UNIVERSIDAD DEL VALLE DE GUATEMALA

Colegio Universitario

Corte 3 Proyecto

Grupo No.7

Catedrática: Lynette García Pérez

Adrián Ricardo González Muralles - 23152
Jose Pablo Ordoñez Barrios – 231329
Manuel Armando Ulin Pérez - 221017
Marcos Rodrigo Ambrocio Larios - 231140
José Alejandro Antón Escobar - 221041
Rene Sebastián Espinal Zamora - 228676

Ingeniería de Software 1
Sección 10
Guatemala, 2025

Resumen

El sistema de movilidad en la Universidad del Valle de Guatemala (UVGride) enfrenta múltiples desafíos que afectan a estudiantes, docentes y colaboradores, como los altos costos y tiempos prolongados de transporte, la inseguridad en medios externos y la falta de estacionamiento adecuado. La falta de opciones accesibles y seguras para trasladarse al campus genera dificultades diarias para la comunidad universitaria, especialmente para quienes viven en zonas alejadas. La dependencia del transporte público y del uso individual de vehículos no solo incrementa los costos y el tiempo de traslado, sino que también contribuye a la congestión vial y a una mayor carga ambiental. Esta problemática impacta directamente en la calidad de vida de los miembros de la universidad, dificultando su acceso oportuno y seguro a sus actividades académicas y laborales.

Introducción

Sistema de Movilidad Universitaria para la Universidad del Valle de Guatemala (UVGride)

Descripción de la entidad

La Universidad del Valle de Guatemala es una institución de educación superior reconocida por su enfoque en la innovación, la investigación y la excelencia académica. Su comunidad está conformada por estudiantes, docentes y colaboradores que diariamente se trasladan al campus para realizar actividades académicas y administrativas. Sin embargo, la falta de un sistema de transporte universitario eficiente representa un desafío significativo, especialmente para aquellos que residen en zonas alejadas dentro del área metropolitana e interdepartamental. Este proyecto se enmarca en la búsqueda de soluciones tecnológicas que mejoren la movilidad dentro de la comunidad universitaria, optimizando tiempos de traslado y ofreciendo un medio de transporte accesible y seguro.

Descripción de la idea

La movilidad dentro de la comunidad universitaria presenta diversas dificultades que impactan negativamente en el acceso a la educación y el bienestar de los estudiantes, docentes y colaboradores. La falta de opciones de transporte adecuadas genera altos costos de traslado, tiempos prolongados de viaje y preocupaciones sobre la seguridad. Muchos estudiantes que viven lejos del campus deben caminar largas distancias, pagar múltiples pasajes o depender de terceros para llegar a sus clases. Además, la ausencia de horarios fijos en los medios de transporte públicos complica la planificación del tiempo de viaje y, en algunos casos, obliga a los estudiantes a mudarse cerca de la universidad, lo que representa un gasto adicional. Quienes poseen vehículo propio también enfrentan desafíos como el alto costo del combustible y la falta de estacionamiento, lo que agrava aún más la problemática de movilidad dentro de la UVG.

Objetivos del proyecto

Objetivo general:
Identificar y analizar las dificultades de movilidad que enfrentan los miembros de la comunidad UVG con el fin de comprender su impacto y proponer posibles estrategias de mejora.

Objetivos específicos:

- Evaluar las necesidades y dificultades de transporte de los estudiantes, docentes y colaboradores de la UVG.
- Identificar los factores que influyen en los costos, tiempos y seguridad en los traslados hacia el campus.

- Analizar las limitaciones del transporte público y privado en el acceso a la universidad.
- Examinar el impacto de la falta de movilidad en la experiencia académica y laboral dentro de la comunidad UVG.
- Explorar posibles enfoques para mejorar la accesibilidad y reducir los inconvenientes relacionados con el transporte.

Este proyecto busca generar un diagnóstico detallado de la situación actual de movilidad en la UVG, permitiendo comprender las dificultades que enfrentan sus miembros y plantear estrategias que contribuyan a mejorar sus condiciones de traslado

Descubrimientos:

- El tráfico es uno de los obstáculos principales, la mayoría de los encuestados tienen problemas con el tráfico.
- Los gastos como la gasolina o el pagar por un transporte como “Uber” es un problema ya que pueden superar los límites de los usuarios.
- Al no contar con un vehículo propio se deben acudir a terceros para poder transportarse.

Necesidades:

- Encontrar alternativas seguras a los transportes públicos por un precio no tan elevado.
- Reducir el impacto que les genera el tráfico en el trayecto diaria hacia la universidad.

Oportunidades:

- Viajes seguros y con personas que sabemos que serán de la universidad.
- Puntualidad y comodidad.
- Apoyo entre colaboradores de la universidad y compañerismo al apoyarse unos a otros.
- Apoyo al estudiante en el transporte todos los días.

Definición del proyecto.

