MicroTracker - Iteración 1

Diseño de Sistema Distribuido

INFO288 - Sistemas distribuidos

Instituto de Informática, Universidad Austral de Chile.

Benjamin Cifuentes

Luis Llanca

Alejandro Villagrán

Introducción

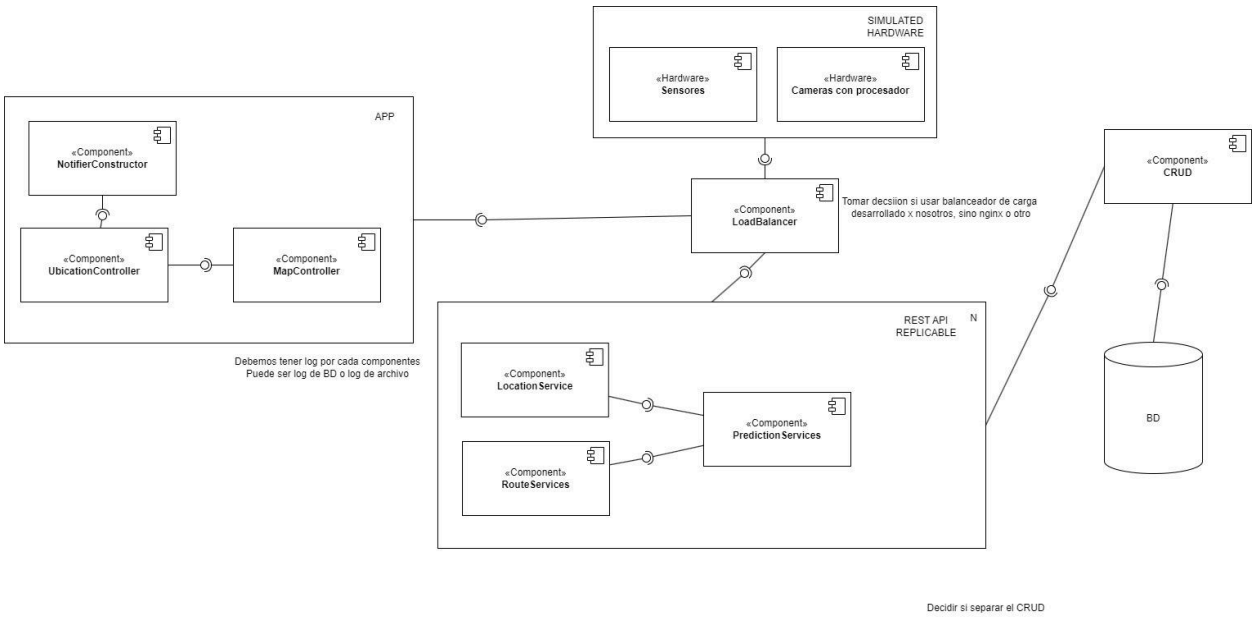
En Valdivia, tanto trabajadores como estudiantes enfrentan dificultades debido a la falta de información precisa sobre los horarios de los microbuses, lo que genera imprevisibilidad en los tiempos de espera y congestión en los vehículos. Esta situación ocasiona una planificación ineficiente de los viajes, con tiempos de espera prolongados y experiencias de transporte incómodas y poco fiables para los usuarios.

Propuesta de solución

Para abordar este desafío, se propone la implementación de una plataforma que integre un sistema distribuido y una aplicación móvil. Esta aplicación proporcionará un mapa interactivo en tiempo real que mostrará la ubicación exacta de los microbuses en circulación, así como predicciones precisas de los tiempos de llegada a destinos específicos. Esto permitirá a los usuarios planificar sus viajes de manera más eficiente, reducir los tiempos de espera y mejorar significativamente su experiencia con el transporte público en Valdivia.

