SOA: Arquitectura orientada a servicios

Estudiantes
Sebastián Josue Alba Vives
Romario Ramírez Ramírez
Michael Shakime Richard Sparks
Steven Badilla Soto

Proyecto 1: SENTIMENT ANALYSIS INFRASTRUCTURE

II Semestre, 2022

Sobre SENTIMENT ANALYSIS INFRASTRUCTURE
Sistema de software que permita analizar los sentimientos de los empleados de la empresa al recibir buenas noticias. La aplicación toma la imagen desde el cloud bucket, y consulta el GCP VISION API, mediante el cual la imagen (foto de una persona) será analizada y la emoción que el empleado refleja será analizada por el API. Esta emoción se imprimirá como resultado de la llamada a la función o contenedor y agregarla a una base de datos.

Componentes y sus responsabilidades

- Jenkins: Es una tecnología utilizada en el pipeline el cual interactúa con git, terraform y con el código de python. Es una herramienta que se utiliza para compilar y probar este proyecto, además permite acelerar el proceso de desarrollo y de entrega.

- Python: Mediante Python se implementará la funcionalidad del sistema descrito, sin contar con un frontend en la aplicación. El sistema acepta una foto, y es inicializado por un trigger, posteriormente consulta el Rest API de Vision AI, para después devolver la respuesta del estado emocional de la foto que se seleccionó.

- Base de datos: Almacena los resultados de todas las imágenes que se han analizado. Y es implementada en google cloud SQL

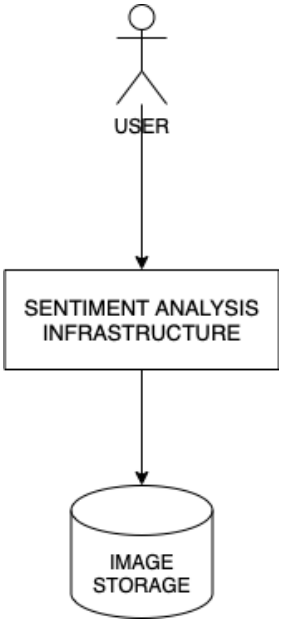
- Serverless: Es quien ejecuta el código

- GCP VISION API: Es el API encargado de tomar la imagen (foto de una persona) y analizar la emoción que el empleado refleja. Esta API ofrece potentes modelos de aprendizaje automático pre-entrenados a través de las API REST y RPC mediante etiquetas de imágenes y clasificación en categorías predefinidas detectando objetos y caras.

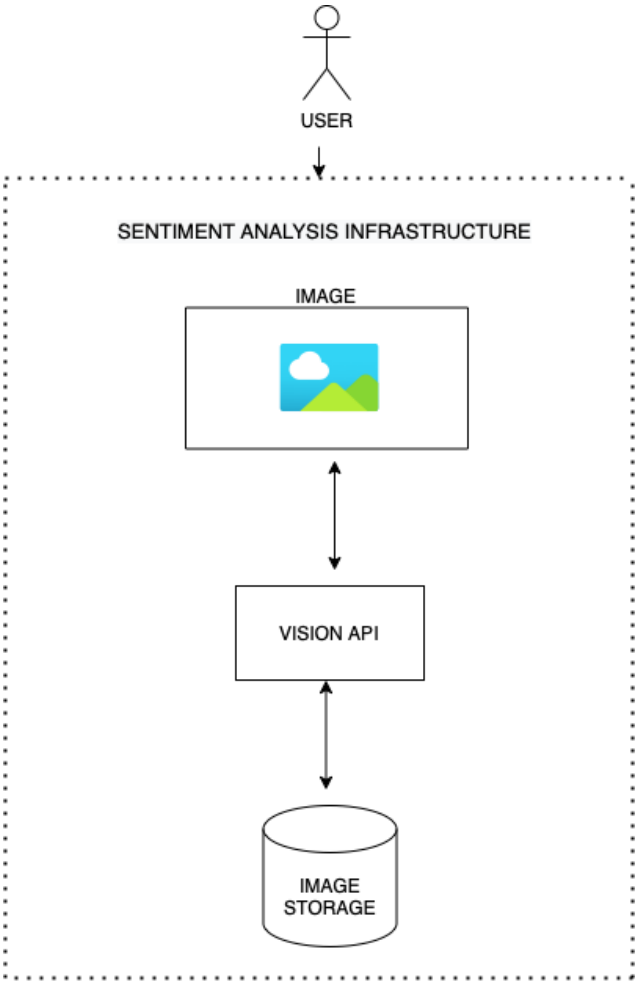
- Cloud Bucket: Utilizado para almacenar la imagen a ser procesada. Representa un depósito en Cloud Storage con objetos a los que se puede acceder mediante sus propios métodos.