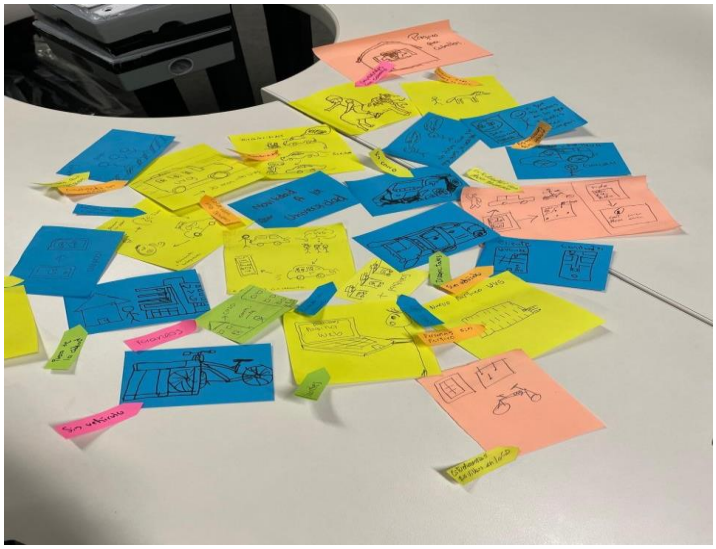
La comunidad universitaria de la Universidad del Valle de Guatemala, compuesta por estudiantes, docentes y colaboradores, enfrenta importantes dificultades de movilidad, especialmente aquellos que residen en áreas distantes al campus. La falta de un sistema de transporte universitario eficiente genera problemas como altos costos de

transporte público, inseguridad, la ausencia de horarios fijos en los buses y escasez de estacionamiento para quienes tienen vehículo propio. Además, la dependencia de terceros para los traslados y la necesidad de mudarse cerca de la universidad debido a la falta de opciones confiables generan estrés y pérdida de tiempo. Este proyecto tiene como objetivo abordar estas dificultades y mejorar la accesibilidad, seguridad y eficiencia en los traslados hacia y desde la UVG, a través de una solución innovadora que beneficie a toda la comunidad universitaria.

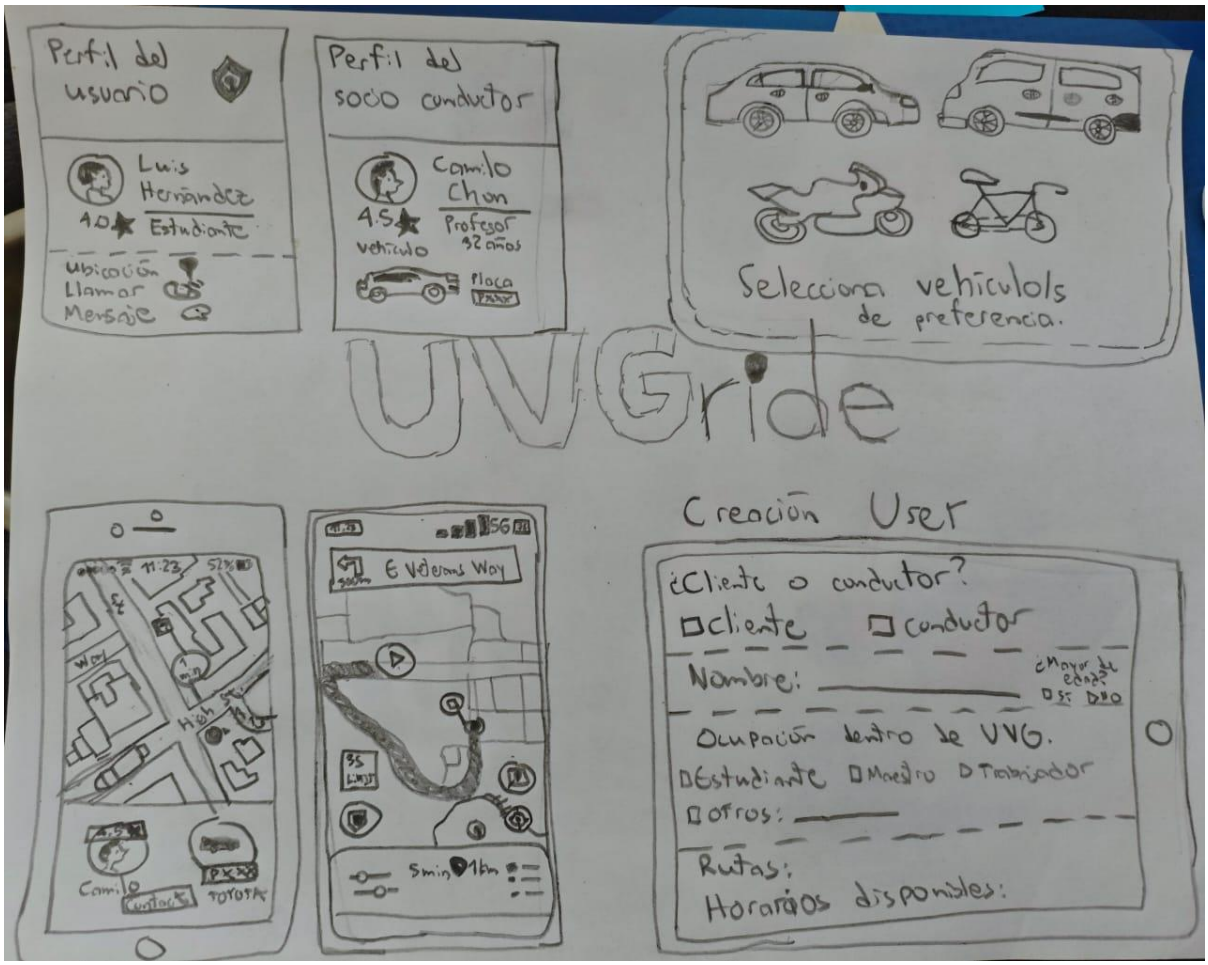
Design Thinking:

