



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Easy Charge, aplicación para la localización de puntos de  
carga de vehículos eléctricos y otras utilidades.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Marí Peñarrubia, Alejandro

Tutor/a: Molina Marco, Antonio

CURSO ACADÉMICO: 2022/2023

# Índice

Glosario.....	3
Prólogo.....	3
Resumen.....	3
Summary.....	4
1 Introducción.....	5
1.1 Motivación.....	5
1.3 Impacto esperado.....	6
1.4 Metodología.....	7
2 Contexto tecnológico.....	8
2.1 Análisis de mercado.....	8
3 Análisis del problema.....	14
3.1 Identificación y análisis de soluciones posibles.....	14
3.2 Identificación de requisitos.....	16
3.3 Lista de requisitos.....	21
3.4 Plan de trabajo.....	22
4 DISEÑO DE LA SOLUCIÓN.....	30
4.1 Arquitectura del Sistema.....	30
4.2 Diseño detallado.....	31
4.3 Tecnología utilizada.....	50
5 Desarrollo de la solución.....	54
5.1 Problemas.....	54
5.2 Dificultades.....	55
5.3 Puntos destacables.....	57
6 Implantación.....	58
6.1 Carga de datos.....	58
6.2 Despliegue.....	59
7 Pruebas.....	60
8 Conclusiones.....	63
9 Referencias.....	65
Índice de figuras.....	68
Índice de tablas.....	69
ANEXO 1 - CÓDIGO.....	70

Consideraciones.....	71
Clase ActiveUser (singleton).....	72
Clase MapChargers ( <i>singleton</i> ).....	73
Clase Charger y clase Memento (Memento).....	75
Paralelización filtrado de puntos de carga.....	78
Pruebas de aceptación GC05 Ocupar cargador y GC06 Desocupar cargador.....	81
ANEXO 2 – DESCRIPCIÓN DE REQUISITOS.....	84
Gestión de usuarios.....	85
Gestión de cargadores.....	87
Gestión del mapa.....	89
ANEXO 3 – OBJETIVOS DE DESARROLLO SOSTENIBLE.....	90

## Glosario

**Punto de carga:** son la infraestructura donde se cargan vehículos eléctricos. Un punto de carga puede estar dividido en uno o más puntos de carga individuales. Abreviaciones: Punto

**Punto de carga individual:** cada uno de los cargadores que componen el punto de carga. Abreviaciones: Cargador.

**EV:** este término hace referencia a los vehículos con batería eléctrica enchufable. Por comodidad también engloba a los vehículos híbridos enchufables. Sinónimos: Vehículo eléctrico.

**GU:** este termino hace referencia a la época de gestión de usuarios.

**GM:** este termino hace referencia a la época de gestión del mapa.

**GC:** este termino hace referencia a la época de gestión de cargadores.

## Prólogo

Esta aplicación tiene como objetivo proporcionar un soporte móvil sencillo para los propietarios de vehículos eléctricos. Encontrar un lugar donde cargar tu vehículo cuando no dispones de un punto de carga en tu casa o necesitas cargarlo fuera no es tarea fácil. Es el momento ideal para que crezcan aplicaciones de este ámbito ya que el mercado de vehículos eléctricos está en auge y cada vez más los usuarios necesitan herramientas que les faciliten ciertos aspectos de este ámbito.

## Resumen

Easy Charge es una aplicación colaborativa que pretende ayudar al usuario con uno de los mayores problemas de tener un vehículo eléctrico: la carga. Entre otras muchas funcionalidades, permite al usuario visualizar los puntos de carga cercanos sobre un mapa para ubicar el más cercano. Otra de las funciones más distintiva es la de marcar un punto de carga como ocupado, por un tiempo concreto o indefinido, lo que permite a otros usuarios de la aplicación identificar un punto como ocupado y evaluar si les conviene ir a un punto más lejano pero disponible, o por el contrario si en breves se va a liberar ir y esperar el tiempo restante. La aplicación también permite añadir información extra sobre los puntos, como averías, horarios, precio, etc.

Si el usuario crea una cuenta podrá acceder a más funcionalidades. Entre ellas destacan un listado de puntos favoritos, un historial de recargas, estadísticas sobre el uso de la batería de tu vehículo y otras más.

Para un primer cargado de datos se ha utilizado una base de datos pública ofrecida por [Open Charge Map](https://map.openchargemap.io/)<sup>1</sup>. Para los puntos de carga que no estuviesen en la base de datos, los usuarios registrados podrán añadirlos a la aplicación.

Para el desarrollo de este proyecto se ha seguido una metodología ágil, utilizando un tablero *Kanban* y *Sprints* para la definición y seguimiento de las unidades de trabajo.

Las tecnologías empleadas son el *framework* de .NET 7.0 con MAUI, un marco multiplataforma que permite hacer aplicaciones para Android, IOS, Windows y Catalyst de manera nativa. La base de datos está alojada en [supabase.com](http://supabase.com/)<sup>2</sup> y emplea PostgreSQL como lenguaje. La conexión a la base de datos se realiza mediante una API REST, que recibe mensajes HTTP. La aplicación obtiene el mapa de la API de Google Maps, que ofrece un gran rango de opciones y funcionalidades fundamentales.

Palabras clave: NET MAUI, PostgreSQL, API, Kanban, Android, Aplicación Móvil, Mapas, Vehículos eléctricos.

## Summary

Easy Charge it's a collaborative app intended to help the user with one of the biggest problems of owning an electric vehicle: the charge. Within all its features, it allows the user to visualise the close charging points on a map so the user can locate the nearest. Another of the most outstanding features is the possibility of marking a charging point as occupied, for a known period of time or undefined, which allows other users in the app to identify the charging point as occupied and assess if it's better go to a charging point that is less close but it is available, or on the other hand, if it is going to be available soon, you prefer going and wait the remaining time. The app also allows the user to add additional information about the charging points such as breakdowns, schedules, price, etc.

If the user creates an account, it can access to more features including a list of favourite charging points, a record of charges, statistics about the battery use of the vehicle and more.

For a first data load, a public database offered by [Open Charge Map](https://map.openchargemap.io/) has been used. For the remaining points that aren't included on the database, the registered users can add them.

The development of this project has followed an agile methodology, using a *Kanban* board and *Sprints* for the definition and monitoring of work units.

The technologies used are .NET 7.0 framework with MAUI, a cross-platform framework that allows native applications for Android, IOS, Windows and Catalyst. The database is hosted on [supabase.com](http://supabase.com/) and uses PostgreSQL as a language. The connection to the database is made through a REST API that receives HTTP messages. The application obtains the map from the Google Maps API, which offers a wide range of options and fundamental functionalities.

---

<sup>1</sup><https://map.openchargemap.io/>

<sup>2</sup><http://supabase.com/>

# 1 Introducción

En este capítulo se presenta la motivación que llevó al desarrollo de este proyecto y las consideraciones previas a su realización, desde un punto de vista personal y profesional.

A día de hoy la infraestructura necesaria para sustentar a los vehículos eléctricos (EV) no sigue el ritmo que la venta de estos mismos. Según [Electromaps](https://www.electromaps.com/es)<sup>3</sup> existen 18.128 puntos de carga, los cuales pueden contar con más de un cargador, pero según la *National Association of Automobile and Truck Manufacturers (ANFAC)*<sup>4</sup> en España en 2022 se han matriculado más de 198.000 vehículos enchufables, de los cuales 127.687 son puramente eléctricos y los 70.532 restantes son híbridos enchufables. El número total de vehículos electrificados asciende hasta los 243.860. La estadística de [Electromaps](https://www.electromaps.com/es) no incluye puntos de carga privados de domicilios particulares, pero aun así hay un claro déficit en el número de puntos de carga frente a la cantidad de vehículos, lo cual se traduce en una preocupación para los propietarios de vehículos eléctricos o personas que se plantean adquirir uno.

## 1.1 Motivación

La motivación de crear Easy Charge viene de una de mis aficiones: los coches. Independientemente de si lo llevaba a cabo como TFG, yo quería crear una aplicación que estuviese relacionada con los coches y además subsanase un problema existente y cumpliera una función real. Además, esta idea la quería combinar con lo aprendido en la rama de ingeniería del software. Aplicar metodologías de desarrollo ágil, desde los diferentes puntos de vista de los participantes de un proyecto: *product owner*, *scrum master* y desarrollador *fullstack*. También aplicar los conocimientos para la recolección de requisitos y su especificación y utilizar un modelo de calidad.

Por eso tras observar el aumento de los vehículos eléctricos comencé a pensar en ideas de aplicaciones enfocadas a ayudar a los propietarios de vehículos eléctrico y tras analizar que problemas y necesidades tienen los propietarios de EV me di cuenta de dos de los mayores problemas a la hora de tener uno de estos vehículos, la batería y la carga. Puesto que subsanar el problema de la batería depende del fabricante del vehículo, decidí centrarme en la carga.

Mucha gente no dispone de un punto de carga en su propia casa, o necesitan cargar el coche estando fuera de casa, ya sea en el trabajo, de viaje, por un imprevisto, etc. Easy Charge te permite localizar el punto de carga más cercano para evitar deambular hasta encontrar uno, además al permitirte marcarlo como ocupado te puedes ahorrar ir a un punto y que no esté disponible y tener que irte a otro o esperar hasta que termine. Esto se combina con una serie de funcionalidades adicionales que complementan al mapa de puntos de carga para convertir a Easy Charge en una aplicación ideal para los propietarios de EV.

---

<sup>3</sup>Electromaps <https://www.electromaps.com/es>

<sup>4</sup>ANFAC <https://anfac.com/>

Al plantear formalmente la idea realicé una búsqueda de aplicaciones similares y en efecto no era la única que planteaba esta solución. Pero al igual que en otros ámbitos de aplicaciones como redes sociales (Facebook, Twitter, Instagram, etc) o aplicaciones de mensajería (WhatsApp, Telegram, SMS, etc), ofrecer variedad de opciones para que el usuario decida cuál de las aplicaciones se acopla mejor a sus necesidades es importante para el usuario, además fomenta la competitividad entre aplicaciones y eso genera mejora y desarrollo.

## 1.2 Objetivos

La misión de Easy Charge es, de la manera más sencilla y accesible para el mayor número de usuarios posible, ayudar a los propietarios de vehículos eléctricos en la mayor cantidad de aspectos posibles del día a día de poseer un vehículo de estas características, para así reducir el tiempo que deben emplear en realizar las tareas asociadas a tener un vehículo de estas características. Además, al reducir dichas preocupaciones de los propietarios, se fomenta la compra de estos vehículos por parte de potenciales compradores.

Para conseguir esta misión hay varios objetivos que la aplicación debe cumplir:

- Mostrar los puntos de carga cercanos al usuario.
- Ser lo más sencillo de utilizar para el usuario.
- Mostrar la información indispensable como la localización, tipo de puertos de carga, número de cargadores y disponibilidad.
- Ser eficiente y eficaz a la hora de realizar los casos de uso.
- Disminuir el tiempo que emplea el usuario en realizar estos casos de uso sin utilizar la aplicación.

La aplicación está construida teniendo en mente estos objetivos como directriz. Por estos motivos se ha intentado mantener la interfaz de la aplicación con el mínimo necesario para que el usuario pueda realizar todas las funciones, sin sobrecargarlo con menús y botones innecesarios. Lo mismo sucede con las pantallas de puntos de carga. Se ha intentado que el usuario tenga que hacer el mínimo número de interacciones con la aplicación para poder encontrar el punto de carga adecuado, obtener la información que necesite y ocuparlo.

## 1.3 Impacto esperado

Easy Charge va dirigida a las personas que disponen de un EV, dentro de esa sección de público potencial se ha intentado hacer la aplicación lo más sencilla y accesible posible para abarcar un mayor número de usuarios.

Easy Charge permitirá al usuario reducir el tiempo que emplea en gestionar su EV y además hacerlo de una manera cómoda y sencilla. Al disponer de un mapa con los puntos de carga, es capaz de visualizar rápidamente cual es el punto más cercano desde cualquier localización para no perder tiempo buscando, ya sea físicamente o a través de por ejemplo Google Maps. También permite al usuario ver si el punto al que tiene pensado dirigirse está ocupado o no, y

así no perder tiempo yendo al punto y tener que esperar a que termine o dirigirse al siguiente más cercano. Permite al usuario definir recordatorios para no olvidarse de cargar su vehículo.

