Emoji position

Integrantes:

- Rodrigo Almonacid
- Hernan Gallardo
- Sebastian Sanhueza

Introducción:

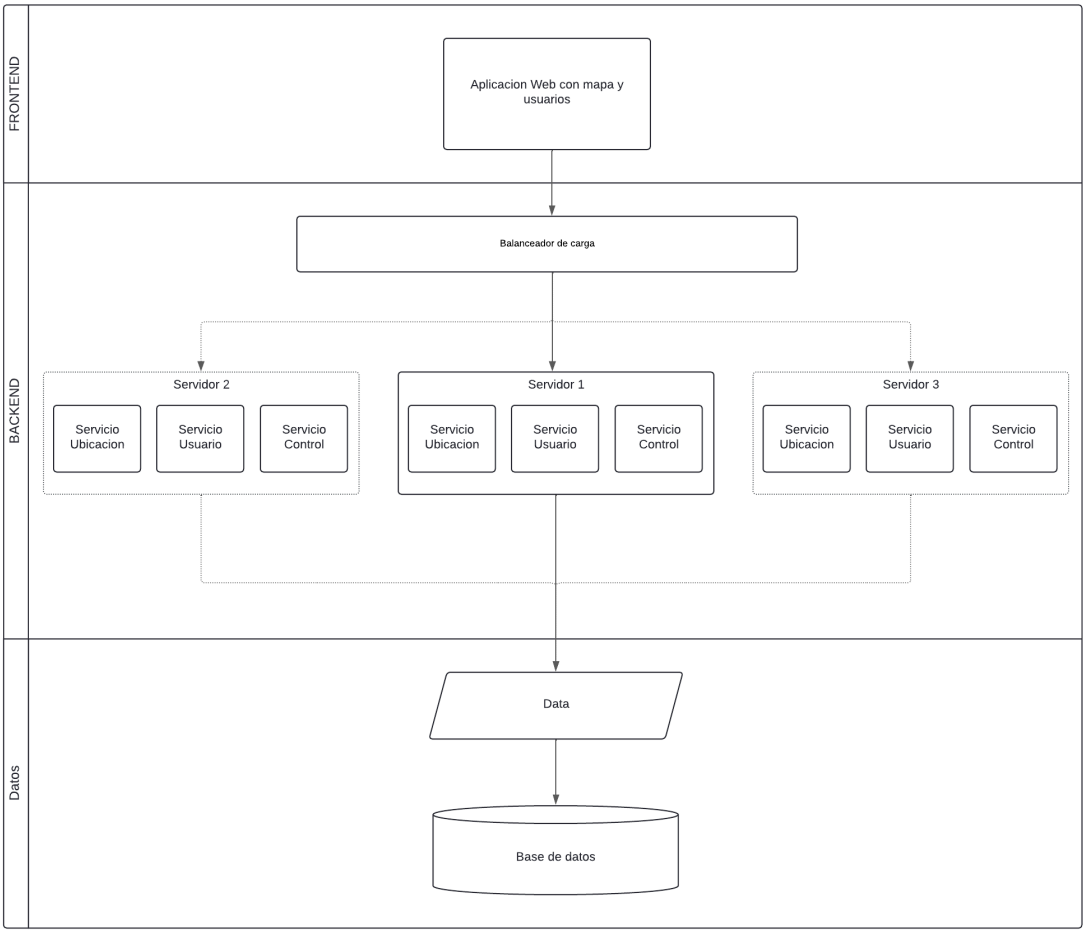
En la actualidad, la tecnología y las aplicaciones móviles han transformado de manera significativa la forma en la que las personas se comunican. Con el aumento a eventos y reuniones masivas, como conciertos, festivales y actividades de otro tipo, la necesidad de herramientas que faciliten la localización y reunión de personas se ha vuelto más evidente. Las aplicaciones de geolocalización han demostrado ser útiles en estos aspectos y permiten mediante una herramienta sencilla, ofrecer soluciones versátiles.

Uno de los desafíos más comunes en eventos multitudinarios es la dificultad para encontrar y reunirse junto a los demás asistentes. Muchas veces la información de los eventos no es precisa o de plano la ubicación corresponde a un recinto muy amplio. Además, es difícil dar indicaciones en espacios de este tipo. Sumado a que las aplicaciones de mapas estándar no están diseñadas para la movilidad en, por ejemplo, parques, nace una necesidad de permitir a los asistentes de eventos masivos en espacios amplios, identificar cual es verdaderamente el punto al que se deben acercar.

