Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores
(Computer Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadores
(Licentiate Degree Program in Computer Engineering)

SOA4ID Arquitectura Orientada a Servicios Aplicada a Sistemas Emergentes

Proyecto 2
Documentación
(Documentation)

Realizado por:
Made by:
Jason Salazar González

Profesora:
(Professor)
Alejandra Bolaños

Fecha de entrega:
(Date)
24 de mayo 2022

Introducción

En este proyecto se construye un sistema de software que permite analizar los sentimientos de los empleados de una empresa al recibir buenas noticias.

La funcionalidad del sistema consiste en que al final de cada día laboral, diez imágenes de empleados deben ser analizadas, y el resumen del análisis debe ser agregado a un archivo local común. Estos análisis posteriormente funcionarán a la empresa como herramienta para comunicar mejor los resultados mensuales de las ganancias de la compañía.

Para la implementación de este sistema se emplean contenedores de software. Estos contenedores contienen todos los servicios necesarios para el correcto funcionamiento del sistema desarrollado.

En este documento se detalla el sistema implementado, así como la toma de decisiones de su diseño.

Documentación de proceso de diseño

En esta sección se detalla cómo se elaboró el primer diseño de la arquitectura (Arquitectura prescriptiva) y el diseño final (Arquitectura descriptiva). Además se explica el funcionamiento de cada servicio que componen la arquitectura final.

El primer paso fue identificar las partes principales del problema basado en la especificación del proyecto. Las tareas principales identificadas fueron:

- Cargar imágenes: La idea principal es que era necesario contar un servicio que se encargará de obtener las imágenes a ser procesadas.
- Procesar imagen: La idea era que este servicio se encargará tanto de analizar las imágenes como de generar un archivo de salida de los resultados.

- Guardar resultados: Se plantea crear un servicio que sea el encargado de guardar los resultados de los análisis en un medio persistente como una base de datos.
- Consultar resultados: Se plantea que exista un servicio para que el usuario pueda consultar los resultados almacenados.

Con lo anterior se obtiene un primer diseño de la arquitectura, el cual se muestra en la figura 1.

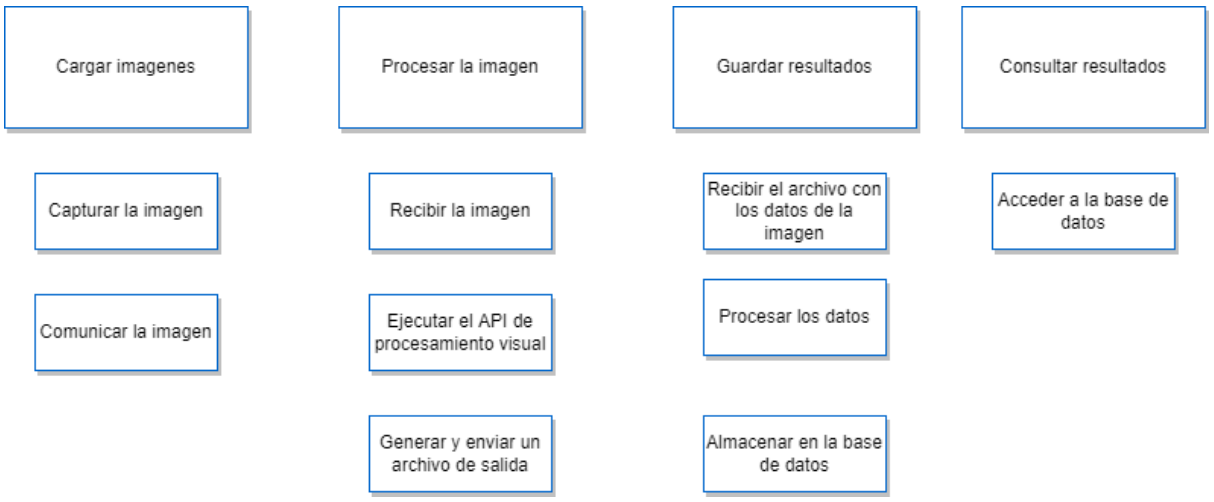

Figura 1. Arquitectura prescriptiva

Después de un análisis más detallado del problema a resolver y habiendo trabajado en la implementación de la solución, se obtiene el diseño de arquitectura final, el cual se muestra en la figura 2.