- Prototipos:

- Primera fase:



Primer prototipo:



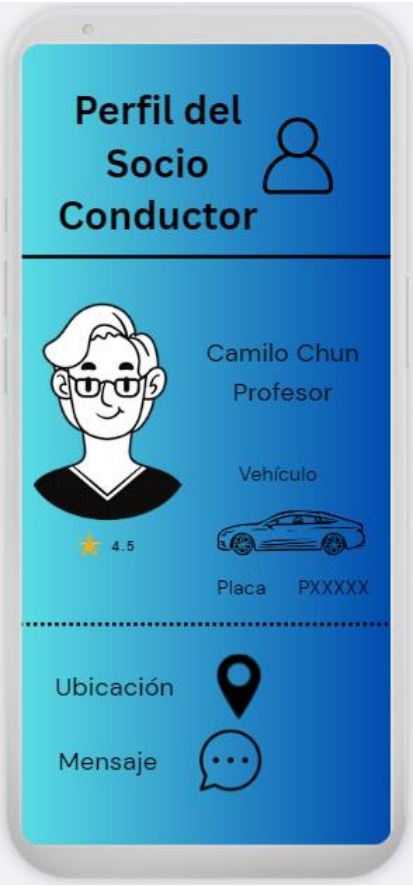
- Prototipo mejorado:

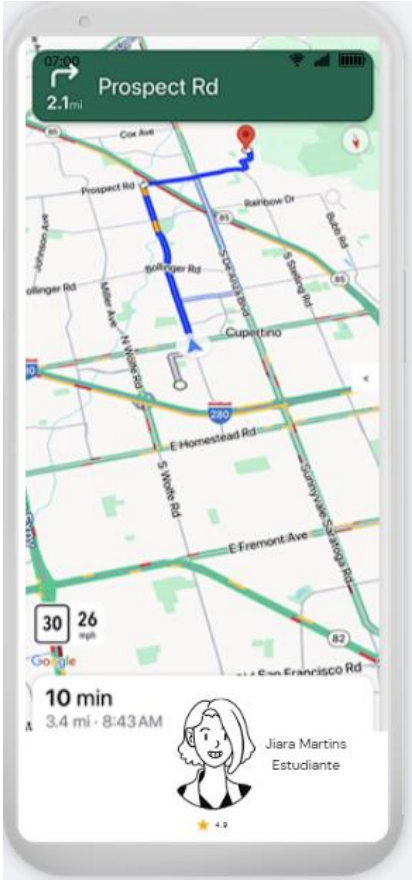

Crea una cuenta
YA ESTAS REGISTRADO? INICIA AQUÍ .
Nombre
JIARA MARTINS
Edad
21
Correo
HELLO@REALLYGREATSITE.COM
contraseña
sign up

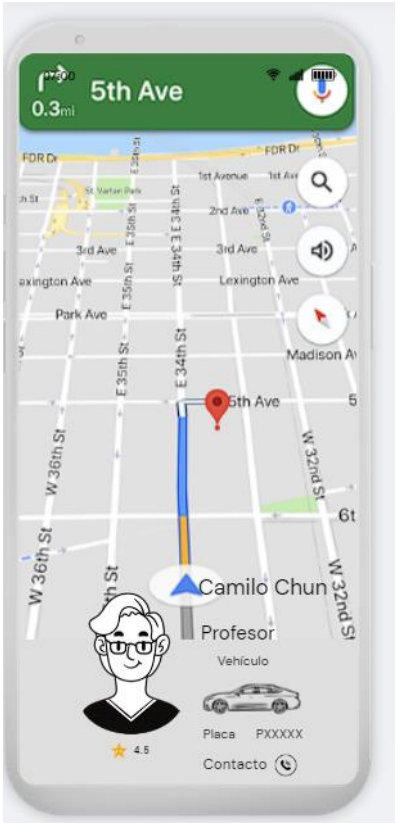

Responde estas preguntas
Para terminar de configurar tu cuenta
CLIENTE O CONDUCTOR?
Cliente
Conductor
OCUPACIÓN DENTRO DE UVG
Estudiante
Maestro
Trabajador
otros
RUTAS
HORARIOS DISPONIBLES
TIENES ALGUNA DISCAPACIDAD?