Existen diferentes perfiles dentro de los potenciales usuarios de la aplicación:

- **Perfil menos experto.** Este tipo de perfil solo quiere la aplicación para ver los puntos del mapa, ver cuál es el más cercano y si están disponibles o no. Para este perfil la aplicación permite hacer uso de estas funciones sin necesidad de crear una cuenta. Todas estas funciones se encuentran contenidas en la página principal de la aplicación, por lo que no necesita navegar dentro de la aplicación para encontrarlas.
- **Perfil estándar.** A este perfil le interesa obtener más información sobre los cargadores, como podría ser la compañía propietaria del punto, horarios, precio, etc. Estas funcionalidades también están disponibles sin crear una cuenta, pero lo más probable es que se cree una cuenta. Este perfil añadirá los puntos de carga más visitados a favoritos, para encontrarlos rápidamente y quizás defina recordatorios y acceda al historial de recargas.
- **Perfil experto.** Este perfil va a ser el más minoritario. A este perfil le interesa obtener toda la información posible sobre los puntos, y quizás modificarla por si hay errores. Este perfil posiblemente haga mucho uso de su EV y le interesa disponer de estadísticas sobre el uso de la batería, distancia recorrida entre recargas, historial de recargas, defina recordatorios y añada observaciones sobre los puntos.

## 1.4 Metodología

La metodología empleada para la realización de este proyecto es la estudiada en las asignaturas de proceso de software y proyecto de ingeniería de software, una metodología ágil pensada para equipos reducidos, haciendo uso de un tablero *Kanban* y dividiendo el trabajo en *sprints* y las funcionalidades en épicas y unidades de trabajo (UT). No se ha planteado una división de roles ya que solo hay una persona desarrollando el proyecto que asume todos los roles del equipo.

Se ha escogido este tipo de metodología ya que es la más adecuada para proyectos con equipos cuyo número de integrantes no sea muy grande, en este caso un solo miembro. Esta metodología permite trabajar de una manera rápida y eficaz y realizar el seguimiento del progreso de manera sencilla, ágil y flexible. La metodología ágil fomenta también el aspecto *fullstack* de los integrantes del equipo y promueve un desarrollo más cercano al usuario que finalmente utilizara la aplicación.

La tendencia en la mayoría de las empresas modernas es trabajar con metodologías ágiles y cada vez hay más herramientas disponibles que dan soporte a estas metodologías, lo que las convierte en la opción más adecuada para este tipo de proyectos.

Para establecer el alcance del proyecto se ha realizado un análisis de mercado de 3 de las aplicaciones con más descargas de la Play Store de Android, para identificar qué es lo imprescindible e identificar las funcionalidades diferenciadoras.



## 2 Contexto tecnológico

En este capítulo se va a hablar del contexto general del ámbito del proyecto. Los análisis realizados de las diferentes aplicaciones ya existentes en el mercado, de la organización que ha realizado dichas aplicaciones, de sus funcionalidades, los puntos fuertes, débiles y nuestra propuesta para entrar en el mercado.

### 2.1 Análisis de mercado

En el mercado actualmente ya existen otras aplicaciones similares que comparten objetivo. Esto no es un impedimento ya que siempre nos encontramos aplicaciones diferentes que comparten el mismo objetivo. Por ejemplo, WhatsApp y Telegram, Skype y Teams, Google Chrome y Microsoft Edge, etc. Es importante también para el usuario poder elegir entre diferentes aplicaciones y escoger la que mejor se adapte a sus necesidades y gustos concretos. Por este motivo se ha llevado a cabo un estudio de las 3 aplicaciones más populares de la Play Store que comparten objetivo con el fin de identificar los puntos donde nuestra aplicación puede destacar sobre el resto.

#### Análisis de competidores

Competidor	Electromaps	Chargemaps	Recarga pública Iberdrola
Fecha de creación	2009	2015	2019
Nivel (1-3)	Nivel 1: competidor principal	Nivel 1: competidor principal	Nivel 1: competidor principal

Tabla 1: Análisis de competidores

#### Perfil del competidor

Aplicación	Electromaps	Chargemaps	Recarga pública Iberdrola
Misión de la empresa	Crear un mapa que localice los puntos de carga de vehículos eléctricos y añada funcionalidades para los usuarios.	Crear un mapa que localice los puntos de carga de vehículos eléctricos y añada funcionalidades para los usuarios.	Crear un mapa que localice los puntos de carga de vehículos eléctricos y el pago a través de su plataforma.
Principales objetivos	Localizar puntos de carga, informar a los	Localizar puntos de carga, informar a los	Localizar puntos de carga, pagar desde la

	usuarios, pagar desde la aplicación.	usuarios, pagar desde la aplicación.	aplicación.
<b>Funciones</b>	Permite el pago desde la aplicación mediante una cartera virtual, guardar puntos de carga en favoritos, registrar tus recargas, indicar el número de cargadores y si existen averías.	Permite el pago desde la aplicación mediante una cartera virtual, guardar puntos de carga en favoritos, registrar tus recargas, indicar el número de cargadores y si existen averías, permite filtrar por diferentes parámetros y planificar un viaje.	Permite el pago desde la aplicación mediante una cartera virtual, guardar puntos de carga en favoritos, registrar tus recargas, indicar el número de cargadores y si existen averías, permite filtrar por diferentes parámetros y planificar un viaje.
<b>Tamaño de la empresa</b>	30 empleados, 355.599 miembros registrados	51 empleados 1.556.142 miembros registrados	Empleados desconocidos, +- 100.000 descargas de la aplicación.

*Tabla 2: Perfil del competidor*

## Nuestra ventaja competitiva

- La principal ventaja es la posibilidad de marcar un punto de carga como ocupado, función que permite al usuario identificar los puntos de carga que no puede utilizar rápidamente.
- Otra ventaja es que no hace falta crear una cuenta para utilizar la aplicación mientras que Electromaps te requiere crear una cuenta para utilizarla. Chargemaps no te lo requiere, pero limita el rango de funciones de las que dispones y Recarga pública Iberdrola también.
- Otra ventaja sería una menor complejidad a la hora de utilizar la aplicación, la idea es que sea sencilla y cómoda de utilizar.

## Oferta de productos

Aplicación	Electromaps	Chargemaps	Recarga pública Iberdrola	Easy Charge
<b>Resumen del producto</b>	Mapa para localizar puntos de carga de EV. Permite el pago desde la aplicación.	Mapa para localizar puntos de carga de EV. Permite el pago desde la aplicación.	Mapa para localizar puntos de carga de EV. Permite el pago desde la aplicación.	Mapa para localizar puntos de carga de EV
<b>Posicionamiento/categoría</b>	Ya instaurado en el mercado	Ya instaurado en el mercado	Ya instaurado en el mercado	Nuevo
<b>Precios</b>	Gratis - 15€	Gratis - 19,90€	Gratis - 26,95€	Gratis, más adelante podrá contar con una versión de la aplicación de pago que añade funcionalidades extra por 2.99€

Tabla 3: Oferta de productos

## Comparación de funciones básicas

Electromaps	Chargemaps	Recarga pública Iberdrola	Easy Charge
✓ Localizar puntos de carga	✓ Localizar puntos de carga	✓ Localizar puntos de carga	✓ Localizar puntos de carga
✓ Marcar un punto como averiado	✓ Marcar un punto como averiado	✓ Marcar un punto como averiado	✓ Marcar un punto como averiado
✗ Marcar un punto individual como averiado	✗ Marcar un punto individual como averiado	✓ Marcar un punto individual como averiado	✓ Marcar un punto individual como averiado
✗ Marcar un punto como ocupado	✗ Marcar un punto como ocupado	✓ Marcar un punto como ocupado	✓ Marcar un punto como ocupado
✗ Programar alarmas, notificaciones o recordatorios	✗ Programar alarmas, notificaciones o recordatorios	✗ Programar alarmas, notificaciones o recordatorios	✓ Programar alarmas, notificaciones o recordatorios
✗ Análisis sobre el uso de la batería de tu EV	✗ Análisis sobre el uso de la batería de tu EV	✗ Análisis sobre el uso de la batería de tu EV	✓ Análisis sobre el uso de la batería de tu EV
✓ Añadir puntos de carga	✓ Añadir puntos de carga	✗ Añadir puntos de carga	✓ Añadir puntos de carga
✓ Guardar en favoritos	✓ Guardar en favoritos	✓ Guardar en favoritos	✓ Guardar en favoritos
✓ Historial de recargas	✗ Historial de recargas	✗ Historial de recargas	✓ Historial de recargas
✓ Pago desde la aplicación	✓ Pago desde la aplicación	✓ Pago desde la aplicación	✗ Pago desde la aplicación
✗ Planificar ruta	✓ Planificar ruta	✓ Planificar ruta	✗ Planificar ruta
✓ Filtro de cargadores	✓ Filtro de cargadores	✓ Filtro de cargadores	Filtro de cargadores

Tabla 4: Comparación de funciones básicas

Del análisis de las tablas 3 y 4 se observa que las otras aplicaciones cuentan con una serie de funcionalidades muy completas que cubren muchos aspectos complejos de tener un EV. Pero descuidan una parte del sector que está comenzando a crecer ahora. Antes los propietarios de EV eran una minoría en comparación a los de combustión tradicionales, y dichos propietarios eran gente que disponía de mayores recursos y un conocimiento más amplio del tema. Ahora disponer de un EV no es algo minoritario y cada vez más gente los compra por lo que los usuarios de estas aplicaciones cada vez están menos especializados y requieren de aplicaciones más sencillas les permitan realizar ciertas funciones como poner recordatorios para recargar tu vehículo o ver si un punto está ocupado.

Por esto Easy Charge apuesta por incluir menos funcionalidades complejas, como podría ser calcular una ruta con sus puntos de carga necesarios, pero mantener una interfaz y una usabilidad más sencilla para todos los usuarios, incluyendo funcionalidades avanzadas también para los usuarios expertos, como podría ser el análisis del uso de la batería de tu vehículo, pero sin añadir complejidad innecesaria a los usuarios que no van a acceder a estas funcionalidades.

### **Modelo de negocio**

La aplicación será gratuita para los usuarios. Los usuarios dispondrán de todas las funciones al crearse una cuenta. La aplicación obtendrá ingresos de anuncios, que se podrán mostrar en la parte inferior de la pantalla, sin llegar a bloquear todo el contenido, ya que esto es bastante incómodo y empeora la experiencia de los usuarios.

Más adelante la aplicación podrá contar con una versión de pago de la misma, que incluya funcionalidades adicionales como análisis en tiempo real sobre el funcionamiento y estado de tu vehículo eléctrico, conectar con la plataforma de los puntos de carga y eliminar los anuncios de la aplicación.

### **Análisis DAFO**

Para analizar en detalle el proyecto se ha realizado un análisis DAFO además del estudio ya visto en el estudio de los competidores.

<b>Fortalezas</b>	<b>Oportunidades</b>
<ul style="list-style-type: none"> <li>• Poder marcar un punto de carga como ocupado es una función muy útil para los usuarios.</li> <li>• La facilidad de uso de la aplicación es un factor clave para atraer usuarios.</li> </ul>	<ul style="list-style-type: none"> <li>• Es un mercado en auge ya que cada vez más gente dispondrá de vehículos eléctricos.</li> </ul>
<b>Debilidades</b>	<b>Amenazas</b>
<ul style="list-style-type: none"> <li>• 1 sola persona trabajando en el proyecto.</li> <li>• Bajo presupuesto para el proyecto.</li> </ul>	<ul style="list-style-type: none"> <li>• Otras aplicaciones ya existentes que disponen de pago desde la aplicación.</li> <li>• Otras aplicaciones ya existentes con una base de usuarios y datos ya establecida.</li> </ul>

*Tabla 5: Análisis DAFO*

Las debilidades encontradas se podrían solventar a medida que avanza el proyecto, más personas podrían ser incluidas en el proyecto y también se aumentaría el presupuesto en general, tanto para contratar personal como desarrolladores, diseñadores, ampliar la base de datos, mejorar la tecnología con el uso de herramientas con licencia de pago, etc.

Por el contrario, las amenazas son complicadas de solventar, ya que depende de las otras aplicaciones ya instauradas en el mercado, pero conforme la aplicación crezca, dicha amenaza se irá reduciendo hasta que la aplicación quede instaurada en el mercado como un competidor más.

## 3 Análisis del problema

En este capítulo se van a identificar y analizar las diferentes elecciones tomadas a la hora de realizar el proyecto. Se van a explicar el porqué de las elecciones y a grandes rasgos describir las alternativas a las elecciones tomadas.

### 3.1 Identificación y análisis de soluciones posibles

Para solventar el problema de la localización de puntos de carga para EV existen diferentes soluciones. Es verdad que la plataforma universal para prácticamente todos los dispositivos es la *Web*, y dicha opción se planteó al comienzo del proyecto. Tiene una serie de pros y contras que ahora detallaré.