La aplicación web propuesta se llama Emoji Position. Una aplicación web que mediante el uso de emojis y geolocalización permitirá a los usuarios seleccionar un emoji y mediante una interfaz que presenta el mapa, entregarle su ubicación y los emojis de usuarios alrededor. Esto facilita que los participantes identifiquen de manera rápida y sin ambigüedades donde se encuentran los demás asistentes, siendo además el emoji utilizado una variable personalizada para el lugar que se busca.

Arquitectura

Se utilizará una arquitectura orientada a servicios, en donde cada componente se asume se encuentra dentro de un contenedor, a excepción de la base de datos.



Componentes:

- Página Web (Frontend): El frontend del sistema estará alojado en un contenedor Docker. Este contenedor contendrá los archivos estáticos de la interfaz de usuario y la lógica necesaria para interactuar con los demás componentes.

- Balanceador de carga: Este componente corresponde a una herramienta que se encargará de distribuir las solicitudes de los usuarios a alguno de los servidores activos. Se utilizará Nginx, ya que su implementación no es compleja y entrega herramientas básicas, pero suficientes para funcionar como un balanceador de carga.

- Servidor: Se encargará de manejar las peticiones respecto a la posición de los usuarios mediante un solo servicio. Esto corresponde a guardar, eliminar y obtener posiciones.

- Base de datos: Se utilizará MongoDB Atlas para almacenar los datos, ya que solo necesitamos guardar múltiples objetos con datos estructurados de manera predefinida. Para nuestro caso, preferimos el uso de MongoDB Atlas por sobre MongoDB para evitar configuraciones manuales.

Los componentes están en contenedores o corresponden a herramientas externas que no requieren configuración, por lo que no se ahondará en el detalle de su instalación.

Objetos nosql:

Usuario	{
	"userId": 1,
	"latitude": 0,
	"longitude": 0,
	"timestamp": "2022-01-01T00:00:00Z",
	"emojiId": "emojiId1"
	}

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 ubicaciones y emojis.
- 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.

Modelo Fundamental

1. **Modelo de Interacción**:

Se utilizará un modelo de interacciones basado en mensajes.

En primera instancia, se tiene la aplicación web a la que acceden los usuarios. Al ingresar, se solicita al usuario el acceso a su ubicación y este además debe elegir un emoji para representarse. Cuando ambos datos son recopilados por el frontend, se envía un mensaje mediante Nginx, que funciona como un proxy inverso y balanceador de carga. Un proxy inverso es un servidor que se sitúa delante de los servidores y reenvía las solicitudes del cliente. El balanceador de carga corresponde a una entidad que, también debe estar frente a los servidores que responden a una aplicación y se encarga de distribuir las solicitudes de alguna manera, ya sea con ponderación o métodos de distribución específicos como Round Robin, IP-Hash, Least Connections, entre otros. Una vez que la consulta es designada a un servidor, éste se encarga de procesarla y conectarse con la base de datos para ejecutar la orden de la consulta, que, en este caso, puede ser para escribir, leer o reemplazar información, usando los métodos de HTTP GET y POST.

Modelo Físico:

Inicialmente, se utilizará un servidor en Linode que permita el uso básico de la aplicación. Actualmente, el servidor de menor costo ofrece los siguientes servicios:

- 1 GB de Ram
- 1 Nucleo CPU
- 25 GB SSD
- 1 TB de Transferencia

El despliegue de la aplicación se ejecutará en el puerto 80.
Los puertos utilizados como servidores son los puertos 5000 y 5001.

Tecnologías:

Se utilizarán las siguientes tecnologías y herramientas:

- JavaScript: Lenguaje esencial para crear páginas web dinámicas e interactivas.
- React 18.*: Biblioteca para construir interfaces de usuario eficientes y reutilizables.
- CSS Vanilla: CSS puro para estilizar páginas web con control preciso.
- Python 3.10: Es versátil y fácil de aprender para desarrollo rápido.
- Flask 3.*: Microframework ligero para desarrollar aplicaciones web y APIs.
- pymongo 4.7: Librería para manejar MongoDB desde Python de manera eficiente.
- Docker 23.*: Plataforma para empaquetar aplicaciones en contenedores, garantizando consistencia.
- Docker Compose 2.*: Herramienta para orquestar aplicaciones multi-contenedor fácilmente.
- Nginx 1.26: Servidor web y proxy inverso eficiente para mejorar rendimiento y seguridad.

Listado de preguntas:

1. ¿Cuál es el propósito de la aplicación?
   R: La aplicación está destinada a ayudar a los usuarios a encontrar y reunirse con otros asistentes a eventos masivos mediante el uso de emojis y geolocalización.

2. ¿Cual es el público objetivo de la aplicación?
   R: El público objetivo son jóvenes y adultos que asisten a eventos masivos como conciertos, festivales culturales, conferencias, etc.