07:00
Bienvenido
Dani Martinez
Busca un conductor
Categorias
Sedán
Suv
Motocicleta
Bicicleta
Inicio
Rutas
Historial
Perfil


Perfil
JIARA MARTINS
4.9
EDITAR PERFIL
TERMINOS Y CONDICIONES
IDIOMA
INFORMACIÓN
POLITICA DE PRIVACIDAD
Inicio
Rutas
Historial
Perfil


Perfil del Socio Conductor
Camilo Chun
Profesor
Vehículo
4.5
Placa PXXXXX
Ubicación
Mensaje


Prospect Rd
2.1mi
Cupertino
E Homestead Rd
E Fremont Ave
30
26
10 min
3.4 mi · 8:43 AM
Jiara Martins
Estudiante


5th Ave
0.3mi
FDR Dr
1st Avenue
2nd Ave
3rd Ave
Lexington Ave
Park Ave
E 34th St
E 35th St
W 36th St
W 32nd St
6th Ave
Madison Av
Camilo Chun
Profesor
Vehículo
Placa PXXXXX
4.5
Contacto

Análisis:

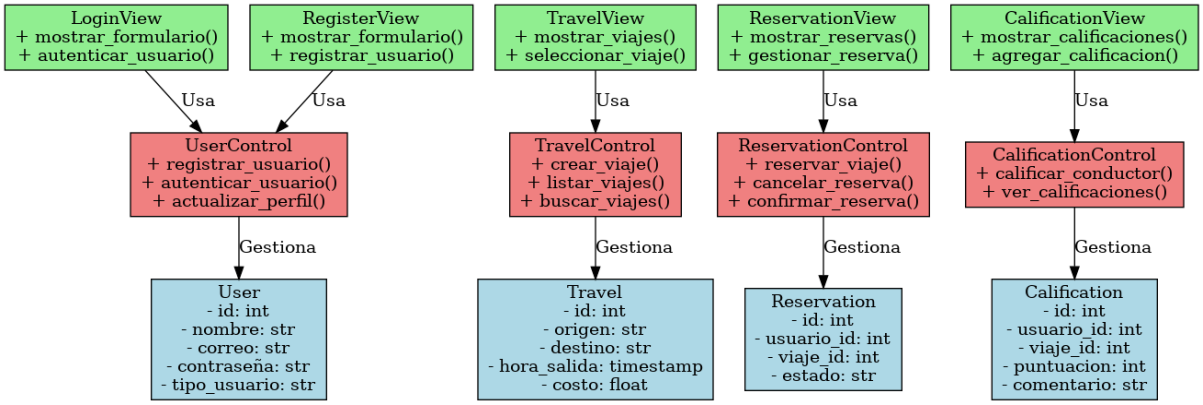
Lista de requisitos funcionales:

Requisito funcional	**Historia de usuario**
El sistema debe permitir a los usuarios registrarse con su correo institucional.	"Como estudiante sin vehículo, quiero una alternativa accesible, segura y confiable para poder llegar a la universidad sin depender de un solo medio de transporte y sin gastar demasiado."
El sistema debe permitir a los usuarios iniciar sesión con credenciales seguras.	"Como profesor, quiero acceder a un medio de transporte seguro y cómodo que me permita transportar pertenencias personales o familiares menores, para poder llegar puntualmente a mis clases."
El sistema debe mostrar opciones de transporte disponibles según la ubicación del usuario.	"Como estudiante que vive lejos de la universidad, quiero encontrar un medio de transporte eficiente y de bajo costo para asistir puntualmente a mis clases sin incurrir en gastos elevados."
El sistema debe permitir filtrar opciones de transporte por precio, ubicación y tipo de viaje.	"Como usuario, quiero filtrar opciones de transporte según mi presupuesto para seleccionar la alternativa más accesible."
El sistema debe permitir a los conductores ofrecer viajes con información de horario, precio y ruta.	"Como usuario, quiero crear un viaje compartido para que otros puedan unirse y reducir costos."
El sistema debe enviar confirmaciones y detalles del viaje a los pasajeros y conductores.	"Como usuario, quiero recibir confirmación y detalles de la reserva."
El sistema debe permitir a los pasajeros reservar un viaje con anticipación.	"Como usuario, quiero reservar transporte con anticipación para asegurar disponibilidad."
El sistema debe incluir un sistema de calificación y comentarios sobre conductores y pasajeros.	"Como profesor, quiero una aplicación que me permita encontrar un conductor confiable con espacio suficiente para transportar mis pertenencias, garantizando un viaje seguro y cómodo hacia la universidad."
El sistema debe ser accesible para usuarios con discapacidad (compatibilidad con lectores de pantalla y opciones de accesibilidad).	"Como persona con discapacidad, quiero encontrar un transporte adaptado a mis necesidades para movilizarme de manera cómoda y sin retrasos."

El sistema debe permitir gestionar los participantes de un viaje compartido.	“Como usuario, quiero unirme a un viaje compartido existente.”

Clases preliminares:

(Revisión) Diagrama de clases MVC



Descripción de las Clases:

Modelo (Entidades)

1. User

• Descripción: Representa a los usuarios del sistema (estudiantes, docentes y conductores).

• Atributos:

• id: int → Identificador único del usuario.

• nombre: str → Nombre completo.

• correo: str → Correo institucional.

• contraseña: str → Contraseña cifrada.

• tipo_usuario: str → Indica si es pasajero, conductor o ambos.

2. Travel

• Descripción: Representa un viaje disponible en el sistema.

• Atributos:

• id: int → Identificador del viaje.

• origen: str → Punto de partida.

• destino: str → Destino final.

• hora_salida: timestamp → Hora de inicio del viaje.

• costo: float → Precio del viaje.

3. Reservation

• Descripción: Representa una reserva de viaje realizada por un usuario.