#### Pros:

- **Multiplataforma.** La *web* actualmente es la plataforma que más dispositivos abarca. Desde prácticamente cualquier dispositivo con acceso a internet puedes conectarte a páginas *web*. Una excepción podrían ser algunos centros multimedia de modelos de vehículos no tan recientes o comprados de manera independiente.
- **Ligereza.** Al acceder a una página *web* los usuarios no tienen que descargar ninguna aplicación. La carga de la base de datos sería la misma y solo haría falta una máquina donde se ejecutase el *backend*.
- **Flexibilidad.** El diseño de las aplicaciones *web* es bastante flexible independientemente de la plataforma desde la cual se acceda. Los cambios entre dispositivos suelen ser de tamaño y suelen ser programados de manera que una única interfaz se reajusta al dispositivo desde el cual se accede.
- **Actualizado.** Al ser una página *web* se actualiza cuando el propietario decide y dicha actualización es inmediata para todos los usuarios que accedan a la *web*.

#### Contras:

- **Usabilidad.** Es un hecho que acceder a páginas *web* en dispositivos móviles es más costoso que utilizar una aplicación, por eso la tendencia de las compañías como Meta o Twitter es disponer de una *web* para ordenadores y una aplicación para dispositivos móviles. Aunque las aplicaciones comienzan a tener más influencia en ordenadores también.
- **Eficiencia.** Las aplicaciones nativas suelen ser más rápidas y eficientes que los servicios *web*, también depende del dispositivo desde el cual se ejecuta y de la calidad de las máquinas donde se ejecute el *backend* de la aplicación *web*, pero en términos generales el rendimiento es mejor en aplicaciones nativas.
- **Uso del dispositivo.** Al ser una aplicación instalada en el dispositivo nos permite acceder a funciones del mismo, como podrían ser las alarmas, notificaciones, cámara, y otros recursos del dispositivo.

Todos estos factores se tuvieron en cuenta a la hora de decantarse por aplicación o página web. La usabilidad es uno de los factores clave de la aplicación y la capacidad de multiplataforma se podía solventar utilizando ciertos *frameworks* que permiten la creación de aplicaciones multiplataforma, por lo que me decanté por el uso de una aplicación móvil.

Para la elección de la tecnología sobre la cual realizar la aplicación también se analizaron diferentes opciones: Kotlin, Swift, Angular-Capacitor, .NET MAUI.

Kotlin es un lenguaje de programación de código abierto creado por JetBrains que ha ganado gran relevancia en la programación de aplicaciones para Android. Es un lenguaje de programación estático que admite la programación funcional y orientada a objetos. Está diseñado para ser totalmente compatible con Java y es el lenguaje en el que se desarrollan mayoritariamente las aplicaciones producidas por Google para Android.

Swift es un lenguaje de programación multiparadigma creado por Apple para programar aplicaciones en las plataformas de iOS y macOS. Swift es un lenguaje rápido y eficaz que proporciona información en tiempo real y se integra con Objective-C, un superconjunto del lenguaje C empleado sobre todo en Mac OS X, iOS y GNUstep.

Angular es un *framework* para aplicaciones web desarrollado en TypeScript, de código abierto y mantenido por Google. La principal característica de este framework es que crea aplicaciones web de una sola página (SPA) y *Progressive Web App* (PWA). Con Capacitor, puedes crear soluciones multiplataforma que aprovechen las características de las *Progressive Web Apps* y permite ser nativo en Android e iOS.

.NET MAUI es un *framework* multiplataforma que permite desarrollar de manera nativa aplicaciones móviles y de escritorio utilizando únicamente como base C# y XAML. Con .NET MAUI, puedes desarrollar aplicaciones que se pueden ejecutar en Android, iOS, macOS y Windows desde una sola base de código compartida.

Para no incluir todos los detalles de todas las tecnologías mencionaré los pros y los contras de .NET MAUI, que ha sido el *framework* elegido.

### Pros:

- **Multiplataforma.** Ofrece un más que suficiente rango de dispositivos donde se puede desplegar la aplicación. Android, IOS, Web, Catalist. Esto cubre la totalidad de los dispositivos donde se planea desplegar la aplicación.
- **Simplicidad** de desarrollo. Este pro tiene un motivo más personal y es que .NET MAUI programa la lógica en C#, lenguaje en el que tengo amplia experiencia frente a otros lenguajes como Kotlin. Además, considero que aun sin conocer el lenguaje es más simple de entender y desarrollar con el que el resto de las tecnologías.
- **Soporte y documentación.** .NET MAUI pertenece a Microsoft y es parte del entorno de .NET por lo que dispone de una amplia documentación oficial y el soporte de Microsoft se encarga de mejorar esta tecnología de manera constante. También se dispone de infinidad de documentación sobre C# y en general del entorno .NET.



**Contras:**

- **Diseño visual.** Visualmente es la tecnología con quizás el diseño visual menos elaborado. Esto se puede suplir utilizando librerías, pero es la opción más limitada en este ámbito.
- **Menos funciones.** Esta tecnología es la más reciente de todas por lo que hay componentes y funcionalidades que no están todavía desarrolladas, como por ejemplo fijar el tamaño de componentes en función de un % de tamaño de pantalla.
- **Bugs.** Al ser la más reciente, como con el menor número de funciones, también contiene algunos *bugs*.

Teniendo en cuenta los pros y contras no estaba seguro de escoger .NET MAUI como tecnología para el proyecto, pero al igual que tiene contras por ser la tecnología más reciente y más nueva, tiene puntos positivos y como todo lo que se encuentra en estadios tempranos de su ciclo de vida, irá mejorando conforme se desarrolle con el tiempo. Por eso he decidido apostar por la .NET MAUI ya que creo que ahora mismo es una opción muy competitiva que además tiene un potencial de mejora mucho mayor que el resto de las tecnologías.

## 3.2 Identificación de requisitos

En este apartado se va a explicar el proceso mediante el cual se han obtenido los requisitos y se han definido las UT del proyecto. No todas las funcionalidades han surgido de estos procesos, algunos requisitos han surgido durante el desarrollo de la aplicación, pero es importante remarcar la importancia de realizar estos ejercicios previos para identificar los posibles requisitos de la aplicación y plantear un correcto inicio del desarrollo.

### Lluvia de ideas

Para la obtención de requisitos hice un planteamiento de lluvia de ideas en solitario y luego una sesión con 4 personas incluyéndome a mí. Debido al reducido número de participantes asumí el rol de moderador y secretario a la vez que el de participante.

Estas 2 sesiones fueron muy fructíferas ya que después de la primera sesión en solitario, donde una gran parte de los requisitos fueron planteados, en la segunda sesión los demás participantes también expusieron ideas diferentes que ayudaron a cubrir casuísticas que en la primera fase no se habían cubierto.

Algunas de las ideas planteadas que fueron descartadas son:

- Chat (directo) entre usuarios.
- Posibilidad de descargar el mapa para su uso *offline*.
- Visualización del estado de los puntos de carga vía satélite.
- Descuentos en las recargas.

Estos son ejemplos de algunas ideas planteadas en las sesiones de lluvia de ideas. Unas son más realistas, como el chat o el mapa *offline*, pero fueron descartados por no aportar el


suficiente valor o no estar completamente alineadas con la misión de la aplicación. En cambio, otras ideas fueron descartadas por ser demasiado fantasiosas o imposibles de realizar. Pongo estos ejemplos ya que quiero resaltar que la libertad a la hora de plantear ideas fomenta la aparición de otras ideas que luego si son incluidas en el desarrollo.

Algunas de las ideas planteadas que sí fueron incluidas son:

- Comentarios en los cargadores.
- Notificaciones y alertas programables.
- Modo satélite en el mapa.
- Historial de recargas.
- Puntos de carga favoritos.

## Personas

Posterior a estas 2 sesiones de lluvias de ideas definí una serie de personas que representan a los usuarios modelos. Definí 3 personas, una por cada perfil planteado anteriormente:

<p>Paco</p>  <p><i>Figura 1: Persona 1.</i></p> <p style="text-align: right;">5</p>	<p><b>Descripción:</b> Paco es informático en una gran empresa. La oficina está lejos de su casa por lo que repara mucho en lo que gasta en desplazamiento. Paco tiene un coche totalmente eléctrico y se interesa mucho por el consumo de su coche y su eficiencia</p>
<p><b>Personal:</b> Trabajo: Ingeniero informático Edad: 30 años Ingresos: Medio-alto Soltero Metódico y preciso</p>	<p><b>Tecnología y conocimientos:</b> Conocimientos informáticos altos. Inglés y alemán. Alto nivel con smartphones y dispositivos móviles.</p>
<p><b>Motivaciones:</b> Ahorrar en los desplazamientos. Aprovechar al máximo la batería de su coche. Disponer siempre de un punto de carga cercano. Agilizar la gestión y tiempo dedicado a su EV.</p>	<p><b>Objetivos:</b> Disponer de una aplicación donde pueda localizar los puntos de carga cercanos y obtener información sobre el rendimiento y compatibilidad de los mismos. Poder hacer un seguimiento del uso de la batería de su coche. Poder guardar los puntos más visitados en favoritos. Realizar el pago y las gestiones de las recargas desde la aplicación.</p>


*Tabla 6: Persona 1*

<sup>5</sup>Figura 1. Extraída del banco de imágenes de Pexels <https://www.pexels.com/es-es/>

<p>María</p>  <p><i>Figura 2: Persona 2.</i> <sup>6</sup></p>	<p>Descripción:</p> <p>María es contable en una gran empresa del centro de su ciudad. La oficina está en una ZBE y utiliza un coche híbrido para desplazarse al trabajo. María utiliza también su vehículo para irse de escapadas los fines de semana con su pareja.</p>
<p>Personal:</p> <p>Trabajo: Contable          Edad: 33 años          Ingresos: Medio-alto          Casada          Espontánea y despistada</p>	<p>Tecnología y conocimientos:</p> <p>Conocimientos informáticos medios.          Inglés y francés.          Nivel medio-alto con smartphones y dispositivos móviles.</p>
<p>Motivaciones:</p> <p>Ahorrar en los desplazamientos.          Utilizar la menor cantidad posible de gasolina.          Ayudar al medioambiente.          Tener siempre la batería de su coche cargada.</p>	<p>Objetivos:</p> <p>Disponer de una aplicación donde pueda localizar los puntos de carga.          Tener algún sistema para no olvidarse de cargar su coche.          Poder guardar los puntos más visitados en favoritos.          Realizar el pago y las gestiones de las recargas desde la aplicación.          Poder buscar los cargadores de los sitios que visita.</p>

*Tabla 7: Persona 2*

<sup>6</sup>Figura 2: Extraída del banco de imágenes de IStock <https://www.istockphoto.com/es/fotograf%C3%ADas-de-stock>

<p>Salvador</p>  <p><i>Figura 3: Persona 3</i> <sup>7</sup></p>	<p>Descripción: Salvador está prejubilado. Con el dinero de la prejubilación se compró un coche híbrido para aprovechar y viajar con su mujer.</p>
<p>Personal: Trabajo: Prejubilado Edad: 54 años Ingresos: Medio Casado Despreocupado</p>	<p>Tecnología y conocimientos: Conocimientos informáticos bajos. Nivel medio-bajo con smartphones y dispositivos móviles.</p>
<p>Motivaciones: Ahorrar en los desplazamientos. Poder entrar en las ZBE</p>	<p>Objetivos: Disponer de una aplicación donde pueda localizar los puntos de carga. Realizar el pago y las gestiones de las recargas desde la aplicación. Disponer de una aplicación sencilla que le permita utilizar sus funciones sin muchas complicaciones.</p>

*Tabla 8: Persona 3*

De este análisis de las personas se consolidó la misión de la aplicación, de ser una aplicación para localizar puntos de carga sobre un mapa, que además incluya una serie de funcionalidades extra para ayudar a los propietarios de vehículos con diferentes aspectos, pero que a su vez fuese sencilla de utilizar para usuarios que no tienen por qué tener un alto nivel en el uso de *smartphones* y dispositivos móviles.

Algunos de los requisitos identificados en este proceso son:

- Estadísticas sobre el uso de la batería del vehículo.
- Información adicional sobre los puntos de carga (velocidad de carga, duración máxima, etc).
- Localizar el punto de carga más cercano.
- Buscador de puntos de carga.

## Análisis de competidores

El último proceso de elicitación y especificación de requisitos fue el análisis de los principales competidores de la aplicación. Al analizar las 3 aplicaciones descritas anteriormente me di

<sup>7</sup>Figura 3: Extraída del banco de imágenes de Pexels <https://www.pexels.com/es-es/>

cuenta de algunas funcionalidades que había pasado por alto y realmente son imprescindibles en una aplicación de este ámbito.

