Universidad Autónoma Gabriel René Moreno

Facultad de Ciencias de la Computación.

Sistema de Información Geográfica

APLICACIÓN MÓVIL PARA CONTROL Y NAVEGACIÓN DE RUTAS DE BUSES

Grupo C

Docente: PEREZ FERREIRA UBALDO

Estudiantes:

- Arnez Fernández Fabio Alejandro
- Montalván Medina Karen Andrea
- Numbela Vedia Ricardo Jhoan
- Romero Cortez Benjamín Antonio
- Soliz Rueda Javier

Santa Cruz de la Sierra, 08/12/2025

Contenido

1 PERFIL

1.1 ASPECTOS GENERALES

1.1.1 INTRODUCCIÓN

Este documento presenta el análisis y diseño del proyecto para desarrollar una aplicación móvil destinada al control, visualización y navegación de rutas de buses dentro de la ciudad. El sistema permitirá a los usuarios identificar el bus más cercano que los pueda llevar a su destino utilizando algoritmos de rutas óptimas como Dijkstra.

1.1.2 JUSTIFICACION

El desarrollo de software orientado al análisis de redes de transporte requiere precisión, organización y coherencia entre las distintas etapas: desde la definición conceptual de las rutas hasta la implementación final en una aplicación funcional. Una de las tareas más críticas dentro de este proceso es la traducción de la información de paradas, líneas y conexiones a un modelo computacional que pueda ser procesado por algoritmos de optimización.

Actualmente, existen herramientas que permiten representar gráficamente redes de transporte o almacenar datos en hojas de cálculo, pero no ofrecen la posibilidad de integrarse directamente con un sistema que calcule y muestre rutas óptimas en tiempo real. Esto obliga al desarrollador a realizar la conversión y programación de manera manual, lo cual es lento, propenso a errores y genera una desconexión entre el diseño de las rutas y su implementación en la aplicación.

Por otro lado, muchos estudiantes y desarrolladores principiantes se enfrentan a la dificultad de comprender cómo un modelo de red de transporte se traduce en algoritmos de búsqueda de caminos y en interfaces gráficas que permitan visualizar las rutas. Esta brecha afecta el aprendizaje y la productividad, generando ineficiencias en el desarrollo y una curva de aprendizaje más prolongada.

1.1.3 OBJETIVO GENERAL

Desarrollar una aplicación móvil que permita registrar rutas de buses, monitorear en tiempo real la ubicación de cada vehículo y ayudar al usuario a desplazarse desde su ubicación actual hasta un destino seleccionado mediante el cálculo de la mejor ruta disponible.

1.1.4 ALCANCE DEL PROYECTO

El sistema permitirá registrar rutas, buses, paradas y conductores, mostrar en tiempo real la posición de cada bus, indicar al usuario qué bus debe abordar según su destino, y calcular la ruta óptima usando Dijkstra. La app incluirá mapas, seguimiento GPS y recomendaciones de viaje.

1.1.5 ANÁLISIS DE REQUISITOS

1.1.5.1 REQUISITOS FUNCIONALES

1. Registro de Buses y Rutas:

   - Registrar buses, conductores, líneas, paradas y recorridos.
   - Registrar datos como número de línea, placa, conductor, capacidad y ruta asignada.

2. Ubicación en Tiempo Real:

   - Registrar y actualizar la ubicación GPS de cada bus.
   - Permitir que el usuario active su ubicación actual.

3. Selección de Destino:

   - El usuario podrá elegir un destino dentro del mapa.
   - El sistema deberá identificar las paradas cercanas al usuario y al destino.

4. Cálculo de Ruta Óptima (Dijkstra):

   - Determinar el bus más cercano al usuario.
   - Calcular la ruta más eficiente desde la ubicación del usuario hasta su destino.
   - Indicar paradas donde debe abordar y descender.

5. Visualización del Recorrido:

- Mostrar la ruta sugerida en un mapa.
- Indicar información relevante: tiempo estimado, distancia y paradas.

1.1.5.2 REQUISITOS NO FUNCIONALES

- Interfaz amigable.
- Alto rendimiento con múltiples buses transmitiendo ubicación.
- Seguridad de los datos y privacidad de usuarios.
- Escalabilidad para soportar más líneas y buses en el futuro.