A continuación se detalla el funcionamiento de cada uno los servicios:

Servicio iniciador

Este servicio permite al usuario poder activar el proceso de análisis de las imágenes. Está compuesto por dos módulos. Uno que se encarga de recibir la

solicitud del usuario. Y otro se encarga de enviar un mensaje de activación al message broker, para que el servicio encargado del análisis de la imagen lea el mensaje y proceda a realizar el análisis.

Servicio analizador de imágenes

Este servicio es el encargado de realizar el análisis de las diez imágenes. Está compuesto por cuatro módulos. El primer módulo se encarga de estar leyendo la cola(del message broker), en busca de un mensaje de activación. Una vez que el mensaje de activación es leído, un segundo módulo se encarga de que se analicen las diez imágenes, para esto utiliza un tercer módulo, el cual tiene la función de encontrar la emoción más probable de un rostro humano en una imagen. Y por último, existe otro módulo de software que se encarga de ir enviando los resultados del análisis de cada imagen, por medio de un mensaje a la cola, esto con la idea de que otro servicio lea los mensajes y se encargue de almacenar los resultados de forma permanente.

Servicio guardar

Este servicio se encarga de almacenar los resultados de los análisis de las imágenes en forma persistente. Este está compuesto por dos módulos. El primer módulo se encarga de estar leyendo los mensajes de la cola, los cuales contienen el resultado del análisis de una imagen, el cual se debe guardar. Una vez leído el mensaje, un segundo módulo es el encargado de guardar de forma persistente el contenido del mensaje en un archivo local común.