Algunos de los requisitos identificados en este proceso son:

- Poder filtrar por diferentes características de los puntos de carga.
- Valorar los puntos de carga.
- Conectar con la plataforma propietaria del punto de carga.

### 3.3 Lista de requisitos

Los requisitos están agrupados por épicas. Las diferentes épicas hacen referencia a grupos que engloban funcionalidades similares o complementarias. Hay requisitos que se podrían categorizar dentro de diferentes épicas, pero al solo pueden estar en una.

#### Gestión de usuarios

- [GU01](#) Nombre en *Shell*: Poder visualizar el nombre del usuario en el menú lateral de la aplicación.
- [GU02](#) Iniciar sesión: Iniciar sesión en la aplicación con una cuenta ya creada.
- [GU03](#) Registrar usuario: Crear una nueva cuenta de usuario.
- [GU04](#) Cambiar la contraseña: Poder cambiar la contraseña mediante una confirmación.
- [GU05](#) Cerrar sesión: Poder cerrar una sesión iniciada.
- [GU06](#) Añadir cargadores a favoritos: Añadir un cargador a la lista de favoritos.
- [GU07](#) Eliminar cargadores favoritos: Poder eliminar un cargador de la lista de favoritos.
- [GU08](#) Ver cargadores favoritos: Poder visualizar la lista de cargadores favoritos.
- [GU09](#) Crear alarmas/recordatorios: Crear alarmas y recordatorios programables en el tiempo.
- [GU10](#) Visualizar alarmas/recordatorios: Visualizar las alarmas y recordatorios programados.
- [GU11](#) Eliminar/ pausar las alarmas/recordatorios: Eliminar o pausar las alarmas y recordatorios programados.
- [GU12](#) Visualizar el historial de recargas: Visualizar la información de las recargas realizadas por el usuario.
- [GU13](#) Ver estadísticas del coche: Visualizar diferentes estadísticas sobre el uso de la batería del vehículo.
- [GU14](#) Añadir coche: Poder asociar un vehículo al usuario.
- [GU15](#) Eliminar coche: Poder eliminar un vehículo asociado a un usuario.
- [GU16](#) Visualizar perfil de usuario: Visualizar el perfil del usuario con sus datos.
- [GU17](#) Editar perfil de usuario: Modificar datos del perfil del usuario.
- [GU18](#) Añadir foto de perfil: Poder subir una foto para personalizar el perfil de usuario.
- [GU19](#) Añadir fotos del vehículo: Poder subir fotos de los vehículos asociados al usuario.

## Gestión de cargadores

- [GC01](#) Crear punto de carga: Crear un punto de carga en la ubicación indicada.
- [GC02](#) Modificar datos del punto de carga: Poder modificar los datos de un punto de carga.
- [GC03](#) Ver punto de carga: Visualizar la información y el estado del punto de carga.
- [GC04](#) Información adicional: Añadir información adicional sobre el punto de carga (precio, horario, propietario, información de contacto, etc).
- [GC05](#) Ocupar cargador: Poder marcar un punto de carga individual como ocupado, por un tiempo fijo o indefinido.
- [GC06](#) Desocupar cargador: Poder marcar un punto de carga individual que estaba ocupado como disponible.
- [GC07](#) Marcar cargador como averiado: Poder marcar un punto de carga individual como averiado.
- [GC08](#) Marcar punto de carga como averiado: Poder marcar el punto de carga completo como averiado.
- [GC09](#) Añadir foto del punto de carga: Poder añadir imágenes del punto de carga.
- [GC10](#) Comentarios/chat/foro: Poder añadir comentarios e interactuar con otros usuarios sobre el punto de carga.
- [GC11](#) Conectar con la plataforma del punto de carga: Poder realizar las gestiones físicas del punto de carga desde la aplicación.
- [GC12](#) Pagar desde la aplicación.

## Gestión del mapa

- [GM01](#) Insertar mapa: Mostrar el mapa en la pantalla principal.
- [GM02](#) Ubicación del usuario: Indicar la ubicación del usuario en tiempo real.
- [GM03](#) Añadir puntos de carga al mapa: Poder visualizar los puntos de carga sobre el mapa.
- [GM04](#) Cargador más cercano: Poder navegar rápidamente hasta el cargador más cercano al usuario.
- [GM05](#) Personalizar mapa: Poder elegir el modo del mapa.
- [GM06](#) Mejoras visuales al mapa: Marcadores de puntos de carga personalizados y mejoras visuales personalizadas al mapa.
- [GM07](#) Filtrar puntos de carga por estado.

## 3.4 Plan de trabajo

Como he mencionado en un punto anterior, para el desarrollo de este proyecto se ha escogido la metodología ágil. En este punto se entrará en detalle de como se ha llevado a cabo el desarrollo del proyecto.

El proyecto se ha dividido en 4 *sprints* de aproximadamente 1 mes de duración, siendo el primero una preparación de la infraestructura necesaria para comenzar el desarrollo. En los 2 siguientes se han desarrollado las funcionalidades principales de la aplicación, obteniéndose

un MVP (mínima versión viable) en cada uno de los sprints En El último se han desarrollado funcionalidades secundarias, se han hecho mejoras estéticas, refactorización y se ha preparado la aplicación para su lanzamiento como versión 1.0.

Como se observará no todas las unidades de trabajo se corresponden con un requisito de la aplicación y también hay unidades de trabajo que hacen referencia al mismo requisito ya que por distribución del esfuerzo y definición era más sencillo dividirlo en diferentes UT.

## Sprint 0

Dentro del *sprint* 0 se han realizado las tareas asociadas al comienzo del proyecto y el despliegue de la infraestructura necesaria para crear la aplicación.

### Listado de UT

- Crear los *mock-ups*.
- Crear el proyecto y el emulador.
- Crear los diagramas.
- Crear la base de datos.
- Crear los modelos de datos.
- Crear los temas de la aplicación.
- Conseguir la API de Google Maps.
- Identificar los requisitos y crear las UTs .



Figura 4: Gráfica trabajo completado Sprint 0.



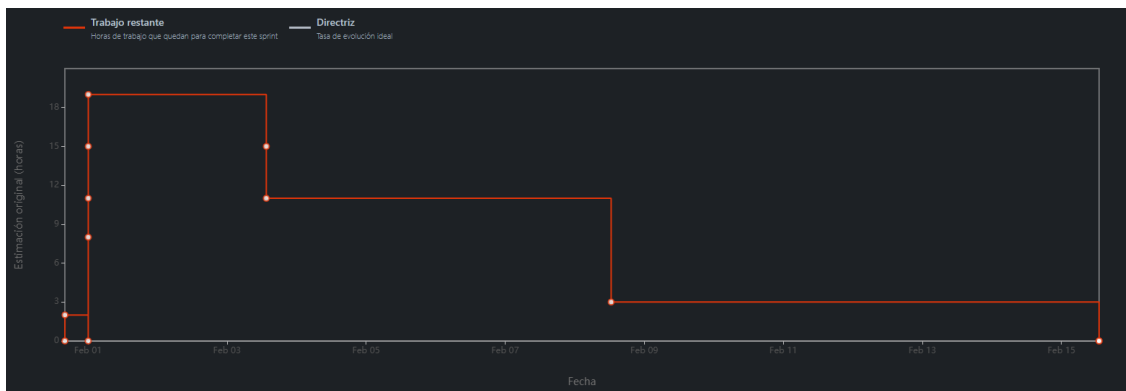


Figura 5: Gráfica trabajo pendiente Sprint 0.

Como se observa en las figuras 4 y 5 la finalización de tareas es bastante lineal. Las tareas de este *sprint* eran tareas grandes que no se podían subdividir.

## Sprint 1

El objetivo de este *sprint* era crear la base de la aplicación, con las funcionalidades más importantes sobre las cuales se basan el resto de las funcionalidades. En este *sprint* se obtiene el primer Producto Mínimo Viable (MVP) de la aplicación, donde ya se puede localizar los puntos de carga sobre el mapa, la ubicación del usuario, visualizar los puntos de carga con su información y su estado, etc.

### Listado de UT:

- [GM01](#) Insertar mapa.
- [GC01](#) Crear punto de carga.
- [GM03](#) Añadir punto de carga al mapa.
- [GC03](#) Ver punto de carga.
- [GC03](#) Control *swipe* cargadores individuales.
- [GM02](#) Ubicación del usuario.
- [GC05](#) Ocupar cargador.
- [GC05](#) Ocupar cargador tiempo fijo.
- [GM02](#) Distancia entre cargador y usuario.
- [GM04](#) Navegar al punto de carga más cercano.
- [GC07](#) Marcar cargador como averiado.
- Optimizar velocidad de respuesta de la página principal.

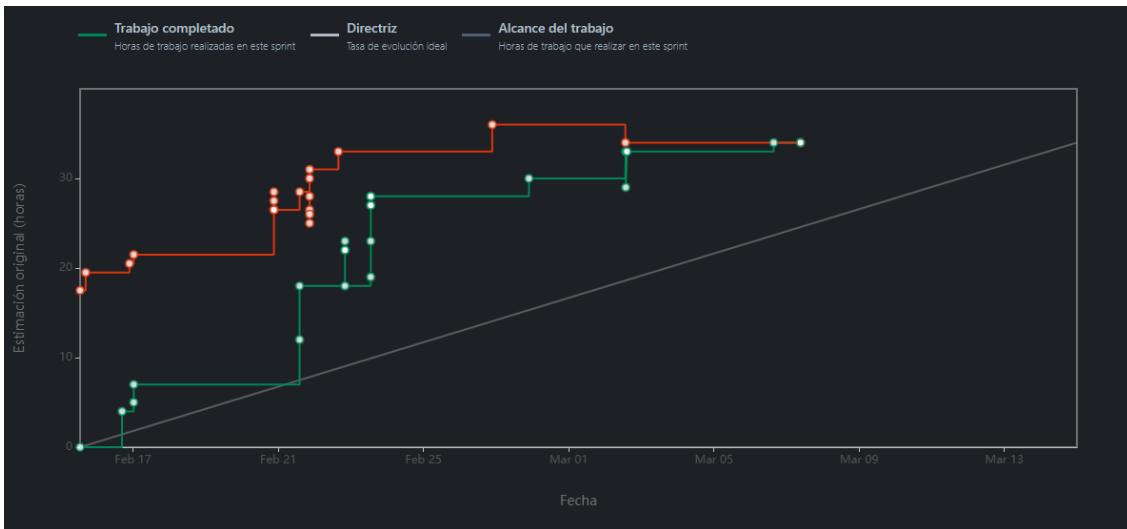


Figura 6: Gráfica trabajo completado Sprint 1.

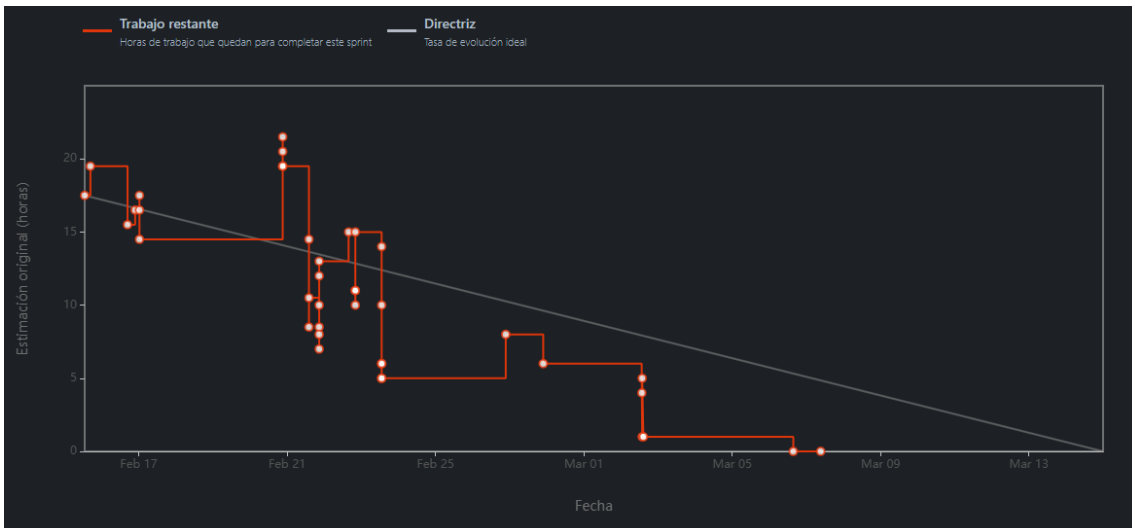


Figura 7: Gráfica trabajo pendiente Sprint 1.

En la figura 7 se observa como el trabajo de este *sprint* fue completado de manera bastante lineal, superando la estimación de trabajo para el *sprint*. Se tuvieron que reestimar algunas UT, algunas hacia arriba y otras hacia abajo. También se incluyeron UTs del *backlog* al *sprint* al reconsiderar la prioridad de estas.