Arquitectura

Diagrama de componentes



Enlace para visualizarlo de mejor manera:
https://app.diagrams.net/#G1mpuNoFtPfiCkKUiOg_D6wNeWUbutCv0V#%7B"pageId"%3A"x-HpuxCrkzUajJuQxPSA"%7D

Lista de componentes

Sistema:

Load Balancer:
El componente Load Balancer se encarga de distribuir equitativamente la carga de trabajo entre las diversas APIs del sistema, garantizando así un rendimiento óptimo y una alta disponibilidad.
Location Service:
El componente Location Service administra los datos de ubicación de los microbuses, proporcionando información sobre sus ubicaciones actuales y procesando el historial de posiciones cuando sea necesario. Este servicio juega un papel crucial en la visualización en tiempo real de la ubicación de los microbuses en el mapa y en la generación de predicciones de tiempos de llegada.
Route Service:
El componente Route Service se encarga de gestionar las rutas de cada línea de microbuses y de proporcionar acceso a esta información. Esto incluye la actualización de las rutas en tiempo real y la consulta de datos relacionados con las paradas y los recorridos de los microbuses.
Prediction Service:
El componente Prediction Service realiza predicciones de tiempos de llegada de los microbuses utilizando la información de ubicación de los vehículos, así como la ubicación actual del usuario o el destino deseado. Estas predicciones son fundamentales para proporcionar una estimación precisa de los tiempos de espera a los usuarios.
CRUD:
El componente CRUD maneja las operaciones de creación, lectura, actualización y eliminación en la base de datos del sistema. Es responsable de procesar las solicitudes que requieren acceso a la base de datos, garantizando la integridad y consistencia de los datos almacenados.
Base de Datos (BD):
La Base de Datos es el componente central que administra todos los datos procesados en el sistema. Almacena información crucial como ubicaciones de microbuses, historial de posiciones, rutas, datos de usuarios y registros de acciones realizadas en la aplicación.

Aplicación Móvil:

Notifier Constructor:
El Notifier Constructor es responsable de generar y enviar notificaciones a los usuarios, proporcionando información relevante como predicciones de tiempos de llegada de los microbuses. Este componente garantiza que los usuarios estén informados de manera oportuna sobre los detalles importantes relacionados con sus viajes.
Ubication Controller:
El Ubication Controller gestiona la ubicación del usuario en la aplicación móvil, permitiendo su seguimiento en el mapa y proporcionando esta información para su uso en las predicciones de tiempos de llegada. Este componente asegura una experiencia de usuario fluida y precisa al mostrar la ubicación actual del usuario en tiempo real.
Map Controller:
El Map Controller se encarga de la gestión del mapa dentro de la aplicación móvil, incluyendo la visualización de elementos como la ubicación del usuario, las paradas de microbuses y los

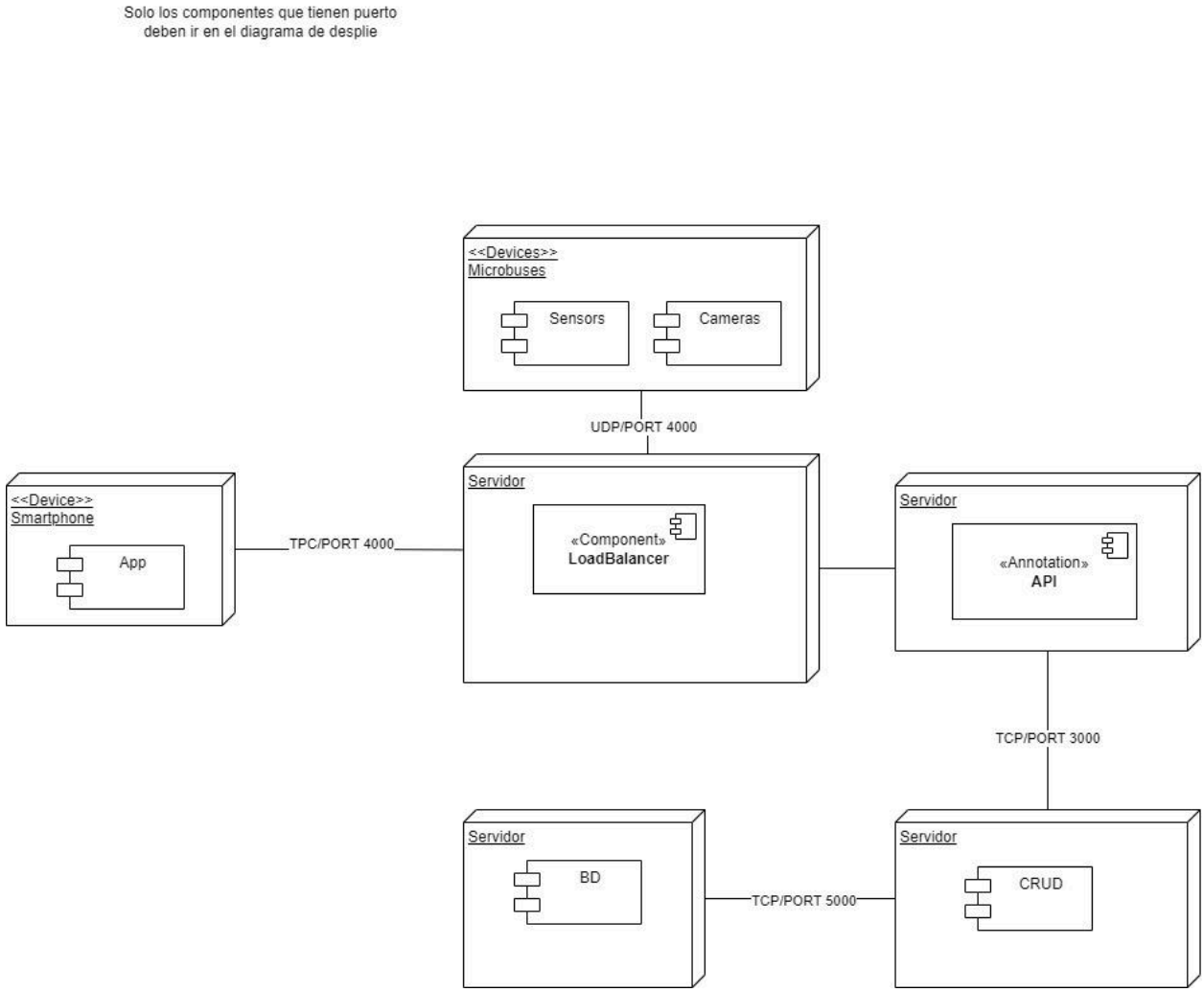
propios vehículos en movimiento. Este componente garantiza una representación clara y precisa del entorno del usuario en la aplicación.

