma. Además podrá ver el resultado de las imágenes anteriormente procesadas y guardadas en una base de datos local. De modo que, para ambas funciones descritas anteriormente, se tienen dos funciones por aparte que deberán de mantenerse intactas, sin embargo al utilizar FLASK de Python, podemos utilizar un router y otra función para poder extender el uso de la aplicación sin tener que modificar el código ya escrito. De igual manera al utilizar servicios se tiene la ventaja de que al estar desacoplados se puede extender la funcionalidad del software sin cambiar los servicios ya existentes.

Liskov Substitution Principle (LSP)

No aplica por la falta de clases hijas.

Interface Segregation Principle (ISP)

Los usuarios pueden utilizar la aplicación sin depender de los servicios que no necesitan, por ejemplo si el usuario no quiere acceder al servicio de leer los resultados anteriores entonces este no está obligado a hacerlo y la aplicación trabajará sin errores. Incluso se puede correr los servicios sin la interfaz y solo ser llamados por otros servicios.

Dependency Inversion Principle (DIP)

El programa puede utilizar servicios implementados de diferentes formas por lo que él mismo no necesita saber cómo funcionan y se debe generar el mismo resultado.

Se selecciona el patrón broker en el que se evita tener un mediador central y los componentes de comunican entre ellos debido a la simpleza del flujo.

Funcionalidad de los servicios

Web-app

Se encarga de interactuar con el analizador de imágenes, el escritor de la base de datos y el lector de la base de datos. Su función principal es darle al usuario una interfaz gráfica sencilla e intuitiva con el fin obtener las imágenes que se van a procesar y devolver los resultados del análisis en la misma interfaz web. El sistema permite cargar 10 imágenes y luego llama al analizador de imágenes, para ver los resultados llama al lector de la base de datos y muestra la misma.

Analyzer-app

Este servicio tiene el fin de comunicarse con el API de Google Vision, el cual se encarga de analizar las emociones del rostro humano en imágenes y devolver un resultado aproximado de la emoción expresada. Este servicio se comunica con el Web-app del cual recibe la imagen a procesar y retorna el resultado que ofrece el API de Google. Esto va a generar un json con los resultados de las imágenes que se le manden.

Writer-app

Con el fin de mantener una base local y poder ser accedida para escritura y sobreescritura se utiliza este servicio el cual obtiene el resultado del análisis de la imagen y lo almacena en una base de datos local (archivo de texto).

Reader-app

La misma funcionalidad del writer-app pero con función inversa, es decir la operación a realizar es la de leer la base de datos, devolver esa información al web-app para que sea desplegada al usuario como tabla cuando este lo solicite mediante un evento.