En la gráfica 3 se observa como el cómputo general del trabajo pendiente iba decreciendo, aunque se variaba puntualmente al reestimar y añadir algunas UT.

## Sprint 2

En este *sprint* se añadieron funcionalidades más centradas en el usuario, así como funcionalidades complementarias al MVP anterior, consiguiendo una versión más completa de la aplicación donde se puede, además de lo disponible en el MVP anterior, crear una cuenta e iniciar sesión en ella, gestionar diferentes aspectos de los puntos de carga.

### Listado de UT:

- [GU03](#) Registrar usuario
- [GU02](#) Iniciar sesión.
- [GU05](#) Cerrar sesión.
- [GC02](#) Modificar cargador.
- [GU06](#) Añadir cargador a favoritos.
- [GU08](#) Ver cargadores favoritos.
- [GC04](#) Información adicional.
- Refactorización.
- [GU01](#) Nombre del usuario en el menú lateral.
- Rediseño visual.

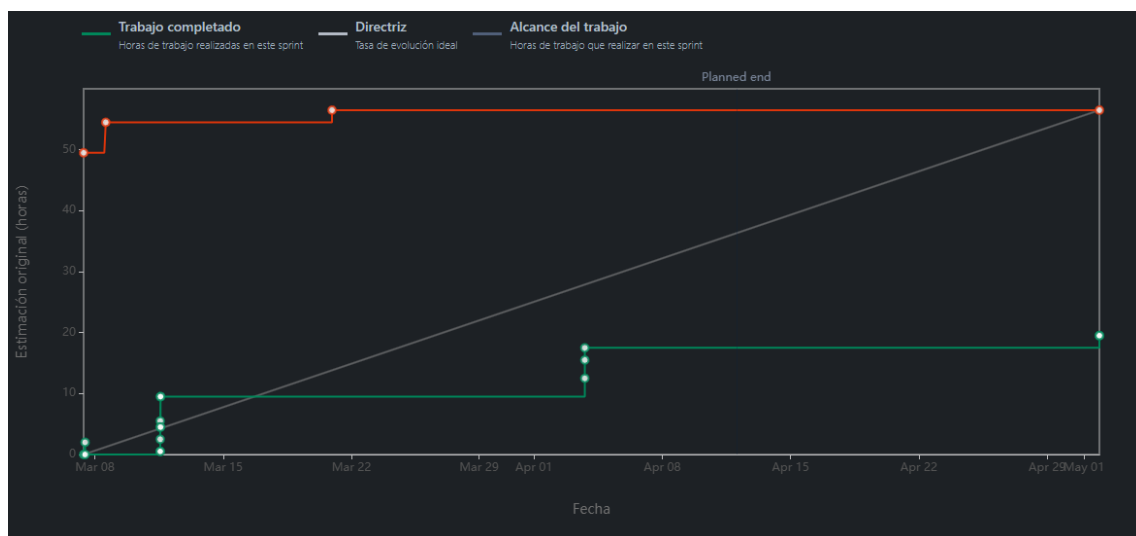


Figura 8: Gráfica trabajo completado Sprint 2.

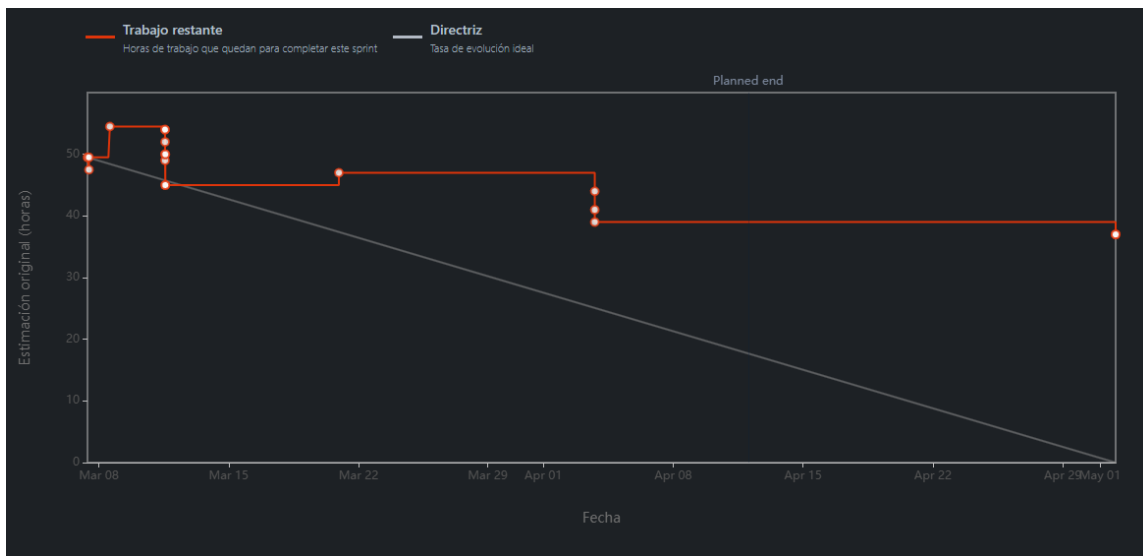


Figura 9: Gráfica trabajo pendiente Sprint 2.

Este *sprint* coincidió con el comienzo de mis prácticas externas por lo que tuve considerablemente menos tiempo para desarrollar el proyecto. También fueron vacaciones durante este *sprint* y no lo tuve en cuenta en la planificación por lo que este *sprint* estuvo parado un tiempo.

### Sprint 3

Este es el último *sprint* del proyecto. El resultado de este último *sprint* es el MVP presentado en este TFG. A pesar de no contar con todas las funcionalidades planteadas la aplicación cuenta con los requisitos imprescindibles para cumplir su misión.

#### Listado de UT

- [GU07](#) Eliminar cargadores de favoritos.
- [GU12](#) Historial de recargas.
- [GC09](#) Añadir fotos a cargadores.
- [GM05](#) Personalizar mapa.
- [GU09](#) Crear alarmas/recordatorios.
- Refactorización.
- Rediseño visual.
- [GM07](#) Filtrar cargadores.

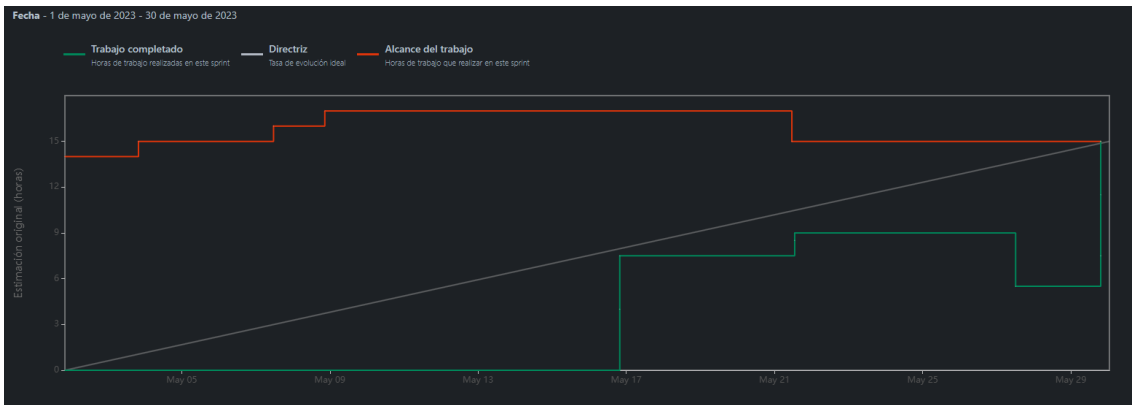


Figura 10: Gráfica trabajo completado Sprint 3.

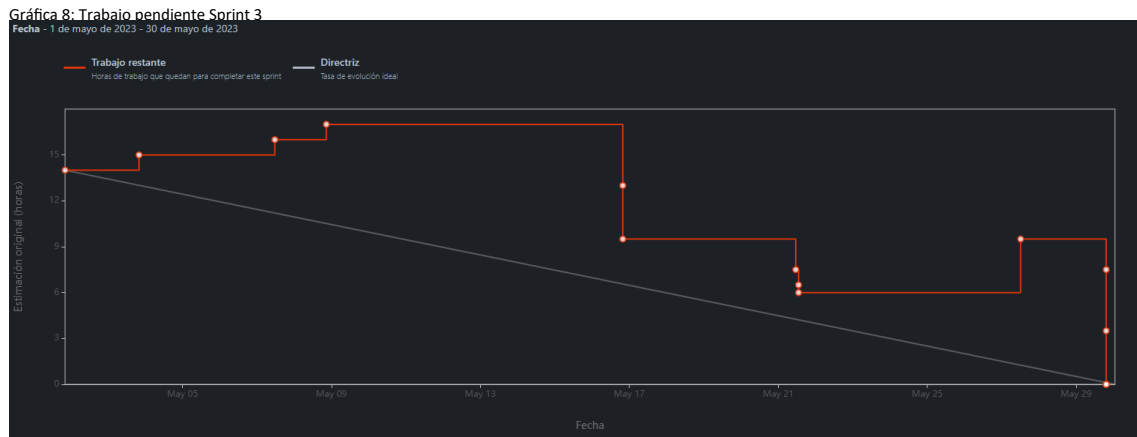


Figura 11: Gráfica trabajo pendiente Sprint 3.

La mitad del *sprint* coincidió con el periodo de descanso en el que paré el desarrollo del proyecto para descansar y centrarme en las prácticas. A mitad del *sprint* retomé el desarrolló de manera más intensiva, cumpliendo el objetivo final.

En este *sprint* se tuvieron que reestimar UT al alta, pero con poco margen. También se tuvo que pasar al *backlog* una UT que al final consideré prescindible.

## Backlog

Fuera del alcance de esta primera versión de la aplicación se quedan algunas funciones menos relevantes para el objetivo de la aplicación, pero que en caso de seguir desarrollando la aplicación serían las próximas en desarrollarse. Estas funcionalidades están orientadas a pequeños detalles que pueden ser útiles para el usuario.

### Listado de UT que han quedado fuera del alcance final

- [GM07](#) Filtrar puntos de carga por estado.
- [GC11](#) Conectar con la plataforma del punto de carga.
- [GC12](#) Pagar desde la aplicación.
- [GU14](#) Añadir vehículo al usuario.
- [GU19](#) Añadir fotos de los vehículos.
- [GU16](#) Visualizar perfil de usuario.
- [GU18](#) Añadir foto de perfil.
- [GU17](#) Modificar perfil de usuario.
- [GU13](#) Obtener estadísticas del vehículo.
- [GC10](#) Comentarios/chat/foro sobre los puntos de carga.
- [GU04](#) Cambiar la contraseña.
- [GC08](#) Marcar el punto de carga completo como averiado.

## Hoja de ruta

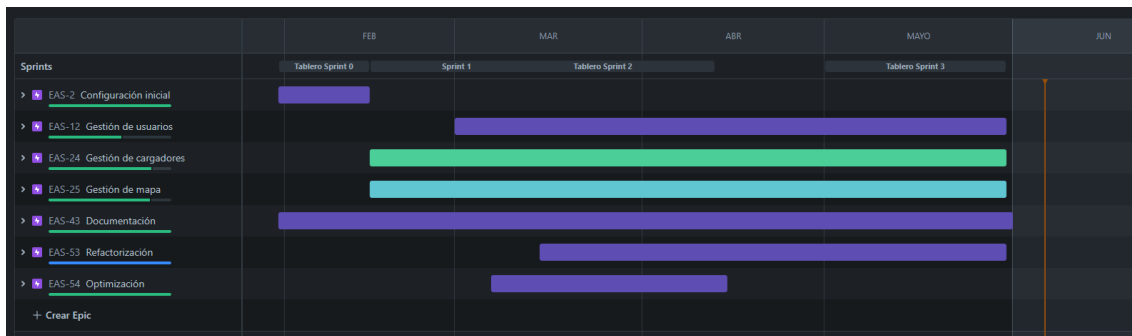


Figura 12: Hoja de ruta del proyecto.

En la figura 12 se muestra la hoja de ruta de como las diferentes épicas han sido abordadas a lo largo del proyecto. Se puede observar como la configuración inicial solo aparece durante el *sprint* 0, mientras que otras como la documentación está presente a lo largo de todo el proyecto. La gestión de mapas y cargadores también está presente en todo el proyecto desde el comienzo del *sprint* 1. La optimización está presente en el *sprint* 2 y la refactorización comienza en el *sprint* 2 hasta el final.