Hardware Simulado:

Sensores:
Los sensores son dispositivos encargados de capturar la ubicación de los microbuses en tiempo real, proporcionando datos precisos para el sistema. Estos dispositivos son fundamentales para la generación de información actualizada sobre la posición de los vehículos en circulación.

Cámaras:
Las cámaras son componentes que registran y procesan la cantidad de pasajeros dentro de los microbuses, detectando acciones como el ingreso y la salida de pasajeros. Esta información es valiosa para monitorear la ocupación de los vehículos y mejorar la gestión de la capacidad de transporte.

Diagrama de despliegue (Modelo fisico)



Modelo fundamental

Modelos de interacción

- Sistema asincrono.
- No es necesario sincronización de relojes debido a que están dentro del mismo país

Modelo de relojes, sistema sincrono o asincrono

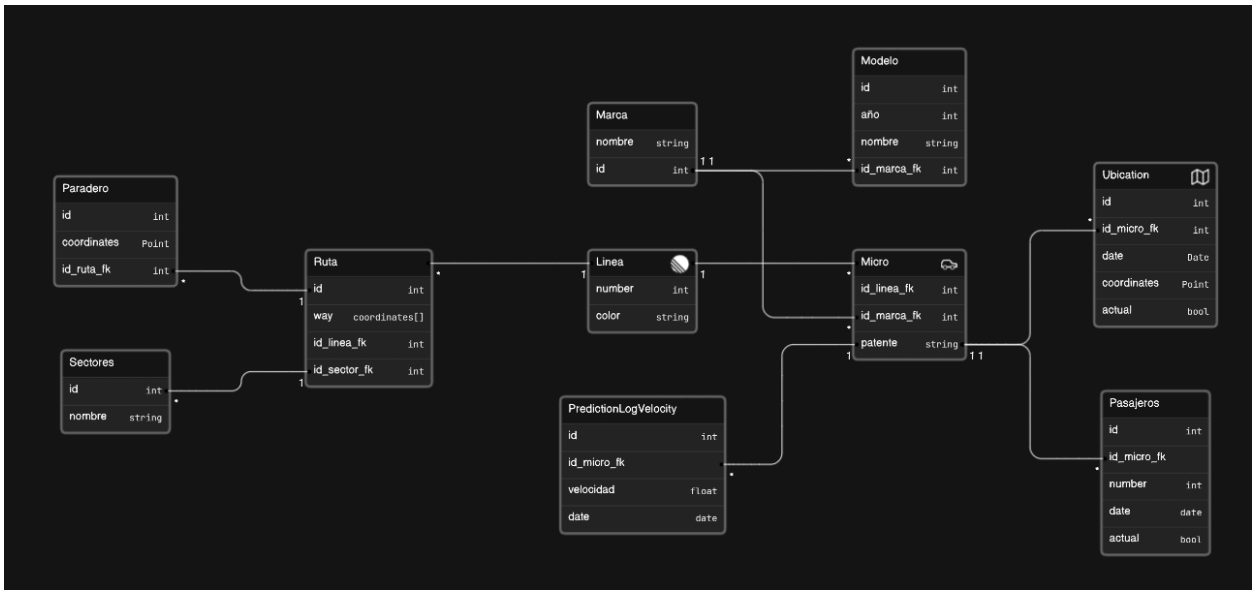
Modelo de fallas

- Respaldos de BD.
- Replicas de BD.
- Multiple APIS.

Modelo de seguridad

- Encriptación de ubicación de “usuarios”.
- Credenciales a las BD.
- Autenticación de interacción entre componentes del sistema.
- Implementación de CORS.
- Implementación de sistemas de log management para recopilar y almacenar registros de eventos de manera segura y eficiente.

Diagrama de persistencia (Modelo relacional)



Enlace para visualizarlo de mejor manera:
https://app.eraser.io/workspace/4oErQzDtr5aIcU4kf2W3?origin=share&elements=sDJp2xs5Bov-LkoHqaUzzg

Diccionario de datos

INFO288 - Diccionario de Datos

Listado de preguntas para Cliente

1. **¿Cual es el alcance de microbuses que desearía que manejara el sistema?**

Sería genial que la app incluyera todas las líneas de micros de Valdivia, pero para empezar, estaría bueno enfocarnos en las líneas 3 y 4. Creo que son las que más usa la gente de la universidad, así que serían perfectas para arrancar con el proyecto.

2. **¿La aplicación deberá ser desarrollada junto al gobierno regional de transporte?**

Claro, sería ideal armar la app con ayuda del gobierno regional de transporte. Estamos en pleno proceso de hacer los contactos necesarios para trabajar con las líneas de micros de la ciudad. Mientras cerramos esos acuerdos y armamos la colaboración oficial, podríamos usar datos simulados para seguir desarrollando y probando la app. Así podemos ajustar todo lo necesario y asegurarnos de que la app cumpla con lo que esperan tanto los usuarios como las autoridades de transporte cuando se lance oficialmente.

