Emoji position

Integrantes:

- Rodrigo Almonacid
- Hernan Gallardo
- Sebastian Sanhueza

Resumen:

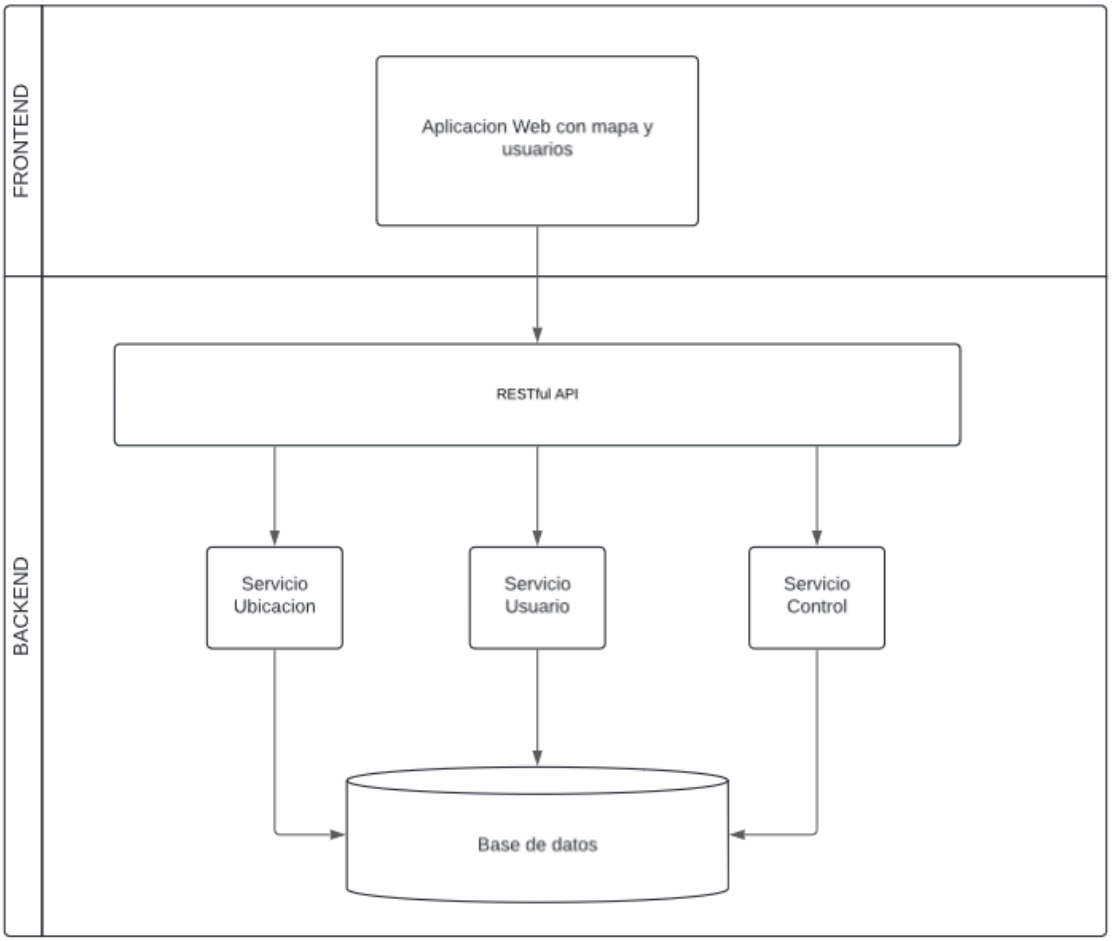
Idea: Crear una página web que muestre un mapa con emojis y descripciones de los usuarios, donde se requiere compartir la ubicación para acceder al contenido.

Requerimientos Front-end:
-Verificación de acceso a la ubicación.
-Despliegue del mapa.
-Permitir a los usuarios cambiar su emoji y descripción.
-Habilitar la interacción para que al hacer clic en un emoji se muestre la descripción correspondiente.

Requerimientos Back-end:
-Definir entidades para representar usuarios, ubicaciones, emojis y descripciones.
-Crear una base de datos normalizada, escalable y eficiente para almacenar la información.
-Implementar la gestión de ubicaciones asociadas a un ID de usuario.
-Desarrollar mecanismos de control de privacidad para proteger la información de los usuarios.
-Construir una API RESTful que permita crear, leer, actualizar y eliminar datos del sistema.