## 4 DISEÑO DE LA SOLUCIÓN

En este capítulo se detallará la arquitectura del sistema, explicando la arquitectura de la aplicación, la división de capas, los elementos internos y externos con los que interactúa el sistema. También se detallarán aspectos del desarrollo como las interfaces gráficas, las clases, los patrones de diseño implementados en el código, con algunos ejemplos del código de la aplicación. Por último, se profundizará en las herramientas y *frameworks* utilizados en el desarrollo de la aplicación.

### 4.1 Arquitectura del Sistema

Para la arquitectura de la aplicación, teniendo en cuenta que el objetivo es cubrir distintas plataformas, se ha optado por una arquitectura de 3 capas. Se han separado las capas de negocio, de persistencia y la presentación. De esta manera desacoplando al máximo la capa presentación de la lógica de negocio, podemos convertir la aplicación de una plataforma a otra solo creando la nueva capa de presentación. La capa de presentación solo contendría las llamadas a los métodos de la lógica y los métodos inherentes a la propia plataforma. Estos métodos por ejemplo podrían ser métodos que modifiquen elementos visuales o que utilicen elementos únicamente disponibles en dicha plataforma.

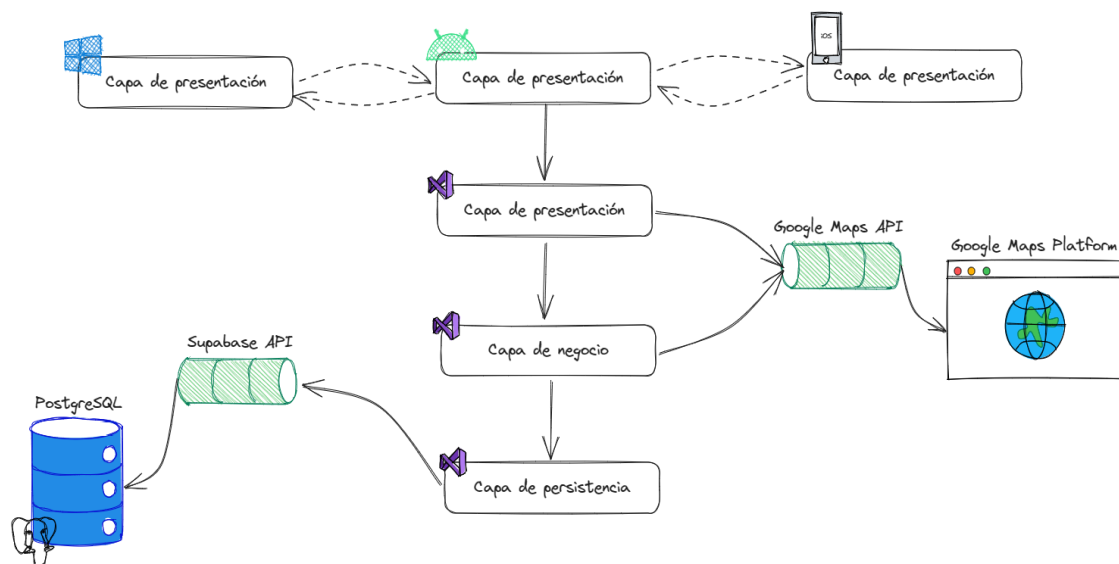


Figura 13: Diagrama de los componentes de la aplicación.

En la figura 13 se observa como las capas inferiores no conocen a sus superiores y solo conocen a su inferior inmediata. La capa de presentación solo conoce a la capa de negocio y la capa de negocio solo conoce la capa de persistencia. La capa de negocio no está totalmente desacoplada de la capa de presentación. A pesar de no conocer directamente a la capa de presentación, si recibe en sus métodos elementos concretos de la plataforma, como el mapa,

para realizar operaciones. No hay problema en esto ya que el mapa es un elemento común entre las plataformas, pero en caso de cambiar de mapa en la capa de presentación, también habría que hacerlo en la capa de negocio.

## 4.2 Diseño detallado

### Capa de presentación

Esta es la capa que contiene las vistas de la aplicación. Se ha intentado desacoplar al máximo posible la capa de presentación de la capa de negocio. Para la aplicación se ha escogido Android como punto de partida y solo se ha desarrollado la capa de presentación para esta plataforma. A futuro se plantea crear las capas de presentación para iOS y página *web*. Este planteamiento nos ofrece la gran ventaja de poder reutilizar la lógica prácticamente en su totalidad. Lo único que tendríamos que hacer es crear las nuevas interfaces adaptadas a las diferentes plataformas, crear la lógica inherente a la plataforma y reutilizar las llamadas a los métodos de la capa de negocio.

### Organización

- ❖ **Presentation.** Contiene todas las páginas y controles visuales de la aplicación.
  - **Controls.** Contiene los controles creados *ad-hoc*<sup>8</sup> para la aplicación.
    - **DayPickerControl.** Elemento que permite seleccionar uno o más días de la semana. En un principio iba a ser utilizado para seleccionar los días de una alarma/notificación, pero al final no se utilizó.
    - **SearchBarControl.** Elemento que permite la búsqueda de ubicaciones y el filtrado de puntos de carga del mapa.
    - **SideButtonsControl.** Panel de botones que contiene los botones laterales de la vista del punto de carga. Muestra la información adicional, lleva a la página de modificar la información del punto de carga y añade o elimina de la lista de cargadores favoritos.
  - **Pages.** Contiene las páginas principales de la aplicación. Todas las páginas de la aplicación contienen el menú desplegable lateral desde donde pueden navegar a la página principal, a crear un punto de carga nuevo, a la lista de favoritos, al historial de recargas y a cerrar o iniciar sesión en función de si el usuario ha iniciado sesión.
    - **MainPage.** Es la página principal de la aplicación, contiene el mapa y las funciones de buscar más cercano, cambiar el modo del mapa, navegar a las alarmas/notificaciones, buscar y filtrar en el mapa.
    - **ChargerPage.** Es la página donde se visualiza la información del punto de carga. Es desde donde el usuario puede visualizar el estado de los cargadores individuales del punto de carga y ocuparlos o desocuparlos. También los puede marcar cómo averiados desde esta página.

---

<sup>8</sup>Es una locución en latín que significa “para esto”.



- **CreateChargerPage.** Es la página desde donde se crean los puntos de carga manualmente. Está página contiene un mapa para seleccionar la ubicación del nuevo punto de carga y un formulario para rellenar la información del punto de carga. Se puede opcionalmente añadir información adicional y una imagen.
  - **EditChargerPage.** Es la página desde la cual se modifican los datos del punto de carga. Es muy similar a la página de crear un punto de carga, pero viene con los datos del formulario ya rellenados con los datos actuales del punto de carga que se desea modificar.
  - **LoginPage.** Es la página desde la cual se inicia sesión o se accede a la aplicación sin iniciar sesión.
  - **RegisterPage.** Es la página desde la cual creas una nueva cuenta de usuario.
  - **FavChargersPage.** Es la página donde se muestra el listado de puntos de carga favoritos del usuario. Si no has iniciado sesión la página te indica que para utilizar esta funcionalidad debes crear una cuenta o iniciar sesión.
  - **ChargerLogPage.** Es la página donde se muestra el historial de recargas del usuario. Si no has iniciado sesión la página te indica que para utilizar esta funcionalidad debes crear una cuenta o iniciar sesión.
  - **NotificationsPage.** Es la página donde se muestran las alarmas y notificaciones creadas por el usuario. Desde esta página se pueden crear nuevas alarmas y notificaciones.
  - **SplashPage.** Es la página de carga de la aplicación. Contiene una animación mientras la aplicación carga y valida si el usuario tiene la sesión iniciada.
- **Popups.** Contiene los Popups creados *ad-hoc* para la aplicación.
- **AcceptPopup.** Este Popup informa al usuario de que va a desocupar una plaza y le pregunta si la plaza estaba ocupada por él, para crear un histórico de la recarga.
  - **ChargerInfoPopup.** Este Popup muestra información adicional sobre el punto de carga en forma de lista, muestra los puertos de los que dispone el punto de carga, la dirección, el propietario, emplazamiento, velocidad de carga, precio, horario, tiempo límite y la información de contacto. Todos estos datos son opcionales y el punto de carga puede no disponer de ellos, en cuyo caso el campo se muestra vacío.
  - **CreateAlarmPopup.** En este Popup creamos las alarmas y notificaciones. Contiene un reloj para seleccionar la hora en la que sonará la notificación y también se puede añadir título, descripción y periodicidad.
  - **SelectPortsPopup.** En este Popup seleccionamos los puertos de los que dispone el punto de carga mientras lo estamos creando.
- **Views.** Contiene los elementos visuales incrustados en las páginas principales pero que no son controles.

- **ChargerLogView.** Es el elemento visual que representa al histórico de una recarga. Está contenido en un panel vertical dentro de la página de ChargerLogPage.
- **FavChargerView.** Es el elemento visual que representa a un punto de carga marcado como favorito. Está contenido en un panel vertical dentro de la página de FavChargersPage.
- **NotificationView.** Es el elemento visual que representa a una notificación o alarma. Está contenido en un panel vertical dentro de la página de NotificationsPage.
- **ChargerBottomSlideView.** Es el desplegable que aparece en la parte inferior de la MainPage al seleccionar un punto de carga en el mapa. Contiene la información imprescindible del punto de carga para que el usuario no necesite entrar en la página del punto de carga. Contiene la imagen, la dirección, la cantidad de cargadores individuales libres y ocupados y los puertos de los que dispone el punto de carga.
- **ChargerTypeView.** Es el elemento visual que muestra cuales son los puertos de los que dispone el punto de carga. Está contenido dentro del ChargerBottomSlideView y del ChargerInfoPopup.
- **SingleChargerSlideView.** Es el elemento incrustado dentro del ChargerPage que muestra la información de un cargador individual de un punto de carga. Muestra el estado del cargador y permite ocuparlo o desocuparlo, marcarlo como averiado y en el caso de que este ocupado por un tiempo fijo, muestra cuanto tiempo falta para que la persona que lo tiene ocupado lo libere. Para ver la información del resto de cargadores hay que deslizar el dedo en sentido horizontal.
- **SingleChargerStateView.** Es el elemento visual que muestra los iconos de los estados de los cargadores individuales. Está incrustado en el ChargerPage.

## Easy Charge

La MainPage implementa todos los requisitos de la época de gestión del mapa ([GM](#)). También implementa los requisitos [GU05](#) Cerrar sesión, parcialmente [GC03](#) Ver punto de carga y [GC04](#) Información adicional.



Figura 14: MainPage.

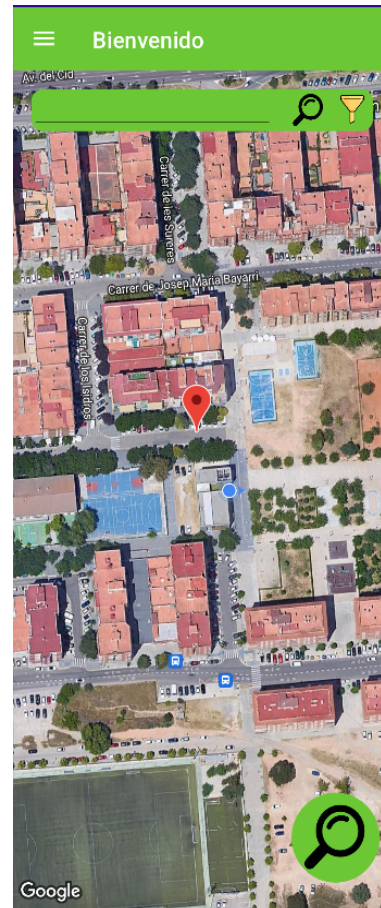


Figura 15: MainPage vista de satélite.



Figura 16: MainPage con menú lateral desplegado.

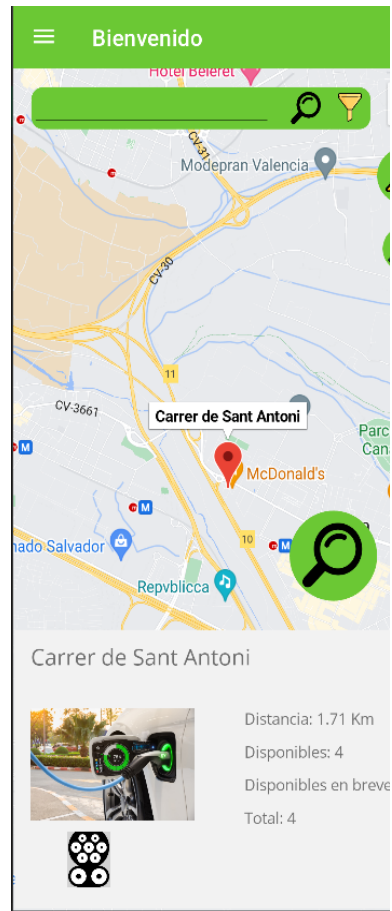


Figura 17: MainPage con punto de carga desplegado.