3. **Pensando en la escalabilidad del sistema ¿Con cuantos servidores o máquinas cuenta para poder hacer la construcción de la aplicación de manera distribuida?**

Ahora mismo tenemos dos servidores listos para empezar a montar y probar la app. Cada uno tiene 8 GB de RAM y 500 GB de almacenamiento HDD, y ambos corren en Ubuntu 20.04 LTS. No son la última palabra en potencia, pero creemos que van a andar bien para las primeras etapas del desarrollo y las pruebas. Igual estamos atentos para ver si necesitamos más recursos a medida que el proyecto vaya creciendo.

4. **¿Cuál es el propósito principal de la aplicación?¿A quién está dirigido el sistema?**

La idea de la app es súper simple: queremos que la gente en Valdivia pueda checar en tiempo real la info de los microbuses directamente desde su celular.

5. **¿La aplicación debe permitir la visualización de rutas de microbuses específicas, o se desea una vista general de todos los autobuses en una determinada área?**

La app tiene que mostrar todos los micros que estén en una zona específica y que tengan los sensores necesarios para seguirlos en tiempo real. Además, estaría buenísimo que los usuarios pudieran tocar cualquier micro que vean en el mapa para ver información detallada sobre las rutas de ese micro en particular.

6. **¿Hay alguna función adicional que le gustaría que la aplicación incluyera, como notificaciones en tiempo real, integración con el pago de tarifas, la capacidad de compartir ubicaciones con otros usuarios, etc.?**

Soñar no cuesta nada, Y sí, sería increíble tener todas esas funciones avanzadas, como el pago de tarifas integrado o compartir ubicaciones con otros usuarios. Pero, siendo realistas y considerando nuestras limitaciones de tiempo y recursos, lo mejor es concentrarnos primero en

algo clave: las notificaciones en tiempo real. Esa funcionalidad es fundamental para mantener a los usuarios al tanto de lo que está pasando con los micros, como retrasos, cambios de ruta o cualquier otro aviso que pueda afectar sus viajes.

