TEC | Tecnológico de Costa Rica

SOA: Arquitectura orientada a servicios

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

Proyecto 2

II Semestre, 2022

Como todo proyecto, la arquitectura bien diagramada y explicada permite un mayor entendimiento y conocimiento de la solución propuesta y del entregable para todos los stakeholders.

Para el proyecto 2 se plantea una arquitectura utilizando los principios de SOLID:

- En este proyecto se trató de reunir las funcionalidades y componentes que cambian por las mismas razones, además separando aquellas que cambian por razones diferentes, se procuró tener una separación de funcionalidades.
- Por otro lado, las clases utilizadas e implementadas están abiertas para poder extenderse lo suficiente y cerradas para modificarse, por ejemplo, las funciones que están en los servicios, se encuentran en el clúster también y tienen la apertura para extender sus componentes sin modificarse.
- Además, en la utilización de ciertas clases es posible usar sus subclases sin que haya algún tipo de problema o interferencia en su funcionamiento.
- Adicionalmente, para nosotros fue de preferencia tener muchas interfaces que definan pocos métodos, a comparación de tener una interface forzada a implementar muchos métodos a los que no dará uso, por ejemplo, hay una separación de interfaces de resultados y de interfaz de usuario en lugar de tener una sola interfaz con muchos métodos.
- Finalmente se propuso reducir las dependencias entre los módulos del código, es decir, alcanzar un bajo acoplamiento de las clases, un ejemplo de ello es que en algunos momentos requerimos cambiar ciertas funcionalidades y componentes y gracias al bajo acoplamiento lo pudimos hacer sin problema.

Para la propuesta de este sistema, hay contenedores corriendo, donde cada uno tiene un servicio. Dicho sistema se encuentra como pods individuales, los cuales son el servicio de la Interfaz, de la base de datos, del API o backend, del broker y del analizador de datos. Dichos servicios están corriendo en Kubernates.

Se decidió implementar el intermediario o Broker, debido a que era posible encontrar una mayor documentación e implementación que ejemplificara cómo utilizar esto en el proyecto y además porque gracias al broker hay mayor independencia en cuanto a que los servicios que implementamos pueden decidir el flujo de los eventos.

Arquitectura Prescriptiva

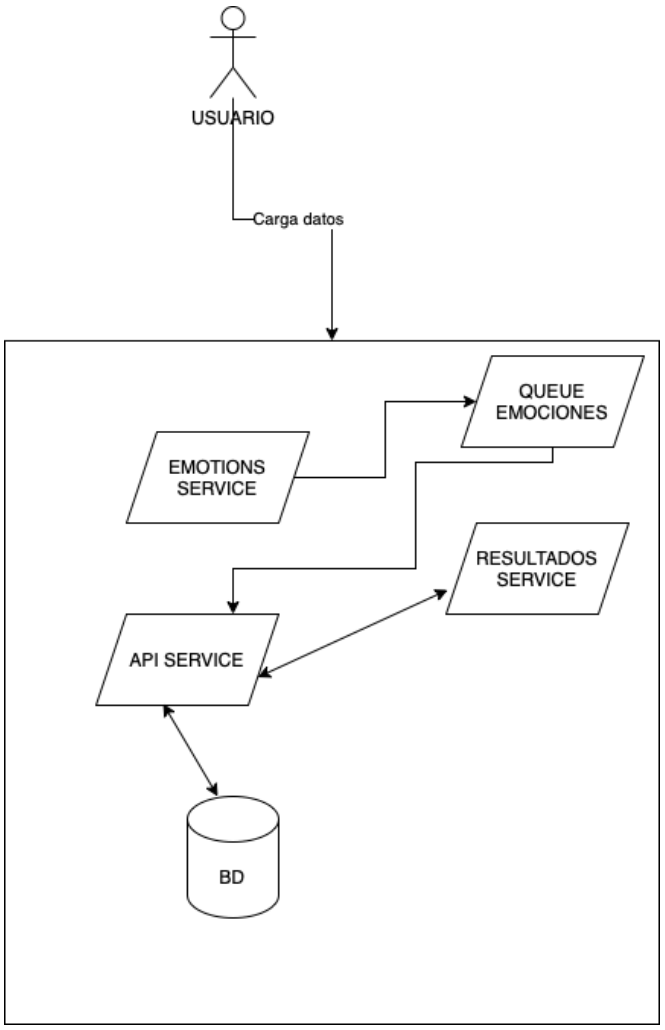
Inicialmente se tenían los mismos diagramas de la arquitectura descriptiva, pero con las variantes de que no se tenía planeado utilizar o implementar el servicio de api de backend. Adicionalmente, teníamos planteado que un evento guardaba algo en la base de datos directamente.