## Easy Charge

La ChargerPage implementa los requisitos de [GC03](#) Ver punto de carga, [GC04](#) Información adicional, [GC05](#) Ocupar cargador, [GC06](#) Desocupar cargador, [GC07](#) Marcar cargador como averiado, [GC08](#) Marcar punto de carga como averiado y a futuro en esta página se implementarán los requisitos [GC10](#) Comentarios/chat/foro, [GC11](#) Conectar con la plataforma del punto de carga y [GC12](#) Pagar desde la aplicación. También implementa las funcionalidades [GU06](#) Añadir cargadores a favoritos y [GU07](#) Eliminar cargadores favoritos.



Figura 18: ChargerPage.

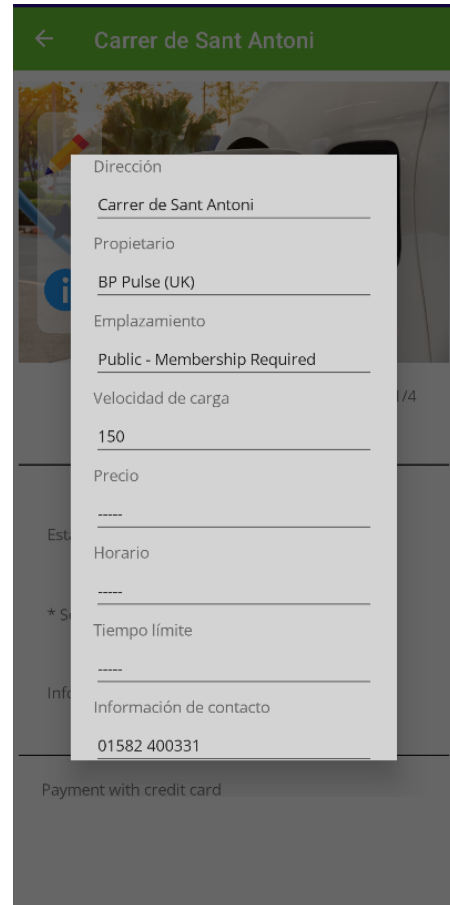


Figura 19: ChargerPage con información adicional desplegada.

## Easy Charge

La CreateChargerPage implementa los requisitos [GC01](#) Crear punto de carga y [GC09](#) Añadir foto del punto de carga.

La EditChargerPage implementa los requisitos [GC02](#) Modificar datos del punto de carga y [GC09](#) Añadir foto del punto de carga.



Figura 20: CreateChargerPage.

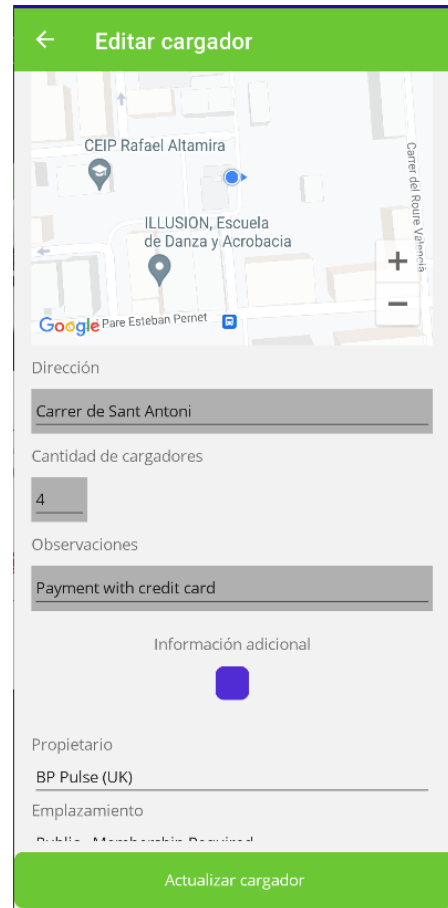


Figura 21: EditChargerPage.

## Easy Charge

La FavChargersPage implementa las funcionalidades [GU08](#) Ver cargadores favoritos.

La ChargeLogPage implementa las funcionalidades [GU12](#) Visualizar el historial de recargas.

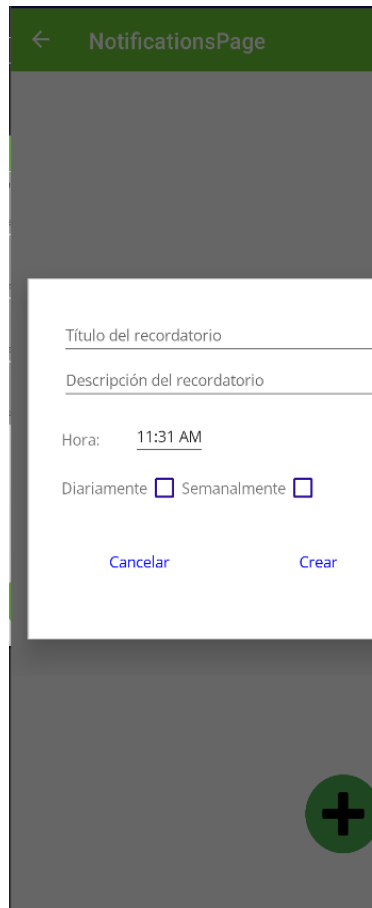


Figura 22: FavChargerPage.



Figura 23: ChargeLogPage.

La NotificationsPage implementa las funcionalidades de [GU09](#) Crear alarmas/recordatorios, [GU10](#) Visualizar alarmas/recordatorios y [GU11](#) Eliminar/pausar las alarmas/recordatorios.



*Figura 24:*  
*NotificationsPage.*



La LoginPage implementa la funcionalidad de [GU02](#) Iniciar sesión. La RegisterPage implementa la funcionalidad de [GU03](#) Registrar usuario.

The LoginPage interface features a green header with the text "Iniciar sesión". Below the header, there are two input fields: "Nombre de usuario" and "Contraseña". A checkbox labeled "Recordarme" is positioned below the password field. The main content area contains three green buttons: "Iniciar sesión", "Continuar sin iniciar sesión", and "Crear una cuenta". A link "He olvidado mi contraseña" is located between the "Iniciar sesión" and "Continuar sin iniciar sesión" buttons.

Figura 25: LoginPage.

The RegisterPage interface features a green header with a back arrow and the text "Crear cuenta". Below the header, there are three input fields: "Nombre de usuario", "Contraseña", and "Correo electrónico". A single green button labeled "Crear cuenta" is positioned at the bottom of the form.

Figura 26: RegisterPage.

## Easy Charge

Para los diseños de las interfaces se han diseñado una serie de *mock-ups* con un diseño básico pero que contiene los elementos principales de las interfaces y mantiene una línea de diseño común. Estos *mock-ups* fueron previos al desarrollo de las interfaces y han recibido actualizaciones, pero no están totalmente actualizados. Tampoco están todas las pantallas de la aplicación con su *mock-up*, ya que algunas han sido pensadas más adelante o no eran tan relevantes como para necesitar un *mock-up*.

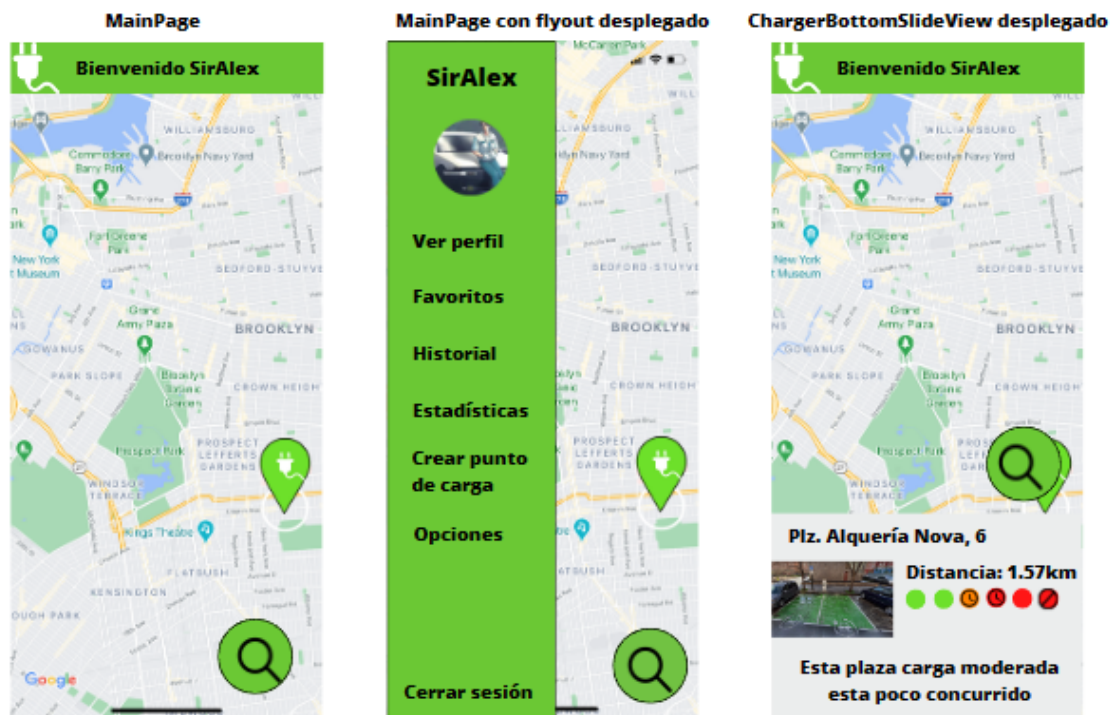


Figura 27: Mock-ups de la MainPage.

La ProfilePage implementará a futuro los requisitos [GU04](#) Cambiar la contraseña, [GU13](#) Ver estadísticas del coche, [GU14](#) Añadir coche, [GU15](#) Eliminar coche, [GU16](#) Visualizar perfil de usuario, [GU17](#) Editar perfil de usuario, [GU18](#) Añadir foto de perfil y [GU19](#) Añadir foto del vehículo.

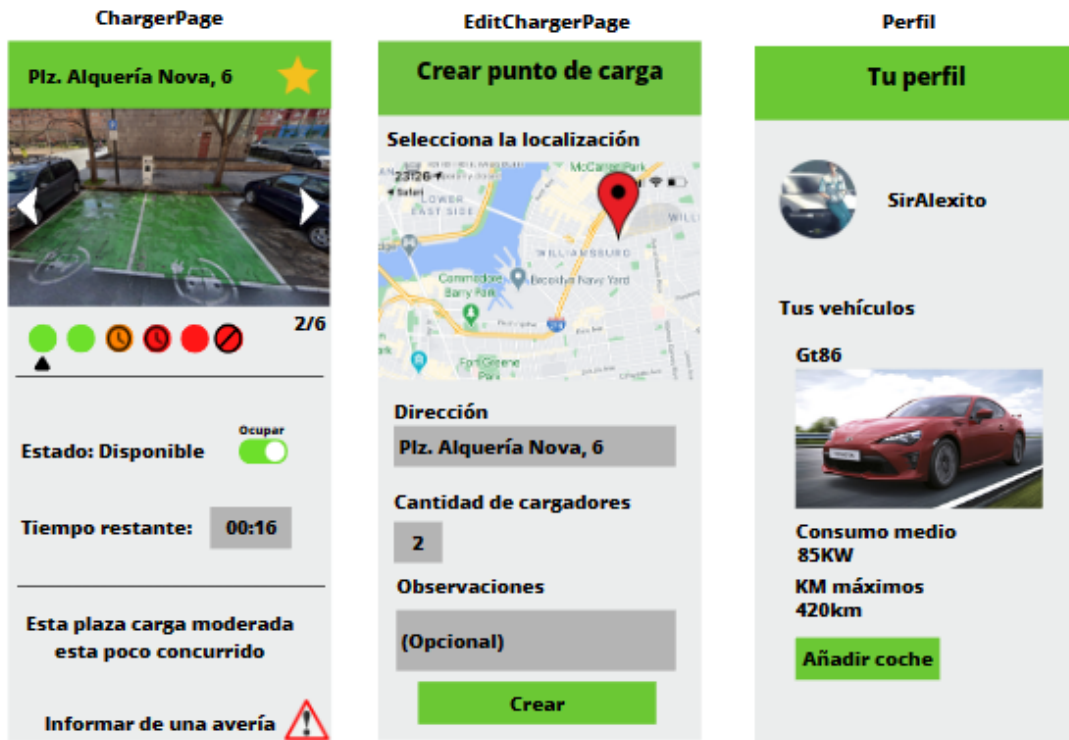


Figura 28: Mock-ups de la ChargerPage, CreateChargerPage y ProfilePage.