1.2 TECNOLOGÍA PARA EL DESARROLLO DEL SOFTWARE

Se utilizará el proceso unificado de desarrollo de software (PUDS), el cual es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software, el proceso unificado es más que un simple proceso, es un marco de trabajo genérico que puede utilizarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones. Diferentes niveles de aptitud y diferentes tamaños de proyectos.

1.2.1 ESTRATEGIA PARA EL DESARROLLO DE SOFTWARE

Habiendo recolectado la documentación requerida para comprender el funcionamiento de las líneas de microbuses y sus rutas, y con el fin de lograr la creación del producto, el desarrollo de este Sistema de Información tendrá un enfoque centrado en dos herramientas de análisis y diseño: el Proceso Unificado de Desarrollo de Software (PUDS) y el Lenguaje Unificado de Modelado (UML).

El PUDS permitirá organizar las fases del proyecto —desde la captura de requisitos hasta la implementación y pruebas— asegurando coherencia y trazabilidad en cada etapa. Por su parte, UML facilitará la representación gráfica de los componentes del sistema

1.2.2 METODOLOGÍA PARA EL DESARROLLO DEL SOFTWARE

1.2.2.1 CARACTERÍSTICAS DEL PUDS

EL PROCESO UNIFICADO ESTA DIRIGIDO POR CASOS DE USO

Para construir un sistema con éxito debemos conocer lo que sus futuros usuarios necesitan y desean. El termino usuario no solo hace referencia a usuarios humanos sino a otros sistemas. En este sentido, el termino usuario representa alguien o algo que interactúa con el sistema que estamos desarrollando. Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso el cual describe la funcionalidad total del sistema.

EL PROCESO UNIFICADO ESTA CENTRADO EN LA AQUITECTURA

El concepto de arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura surge de las necesidades de la empresa, como las perciben los usuarios, y los inversores, y se refleja en los casos de uso. Sin embargo, también se ve influida por muchos otros factores, como la plataforma en la que funcionar el software. El proceso ayuda al arquitecto a centrarse en los objetivos adecuados, como la compresibilidad, la capacidad de adaptación al cambio, y la reutilización.

Debe haber interacción entre los casos de uso y la arquitectura. Tanto la arquitectura como los casos de uso deben evolucionar en paralelo. Los arquitectos moldean el sistema para darle una forma, podemos decir que el arquitecto:

- Crea un esquema en borrador de la arquitectura, comenzando por la parte de la arquitectura que no es especifica de los casos de uso
- A continuación, el arquitecto trabaja con un subconjunto de los casos de uso especificados, con aquellos que representen las funciones clave del sistema en desarrollo
- A medida que los casos de uso se especifican y maduran, se descubre más de la arquitectura. Esto, a su vez, lleva a la maduración de más casos de uso.

Este proceso continuo hasta que se considere que la arquitectura es estable.

EL PROCESO UNIFICADO ES ITERATIVO E INCREMENTAL

El desarrollo de un producto software comercial supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el producto en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo, y en los incrementos, al crecimiento del producto.

En cada iteración, los desarrolladores identifican y especifican los casos de uso relevantes, crean un diseño utilizando la arquitectura seleccionada como guía, implementan el diseño mediante componentes, y verifican que los componentes satisfacen los casos de uso.

Los beneficios de un proceso iterativo controlado:

· La iteración controlada reduce el coste del riesgo a los costes de un solo incremento.

· La iteración controlada reduce el riesgo de no sacar al mercado el producto en el calendario previsto.

· La iteración controlada acelera el ritmo del esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan de manera más eficiente para obtener resultados claros a corto plazo.

· La iteración controlada reconoce una realidad que a menudo se ignora que las necesidades del usuario y sus correspondientes requisitos no pueden definirse completamente al principio.

FLUJOS DE TRABAJO

CAPTURA DE LOS REQUISITOS Es el proceso de averiguar, normalmente en circunstancias difíciles, lo que se debe construir.

ANALISIS

Analizamos los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que nos ayude a estructurar el sistema entero.

DISEÑO

En el diseño modelamos el sistema y encontramos su forma para que soporte todos los requisitos incluyendo los requisitos no funcionales y otras restricciones que se le suponen.