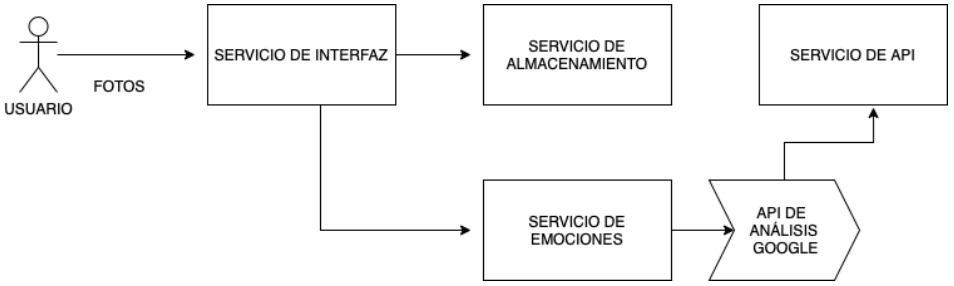
Manejo de eventos

- **Secuencia de funcionalidad**

1. El usuario carga los datos al bucket.
2. El servicio de emotions está en una rutina que cada cierto tiempo manda un mensaje a la cola de emociones.
3. A la escucha de la cola de emociones va a estar el api o backend que en el momento que recibe, hace insert en la base de datos.
4. De forma asíncrona va a estar el cliente en el servicio de api, pero ese servicio es asíncrono, entonces cuando se recargue la página ese request va a ir al backend, y el backend llama a la base de datos y envía los resultados.

Al haber problemas de sincronización se llegó a esa solución de llamar directamente a la base de datos. El problema surge producto a la interfaz, pues esta no se puede quedar esperando a que le conteste el broker infinitamente, entonces es necesario que se conecte con un api usual que le devuelva una determinada respuesta.