7. **¿Tienen alguna preferencia en cuanto al tipo de notificaciones que la aplicación debe enviar a los usuarios, como alertas de llegada, recordatorios de horarios, noticias del servicio, etc.?**

Totalmente, tener alertas de llegada, recordatorios de horarios y noticias del servicio sería un puntazo porque cada función le agrega algo especial a la experiencia de los usuarios. Aunque, quizás podríamos dejar lo de las noticias para más adelante. Es que integrar noticias significa manejar más datos de distintas fuentes y eso podría complicar las cosas ahora al principio. Mejor nos centramos en las alertas de llegada y los recordatorios de horarios, que están más relacionados con el uso diario de la app y van a ser un beneficio directo y rápido para los usuarios. Luego, cuando tengamos todo más rodado, podríamos añadir las noticias del servicio.

8. **¿Qué tipo de datos específicos les gustaría que se recopilaran y mostrarán en la aplicación? Por ejemplo, ¿números de autobús, horarios, velocidad, historial de ubicaciones, etc.?**

Sería genial que, además del seguimiento básico, pudiéramos integrar datos más específicos que nos ayuden a entender y mejorar el servicio. Por ejemplo, sería muy útil mostrar los horarios peak de pasajeros para cada ruta, algo que podría ayudar a los usuarios a planificar mejor sus viajes evitando las horas más congestionadas. También sería interesante tener datos sobre la velocidad promedio de los buses y los tiempos promedio de recorrido entre paradas. Esto no solo sería útil para los usuarios, sino que también nos ayudaría a analizar y optimizar las rutas. Si existiera un sistema de gestión y monitoreo que trabajara junto con la aplicación de trackeo, podríamos realmente hacer un uso completo de estos datos para mejorar constantemente el servicio de transporte.

9. **¿La aplicación necesita soporte multilingüe para llegar a una audiencia diversa?**

No hace falta meter soporte multilingüe en la app por ahora, ya que vamos a lanzarla en Valdivia y aquí la mayoría habla español. Esto nos simplifica un montón el desarrollo y el mantenimiento inicial. Si más adelante decidimos expandir el servicio a zonas con más variedad de idiomas, ahí sí podríamos pensar en agregar opciones multilingües.

10. **¿Cuál es el volumen esperado de usuarios concurrentes que la aplicación debe ser capaz de manejar durante las horas pico?**

Las horas más movidas suelen ser entre las 8 y 11 de la mañana y después entre las 5 y 7 de la tarde, que es cuando la mayoría se mueve hacia o desde el trabajo o la uni. Tenemos que asegurarnos de que la app aguante bien el tráfico alto de usuarios en esos momentos. Es super importante que funcione de manera eficiente en esas horas para que la experiencia de los usuarios sea buena y el servicio sea confiable.

Presupuesto

Dentro de las herramientas necesarias para la implementación del sistema es la adquisición de sensores IOT, además de sensores gps, y cámaras con microprocesadores que puedan realizar el conteo de ingreso y salida de pasajeros.

Herramientas necesarias:

-

Sensores IoT:
Rango de precio: Entre 30.000 CLP y 70.000 CLP por unidad.

Consideraciones adicionales:
El precio final dependerá del tipo de sensor, la precisión requerida y las características adicionales que ofrezca.
Algunos ejemplos de sensores IoT comunes incluyen sensores de temperatura, humedad, presión, movimiento y presencia.

Sensores GPS:
Rango de precio: Entre 30.000 CLP y 50.000 CLP por unidad.

Consideraciones adicionales:
El precio final dependerá de la precisión del GPS, la conectividad que ofrece (por ejemplo, GPS con cable o GPS inalámbrico) y las características adicionales que incluya.
Algunos ejemplos de características adicionales en sensores GPS incluyen la capacidad de registrar datos de ubicación, la brújula digital y el acelerómetro.

Cámaras con microprocesadores para conteo de personas:
Rango de precio: Entre 50.000 CLP y 300.000 CLP por unidad.

Consideraciones adicionales:
El precio final dependerá de la resolución de la cámara, la velocidad de fotogramas, el campo de visión, la precisión del conteo de personas y las características adicionales que ofrezca.
Algunas características adicionales que se pueden encontrar en las cámaras con conteo de personas incluyen la detección de rostros, el análisis de movimiento y el almacenamiento local de grabaciones.