IMPLEMENTACION

El propósito principal de la implementación es desarrollar la arquitectura y el sistema como un todo.

PRUEBA

Verificamos el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas a terceros.

FASES

INICIO

Se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis de negocio para el producto. Esencialmente, esta fase responde a las siguientes preguntas:

ELABORACION

Se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema.

Al final de la fase de elaboración, el director de proyecto está en disposición de planificar las actividades y estimar los recursos necesarios para terminar el proyecto.

TRANSICION

Cubre el periodo durante el cual el producto se convierte en versión beta. En la versión beta un número reducido de usuarios con experiencia prueba el producto e informa de defectos y deficiencias. La fase de transición conlleva actividades como la fabricación, formación del cliente, el proporcionar una línea de ayuda y asistencia y la corrección de los defectos que se encuentren tras la entrega. El equipo de mantenimiento suele dividir esos defectos en dos categorías: los que tienen suficiente impacto en la operación para justificar una versión incrementada (versión delta) y los que pueden corregirse en la siguiente versión normal.

CONSTRUCCION

Se crea el producto. Es esta fase, la línea base de la arquitectura crece hasta convertirse en el sistema completo.

Al final de esta fase, el producto contiene todos los casos de uso que la dirección y el cliente ha acordado para el desarrollo de esta versión.

1.2.2.2 CARACTERÍSTICAS DE UML

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos (modelos) de un sistema que involucra una gran cantidad de software, desde una perspectiva orientada a objetos

UML es una notación, no es un proceso

Se han definido muchos procesos para UML.

Racional ha ideado RUP, el “proceso unificado”. Utilizable para sistemas que no sean software

1. Bloques básicos de construcción
- Elementos Estructurales, Comportamiento, Agrupación, Anotación
- Relaciones
- Diagramas
2. Reglas para combinar bloques
    - Establecen qué es un *modelo bien formado*

3. Mecanismos comunes

   Especificaciones, Extensibilidad, Dicotomía interfaz-relación

4. El Lenguaje Unificado de Modelado (UML) es una notación que combina elementos de tres importantes estándares de diseño OO:

- OMT de Rumbaugh
- Análisis y diseño OO de Booch y
- El modelo de Jacobson

Ha tenido varias modificaciones desde su creación siendo la última versión la 2.0

1.3 HERRAMIENTAS DE DESARROLLO

Las herramientas que usaremos:

1.3.1 SOFTWARE

- Framework backend Django (Python)
- Motor de base de datos PostgreSQL
- Repositorio remoto GitHub
- Editor de código Visual Studio Code
- Contenedorización y despliegue con Docker
- Editor/gestor de base de datos DBeaver o pgAdmin
- Framework frontend Flutter para la aplicación móvil

1.3.2 HARDWARE

- Procesador Core i3, Ryzen 3 o superiores
- Memoria RAM 4 GB o superior (recomendado para ejecutar Docker y Flutter con fluidez)
- Disco duro 120 GB o superior
- Conexión a Internet de al menos 5 MB (para sincronización con repositorios y pruebas de despliegue)