• Atributos:

• id: int → Identificador de la reserva.

• usuario_id: int → ID del usuario que reserva.

• viaje_id: int → ID del viaje reservado.

• estado: str → Estado de la reserva (pendiente, confirmada, cancelada).

4. Calification

• Descripción: Permite que los usuarios califiquen y comenten sobre su experiencia en un viaje.

• Atributos:

• id: int → Identificador de la calificación.

• usuario_id: int → Usuario que califica.

• viaje_id: int → Viaje asociado.

• puntuacion: int → Calificación numérica (1-5).

• comentario: str → Opinión del usuario.

Controlador (Lógica de Negocio)

5. UserControl

• Descripción: Maneja la autenticación, registro y gestión de usuarios.

• Métodos:

• registrar_usuario() → Crea un nuevo usuario.

• autenticar_usuario() → Valida credenciales.

• actualizar_perfil() → Modifica información del usuario.

6. TravelControl

• Descripción: Gestiona los viajes en la aplicación.

• Métodos:

• crear_viaje() → Permite a un conductor registrar un viaje.

• listar_viajes() → Devuelve la lista de viajes disponibles.

• buscar_viajes() → Filtra viajes por destino, precio o fecha.

7. ReservationControl

• Descripción: Maneja la gestión de reservas.

• Métodos:

• reservar_viaje() → Permite a un usuario reservar un viaje.

• cancelar_reserva() → Cancela una reserva antes del viaje.

• confirmar_reserva() → Confirma la asistencia del pasajero.

8. CalificationControl

• Descripción: Gestiona las calificaciones y comentarios de los usuarios.

• Métodos:

• calificar_conductor() → Permite calificar a un conductor.

• ver_calificaciones() → Obtiene las calificaciones de un usuario.

Vista (Interfaz de Usuario)

9. LoginView

• Descripción: Muestra la pantalla de inicio de sesión.

• Métodos:

• mostrar_formulario() → Despliega el formulario de login.

• autenticar_usuario() → Valida los datos y redirige según el resultado.

10. RegisterView

• Descripción: Permite registrar un nuevo usuario.

• Métodos:

• mostrar_formulario() → Despliega el formulario de registro.

• registrar_usuario() → Envía los datos al controlador.

11. TravelView

• Descripción: Muestra los viajes disponibles para el usuario.

• Métodos:

• mostrar_viajes() → Lista los viajes disponibles.

• seleccionar_viaje() → Permite elegir un viaje específico.

12. ReservationView

• Descripción: Administra la información de las reservas.

• Métodos:

• mostrar_reservas() → Muestra las reservas activas.

• gestionar_reserva() → Permite cancelar o modificar una reserva.

13. CalificationView

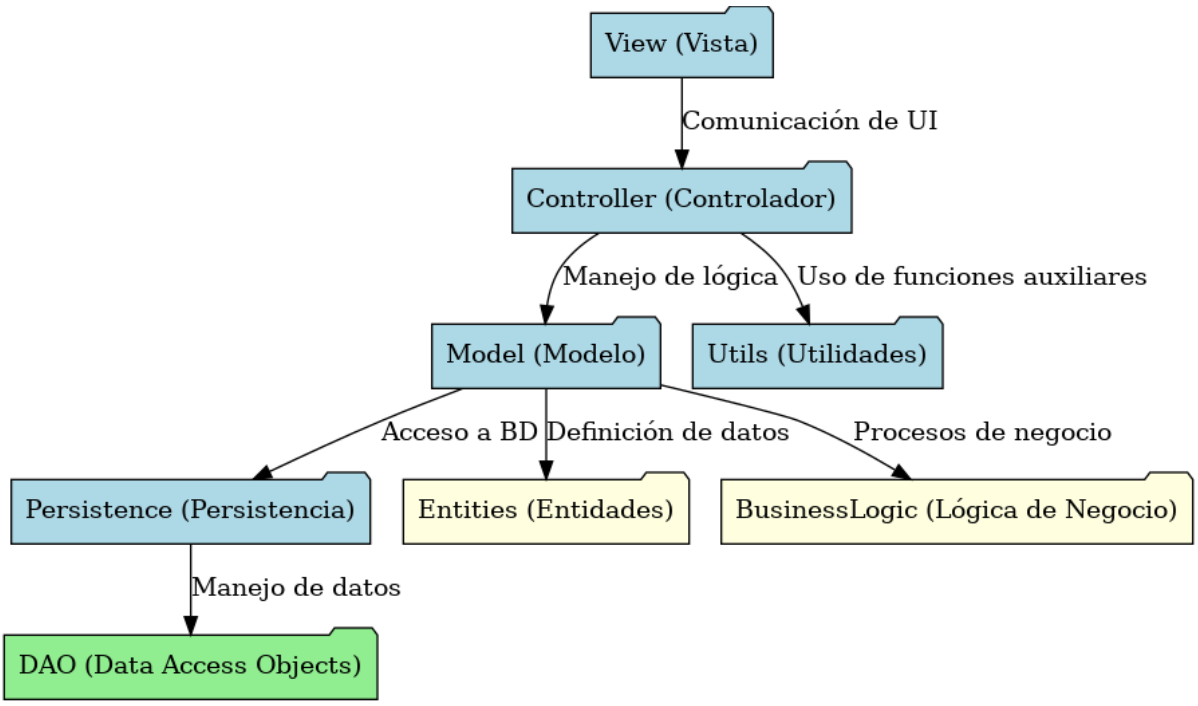
• Descripción: Muestra las calificaciones de los conductores y pasajeros.

• Métodos:

• mostrar_calificaciones() → Lista calificaciones y comentarios.

• agregar_calificacion() → Permite a un usuario dejar su evaluación.

Diagrama de paquetes:



Descripción de cada paquete y de sus componentes:

1. View

Este paquete maneja la interfaz de usuario y la interacción con el sistema. Se encarga de mostrar los datos procesados por el controlador y capturar la información ingresada por los usuarios.

Componentes:

• LoginView: Muestra la pantalla de inicio de sesión y envía los datos al controlador.

• RegisterView: Permite el registro de nuevos usuarios.

• TravelView: Muestra la lista de viajes disponibles.

• ReservationView: Permite gestionar reservas activas.

• CalificationView: Muestra y registra calificaciones de conductores y pasajeros.

• PaymentView: Muestra opciones de pago y estado de transacciones.

• NotificationView: Muestra notificaciones de reservas, pagos y cambios en el sistema.

Relación: Se comunica con el paquete Controller para enviar y recibir datos.

2. Controller

Este paquete gestiona la lógica de negocio. Recibe información de la Vista, la procesa y la envía al Modelo. También obtiene datos desde el Modelo y los devuelve a la Vista en un formato adecuado.

Componentes:

• UserControl: Gestiona autenticación, registro y actualización de usuarios.

• TravelControl: Maneja la creación y búsqueda de viajes.

• ReservationControl: Administra reservas y confirmaciones.

• CalificationControl: Permite registrar y consultar calificaciones.

• PaymentControl: Gestiona métodos de pago y transacciones.

• NotificationControl: Maneja la lógica de notificaciones para usuarios.

Relación: Se comunica con la Vista para recibir y mostrar datos, y con el Modelo para aplicar reglas de negocio.

3. Model

Este paquete define la estructura de los datos del sistema y la lógica de negocio sin depender de la interfaz gráfica.

Subpaquetes:

• Entities (Entidades): Clases que representan las tablas de la base de datos.

• User: Representa a los usuarios (estudiantes, docentes, conductores).

• Travel: Representa un viaje con su información (origen, destino, horario).

• Reservation: Representa una reserva asociada a un viaje y un usuario.

• Calification: Registra puntuaciones y comentarios de los usuarios.

• Payment: Gestiona métodos de pago y transacciones.

• Notification: Representa notificaciones enviadas a los usuarios.

• BusinessLogic (Lógica de Negocio): Métodos que procesan los datos antes de enviarlos a Persistencia.

• Pricing: Calcula tarifas y costos de viajes.

• MatchingService: Asigna pasajeros a viajes disponibles.

• Security: Maneja validaciones y cifrado de contraseñas.

Relación: Se comunica con Persistencia para acceder a la base de datos y con Controller para proporcionar datos procesados.

4. Persistence

Este paquete maneja la conexión con la base de datos y realiza operaciones CRUD (Crear, Leer, Actualizar, Eliminar).

Subpaquete:

• DAO (Data Access Objects): Clases encargadas de la comunicación con la base de datos.

• UserDAO: Gestiona consultas sobre los usuarios.

• TravelDAO: Maneja los viajes almacenados en la base de datos.

• ReservationDAO: Controla las reservas registradas.

• CalificationDAO: Administra las calificaciones y comentarios.

• PaymentDAO: Gestiona pagos y transacciones.

• NotificationDAO: Administra el almacenamiento y envío de notificaciones.

Relación: Se comunica con el Modelo para almacenar y recuperar datos de la base de datos.

5. Utils

Este paquete contiene herramientas auxiliares utilizadas en todo el sistema.

Componentes:

• EmailService: Maneja el envío de correos electrónicos (confirmaciones, alertas).

• Logger: Registra eventos, errores y auditorías del sistema.

• Security: Contiene métodos para cifrado de contraseñas y validaciones de seguridad.

Relación: Es utilizado por Controller y Persistencia para funcionalidades adicionales.

Diagrama de clases persistentes:

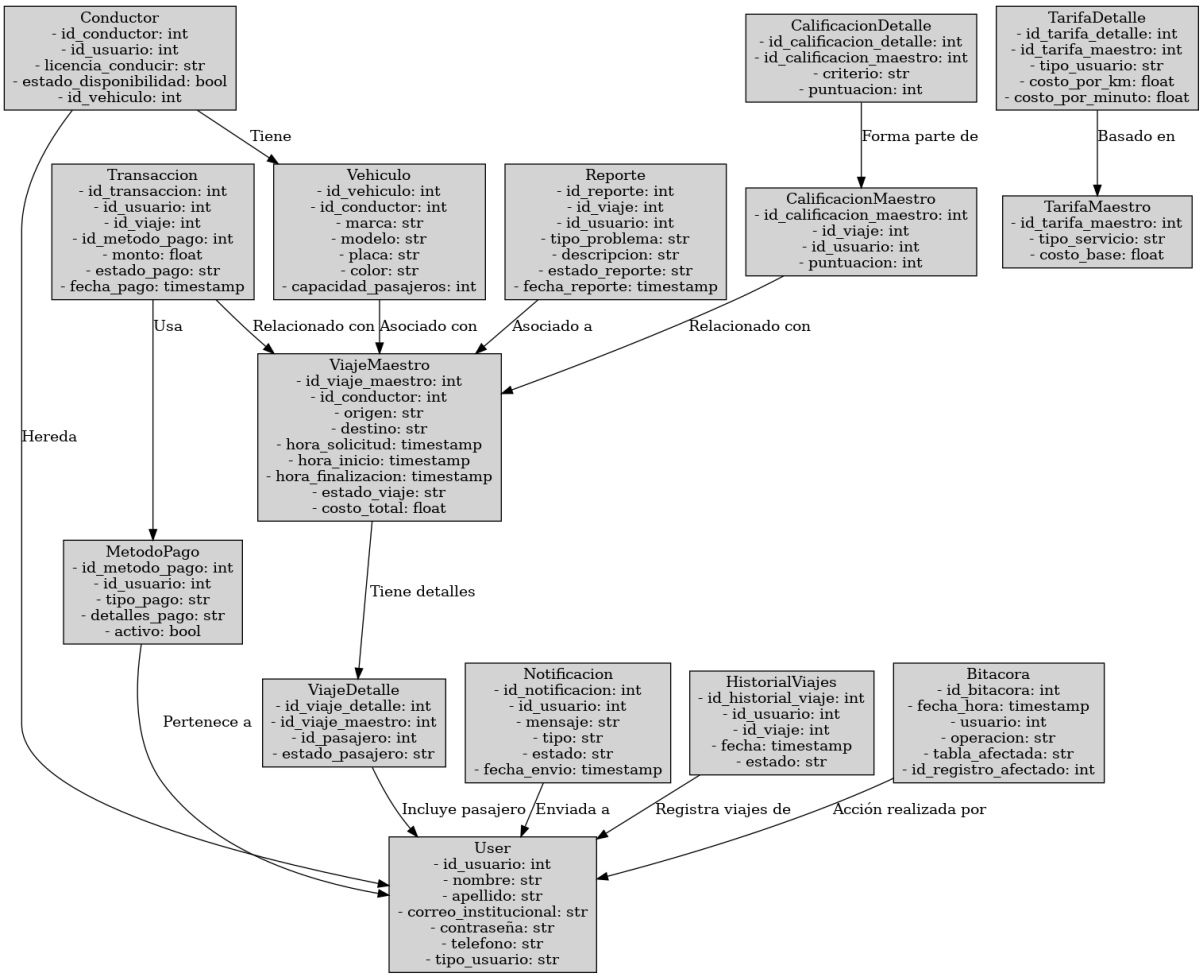
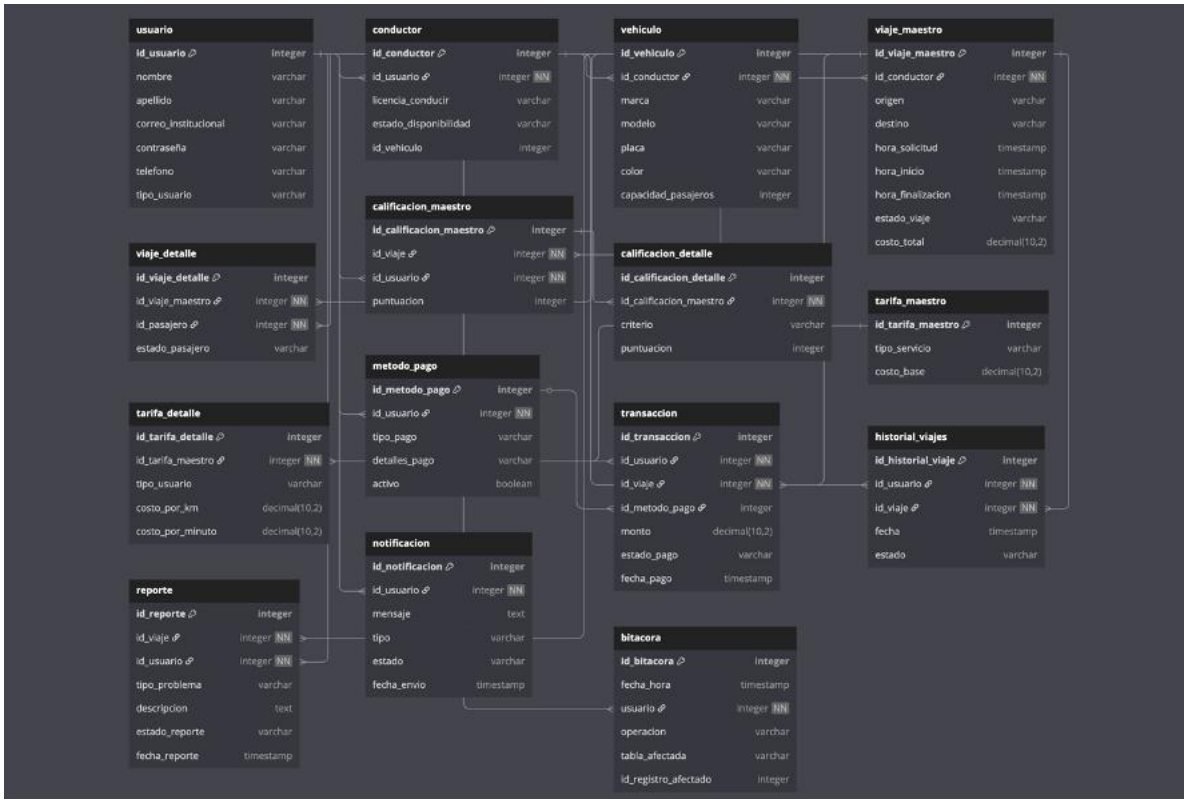


Diagrama Entidad-Relación:



VI. Diseño

1. Estimación de Dimensiones del Sistema

- Cantidad de usuarios probables

La UVG tiene aproximadamente 9,000 - 15,000 entre estudiantes y trabajadores.

Se estima que 1,000 - 2,500 usuarios podrían registrarse en la app dentro del primer año.

Usuarios concurrentes estimados: 50 - 300 en horas pico.

- Tiempo promedio de respuesta

La app debe procesar solicitudes en menos de 1 segundo en acciones clave como:

Buscar viajes disponibles.

Crear un nuevo viaje.

Confirmar participación en un viaje.

Para esto, el backend debe estar optimizado con caché (Redis) y consultas eficientes.

- Cantidad de usuarios promedio simultáneos

Durante picos (mañana y tarde), se espera 50 - 300 usuarios concurrentes.

Se requiere balanceo de carga para evitar cuellos de botella.

- Escalabilidad del sistema,

para esto debe considerarse la información que deberá
gestionar a medida que vaya creciendo por el uso. Deberían estimar el tamaño en disco que se esperaría de la información almacenada.

- Disponibilidad de la información.
- Confiabilidad, tolerancia a fallos masivos, replicación de la información.

Solución: Uso de PostgreSQL para datos estructurados + Firebase Firestore para datos en tiempo real.

- Disponibilidad de la Información

La app debe estar disponible 24/7, con tiempo de inactividad mínimo. Los backups se harían automáticamente al menos 1 al día

- Confiabilidad y Tolerancia a Fallos

Autenticación segura con Firebase Authentication (Google, correo, contraseña).

Replicación de base de datos en AWS RDS para evitar pérdida de información.

Mensajería en tiempo real con Firebase para mantener las notificaciones activas.

Selección de Tecnologías para el Desarrollo

Frontend: React Native con TypeScript

- Ventajas:

Multiplataforma (iOS y Android).

Alto rendimiento con React Native Hermes.

Comunidad amplia y gran soporte (Facebook/META).

Reutilización de código (posible integración con web).

Soporta WebSockets y notificaciones push fácilmente.

- Desventajas:

Mayor dependencia de librerías de terceros.

Algunas funciones requieren módulos nativos adicionales.

- ¿Por qué React Native?

Elegimos React Native por el amplio soporte que tiene para desarrollar y que es una crossplataform y nos permite usar el mismo codigo para todos los móviles tanto android como IOs

Backend: Node.js con Express.js

- Ventajas:

Alto rendimiento con asincronía.

Compatible con WebSockets para tiempo real.

Escalabilidad en aplicaciones con múltiples conexiones.

- Desventajas:

No es eficiente para procesamiento intensivo en CPU.

Puede volverse desordenado sin una buena estructura.

- ¿Por qué Node.js?

Node.js se integra fácilmente con React Native mediante APIs REST o GraphQL, permitiendo una comunicación fluida entre el backend y la aplicación móvil. Su arquitectura basada en eventos y su modelo asíncrono lo hacen altamente escalable, soportando múltiples usuarios simultáneos sin afectar el rendimiento. Además, al combinarlo con WebSockets, facilita la comunicación en tiempo real, ideal para funcionalidades como notificaciones, chats y actualizaciones en vivo dentro de la app.

Bases de Datos

PostgreSQL

- Ventajas:

Relacional y estructurado, ideal para datos de usuarios y viajes.

Alta consistencia y transacciones seguras.

Compatible con escalabilidad vertical y horizontal.

Funciones avanzadas para geolocalización (PostGIS).

- Desventajas:

Más configuración inicial que otros sistemas NoSQL.

Menos flexible que NoSQL para datos dinámicos.

- ¿Por qué PostgreSQL?

PostgreSQL es la mejor opción para este proyecto debido a su capacidad para manejar datos estructurados, permitiendo gestionar eficientemente usuarios, historial de viajes y pagos con consultas SQL avanzadas. Además, su extensión PostGIS facilita el almacenamiento y análisis de datos geoespaciales, ideal para rastrear ubicaciones y optimizar rutas. Su escalabilidad permite mejorar el rendimiento mediante índices y particiones, asegurando que el sistema pueda crecer sin comprometer la velocidad ni la estabilidad.

Documento donde se trabajo:

https://uvggt-my.sharepoint.com/:w:/g/personal/uli221017_uvg_edu_gt/ES2MCcdyBEdAhkR9NQHy6WEBG25RnnqHHF10yPj5rgC1XA?e=FZrDhg