Arquitectura Descriptiva

Primer Nivel

Segundo Nivel

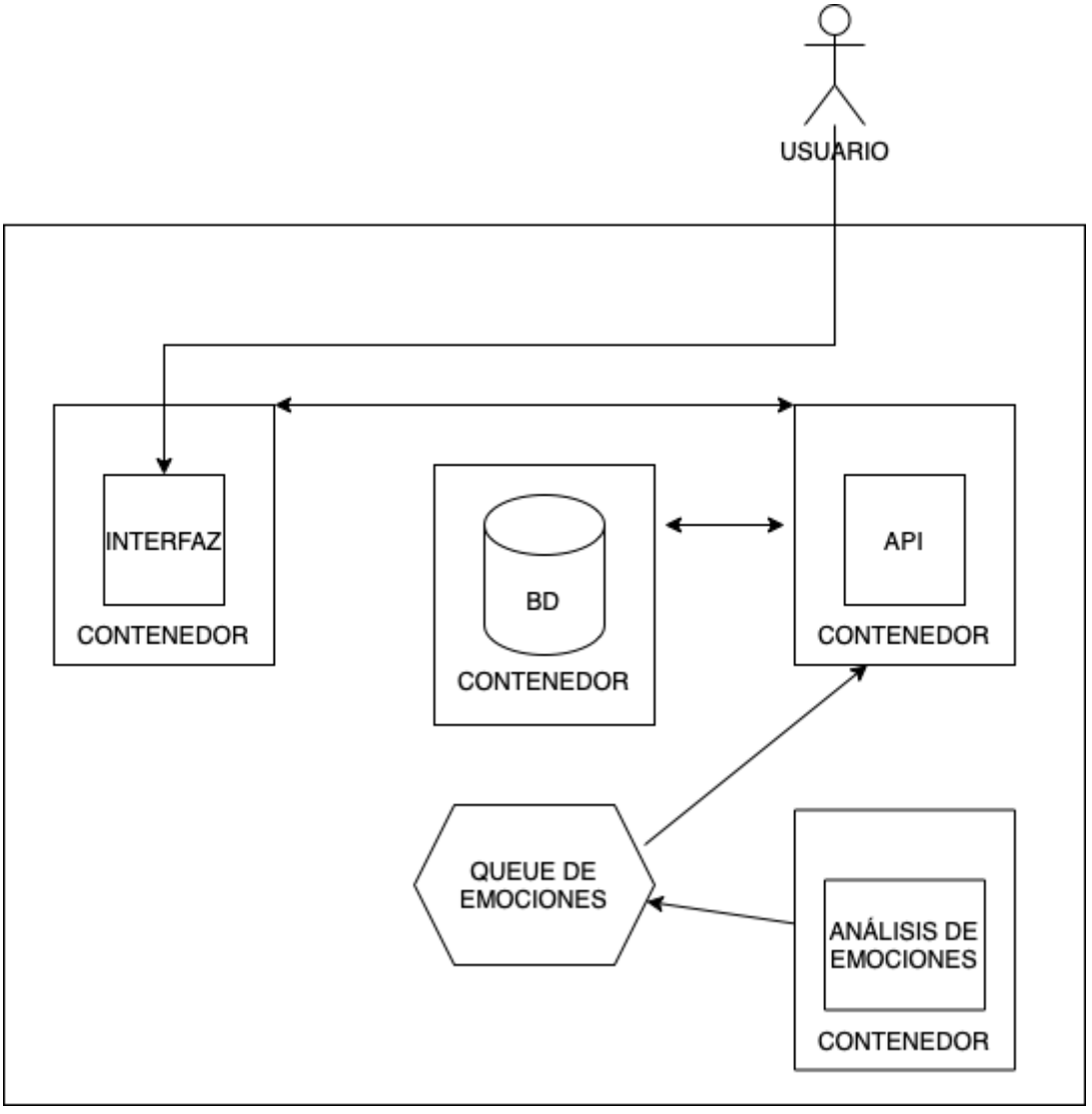
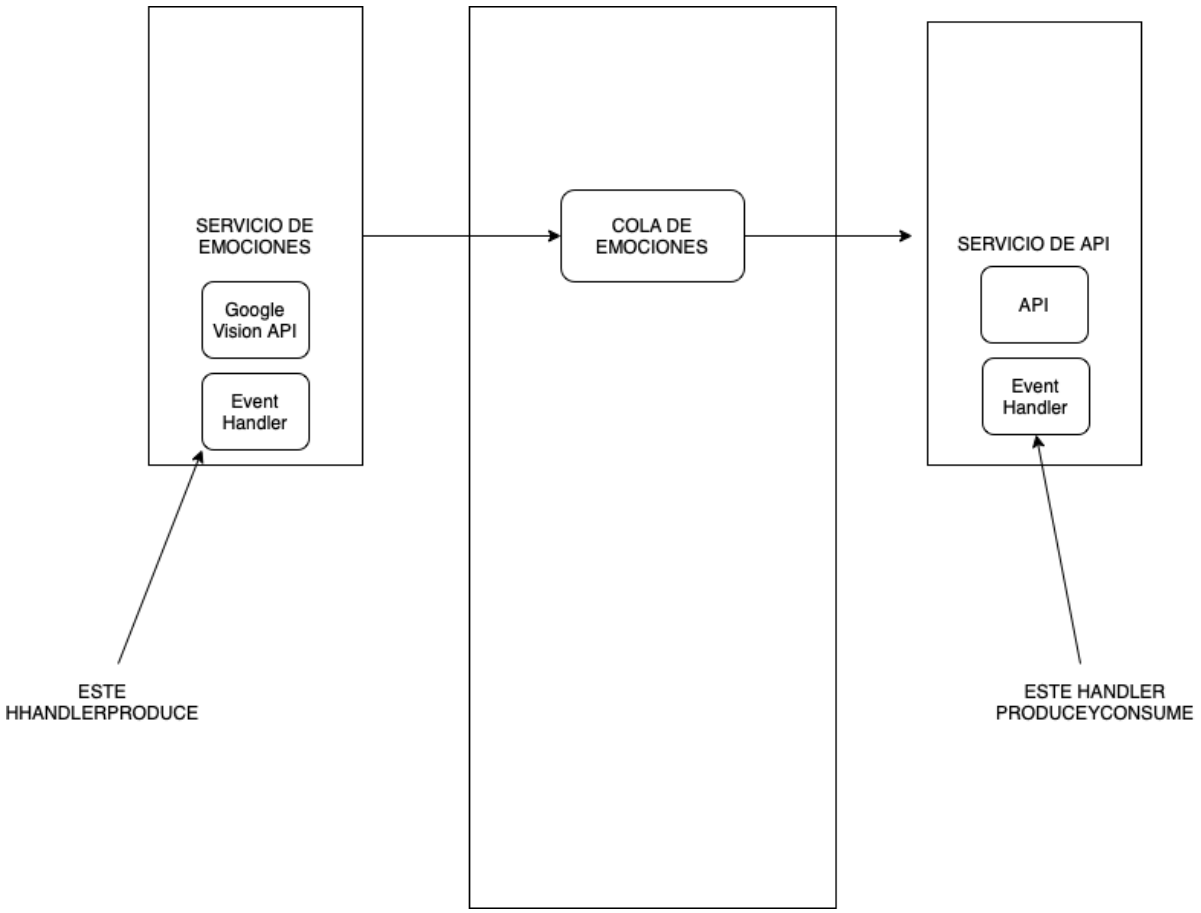


Diagrama de Eventos



Descripción de servicios

Interfaz: Es la interfaz con la que interactúa el usuario seleccionando las imágenes requeridas. Dicha interfaz está implementada en Angular con html, css y typescript.
Base de datos: Es el servicio encargado del almacenamiento de los datos productos de los análisis de las imágenes. Esta base de datos está hecha en Mysql.
API o backend: Este servicio es quien envía a la base de datos los resultados y la información desde el servicio de analizador de emociones al servicio de la base de datos. Este backend está implementado en Python.
Broker: Este servicio logra intermediar la información, resultados y datos mediante métodos asincrónicos, este servicio está en rabbitmq.
Emociones: Este servicio analiza las imágenes que son introducidas y determina el tipo de emoción respectiva. Este servicio está implementado en Python.

Despliegue de servicios
Consiste un archivo con instrucciones para la creación de los componentes de kubernetes mediante una imagen