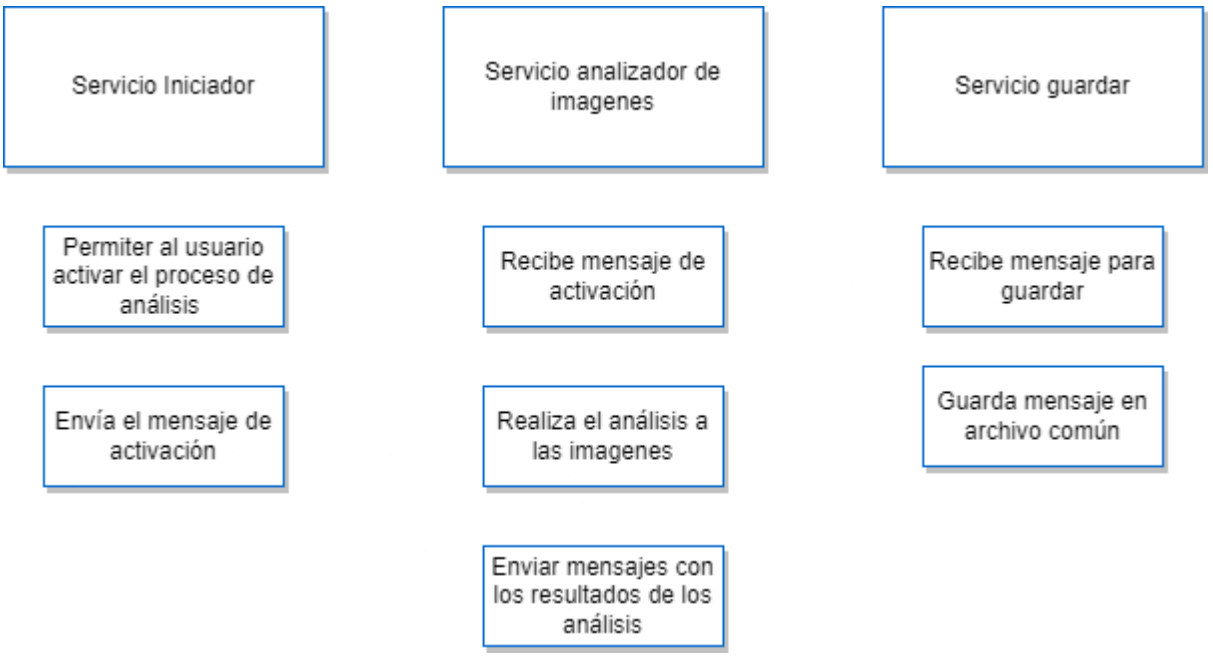

Figura 2. Arquitectura descriptiva

Diagramas de componentes

El diagrama de componentes se muestra en la figura 3. En este se puede ver cómo se relacionan cada uno de los servicios y partes del sistema. Para conectar cada uno de los servicios de la arquitectura se utiliza una message broker, específicamente RabbitMQ.

El flujo de este diagrama es el siguiente:

- El servicio Iniciador por medio del Direct exchange envía un mensaje a Queue1, esto para indicar que el proceso de análisis de imágenes debe comenzar.
- El servicio Análizador lee un mensaje de activación de Queue1. Después de realizar el análisis de una imagen utilizando Cloud Vision API, el servicio envía un mensaje a Queue2 con el resultado de este.

- El servicio Guardar lee de Queue2 los mensajes que representan el resultado de los análisis. Después de cada lectura, este los almacena en un archivo de texto en un directorio local común.