2 MODELO DE DOMINIO

2.1 CLASES PRINCIPALES

2.1.1 LINEA

- id
- nombreLinea
- colorLinea
- imagenLinea
- fecaCreacion

2.1.2 PUNTOS

- id
- latitud
- longitud
- Descripción

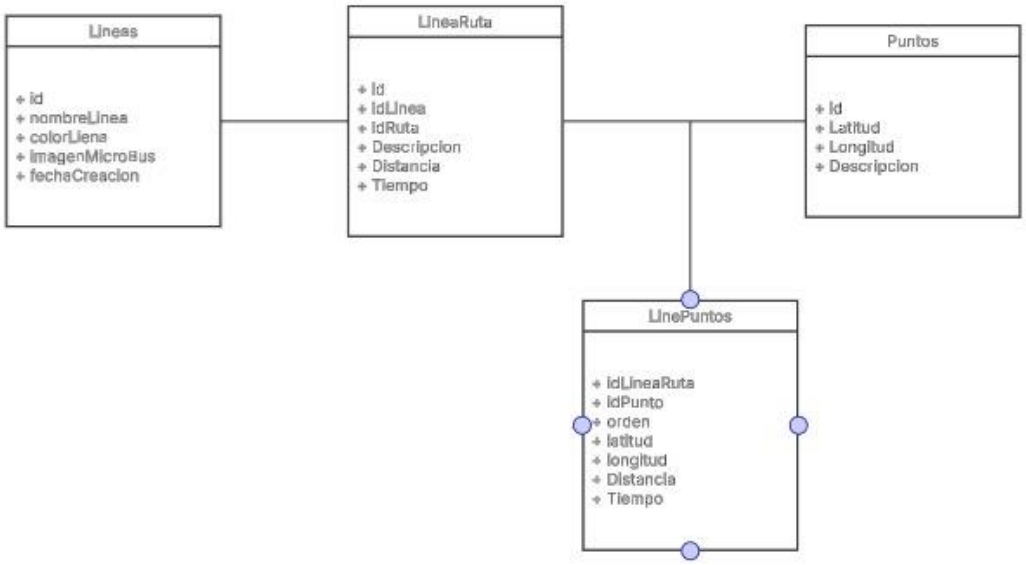
2.1.3 LINEARUTA

- Id
- idLinea
- idRuta
- Descripcion
- Distancia
- Tiempo

2.1.4 LINEAPUNTOS

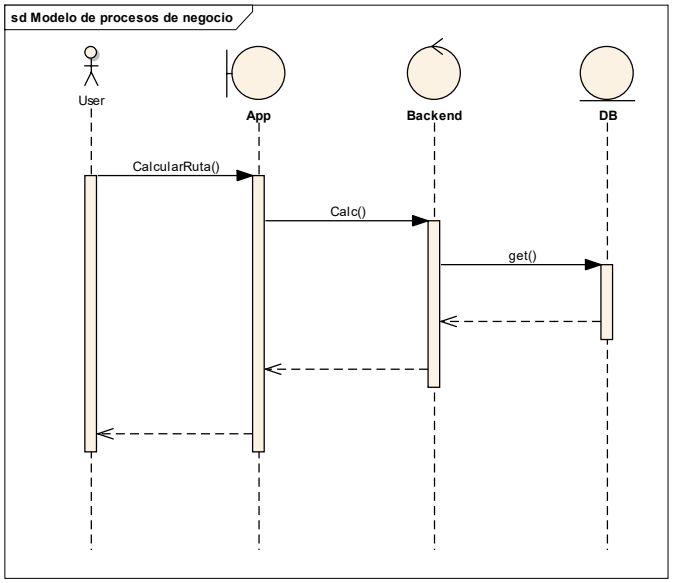
- idLineaRuta
- idPunto
- orden
- latitud
- longitud
- distancia
- tiempo