3. ¿Qué nivel de tráfico se espera en la aplicación? ¿Cuántos usuarios concurrentes o solicitudes por segundo?
   R: Inicialmente nos gustaría lanzar la aplicación para un público reducido. Eventos de 30 a 50 personas. Si nos va bien, promocionarla para eventos de 500 o 1000 personas, incluso más.

4. ¿Cual es el entorno de despliegue preferido? ¿Tienen hardware para poder hostear este servicio?
   R: No tenemos computadoras para la aplicación, pero pensamos que es buena idea pagar por un servicio en la nube.

5. ¿Qué características de seguridad son prioridad para la aplicación?
   R: Nos interesa que el servicio pueda estar disponible y los datos mostrados en el mapa sean fidedignos. Como no existen cuentas de usuario, nos basta con que la aplicación no sufra algún ataque que pueda provocar la caída de la aplicación o corromper la información mostrada.

6. ¿Qué niveles de rendimiento espera del sistema y cómo planea medirlo?
   R: Nos gustaría que fuera constantemente en tiempo real, pero existe holgura respecto a eso. Mientras actualice la información a un ritmo constante, nos parece bien que no sobrepase los 20 segundos. Inicialmente no nos interesa medir otras aristas.

7. ¿Qué idiomas y localizaciones debe soportar la aplicación?
   R: Nos gustaría que cubra tanto zonas urbanas como no urbanizadas de toda la región. Luego podemos evaluar el uso a nivel nacional. Por lo mismo, basta con que el idioma sea sólo español.

8. ¿Qué dispositivos y navegadores deben ser compatibles con la aplicación?
   R: Debe ser compatible con cualquier dispositivo móvil y sus navegadores disponibles.

9. ¿Cómo le gustaría que se gestionen las fallas de nodos en el entorno distribuido?
   R: Nos basta con que si la aplicación se cae, por cualquier razón, pueda levantarse rápidamente otra vez.

10. ¿Se requiere interacción con otras aplicaciones o servicios?
    R: Nos gustaría, más adelante, permitirnos conectarse con redes sociales y dar un aviso de “Me encuentro usando Emoji Position en Lollapalooza” para llegar a más usuarios, pero no es algo estrictamente necesario.

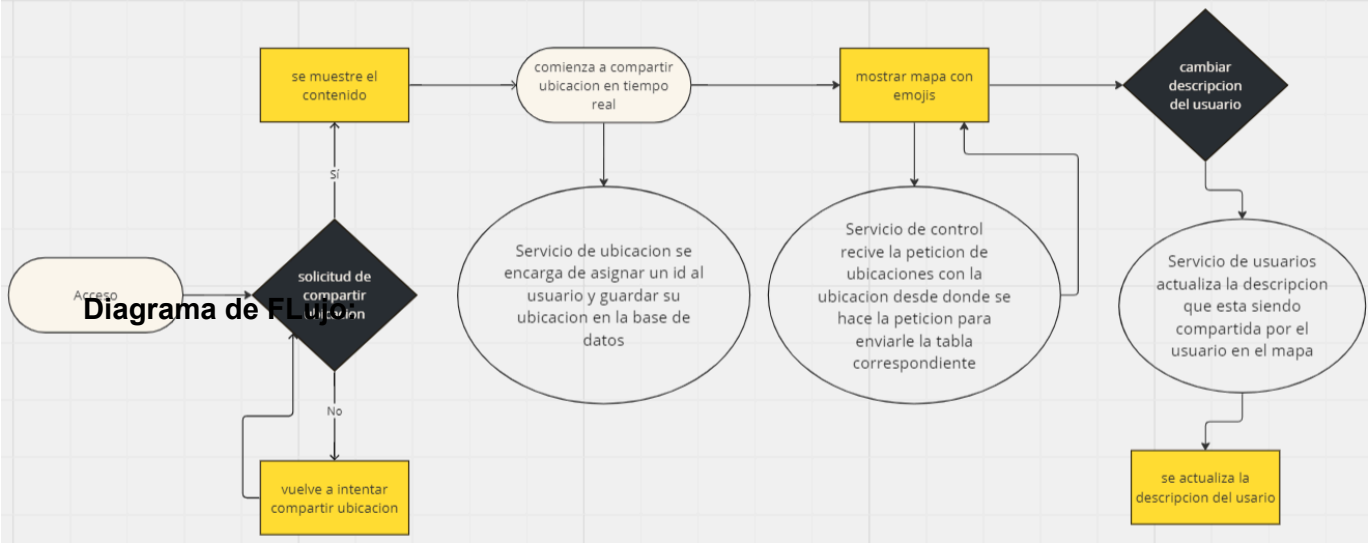


Diagrama de FLujo