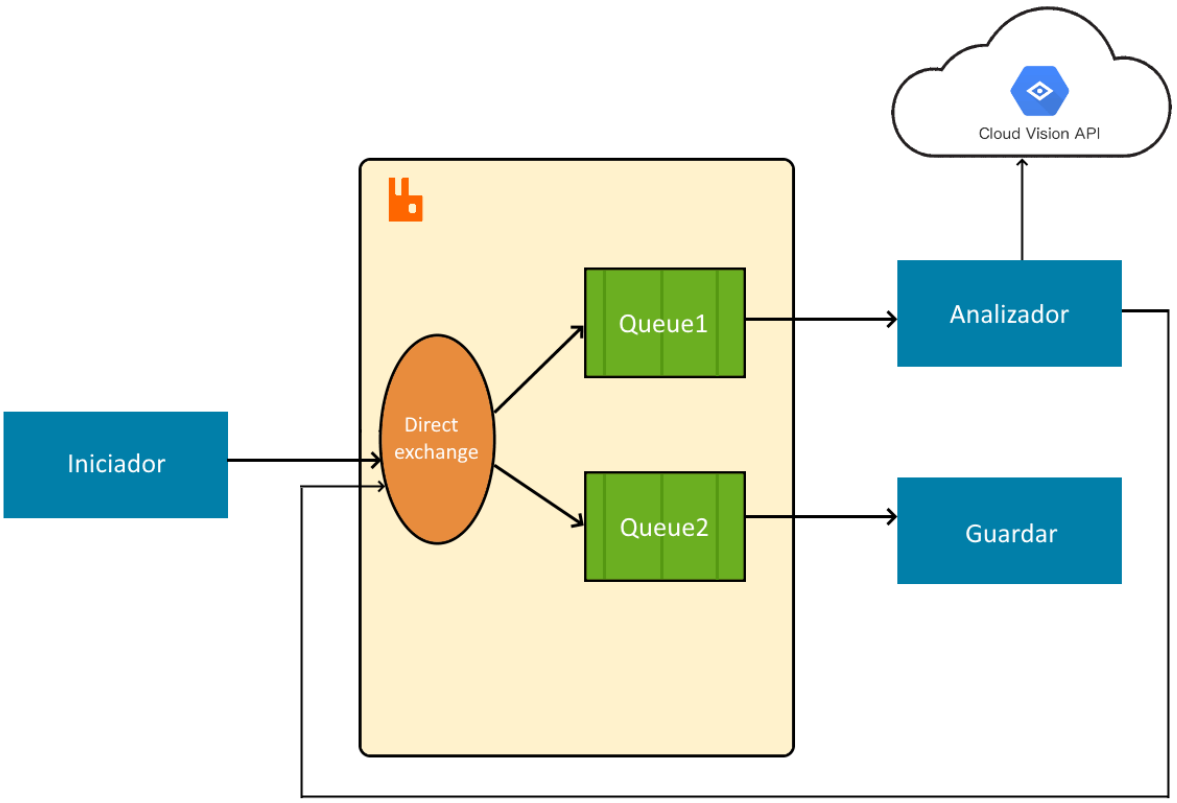

Figura 3. Diagrama de componentes

Justificación de las decisiones de diseño

A continuación se mencionan algunas de las razones que se tomaron en cuenta en la toma de decisiones en el desarrollo de este proyecto.

Message broker

Para gestionar la comunicación entre los servicios del sistema se decide utilizar el message broker RabbitMQ. Ya que este es ligero y fácil de desplegar On Premise y en la nube. Soporta múltiples protocolos de mensajería [1]. RabbitMQ puede

desplegarse en configuraciones distribuidas y federadas para cumplir con los requisitos de alta escala y alta disponibilidad [1].

Minikube

Minikube es la solución ideal para pequeños proyectos basados en contenedores. Permite, por ejemplo, configurar un clúster de Kubernetes en privado sin tener que trabajar directamente con todo un servidor o una nube [2]. Por esta razón se decide utilizar está herramienta.

Google Vision API

Para el caso de realizar el análisis de las imágenes se decide utilizar la herramienta Vision API.

Google Cloud Vision API es una herramienta en la nube de Google que permite analizar gran cantidad de imágenes y extraer información valiosa para comprender su contenido [3]. Proporciona una interfaz RESTful que facilita la tarea de tener que desarrollar algoritmos de procesamiento de imágenes [3]. Por esta razón se decide utilizar esta herramienta.

Principios SOLID

Al diseñar e implementar cada módulo del sistema se trata de apegarse lo máximo posible a los principios SOLID.

Por ejemplo, para cumplir con el principio Single Responsibility, se diseña cada módulo de cada uno de los servicios con una única responsabilidad. Por ejemplo en el servicio **Analizador,** un módulo está encargado únicamente de la lectura de la cola queue1 del message broker. Otro módulo es el encargado de asegurarse de que las diez imágenes serán analizadas. Un tercer módulo tiene como única función detectar la emoción en una imagen. Y por último hay un módulo cuya única función

es ser el encargado de enviar mensajes a la cola queue2 con el resultado del análisis. De forma análoga sucede en la implementación del resto de servicios.

Tipo de almacenamiento de los resultados

Dado que es necesario que los resultados sean consultados fácilmente y tomando en cuenta la simplicidad de los mismos, se decide utilizar un archivo de formato txt para almacenar los resultados. Para esto, cuando se realiza el deployment del servicio encargado de almacenar los resultados, se crea un volumen persistente dentro del servicio, el cual es mapeado a un directorio dentro de Minikube, el cual puede ser fácilmente consultado por otro servicio o incluso ser mapeado a otro directorio fuera de Minikube. Es en este volumen donde se almacenan los resultados.

Referencia bibliográficas

[1] "Messaging that just works". RabbitMQ. https://www.rabbitmq.com/ (accedido el 14 de mayo de 2022).

[2] "Minikube: lo mejor de Kubernetes". IONOS Digital guide. https://www.ionos.es/digitalguide/servidores/herramientas/kubernetes-minikube/. (accedido el 12 de mayo de 2022).

[3] "Google Cloud Vision API". Davinci Group. https://www.davincigroup.es/google-cloud-vision-api/ (accedido el 14 de mayo de 2022).