Para calcular los costos aproximados de la implementación del sistema con las herramientas necesarias, tomaremos los precios promedio de cada tipo de sensor y cámara, y luego los multiplicaremos por el número de unidades requeridas, considerando 10 máquinas por línea y dos líneas de microbuses.

Sensores IoT:
Precio promedio por unidad: (30.000 CLP + 70.000 CLP) / 2 = 50.000 CLP
Total por línea: 50.000 CLP x 10 = 500.000 CLP
Total para dos líneas: 500.000 CLP x 2 = 1.000.000 CLP

Sensores GPS:
Precio promedio por unidad: (30.000 CLP + 50.000 CLP) / 2 = 40.000 CLP
Total por línea: 40.000 CLP x 10 = 400.000 CLP
Total para dos líneas: 400.000 CLP x 2 = 800.000 CLP
Cámaras con microprocesadores para conteo de personas
Precio promedio por unidad: (50.000 CLP + 300.000 CLP) / 2 = 175.000 CLP
Total por línea: 175.000 CLP x 10 = 1.750.000 CLP
Total para dos líneas: 1.750.000 CLP x 2 = 3.500.000 CLP

Resumen de costos totales para dos líneas:
Sensores IoT: 1.000.000 CLP
Sensores GPS: 800.000 CLP
Cámaras para conteo de personas: 3.500.000 CLP

Costo total estimado para la implementación en dos líneas:
1.000.000 CLP + 800.000 CLP + 3.500.000 CLP = **5.300.000 CLP**

Tipo de herramientas para la implementación

Para el desarrollo del sistema, se ha decidido recurrir a opciones de software libre, basándose en la ausencia de un presupuesto destinado para la adquisición de software comercial. Además, se ha evaluado que las alternativas gratuitas disponibles cubren adecuadamente las necesidades del sistema, lo que hace innecesaria la utilización de software de pago. Esta elección se fundamenta en la filosofía de acceso abierto y colaborativo inherente al software libre, así como en la diversidad y calidad de las soluciones disponibles en este ámbito. Al optar por opciones de software libre, se busca maximizar la eficiencia y la viabilidad del proyecto, garantizando al mismo tiempo la legalidad y la ética en el uso de recursos tecnológicos.

Bibliotecas y frameworks

Para el desarrollo del sistema backend, se ha decidido emplear el lenguaje Python, en cumplimiento con los requisitos establecidos por el profesor. En este contexto, se ha seleccionado FastAPI como el framework principal debido a su reputación por ofrecer una integración y desarrollo sencillos y eficientes. En comparación con otras alternativas, como Flask o Django, FastAPI se destaca por su capacidad para manejar de manera eficaz solicitudes HTTP asíncronas y su generación automática de documentación interactiva basada en estándares de la industria, como OpenAPI. Además, la elección de FastAPI se sustenta en su rendimiento superior en comparación con otros frameworks Python para API web, gracias a su uso de funciones asíncronas y tipado estático. Al optar por esta combinación de tecnologías, se busca simplificar el desarrollo de la API y centrar los esfuerzos en garantizar el correcto funcionamiento e integración del sistema distribuido, sin comprometer la calidad y la eficiencia del backend.

Frameworks frontend

Para el desarrollo de la aplicación móvil, se ha optado por utilizar Flutter, un software multiplataforma ampliamente reconocido por su eficiencia en la construcción de aplicaciones móviles. Esta elección se fundamenta en el conocimiento profundo del equipo respecto al framework y en experiencias previas exitosas con su implementación. Flutter destaca por su capacidad de generar interfaces de usuario fluidas y atractivas mediante el uso de widgets personalizables y una arquitectura de compilación avanzada que elimina la necesidad de puentes de comunicación con el sistema operativo subyacente. A diferencia de otros frameworks móviles, como React Native o Xamarin, Flutter adopta un enfoque de renderización directa, donde cada píxel en la pantalla es controlado directamente por el framework, lo que resulta en un rendimiento superior y una experiencia de usuario más consistente en todas las plataformas compatibles. Además, Flutter ofrece una amplia gama de herramientas y bibliotecas que facilitan el desarrollo y la depuración de aplicaciones, lo que lo convierte en una opción atractiva y sólida para este proyecto.

Referencias

Con este link accede al diagrama de despliegue y el de componente.

https://app.diagrams.net/#G1mpuNoFtPfiCkKUiOg_D6wNeWUbutCv0V#%7B"pageId"%3A"x-HpuxCrkzUajJuQxPSA"%7D