2.2 DIAGRAMA DE CLASES

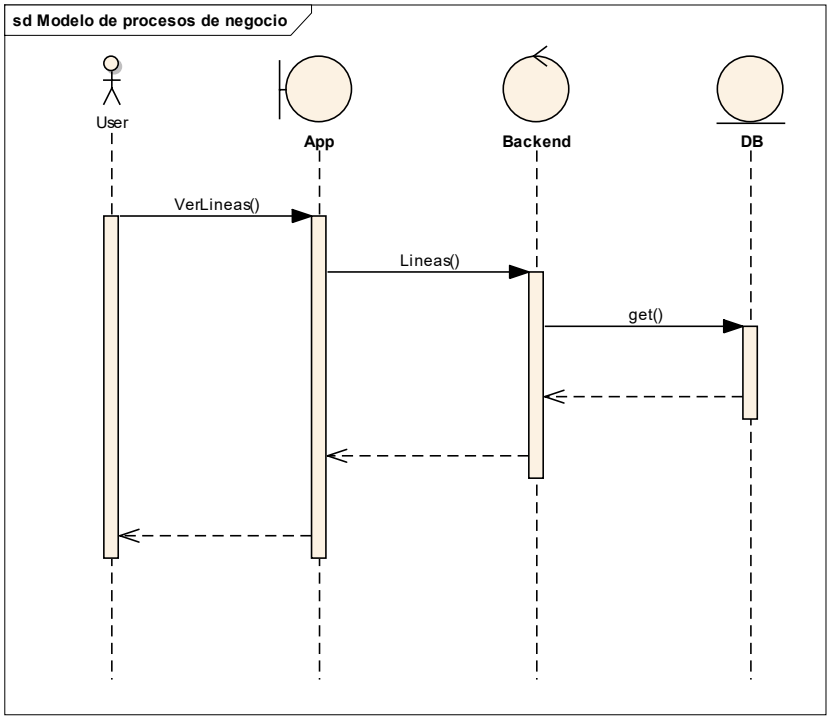


3 MODELO DE NEGOCIO

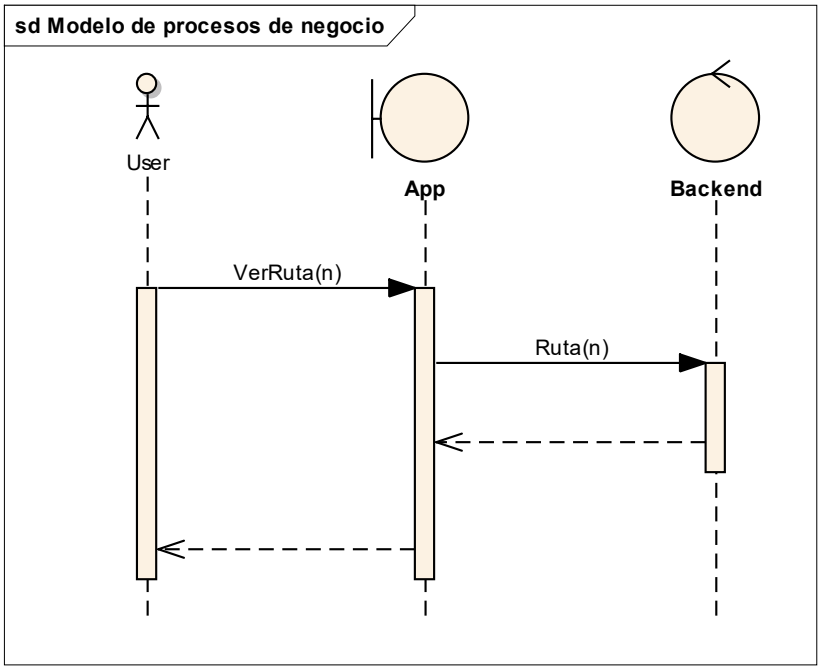
3.1 CALCULAR RUTAS



3.2 VER LINEAS



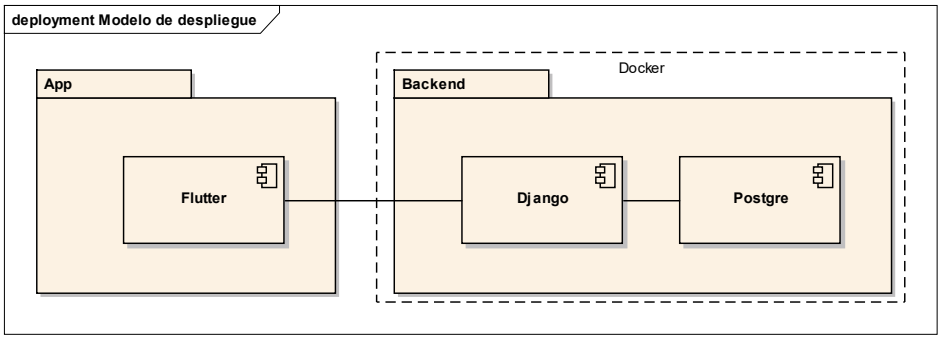
3.3 VER RUTA



4 ANALISIS DE ARQUITECTURA

Dado que el proyecto tiene un alcance reducido y está centrado en una funcionalidad específica —el cálculo y visualización de rutas óptimas de microbuses— no se ha realizado un análisis arquitectónico detallado por paquetes. Sin embargo, se ha definido una estructura clara entre los módulos principales: interfaz móvil (Flutter), backend (Django), base de datos (PostgreSQL) y contenedorización (Docker), lo que permite mantener una separación lógica y funcional entre componentes.

Se utilizarán APIs como Google Maps o Mapbox para geolocalización y visualización de mapas.



5 REPOSITORIO

https://github.com/SuperLoser02/proyecto_SIG.git