Arquitectura

Se pretende formular una arquitectura orientada a servicios. Donde cada componente se asume se encuentra dentro de un contenedor, a excepción de la base de datos.



- Se contempla que cada servicio pueda tener su propia base de datos.

Componentes

ESB/RESTful API: Este componente se piensa como un middleware ligero sin grandes conversiones o transformaciones de datos.

Servicio de Gestión de Ubicaciones: Este servicio estará en un contenedor Docker dedicado. El contenedor incluirá el código y las dependencias necesarias para recibir, almacenar y procesar las ubicaciones de los usuarios.

Servicio de Gestión de Usuarios: Este servicio se encargará en guardar y modificar datos asociados al id de un usuario como su descripción, correra en un contenedor Docker

separado. El contenedor contendrá el código y las dependencias necesarias para manejar las funciones CRUD a la capa de usuario.

Servicio de Página Web (Front-end): El front-end del sistema estará alojado en un contenedor Docker. Este contenedor contendrá los archivos estáticos de la interfaz de usuario (HTML, CSS, JavaScript) y la lógica necesaria para interactuar con los demás servicios a través de API REST.

Servicio de control: Se encargará de entregar las ubicaciones de los usuarios dependiendo la ubicación de la solicitud, según la ubicación desde donde se hace la ubicación, será el la tabla de ubicaciones que se enviará a la front end

Modelo Fundamental

1. **Modelo de Interacción**: Se utilizaran protocolos estándar de comunicación como HTTP/RESTful API para permitir la interacción entre la aplicación web y los servicios backend. En gran parte de la comunicación se intentará usar un modelo de comunicación asíncrono exceptuando las situaciones donde la complejidad de desarrollo tome demasiado tiempo.

Esta RESTful API funcionará como **ESB**, o bus de servicio empresarial, dentro de la arquitectura orientada a servicios. Permitiendo la integración y comunicación entre servicios distribuidos de manera eficiente y escalable.

2. **Modelo de Seguridad:** Se utilizarán técnicas como tokens JWT para gestionar la autenticación y autorización de los usuarios de manera segura y sin estado. En búsqueda de un sistema robusto de autorización, para proteger los datos sensibles y restringir el sistema. Además el cifrado SSL/TSL garantizara la confidencialidad de las comunicaciones entre el frontend y backend. **** Definir aspectos claves del sistema**

3. Modelo de Fallas y Tolerancia a Fallos: Se diseñará cada servicio con la vista en principios de resiliencia, esperando que al momento de fallar, tenga la capacidad de recuperarse automáticamente. Y mantener operativo el sistema de geolocalización, independiente de la escala.

También para esto esperamos nos apoyaremos en el escalamiento vertical (incrementando los recursos de hardware) para gestionar la carga de recursos. (Utilizar servicios de aws tal vez)

Modelo Físico:

Modelo cliente-servidor: La interacción entre los usuarios y el sistema se estructura bajo el paradigma cliente-servidor, donde los clientes solicitan servicios y los servidores los proporcionan. Esto facilita la comprensión y el mantenimiento del sistema.

Distribución de componentes en la nube: Los componentes del sistema se alojan en una infraestructura de nube, aprovechando las ventajas de escalabilidad, flexibilidad, alta disponibilidad y reducción de costos que ofrece este entorno.

Distribución en servicios: Los componentes del sistema se organizan en servicios autónomos e interoperables, promoviendo la modularidad, el reuso de código y la facilidad de implementación.

Modelo de datos centralizado: Los datos críticos del sistema se almacenan en un único lugar, simplificando la gestión, la consistencia y la seguridad de la información**.**

Tecnologías:

Lenguaje de programación: se considera usar JavaScript para el front end con React y CSS vanilla. Para el backend se usará Python y se tiene pensado Flask para facilitar el desarrollo.

Base de datos: se considera usar un paradigma de base de datos relacionales, en tablas, para poder modularizar las ubicaciones asociadas a un id, ese mismo id asociarlo a información como descripción, etc. además que una base de datos como esta bien normalizada puede ser fácil de escalar a nuevas funcionalidades, es por esto que se elige PostgreSQL

Contenedores: para el sistema de contenedores se plantea usar Docker por su robustez y confiabilidad

Diagrama de FLujo: