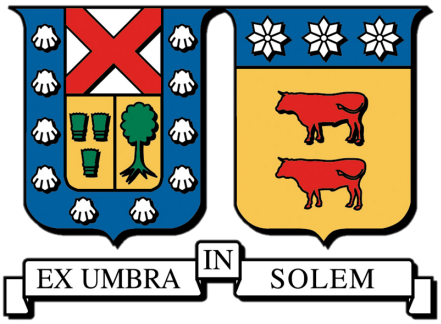
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



"DISEÑO Y DESARROLLO DE UNA PLATAFORMA DE VIAJES COMPARTIDOS, INTEGRANDO A LAS TIC PARA FACILITAR Y MODERNIZAR EL TRANSPORTE INTERURBANO DE PERSONAS"

CARLOS ALBERTO IGNACIO ELGUETA ESTAY
CLAUDIO ANDRÉS CAMPUZANO BARRAZA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL TELEMÁTICO

PROFESOR GUÍA 1:	**NICOLÁS JARA**
PROFESOR GUÍA 2:	**JOSÉ MANUEL MARTÍNEZ**

MAYO 2022

Agradecimientos Carlos

En primer, segundo y tercer lugar quiero agradecer a familia por su apoyo incondicional, donde a veces unas palabras de ánimo, un abrazo o simplemente la compañía, me sostuvieron cuando más lo necesité, a mis padres y hermanas, sin ustedes nunca hubiera llegado aquí, los amo.

También debo agradecer a todos aquellos compañeros con quienes compartí durante estos años, a aquellos que se fueron, aquellos que quizá no son mis amigos, pero que de algún u otro modo nos cruzamos en múltiples (asados) ramos durante la carrera.

La mítica B110, como no recordar con quienes más compartí y pasamos grandes batallas, dándonos ánimos (¡¡"Wooh dale [nombre]!!") Gracias a todos... En especial al Lucho y al Crack, con quienes además compartí mi primer trabajo.

No voy a terminar sin agradecerle particularmente a este último, con quien a pesar de haber tenido encuentros durante el desarrollo, supimos sacar adelante el proyecto como equipo, como amigos, gracias por todo compadre.

Carpe Diem

Carlos Alberto Ignacio Elgueta Estay

Agradecimientos Claudio

Mi más intenso y sincero agradecimiento a mi familia, por todo el apoyo durante toda mi etapa universitaria, sé que la incertidumbre a veces fue intensa, pero finalmente llegamos a la meta.

A Carolina, por estar en gran parte de mi proceso universitario, apoyándome y aconsejándome, haciéndome feliz en esos momentos de gran estrés. Esto también es por ti *wey*.

A mis compañeros y amigos de la universidad (que pasaron a ser de la vida), que hicieron muy ameno el paso por esa etapa, Machine, Jetson, Nakio, Bboy, Jiro, Alonsito, Santi, y la bro Nicol. Y a mis amigos de la vida, Fernandito y la vieja. Nunca dejen de existir.
Un agradecimiento especial a mi compadre Roto, por el gran esfuerzo y aguante que le dimos todo este tiempo, al fin lo sacamos compadre, se hizo, un grande.

A todos, gracias infinitas, se viene su *asadito*.

Recuerda, la Fuerza estará contigo, siempre.
Obi-Wan Kenobi

Claudio Andrés Campuzano Barraza

Resumen

Cotidianamente, las personas realizan viajes con carácter de interurbano, utilizando tanto vehículos particulares, como trenes o buses interurbanos. De estas formas de viajar, el primero es el más rápido y seguro, pero es ambientalmente costoso, ya que generalmente quedan varios asientos sin ocupar. Mientras tanto, en los otros transportes mencionados, no existe ese problema (por la naturaleza de estos), sin embargo, existen otros problemas como; lentitud, inseguridad e incomodidad. Además de esto, también se encuentran problemas asociados al mercado del transporte, y que, al existir rutas con pocas empresas, estas pueden controlar los precios según les convenga, viciando el mercado.

Dadas las razones antes mencionadas, surge la necesidad de idear una solución a estos problemas, rescatando las características destacables de ambos sistemas. Para esto se hace uso de las TIC, permitiendo a los usuarios experimentar todas las ventajas de la tecnología, como viajes más inteligentes, rápidos, seguros, cómodos y personalizables, a través de la implementación del Carpool (viaje compartido).

En la presente memoria de titulación, con el objetivo de mejorar el transporte integrando las TIC, se analizan las diferentes dificultades que presenta el panorama actual de transporte Chile, y como estas son solucionadas a través del Carpool, para luego mostrar el diseño e implementación a través de Liion, plataforma que permite a conductores y pasajeros, poder coordinar viajes, detallando las tecnologías a utilizar, en conjunto a la implementación técnica de cada módulo.

Aunque el *Carpooling* es presentado como la solución a los problemas del transporte interurbano, en realidad, no es así, o al menos no del todo. Sin embargo, y a pesar de que Liion no resuelve todas las problemáticas planteadas, si es una solución viable de implementar en el contexto de Chile, ya que hay soluciones similares orientadas al desplazamiento de personas en formato de taxi.

Finalmente, Liion es una solución tecnológica que resuelve todos los requisitos que una aplicación de transporte debe cumplir, incluyendo los inherentes de las tecnologías,

y aquellos relacionados con el negocio del transporte de pasajeros, como brindar las herramientas para que los usuarios confíen en la plataforma.

keywords: Carpool, mobile application, shared rides, React Native, Firebase

Abstract

Daily, there is intercity travel using private vehicles, trains, or intercity buses. Of these methods, the first is the fastest and safest, but it is environmentally costly, as there are usually several seats left unoccupied. Meanwhile, in the other mentioned transports, there is no such problem due to the nature of these, however, there are other problems such as; slowness, insecurity, and discomfort. In addition to this, there are also problems associated with the transportation market, and that, as there are routes with few companies, these can control prices as they see fit, vitiating the market.

Given the aforementioned reasons, the need arises to devise a solution to these problems, rescuing the outstanding features of both systems. For this purpose, ICT is used, allowing users to experience all the advantages of technology, such as smarter, faster, safer, more comfortable, and customizable trips, through the implementation of Carpool.

In this thesis to improve transportation by integrating ICT, we analyze the different difficulties of the current transportation scenario in Chile, how these are solved through the Carpool, and then show the design and implementation through Liion, a platform that allows drivers and passengers to coordinate trips, detailing the technologies to be used, together with the technical implementation of each module.

Although *Carpooling* is presented as the solution to the problems of intercity transportation, in reality, it is not so, or at least not entirely. However, although Liion does not solve all the problems raised, it is a viable solution to implement in the context of Chile, since there are similar solutions oriented to the displacement of people in cab format.

Finally, Liion is a technological solution that solves all the requirements that a transportation application must fulfill, including those inherent to the technologies, and those related to the passenger transportation business, such as providing the tools for users to trust the platform.

keywords: Carpool, mobile application, shared rides, React Native, Firebase

Glosario y Acrónimos

Carpool o Carpooling: Práctica que consiste en compartir un automóvil con otras personas tanto para viajes periódicos como para trayectos puntuales.

ENUSC: Encuesta Nacional Urbana De Seguridad Ciudadana.

Seremi: Secretario Regional Ministerial.

IIASA: Instituto científico internacional que realiza investigaciones sobre los problemas críticos del cambio ambiental, económico, tecnológico y social global que enfrenta el siglo XXI.

HOV: De *High Occupancy Vehicle*, se traduce a vehículo altamente ocupado y hace referencia a los vehículos los cuales tienen un alto porcentaje de sus asientos ocupados

Ridesharing: Hace referencia a todas las aplicaciones de transporte que permiten a dos o más personas compartir un vehículo durante un viaje.

Front-end: Corresponde a la capa de la vista en la aplicación web, basado en el esquema Modelo-Vista-Controlador.

Back-end: Capa de acceso a los datos, donde se desarrolla el motor de la aplicación.

Base de datos: Es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite.

Requisito funcional: Define una función del sistema de software o sus componentes, que es descrita como un conjunto de entradas, comportamientos y salidas.

GPS: De *Global Positioning System* Es un sistema que permite determinar en toda la Tierra la posición de cualquier objeto o Ser, con una precisión de hasta centímetros, aunque lo habitual son unos pocos metros de precisión.

Login: Es el proceso que controla el acceso individual a un sistema informático mediante la identificación del usuario utilizando credenciales provistas a este.

API: De *Application Programming Interface*, es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software.

JSON: Es un formato de texto sencillo para el intercambio de datos.

SVG: Es un tipo de formato el cual especifica un gráfico vectorial con gran facilidad para escalar en diversos tamaños de pantallas.

MVP: De *minimum viable product*, corresponde al conjunto de funcionalidades mínimas que tendrá un proyecto.

Framework: Esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos.

Widget: Herramienta que muestra información interactiva en la pantalla sin necesidad de abrir la aplicación principal.

ETL: Por sus siglas extracting, transform, load, son programas especializados para analizar streams de datos en tiempo real.

RFC: *Request for Comments*, serie de publicaciones que describe las bases del funcionamiento de Internet.

FCM: *Firebase cloud messaging*, solución de mensajería multiplataforma que permite enviar mensajes de forma fiable y sin coste alguno.

OMS: Organización mundial de la salud.

SQL: Structured query language es un lenguaje estructurado de para realizar operaciones a las bases de datos.

Inyecciones SQL: Vulnerabilidad donde se inyecta código Sql en campos de texto

openSource: Software cuyo código esta al alcance de quien lo quiera

HTTP: Hypertext transfer protocol, protocolo de capa de aplicación diseñado comunicar servidores web y sus clientes.

HTML: HyperText markup language, lenguaje que define el significado y contenido del contenido web

request: De petición en inglés hace referencia a una petición de acceso a algún recurso usando el estándar HTTP

request body: Hace referencia al cuerpo de las peticiones HTTP, tanto en las peticiones HEAD, como GET, este es nulo.

request query string: Es una parte del URL en la request, donde se pueden especificar parámetros.

Microservicios: Los microservicios son un enfoque arquitectónico y organizativo para el desarrollo de software donde el software está compuesto por pequeños servicios independientes que se comunican a través de API bien definidas.

Kubernetes: Es una plataforma de código abierto para gestionar cargas de trabajo y servicios en contenedores

UID: User ID, identificador único para cada usuario.

Código QR: Es un tipo de código de barras que puede ser leído fácilmente por un dispositivo digital y que almacena la información como una serie de píxeles en una cuadrícula.

Índice de figuras

1.1. Mega-toneladas de emisiones de Dióxido de Carbono (CO_2) generadas por transporte en ruta, según categoría vehicular, entre 2005-2017 . . . 11
1.2. Percepción de inseguridad en los microbuses en los últimos 12 meses ENUSC 2018. 14
1.3. Costo de un viaje a Viña del Mar Santiago según la aplicación Carretera [1] 15

2.1. Publicidad incentivando el uso del *carpool* durante la Segunda Guerra Mundial 20
2.2. Resultados a la pregunta "¿Posees automóvil propio?" 30
2.3. Resultados a la pregunta "¿Utilizas tu auto para realizar viajes de larga distancia" 31
2.4. Resultados a la pregunta "¿Estarías dispuesto a compartir tu automóvil para hacer viajes compartidos de larga distancia?" 32
2.5. Resultados a la pregunta "¿Disposición a compartir como pasajero el automóvil?" 33
2.6. Resultados a la pregunta "¿Estarías dispuesto a usar Liion?" 34

4.1. Arquitectura general de una aplicación de 3 capas (de elaboración propia). 45
4.2. Interacción de Flutter con componentes nativos. 47
4.3. Interacción de React Native con los componentes nativos. 48
4.4. Arquitectura Xamarin. 50

4.5. Arquitectura de Native Scripts. 51
4.6. Arquitectura de Ionic. 52
4.7. Cantidad de consultas en Stack OverFlow para cada *framework* al 12/05/2021. 53
4.8. Cantidad de repositorios en GitHub para cada *framework* al 12/05/2021. 53
4.9. Cantidad de consultas por tag en Stack OverFlow para cada *framework* al 02/06/2021. 60
4.10. Cantidad de repositorios en GitHub para cada *framework* al 03/06/2021. 61
4.11. SQL y No-SQL 65
4.12. Base de datos *Column Family* 66
4.13. Funcionamiento de una API. 73
4.14. Arquitectura del Sistema 75
4.15. Arquitectura de la API 81
4.16. Flujo del guardián de sesión 83
4.17. Entidad users 88
4.18. Entidad travels 91
4.19. Entidad request travels 95
4.20. Vista general de la base de datos de *Liion* 96
4.21. Modelo navegación de las funcionalidades principales 97
4.22. Diagrama de flujo para el registro de usuarios. 98
4.23. Diagrama de flujo para el ingreso a la plataforma. 99
4.24. Diagrama de flujo para el alta perfil conductor. 100
4.25. Diagrama de flujo para la creación de viajes. 101
4.26. Diagrama de flujo para la búsqueda de viajes. 102
4.27. Gestos de viajes general para los usuarios de la aplicación. 103
4.28. Diagrama de flujo del proceso de inicio de viaje. 104
4.29. Etapas del itinerario para la vista del conductor. 105
4.30. Etapas del itinerario para la vista del pasajero. 106

5.1. Vista del ingreso a plataforma y recuperación de cuenta. 108

5.2. Vistas del registro de usuario. 109
5.3. Vista del alta de conductor. 110
5.4. Vista de la creación de viajes. 111
5.5. Vista de la búsqueda de viajes. 112
5.6. Gestor de viajes desde la perspectiva conductor. 113
5.7. Gestor de viajes desde la perspectiva pasajero. 114
5.8. Viaje en proceso desde la perspectiva pasajero. 115
5.9. Viaje en proceso desde la perspectiva pasajero. 116

Índice de cuadros

1.1. Tiempos promedios de traslado para llegar a lugares de trabajo, según modo de transporte a nivel país [2]. 13

2.1. Resultados a la pregunta "¿A través de que medios viajas?" 30
2.2. Resultados a la pregunta "¿Por qué razón no utiliza su automóvil para realizar viajes de larga distancia?" 31

3.1. Trayectos de interés 38
3.2. Comisión por viaje 38
3.3. Proyección de viajes por uso de la aplicación 39
3.4. Proyección de ingresos bajo escenario pesimista en UF 39
3.5. Cantidad de viajes estimados por trayecto, considerando escenarios con/sin pandemia 40
3.6. Ganancias presupuestadas con medidas por Covid-19 41

4.1. Tabla de posición de los *framework* de *frontend* (tamaño de las comunidades) 54
4.2. Tabla de posición de los *framework* de *frontend* (rendimiento) 55
4.3. Encuesta tomada a los desarrolladores sobre su conocimiento de los lenguajes de programación para *frontend* 56
4.4. Tabla de posición de los *framework* de *frontend* (Costo de aprendizaje) 58

4.5. Tabla de posición de los Framework de *backend* (tamaño de las comunidades) 61
4.6. Tabla de *benchmark* para *framework* de *backend* 62
4.7. Encuesta tomada a los desarrolladores sobre su conocimiento de los lenguajes de programación para *backend* 63
4.8. Tabla de posición de los *framework* de *frontend* (Costo de aprendizaje) 63
4.9. Tabla de flexibilidad de bases de datos 68
4.10. Tabla de resultados de flexibilidad de bases de datos 68
4.11. Las 20 bases de datos más utilizadas. 69
4.12. Posición final de las bases de datos en comunidad 70
4.13. Puntajes según experiencia previa y posición final. 71
4.14. Posición final de los tipos de bases de datos según el caso de uso de Liion 71
4.15. Puntajes para cada tipo de bases de datos para Liion 72
4.16. *Endpoints* de métodos *GET* 84
4.17. *Endpoints* de métodos *POST* 85
4.18. *Endpoints* de métodos *DELETE* 85
4.19. *Endpoints* de métodos *PATCH* 86
4.20. *Endpoints* de métodos *PUT* 86
4.21. Entidad users 89
4.22. Sub entidad driverData 90
4.23. Entidad travels 92
4.24. Sub entidad destinationDetails y originDetails 93
4.25. Sub entidad extraBaggage 93
4.26. Sub entidad itinerary 94
4.27. Sub sub entidad address_components 94
4.28. Sub entidad requestTravel 95

6.1. Importancia de características de la aplicación, desde el punto de vista del conductor. 147
6.2. Pregunta sobre importancia de características de la aplicación, desde el punto de vista del pasajero. 149
6.3. Pregunta sobre percepción femenina de los viajes compartidos 151

Índice general

1. Introducción **7**

1.1. Contexto 9

1.1.1. Transporte interurbano en Chile 9

1.1.2. Parque automotor 9

1.1.3. Contaminación vehículos motorizados en Chile 10

1.2. Problema y solución 12

1.2.1. Problemas del sistema actual 12

1.2.2. Solución propuesta 16

1.3. Objetivos 17

1.3.1. Objetivo general 17

1.3.2. Objetivos específicos 17

2. Marco Teórico **19**

2.1. Carpool 21

2.1.1. Beneficios 22

2.1.2. Inconvenientes 23

2.2. Estado del arte 24

2.2.1. Carpool en Chile y el mundo 25

2.3. Medidas y restricciones al *carpooling* por el virus SARS-CoV-2 en el mundo 26

2.4. Análisis de encuesta sobre la percepción de los viajes compartidos . . . 29

3. Plan de Negocios **35**

3.1. Propuesta de valor 35

3.2. Segmento de potenciales clientes 36

3.3. Fuente de ingreso 37

3.4. Impacto de la pandemia COVID-19 en la movilidad y el *carpool* 40

3.5. Inserción en el mercado 42

4. Desarrollo de la plataforma **43**

4.1. Arquitectura de tres capas 44

4.2. Elección de tecnologías para el desarrollo de la solución 46

4.2.1. Capa de presentación 46

4.2.2. Capa de negocio 58

4.2.3. Capa de datos o de persistencia 64

4.2.4. Servicios externos y datos para el consumo 72

4.2.5. Entorno de desarrollo 74

4.3. Arquitectura del sistema y resumen de tecnologías a ocupar 75

4.4. Análisis de requerimientos 76

4.4.1. Requerimientos funcionales 76

4.4.2. Requerimientos no funcionales 78

4.5. Mínimo producto viable 80

4.6. Capa de negocio 81

4.6.1. Estructura de la API 81

4.6.2. Desarrollo de API de servicios 82

4.6.3. Modelo de datos 87

4.7. Capa de presentación 97

4.7.1. Flujos de navegación y/o casos de uso 98

5. Resultados **107**

5.1. Acceso a la plataforma 108

5.2. Registro de usuario 109

5.3. Alta de perfil de conductor 110
5.4. Creación de viaje 111
5.5. Búsqueda de viajes 112
5.6. Gestor de viajes 113
5.7. Viaje en proceso 115
5.7.1. Conductor 115
5.7.2. Pasajero 116

6. Conclusiones **117**
6.1. Trabajos futuros 120

Anexos **158**

Capítulo 1

Introducción

Trabajo, estudio, visitas familiares o bien viajes de placer son algunas de las razones por las cuales, las personas realizan viajes interurbanos, donde en su mayoría se realizan por tierra, y suelen hacerse mediante, vehículos particulares, trenes o buses interurbanos. Cada uno de estos medios de transporte tiene sus ventajas y desventajas, sin embargo, existe una clara diferencia entre el transporte vehicular particular y el público. En el primero se obtienen ventajas como: rapidez, comodidad, seguridad y un mayor control sobre las medidas sanitarias. Pero también se tienen algunas desventajas, como el precio del viaje, o el hecho que el promedio de pasajeros por vehículo sea menor a la capacidad total de los mismos, lo que a su vez genera un aumento en la congestión vehicular por la cantidad de vehículos en circulación. En concordancia con lo anterior, también se tiene una mayor huella de carbono, tanto a nivel total como a individual, debido al material particulado per cápita. El transporte público interurbano resuelve las grandes desventajas del transporte particular; sin embargo, esto a un costo significativo, lentitud de un viaje, inseguridad e higiene son algunos de ellos. Fuera de estos problemas intrínsecos a la naturaleza del transporte público. También hay que tener en consideración que el mercado posee una serie de vicios, así como el oligopolio que poseen las empresas de buses sobre algunas rutas de transporte, derivando por ejemplo en un aumento arbitrario de los precios en ciertas temporadas.

Con estos antecedentes, ¿Se pueden atacar los problemas intrínsecos a ambos medios de transporte, y al mismo tiempo los abusos a los viajeros? La respuesta pareciera ir de la mano con el concepto del viaje compartido, el cual a grandes rasgos consiste en compartir un vehículo personal, con el fin de abaratar costos durante el viaje. Este modelo ya se ha probado en países Europeos, Norteamericanos, y también en países más cercanos culturalmente a Chile, como lo son Brasil o Argentina, siendo adoptada por la ciudadanía.

Actualmente, en nuestro país, ya existen aplicaciones exitosas que permiten coordinar viajes en modo taxi, como lo son *Uber, Cabify, DiDi, entre otros*. Por lo que posee sentido plantearse la idea de desarrollar una plataforma que haga uso de la tecnología para coordinar viajes bajo la premisa del viaje compartido, modernizando el transporte interurbano de pasajeros, visualizando y coordinando los viajes a través de un dispositivo móvil e incorporando funcionalidades tales como: la creación y búsqueda de viajes basándose en parámetros como el género, la coordinación y ejecución de viajes y la puntuación cruzada entre los usuarios, junto con todas las medidas de seguridad propias de una plataforma informática.

En tal sentido, dentro del marco del programa de Memorias Multidisciplinarias, se conforma un equipo técnico de dos estudiantes de Ingeniería Civil Telemática, Carlos Elgueta y Claudio Campuzano, y se propone, diseña y desarrolla una plataforma de viajes compartidos como respuesta al desafío planteado: “¿Cómo podemos mejorar el transporte interurbano utilizando las tecnologías de información y comunicación?”

1.1. Contexto

1.1.1. Transporte interurbano en Chile

Un servicio de transporte considerado interurbano es aquel en donde los viajes realizados deban superar los 200 [km] de recorrido o en caso contrario, ser trayectos que unan la ciudad de Santiago con localidades o ciudades costeras ubicadas en la región de Valparaíso [3]. Actualmente, en Chile existen alrededor de 193 empresas que ofrecen servicios de transporte de pasajeros a nivel interurbano (público y privado).

Respecto a las tarifas de los pasajes, tanto para servicios rurales como interurbanos, estás son determinadas de manera autónoma por las mismas empresas de transporte, las que tienen la obligación de informar de dichos valores con debida antelación ante la secretaria regional Ministerial de transporte y telecomunicaciones. Es importante destacar que la subsecretaria de transportes no tiene facultades en cuanto a la regulación de los valores tarifarios de estos servicios.

1.1.2. Parque automotor

Durante el año 2018, el parque automotor en Chile se constituye por 5.498.895 unidades, de las cuales 97,9 % corresponden a vehículos motorizados y un 2,1 % a no motorizados. La cantidad de vehículos motorizados aumenta en un 6,0 % respecto al año 2017; mientras que los vehículos no motorizados experimentan un crecimiento en 5.305 unidades (4,8 % más). Los medios de transporte utilizados de manera particular aumentan en un 6,2 %, los cuales representan un 90,5 % de participación del parque vehicular; por su parte, el transporte de carga y colectivo abarcaron el 5,7 % y 3,8 % respectivamente.

Es importante destacar que en cuanto a la cantidad de vehículos existentes por zonas; la región Metropolitana lidera con un 39,2 %, seguida de la región de Valparaíso

(10,6 %) y la región del Biobío (8,1 %). Por el contrario, las menores participaciones se registran en Aysén (0,8 %), Magallanes y Antártica Chilena (1,4 %). [4].

Adicionalmente, según cifras de la Asociación Nacional Automotriz de Chile en su "Informe del Mercado Automotor del mes de abril de 2019", se indica que en el primer cuatrimestre de 2019 se comercializaron 127.370 unidades nuevas de vehículos livianos y medianos, siendo esta la cifra más alta de la industria desde que se llevan estadísticas, con excepción del año 2018. A nivel regional, se informa que las regiones con más unidades nuevas comercializadas son: región Metropolitana con 77.962 unidades, región de Valparaíso con 11.669 unidades, región del Biobío con 8.006 unidades y por último región Los Lagos con 4.902 unidades, manteniendo una relación de proporcionalidad con los porcentajes de participación nombrados anteriormente. Asimismo, la comercialización de buses durante enero y abril de 2019 fue de 1.531 buses nuevos, lo que representa un aumento acumulado de 59,3 % respecto al mismo periodo del año 2018.

1.1.3. Contaminación vehículos motorizados en Chile

La Organización Mundial de la Salud (OMS) estima que una de cada nueve muertes en todo el mundo es el resultado de condiciones relacionadas con la contaminación atmosférica[5]. En muchos países existen diversas iniciativas para controlar la contaminación generada por el tráfico vehicular, por ejemplo, en Europa se instaura la 'Norma Euro 5'; que establece los requisitos técnicos para la homologación de los vehículos en relación con las emisiones contaminantes. En septiembre del 2013, Chile también adopta esta norma, la cual establece que los automóviles a diésel nuevos (livianos y medianos), deben ser homologados por esta nueva norma. Durante el año 2020 se adopta la norma Euro 6 [6]. Ambas buscan limitar las emisiones de ciertos gases contaminantes que emiten los vehículos, en particular la emisión de partículas y óxidos de nitrógeno (NOx) [7].

En la figura 1.1 se visualizan las emisiones contaminantes generadas por transporte motorizado entre los años 2005 a 2017 [8]. En específico, para el caso de los vehículos particulares, las cantidades de dióxido de carbono (CO_2) emitidas en el año 2017 fueron 6.009.001,61 [toneladas], donde 638,18[toneladas] corresponden a material particulado y 17.455,10 [toneladas] a óxidos de nitrógeno (NO).

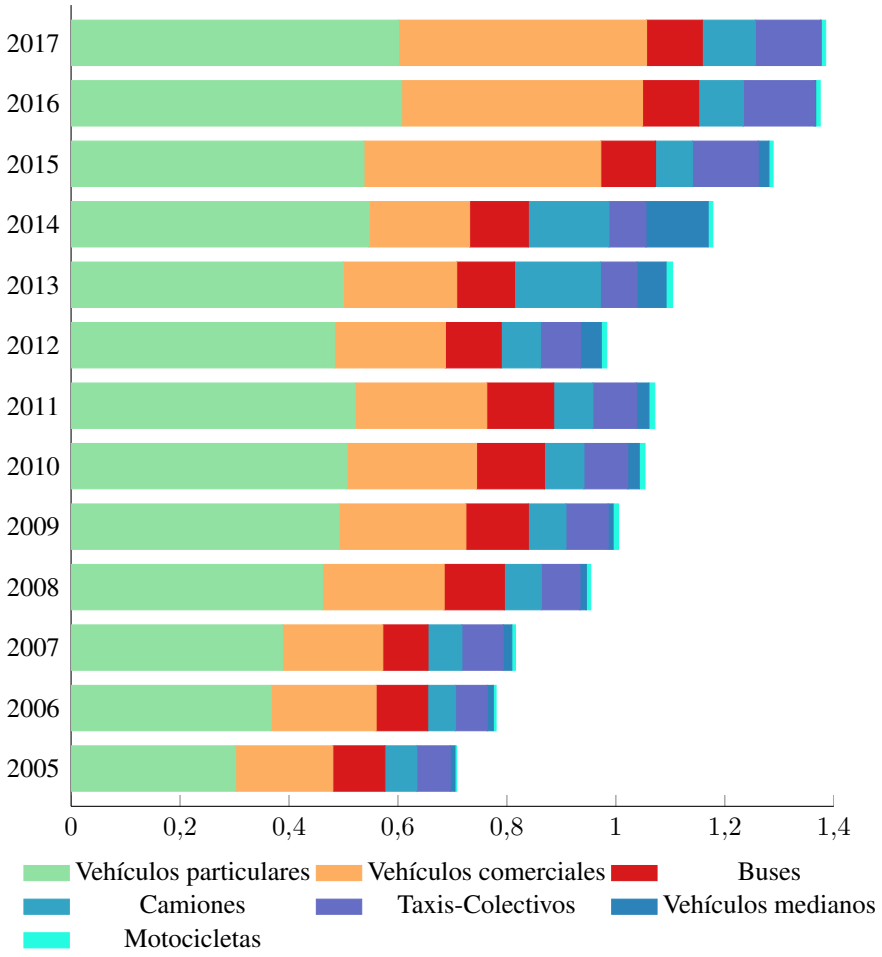


Figura 1.1: Mega-toneladas de emisiones de Dióxido de Carbono (CO_2) generadas por transporte en ruta, según categoría vehicular, entre 2005-2017

1.2. Problema y solución

1.2.1. Problemas del sistema actual

Los problemas del transporte en Chile no son sencillos, ni dependen de un solo factor, es mas, estos obedecen a múltiples variables, que en algunos casos están relacionados entre sí, sin embargo; a continuación, se listan y describen los más importantes.
Por un lado, para transporte interurbano en buses, se tiene:

- Lentitud: Realizar un viaje en bus es por naturaleza lento en comparación a un automóvil. Esta tesis se puede afirmar gracias a lo expuesto por la encuesta CASEN [2], desde donde se desprende el cuadro 1.1 que muestra como un viaje en transporte público presenta mayores tiempos de traslados absolutos (no relativos a la distancia) en comparación a cualquier otro medio de locomoción a motor. A esto se suma el análisis hecho por la CEP [9] que expone como dependiendo de los tamaños de las ciudades estas brechas varían. En adición a esto, se suma el hecho que dependiendo de la distancia que se tenga que recorrer hasta un punto de subida o bajada (como terminales o lugares en el trazado del recorrido), el usuario podría verse en el caso de abordar otras locomociones, lo que aumentara aún más tiempos de viaje.

Tipo de transporte	**Tiempo de desplazamiento [min]**
Transporte publicó	50
A pie	16
Vehículo particular	*34*
Bicicleta	*24*
Otro	*44*

Cuadro 1.1: Tiempos promedios de traslado para llegar a lugares de trabajo, según modo de transporte a nivel país [2].

- Inseguridad: Este corresponde a uno de los puntos más débiles del transporte público. Es evidente que la exposición en un bus donde hay, por ejemplo: veinte, cuarenta o más pasajeros es mayor a la que ocurre en un vehículo particular, lo que en consecuencia aumenta la percepción de inseguridad. Esto se ratifica en lo expuesto por la encuesta nacional urbana de seguridad ciudadana (ENUSC) [10] desde la cual se rescata el gráfico de "barra de error" 1.2, en donde es apreciable que aproximadamente el 52 % de las personas, se sienten inseguras en el transporte público. Es importante destacar además que un 59.3 % de los encuestados corresponden a personas identificadas como mujeres.

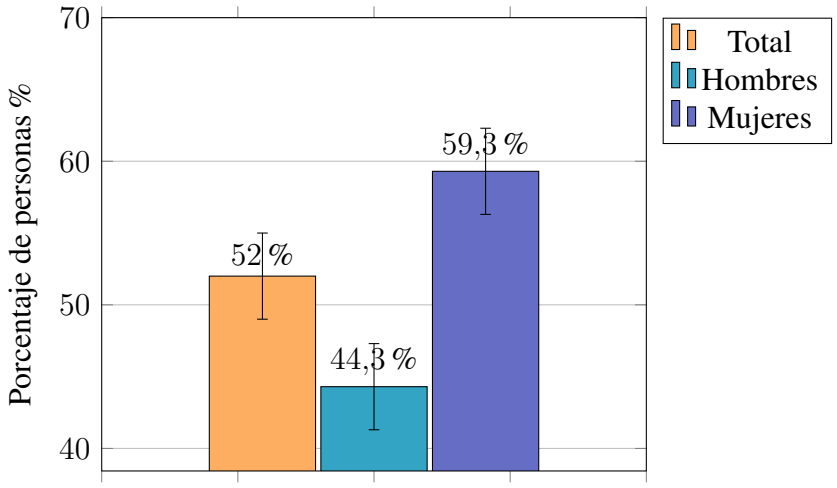


Figura 1.2: Percepción de inseguridad en los microbuses en los últimos 12 meses ENUSC 2018.

- Incomodidad: El viaje en bus no siempre garantiza comodidad, de hecho, en muchos casos depende de la flota que la empresa disponga en ese momento. La comodidad del viaje es un tema complejo, ya que son variadas las características que dan una experiencia de confort y más aún, considerando que la experiencia de viaje es personal y, por lo tanto, subjetiva, sin embargo; dentro de estas se han identificado las más importantes como: el espacio entre asientos, ya que limita el margen de movilidad de los pasajeros; la climatización, que solo puede ser regulada a discreción del conductor sin posibilidad de exigir algún cambio; control sobre artículos de entretención como radio o televisión; o bien iluminación del vehículo.

- Imposibilidad de personalizar viajes: Un bus vende un paquete, al cual el pasajero debe adecuarse a su ruta, eliminando la posibilidad de personalización.

Por otro lado, para quienes se movilizan en vehículo particular se tiene:

- Gastos relacionados con el viaje: El gasto promedio de un viaje desde la quinta región hacia Santiago en un vehículo que rinde 10 [km/L], esto en peso chileno, ronda aproximadamente los $16.500 (Según la aplicación Carretera [1], lo cual

es apreciable en la figura 1.3). Un costo alto considerando que, si se viaja en bus, el pasaje ronda los $4.000.

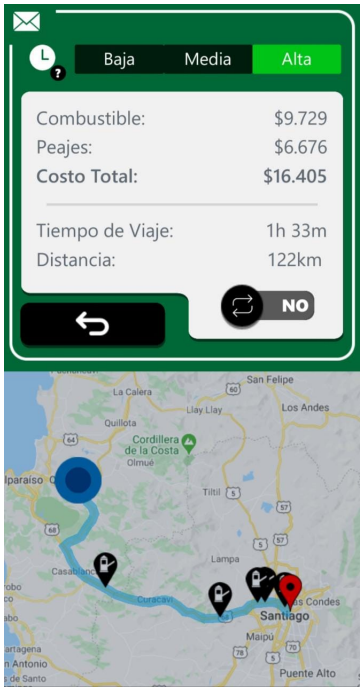


Figura 1.3: Costo de un viaje a Viña del Mar Santiago según la aplicación Carretera [1]

- Congestión Vehicular: Propia de las grandes urbes, donde a medida que aumenta la población, aumenta el parque automotor, lo que conlleva, en muchos casos, al colapso de calles en horarios de alto flujo, como las horas donde ocurren los desplazamientos desde y hacia el trabajo.

- Contaminación: Tal como se señala en la subsección anterior, la contaminación generada por los vehículos motorizados es alta. Esta se ve potenciada por el hecho que, en muchos casos, las personas viajan solas, provocando un aumento de la cantidad de vehículos en las vías y con ello mayor cantidad de gases contaminantes. Esto se ve reforzado en lo expuesto por Jens Borken-Kleefeld de IIASA, quien expone lo siguiente; “Viajar solo en un coche grande puede ser tan malo para el clima como volar, pero conducir con tres (otras personas) en un auto pequeño puede tener un impacto tan bajo como tomar un tren” [11].

1.2.2. Solución propuesta

En el contexto del transporte, ya es conocida la amplia gama de aplicaciones que ofrecen sus servicios, como por ejemplo: Uber, Cabify o Didi. Teniendo en consideración lo anterior, y dadas la serie de problemáticas presentadas en el apartado anterior, se propone en la presente memoria construir una aproximación similar, pero en el contexto del transporte interurbano en Chile, donde se abarquen las rutas principales, y también acerque la solución a las áreas rurales. Esto se lleva a cabo a través del diseño, desarrollo e implementación de una plataforma informática, la cual tiene como nombre comercial Liion, permitiendo a través de un 'smartphone' coordinar la realización de viajes de carácter compartidos "(Carpool/Carpooling)".

1.3. Objetivos

1.3.1. Objetivo general

Diseño y desarrollo de una solución que permita la coordinación y realización de viajes compartidos (desde ahora en adelante *carpool*) cuando estos son de carácter interurbano, salvaguardando la seguridad de sus usuarios por medio de diferentes elementos, y generando suficiente confianza que incentive su uso.

1.3.2. Objetivos específicos

- Investigar plataformas de viaje compartido en Chile y el mundo, junto con entender su impacto en el transporte interurbano.
- Estudio de factibilidad de una plataforma de viajes compartido en Chile, en tiempos normales y bajo normas sanitarias por COVID-19.
- Diseñar e implementar una solución que haga uso de las TIC en todas sus capas, desprendiéndose los siguientes puntos:
    - Definición de funcionalidades claves para la solución, junto con establecer cuáles constituyen el *MPV*.
    - Investigación y definición de las tecnologías a utilizar en la implementación de la solución, en concordancia con las funcionalidades establecidas.
    - Diseño e implementación de un *frontend* para la interacción de aplicación con los usuarios.
    - Desarrollo e implementación de un *backend* para la conexión de las distintas funcionalidades.

    - Desarrollo e implementación de los modelos de datos para el almacenamiento de información.
    - Creación de marca asociada a la solución, junto con nombre, logos, tipografía y paleta de colores.
    - Diseño de interfaces visuales de la aplicación, acorde a la imagen a transmitir para el proyecto, que además definirán el flujo de esta.

- Identificar fortalezas y debilidades en la aplicación, junto con posibles trabajos futuros.

Capítulo 2

Marco Teórico

Durante la Segunda Guerra Mundial, materiales como el caucho, la gasolina, o diversos metales empleados en la fabricación de vehículos, fueron afectados por políticas de ahorro, redistribución y reciclaje con el fin de ser utilizados para la fabricación de armas, prueba de esto son los afiches de la época como es apreciable en la figura 2.1 [12]. Esto hizo que los medios de transportes y los insumos para que estos pudieran funcionar (como la gasolina) escaseasen, por lo que fue necesario proveer de un método que permitiese a los trabajadores de los diversos rubros de la económica estadounidense, movilizarse a sus puestos de trabajo. En consecuencia de dichas políticas de la época, yace el nacimiento de la práctica de los viajes compartidos, o mejor conocido como *carpool*.



Figura 2.1: Publicidad incentivando el uso del *carpool* durante la Segunda Guerra Mundial

En sus orígenes, funcionaba mediante tarjetas de contacto en la oficina de la 'Defensa Civil', donde los participantes debían rellenar con su información personal y datos de los viajes a realizar, para luego ser contactados por otras personas que buscarían coincidencias que cumplan con sus criterios. Finalmente, el modelo pierde trascendencia al finalizar la guerra, sin embargo; vuelve a tomar fuerza con la crisis del petróleo estadounidense en 1973, popularizando nuevamente los viajes compartidos, estableciéndose servicios metropolitanos de viajes, donde la gente podía registrarse por correo postal o teléfono, para luego estos ser agrupados por voluntarios especializados, utilizando miniordenadores (con entrada de datos mediante tarjetas perforadas) para encontrar coincidencias. Ya en la década de los noventa, con el surgimiento de nuevas tecnologías, la práctica pudo automatizarse, a tal punto de no necesitar personal que estuviera encargado de hacer coincidir a conductores y pasajeros, mejorando los tiempos de respuesta [13].

En el presente capítulo se presentarán importantes antecedentes, investigaciones previas y consideraciones teóricas, que justifican las decisiones en los próximos apartados.

2.1. Carpool

Al día de hoy, el *carpool* es la práctica que consiste en compartir un vehículo para movilizarse por un trayecto previamente establecido, fijando de antemano los puntos de subida y bajada de cada pasajero, en donde el aporte a realizar, y las reglas son impuestas por el conductor.

El modelo consta de al menos dos usuarios, un conductor, quien facilita su vehículo para llevar a otras personas en el transcurso de su viaje, desde o hacia su destino, y el pasajero, quien busca ser llevado. El objetivo principal de compartir un viaje es disminuir los costos y optimizar los beneficios de un viaje en vehículo particular. Para los conductores, esto significa dividir equitativamente los gastos de gasolina y peajes, que de una u otra manera igual deberá pagar, y para los pasajeros es desembolsar un precio similar al del transporte público, por un servicio que será más cómodo, rápido y personalizado. El uso de vehículos compartidos es un modelo intrínsecamente no lucrativo que reúne a personas con necesidades de viaje similares para compartir los gastos. Los costos implicados son los propios del viaje, es decir, gastos en combustible y posibles peajes. Sin embargo, no considera los gastos asociados a mantención del vehículo, patente, seguros, permisos de circulación, entre otros. Esto sería considerado como una desventaja en un modelo en el cual el conductor espera tener utilidades.

Generalmente, el viaje compartido se da entre el círculo cercano al conductor, los que pueden ser familiares, amigos o compañeros de trabajo. Sin embargo, en la actualidad se han creado aplicaciones tecnológicas que permiten conectar a conductores con pasajeros que tengan el mismo destino, sin que necesariamente se conozcan. Todo esto va de la mano con medidas de seguridad en la aplicación, que permitan generar a los usuarios la suficiente confianza como para optar por los viajes con personas descono-

cidas.

2.1.1. Beneficios

Junto con la ventaja natural de generar un ahorro en conductores y pasajeros, al compartir los gastos totales del viaje, se destacan los siguientes beneficios:

- Reducción de la contaminación: En el análisis "Minimizing CO2 emissions in a practical daily carpooling"[14], se demuestra como al compartir viajes al trabajo todos los días, se logra una reducción de hasta el 28 % de las emisiones de CO_2.
- Reducción del consumo de combustibles: Según el estudio "The benefits of carpooling [15]", si en EE.UU añadiera tan solo un pasajero más por cada cierta cantidad de vehículos, la cifra de galones de combustibles ahorrados rondaría en cerca del millón.
- Reducción de la congestión vehicular y de la demanda de estacionamientos: Un medio de transporte implica tres elementos necesarios, sin los cuales no es posible que funcione:
  - El medio de transporte.
  - La vía de desplazamiento.
  - El terminal de llegada.

  En el caso del transporte vehicular, estos corresponden al vehículo, la carretera y un lugar para aparcar respectivamente.

  Un aumento de los vehículos en circulación implica, de manera natural, una mayor densidad de vehículos por las carreteras y una mayor cantidad de espacios necesarios para estacionar, por lo que si los pasajeros comparten viajes, se podría reducir la cantidad de vehículos en circulación y junto con esto, los porcentajes de ocupación en terminales (estacionamientos). Lo anterior es demostrado

en la investigación “Carpooling: A Step to Reduce Congestion [16]”, donde se demuestra como el uso del *carpool* ayuda a descongestionar las calles en Delhi, India.

- Disminución de los tiempos de viaje: Según estudios [17], en las grandes urbes, el transporte privado tarda en promedio 36 minutos, versus 53 minutos que toma el transporte público, como en el *carpool* se utilizan vehículos particulares; de forma natural se deduce que existiría una disminución en los tiempos de traslado.

Es importante tener en cuenta que la mayoría de estos beneficios, generaran un mayor impacto cuando el modelo de transporte se masifica, por eso es importante visualizar y publicitar todas las ventajas del *carpool*.

2.1.2. Inconvenientes

Si bien, los beneficios son varios, se listan los siguientes inconvenientes que podrían presentarse a los usuarios de *carpool*.

- Perdida de libertad: Al tener viajes ya agendados, recae sobre el usuario una cuota de responsabilidad por cumplir los tiempos y fechas de viajes acordados de antemano.

- Desconfianza: En el *carpool*, tanto el conductor como sus pasajeros, posiblemente no se conozcan, lo que puede generar cierto ambiente de inseguridad, más aun considerando el artículo de Bruno Cordova [18], donde se indica que solo un 12.4 % de los chilenos concuerda que es posible confiar en la mayoría de la gente.

Un inconveniente intrínseco del transporte de pasajeros, del que no escapa el *carpool* y el cual es importante mencionar por el contexto actual de pandemia, es el contagio de enfermedades virales. Dada la poca distancia que existe entre pasajeros, conlleva

a un aumento de las probabilidades de contagio, algo que debe ser tomando en cuenta, si es que se quiere tener un viaje sanitariamente seguro.

2.2. Estado del arte

La revisión bibliográfica se centra en la identificación de los diferentes proveedores de servicios de viajes compartidos, donde la mayor parte son utilizados en países de habla inglesa. Luego de la exploración de los diversos aplicativos, es posible notar como existen varios términos para definir el servicio de viaje compartido, entre los que se encuentran: "carpooling, ride-sharing, vanpooling, dynamic ride-sharing, ride-hailing y ride-pooling". Esto demuestra que no existe una definición completamente universal que defina un "viaje compartido". Es importante destacar que el término *ride-sharing* es utilizado en servicios con y sin ánimos de lucro, sin embargo, el presente estudio se centra en el *carpooling*.
Actualmente, las aplicaciones de *carpool* se implementan de dos maneras principales:

- **Plataformas abiertas al público general**: Las personas interesadas en realizar la actividad, pueden registrarse y buscar viajes que cumplan con sus criterios de búsqueda, pudiendo viajar con personas conocidas o desconocidas.

- **Modalidad cerrada**: Los aplicativos tienen un enfoque de comunidad (ya sea empresa, universidad u otra organización), la cual paga por el uso de la plataforma, permitiendo a sus colaboradores coordinar los viajes compartidos. Este último modelo es altamente utilizado en empresas y universidades en EEUU y Europa.

2.2.1. Carpool en Chile y el mundo

La filosofía del *carpool* en el extranjero lleva años funcionando, es por esta razón que internacionalmente existen diversas aplicaciones que ponen en juego el modelo, y a continuación, se listan las más populares:

- BlaBlaCar: Plataforma francesa la cual se origina en 2006 a través de una página web. Hoy en día es la mayor empresa dedicada al *carpool* en el mundo, con alrededor de 90 millones de usuarios en 22 países [19].

- UberPool: Plataforma desarrollada por Uber (empresa líder en *ridesharing*), la cual está destinada a realizar *carpooling* [20].

- Lyft: Plataforma parte del ecosistema de Lyft en donde se puede realizar *carpooling*, es exclusiva de Estados Unidos y posee una serie de servicios extra como modo *ridesharing*, arriendo de vehículos, bicicletas y scooters. Por contingencia de covid-19, no está disponible el servicio de *carpool* [21].

- Liftshare: Plataforma del Reino Unido dedicada al *carpooling* público y también para empresas [22].

- Waze Carpool: Plataforma de *carpooling* desarrollada por Waze (plataforma líder en su tipo). [23]

- Zimride: Plataforma que permite el *carpooling* a través de redes privadas como corporaciones, universidades, entre otros. Por contingencia de Covid-19 no está disponible el servicio de *Carpool* [24].

- Carma: Empresa la cual dentro de su gama de productos posee *carpooling*. Opera en Estados Unidos, y provee un sistema de verificación de ocupantes a través de su aplicación, con el objetivo optar a los descuentos por *High Occupancy Vehicle* u HOV [25].

En contra parte, en Chile el *carpoo* no ha sido una práctica masiva, el término es desconocido para el común de la gente. Reflejo de esto, es que no haya aplicaciones que sean conocidas como si lo son Uber o Cabify. A pesar de esto, existen plataformas que han permitido el *carpooling*, sin embargo; no poseen popularidad. A continuación, se listan algunas plataformas:

- Allride: Plataforma que permite realizar *carpooling* público y privado, vía convenios para corporaciones, que en 2019 se internacionalizó. [26]
- Karpool: Plataforma Chilena, que ofrece *carpooling*. [27]
- Nosfuimos: Plataforma web, para coordinar viajes compartidos, donde se puede realizar *carpooling* público como también en corporaciones. [28]

Es importante destacar que en Chile y en el mundo, existen grupos informales donde se practica en *carpool*, esto usualmente se hace a través de aplicaciones destinadas a usos más genéricos, como lo son las redes sociales (Facebook), o a través de aplicaciones de mensajería donde también es posible organizarse para un viaje (WhatsApp o Telegram).

2.3. Medidas y restricciones al *carpooling* por el virus SARS-CoV-2 en el mundo

La pandemia de Covid-19, causada por el virus SARS-CoV-2, surge a finales del 2019 y de forma progresiva afectó a todos los países del mundo. Debido a las medidas preventivas tomadas por los Gobiernos, una gran cantidad de servicios tuvieron actuar acorde a esto, incluyendo el transporte público y privado.

En el mundo, las aplicaciones de *carpool* han tomado diversas medidas para prevenir los contagios. Estas van de la mano con los reglamentos que los Gobiernos han impuesto. Empresas como Zimride [24] o Lyft [21] han dejado de funcionar por motivos de la pandemia.

Dentro de las grandes empresas que siguen operando a nivel internacional, y que han tomado medidas de prevención, se encuentran: Blablacar [29], Uber Pool [20] y Liftshare [30].
A continuación, se listan las medidas:

- Utilizar mascarilla en todo momento.
- Lavar o higienizar las manos antes de abordar.
- Mantener las ventanas abiertas del vehículo para una buena ventilación.
- Sentarse en los asientos traseros.
- Compartir vehículo con personas del círculo social del conductor; con un solo pasajero desconocido o bien pasajeros del mismo grupo social, que habiten en el mismo hogar.
- Desinfectar constantemente puertas y áreas de contacto común.

Internacionalmente, a pesar de todas las restricciones al transporte de pasajeros por normativas de prevención, BlaBlaCar solo registra una disminución del 30 % del uso de su servicio en 2020 en comparación a años anteriores, lo que da a entender que, a pesar de la pandemia, el *carpool* sigue siendo considerado una opción de transporte válida [31].

En cuando al panorama en Chile, Allride [26] y Nos Fuimos [28] no se refieren al tema en sus plataformas, en cambio; Karpool [27] ha implementado una sección especial dentro de su "guía de viajes seguros"[32]. En general las normativas son similares a las nombradas anteriormente, sin embargo, hay algunas diferencias clave, como:

- Máximo de 2 asientos disponibles por viaje, con el objetivo de respetar distanciamiento social, pudiendo ser pasajeros de distintos orígenes.
- Porte de salvoconducto o permiso de traslados, cuando el origen o destino sean comunas en etapas donde se limita el desplazamiento.

- No uso de dinero en efectivo, pagos solo a través de su monedero virtual.

2.4. Análisis de encuesta sobre la percepción de los viajes compartidos

Con el fin de validar las hipótesis que se tienen respecto a: la problemática del transporte interurbano, la solución propuesta y el perfil del cliente potencial, se realiza una encuesta, donde el universo de la muestra está compuesto por 112 personas, de las cuales 95 cumplen con el perfil de usuario objetivo, segmento al que apunta la solución propuesta por la aplicación Liion.

El tipo de muestreo en la encuesta se caracteriza por ser no probabilístico y no aleatorio, utilizado para crear muestras de acuerdo a la facilidad de acceso, la disponibilidad de las personas de formar parte de la muestra, o cualquier otra especificación práctica. El motivo para realizar este tipo de muestreo se debe a que permite recolectar respuestas de manera rápida y económica, permitiendo un análisis más expedito.

La primera hipótesis por validar corresponde a saber si existe un mercado objetivo al cual apuntar, con esta pregunta se espera saber si existen clientes. Esto se hace a través de la pregunta "¿A través de qué medios viajas?". Obteniéndose los resultados de la tabla 2.1, donde se aprecia que la mayoría de los encuestados utiliza bus interurbano y automóvil propio para realizar viajes de larga distancia, con un 46,09 % y 42,97 % respectivamente, dando un total de 89.06 %.

Medios de transporte utilizados	Frecuencia	Porcentaje
Bus interurbano	59	47.62 %
Automóvil propio	55	19.05 %
Avión	10	4.76 %
Tren	3	19.05 %
interurbano	1	9.52 %
Total	128	100 %

Cuadro 2.1: Resultados a la pregunta "¿A través de que medios viajas?"

Con el fin de poder segmentar a los encuestados entre quienes poseen auto (potenciales conductores) y quienes no (potenciales pasajeros), se realiza la pregunta "¿Posees automóvil propio?", obteniéndose los resultados de la figura 2.2. Donde de 95 personas, el 51.58 % posee automóvil propio.

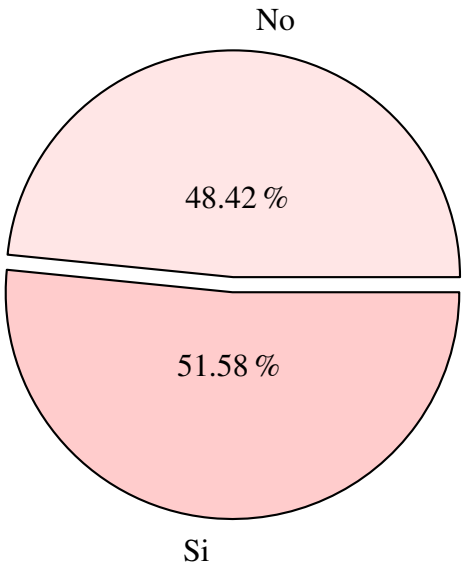


Figura 2.2: Resultados a la pregunta "¿Posees automóvil propio?"

A las personas que respondieron positivamente, se les hace otra pregunta, esta vez referente a si lo utilizan para realizar viajes de larga distancia. Dando como resultado lo apreciable en la figura 2.3, donde un 24.49 % de los 49 dueños de autos, no lo usan

para tales fines.

Luego de esto, es importante poder capturar por qué no lo ocupan para viajes de larga distancia, dando como resultado lo apreciable en el cuadro 2.2. Destacando de esta, que el 47,62 %, no lo usa para "evitar gastos de bencina, peaje y estacionamiento", problema principal al que ataca el *carpool*.

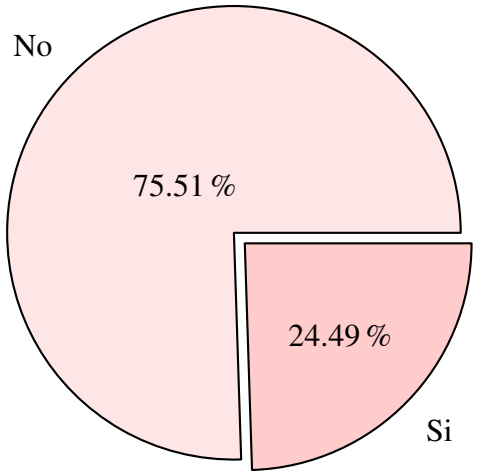


Figura 2.3: Resultados a la pregunta "¿Utilizas tu auto para realizar viajes de larga distancia"

¿Por qué razón no utiliza su automóvil para realizar viajes de larga distancia?	**Frecuencia**	**Porcentaje**
Para evitar gastos de bencina, peaje y estacionamiento	10	47.62 %
Para evitar congestión vehicular	4	19.05 %
Por falta de estacionamiento	1	4.76 %
Para no dañar el medio ambiente	4	19.05 %
No me gusta conducir	2	9.52 %
No me gusta viajar solo	0	0.00 %
Total	21	100 %

Cuadro 2.2: Resultados a la pregunta "¿Por qué razón no utiliza su automóvil para realizar viajes de larga distancia?"

La siguiente pregunta es sumamente importante, ya que de esta depende el funcio-

namiento y financiamiento de la aplicación, y consiste en conocer cuántos de aquellos que tienen automóvil propio, estarían dispuestos a compartirlo para realizar *carpooling*. Los resultados son apreciables en la figura 2.4 y estos dicen que un 59.18 % si estaría dispuesto versus un 40.82 % que no lo estaría. Esto valida hipótesis de que existirán conductores que ofrecerán sus vehículos para realizar viajes, e incluso este porcentaje podría aumentar, si a los conductores se les ofrece un incentivo como el aporte en monetario a su viaje (lo que le permite rebajar gastos basales del viaje).

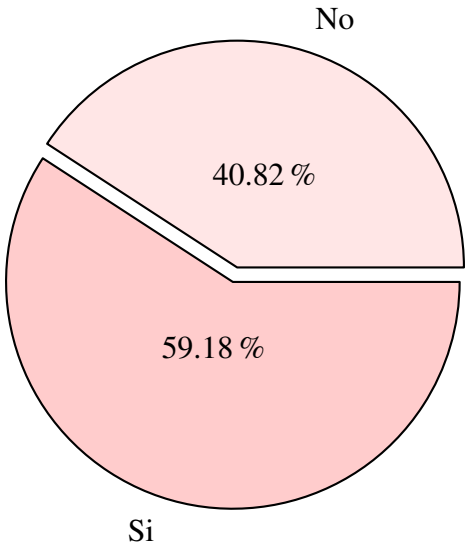


Figura 2.4: Resultados a la pregunta "¿Estarías dispuesto a compartir tu automóvil para hacer viajes compartidos de larga distancia?"

Por el lado del pasajero, es importante conocer si estos están dispuestos a compartir un viaje con personas desconocidas, es por esto que a encuestados "no conductores" se les realiza dicha pregunta obteniéndose los resultados vistos en la figura 2.5. Donde es posible apreciar que la proporción de personas que estarían dispuestas a viajar como pasajeros equivale al 71,58 % de las respuestas (95 en total). Con este porcentaje se presume que habrán pasajeros dispuestos a viajar, obviamente si es que se toman las decisiones correctas desde el punto de vista, del diseño, seguridad y confianza que tenga la aplicación.

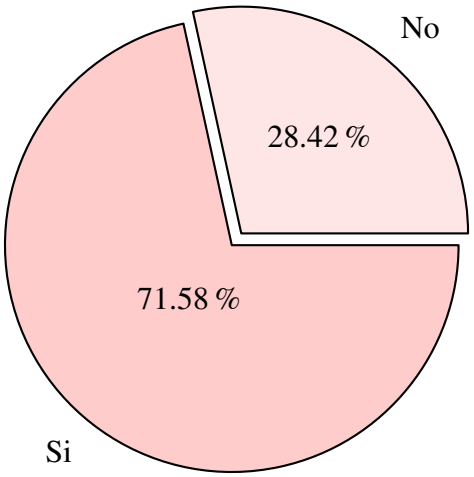


Figura 2.5: Resultados a la pregunta "¿Disposición a compartir como pasajero el automóvil?"

Finalmente, luego de mostrarles diferentes características que tendría la aplicación Liion, entre las que se encuentran:

- Poder compartir un viaje con personas desconocidas, con el fin de abaratar costos de traslados, con todas las medidas de seguridad que se podrían llegar a brindar (filtrado de usuarios por medio del feedback de la comunidad, entre otros).
- Posibilidad de pagar con múltiples medios de pago, como por ejemplo débito, crédito, o tarjetas prepago.
- Que se pueda abordar y descender de los vehículos cerca de un punto en específico.
- Disposición de medidas de seguridad en casos extremos, como lo es la implementación de un botón de seguridad una vez el viaje haya iniciado.

Se les pregunta a todos los encuestados (tanto pasajeros como conductores), si estarían interesados en usar Liion. Obteniendo los resultados de la figura 2.6, es importante notar que un 72.63 % de los encuestados si estarían dispuestos a utilizarla (de un total

de 95), lo que da la suficiente confianza para asegurar que hay un nicho tecnológico no lo suficiente cubierto.

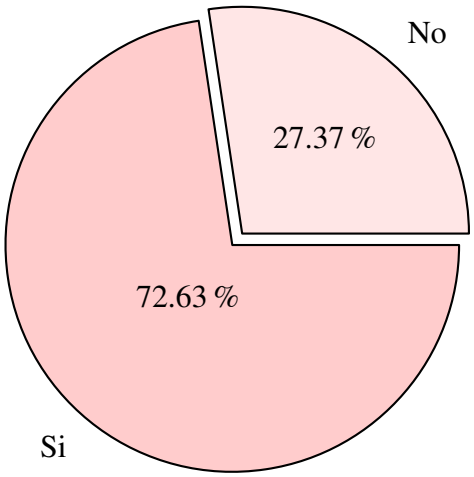


Figura 2.6: Resultados a la pregunta "¿Estarías dispuesto a usar Liion?"

En consecuencia de lo expuesto durante esta sección, se puede concluir que en Chile una aplicación que permita realizar *carpooling* es una idea que podría ser adoptada por la población a nivel general y, por lo tanto, es una filosofía que tiene cabida y es viable desarrollar.

Capítulo 3

Plan de Negocios

3.1. Propuesta de valor

Liion es una plataforma multilateral de economía colaborativa, en donde los usuarios (conductores y pasajeros) que tienen necesidades de viajes similares, como usar rutas en común, acceden a los beneficios de compartir viajes. Por una parte, a los conductores les permite ahorrar costos como: bencina, peaje o estacionamiento, y, por otra parte, a los pasajeros les permite viajar cómodamente, en un menor tiempo de viaje, a diferencia de uno realizado en bus interurbano.

Las ventajas de utilizar Liion es que los viajes se pueden realizar a conveniencia de los usuarios, es decir, el día, la hora, el lugar y según las necesidades que tengan, ya que la plataforma permite la comunicación entre conductores y pasajeros, mediante un chat grupal.

La aplicación Liion entregará un servicio seguro y amigable, utilizando para ello elementos como: botones de emergencia, en caso de una situación de auxilio para los usuarios; sistemas de pagos confiables para los pasajeros; sistema de puntuación y comentarios que los usuarios realizan según sus experiencias; nivel de completitud del perfil de los usuarios registrados; confirmación de identidad a través de un escáner de un código QR donde los usuarios pueden validar los datos entre ellos.

Liion posee además dos grandes elementos diferenciadores respecto a su competencia directa, estos son, en primer lugar; que Liion permite al conductor ir a buscar a un pasajero a su lugar de origen, operando como aplicación taxi durante ese trayecto y, en segundo lugar; permitir a los usuarios pasajeros proponer viajes, que son aceptados por los conductores en caso de que sean de su interés.

Finalmente, otros beneficios asociados al uso de los viajes compartidos gracias a Liion, son que: ayuda a reducir la congestión vehicular, sobre todo en rutas muy concurridas, como por ejemplo Valparaíso, Santiago; y en consecuencia reducir las emisiones en CO2, NOx y otros tipos de emisiones contaminantes.

3.2. Segmento de potenciales clientes

El segmento de clientes está conformado por dos tipos de usuarios que interactúan entre sí mediante la aplicación. En primer lugar, se encuentran a los conductores, que son hombres y mujeres entre 18 y 40 años, de ingresos medios y su principal característica es que posean automóvil propio y realicen viajes de larga distancia frecuentemente, ya sea por motivos de trabajo, estudio u otro. Este grupo de usuarios se caracteriza por viajar solos en sus automóviles particulares, y desean rebajar sus costos de viaje de bencina y peajes. Estos usuarios también utilizan o han utilizado al menos una vez aplicaciones digitales de transporte y estarían dispuestos a compartir su automóvil con otras personas para realizar viajes de largas distancias. En segundo lugar, están los pasajeros, que son hombres y mujeres, que tienen alrededor de 18 y 35 años, de ingresos medios y que pueden o no poseer automóvil propio, pero prefieren utilizar otros medios de transporte como buses interurbanos. Además, también se caracterizan por realizar viajes frecuentes de mediana o larga distancia, por razones de trabajo, estudio u otro. Además, estos usuarios utilizan o han utilizado aplicaciones digitales de transporte y estarían dispuestas a utilizar una aplicación que les permita viajar largas distancias en un viaje compartido.

Cabe destacar, que el cliente de la aplicación es el pasajero, ya que es de quien se

obtienen los ingresos, a través del cobro de una comisión por asiento disponible, el cual es publicado por conductores en la plataforma.

3.3. Fuente de ingreso

El estudio a continuación contempla datos utilizados durante la operación normal del país, hasta antes de la pandemia Covid-19, por lo tanto, los datos son válidos dentro de ese contexto.

La fuente de ingresos de la aplicación Liion proviene del cobro de una comisión sobre el valor del asiento que pagan los pasajeros. Este porcentaje de cobro equivale al 5 % sobre el costo por el asiento disponible que fija el conductor con relación al costo total de un viaje. En concreto, la aplicación va a sugerir un costo total aproximado del viaje, que incluye gasto en bencina y peajes, y que luego el conductor podrá asignar un cobro por los asientos que tenga disponible en su automóvil.

Para el cálculo de los ingresos se utilizan los datos del estudio "Análisis y desarrollo del sistema de transporte interurbano, Macrozona Centro-Norte" de la Dirección de Investigaciones Científicas y Tecnológicas de la Pontificia Universidad Católica de Chile, MIDEPLAN y la Secretaría Interministerial de Planificación y Transporte (Sectra), del cual se utilizaron la matriz de valores sobre viajes interregionales de algunos tramos de interés para el proyecto.

Uno de los supuestos utilizados es que los trayectos que tienen una duración menor a 8[h]. A pesar de que la aplicación utiliza viajes de larga distancia, los datos se estiman con base en la duración en trayectos similares entre Valparaíso y Santiago, la cual es la ruta más recurrente del país. Los trayectos de interés se presentan en la tabla 3.1.

Trayecto viceversa	Horas estimadas de viaje
Iquique-Antofagasta	4h 59
Antofagasta-Copiapó	6h 17m
Copiapó-La Serena	3h 54m
La Serena-Valparaíso	4h 44m
La Serena-RM	5h 0m
La Serena-Rancagua	5h 55m
Valparaíso-Rancagua	2h 14m
Valparaíso-RM	1h 25m

Cuadro 3.1: Trayectos de interés

Adicionalmente, se considera que hay ocupación máxima en los automóviles por cada viaje de la estimación. De acuerdo con esto, se realiza el cálculo de la comisión basándose en el costo total del viaje por cada trayecto de interés para el proyecto, obteniendo lo mostrado en la tabla 3.2.

Trayectos viceversa	Costo promedio de (CLP)	Comisión por viaje (5 %) (CLP)
Iquique-Antofagasta	$ 33.500	$ 1.675
Antofagasta-Copiapó	$ 45.800	$ 2.290
Copiapó-La Serena	$ 36.400	$ 1.820
La Serena-Valparaíso	$ 49.500	$ 2.475
La Serena-RM	$ 52.950	$2.648
La Serena-Rancagua	$ 71.200	$ 3.560
Valparaíso-Rancagua	$ 24.100	$ 1.205
Valparaíso-RM	$ 15.000	$ 750

Cuadro 3.2: Comisión por viaje

Se determina que bajo un escenario pesimista, la aplicación será usada en un 0,5 % de los viajes anuales realizados en 2020 para los trayectos entre regiones, y de un 1 %

en el tramo Valparaíso-Santiago. Sumado a lo anterior, la tasa de crecimiento de los viajes está relacionada con la evolución anual del porcentaje de usuarios en plataformas digitales de transporte durante el año 2019, que equivale a un 9,1 %.
En tabla 3.3, es posible apreciar la evolución de la cantidad de viajes, desde que la aplicación es publicada.

Trayecto vice versa	**Año 1**	**Año 2**	**Año 3**	**Año 4**	**Año 5**
Iquique-Antofagasta	4.738	5.169	5.640	6.153	6.713
Antofagasta-Copiapó	465	507	553	604	659
Copiapó-La Serena	3.045	3.322	3.624	3.954	4.314
La Serena-Valparaíso	5.315	5.798	6.326	6.901	7.529
La Serena-RM	327	356	389	424	463
La Serena-Rancagua	6.995	7.632	8.326	9.084	9.910
Valparaíso-Rancagua	4.765	5.199	5.672	6.188	6.752
Valparaíso-RM	336.456	367.073	400.477	436.921	476.680

Cuadro 3.3: Proyección de viajes por uso de la aplicación

Tomando los factores anteriores, es posible proyectar los ingresos y ganancias que tendrá la aplicación Liion, los cuales son apreciables en la tabla 3.4, donde se denota la rentabilidad desde el año uno.

Ingresos anuales	**Año 1**	**Año 2**	**Año 3**	**Año 4**	**Año 5**
Valor en UF	10.798,93	11.781,64	12.853,76	14.023,46	15.299,59
Utilidades en UF	4.002,23	4.640,45	5.288,97	6.014,34	149.868,86

Cuadro 3.4: Proyección de ingresos bajo escenario pesimista en UF

3.4. Impacto de la pandemia COVID-19 en la movilidad y el *carpool*

Mientras menor sean el número de personas reunidas en un lugar, menor será la probabilidad de que el virus SARS-CoV-2 se transmita. Por lo tanto, para evitar reuniones, una regla básica del combate contra la pandemia, consiste en reducir la movilidad de la población. Para esto se implementan normas sanitarias que restringen la movilidad, como por ejemplo las cuarentenas. Es por esta razón que en Chile durante el 2020 se implementaron políticas públicas con el objetivo de reducir el desplazamiento, logrando una disminución en un 43 % (un 57 % del total versus año anterior). Esto según el informe de movilidad de proporcionado Google [33].
En la tabla 3.5, se muestra esta reducción:

Trayectos (Viceversa)	**Viajes estimados sin pandemia**	**Viajes estimado con pandemia**
Iquique - Antofagasta	947.596	540.129
Antofagasta - Copiapó	92.965	52.990
Copiapó - La Serena	608.946	347.099
La Serena - Valparaíso	1.062.908	605.857
La Serena - RM	65.325	37.235
La Serena - Rancagua	1.398.992	1.398.992
Valparaíso - Rancagua	953.088	797.425
Valparaíso - RM	33.645.600	19.177.992

Cuadro 3.5: Cantidad de viajes estimados por trayecto, considerando escenarios con/sin pandemia

El estudio económico original considera tener a cuatro pasajeros viajando al mismo tiempo; sin embargo, como se explica en la sección 2.3, en los casos donde el Carpooling continúa operando, la cantidad de pasajeros se reduce a un máximo de dos. Considerando esto y lo anterior, en la tabla 3.6 se visualizan los ingresos estimados,

bajo estos nuevos supuestos.

	Año 1	**Año**	**Año 3**	**Año 4**	**Año 5**
Valores en UF	3.077,7	3.357,8	3.663,3	3.996,7	4.360,4
Utilidades en UF	-1.070,62	-894,03	-749,15	-573,25	-7.223,48

Cuadro 3.6: Ganancias presupuestadas con medidas por Covid-19

Basándose en estos resultados, se puede concluir que en un estado de pandemia, no es rentable lanzar al mercado una aplicación de Carpool como lo es Liion, posponiendo el lanzamiento hasta que las medidas de restricción se flexibilicen o terminen. Esto coincide con lo publicado por la empresa chilena Karpool [27] en su plataforma durante el 2020, quienes aseguran que durante la pandemia, su aplicación no ha sido rentable, extracto:

"En un contexto en donde marcas de todo tipo están en plan de reinventarse y así sobrevivir a la crisis, la economía colaborativa ha debido buscar alternativas para operar con cierta normalidad durante la pandemia. Si bien el transporte de personas no está siendo totalmente rentable, debido a las restricciones sanitarias a lo largo del país, Karpool ha decidido seguir activa y ayudando a las personas que con bases responsables necesitan recorrer en país." Karpool, 2020 [32].

Además de lo anterior, existe un tercer factor, el cual no se ha tomado en consideración en los cálculos previos, que tiene el potencial de a aumentar más las posibles pérdidas. Este corresponde a la disposición que tendrán las personas en viajar en un vehículo junto con desconocidos o compartir el vehículo propio.

El cálculo hasta aquí corresponden a una cifras sin considerar los efectos de la pandemia, por lo cual su efecto solo puede empeorar las cifras y en el mejor de los casos dejarlas como se proponen en la tabla 3.6.

3.5. Inserción en el mercado

Los canales de distribución de la aplicación, es decir, los lugares donde Liion podrá ser adquirida y descargada por los usuarios, corresponden a las siguientes tiendas virtuales: Google Play, App Store y App Gallery, que pertenecen a Android, IOS y Huawei respectivamente.

Para difundir la propuesta de valor y capturar a los *early adopters*, se considera el uso de publicidad en lugares donde se mueven los segmentos de interés previamente descritos, utilizando para ello medios como pantallas led en sectores estratégicos de Santiago y Viña del Mar, letreros publicitarios en las rutas más transitadas de Chile, publicidad en el metro de Santiago y Valparaíso. Sumado a lo anterior, se utilizarán redes sociales como Facebook e Instagram para dar a conocer la propuesta de valor y los beneficios de usar Liion. En conjunto, se utilizarán estrategias de tráfico web mediante Google AdWords, anuncios en Facebook e Instagram para hacer más visibles las búsquedas.

Finalmente, se espera realizar anuncios a través de YouTube y crear una página web que permita a los usuarios conocer los diferentes aspectos de Liion.

Capítulo 4

Desarrollo de la plataforma

En el presente capítulo se expone la solución al problema presentado en los capítulos anteriores, aquí se abordan las diferentes áreas que permiten su desarrollo desde un punto de vista técnico. Se presentan los requerimientos de la aplicación, y se delimitan los esenciales para cumplir con un mínimo producto viable (MPV), se definen en detalle las diferentes tecnologías a utilizar, en cada uno de los diferentes niveles lógicos que permitirán un correcto funcionamiento de la aplicación.

4.1. Arquitectura de tres capas

Una aplicación móvil como Liion, se caracteriza por la fuerte interacción que tiene el usuario con la lógica oculta que hay por detrás, bajo esta premisa destacan las siguientes funcionalidades e interacciones que ejecuta el sistema:

- El cliente interacciona con un terminal móvil, el cual emite solicitudes a un servidor, que responderá con información correspondiente.
- El terminal móvil ejecutará solicitudes, donde el servidor deberá responder métricas procesadas, basadas en los datos guardados.
- El servidor proporcionará servicios de terceros para facilitar el desarrollo.
- El usuario generará datos, los cuales deberán ser almacenados.

Basado en lo anterior, se propone utilizar una arquitectura de tres capas, la cual es una forma de organizar las aplicaciones de software en tres niveles informáticos lógicos: la capa de presentación o interfaz de usuario; la capa de negocio, donde se procesan los datos; y la capa de datos o de persistencia, donde se almacenan y administran los datos asociados y/o generados por la aplicación, la figura 4.1, muestra a modo general una representación de la arquitectura, y las interacciones que se dan entre las capas. La principal ventaja de este tipo de arquitectura es la independencia que presenta entre sus capas, obteniéndose beneficios tales como:

- **Trabajo especializado de cada integrante del grupo de desarrollo**: Cada capa se puede trabajar con lenguajes, *frameworks* o tecnologías diferentes entre sí, lo que permite la separación del trabajo, y con ello el desarrollo individualizado, distribuyendo de mejor manera la responsabilidad sobre las capas.
- **Facilidad en el mantenimiento del sistema**: El hecho que cada capa sea independiente entre sí, implica que sea más sencillo realizar modificaciones aisladas, sin afectar otras.

- **Posibilidad de escalar la aplicación**: Debido a la naturaleza de la arquitectura, cualquier nivel puede escalar independientemente según sea necesario.
- **Integración de nuevas tecnologías**: Al tener cada capa separada, es posible integrar de forma prudente nuevas tecnologías, es decir, sin tener que modificar todo el sistema.
- **Seguridad mejorada**: Como no existe una comunicación directa entre la capa de presentación y la de datos, es posible evitar ataques como inyecciones SQL, u otras vulnerabilidades, que puedan comprometer la información presente en el sistema, lo cual es de vital importancia para el funcionamiento de este.

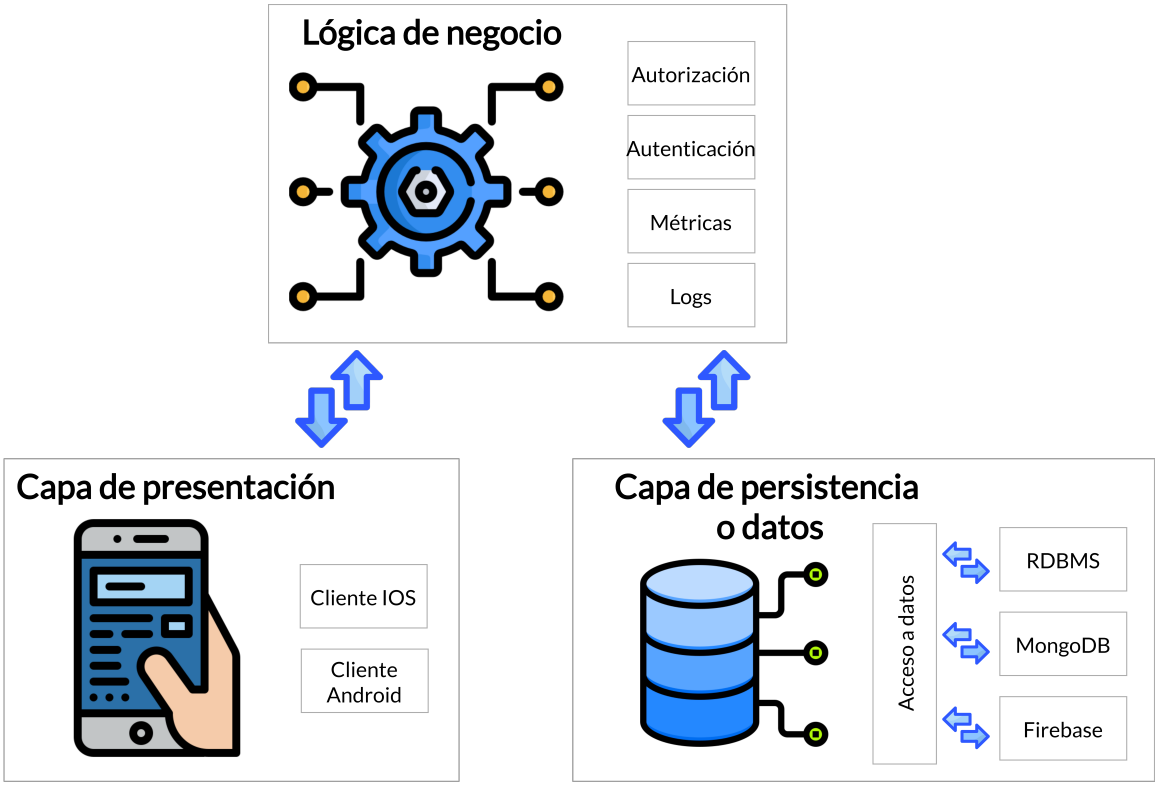


Figura 4.1: Arquitectura general de una aplicación de 3 capas (de elaboración propia).

4.2. Elección de tecnologías para el desarrollo de la solución

A continuación, se describe como se realiza el proceso de comparación y elección de las tecnologías a utilizar en la presente memoria. Esto, con el fin de obtener el software de desarrollo que mejor se adapte tanto a los requerimientos de la aplicación como a los conocimientos del equipo desarrollador. Para esto las tecnologías, se clasifican en distintas categorías en donde los diferentes *frameworks* serán ordenados de mayor a menor, ocupando un puesto dependiendo de la apreciación. Esta puede ser subjetiva basada en la opinión del equipo (como lo que es conocimiento previo) o bien, objetiva basándose en la documentación de este (como lo es el rendimiento).

4.2.1. Capa de presentación

El denominado *frontend*, es la parte visible de la aplicación, con la cual el usuario final interactúa, en donde no solamente los aspectos técnicos son importantes, sino que también lo son la usabilidad, estilo y accesibilidad. Esta capa se comunica directamente con la capa del negocio, para pedir o entregar datos.

Actualmente, en dispositivos móviles hay varias alternativas de desarrollo *frontend*. En primer lugar, se tienen los lenguajes nativos de los dispositivos Android y IOS, que son Java/Kotlin y Objective-C/Swift respectivamente. Desarrollar en estos lenguajes tiene ventajas y desventajas, sin embargo, para el caso de Liion estas se descartan inmediatamente, ya que implicaría hacer el desarrollo dos veces, sin poder reutilizar el código, ni lógica, lo que aumentaría significativamente la dificultad y el tiempo de desarrollo.

En segundo lugar, están los *frameworks* o marcos de trabajo, que entregan un conjunto estandarizado de conceptos, prácticas y criterios, que permiten escribir código o

desarrollar una aplicación de manera sencilla. En el caso de Liion es importante considerar aquellos *framework* que permitan abstraer la lógica del *frontend*, para que este pueda ser implementada en diferentes ambientes de operación, como lo son Android y IOS.

La siguiente comparativa solo se centrará en las tecnologías más utilizadas para el desarrollo móvil del tiempo actual. Estas tienen en común que todas son multiplataformas (algunas con más adaptabilidad que otras), lo que es algo sumamente importante a considerar para reducir los tiempos de desarrollo.

4.2.1.1. Flutter

Flutter es un *framework* de código abierto desarrollado por Google, el cual permite crear aplicaciones que se compilan para cada dispositivo, de forma que se genera una aplicación nativa que no necesita de un módulo *runtime* o un navegador. Flutter utiliza el lenguaje de programación Dart, lo que permite programar en dispositivos IOS y Android, ya que genera código nativo para cada dispositivo. Este *framework* está fuertemente orientado a objetos, ya que todo se trata como un *widget*

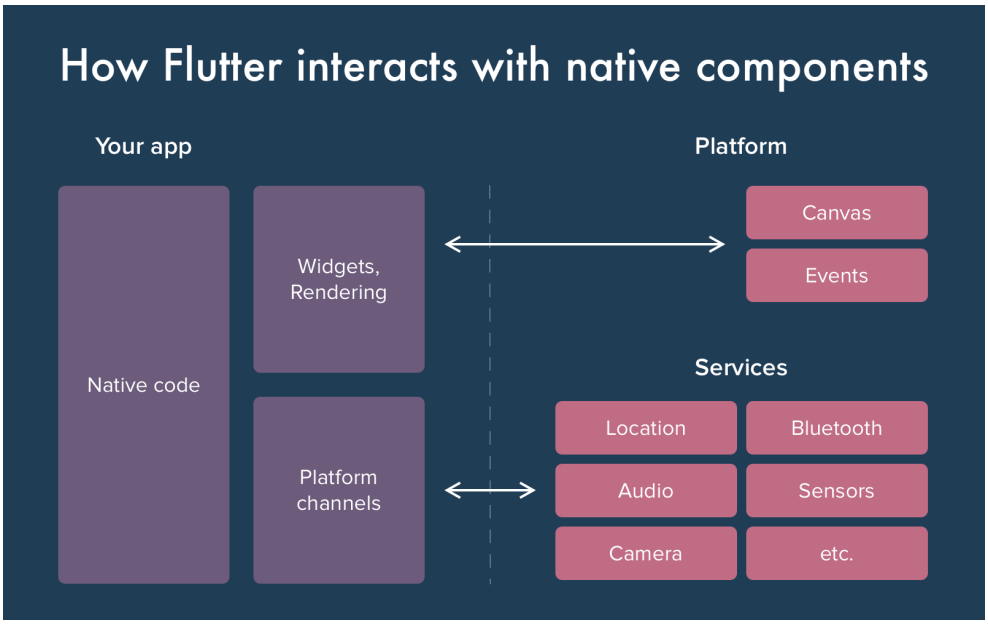


Figura 4.2: Interacción de Flutter con componentes nativos.

Una gran ventaja que posee Flutter sobre otros *frameworks* es que posee sus propios módulos empacados, por lo que no necesita comunicarse con módulos nativos preexistentes, esto ofrece una ventaja competitiva, ya que las aplicaciones pueden operar exactamente igual en distintos sistemas operativos, sin necesidad de utilizar librerías externas. En la figura 4.2 [34] se aprecia como se estructura Flutter y como este interactúa de manera general con los componentes del sistema base.

4.2.1.2. React Native

React Native es un *framework* de código abierto desarrollado por Facebook, el cual es un derivado del ya popular ReactJS, para desarrollo de web. Este utiliza Javascript, y al igual que Flutter, permite la abstracción del dispositivo, esto quiere decir, que es posible la utilización de la misma lógica para diferentes sistemas operativos (IOS y Android), ayudando a un desarrollo más eficaz y sencillo. React Native proporciona un desarrollo a través de la orientación a objetos, así como también la programación funcional (a través de *hooks*). Este *framework* es altamente utilizado en el mundo del desarrollo, ya que tiene un buen soporte y ha sido probado exitosamente en producción por años en variadas aplicaciones.

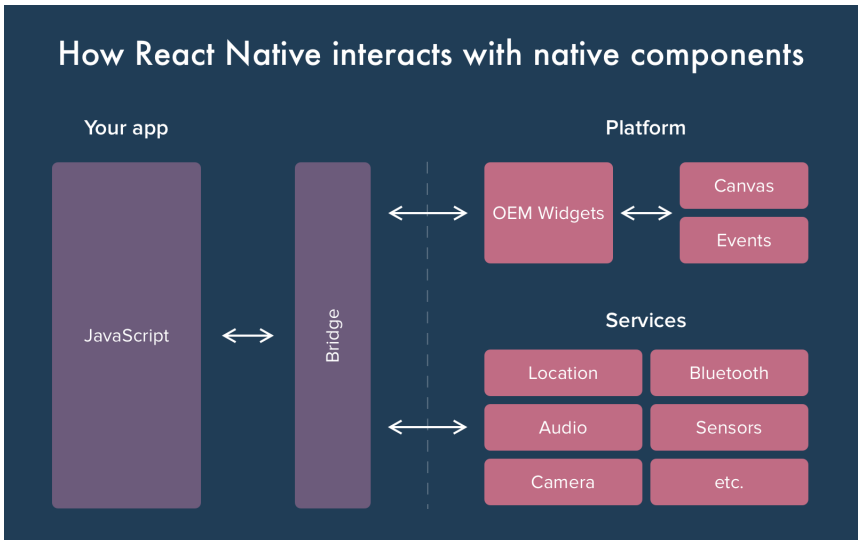


Figura 4.3: Interacción de React Native con los componentes nativos.

En la figura 4.3 [34] se aprecia como React Native interactúa con los componentes nativos de las plataformas. En esta, se puede destacar que Javascript no pasa por un proceso de transformación de código, sino que se ejecuta de forma virtualizada en el equipo de manera *just in time (JIT)*, para luego a través de una API, llamar a los módulos nativos que se ejecutan en el dispositivo, de esta manera se obtiene un rendimiento muy similar al que se obtendría utilizando Java/Kotlin u Objective-C/Swift para Android o IOS respectivamente.

4.2.1.3. Xamarin

Xamarin es un *framework* de desarrollo móvil multiplataforma de código abierto para construir aplicaciones móviles con .NET, originalmente fue creado por la empresa Xamarin (2011) basado en el proyecto Mono [35], el cual que fue comprado y adoptado posteriormente por Microsoft en 2016.

Xamarin utiliza el lenguaje C#, y Windows sostiene que alrededor del 90 % del código es portable entre plataformas a través de los módulos. En Android se precompila en un lenguaje intermedio y luego se compila a lenguaje nativo en runtime (JIT), mientras que en IOS se compila directamente en lenguaje nativo (AOT), para ARM.

La arquitectura de Xamarin es apreciable en la figura 4.4 [36], junto con la forma en que interactúa con los componentes nativos.

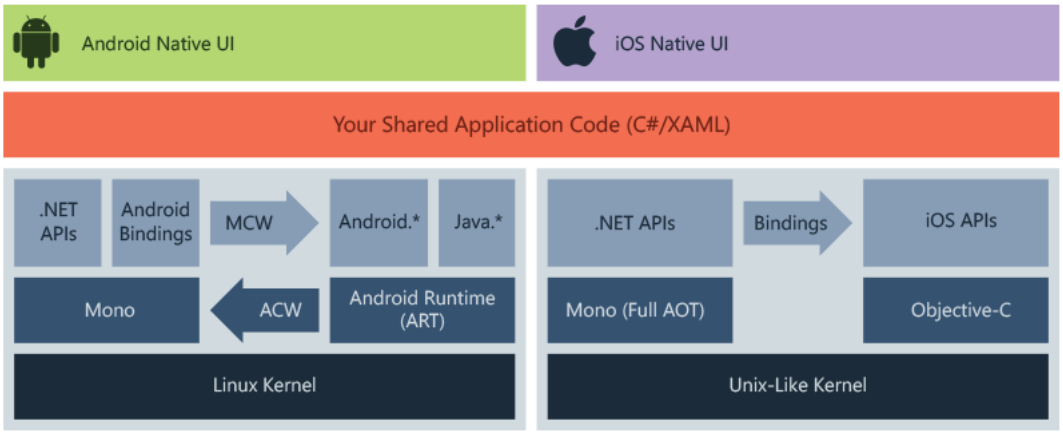


Figura 4.4: Arquitectura Xamarin.

4.2.1.4. Native Scripts

Native Scripts es un *framework* orientado a desarrollo móvil de código abierto, creado en el 2014 originalmente por Telerik, posteriormente comprado por la compañía Progress. El principal propósito de este es ofrecer una alternativa de desarrollo móvil que permita usar Javascript, o cualquier lenguaje que se pueda transformar a JavasScript, como lo es TypeScript. Hoy en día Native Scripts soporta no solo vainilla Javascript (o Typescript), sino que también *frameworks* de desarrollo web como lo son Angular, Vue, React y Svelte.

Native Scripts funciona de manera similar a como lo hace React Native. Ambos se empaquetan en un ambiente virtual en el cual se ejecuta el código en Javascript de manera *Just in time (JIT)*; sin embargo, la manera como se comunica con las *API's* nativas del equipo es distinta, si bien ambos pueden realizar llamadas a las *API's* nativas, Native Scripts las inyecta desde el dispositivo a su máquina virtual, haciendo su acceso más sencillo, y no requiriendo conocimiento del lenguaje nativo (En React Natuve existen muchas librerías proporcionadas por la comunidad que permiten la comunicación con el hardware). En la figura 4.5 [37] se puede apreciar la arquitectura de Native Scripts, y como esta interacciona con las librerías nativas.

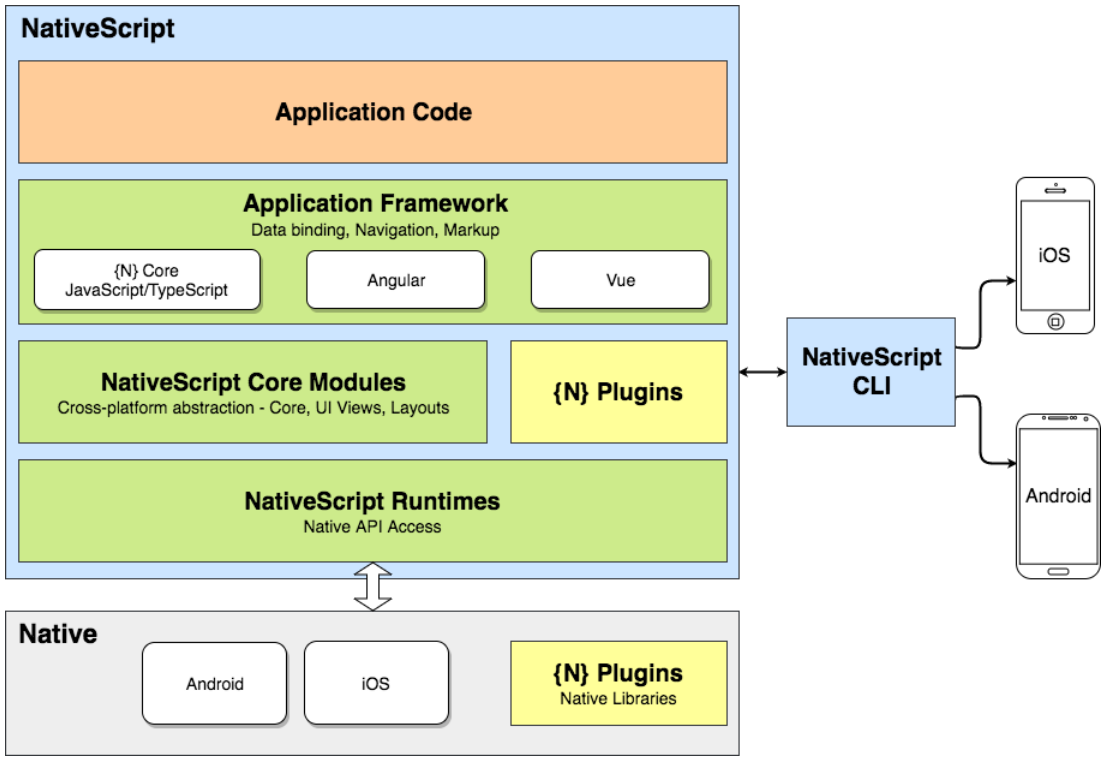


Figura 4.5: Arquitectura de Native Scripts.

4.2.1.5. Ionic

Ionic es un *framework* de código abierto el cual permite desarrollar aplicaciones híbridas en móviles bajo el lenguaje Javascript. Fue creado en 2013, y en su primera versión funcionaba a base de AngularJS, con los años ha ido evolucionando hasta la actualidad donde permite utilizar *AngularJS, VueJS y ReactJS*.

Ionic utiliza *webView* y no tiene la capacidad de utilizar *API's* nativas directamente como si lo hacen NativeScript, React Native, Xamarin o Flutter. En cambio, utiliza los módulos de Apache Cordova para comunicarse con el hardware. En la figura 4.6 [38], se puede apreciar este comportamiento, y como Ionic opera utilizando puentes por medio de Cordova, para el acceso a las *API's* nativas.

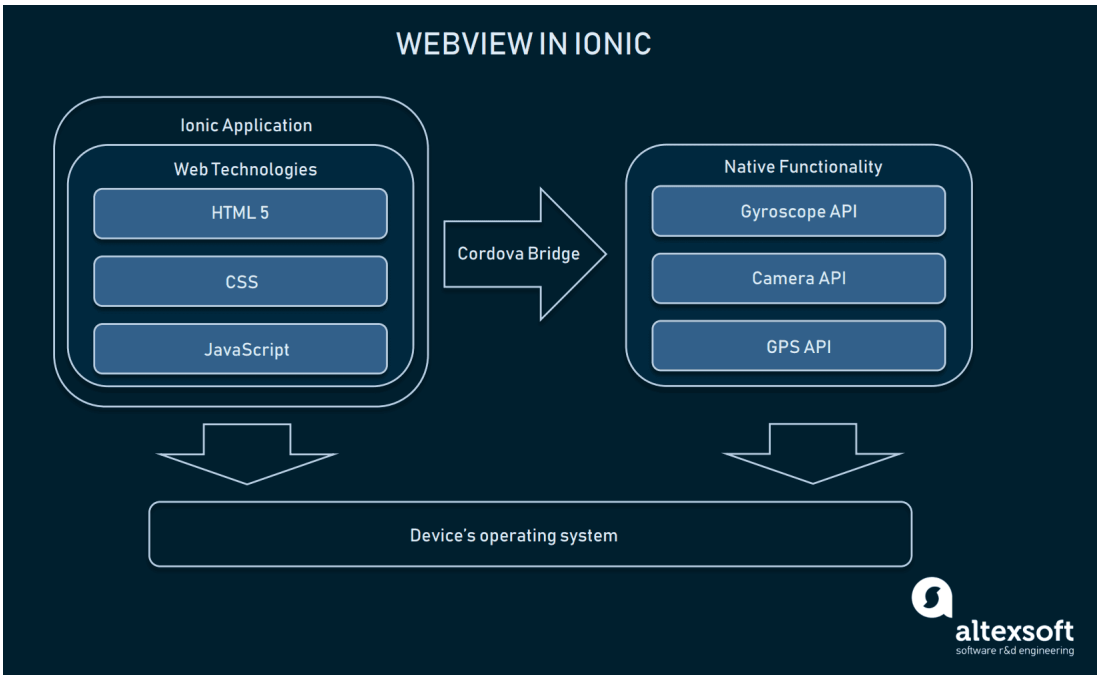


Figura 4.6: Arquitectura de Ionic.

4.2.1.6. Comparativa

En la presente subsección, se hace la comparativa de todos los *frameworks* nombrados, evaluando cada punto definido en el inicio de la sección, para luego definir el elegido.

- **Tamaño de las comunidades:**

  Cuando se trata del desarrollo de aplicaciones móviles, particularmente el desarrollo multiplataforma, es bueno tener una comunidad activa de usuarios. Estas ofrecen una gran cantidad de paquetes listos para utilizar, así como también soluciones a problemas comunes. Para comparar en esta categoría, se utiliza la información provista por las figuras 4.8 y 4.1, las cuales fueron confeccionadas a partir de los datos directamente sacados por cada sitio web, en estas se muestran el comportamiento de los usuarios en las plataformas: GitHub, que posibilita a los usuarios almacenar código; y Stack Overflow, sitio web donde desarrolladores escriben consultas, las cuales son importantes, ya que permiten, de una forma

directa, tener acceso a una gran cantidad de soluciones a problemas que, de alguna u otra manera, podrían surgir durante en el desarrollo. En la tabla 4.1, es posible apreciar cuáles serían los puestos ocupados por cada *framework* basados en el tamaño de las comunidades.

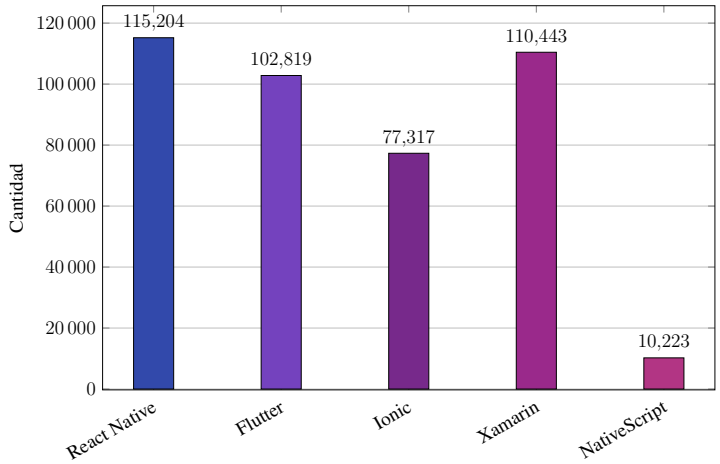


Figura 4.7: Cantidad de consultas en Stack OverFlow para cada *framework* al 12/05/2021.

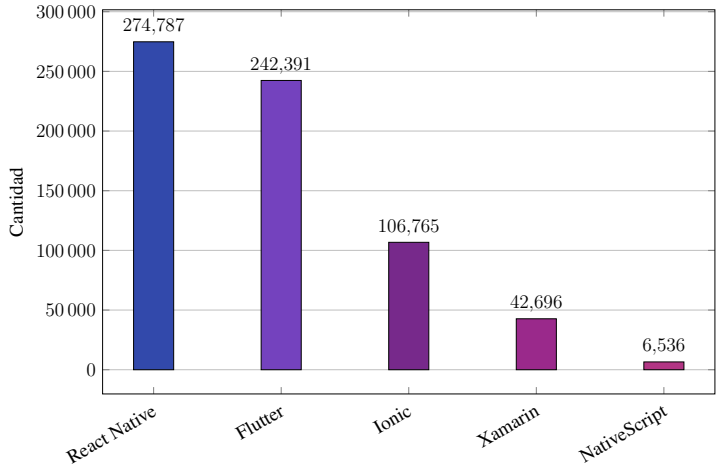


Figura 4.8: Cantidad de repositorios en GitHub para cada *framework* al 12/05/2021.

Puesto	Framework
1	React Native
2	Flutter
3	Ionic
4	Xamarin
5	NativeScript

Cuadro 4.1: Tabla de posición de los *framework* de *frontend* (tamaño de las comunidades)

- **Rendimiento del Framework:**

  Esta categoría analiza cómo se compara una misma aplicación móvil diseñada con los diferentes *frameworks*, desde un punto de vista del rendimiento en tiempo de ejecución. Sin embargo, comparar el rendimiento y evaluarlo no es sencillo, ya que depende de muchos factores y variables, incluidos el dispositivo, el código, la aplicación y las funciones que se utilizan. Aquí, a grandes rasgos, es posible identificar dos grupos altamente marcados por el tipo de aplicación resultante:

  - Aplicaciones pseudo nativas: Se caracterizan por procesar sus componentes, o partes del código nativamente. Esto brinda un rendimiento similar al que se obtendría utilizando los lenguajes cien por ciento nativos. Los *frameworks* que entregan este tipo de aplicaciones son:
    - React Native.
    - Flutter.
    - Xamarin.
    - NativeScripts.
  - Aplicaciones web empaquetadas: Al utilizar la tecnología web para representar una aplicación se reduce el rendimiento, debido a que no hay un acceso directo a las API del sistema. Dentro de esta categoría se encuentra:

- Ionic.

Según lo explicado anteriormente, la tabla 4.2, corresponderá a las posiciones ocupadas por los *framework*.

Puesto	Framework
1	React Native, Flutter, Ionic, Xamarin
2	Ionic

Cuadro 4.2: Tabla de posición de los *framework* de *frontend* (rendimiento)

- **Costo del aprendizaje de Frameworks y sus lenguajes:**

  Los cinco *frameworks* expuestos se caracterizan por utilizar diferentes tipos de lenguaje de programación. Estos toman un rol fundamental en la decisión de que *frameworks* ocupar, ya que toma importancia el conocimiento previo que posee el equipo de desarrollo respecto a los lenguajes, donde privilegiar un *framework* con un lenguaje de programación ya conocido, será menos costoso en tiempo y dinero, que elegir uno donde los conocimientos sean nulos. A modo de resumen, cada *framework* ocupa los siguientes lenguajes:

  - React Native y NativeScripts: Utilizan JavaScript, en la actualidad es de los lenguajes más populares y dinámicos para desarrollar aplicaciones híbridas. También es posible ocupar TypeScript, superconjunto de JavaScript que añade tipos estáticos y objetos basados en clases.
  - Flutter: Utiliza Dart, un lenguaje poco común desarrollado en Google con el objetivo de permitir a los desarrolladores utilizar un lenguaje orientado a objetos.
  - Ionic: Utiliza los lenguajes HTML, CSS y JavaScript, ya que utiliza un *webview* para desplegar sus aplicaciones.

- Xamarin: Utiliza C#, derivado de C/C++ que utiliza el modelo de objetos de la plataforma .NET.

Con el fin de exponer el conocimiento previo del equipo, estos respondieron una breve encuesta, expresando su percepción de cada lenguaje, asignándoles un puntaje con base en las siguientes premisas:

(1) Considero que no se nada. (2) Considero que recién estoy iniciando en el lenguaje. (3) Considero que tengo los conocimientos, puedo realizar aplicaciones medianas. (4) Considero que puedo realizar aplicaciones complejas.

Obteniéndose como resultado de la tabla 4.3 (Dev, es acrónimo de *developer* o desarrollador), donde es posible apreciar que el lenguaje JavaScript es una característica fuerte del equipo de desarrollo versus cualquier otro, lo que determina cuáles serán las posiciones ocupadas por los *framework* en esta categoría 4.4.

Privilegiar un *framework* con un lenguaje de programación ya dominado, ahorra tiempo de adaptación y aprendizaje, ya que los *framework* son herramientas que ayudan a programar, pero el núcleo de estos son los lenguajes de programación.

Lenguaje	**Dev**		**Puntaje**
	1	2	
JavaScript	3	3	3
HTML	3	2	2.5
CSS	2	2	2
C#	1	1	1
DART	1	1	1
TypeScript	2	3	2.5

Cuadro 4.3: Encuesta tomada a los desarrolladores sobre su conocimiento de los lenguajes de programación para *frontend*

4.2.1.7. Tabla resumen y decisión final

	Comunidad	Rendimiento	Costo del aprendizaje
Expectativas	Suficientemente grande como para solucionar problemas comunes u obtener implementaciones.	Buen rendimiento frente a la escalabilidad y al uso del *hardware*.	Que funcione basándose en un lenguaje de programación dominado por los desarrolladores.
React Native	Gran comunidad de desarrollo tanto en Github como StackOverflow.	Permite de utilización de API nativas lo que genera un rendimiento optimó	Tanto Typescript como Javascript son dominados
Flutter	Gran comunidad de desarrollo tanto en Github como StackOverflow.	Permite de utilización de API nativas lo que genera un rendimiento óptimo	Si bien Dart es similar a Javascript este no es dominado por los desarrolladores
Xamarin	Gran comunidad de desarrollo en Github, dispar a lo presentado en StackOverflow.	Permite de utilización de API nativas lo que genera un rendimiento óptimo	C# no es dominado por los desarrolladores.
Ionic	Gran comunidad de desarrollo menor a sus contendientes	Funciona a través de un *webView*, por lo que rendimiento no es óptimo	Tanto Typescript como HTTP y CSS son dominados
NativeScript	Baja comunidad presente en ambas plataformas.	Permite de utilización de API nativas lo que genera un rendimiento óptimo	Tanto Typescript como Javascript son dominados

Puesto	Framework
1	React Native y NativeScript
2	Ionic
3	Xamarin y Flutter

Cuadro 4.4: Tabla de posición de los *framework* de *frontend*
(Costo de aprendizaje)

Finalmente, y obteniendo la posición ponderada uno, el elegido y que mejor se adapta tanto al proyecto como al equipo de desarrollo es React Native, utilizándolo junto con el lenguaje de programación Javascript.

4.2.2. Capa de negocio

La capa lógica o de negocio, es donde se abordan los problemas que la aplicación debe resolver, en concreto, se procesan las solicitudes hechas por la capa de presentación, para decidir qué información será necesaria solicitar a la capa de datos, para luego devolver dichos resultados a la capa de presentación. Como se puede inferir, esta capa actúa como intermediario entre las demás capas, ejerciendo como un filtro entre estas.

4.2.2.1. Python Flask

Corresponde a un *framework* de código abierto, el cual permite desarrollar aplicaciones de *backend* con Python. Tiene una filosofía de promover lo justo para desarrollar funcionalidades y no juntar una gran cantidad de módulos *out of the box*, que podrían no utilizarse. Es por esto que ofrece libertad a la hora de desarrollar, dando la opción de utilizar diferentes extensiones para mejorar sus funcionalidades de ser necesario, o bien programarlas desde cero, según sea necesario.

4.2.2.2. Express

Es un *framework* de código abierto para NodeJS (que corresponde a una plataforma de código abierto) que permite construir aplicaciones web con JavaScript, se define como un *runtime* basado en el motor de ejecución ChromeV8. *Express* se encarga de implementar toda la lógica de un servidor web sin tener que programar todo.

4.2.2.3. Laravel

Corresponde a un *framework* de código abierto, que fue desarrollado en 2011 para facilitar la programación en PHP, tiene una basta cantidad de funciones por defecto y compatibilidad con diversas tecnologías. Provee autenticación *out of the box* desarrollada y mantenida por el equipo desarrollador de Laravel. Esta es una gran diferencia en comparación a otros *frameworks* como Express, que dependen de bibliotecas externas para llevar a cabo estas funciones. Si bien se asocia a Laravel con aplicaciones monolíticas como blogs o sitios de varias páginas, este también puede operar en forma de *API*, siendo capaz de atender las llamadas HTTP. Además, en el área de aplicaciones de navegador también posee API's para integrarse directamente a *frameworks* como lo son, ReactJS, AngularJS, VueJS entre otros.

4.2.2.4. Ruby on Rails (RoR)

Es un *framework* de código abierto basado en Ruby. Oficialmente lanzado en 2005, convirtiéndose en el más antiguo de los listados. En sus inicios fue pionero en adopción e implementación del modelo MVC (modelo-vista-controlador), migraciones, *scafolding,* entre otras innovaciones. Hasta el día de hoy es utilizado y aún es visto como una opción viable, más aún cuando se habla de un MPV, sin embargo, ha ido perdiendo popularidad en el desarrollo web. Posee una variada librería a través su sistema de gemas, las cuales poseen el soporte y validación de la comunidad.

4.2.2.5. Spring Boot

Es un *framework* de código abierto, que está construido sobre Spring, un marco de desarrollo que posibilita desarrollar aplicaciones de *backend*, este adopta la filosofía MVC (modelo-vista-controlador). SpringBoot es una capa más de abstracción, donde su principal función es acortar y simplificar el desarrollo y configuración de Spring. Está desarrollado en Java. SpringBoot está fuertemente orientado a los microservicios, lo que permite desarrollar aplicaciones altamente escalables y tolerante a fallas.

4.2.2.6. Comparativa

- **Tamaño de las comunidades:** Tal como se explicó en la subsección anterior, una buena comunidad de usuarios activos, ofrecen una gran capacidad de control sobre los problemas, dificultades o requerimientos que deba tener la aplicación. Las figuras 4.9 y 4.10 (de confección propia), a partir de los datos provistos por sus respectivos sitios web, muestran el comportamiento de los usuarios en las dos principales plataformas de desarrollo de internet. Donde se ve una clara tendencia al uso de RoR. La tabla 4.5 muestra los puestos ocupados por cada *framework* basándose en los gráficos.

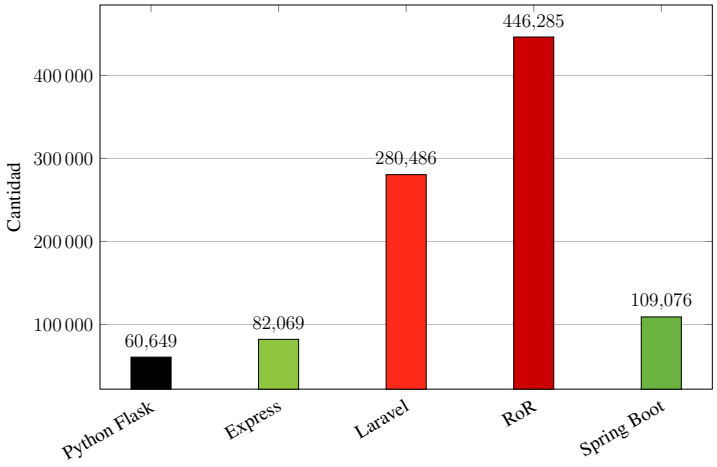


Figura 4.9: Cantidad de consultas por tag en Stack OverFlow para cada *framework* al 02/06/2021.

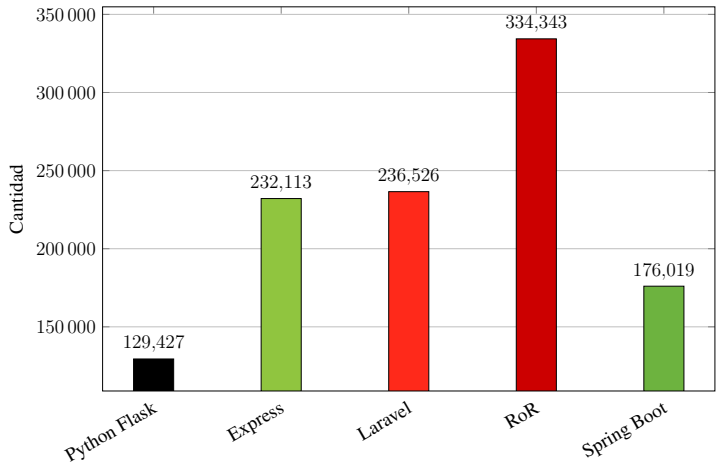


Figura 4.10: Cantidad de repositorios en GitHub para cada *framework* al 03/06/2021.

Puesto	**Framework**
1	Ruby on Rails
2	Laravel
3	Express
4	Spring Boot
5	Python Flask

Cuadro 4.5: Tabla de posición de los Framework de *backend* (tamaño de las comunidades)

- **Rendimiento del *framework*:** Uno de los aspectos más relevante a la hora de elegir un *framework* es qué tan rápido y eficiente es, esto varía dependiendo del punto de vista porque existen distintas métricas para medir el rendimiento. Entre las que se encuentra:

  - Velocidad de respuesta para archivos estáticos.
  - Uso de CPU y RAM.
  - Integración con bases de datos.

Sin embargo, la presente comparativa se centra en estas de forma general, ya que se medirá como estas se degradan conforme más conexiones se realizan. Es importante porque ciertos frameworks pueden responder muy bien en las primeras peticiones, pero mientras más usuarios se conectan, las métricas se van degradando, es decir, entre más usuarios, más lento se vuelve, un aspecto esencial a tener en cuenta en una aplicación de uso masiva.

Para esto se utiliza la información provista por *TechEmpower*[39], donde en su web muestran una comparativa de rendimiento de variados *frameworks* web ejecutando tareas fundamentales, para luego ser ordenados en un ranking. En la tabla 4.6 es posible apreciar las posiciones ocupadas en dicho ranking. Es importante señalar que se seleccionaron las implementaciones donde se utilizan base de datos PostgreSQL y MySQL, con el fin de estandarizar las pruebas.

Posicion en el ranking	**Framework**	**Posición en esta comparativa**
378	Ruby on Rails	5
343	Laravel	4
235	Express	1
317	Spring Boot	3
315	Python Flask	2

Cuadro 4.6: Tabla de *benchmark* para *framework* de *backend*

- **Costo del aprendizaje de *frameworks* y sus lenguajes:**

  El principal foco de análisis en esta parte, al igual que en la sección anterior, estará en el lenguaje de programación utilizado por cada *framework*, donde tendrá mayor puntaje, el que sea más dominado por los desarrolladores, ya que esto implicaría un menor costo desde el punto de vista del tiempo.

  Con el fin de exponer el conocimiento previo del equipo, estos respondieron una breve encuesta, expresando su percepción de cada lenguaje, asignándoles un puntaje basándose en las siguientes premisas:

(1) Considero que no se nada.

(2) Considero que recién estoy iniciando en el lenguaje.

(3) Considero que tengo los conocimientos, puedo realizar aplicaciones medianas.

(4) Considero que puedo realizar aplicaciones complejas.

Obteniéndose como resultado lo apreciable en la tabla 4.7.

Lenguaje	**Dev**		**Puntaje**
	1	2	
Python	4	4	4
JavaScript	4	3	3.5
Php	1	2	1.5
Ruby	1	1	1
Java	2	2	2

Cuadro 4.7: Encuesta tomada a los desarrolladores sobre su conocimiento de los lenguajes de programación para *backend*

Puesto	**Framework**
1	Express
2	Flask
3	laravel
3	Spring Boot y RoR

Cuadro 4.8: Tabla de posición de los *framework* de *frontend*
(Costo de aprendizaje)

Finalmente, y obteniendo la posición ponderada uno, apreciable en la tabla 4.8, el elegido y que mejor se adapta tanto al proyecto como al equipo de desarrollo es Express, utilizándolo junto con el lenguaje de programación Javascript, con esto se

define un único de lenguaje de programación para todo el *stack*, lo que facilitara el desarrollo.

4.2.3. Capa de datos o de persistencia

La capa de datos (o persistencia) es aquella encargada de guardar y resguardar todos los datos útiles para el correcto funcionamiento de la aplicación, es importante porque generalmente se requiere conocer algún estado anterior, que en la mayoría de los casos no se dispone en la sesión actual, como por ejemplo la autenticación (no resulta útil tener que registrarse cada vez que se utiliza la aplicación).

Liion, al ser una aplicación de transporte, debe manejar sesiones, datos de viajes, preferencias de usuario, así como también debe mover datos de ubicación constantemente, realizar reportes de funcionamiento, entre otras funcionalidades.

En el almacenamiento de datos, hay diferentes opciones según naturaleza de la aplicación, en ocasiones se necesitan incluso distintas bases de datos para una misma aplicación. En la actualidad se distinguen dos grandes grupos de bases de datos, las cuales se pueden preciar en la figura 4.11 [40].

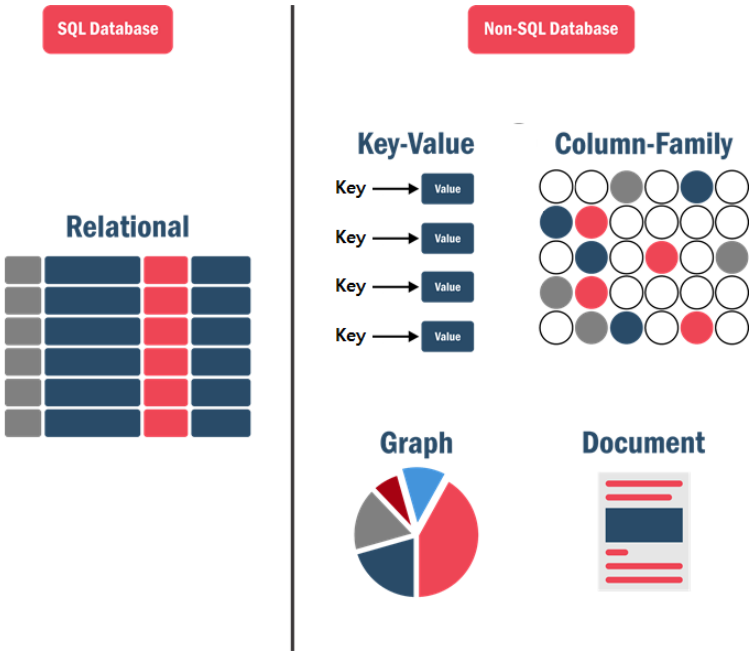


Figura 4.11: SQL y No-SQL

4.2.3.1. Base de datos SQL

- Bases de datos relacionales

  Son probablemente las más conocidas y utilizadas hoy en día, estas se caracterizan por almacenar los datos en formas de tablas, las cuales poseen referencias entre estas para lograr agrupar los datos eficientemente, brindando facilidad de uso, precisión, y garantizando la integridad de datos al utilizar referencias.

4.2.3.2. Base de datos no-SQL

- Bases de datos *Key-Value*

  Como su nombre lo indica, estas bases de datos solo contienen llaves y valores, son especialmente útiles cuando no se requieren realizar operaciones complejas sobre los datos, y más bien las búsquedas son simples. Una desventaja evidente en comparación a las bases de datos relaciones es que solo se pueden realizar *queries* simples, sin embargo, gracias a esto, estas pueden llegar a ser sustancialmente más rápidas.

- Bases de datos *Column Family*

  Son similares a las bases de datos relacionales desde el punto de vista de la forma (si se mira la figura 4.11 ambas parecen matrices), sosteniendo esta idea, de manera simple, una base de datos *Column Family* podría entenderse como una base de datos relacional transpuesta, esto quiere decir, que se almacenan las columnas en forma de filas y las filas en forma de columna. Esta filosofía permite realizar *queries* complejas de tipo agregación de manera mucho más rápida que las bases de datos relacionales tradicionales. La figura 4.12 [41] puede ayudar a entender mejor esta idea.

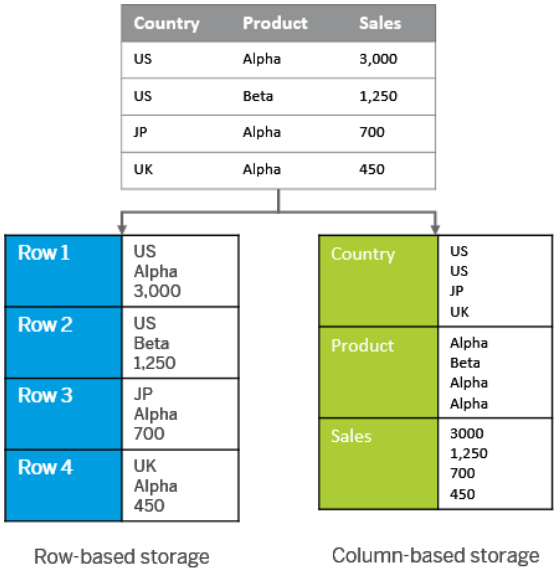


Figura 4.12: Base de datos *Column Family*

- Bases de datos de Grafos

  Las bases de datos tipo grafo, podrían parecer en teoría similar a las relacionales, en el sentido que ambas poseen datos relacionados entre sí, la diferencia radica en que las de grafos guardan las relaciones en el nivel del dato individual, a diferencia de las relaciones que hacen esto a nivel de tabla, esto permite tener relaciones más complejas de manera más sencilla que las bases de datos relaciones, lo que es útil cuando se tienen datos irregulares y/o con relaciones complejas

entre ellos.

- Bases de datos de Documentos

  Estas bases de datos como su nombre lo indica almacenan los datos en forma de documentos, esto significa utilizar una estructura de datos similar a lo que es un JSON [42], estructura muy utilizada en *frontend* cuando se trabaja con Javascript, lo que permite incorporar directamente objetos del *frontend* a la base de datos, también al no ser estructuradas rígidamente, se permite una gran flexibilidad en el tipo de datos a almacenar, lo que es óptimo para bases de datos durante el periodo de desarrollo, ya que en muchos casos no hay certeza de la estructura de los datos final.

4.2.3.3. Elección de Base de datos

Dada la información anteriormente planteada, y teniendo en consideración el MPV, se plantean los siguientes criterios para la elección de la base de datos a utilizar:

- Flexibilidad: Las bases de datos se evalúan según sean rígidas o flexibles según sea el caso.

- Popularidad: Se ordenan los tipos de bases de datos según su popularidad, ya que mientras más se utilizan, más material disponible existe al respecto.

- Experiencia: Al igual que con los lenguajes de programación del *frontend* se puntúa según: (1) Considero que no se nada. (2) Me considero iniciado con este tipo de bases de datos. (3) Considero que tengo conocimientos utilizarla la base de datos en proyectos medianos. (4) Considero que tengo un buen manejo de la tecnología.

- Caso de uso: Se clasifican las bases de datos según como se percibe que se adecuen al proyecto de Liion o no.

4.2.3.4. Flexibilidad

A continuación se adjunta una tabla con la flexibilidad de cada base de datos.

Base de Datos	Flexibilidad
Relacional	Rígida
Llave Valor	Flexible
Grafos	Flexible
Columnares	Rígida
Documentos	Flexible

Cuadro 4.9: Tabla de flexibilidad de bases de datos

Ya que Liion busca generar un MPV, la flexibilidad es importante para realizar cambios en el tipo, estructuras de datos y la relaciones entre estos, sin tener que generar todo un esquema de migraciones nuevamente, es por esto que se prefiere bases de datos flexibles, obteniéndose tabla 4.10 la cual muestras las posiciones tomadas por cada tipo.

Puesto	Base de datos
1	Documentos, Llave valor, Grafos
2	Relacional, Columnar

Cuadro 4.10: Tabla de resultados de flexibilidad de bases de datos

4.2.3.5. Popularidad

La popularidad de las bases de datos se basa en el estudio de DB-Egines [43]. En su ranking, se calcula un puntaje de popularidad basado en las siguientes variables, número de búsquedas en Google y Bing, las tendencias de Google Trends [44], frecuencia de aparición en Stack Overflow y DBA Stack Exange, cantidad de ofertas de trabajo en sitios como Indeed e Hired, número de perfiles en Linkedin asociados a bases de datos, y relevancia en Twitter.

Posición	Nombre	Modelo	Puntaje
1	Oracle	Relacional	1270,94
2	MySQL	Relacional	1227,07
3	Microsoft SQL Server	Relacional	991,01
4	PostgresSQL	Relacional	568,51
5	MongoDB	Documentos	488,22
6	IBM Db2	Relacional	167,03
7	Redis	Llave Valor	165,25
8	Elastic Search	Motor de Búsquedas	154,71
9	SQLite	Relacional	130,54
10	Microsoft Acces	Relacional	114,94
11	Cassandra	Columnar	114,11
12	MariaDB	Relacional	96,79
13	Splunk	Motor de Búsquedas	90,27
14	Hive	Relacional	79,69
15	Microsoft Azure SQL	Relacional	74,79
16	Amazon DynamoDB	Llave Valor	73,76
17	Teradata	Relacional	69,34
18	Neo4j	Grafos	55,75
19	SAP HANA	Relacional	54,11
20	Solr	Motor de Búsquedas	52,11

Cuadro 4.11: Las 20 bases de datos más utilizadas.

Es importante destacar que se considera *Wide Column* como Columnar y en las bases de datos que tienen la opción de poder utilizar varios modelos, solo se deja el principal, o el más popular. También se omiten los *Search Engines*.

De la tabla 4.11 se aprecia por simple inspección que las más populares son las bases de datos relacionales. Para ordenar el resto de tipos lo que se hace es promediar

el puntaje por cada categoría, lo que finalmente se puede resumir como:

- Documentos: 488,22
- Llave Valor: 119,505
- Columnar: 114,11
- Grafos: 55,75

Finalmente, el ranking final queda como:

Posición	Tipo de Bases de Datos
1	Relacional
2	Documentos
3	Llave Valor
4	Columnar
5	Grafos

Cuadro 4.12: Posición final de las bases de datos en comunidad

4.2.3.6. Experiencia

Según lo expuesto anteriormente y lo explicado en los capítulos pasados en esta sección, se evalúan las tecnologías según la experiencia de los desarrolladores, lo cual es importante para un desarrollo óptimo y rápido del producto. En la tabla 4.13 se adjuntan los puntajes y, las posiciones finales por experiencia para cada tipo de base de datos.

Base de datos	Dev		Puntaje	Posición
	1	2		
Relacional	2	2	2,5	1
Documentos	3	2	2,5	1
Grafos	1	1	1	3
Llave Valor	1	2	1,5	2
Columnar	1	1	1	3

Cuadro 4.13: Puntajes según experiencia previa y posición final.

El caso de uso es importante porque determina la factibilidad de uso de una tecnología por sobre otra (para Liion). En el caso de esta plataforma, se almacenan datos de usuarios, viajes, solicitudes, mensajes, notificaciones, entre otros. En general, los datos a consumir y escribir son tradicionales, en donde no se necesita la extrema velocidad de las bases de datos del tipo llave valor. La tabla 4.14 muestra el orden según como se adaptan al caso de uso de Liion.

Base de Datos	**Posición**
Relacional Documentos	1
Columnar	2
Llave Valor Grafos	3

Cuadro 4.14: Posición final de los tipos de bases de datos según el caso de uso de Liion

4.2.3.7. Elección

A continuación se muestra la tabla 4.15 con las posiciones de cada tipo de base de datos. Donde se puede notar que la mejor posición está compartida por las bases de datos relaciones junto a la de documentos.

Base de Datos	Flexibilidad	Popularidad	Experiencia	Caso de Uso	Posición Ponderada	Posición
Relacional	2	1	1	1	1,25	1
Documentos	1	2	1	1	1,25	1
Llave Valor	1	3	2	3	2,25	2
Columnar	2	4	3	2	2,25	3
Grafos	1	5	3	3	3	4

Cuadro 4.15: Puntajes para cada tipo de bases de datos para Liion

Finalmente, se tiene un empate en la primera posición con las bases de datos relacionales y de documentos, por lo tanto, se considera que ambas son buenas opciones; sin embargo, el equipo se decanta por utilizar una base de datos de documentos, en particular Firestore, ya que al utilizar la autenticación de Firebase resulta natural la integración dentro del mismo ecosistema.

4.2.4. Servicios externos y datos para el consumo

Liion hace uso de múltiples interfaces que facilitan el desarrollo e implementación de funcionalidades esenciales, como lo son la autenticación o el acceso a las rutas de conducción. Estas corresponden a las llamadas API's que permiten el intercambio de información entre dos componentes de software independientes. A modo general, estas actúan como intermediarios para la comunicación con funciones externas, obviando todo el desarrollo lógico que hay por detrás. Esto genera que el intercambio de información y/o servicios sea sencillo, al punto de pasar desapercibido para el usuario final. Además de proveer de las funciones anteriormente nombradas, las API's también actúan como bibliotecas desde donde es posible buscar información en específico, tal como ve en la figura 4.13 [45], donde la interfaz actúa como un intermediario entre el cliente y la base de datos, desligando a la capa de presentación de la lógica de negocios.

Entre las API's utilizadas se encuentran:

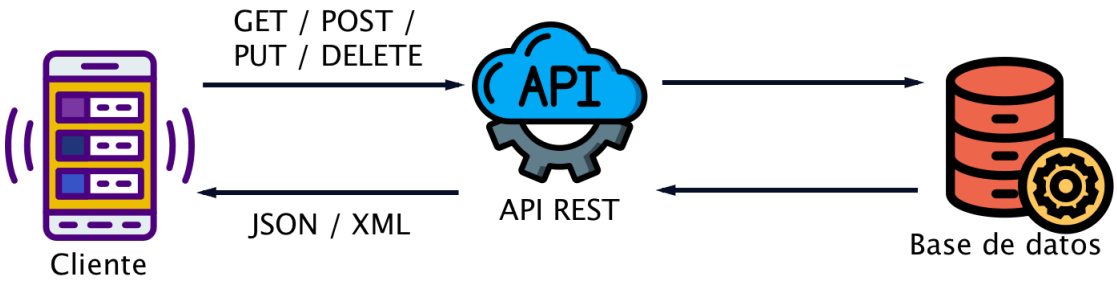


Figura 4.13: Funcionamiento de una API.

A continuación se listan las API's externas utilizadas para el correcto funcionamiento de Liion.

- Servicios de Google:
  - Firebase Auth: Permite realizar autenticación de usuarios a través de email, número telefónico y redes sociales.
  - Geocoding: Permite transformar direcciones en coordenadas decimales y viceversa.
  - Directions: Permite tener acceso a rutas de conducción a través de la indicación de un punto de origen y destino.
- Combustibles CNE: La comisión nacional de energía da acceso gratuito a su API de valores de combustibles, para el uso en el cálculo del costo del viaje.

4.2.5. Entorno de desarrollo

Para el desarrollo de la aplicación, se requiere un computador que soporte los diferentes softwares involucrados. De preferencia, un notebook con procesador Intel Dual Core de 2 [Ghz] o superior, 4 [Gb] de memoria RAM y 100 [Gb] de almacenamiento. Al ser una aplicación móvil, para su ejecución se requerirá de un smartphone que pueda ejecutar alguna versión reciente de Android, en particular las pruebas se ejecutan en un Motorola G6 Play y un Huawei P20 Lite, ya que el equipo no posee móviles con IOS.

En cuanto al software utilizado para el desarrollo de la aplicación, se emplean diferentes tecnologías dependiendo del entorno de desarrollo y su uso.

4.3. Arquitectura del sistema y resumen de tecnologías a ocupar

La figura 4.14, de elaboración propia, muestra el diagrama de arquitectura propuesto para Liion. Se hace énfasis en mostrar claramente la separación que existe entre el *frontend*, el *backend* y la base de datos, esto para seguir lo mas fielmente posible la arquitectura de tres capas propuesta en los apartados anteriores (con el fin de abstraer las diferentes funciones a sus propios niveles lógico). También es posible notar, la distribución de las tecnologías a ocupar para cada capa, donde en el *frontend* (o lado del cliente) se hará uso de React Native para crear toda la visualización; por el lado *backend* (o lado del servidor) se desarrollara una API en ExpressJs que alimentará al cliente con información, y por el lado de las bases de datos la elegida es Firestore, una base de datos de documentos.

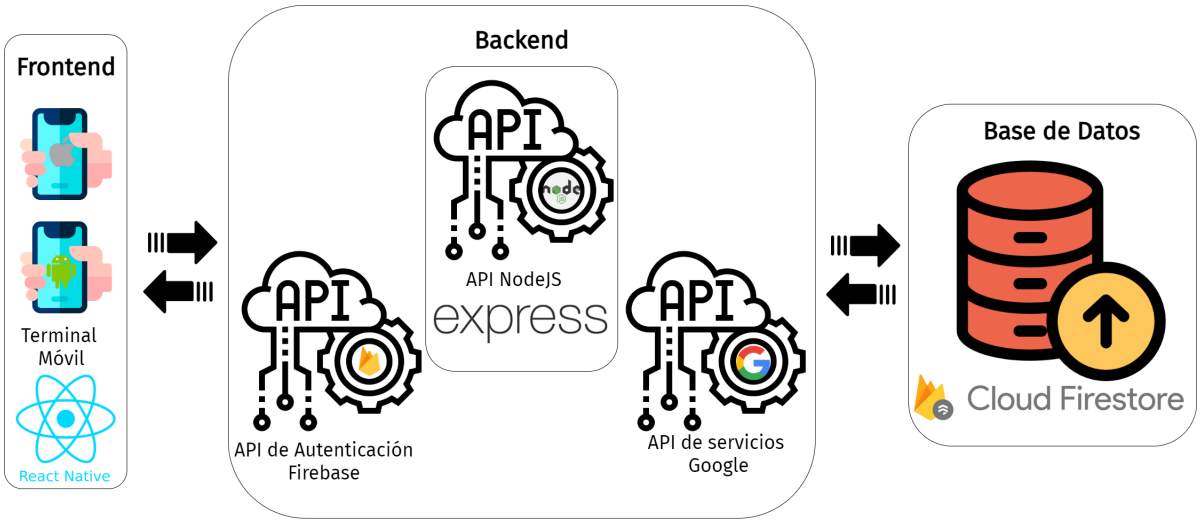


Figura 4.14: Arquitectura del Sistema

4.4. Análisis de requerimientos

Los requerimientos son la vía principal que tiene el cliente para dar a conocer sus necesidades al equipo de desarrollo. Conocerlos ayuda a desarrollar y refinar una solución.

Existen dos tipos de requerimientos, el primero de ellos corresponde a los funcionales, los cuales definen el comportamiento básico del sistema, e incluyen cálculos, entradas de datos y procesos comerciales. El segundo tipo de requerimiento, se denomina no funcionales, los que especifican cómo el sistema debe ejecutar sus funciones. Estos no afectan la funcionalidad básica del sistema. Incluso si no se cumplen, el sistema seguirá cumpliendo su propósito básico. Sin embargo, son importantes porque dan garantías desde el punto de vista la usabilidad y calidad.

4.4.1. Requerimientos funcionales

- Poder registrarme con mi correo electrónico, para poder desenvolverme como conductor o pasajero.
- Visualizar el grado de confianza y las calificaciones que se tengan sobre el perfil de usuario.
- Poder validar la identidad de los usuarios, con reconocimiento facial, al comparar con la cédula de identidad, o a través de la verificación con alguna base de datos del registro civil.
- Observar en tiempo real la ubicación en un mapa, utilizando datos georreferenciados.
- Poder calificar usuarios que hayan compartido viaje.
- Tener variedad de métodos de pago.

- Tener un chat para los usuarios de un viaje, que permita el cifrado *end-to-end*.
- Poseer un botón de emergencia para poder notificar a diferentes contactos de confianza.
- Poder cancelar mi participación en algún viaje, con una anticipación prudente y no sufrir de una penalización monetaria.
- Poder reservar un asiento de un viaje, para que posteriormente el conductor pueda confirmarlos.
- Poder publicar la solicitud de un viaje, indicando un origen y destino, para que sea visto por potenciales conductores.
- Filtrar los viajes disponibles que tiene la aplicación basándose en un origen y destino, el tipo de maleta/mochila que quiera llevar, si se permiten o no animales, o si será solo un viaje para un determinado género o mixto.
- Indicar a la aplicación, el rango máximo que me puedo desviar de la ruta principal para recoger o dejar a un pasajero, que quiera tener un viaje conmigo.
- El sistema debe tener una sugerencia de ruta óptima que exista entre origen y destino, permitiendo hacer modificaciones a este, definiendo así el trayecto definitivo para que sea visto por los pasajeros potenciales.
- El dinero pagado por los pasajeros se mantendrá en un monedero virtual hasta que el conductor decida sacarlo.
- El sistema debe sugerir un precio y un rango de variación de este, acorde a la distancia a recorrer, características de vehículo, tipo de ruta, y variables de mercado, como lo son los precios de otros medios de transporte.
- Poder personalizar mi oferta de viajes, indicando si se aceptan animales, maletas/mochila e indicando si será un viaje mixto o no.

- Un conductor debe poder confirmar las reservas hechas por los pasajeros, para poder realizar el viaje, o bien dejar la confirmación en modo automático, (aceptar por orden de llegada).
- Los usuarios deben tener la capacidad de modificar sus datos de perfil.
- Poder seguir al conductor para que se me notifique cuando generen nuevos viajes, y de esta manera solicitar reserva de asientos de forma oportuna.
- Poder crear y/o unirme a viajes recurrentes.
- Poder marcar para revisión comentarios con insultos e improperios hacia mi persona u otros.
- Poder revisar los comentarios marcados para revisión, con el fin de ver si efectivamente hay comentarios censurables. No se eliminarán puntuaciones bajas, solo comentarios que infrinjan normas de convivencia.
- Poder eliminar usuarios para controlar usos fraudulentos o con mal comportamiento.
- Poder modificar puntuaciones con el fin de arreglar posibles errores
- Poder cancelar viajes previos a realizarse de ser necesario en caso de emergencia.

4.4.2. Requerimientos no funcionales

- La aplicación debe cumplir con estándares de diseño, usabilidad y escalabilidad.
- Se requiere que el sistema sea mediante una aplicación para Android y IOS.
- El flujo de datos debe hacerse de forma segura y asegurando la privacidad de los datos de los usuarios, esto quiere decir que todo flujo de datos entre servidor, aplicación y cliente debe estar cifrado.

- La aplicación debe funcionar un 99 % del tiempo, es decir, poder registrar y operar viajes 24/7.
- Se debe minimizar el espacio utilizado por la aplicación en el dispositivo, privilegiando siempre el almacenamiento en la nube.
- La aplicación debe estar disponible al menos en español e inglés.
- El sistema debe ser capaz de operar con al menos 10.0000 usuarios con sesiones frecuentes.
- El sistema debe tener una respuesta rápida frente a alguna acción generada por los usuarios, determinando un tiempo máximo de respuesta de 3 [s].
- Si se deben incorporar funcionalidades ya realizadas por algún software, de preferencia deben ser *openSource*.

4.5. Mínimo producto viable

Se define como el mínimo producto viable, aquel que contenga las mínimas características implementadas para llevar a cabo la principal función de este, en particular para Liion, es todo el proceso que involucra usar la aplicación para crear, solicitar y ejecutar un viaje.
Para esto se consideran los siguientes hitos:

- Tener un módulo de identificación y registro funcionales.
- Crear un viaje dado un inicio, destino, fecha y hora.
- Buscar viajes compatibles según inicio, destino, fecha y hora (rango).
- Filtrar viajes de acuerdo a la capacidad de equipaje, preferencias de viaje y género.
- Realizar reserva de un asiento a un viaje determinado.
- Ejecución de un viaje una vez confirmados los asistentes, según su fecha y hora indicada.
- Verificación de usuarios, por medio código QR, desde conductor a pasajeros o viceversa.
- Calificación funcional de conductor y usuarios una vez terminado el viaje.

4.6. Capa de negocio

En esta sección, se presenta la lógica de la solución, es decir, el componente encargado de recibir peticiones, ejecutar cálculos y enviar información a la capa de presentación.

En detalle se explica como y que hace cada uno de los métodos implementados en el *backend*, como estos se comunican (de ser necesario) con otros servicios externos para responder las peticiones de la aplicación.

4.6.1. Estructura de la API

Para hacer uso de recursos disponibles de Firebase utilizados en Liion como, Firebase Authentication [46], Firestore [47], Cloud Storage [48] y Fcm [49] desde el *backend*; Firebase provee la librería *Admin SDK*.

Esto se visualiza en la siguiente figura:

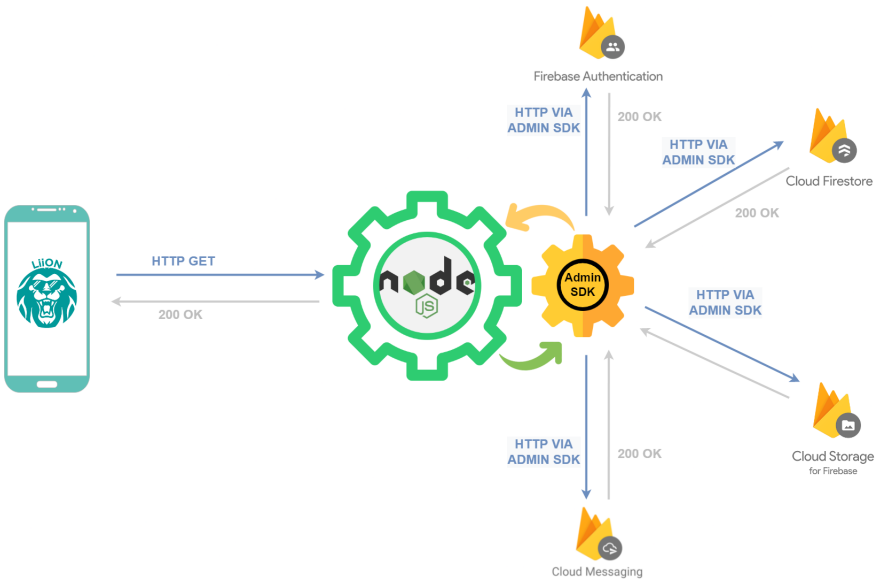


Figura 4.15: Arquitectura de la API

Como se puede observar, el cliente (aplicación Liion) interactúa con la API de Liion, y esta se encarga de hacer solicitudes a los diferentes servicios que se utilizan. Es importante mencionar que, las peticiones que se realizan a la API no son todas *GET* y también que el Admin SDK funciona como caja negra, por lo tanto, los ejemplos de la figura 4.15 son solamente ilustrativos.

4.6.2. Desarrollo de API de servicios

El *backend* de Liion está basado y ajustado al estándar HTTP vigente, que define los métodos y códigos de respuesta en el RFC 7231 [50] y RFC 5789 [51], basados originalmente en el RFC 2616 [52]. A continuación, se ordenan los métodos, en primer lugar por su función lógica dentro de la aplicación y en segundo lugar por tipo de método HTTP utilizado: *GET, POST, PUT, DELETE y PATCH.*

4.6.2.1. Guardián de sesión

Para asegurar que las peticiones sean solo ejecutadas por usuarios autenticados, todas las peticiones que llegan al *backend* de Liion, deben contener el token de sesión JWT [53] [54] proporcionado por Firebase Authentication en el *frontend*. Para esto se proveen 2 métodos análogos que verifican la validez del token. Uno obtiene el token desde el *Body* de la petición, y el otro desde el *Query String* (url).
En la figura 4.16 se muestra el flujo del guardián de sesión:

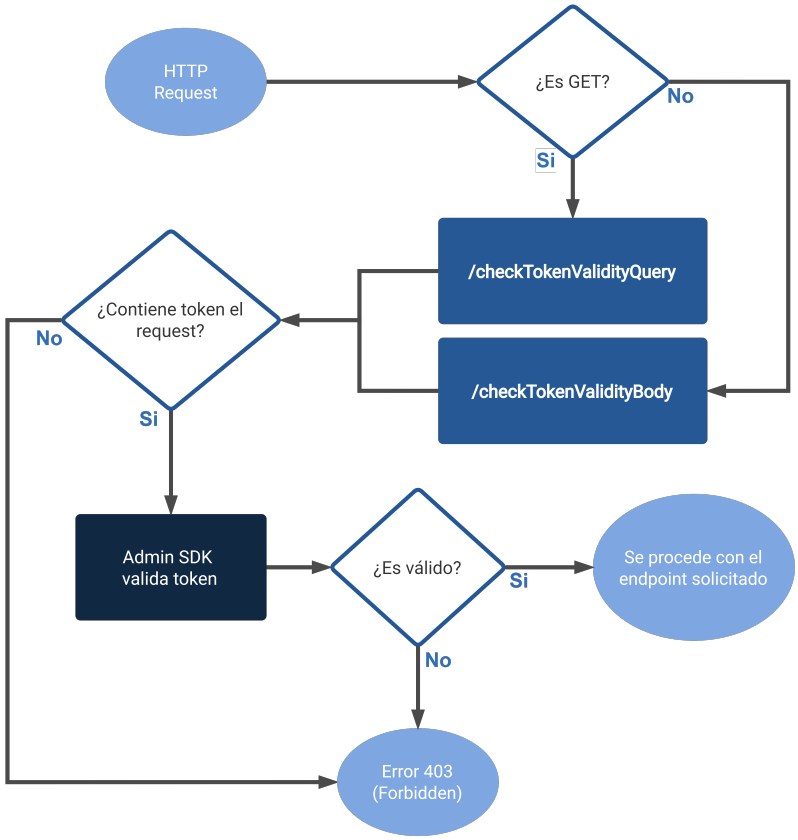


Figura 4.16: Flujo del guardián de sesión

Como se puede ver en la figura 4.16, si el método el cual recibe la llamada es un *GET* entonces se pasa el token por *Query*, si no es asi (*POST, PUT, DELETE y PATCH*) se recibe el token a través del *body* del *request*. Para ambos casos, si el token no existe, o no es válido, se devuelve un error, caso contrario, el guardián permite que el *request* sea recibido por el *endpoint* correspondiente.

4.6.2.2. Métodos GET

En el cuadro 4.16 se muestran los *endpoints* que usan el método *GET*.

Funcionalidad	Endpoint	Query
Obtener si el RUN se encuentra registrado	/getStatusRun	run
Obtener los datos de usuario	/getUserData	uid
Obtener viajes disponibles	*/getTravels*	genderApplicant, date, localityDestination, localityOrigin, time, uid
Obtener viajes vigentes de usuario pasajero	*/getTravelsPassenger*	passengerUID
Obtener viajes vigentes de usuario conductor	*/getTravelsDriver*	driverUID
Obtener itinerario del viaje para el usuario conductor	/getTravelItinerary	travelId
Obtener itinerario del viaje para el usuario pasajero	/getPassengerTravelItinerary	travelId, passengerUID
Obtener los usuarios con que se comparte el viaje	/getTravelPartners	travelId, userUid
Obtener detalles de un viaje	/getDetailsOfTravel	travelId
Obtener las coordenadas de la ruta de un viaje	/getRouteCoordinates	travelId
Obtener los viajes próximos a iniciar de un usuario conductor	/getUpcomingTravels	travelId

Cuadro 4.16: *Endpoints* de métodos *GET*

4.6.2.3. Métodos POST

En el cuadro 4.17 se muestran los *endpoints* que usan el método *POST*.

Funcionalidad	Endpoint	Body
Registrar usuario	*/register*	name, lastname, run, email, birth, password, gender, isDriver, photo64
Registrar solicitud de viaje	*/registerPassengerRequest*	travelId, passengerUID, extraBaggage, pickUp, dropOff, payMode
Registrar nuevo viaje	*/createTravel*	usefullTravelData 4.23
Registrar como conductor a un usuario regular	*/updateUsersDriverStatus*	uid, flagDriver, driverData 4.22
Generar notificaciones a usuarios de un viaje	*/notifToPassengers*	travelId

Cuadro 4.17: *Endpoints* de métodos *POST*

4.6.2.4. Métodos DELETE

En el cuadro 4.18 se muestran los *endpoints* que usan el método *DELETE*.

Funcionalidad	Endpoint	Body
Eliminar solicitud de viaje	*/deletePassengerRequest*	travelId, requestId
Eliminar viaje	*/deleteDriverTravel*	travelId

Cuadro 4.18: *Endpoints* de métodos *DELETE*

4.6.2.5. Métodos PATCH

En el cuadro 4.19 se muestran los *endpoints* que usan el método *PATCH*.

Funcionalidad	Endpoint	Body
Actualizar contador de visitas del viaje	/updateSeenTravel	travelId
Actualizar las coordenadas de un usuario en el viaje	/updateUserLocationInTravel	travelId, location, uid
Actualizar la puntación del usuario	*/updateUserRanting*	userList, travelId
Actualizar el estado del viaje	*/updateStateTravel*	travelId, state
Actualizar el token FCM del usuario	*/updateTokenFcm*	uid, fcmToken

Cuadro 4.19: *Endpoints* de métodos *PATCH*

4.6.2.6. Métodos PUT

En el cuadro 4.20 se muestran los *endpoints* que usan el método *PUT*.

Funcionalidad	Endpoint	Body
Actualizar el itinerario de un viaje en curso	*/updateTravelItinerary*	travelId, step, type, data

Cuadro 4.20: *Endpoints* de métodos *PUT*

4.6.3. Modelo de datos

Para los datos, como se define en la sección 4.2.3 en Liion, se utiliza una base de datos no relacional, en particular; de documentos, para almacenar los datos de usuario. En este caso se utiliza Firebase Firestore, ya que se puede utilizar como servicio, y permite acelerar el proceso de desarrollo.

Para archivos, en cambio, la historia es distinta, ya que no se debe tomar una decisión arquitectónica sobre que almacenamiento utilizar, las opciones son todas parecidas (Amazon S3 o Firebase Storage). Se elige Firebase Storage, ya que pertenece al ecosistema de Firebase, por lo que su integración resulta natural.

4.6.3.1. Almacenamiento de Archivos

Como se explica en la sección anterior, para almacenar archivos que no se pueden guardar en una base de datos se utiliza *Firebase Storage*.Dentro del contexto de la aplicación, se almacenarían datos de identificación de usuario, documentos de vehículo y fotos de perfiles. En cambio, el MPV no se consideran estos aspectos, sin embargo, se ha agregado la capacidad de subir fotos al perfil del usuario, estas se suben al momento de registrar un usuario y en la base de datos se guarda una referencia a esta.

4.6.3.2. Base de datos de Liion

La base de datos se Liion cuenta con 3 entidades principales, estas se encuentran con nombres en inglés, y se describen como:

1. **users**: Entidad que representa a todo usuario registrado en la aplicación, ya sea conductor o pasajero. Contiene datos referentes a la identidad de usuario y datos propios de un conductor o pasajero (como sus calificaciones).

2. **travels**: Entidad que representa a todos los viajes de la aplicación, incluyendo viajes ya cursados, por ocurrir y abortados, en conjunto con datos de este.

3. **requestTravel**: Entidad que representa las peticiones de pasajeros a un viaje determinado, pensado para implementaciones futuras.

A continuación, se presentan en detalle cada una de ellas.

4.6.3.3. Users

La entidad de usuario tiene la siguiente estructura.

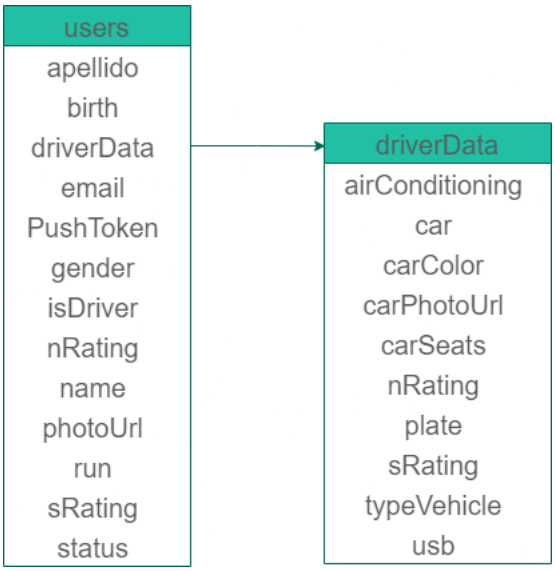


Figura 4.17: Entidad users

Llave	Tipo	Significado	Ejemplo
name	String	-	‘Claulos’
lastName	String	-	‘Elzano’
birth	String	-	‘1935-01-01’
email	String	-	‘Claulos@hub.com’
run	String	-	‘12345678-9’
gender	String	-	‘Mujer’,‘Hombre’, ‘Ninguno de los anteriores’
isDriver	Boolean	-	true
driverData	Object	-	Referencia a tabla 4.22
photoUrl	String	-	‘https://storage.googleapis.com/liiAQFJ7j’
pushToken	String	Identificación del dispositivo del usuario, para el uso de *FCM*	‘d9ta:Ur3_5G4MQCs-N’
nRating	Number	Número de calificaciones recibidas	5
sRating	Number	Sumatoria de todas las calificaciones recibidas	50
status	String	Estado que indica si como pasajero tiene algún viaje en curso	‘travelOn’, ‘travelOff’

Cuadro 4.21: Entidad users

Al registrarse un usuario, este se guarda con *isDriver* en falso, y con el sub-documento *driverData* en vacío. Una vez este accede a la sección de alta de conductor y completa el proceso exitosamente, se modifica el subcampo *isDriver* a verdadero y se llena el documento *driverData* con datos inherentes del conductor, apreciables en la tabla 4.22.

Llave	Tipo	Ejemplo
plate	String	‘DL-YR-01’
car	String	‘Toyota AE86’
carColor	String	‘gris’
carPhoto	String	‘https://storage.googleapis.com/liiAQFJ7j’
carSeats	Number	3
typeVehicle	String	‘Suv’, ‘Sedan’, etc.
airConditioning	Boolean	true
usb	Boolean	false
nRating	Number	10
sRating	Number	5

Cuadro 4.22: Sub entidad driverData

4.6.3.4. Travels

La entidad de viajes tiene la siguiente estructura:

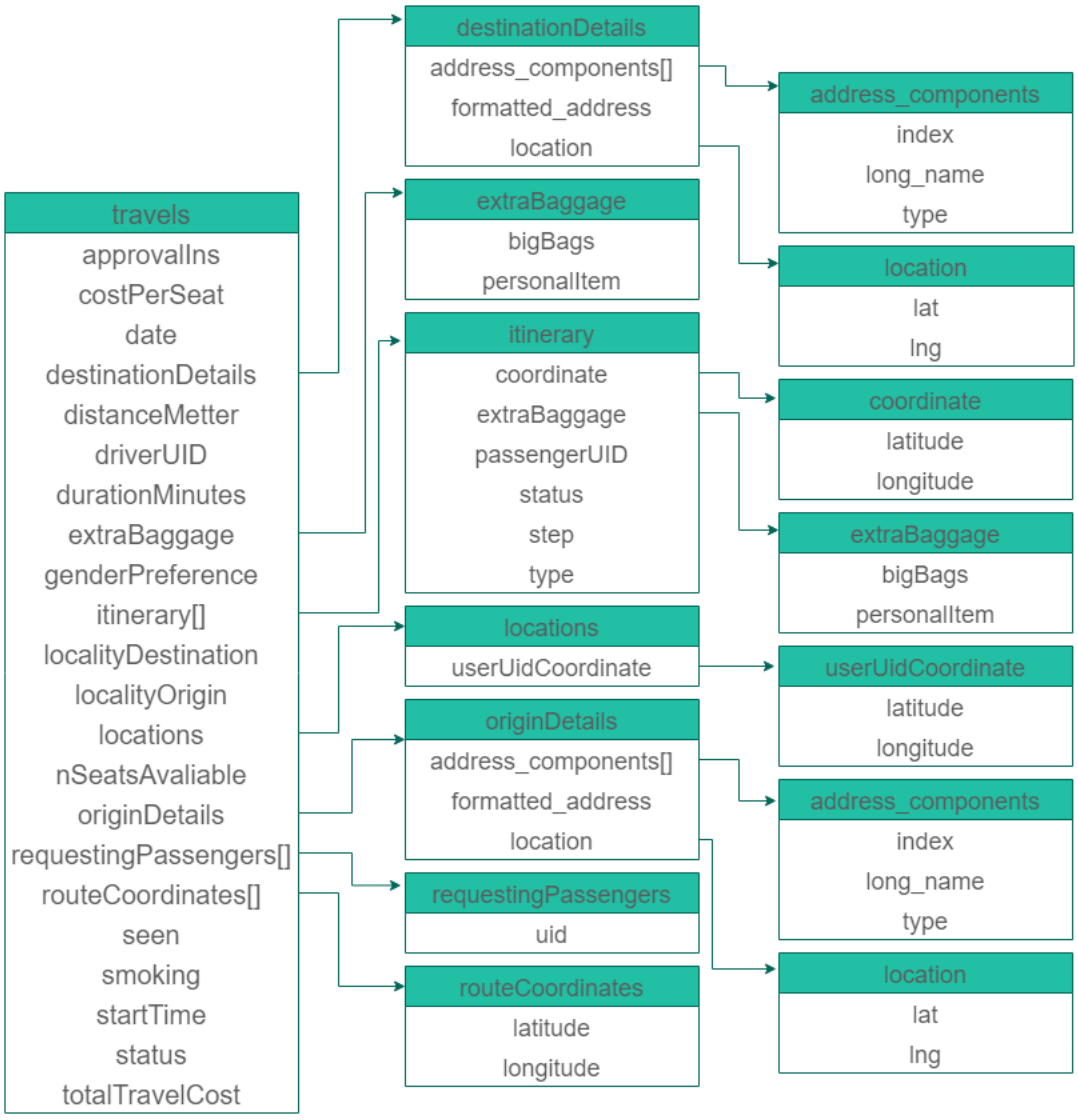


Figura 4.18: Entidad travels

Llave	Tipo	Significado	Ejemplo
approvalIns	Boolean	Inscripción al viaje son aceptadas automáticamente	true
costPerSeat	Number	-	500
destinationDetails	Object	-	Referencia a tabla 4.24
originDetails	Object	-	Referencia a tabla 4.24
distanceMetter	Number	Distancia viaje [m]	128.56
driverUID	String	-	‘daMgFnFFCsoED2’
durationMinutes	Number	Duracion viaje [min]	93.51
extraBaggage	Object	-	Referencia a tabla 4.25
genderPreference	String	Distinción por género	‘Allgender’,‘Onlymen’ ‘Onlywoman’
itinerary	Array<Object>	Puntos donde el conductor debe detenerse.	Referencia a tabla 4.26
localityDestination	String	-	‘Sprinfield’
locationOrigin	String	-	‘Calama’
locations	Object	Ubicaciones de los usuarios del viaje en curso	{{UID_1 : latitude, longitude} {UID_2 : latitude, longitude}}
nSteatsAvaliable	Number	Asientos disponibles	3
nSeatsOffered	Number	Asientos totales	5
date	String	-	‘20/01/1945’
routeCoordinates	Array<Object>	Coordenadas de la ruta	[{ latitude’: -37.45, ‘longitude’: -71.3 },...]
seen	Number	Visualizaciones del viaje	2
smoking	Boolean	-	false
startTime	String	-	‘01:02’
status	String	Estado general del viaje	‘open’, ‘closed’, ‘ongoing’, ‘finished’, ‘aborted’
totalTravelCost	Number	-	$10.111
requestingPassengers	Array<String>	UID de los usuarios inscritos (Request Travel)	[‘daMgJNbJF2’, ‘BpJhjJi9’]

Cuadro 4.23: Entidad travels

El campo *requestingPassengers*, se inicia como vacío, y este se llena a medida que usuarios aplican al viaje. De la misma manera, el *itinerary* se crea al momento de iniciar un viaje por parte del conductor.

A continuación se detallan los objetos anidados a la entidad:

Llave	**Tipo**	**Ejemplo**
address_components	Array<Object>	Referencia a tabla 4.27
formatted_address	String	'Badajoz, Región Metropolitana, Chile'
location	Object	{ latitude': -37.45,'longitude': -71.3 }

Cuadro 4.24: Sub entidad destinationDetails y originDetails

Llave	**Tipo**	**Significado**	**Ejemplo**
bigBags	Number	Número de maletas grandes extras que pueden haber en el viaje	1
personalItem	Number	Numero de artículo personal extra que pueden haber en el viaje	1

Cuadro 4.25: Sub entidad extraBaggage

Llave	Tipo	Significado	Ejemplo
coordinate	Object	Coordenada de la parada	{ latitude': -37.45,'longitude': -71.3 }
extraBaggage	Object	-	Referencia a tabla 4.25
passengerUID	String	UID de pasajero	'daMgFnFM0txAMFCsoED2'
status	String	Estado del itinerario para cada pasajero, en particular: active: usuario pronto a bajar o subir deactive: estado por defecto finished: ya se pasó por este punto	'active', 'deactive', 'finished'
type	String	Tipo de parada	'pickUp', 'dropOff'
step	Number	Etapa del itinerario en la que va el viaje	1

Cuadro 4.26: Sub entidad itinerary

A continuación se detallan los objetos anidados en un nivel más de profundidad:

Llave	Tipo	Significado	Ejemplo
index	Number	-	1
long_name	String	-	'Las Condes'
type	String	Nombre que maneja google directions internamente	'locality'

Cuadro 4.27: Sub sub entidad address_components

4.6.3.5. Request Travel

La entidad de requestTravel tiene la siguiente estructura, apreciable en la figura 4.19:

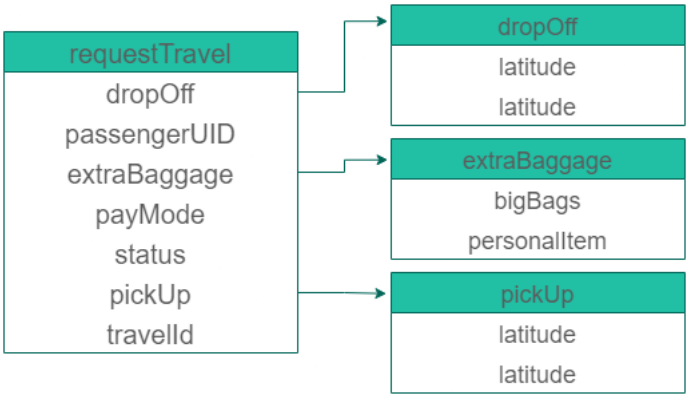


Figura 4.19: Entidad request travels

Llave	Tipo	Significado	Ejemplo
dropOff	Object	Coordenada de bajada	{ latitude': -37.45, 'longitude': -71.3 }
passengerUID	String	Id único de pasajero	'daMgFnFM0tVNkPIxAMFCsoED2'
extraBaggage	Object	-	Referencia a tabla 4.25
payMode	String	-	'Debito', 'Credito', 'ShibaInu'
status	String	accepted: Pasajero aceptado en viaje rejected: Pasajero no aceptado en viaje	'accepted', 'rejected'
pickUp	Object	Coordenada de subida	{ latitude': -37.45, 'longitude': -71.3 }
travelId	String	Id único de viaje	'Wv0AXfydF7FKxX2dfypEAy2Gc2'

Cuadro 4.28: Sub entidad requestTravel

4.6.3.6. Vista General de la base de datos

Finalmente, en la figura 4.20 se puede apreciar la estructura general de la base de datos de *Liion*.

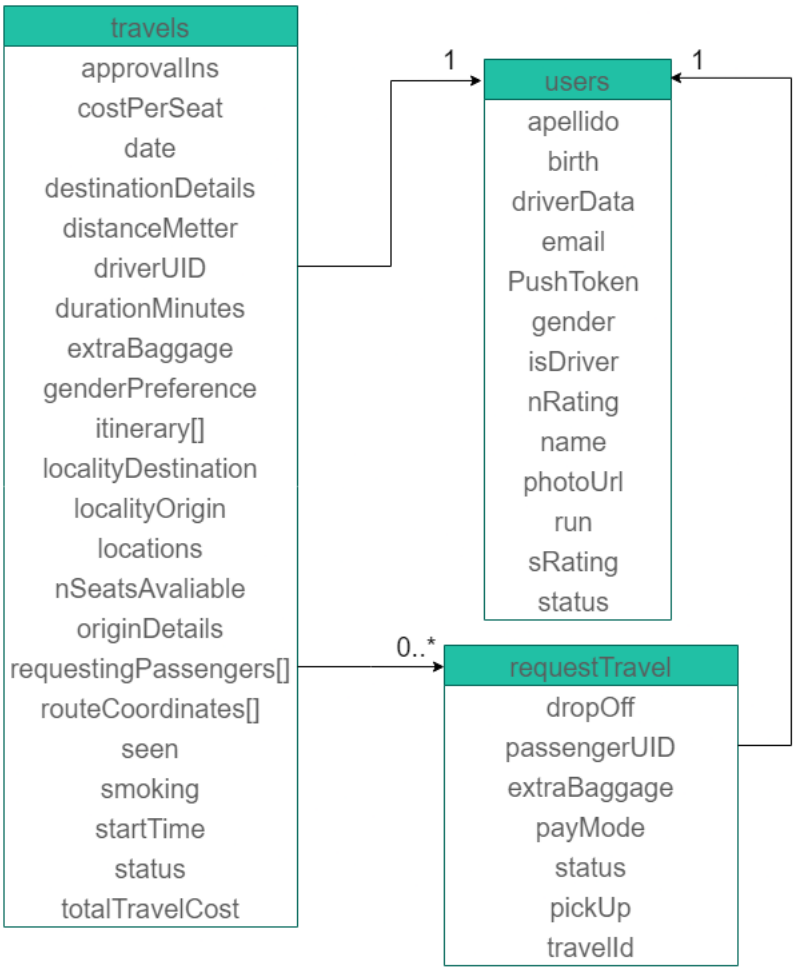


Figura 4.20: Vista general de la base de datos de *Liion*

4.7. Capa de presentación

En esta sección se definen diferentes flujos y/o casos de uso que definen el comportamiento de la aplicación, esto permite conocer como funciona el proyecto, estableciendo, las diferentes interacciones del *frontEnd* con la *API* (*backEnd*).

En la figura 4.21 (de elaboración propia), es apreciable el modelo de navegación de la capa de presentación, este sirve para describir de manera general el flujo presente en la aplicación y mostrar los posibles caminos que el usuario puede tomar, además da una visión de las diferentes funcionalidades. Como los procesos en la aplicación son complejos, se generalizaron en diferentes módulos.

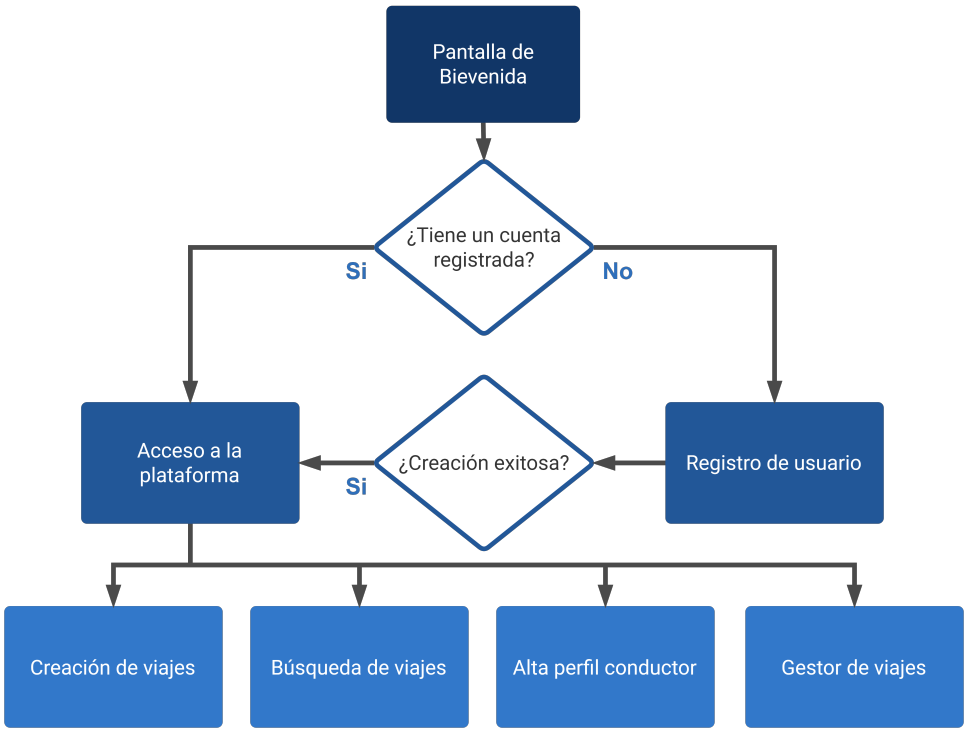


Figura 4.21: Modelo navegación de las funcionalidades principales

4.7.1. Flujos de navegación y/o casos de uso

En Liion, y al igual que cualquier otra plataforma informática, existen variados procesos por medio de los cuales el usuario interactúa con la aplicación. Dichos procesos, corresponden a los pasos a seguir para completar la tarea de un componente en específico. Si bien, estos flujos para el usuario serán intuitivos o incluso desapercibidos, se hace relevante poder visualizar por medio de una documentación detallada cada paso llevado a cabo, con el fin de dar a conocer el funcionamiento de la aplicación.

4.7.1.1. Registro de usuarios

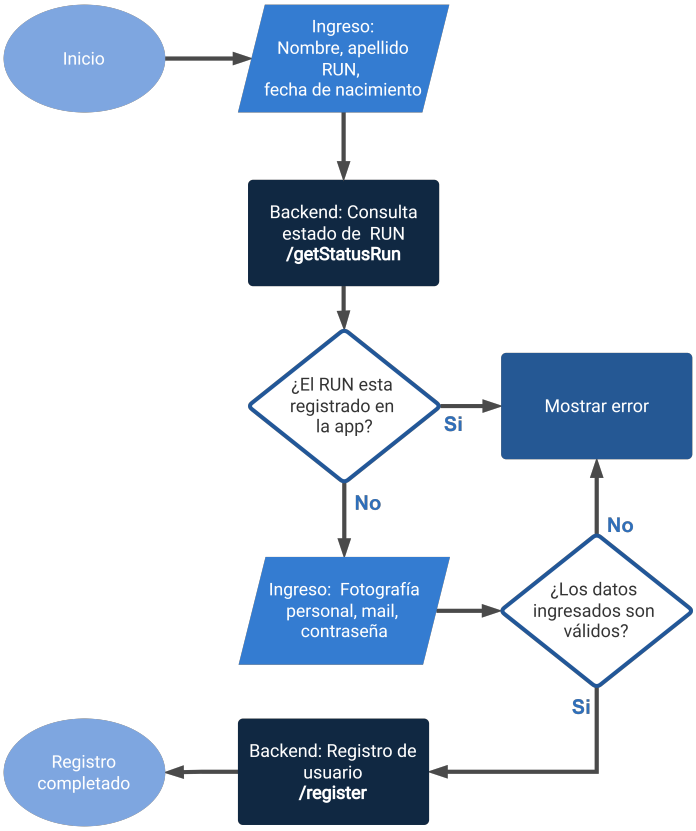


Figura 4.22: Diagrama de flujo para el registro de usuarios.

4.7.1.2. Acceso a la plataforma

El usuario, al ingresar a la aplicación, se puede encontrar con dos situaciones:

- Registrar ya un ingreso exitoso sin haber cerrado sesión.
- Nunca haber ingresado o habiendo cerrado sesión.

En el primer caso, la biblioteca de Firebase para React Native permite que los datos de sesión perduren en el dispositivo, por lo que el flujo de la aplicación hace que se omita todo el proceso de ingreso que implique entrada de datos por parte del usuario, y procede a cargar los datos refrescados desde el Backend. Este proceso es apreciable en la figura 4.23

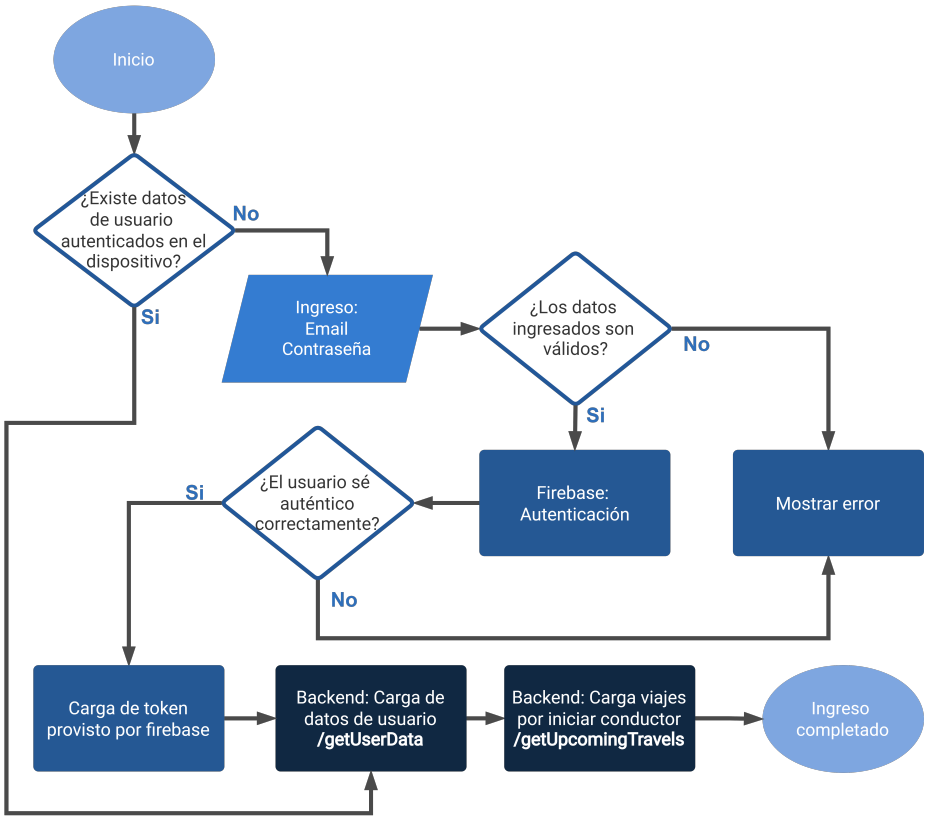


Figura 4.23: Diagrama de flujo para el ingreso a la plataforma.

4.7.1.3. Alta perfil conductor

El proceso de alta de perfil de conductor, es un paso esencial para habilitar la creación de viajes. Si bien el MPV, no incluye la validación de los documentos de conducción, el flujo que de acciones sería similar, añadiendo una etapa de validación, la cual podría ser automática o manual. Al completar el alta de conductor, al usuario se le asignará un vehículo y sus características preestablecidas (el registro de vehículo se encuentra fuera del MPV), con el cual podrá crear un viaje.

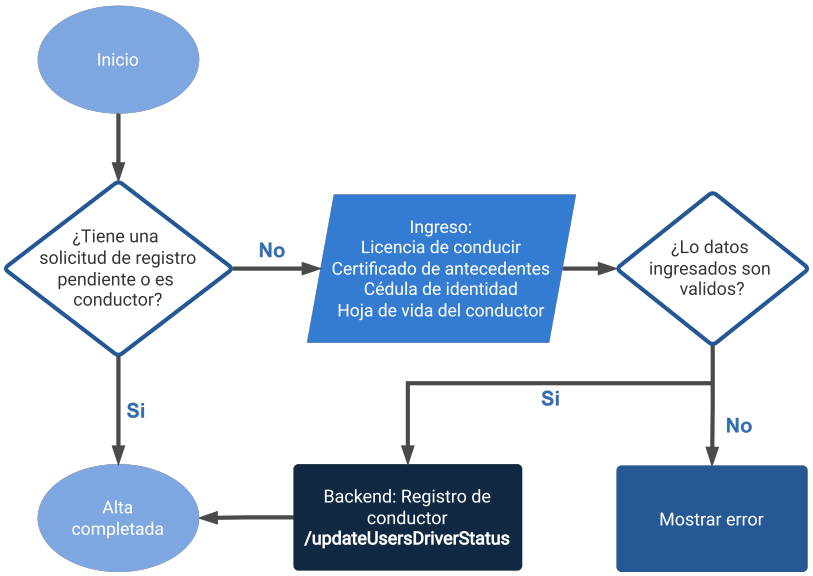


Figura 4.24: Diagrama de flujo para el alta perfil conductor.

4.7.1.4. Creación de viajes

La creación de viajes, solo estará habilitada para los perfiles de usuarios que hayan completado el alta de perfil de conductor, debido a que, esta depende del auto que registre el usuario y sus diversos parámetros (como rendimiento por litro de combustible, comodidades, etc.).

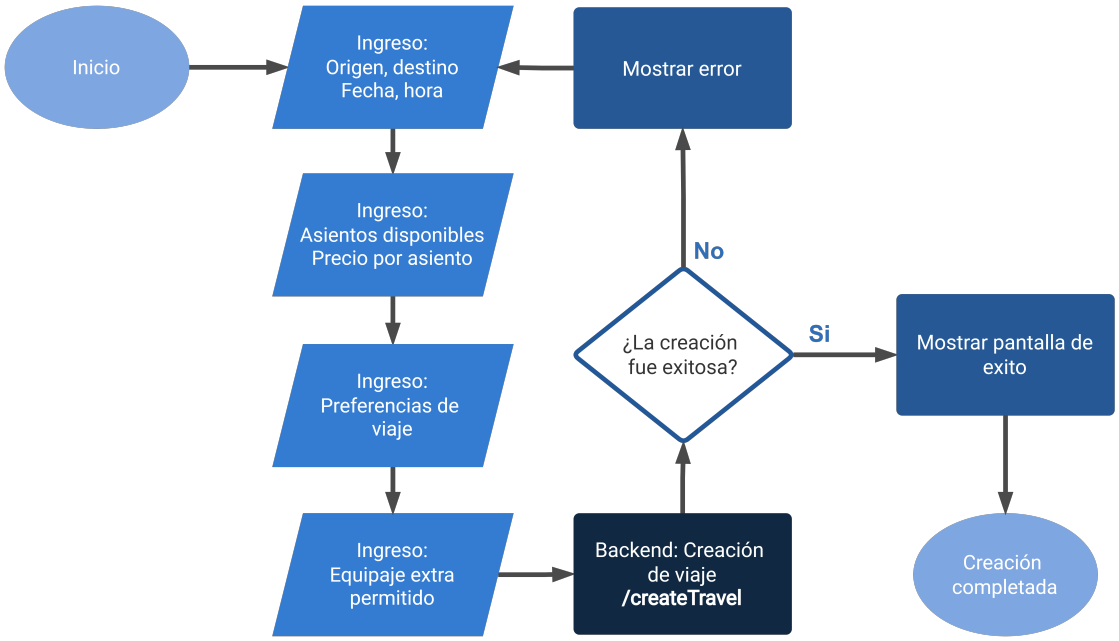


Figura 4.25: Diagrama de flujo para la creación de viajes.

4.7.1.5. Búsqueda de viajes

La búsqueda de viajes es un proceso transversal a todos los usuarios de la plataforma, por lo tanto, tanto conductores como pasajeros tienen esta posibilidad. Es importante destacar que la 'búsqueda' (figura 4.26) entregará todos los viajes que coincidan exactamente con las comunas objetivas señaladas, en origen y en el destino, considerando además de parámetros como la hora, género del usuario, etc.

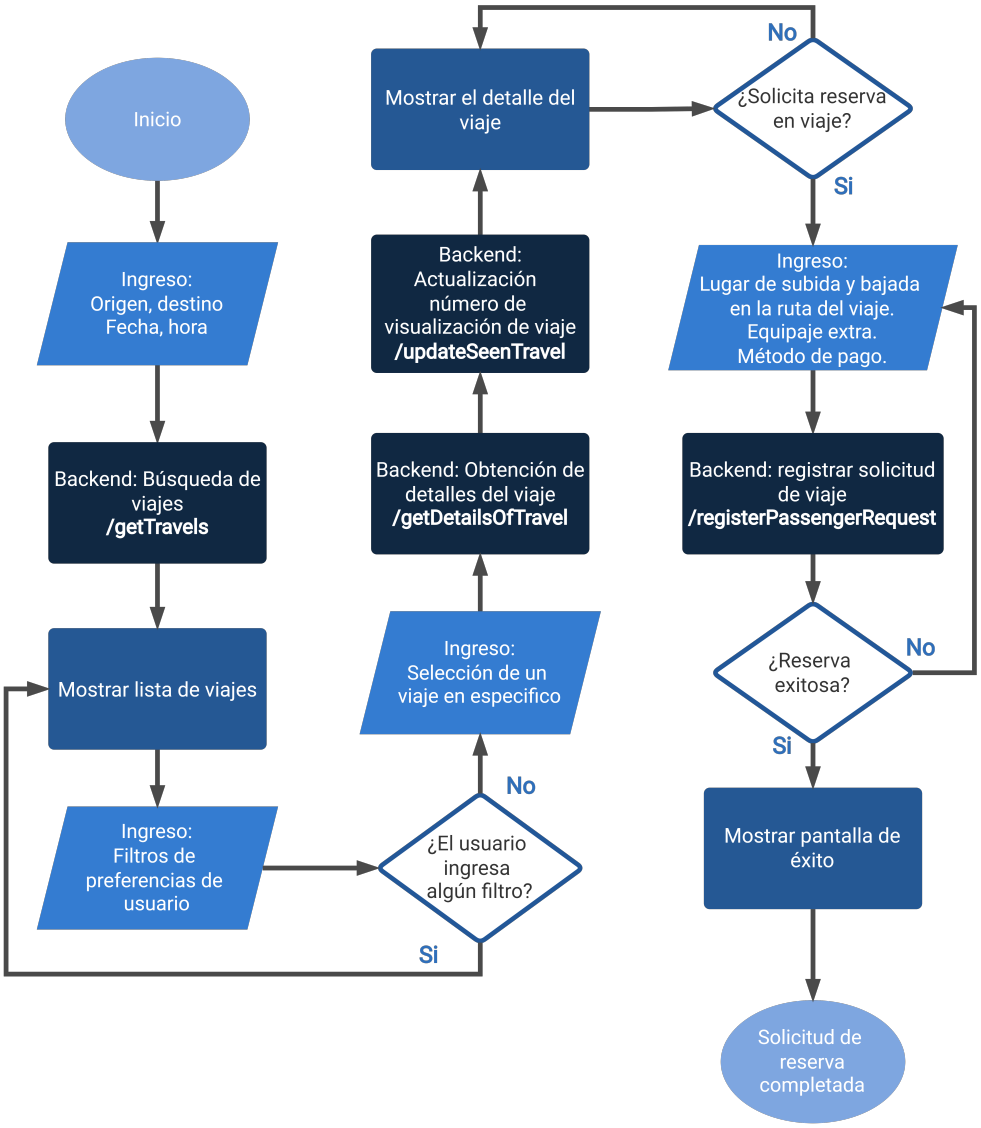


Figura 4.26: Diagrama de flujo para la búsqueda de viajes.

4.7.1.6. Gestor de viajes

Es el encargado de administrar todos los viajes que los usuarios crean o en los cuales se subscriben. Es importante señalar que el funcionamiento de este, apreciable en la figura 4.27, dependerá del perfil que lo visualice, dándole además diferentes posibilidades de administración. Por lo tanto, se tendrán dos casos:

- Conductor visualizando sus viajes creados, donde podrá eliminar pasajeros que hayan reservado, iniciar y cancelar el viaje.
- Pasajero visualizando los viajes donde haya reservado, donde podrá cancelar la reserva.

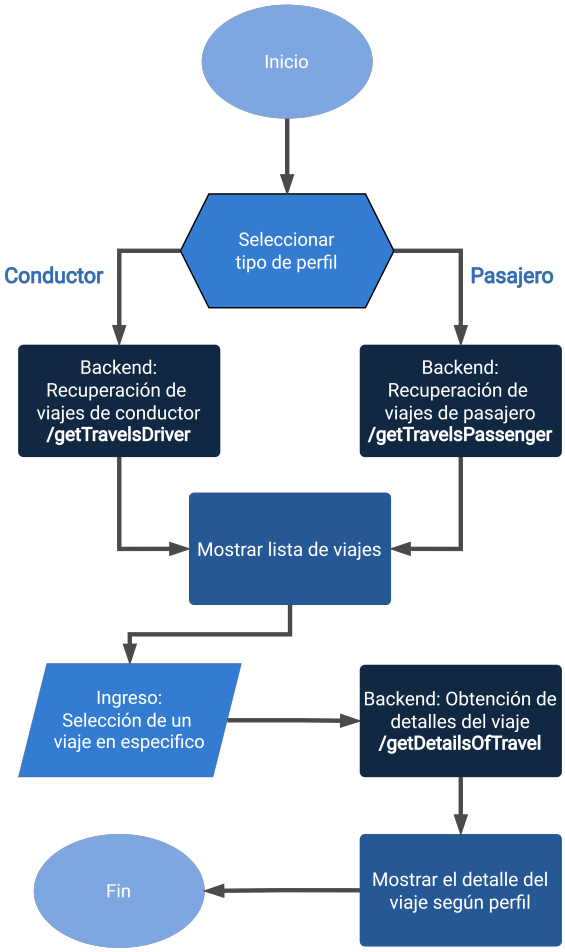


Figura 4.27: Gestos de viajes general para los usuarios de la aplicación.

4.7.1.7. Viaje en proceso

La figura 4.28 detalla la seguidilla de eventos que ocurren al iniciar un viaje, (gatillados por el usuario conductor, estando en el 'gestor de viajes'). La vista de 'viaje en proceso' se caracteriza por ser en tiempo real, utilizando un sistema de *pulling* que cada cierta cantidad de tiempo o a partir de eventos específicos (dependiendo del tipo de usuario), consulta a la *API* la etapa del viaje.

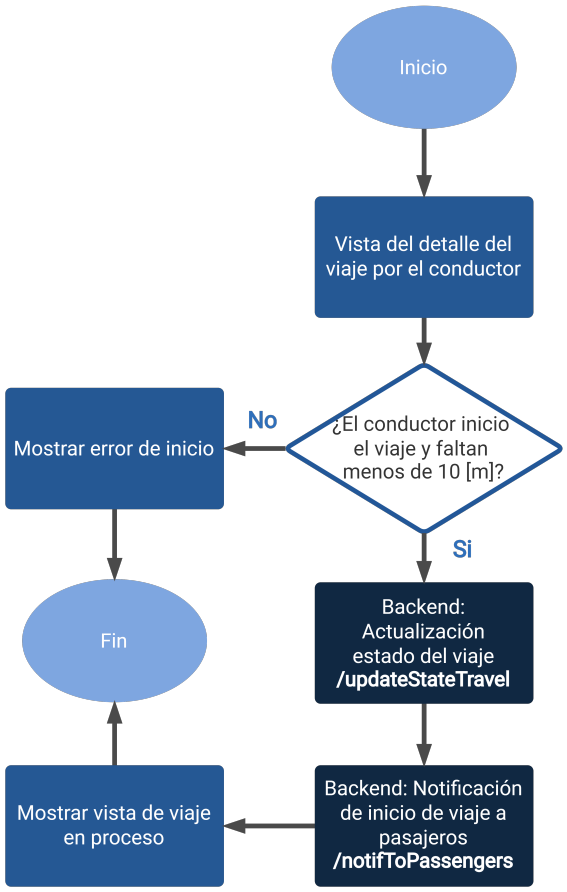


Figura 4.28: Diagrama de flujo del proceso de inicio de viaje.

Las figuras 4.29 y 4.30, muestran los diferentes estados en la 'vista de viaje en proceso' en un viaje con un pasajero, tanto desde la perspectiva del conductor como la del pasajero respectivamente. Como es explicado en 6.1.0.10, al crearse el itinerario, los puntos de subidas y bajadas se ordenan de modo tal que los usuarios se suban y bajen ordenadamente, sin hacer al conductor devolverse en su ruta.

La figura 4.29, correspondiente al diagrama de estado de la vista de conductor, se muestra que existen tres estados, los que corresponden a la subida y bajada del pasajero más una etapa final de *feedback*. El cambio de estado depende exclusivamente de la validación del código QR del pasajero por parte del conductor y el registro del evento de bajada, también dependiente del conductor. Es importante notar dos cosas, la primera, que al agregar otro pasajero se deberían agregar dos estados más y que siempre los estados de 'subida' irán en un principio del itinerario, seguidos de todos los estados de 'bajada', ordenados en la ruta del conductor, ya que en esta versión de la aplicación los viajes son desde una comuna de origen a otra de destino, sin permitir paradas intermedias.

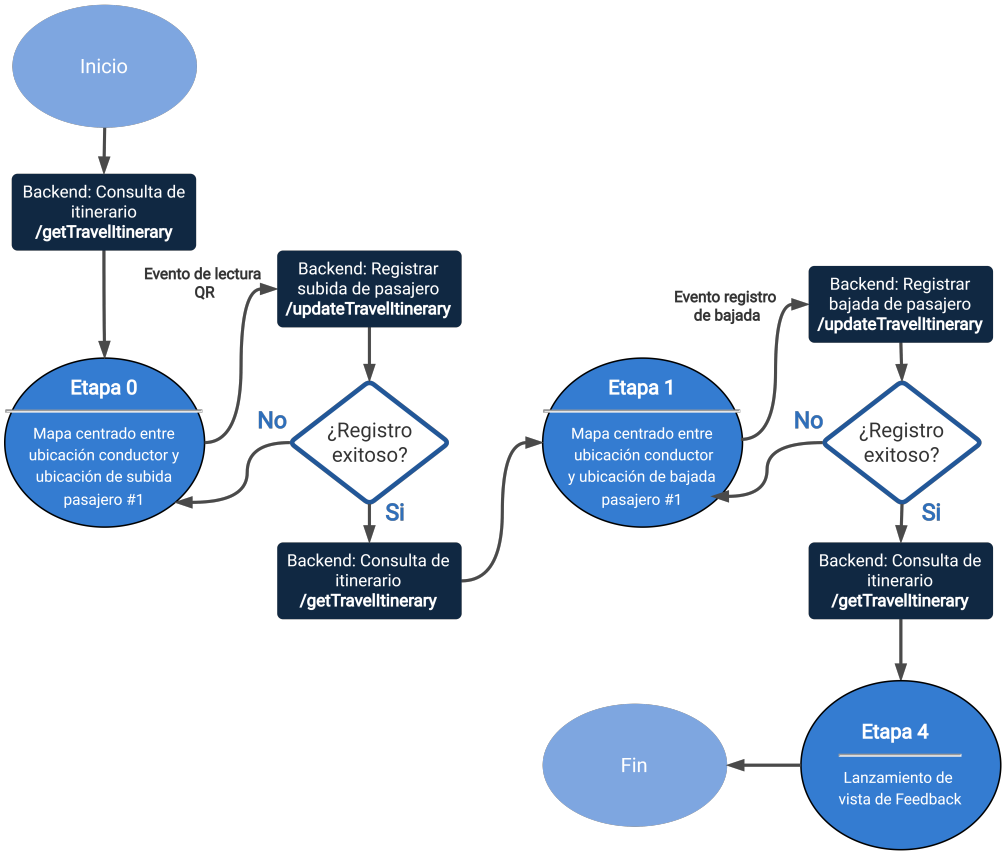


Figura 4.29: Etapas del itinerario para la vista del conductor.

La figura 4.30, correspondiente al diagrama de estado de la vista de pasajero, muestra que se tienen tres etapas, esperando, ‘subir’ y ‘bajar’ del vehículo, además de una etapa final de *feedback*. Es importante destacar que el cambio de estado depende de la validación por parte del conductor, del código QR mostrado por el pasajero al momento de subir al vehículo y de la confirmación de bajada, también dependiente del conductor.

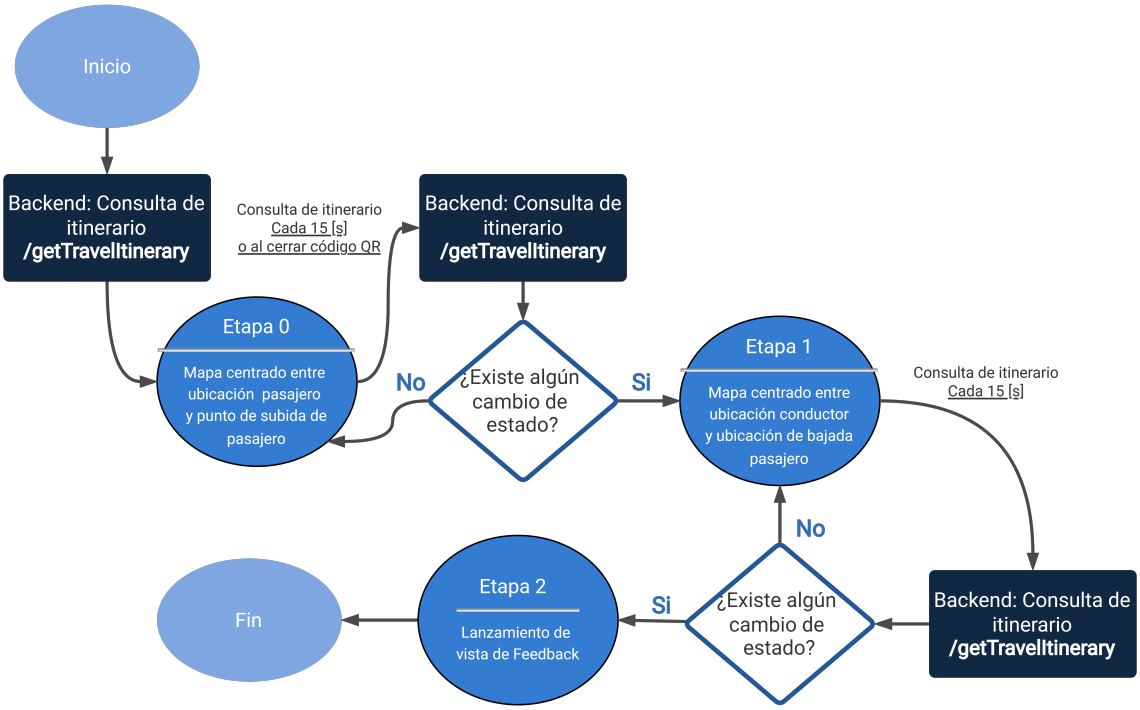


Figura 4.30: Etapas del itinerario para la vista del pasajero.

Capítulo 5

Resultados

En esta sección se muestran las representaciones finales de lo explicado en la sección 4.7.1. Estas correspondes a capturas de pantalla directas desde la aplicación corriendo en un celular con Android 10, en un ambiente de pruebas local.

5.1. Acceso a la plataforma

La figura 5.1 muestra lo representado en la sección 4.7.1.2.

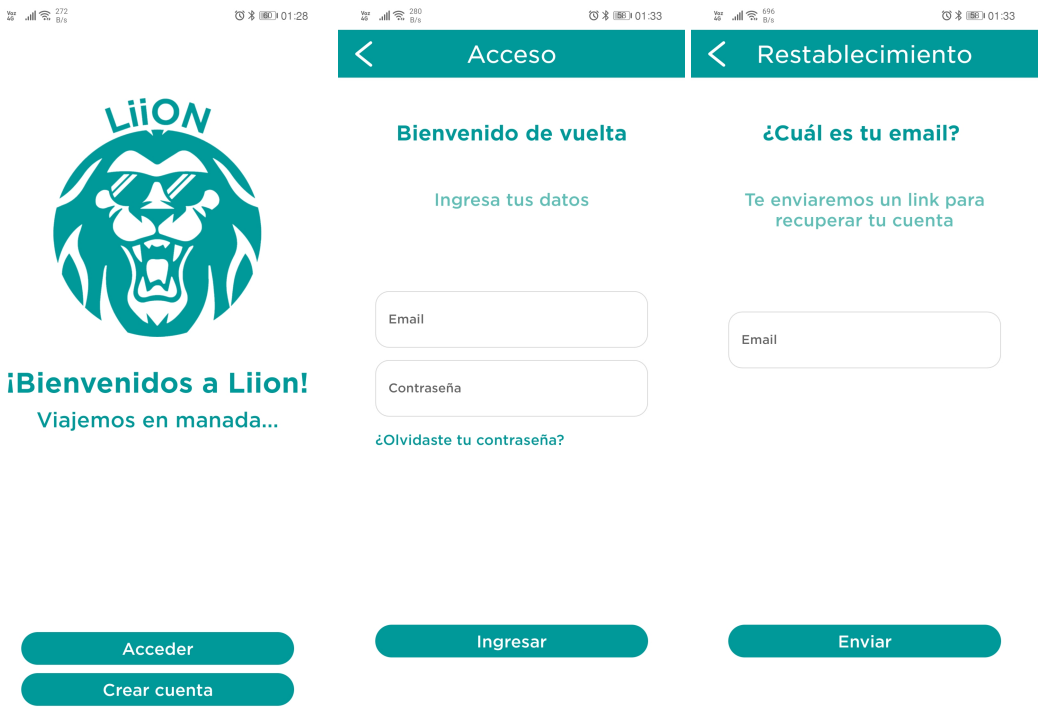


Figura 5.1: Vista del ingreso a plataforma y recuperación de cuenta.

5.2. Registro de usuario

La figura 5.2 muestra lo representado en la sección 4.7.1.1.

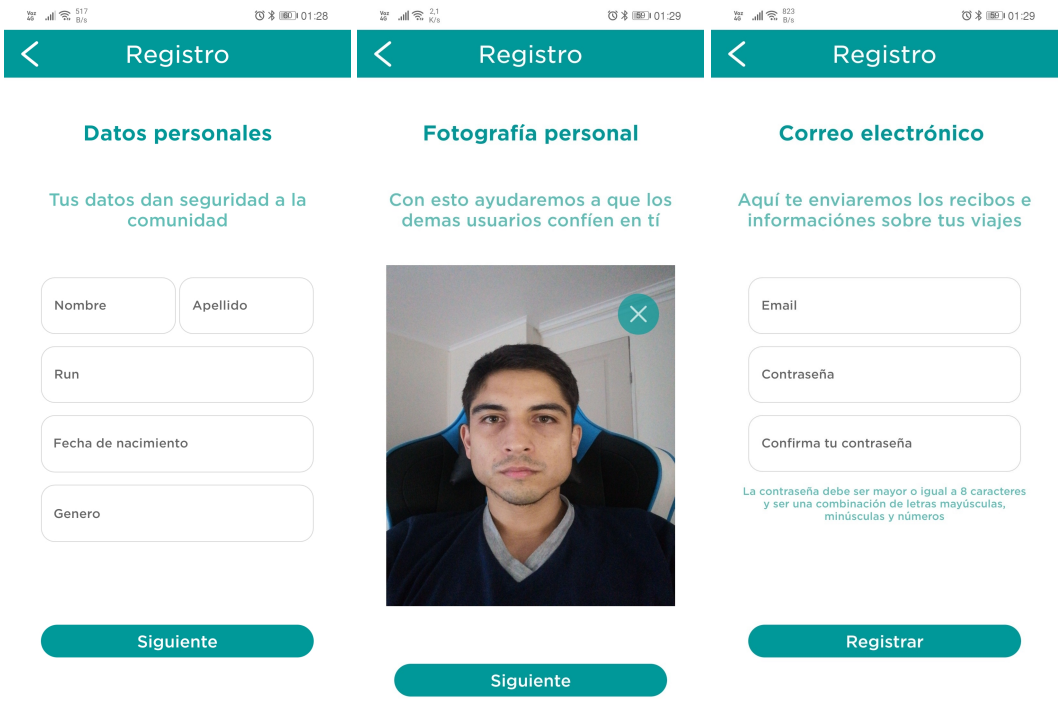


Figura 5.2: Vistas del registro de usuario.

5.3. Alta de perfil de conductor

La figura 5.3 muestra lo representado en la sección 4.7.1.3.

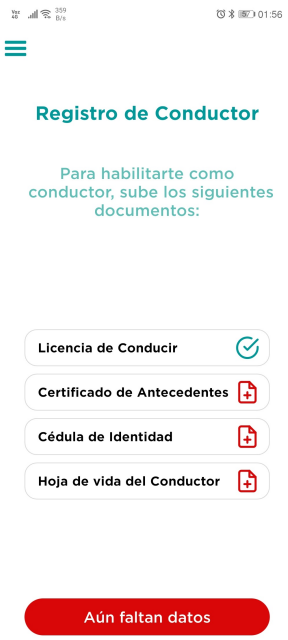


Figura 5.3: Vista del alta de conductor.

5.4. Creación de viaje

La figura 5.4 muestra lo representado en la sección 4.7.1.4.



Figura 5.4: Vista de la creación de viajes.

5.5. Búsqueda de viajes

La figura 5.5 muestra lo representado en la sección 4.7.1.5.



Figura 5.5: Vista de la búsqueda de viajes.

5.6. Gestor de viajes

Las figuras 5.6 y 5.7 muestra lo representado en la sección 4.7.1.6

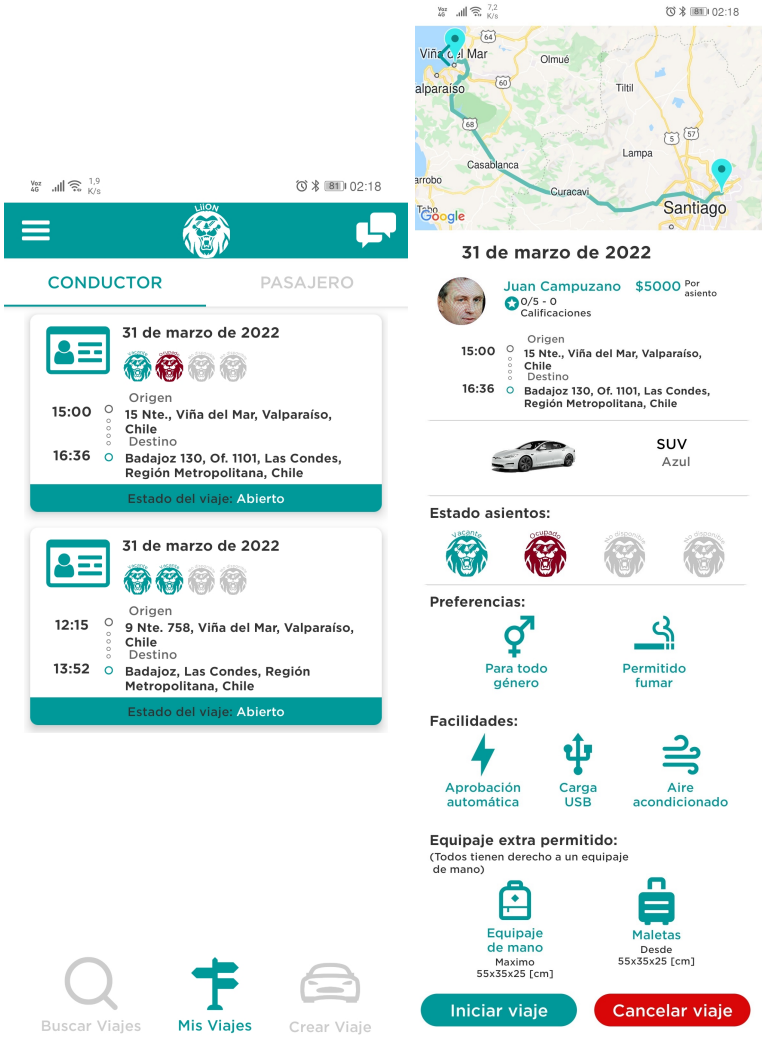


Figura 5.6: Gestor de viajes desde la perspectiva conductor.

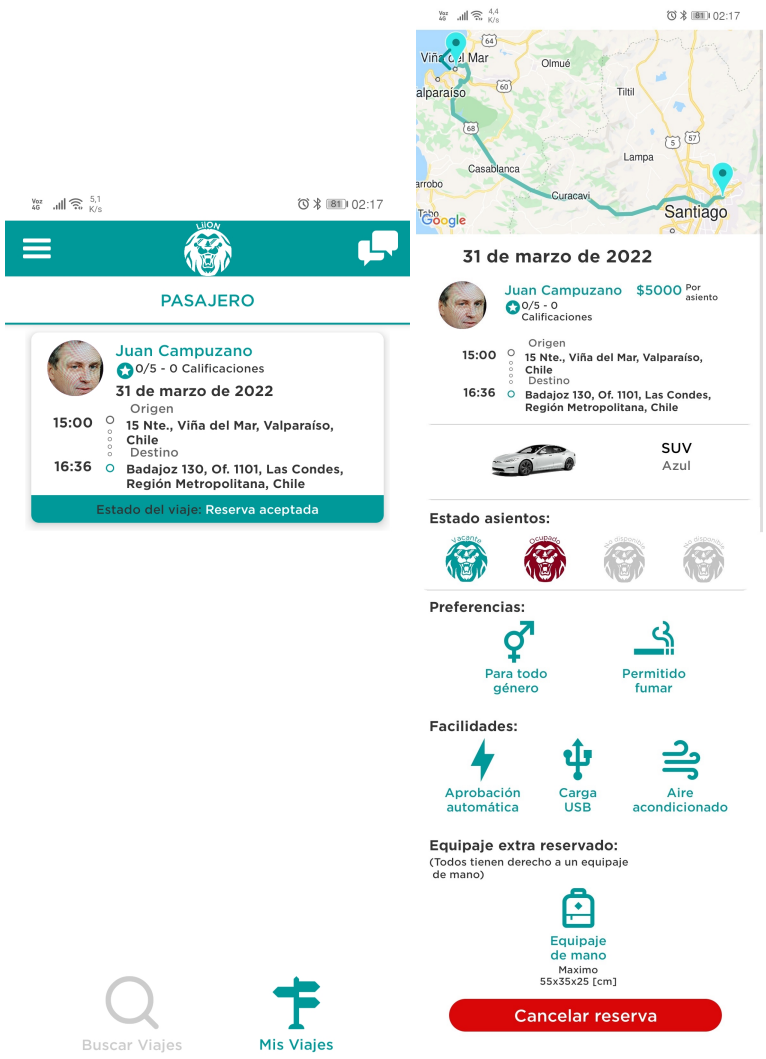


Figura 5.7: Gestor de viajes desde la perspectiva pasajero.

5.7. Viaje en proceso

Las figuras 5.8 y 5.9 muestran lo representado en la sección 4.7.1.7 para cada tipo de usuario.

5.7.1. Conductor

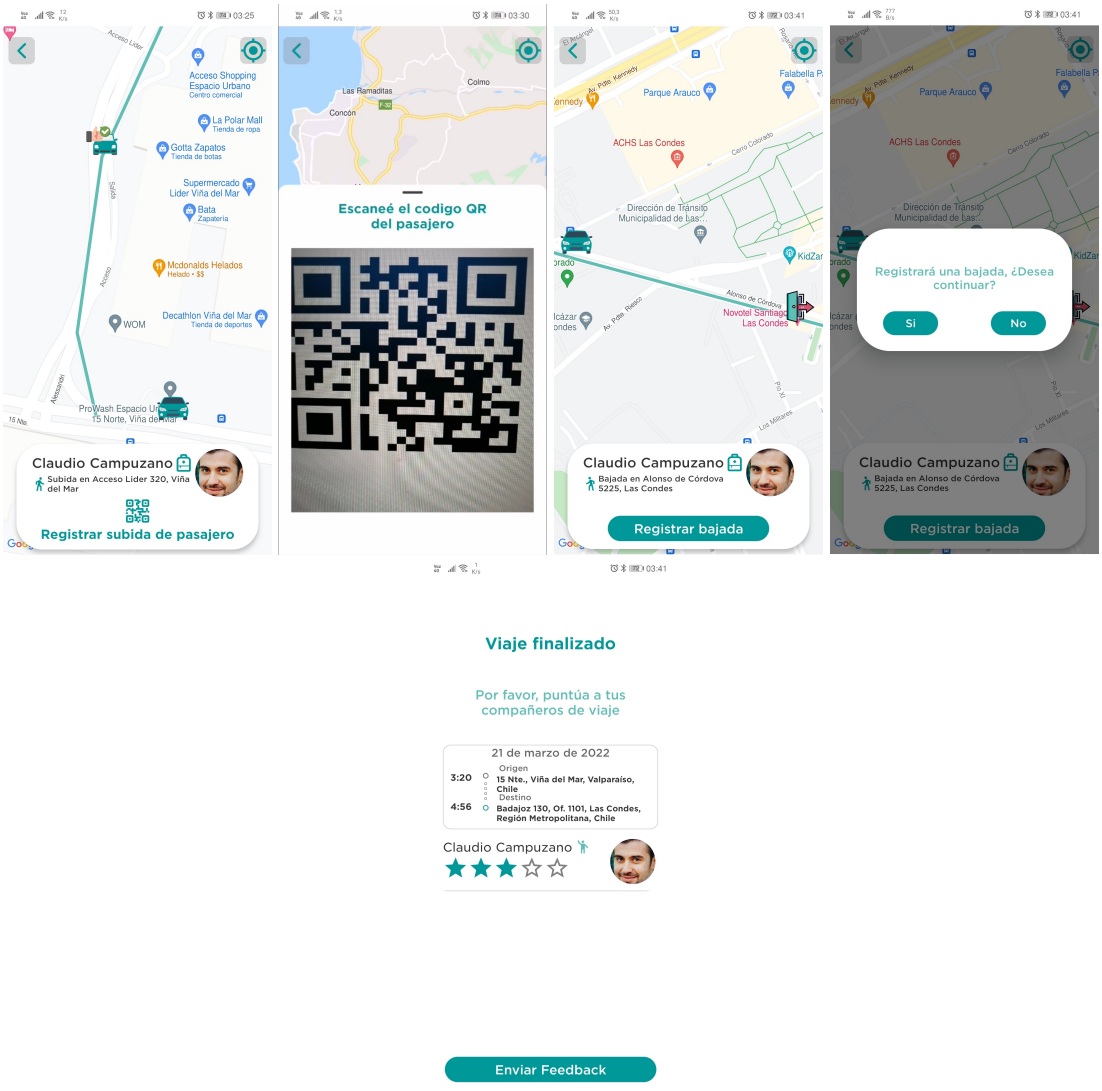


Figura 5.8: Viaje en proceso desde la perspectiva pasajero.

5.7.2. Pasajero

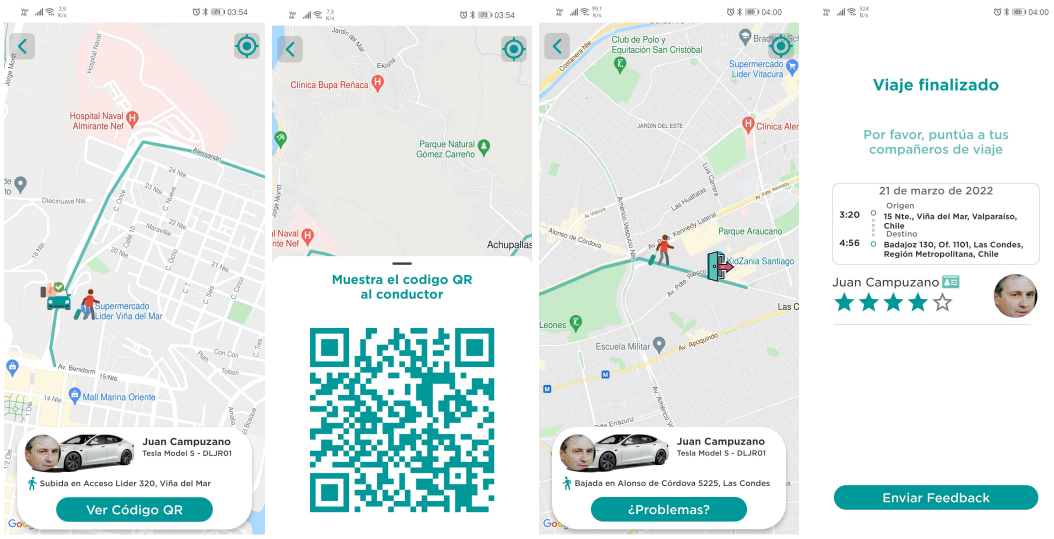


Figura 5.9: Viaje en proceso desde la perspectiva pasajero.

Capítulo 6

Conclusiones

Si bien, el *carpooling* es presentado como la solución de los problemas de transporte interurbano, en realidad, no es así, o al menos no del todo. A partir de los estudios presentados en los anteriores capítulos, se puede afirmar que el *carpool* efectivamente descongestiona las vías, reduce las emisiones de gases de invernadero y agiliza los viajes de aquellas personas que lo utilizan, en contraste con el viaje interurbano en buses; sin embargo, también se han descubierto importantes problemas, como por ejemplo, que exista una desconfianza recíproca entre conductores y pasajeros, lo que podría dificultar el comienzo a aquellos usuarios que no tengan puntuaciones y/o experiencia en la plataforma. Además, considerando el contexto actual de pandemia (por normas sanitarias) debe haber una separación mínima entre personas, derivando en que no se puedan utilizar todos los asientos, reduciendo uno de sus beneficios clave, como la descongestión vehicular.

Sin embargo, y a pesar de que Liion no resuelve todas las problemáticas planteadas, si es una solución viable a implementar en Chile, ya que tecnológicamente se consigue llegar a un MPV. Además es una solución que sería potencialmente usada por un público objetivo 2.6 .

Es importante señalar que en el ámbito legal no existe ningún problema en que exista una plataforma basada en la compartición de viajes, ya que estas se encuentran

no reguladas. En la práctica, no se avizora una normativa para dichos efectos, a pesar exista un proyecto de ley que pretenda regular todas las formas de transporte digitales 6.1.0.10.

Actualmente, Liion tiene las funcionalidades propuestas en el MPV, se puede autenticar, crear y unirse a viajes, además cuenta con sistema de notificaciones en tiempo real que avisa a los pasajeros cuando su viaje ha sido iniciado. Uno de las funciones más importantes (extraído de las opiniones de los usuarios) hace referencia a la desconfianza entre estos, lo cual Liion resuelve con dos mecanismos, en primer lugar, la puntuación de usuarios que permite democratizar la aplicación, identificando a los usuarios confiables, y en segundo lugar, validando a los pasajeros a través de un *código QR* (al momento de abordar un viaje) logrando con esto que sea sumamente difícil que un usuario este sea suplantado, ya que el código es único para cada par viaje-pasajero.

Es importante destacar que existen variadas características importantes que han quedado fuera del MPV, las cuales formarían parte del producto final, ya que ayudan a desmarcarse de las demás aplicaciones de *carpooling* y así, encontrar una propuesta de valor única, estas características son: notificaciones en caso de emergencia a cercanos y policías, operación en modo taxi para recogidas de pasajeros fuera de la ruta del viaje, solicitudes de viajes por partes de pasajeros a conductores, creación de viajes recurrentes, y coordinación de viajes privados para grupos específicos de pasajeros. Además de estas, hay otras que son inherentes si se plantean para una aplicación en producción, como lo es el añadir métodos de pago, junto con toda la lógica de una billetera que permita solventar la operación de la aplicación.

Liion es una solución desarrollada con el propósito de ser una alternativa más dentro del transporte interurbano, combatiendo los vicios del negocio del transporte interurbano, fomentando la competencia, y bajo ninguna circunstancia, cayendo en la *competencia desleal*. Lo ideal es que los usuarios de transporte no solo usen esta aplicación, sino todas las alternativas y de acuerdo a eso saquen sus propias conclusiones. Una vez la aplicación haya salido a producción se espera un aumento considerable de usuarios con los años 3.4, con lo cual finalmente cumpliría el propósito del tema de

memoria: "*Facilitar y modernizar el transporte interurbano de personas*", personificando la carrera de Telemática, aportando a la sociedad a través de la integración de las tecnologías de información y la comunicación, ayudando (en el proceso) al cuidado del medio ambiente.

"La Telemática consiste en utilizar herramientas de software para agregar inteligencia a los tradicionales sistemas de telecomunicaciones o, bien, en ofrecer servicios informáticos a distancia usando sistemas de telecomunicaciones como, por ejemplo, la internet."

6.1. Trabajos futuros

Liion está sujeta a constantes actualizaciones y cambios surgidos del constante desarrollo, entre las cosas que quedan por implementar y/o diseñar se enumeran:

6.1.0.1. Implementaciones futuras

- Viajes grupales: Diseñar e implementar la manera de poder organizar un viaje con un grupo de amigos/familiares, de forma que todos se inscriban en un viaje en específico.
- Publicar solicitud de viajes: Diseñar e implementar la opción de solicitar viajes (como pasajero) y que los conductores reaccionen a estos con ofertas de viajes.
- Viajes recurrentes: Es sabido que muchos de los conductores que ocuparan la aplicación lo harán de forma periódica, y una forma de motivarlos a ocupar Liion, es haciendo que la acción de publicación de viaje, la hagan solo una vez para todos los días que sean necesarios. Por ejemplo, todos los lunes, pudiendo desactivarlo en casos fortuitos.
- Migración a micro-servicios: Como se espera que Liion de servicios a miles de usuarios, utilizar una arquitectura tal cual como fue presentada en esta memoria, puede no ser la mejor solución desde el punto de vista de la escalabilidad, por esto, que se hace necesario la migración, ya que este tipo de arquitectura facilita la ampliación de la plataforma mediante orquestadores como Kubernetes. Además da facilidad de agregar nuevas características, ya que no es necesario modificar el *core*, si no, solo agregar un micro-servicio más.

Anexos

Lista detallada de cada *endpoints* de la API de servicios

6.1.0.2. Métodos asociados a la identificación de usuarios

Endpoint /getStatusRun

Método: GET

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: Recibe objeto con los siguientes atributos: {*run*}
- **Body:** No aplica

Descripción: Método encargado de buscar si existe o no un usuario en la base de datos con ese RUN inscrito.

Respuesta:

- **Status 200**: Retorna objeto con atributo 'check' de la siguiente manera, {*check:true*}, si es que se encuentra ese RUN ya registrado, o {*check:false*} si no se encuentra.
- **Status 500**: Retorna un objeto con atributos {*sucess:false, res:error*} en caso de que haya algún error interno.

Endpoint /getUserData

Método: GET

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: Recibe objeto con los siguientes atributos: {*uid*}
- **Body**: No aplica

Descripción: Método encargado de buscar los datos del usuario en la base de datos, dado su identificador de *Firebase (UID)*

Respuesta:

- **Status 200**: Retorna todos los datos del usuario en caso de encontrarlos, estos se definen en la sección 4.21 en formato JSON.
- **Status 404**: Retorna un objeto por defecto {*data:'User not found'*} en caso de que no haya encontrado al usuario.
- **Status 403**: Retorna un objeto por defecto {*data:'Token UID Inválido'*}, en caso de que el *UID* sea inválido.

Endpoint /register

Método: POST

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*name, lastname, run, email, birth, password, gender, isDriver, photo64*}

Descripción: Método encargado crear un usuario en la base de datos.

Respuesta:

- **Status 200**: Retorna un objeto {*message:'Successful Registration'*} en caso de un registro exitoso.

- **Status 400**: Retorna un objeto {*message:'Failed registration'*} en caso de que *Firebase* no pueda guardar el dato o falte algún campo.
- **Status 500**: Retorna el error atrapado en formato *JSON*, en caso de que haya habido cualquier otro error interno.

6.1.0.3. Métodos de Registro de conductores

Endpoint /updateUsersDriverStatus

Método: POST

Usuario invocador: Pasajero

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*uid, flagDriver, driverData*}

Descripción: Método encargado de registrar un conductor. Por defecto, todo usuario es solamente pasajero hasta registrarse como conductor, habilitándole las funciones de este.

Respuesta:

- **Status 200**: Retorna objeto por defecto {*data:'Actualización de Driver Status exitoso'*} en caso de un registro exitoso.
- **Status 403**: Retorna un objeto por defecto {*data:'Token UID Inválido'*} en caso de que falte algún atributo o en caso de recibir un *UID* inválido.

6.1.0.4. Métodos para crear viajes

Endpoint /createTravel

Método: POST

Usuario invocador: Conductor

Parámetros:

- **Query**: No aplica
- **Body**: Recibe el objeto *usefullTravelData* que contiene toda la información almacenada en la tabla4.23.

Descripción: Método encargado de crear un viaje bajo el formato de datos antes mencionados.

Respuesta:

- **Status 200**: Retorna un objeto por defecto {*data:'Viaje Creado exitosamente'*} en caso de que se haya creado un viaje exitosamente.
- **Status 403**: Se retorna en los siguientes casos:
  - Retorna un objeto por defecto {*data:'Ya tienes un viaje en ese rango de tiempo'*} en caso de que el usuario ya tenga viaje en ese intervalo de tiempo.
  - Se retorna objeto por defecto {*data:'Error'*} en caso de haber un error interno, o en *Firebase*.

6.1.0.5. Métodos de búsqueda y solicitud de viajes

Endpoint /getTravels

Método: GET

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: Recibe objeto con los siguientes atributos: {*genderApplicant, date, localityDestination, localityOrigin, time, uid*}
- **Body**: No aplica

Descripción: Método encargado de buscar los viajes disponibles dadas las ubicaciones de salida y termino, fecha, horario y preferencias de género.

Respuesta:

- **Status 200**: Retorna un arreglo con todos los viajes que cumplan con los filtros señalados. El formato de la respuesta es: *[*{*Viaje*}*]*. Viaje se define en la sección 4.23 en la base de datos, con la diferencia que añaden campos para identificar al conductor como:
  {*nameDriver, driverPhoto, nRating, sRating*}.
- **Status 403**: Retorna objeto por defecto {*data:'Error'*} en el caso que haya algún error interno o en *Firebase*.

Endpoint /updateSeenTravel

Método: PATCH

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*travelId* }

Descripción: Método encargado de modificar el contador de visitas del viaje con el fin de tener el dato de cuentas veces se ha visto el viaje.

Respuesta:

- **Status 200**: Retorna un objeto {*sucess: true*} en caso de que se haya modificado el contador de vistas exitosamente.
- **Status 500**: Retorna objeto por defecto {*data:'Error'*} en caso de haber un error interno o en *Firebase*.

Endpoint /registerPassengerRequest

Método: POST

Usuario invocador: Pasajero

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*travelId, passengerUID, extraBaggage, pickUp, dropOff, payMode* }

Descripción: Método encargado de crear las solicitudes de viaje a los viajes creados. Estas solicitudes son aceptadas automáticamente en el viaje si se cumplen las condiciones para entrar en este, vale decir, el conductor no acepta o rechaza las solicitudes.

Respuesta:

- **Status 200**: Retorna un objeto {*sucess: true*} en caso de que la solicitud de viaje se haya creado exitosamente y consecuentemente se haya actualizado el viaje en cuestión.
- **Status 403**: Se retorna en los siguientes casos:
  - Se retorna el objeto {*sucess: false, error: 'Equipaje extra no disponible'*} si el usuario requiere equipaje extra (ya sea maleta grande o de mano) y no haya disponible en el viaje.

- Se retorna el objeto { *sucess: false, error: 'Ya tienes una solicitud en este viaje'* } si el usuario ya ha hecho solicitud a este viaje.
- Se retorna el objeto {*sucess: false, error: 'Se acabaron los asientos disponibles'* } si no quedan asientos disponibles en el vehículo.
- Se retorna objeto { *sucess: false, error: 'Ya tiene un viaje en este horario'* } si el usuario ya tiene un viaje en un intervalo de tiempo que se traslape con este nuevo viaje.

- **Status 500**: Retorna objeto por defecto {*data:'Error'*} en caso de haber un error interno o en *Firebase*.

6.1.0.6. Métodos relacionados a los viajes personales

Endpoint /getTravelsPassenger

Método: GET

Usuario invocador: Pasajero

Parámetros:

- **Query**: Recibe objeto con los siguientes atributos: {*passengerUID* }
- **Body**: No aplica

Descripción: Método encargado de buscar todos los viajes vigentes que el usuario tiene como pasajero (aceptados), hasta 6 horas hacia atrás. En estos viajes se consideran también aquellos en curso.

Respuesta:

- **Status 200**: Retorna un arreglo con todos los viajes que cumplan lo antes mencionado, con los siguientes datos { *id, requestId, costPerSeat, extraBaggage, approvalIns, smoking, genderPreference, nSeatsAvailable, date, startTime, destinationDetails, originDetails, durationMinutes, nameDriver, driverPhoto, carModel, carPhoto, plate, nRating, sRating, status, statusRequest* }
- **Status 500**: Retorna objeto por defecto {*data:'Error'*} en caso de haber un error interno o en *Firebase*.

Endpoint /getTravelsDriver

Método: GET

Usuario invocador: Conductor

Parámetros:

- **Query**: Recibe objeto con los siguientes atributos: {*driverUID* }
- **Body**: No aplica

Descripción: Método encargado de buscar todos los viajes vigentes que el usuario tiene como conductor, hasta 6 horas hacia atrás. En estos viajes se consideran también aquellos en curso.

Respuesta:

- **Status 200**: Retorna un arreglo con todos los viajes que cumplan lo antes mencionado, con los siguientes datos { *id, date, startTime, destinationDetails, originDetails, durationMinutes, status, nSeatsAvailable, nSeatsOffered, costPerSeat, extraBaggage, approvalIns, smoking, genderPreference* }
- **Status 500**: Retorna objeto por defecto {*data:'Error'*} en caso de haber un error interno o en *Firebase.*

Endpoint /deletePassengerRequest

Método: DELETE

Usuario invocador: Pasajero

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*travelId, requestId* }

Descripción: Método encargado de crear las solicitudes de viaje a los viajes creados. Estas solicitudes son aceptadas automáticamente en el viaje si se cumplen las condiciones para entrar en este, vale decir, el conductor no acepta o rechaza las solicitudes.

Respuesta:

- **Status 200**: Retorna un objeto {*sucess: true, res: 'Reserva cancelada'*} en caso de que la solicitud haya sido cancelada exitosamente. Se actualizan el *request* de viaje a 'unsuscribe' y el viaje, particularmente el número de asientos disponibles, equipaje extra y el estado en caso estar cerrado.

- **Status 403**: Retorna objeto {*sucess: false, res: 'Su reserva ya no se encuentra aceptada'*} si es que al momento de hacer el *request*, la solicitud no ha sido aceptada.
- **Status 500**: Retorna objeto por defecto { *sucess: false, res: 'Error'*} en caso de haber un error interno o en *Firebase*.

Endpoint /deleteDriverTravel

Método: DELETE

Usuario invocador: Conductor

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*travelId*}

Descripción: Método encargado cancelar un viaje ya creado, siempre y cuando no esté en curso, ni tampoco abortado. Además de modificar el viaje en cuestión, también actualiza el estado de todos los *request* al viaje para mantener la consistencia en los datos.

Respuesta:

- **Status 200**: Retorna un objeto {*sucess: true, res: 'Viaje cancelado'*} en caso de que el viaje haya sido cancelado exitosamente y también la actualización de todos los *request* que dependen de dicho viaje.
- **Status 403**: Retorna objeto {*sucess: false, res: 'Su viaje ya es imposible cancelarlo'*} si al momento de intentar cancelar el viaje, este está en curso o ya paso (finalizado o abortado).
- **Status 500**: Retorna objeto por defecto { *sucess: false, res: 'Error'*} en caso de haber un error interno o en *Firebase*.

6.1.0.7. Métodos para notificaciones en tiempo real

Endpoint /notifToPassengers

Método: POST

Usuario invocador: Conductor (automático)

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*travelId* }

Descripción: Método encargado mandar notificaciones *Push* a los pasajeros cuando un conductor pone en marcha un viaje.

Respuesta:

- **Status 200**: Retorna un objeto {*sucess: true, res: resfcm*} en caso de que la notificación haya sido enviada exitosamente al servidor *fcm* con la lista de los usuarios inscritos en el viaje.
- **Status 400**: Se retorna en los siguientes casos:
  - Se retorna el objeto {*sucess: false, res: 'Envie un id de viaje'*} en caso de que no haya sido enviado el *ID* de viaje requerido.
  - Se retorna el objeto {*sucess: false, res: 'No se encuentra viaje'*} en caso de que el viaje no exista.
  - Se retorna el objeto {*sucess: false, res: 'No se encuentran id de viajeros'*} en caso de que no haya ningún usuario en el viaje al momento del request.
  - Se retorna el objeto *sucess: false, res: 'Ops ha ocurrido un error'* } En caso de que haya ocurrido un error en alguna de las peticiones internas.
- **Status 500**: Retorna objeto por defecto { *sucess: false, res: e*} en caso de que *Firebase* no haya podido enviar los mensajes. El objeto 'e' contiene información del posible error específico, y este dependerá de *FCM*.

Endpoint /updateTokenFcm

Método: PATCH

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*uid, fcmToken* }

Descripción: Método encargado de actualizar el token *FCM* cada vez que este cambie en el dispositivo móvil.

Respuesta:

- **Status 200**: Retorna un objeto {*data: 'Actualización de token FCM exitoso'* } en caso de que se haya actualizado el token *FCN* exitosamente.
- **Status 403**: Se retorna en los siguientes casos:
  - Retorna un objeto por defecto {*data:'Token UID Inválido'*} en caso de que el *UID* sea inválido.
  - Retorna un objeto por defecto {*data:'Token UID Inválido o error'*} en caso de que *UID* no venga en el *request* junto con el token *FCM*.

6.1.0.8. Métodos de un viaje en curso

Endpoint /getTravelItinerary

Método: GET

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: Recibe objeto con los siguientes atributos: {*c* }
- **Body**: No aplica

Descripción: Método encargado de retornar el itinerario de viaje, es decir, el próximo punto de subida o bajada, a quien corresponde y los demás puntos restantes.

Respuesta:

- **Status 200**: Se retorna en los siguientes casos:
  - Retorna un objeto con todos los datos del *itinerario* 4.26 mas próximo, en conjunto a los datos del usuario que suba o baje en este y una lista de coordenadas junto con el tipo y status de los demás puntos de subida o bajada. Objeto a retornar: {*coordinate, extraBaggage, passengerUID, status, step, type, photo, fullname, markerList* }
  - Retorna un objeto { *status: 'finished'*} en caso de que no haya ningún punto del itinerario pendiente.
- Status 403: Se retorna objeto { *sucess: false* } en caso de que el viaje aún no parta o haya sido abortado.
- **Status 400**: Retorna objeto {*sucess: false*} en caso de no recibir el *travelid* en el request o algún otro error interno en las *queries* a *Firebase*.

Endpoint /getPassengerTravelItinerary

Método: GET

Usuario invocador: Pasajero

Parámetros:

- **Query**: Recibe objeto con los siguientes atributos: {*travelId, passengerUID* }
- **Body**: No aplica

Descripción: Método encargado de retornar el itinerario de viaje para un usuario en particular. Obteniendo los puntos que este tenga de bajada o subida, que aún no sucedan.

Respuesta:

- **Status 200**: Se retorna en los siguientes casos:
  - Se retorna objeto con todos los datos del punto más cercano del itinerario 4.26 en conjunto con el estatus, coordenadas y el tipo para los siguientes puntos (Siempre menor o igual a dos por cada usuario).
  - Se retorna objeto { *status: 'finished'* } en caso de que no queden puntos pendientes en el itinerario.
- **Status 400**: Retorna objeto {*sucess: false*} en caso de no recibir el *travelid* en el *request* o algún otro error interno en las *queries* a *Firebase*.

Endpoint /updateTravelItinerary

Método: PUT

Usuario invocador: Conductor

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*travelId, step, type, data*}

Descripción: Método encargado de actualizar el itinerario de acuerdo al orden de subida y bajada de pasajeros. En la subida se entrega el *ID* de usuario por parte del pasajero en un *QR*. **Respuesta**:

- **Status 200**: Se retorna en los siguientes casos:
  - Se retorna un objeto { *status:true* } en caso de que la actualización del itinerario haya sido exitosas, lo que implica que en la subida al viaje, la persona sea la correcta en cada punto y que el paso no haya terminado. Si es el último punto, el viaje se marca como finalizado.
  - Se retorna un objeto { *status:false* } en caso de que quien genere el código *QR* tenga un id de viaje o de id usuario equivocado.
- **Status 403**: Se retorna el objeto { *sucess: false* } en el caso que se proporcione información errónea respecto al tipo de 'pickup' en el itinerario.
- **Status 400**: Se retorna el objeto { *sucess: false* } en caso de que no se hayan proporcionado los datos pedidos en el *body*, que haya habido un error en *queries* de *Firebase*, o bien algún error interno.

Endpoint /updateUserLocationInTravel

Método: PATCH

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*travelId, location, uid*}

Descripción: Método encargado de actualizar las coordenadas del usuario durante un viaje en curso.

Respuesta:

- **Status 200**: Retorna el objeto {*sucess: true* } en caso de que se hayan actualizado correctamente las coordenadas del usuario en el viaje.

- **Status 400**: Se retorna el objeto { *sucess: false, res: 'Error'*} en el caso que no se hayan proporcionado los campos requeridos, haya ocurrido una falla en las *queries* a *Firebase*, o bien un error interno.

6.1.0.9. Métodos para puntuar usuarios

Endpoint /getTravelPartners

Método: GET

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: Recibe objeto con los siguientes atributos: {*travelId, userUid* }
- **Body**: No aplica

Descripción: Método encargado de retornar los demás usuarios del viaje, si es un pasajero en el viaje, entonces retorna el conductor en primer lugar, y los demás pasajeros; sin embargo, si es el conductor, retorna una lista con todos los pasajeros. En cualquier caso se retorna si el usuario es conductor o pasajero.

Respuesta:

- **Status 200**: Retorna un objeto con una lista de usuarios. { *[user]*}, donde el 'user' contiene los siguientes campos: { *uid, nameUser, type, photo* }
- **Status 400**: Se retorna el objeto { *sucess: false, res: 'Error'*} en el caso que no se hayan proporcionado los campos requeridos, haya ocurrido una falla en las *queries* a *firebase*, o bien un error interno.

Endpoint /updateUserRanting

Método: PATCH

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*userList, travelId*}.

Descripción: Método encargado de actualizar la puntuación que un usuario da a los demás, ya sean al conductor o a los pasajeros. Cada usuario tiene una puntua-

ción como conductor y otra como pasajero, y en donde objeto userList contiene la puntuación e id que el usuario ha dado a los demás.

Respuesta:

- **Status 200**: Retorna objeto {*sucess: true* } en caso de que se hayan actualizado correctamente las puntuaciones.
- **Status 400**: Se retorna el objeto { *sucess: false, res: 'Error'*} en el caso que no se hayan proporcionado los campos requeridos, haya ocurrido una falla en las *queries* a *Firebase*, o bien un error interno.

6.1.0.10. Métodos transversales a todos los procesos

Endpoint /getDetailsOfTravel

Método: GET

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: Recibe objeto con los siguientes atributos: {*travelId* }
- **Body**: No aplica

Descripción: Método encargado de retornar los datos específicos relevantes para la vista que presenta el viaje y permite inscribirse en él.

Respuesta:

- **Status 200**: Retorna un objeto con los siguientes atributos: { *seen, routeCoordinates, nSeatsOffered, usb, airConditioning, carSeats, carColor, typeVehicule, carPhoto* }
- **Status 500**: Se retorna un objeto por defecto {*data:'Error'*}, si es que falta el parámetro requerido o bien ocurren algún de *Firebase* o interno.

Endpoint /getRouteCoordinates

Método: GET

Usuario invocador: Pasajero y Conductor

Parámetros:

- **Query**: Recibe objeto con los siguientes atributos: {*travelId* }
- **Body**: No aplica

Descripción: Método encargado de retornar las coordenadas del viaje especificado.

Respuesta:

- **Status 200**: Retorna un objeto con un arreglo de coordenadas.
- **Status 400**: Se retorna un objeto {*sucess: false*}, si es que falta el parámetro requerido, o bien ocurren algún de *Firebase*, o interno.

Endpoint /getUpcomingTravels

Método: GET

Usuario invocador: Conductor

Parámetros:

- **Query**: Recibe objeto con los siguientes atributos: {*travelId* }
- **Body**: No aplica

Descripción: Método encargado de retornar los viajes que tiene un conductor abierto o cerrados, en un intervalo de 20 minutos hacia atrás o adelante desde el momento que se ejecuta la *query*. Si algún viaje está por partir, y está aún abierto, en este punto se cierra automáticamente.

Respuesta:

- **Status 200**: Ocurre en las siguientes situaciones:
  - Retorna objeto { *sucess: true, res* }, donde en res contiene el viaje más cercano disponible.
  - Retorna un objeto {*sucess: true, res: [*{ *status: 'nada'* }*]* } en caso de obtener datos sin sentido desde firebase.
  - Retorna objeto { *sucess: false, res: []*} en caso de no encontrar viajes en ese intervalo de tiempo.
- **Status 500**: Retorna objeto { *sucess: false, res: 'Ops hubo un error'*} en que falte el parámetro requerido, o bien ocurren algún de *Firebase*, o interno.

Endpoint /updateStateTravel

Método: PATCH

Usuario invocador: Conductor

Parámetros:

- **Query**: No aplica
- **Body**: Recibe objeto con los siguientes atributos: {*travelId, state*}.

Descripción: Método encargado de iniciar o finalizar un viaje en curso, generando en el primer caso el itinerario de viaje.

Respuesta:

- **Status 200**:Se retorna en los siguientes casos:
  - Retorna objeto {*sucess: true, res: 'Viaje iniciado'* } en caso de que se haya creado el itinerario exitosamente y se haya modificado el estado a viaje en curso.
  - Retorna objeto { *sucess: true, res: 'Viaje finalizado'* } en caso de que se haya finalizado el viaje exitosamente.
- **Status 403**: Se retorna en los siguientes casos:
  - Se retorna el objeto { *sucess: false, res: 'Imposible iniciar aun el viaje'* }, si es que al momento del *request* no se está en un rango de 20 minutos hacia atrás o adelante.
  - Se retorna el objeto { *sucess: false, res: 'Imposible iniciar sin pasajeros'* }. En caso de que no hayan pasajeros aceptados en el viaje.
  - Retorna objeto { *sucess: false, res: 'Imposible finalizar sin haberlo iniciado'* } en caso de que se intente finalizar un viaje que no esté en curso.
  - Retorna objeto { *sucess: false, res: 'Estado no válido'* } en caso de que el estado ingresado no sea ni *en curso* ni *terminado*
- **Status 500**: Se retorna objeto { *sucess: false, res: 'Error'*} en que falte algún parámetro requerido o bien ocurren algún de *Firebase*, o interno.

Encuesta sobre autos compartidos

¿Sabes lo que es el Carpool? El carpool es la práctica que consiste en compartir un automóvil con otras personas tanto para viajes periódicos como para trayectos puntuales. Esto se puede hacer como conductor publicando tu viaje y compartiendo tu vehículo, o como pasajero uniéndote a un viaje que coincida con el tuyo.

La finalidad de esta encuesta es conocer sus preferencias al momento de realizar viajes.

Toda la información suministrada por usted es confidencial y será utilizada solamente con fines estadísticos, para una investigación realizada por estudiantes de la Universidad Técnica Federico Santa María. Recuerde que no existen respuestas correctas o incorrectas, considere responder esta encuesta en una situación normal sin pandemia.

Muchas gracias.

1. ¿Has realizado viajes "interurbanos" o de "larga distancia"? Ej: Valparaíso – Santiago, Santiago - Rancagua, Valparaíso - Concepción.

   a) Sí.

   b) No.

   Ir a la pregunta 2.

2. ¿Cada cuánto realizas viajes "interurbanos" o "de larga distancia"?

   a) Una vez al mes (ir a la pregunta 3).

   b) 3 veces al mes (ir a la pregunta 3).

   c) Todas las semanas (ir a la pregunta 3).

   d) Todos los días del mes (ir a la pregunta 3).

   e) Solo los fines de semana (ir a la pregunta 3).

   f) Nunca (ir a la pregunta 21).

g) Otros (indicar) (ir a la pregunta 3).

3. ¿Cuáles son los motivos de los viajes? (Selección múltiple).

   a) Trabajo.

   b) Estudio.

   c) Familia y amigos.

   d) Otros (indicar).

   Ir a la pregunta 4.

4. ¿A través de qué medios viajas? (Selección múltiple).

   a) Bus interurbano.

   b) Automóvil propio.

   c) Avión.

   d) Tren interurbano.

   e) Otros (indicar).

   Ir a la pregunta 5.

5. ¿Has utilizado alguna forma de hacer viajes compartidos?

   a) Si (ir a la pregunta 6).

   b) No (ir a la pregunta 7).

6. ¿Por qué medios has coordinado el viaje compartido? (Selección múltiple).

   a) Facebook.

   b) Instagram.

   c) Twitter.

   d) Aplicación móvil.

e) Otra.

Ir a la pregunta 7.

7. ¿Posees automóvil propio?

   a) Si (ir a la pregunta 8).

   b) No (ir a la pregunta 15).

8. ¿Utilizas tu automóvil para hacer los viajes de "larga distancia"?

   a) Si (ir a la pregunta 11).

   b) No (ir a la pregunta 9).

9. ¿Por qué razón no utiliza su automóvil para realizar viajes de larga distancia? (Selección múltiple).

   a) Para evitar gastos de bencina, peaje y estacionamiento.

   b) Para evitar congestión vehicular.

   c) Por falta de estacionamiento.

   d) Para no dañar el medio ambiente.

   e) No me gusta conducir.

   f) No me gusta viajar solo.

   g) Otros (indicar).

   Ir a la pregunta 11.

10. ¿Por qué motivos no compartiría su automóvil?

    a) Inseguridad de viajar con desconocidos.

    b) No me interesa.

    c) Incomodidad de viajar con desconocidos.

d) Desconfianza en la puntualidad de los pasajeros.

e) Otros (indicar).

Ir a la pregunta 15.

11. ¿Estarías dispuesto a compartir tu automóvil para hacer un viaje compartido de "larga distancia"?

a) Si (ir a la pregunta 12).

b) No (ir a la pregunta 10).

Liion es una nueva aplicación móvil que permite hacer viajes compartidos, con la posibilidad de organizar un viaje y que gestione los pagos de forma sencilla (similar a aplicaciones como Uber, cabify, etc.) aplicada a viajes de largas distancias

12. Considerando que participarías en nuestra aplicación, desde el punto de vista del conductor, indique el nivel de importancia de las siguientes afirmaciones. Recuerde que el objetivo no es obtener beneficios monetarios, sino compartir gastos del viaje.

	No importante	Poco importante	Indiferente	Importante	Muy importante
Que la aplicación me sugiera un precio máximo por mis asientos, en base a la distancia y a valores de mercado.					
Poder retirar el dinero ganado en un tiempo prudente					
Poder puntuar a los diferentes pasajeros con los que viaje					

Cuadro 6.1: Importancia de características de la aplicación, desde el punto de vista del conductor.

Ir a la pregunta 13.

13. El conductor tendrá la alternativa de poder personalizar el viaje, ajustando diferentes filtros para seleccionar a los pasajeros. ¿De los siguientes, cuáles te parecen pertinentes?

    a) Por genero.

    b) Por edad.

    c) Por puntuación en la aplicación.

    d) Por antigüedad en la aplicación.

    e) Otros (indicar).

    Ir a la pregunta 14.

14. ¿Aceptaría el cobro de una pequeña comisión por cada viaje de larga distancia que realice?

    a) Si.

    b) No.

    Ir a la pregunta 15.

15. ¿Estaría dispuesto a compartir viaje como “pasajero” junto a otras personas en un viaje de larga distancia?

    a) Si (ir a la pregunta 16).

    b) No (ir a la pregunta 18).

16. Considerando que participarías en nuestra aplicación, desde el punto de vista del pasajero indique el nivel de importancia de las siguientes afirmaciones.

	No importante	Poco importante	Indiferente	Importante	Muy importante
Que pueda cancelar mi reserva con un tiempo prudente sin penalización					
Tener un botón de emergencia para poder ser auxiliada o auxiliado de forma rápida (caso de robo, accidente)					
Poder opinar respecto al viaje, y puntuar a mis compañeros de viaje.					
Que te pasen a buscar/dejar cercano a tu punto de salida/llegada					

Cuadro 6.2: Pregunta sobre importancia de características de la aplicación, desde el punto de vista del pasajero.

Ir a la pregunta 17.

17. ¿Qué formas de pago te gustaría que tuviera la aplicación? (Selección múltiple).

    a) Asociada a tarjeta Crédito (cargo automático una vez confirmado el viaje).

    b) Asociado a la tarjeta de débito (pagar una vez confirmado el viaje).

    c) Efectivo (pago una vez se vaya a abordar el auto).

    d) Otros (indicar).

Ir a la pregunta 17.

18. ¿Te interesaría utilizar Liion, la nueva aplicación móvil que permite hacer viajes compartidos, con la posibilidad de organizar un viaje y que gestione los pagos de forma sencilla (similar a aplicaciones como uber, cabify, etc.) aplicada a viajes de largas distancias?

    a) Si.

    b) No.

    Ir a la pregunta 19.

19. Indique el género que más te identifique.

    a) Femenino (ir a la pregunta 20).

    b) Masculino (ir a la pregunta 21).

20. Con el fin de conocer las preferencias del género, exprese su percepción respecto a las siguientes afirmaciones.

	Totalmente en desacuerdo	En desacuerdo	Neutral	De acuerdo	Totalmente de acuerdo
Me siento segura en un viaje de sólo mujeres					
Si en un viaje solo hubiese hombres inscritos, no solicitaría la reserva de asiento o cancelaría la reserva en caso de ya haberla solicitado					
Siempre elegiría viajar con alguien que tenga suficiente antigüedad y puntuación dentro de la aplicación sin importar su género.					
Que el viaje sea organizado por una conductora es un plus a la hora de elegir					

Cuadro 6.3: Pregunta sobre percepción femenina de los viajes compartidos

Gracias por participar de esta encuesta, cuando nos titulemos te agregaremos en los agradecimientos.

21. Indique su rango de edad

    a) Entre 18 y 25 años.

b) Entre 26 y 35 años.

c) Entre 36 y 45 años.

d) Entre 46 y 55 años.

e) 56 o más.

Informe ámbito legal Ley Uber

En el presente informe se expondrán los diferentes alcances legales en los que se encontrara la solución al desafío planteado por el académico Nicolás Jara, donde si llegara a corresponder se nombraran y detallarán las diferentes normativas que pudiesen limitar el funcionamiento de esta, con el fin de servir de insumos a la hora del desarrollo de la aplicación.

Contextualizando, el desafío consiste en poder mejorar el transporte interurbano utilizando las TIC, para esto se planteó la solución donde se desarrollara una aplicación, que en su primera etapa será una plataforma digital para smartphones con sistema operativo Android, la que proveerá a sus usuarios de un sistema de compartición de vehículos, donde una persona que vaya a realizar algún viaje podrá ofrecer los asientos de su auto indicando diferentes parámetros (como precio del asiento, origen y destino, hora, etc.), para que luego los demás usuarios de la aplicación puedan ver dicha oferta de asientos y de esta manera reservar un puesto para poder viajar.

Luego de una búsqueda exhaustiva de información, se logró determinar que el tema de las aplicaciones de transporte digitales en Chile, se encuentra en calidad de no regulado, por lo que el desarrollo de un potencial sistema no contravendría ninguna ley vigente dentro del marco jurídico del estado de Chile. Sin embargo, el viernes 20 de julio del 2018, hizo ingreso a la cámara de diputados un proyecto que tiene con fin el poder regular a las empresas de aplicación de transportes, el proyecto titulado "Regula a las aplicaciones de transporte remunerado de pasajeros y los servicios que a través de ellas se presten" [1], expone diferentes normativas dan un marco legal para

el funcionamiento de las diferentes plataformas que existen. Actualmente, el proyecto se encuentra en la cámara del senado, donde espera seguir en tramitación, su última modificación puede ser vista en la referencia [2], de donde se rescata el siguiente extracto, que crea la denominación EAT, que corresponderá a las empresas, las cuales serán reguladas:

Artículo 1.- Se denominará Empresa de Aplicación de Transportes, en adelante "EAT", a toda persona jurídica que preste o ponga a disposición de las personas un servicio de plataforma digital, sistema informático o tecnología de cualquier tipo, que permita a un pasajero contactarse con el propietario, administrador o conductor de un vehículo de transporte menor de pasajeros, para ser transportado desde un origen a un destino determinado, pagando una tarifa por el servicio recibido. Estas serán consideradas para todos los efectos como empresas de transporte remunerado de pasajeros y, asimismo, sus servicios serán calificados como servicios de transporte remunerado de pasajeros. Como se puede notar, la solución planteada al desafío, cae en la denominación de lo que vendría siendo una EAT, por lo que a continuación se analizarán los diferentes artículos [2] relevantes para el desarrollo de la solución:

- Los Conductores habilitados y los vehículos adscritos por cada región. Los primeros estarán habilitados solamente para tomar pasajeros e iniciar rutas de transporte remunerado de pasajeros en la región cuya inscripción corresponda, salvo en el caso que la siguiente ruta tenga por destino la región donde se encuentran inscritos el conductor habilitado y el respectivo vehículo (Artículo 2.d)

- Únicamente podrán registrarse vehículos cuyos propietarios o meros tenedores sean personas naturales. No podrán registrarse más de dos vehículos totales en el registro por cada propietario, los que podrán operar en distintas empresas de aplicación de transportes. (Artículo 3)

- Mantener de manera permanente a disposición de los usuarios medios de comunicación para consultas, reclamos o denuncias (Artículo 4.c).

- Otorgar información al usuario sobre las características de la aplicación, el recorrido propuesto de acuerdo al requerimiento efectuado y el tiempo y costo estimado del traslado, de manera de permitirle comparar opciones y adoptar decisiones de contratación de estos servicios de manera informada (Artículo 5.a).

- Otorgar información al conductor sobre el recorrido propuesto, destino, tiempo de viaje, nombre y calificación del usuario, saber si el viaje es en dinero efectivo o mediante tarjeta bancaria, para permitir al conductor adoptar la decisión de realizar el servicio (Artículo 5.b).

- Operar solamente con conductores inscritos en el Registro (Artículo 5.c).

- Informar al usuario la tarifa en forma previa al inicio del viaje, la que no podrá variar una vez informada al pasajero, a menos que este decida cambiar la ruta o trazado, o por interrupciones viales, de accidente o imprevistos de tráfico se vea alterada la ruta y tiempo original, adecuando a este efecto el cobro en tiempo y distancia. En el caso de que el recorrido incluya pago de peajes, estos deberán estar incluidos en la tarifa informada y no podrán cobrarse separadamente. (Artículo 5.d)

- Disponer de una tarifa sustentable para los conductores de los vehículos inscritos, que permita solventar los costos operativos del servicio y generar un margen de utilidades razonable (Artículo 5.e).

- Informar al pasajero la marca, modelo y año del vehículo y su placa patente, y la identificación del conductor, con su nombre y la calificación efectuada por otros usuarios (Artículo 5.f).

- Operar solamente con vehículos que cumplan con los requisitos legales y reglamentarios aplicables (Artículo 5.g)

- Mantener un acceso dentro de la misma plataforma o en sus sitios web asociados, informando detalladamente los términos y condiciones en que se presta el

servicio, tanto para el conductor, los propietarios de vehículos y los usuarios. Los términos y condiciones de las EAT para la inscripción de conductores, deberán establecer expresamente las causales de eliminación, suspensión o bloqueo a los conductores, y solo podrán aplicar dichas causales, informando debidamente al afectado, y considerando la posibilidad de apelar a esta medida. Asimismo, los referidos términos y condiciones deberán contener las posibles promociones que las empresas de aplicación de transporte pueden ofrecer a sus usuarios y conductores, y los mecanismos de funcionamiento de las mismas, estableciendo con cargo a quién aplican (Artículo 5.h).

- Informar de manera permanente a usuarios y conductores los sistemas generales de evaluación del servicio y sus efectos, como asimismo, la evaluación o ranking específico del conductor que ofrece sus servicios en un viaje determinado (Artículo 5.i).

- En ningún caso, estos vehículos podrán recoger pasajeros en la vía pública si estos no han concertado una reserva previa mediante la plataforma tecnológica (Artículo 5.g).

- Se establece que los conductores deben contar con una licencia profesional clase A-1 (artículo 6).

- La EAT deberá solicitar semestralmente al conductor el certificado de antecedentes, para comprobar que no tienen anotaciones relativas a los delitos previstos, ley 20.000, 18.290 (artículo 6).

- Sin perjuicio de lo anterior, para registrarse en una EAT, los vehículos deberán cumplir, como principal exigencia, tener una antigüedad, no superior a tres años, al solicitar su inscripción por primera vez en el Registro referido en el artículo 2, y no superior a 5 años en caso de reemplazo. La antigüedad se calculará como la diferencia entre el año en que se solicita la inscripción y el año de fabricación o

modelo del vehículo anotado en el Registro Nacional de Vehículos Motorizados (artículo 7).

- **Prohíbase a las empresas de aplicación de transportes realizar servicios de carácter compartido, esto es, aquellos en que existe una ruta o trazado establecido y dentro de un mismo viaje se recoge a distintos pasajeros sin relación entre sí, los que solo podrán prestarse mediante taxis inscritos en el Registro Nacional de Servicios de Transporte de Pasajeros, en la modalidad de taxi colectivo (artículo 9).**

- Artículo tercero.- Durante los primeros dieciocho meses de vigencia de la presente ley, no será exigible a los conductores adscritos a las EAT el contar con la licencia profesional a que se refiere el inciso primero del artículo 6 de esta ley. Transcurrido tal plazo, aquellos deberán poseer dicho instrumento para operar válidamente (DISPOSICIÓN TRANSITORIA).

- Artículo cuarto.- Durante los primeros treinta y seis meses de vigencia de la presente ley, no será exigible la antigüedad máxima de los vehículos de tres años a que se refiere el inciso final del artículo 7, y se aceptará la inscripción en las EAT de vehículos de hasta seis años de antigüedad, los que a la fecha de término del referido plazo deberán ser eliminados o reemplazados por vehículos que cumplan el requisito del referido artículo 7 (DISPOSICIÓN TRANSITORIA).

Como conclusión, se estima es que es pertinente tomar una postura de desarrollo frente a los antecedentes anteriormente mostrados, donde se debe elegir si tomar en consideración la futura ley (la cual ya se encuentra en su segundo trámite legislativo), lo cual sería lógico pensando en que este proyecto pueda escalar, lo que, sin embargo, llevaría a la solución propuesta a ser inviable por contravenir con lo explícito por la futura ley (artículo 9), esto último se debe analizar con detenimiento. El no considerar la ley también es válido en algunos casos, como en el ámbito académico, porque servirá como proyecto de investigación para otras aplicaciones. Si se decide no tomar

en consideración con lo expuesto por la futura ley, de todas maneras se podría operar mientras no entre en vigencia dicha ley, o en otros países con legislaciones distintas. Referencias [55] [56].

Bibliografía

[1] Aplicación móvil carretera. https://play.google.com/store/apps/details?id=cl.carretera.carretera&hl=es_CL&gl=US.

[2] Ministerio de Desarrollo Socia. Sintesis de resultados encuesta casen vivienda y entorno. http://observatorio.ministeriodesarrollosocial.gob.cl/storage/docs/casen/2015/CASEN_2015_Resultados_vivienda_y_entorno.pdf, Noviembre 2016.

[3] Ministerio de transportes y telecomunicaciones. Decreto 80, reglamenta el transporte privado remunerado de pasajeros, modifica el decreto nº 212, de 1992, reglamento de los servicios nacionales de transporte publico de pasajeros y deja sin efecto decreto que indica. https://www.bcn.cl/leychile/navegar?idNorma=230180&idVersion=2020-05-19&idParte=, Mayo 2020.

[4] INE. Parque de vehículos en circulación. https://regiones.ine.cl/documentos/default-source/region-xiv/estadisticas-r14/boletines-informativos/parque-de-veh%C3%ADculos/parque-de-veh%C3%ADculos-en-circulaci%C3%B3n---periodo-2018.pdf, Octubre 2019.

[5] OPS. Calidad del aire ambiente. https://www.paho.org/es/temas/calidad-aire-salud/calidad-aire-ambiente, 2016.

[6] Normas 5 y 6. https://data.consilium.europa.eu/doc/document/ST-3602-2007-REV-2/es/pdf.

[7] Universidades chilenas desarrollan sistema de mediciones contaminantes del transporte motorizado: Aplicación de la norma euro 6 en chile. https://vtte.utem.cl/2019/11/19/universidades-chilenas-desarrollan-sistema-de-mediciones-contaminantes Noviembre 2019.

[8] Ministerio del Medio Ambiente. Emisiones de co_2 generadas por transporte en ruta según categoría vehicular, 2005-2017. https://retc.mma.gob.cl/transporte-en-ruta/, 2017.

[9] Andrea Herrera y Slaven Razmilic (CEP). De la casa a trabajo: Análisis de un tiempo perdido. https://www.cepchile.cl/cep/site/artic/20180405/asocfile/20180405120239/dpp_029_abril2018_srazmilic.pdf, Abril 2018.

[10] INE. Informe de resultados enusc 2018. https://www.ine.cl/docs/default-source/seguridad-ciudadana/publicaciones-y-anuarios/2018/informe-de-resultados---xv-enusc-2018.pdf?sfvrsn=3b356305_2, Mayo 2019.

[11] Borken-Kleefeld. Study identifies travel choices for a smaller carbon footprint. https://iiasa.ac.at/web/home/about/news/EST--Carbon-Footprint.en.html, 2013.

[12] They do it.. so can we carry more to win the war. https://oac.cdlib.org/ark:/28722/bk0007s8h9v/?brand=oac4.

[13] Historia del carpool. https://www.tripspark.com/blog/the-history-of-carpooling#:~:text=Ridesharing%20or%

20carpooling%20began%20as,Exchange%20and%20Self%2DDispatching%20System, Septiembre 2019.

[14] Minimizing co2 emissions in a practical daily carpooling problem. https://www.sciencedirect.com/science/article/pii/S030505481630301X, Mayo 2017.

[15] Carpooling: A step to reduce congestion. https://escholarship.org/uc/item/7jx6z631, Octubre 2018.

[16] Kum Kum Dewan and Israr Ahmad. Carpooling: A step to reduce congestion. https://www.researchgate.net/publication/26623758_Carpooling_A_Step_to_Reduce_Congestion_A_Case_Study_of_Delhi, Febrero 2007.

[17] En todas las ciudades el auto es más rápido que el transporte público. https://www.infraestructurapublica.cl/en-todas-las-ciudades-el-auto-es-mas-rapido-que-el-transporte-pub Enero 2018.

[18] Bruno Córdova. Chile, un país que le tiene miedo al otro. https://www.cnnchile.com/lodijeronencnn/chile-un-pais-que-le-tiene-miedo-al-otro_20181031/, Octubre 2018.

[19] Blablacar. https://www.blablacar.com/.

[20] Uber pool. https://www.uber.com/us/en/ride/uberpool/.

[21] Lyft. https://www.lyft.com/.

[22] Liftshare. https://liftshare.com/uk.

[23] Waze carpool. https://www.waze.com/carpool.

[24] Zimride. https://zimride.com/.

[25] Carmago. https://www.gocarma.com/.

[26] Allride. https://www.allrideapp.com/.

[27] Karpool. https://karpool.cl/.

[28] Nos fuimos. http://www.nosfuimos.cl/.

[29] Medidas covid por blablacar. https://support.blablacar.com/hc/en-gb/articles/360015124400-Coronavirus-COVID-19-Information.

[30] Medidas covid por liftshare. https://business.liftshare.com/commuter-guidelines/.

[31] Blablacar 2020. https://blog.blablacar.com/newsroom/news-list/blablacar-withstands-the-crisis-with-50-million-passengers-

[32] Medidas covid por karpool. https://karpool.cl/index.php/2020/06/15/como-karpool-ha-innovado-en-medio-de-la-crisis-sanitaria/.

[33] Cambios en movilidad de chile en 2020. https://www.gstatic.com/covid19/mobility/2021-04-14_CL_Mobility_Report_es-419.pdf.

[34] Sitio de comparación entre flutter y react native. https://levelup.gitconnected.com/flutter-vs-react-native-comparing-the-features-of-each-framework-f61b

[35] Sitio del projecto mono. https://www.mono-project.com/.

[36] Arquitectura de xamarin. https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin.

[37] Como nativescript funciona. https://www.kodehero.in/how-nativescript-works/.

[38] Lo bueno y lo malo de ionic. https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-ionic-mobile-development/.

[39] Bnechmarks backend. https://www.techempower.com/benchmarks/.

[40] Databases. https://blog.pentesteracademy.com/cover-basic-commands-of-mongodb-in-5-minutes-5abcda45969f.

[41] Base de datos columnar. https://help.sap.com/viewer/6b94445c94ae495c83a19646e7c3fd56/2.0.00/en-US/bd2e9b88bb571014b5b7a628fca2a132.html.

[42] Json. https://www.json.org/json-en.html.

[43] Estudio de motores de bases de datos. https://db-engines.com/en/ranking.

[44] Google trends. https://trends.google.com/trends/?geo=CL.

[45] Uriel Infante. Conectando una aplicacion de angular con una api. https://rd.rocktech.mx/publicaciones/entrada/conectando-una-aplicacion-de-angular-con-una-api-rest, Febrero 2020.

[46] Firebase authentication. https://firebase.google.com/products/auth?gclsrc=ds&gclsrc=ds.

[47] Firebase firestore. https://firebase.google.com/products/firestore?hl=es-419&gclsrc=ds&gclsrc=ds&gclid=CJqWoeiZ6fUCFRk_rQYd9VELXg.

[48] Firebase cloud storage. https://firebase.google.com/products/storage?gclsrc=ds&gclsrc=ds&gclid=CImd6o-a6fUCFasdrQYdt_IGzg.

[49] Firebase fcm. https://firebase.google.com/products/cloud-messaging.

[50] Rfc 7231. https://datatracker.ietf.org/doc/html/rfc7231#section-6.5.1, Junio 2014.

[51] Rfc 5789. https://datatracker.ietf.org/doc/html/rfc5789, Marzo 2010.

[52] Rfc 2616. https://datatracker.ietf.org/doc/html/rfc2616#section-10, Junio 1999.

[53] Token jwt. https://jwt.io/.

[54] Rfc 7519. https://datatracker.ietf.org/doc/html/rfc7519.

[55] Gobierno de Chile. Mensaje de s.e. el presidente de la república con el que inicia un proyecto de ley que regula a las aplicaciones de transporte remunerado de pasajeros y los servicios que a través de ellas se presten. https://www.senado.cl/appsenado/index.php?mo=tramitacion&ac=getDocto&iddocto=12456&tipodoc=mensaje_mocion, julio 2018.

[56] Senado de Chile. Proyecto de ley, en segundo trámite constitucional, que regula a las aplicaciones de transporte remunerado de pasajeros y los servicios que a través de ellas se presten. https://www.senado.cl/appsenado/index.php?mo=tramitacion&ac=getDocto&iddocto=3232%&tipodoc=compa, julio 2018.

[57] Google. Informe de movilidad de las comunidades ante el covid-19. https://www.gstatic.com/covid19/mobility/2021-04-14_CL_Mobility_Report_es-419.pdf, Abril 2020.

[58] StrongLoop TJ Holowaychuk et al. Express js. https://expressjs.com, 2010.

[59] Ryan Dahl. node js. https://nodejs.org, Mayo 2009.

[60] Isaac Schlueter. Node package manager. https://www.npmjs.com/, febrero 2014.

[61] LearnBoost. Mongoose. https://mongoosejs.com/, febrero 2011.

[62] Guia para pasajeros en reino unido. https://www.gov.uk/guidance/coronavirus-covid-19-safer-travel-guidance-for-passengers.

[63] Articulo sobre como protegerse del covid en el automovil. https://www.usatoday.com/story/opinion/2020/04/22/coronavirus-car-protect-yourself-column/5166146002/, Abril 2020.

[64] Diagrama sql vs no-sql. https://arstechnica.com/information-technology/2016/03/to-sql-or-nosql-thats-the-database-question/.

[65] Explicación de nodejs. https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/.

[66] Fiber en go. https://github.com/gofiber/fiber.

[67] Dirección regional de arica y parinacota realizó estudio de comparación de precios en pasajes en medios de transporte interurbano. https://www.sernac.cl/portal/619/w3-article-8977.html.