Diagrama de la arquitectura de la infraestructura

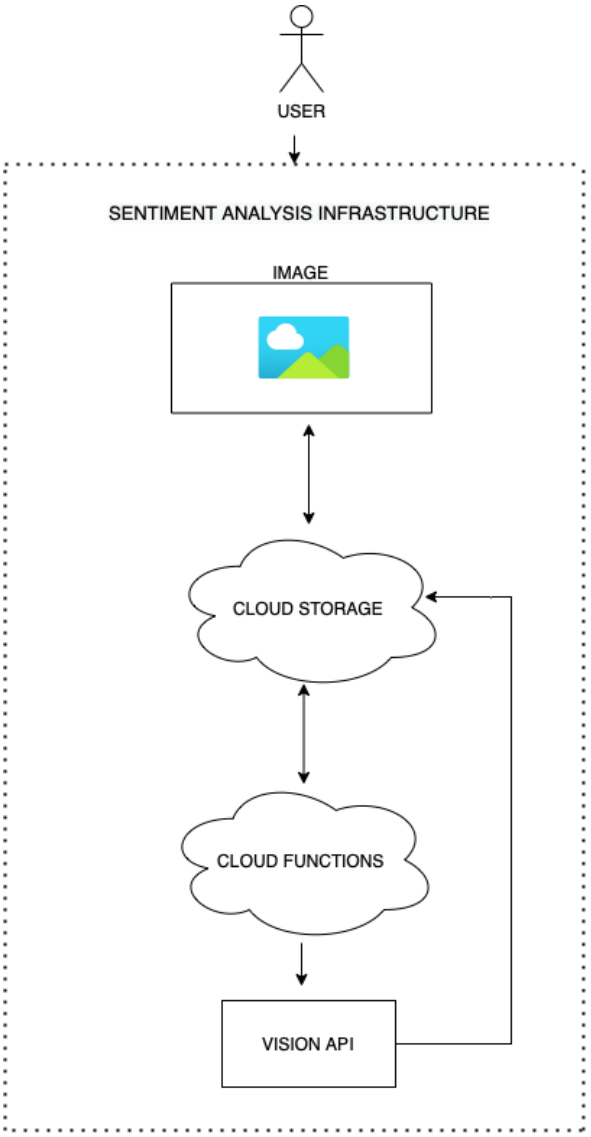
- **Primer nivel**



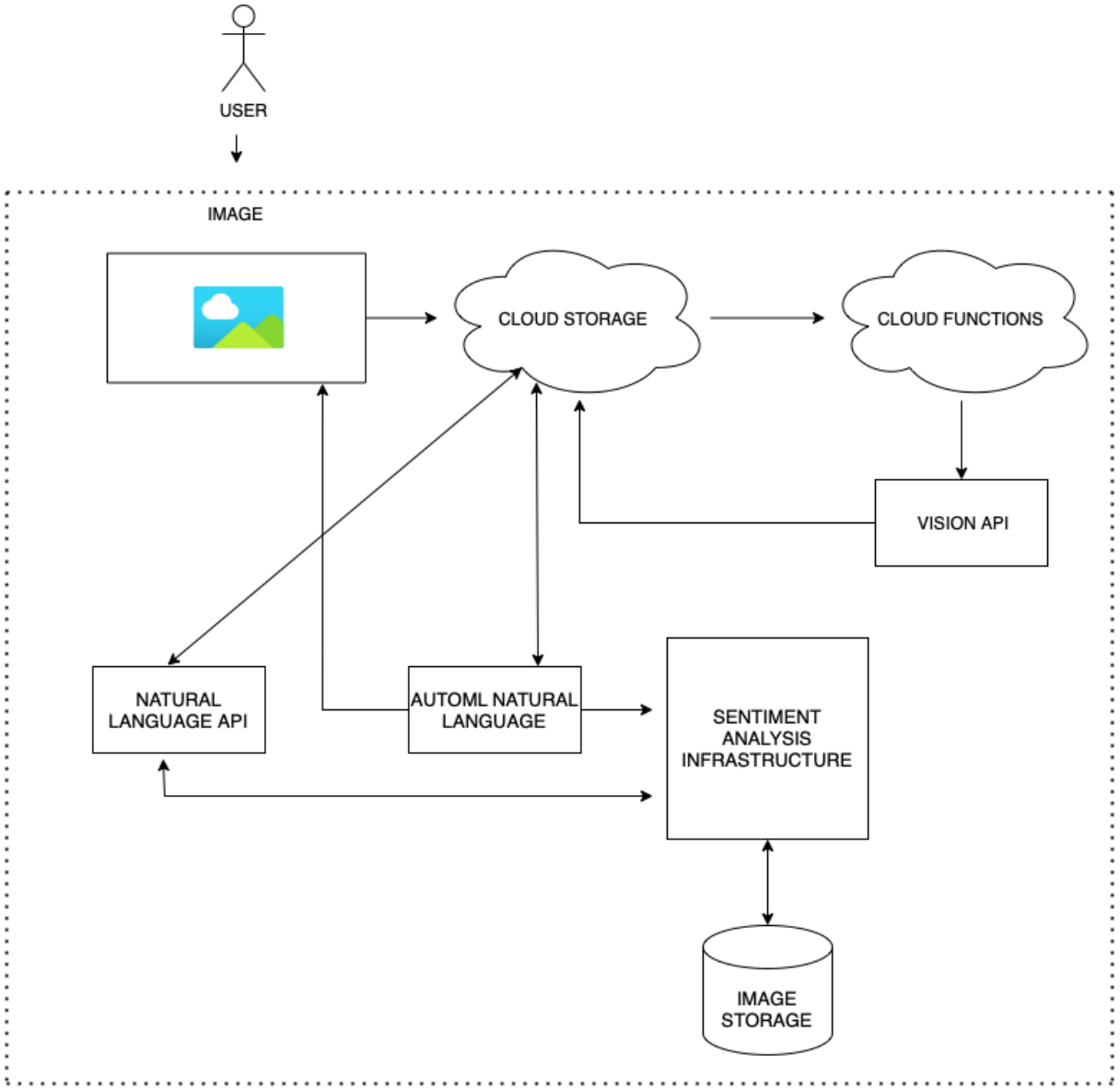
- **Segundo Nivel**



- **Tercer Nivel**



- **Cuarto Nivel**



Documentación de decisiones

- **Organización del código**

Se diseñó primeramente la arquitectura, y se tomó como prioridad y punto de partida la organización, toma de decisiones, estándares y prácticas para llevar a cabo el proyecto, es decir, primero se hizo un planeamiento y análisis y posteriormente se hizo la implementación. Además que esto fomenta la transparencia del equipo, lo cuál es uno de los pilares de las metodologías ágiles.

Se decidió colocar todo el código del API en un mismo repositorio. Para simplificar lo que viene siendo el control de versiones y para mantener todo el proyecto en un mismo repositorio simplificando el flujo de integración y mantenimiento continuo.

Se decidió hacer 2 sesiones por semana para lograr tener un mejor entendimiento y organización del equipo, así para poder ver el avance con cada entregable

- **Estándar y prácticas de programación**

Primeramente se hizo una serie de estándares como equipo, donde se plasmó el conocimiento adquirido en cursos anteriores, dichos estándares y prácticas fueron los siguientes:

- Indentar 3 espacios para cada bloque de código.
- Cuando se declaran objetos espaciar con una línea por cada atributo.
- Las variables que se requiera declarar se deben hacer localmente.
- Los números, booleanos y strings no se usan ni se declaran como objetos.
- Las variables solo son inicializadas cuando son declaradas.
- Todo nombre de función y variable inicia con una letra.
- Se dejan espacios entre cada operador y las comas.
- No utilizar Nulo cuando una variable no tiene valor, se utiliza **undefined**.
- Si se tiene un archivo y este se trata de un componente de interfaz gráfica, colocar ".component" dentro de su nombre.
- El los bucles y declaraciones condicionales siempre utilizar "{}" y el
- El "{" de un bucle o declaración condicional, siempre va en la misma línea.
- El "}" de un bucle o declaración condicional, siempre va en una nueva línea.