Diseño inicial de la arquitectura

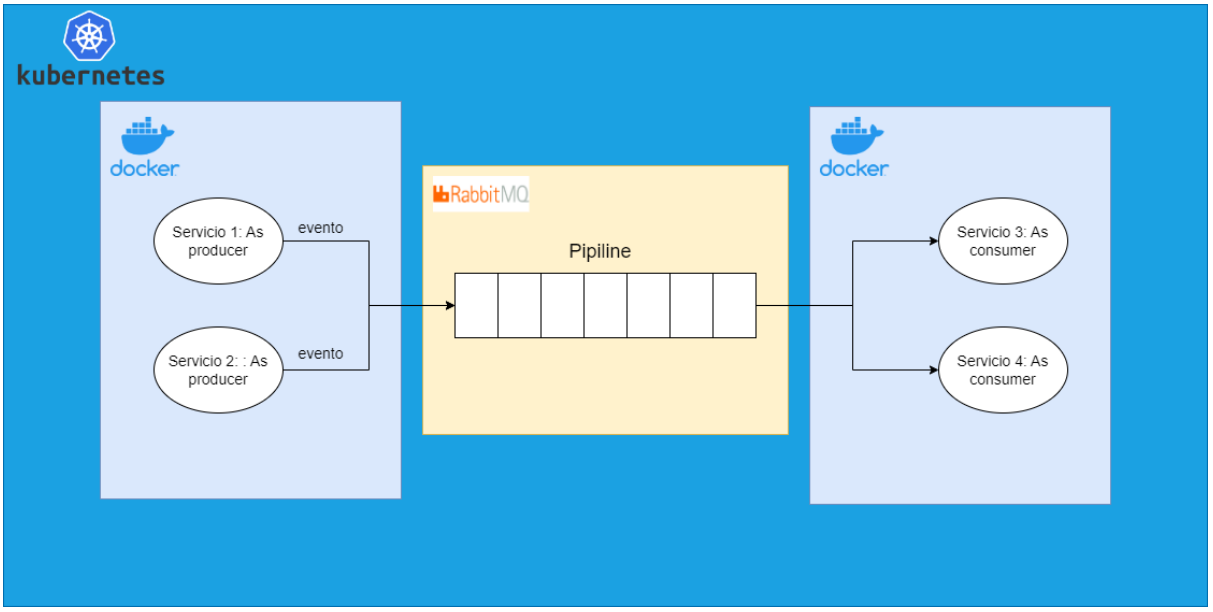

Figura 1. Diagrama de arquitectura prescriptiva

Diseño final de la arquitectura

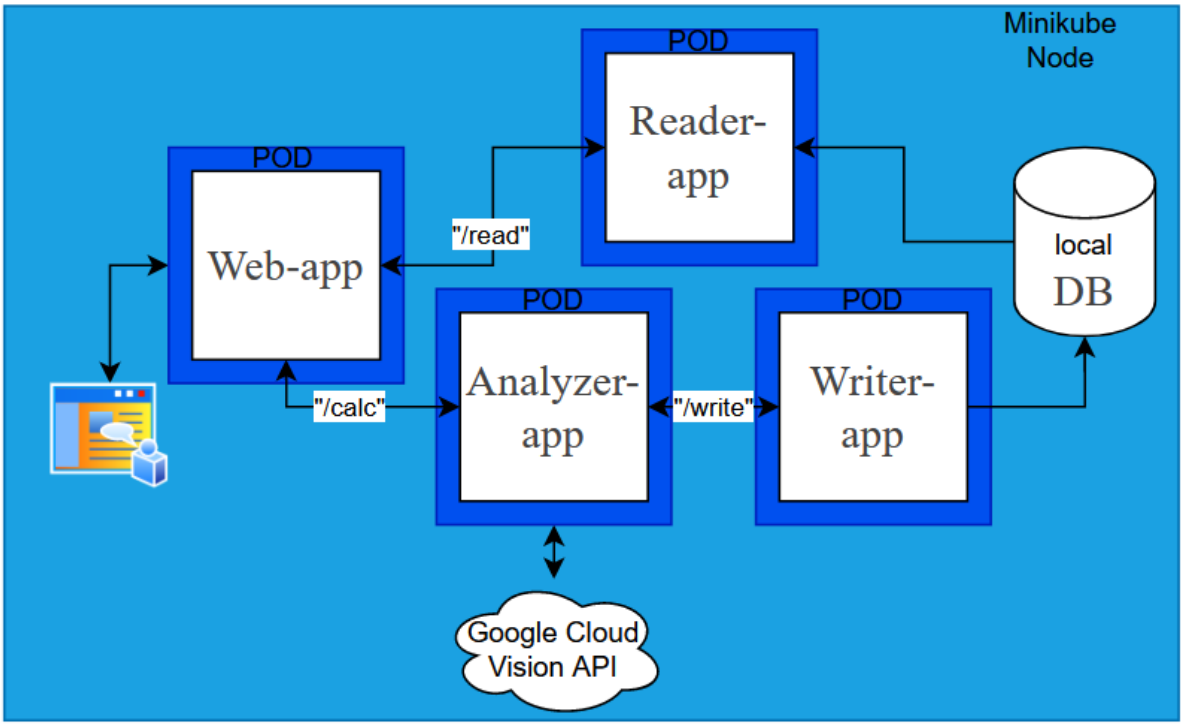

Figura 1. Diagrama arquitectura descriptiva y componentes