Posteriormente se hizo una consulta e investigación sobre buenas prácticas de programación para unificar el entendimiento del equipo. Perez (2019) menciona una serie de ellas

- Definir variables con nombres que tengan algún sentido o fácilmente identificables.
- En los print, las líneas de código no deben ser muy largas, como mucho 72 caracteres. Si se tiene una línea larga, se puede cortar con una barra invertida (\) y continuar en la siguiente línea
- Dentro de paréntesis, corchetes o llaves, no dejar espacios inmediatamente dentro de ellos
- Justo después de coma, punto y coma y punto, separar con un espacio, para mayor claridad, pero no antes
- Aunque en Python se pueden hacer varias declaraciones en una línea, se recomienda hacer sólo una en cada línea

Se utilizó el gitflow como branching strategy, ya que permite tener un mejor control de las funciones que se deben ir haciendo, realizar un proceso de revisión antes de escalar a las ramas principales o de desarrollo del proyecto. Se creó 1 feature “vision-api” con el objetivo de poder integrar el código realizado en python con la funcionalidad y el API.

Por otra parte se implementó el branch “test” con el objetivo de poder realizar un test para una parte del código.

Ambos branches se lograron revisar de manera tal que se diera como terminada la funcionalidad y ser incorporado a las ramas principales.

La rama main o principal almacena el historial de publicación oficial y la rama develop o de desarrollo sirve como rama de integración para las funciones, en este caso el feature que integra la funcionalidad realizada en Python y el API.

Referencias

1. Perez Prieto, J. (2019). Estilo de codificación y buenas prácticas — documentación de Curso de Python para Astronomia - 20191128. Retrieved 16 September 2022, from http://research.iac.es/sieinvens/python-course/estilo.html
2. Vision AI | Derive Image Insights via ML | Cloud Vision API | Google Cloud. (2022). Retrieved 16 September 2022, from https://cloud.google.com/vision
3. Jenkins User Documentation. (2022). Retrieved 16 September 2022, from https://www.jenkins.io/doc/
4. Knüppel, M. *The Scrum Guide Explained*.