## Easy Charge

También se han diseñado *mock-ups* con diferentes tonalidades de color para diferentes temas, que es una mejora que se planea permitiendo al usuario elegir entre diferentes temas, claros y oscuros.

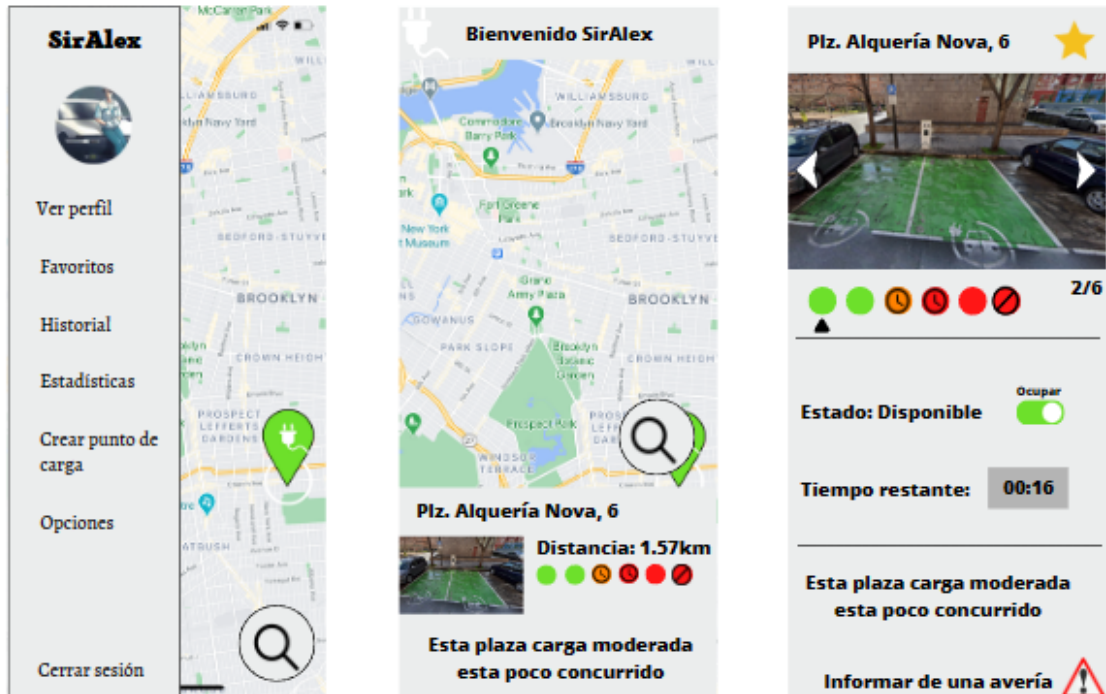


Figura 29: Mock-ups tema gris claro.

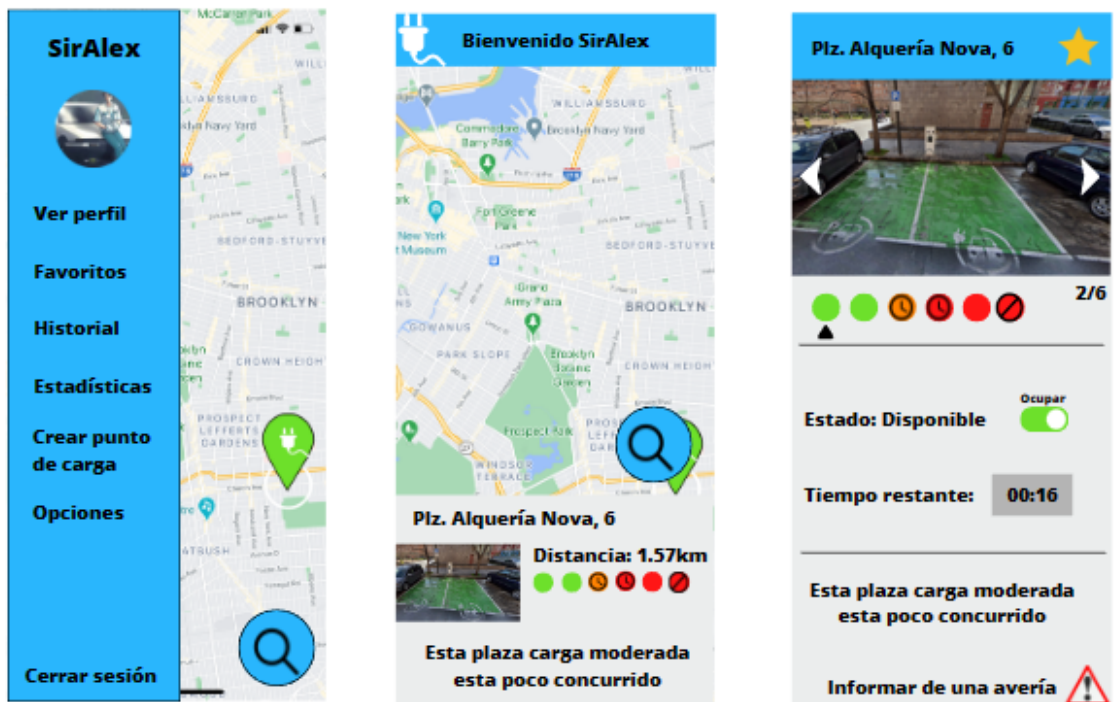


Figura 30: Mock-ups tema azul.

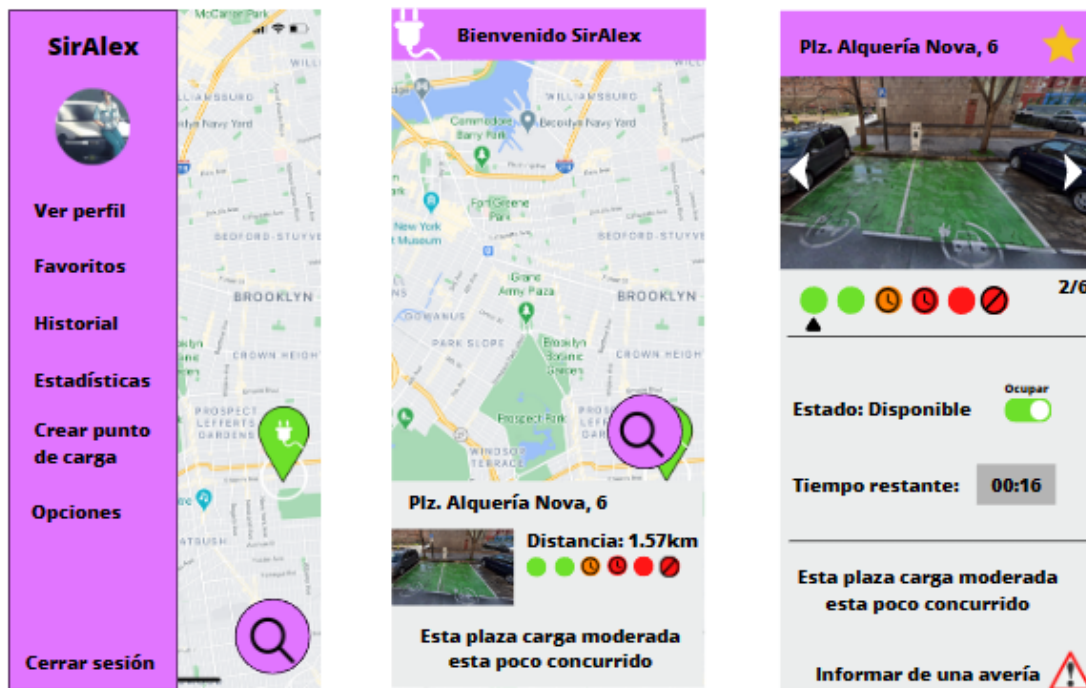


Figura 31: Mock-ups tema rosa.

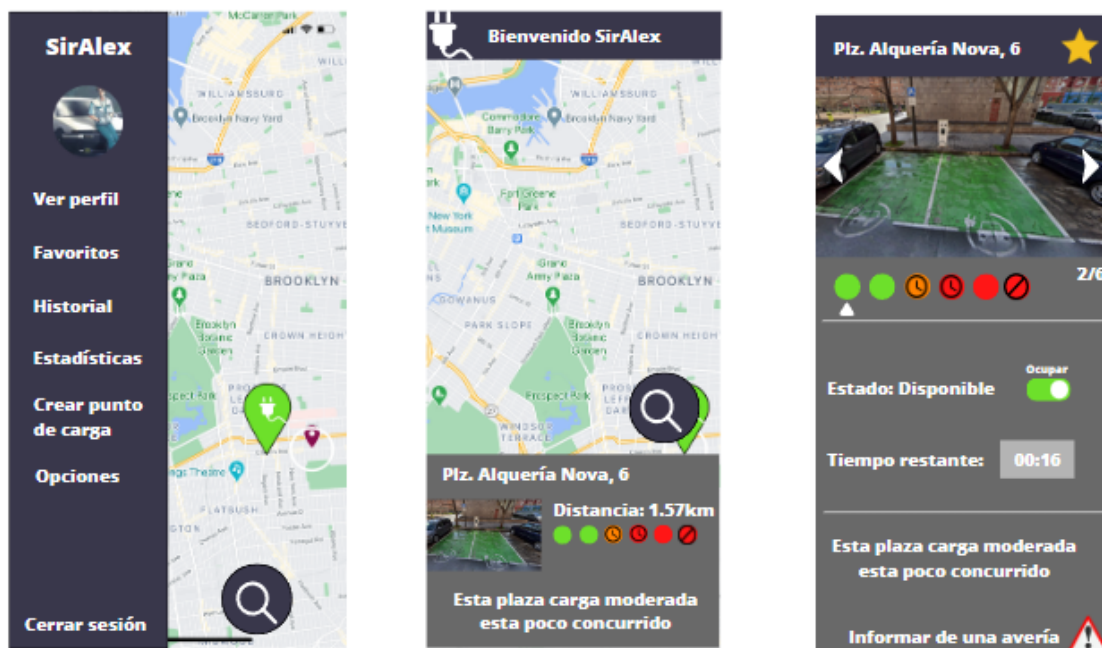


Figura 32: Mock-ups tema gris oscuro.

## Capa de negocio

Esta capa está prácticamente desacoplada de la capa de presentación, el único acoplamiento se encuentra en algunos métodos que reciben como parámetro el propio objeto del mapa, para realizar operaciones sobre él. Esto no es un problema puesto que el mapa es un objeto del propio *framework* y será compartido entre plataformas, por lo que mantenemos el objetivo de reutilización entre plataformas.

En esta capa se encuentran los objetos del dominio, las clases encargadas de representar los objetos y las operaciones asociadas a estos. Estas clases hacen de puente entre las interfaces de la capa de presentación y de los modelos de la persistencia. En estas clases se encuentran los conversores entre el dominio y los modelos.

En esta capa también se encuentran clases estáticas que realizan funciones que no se pueden incluir en una sola clase. Estas clases son estáticas y son llamadas desde la capa de presentación.

## Organización

- ❖ **Logic.** Contiene las clases del dominio y clases estáticas.
  - **Models.** Contiene las clases que representan los objetos del dominio y sus operaciones.
    - **Car.** La clase que representa al vehículo del usuario.
    - **User.** La clase que representa a los usuarios de la aplicación. Contiene la información del usuario y sirve para otorgar funcionalidades que los usuarios no registrados no pueden acceder.
    - **Charger.** La clase que representa los puntos de carga. Contiene toda la información de los puntos de carga y las operaciones que se pueden realizar sobre estos. Esta clase contiene 2 clases que se emplean dentro de esta:
      - **AdditionalInfo.** Esta clase representa la información adicional que se puede añadir a los puntos de carga. Esta clase luego se serializa a JSON para persistirla.
      - **Memento.** Esta clase representa la instantánea que se toma del punto de carga cuando se va a modificar, para mantener un historial de cambios y poder recuperar el estado anterior. Esta clase también se serializa a JSON para persistirse.
    - **SingleCharger.** La clase que representa los cargadores individuales y las operaciones que se pueden realizar sobre estos. En esta clase están los métodos que permiten ocupar, liberar, marcar como averiado un punto, etc.
    - **ChargeLog.** Esta clase representa el histórico de una recarga.
  - **Controllers.** Contiene las clases estáticas que sirven de librerías con funcionalidades de la lógica de negocio.
    - **ChargerController.** Contiene funcionalidades adicionales a la lógica de los puntos de carga.

- **MapController.** Contiene funcionalidades adicionales que operan sobre el mapa, como obtener la posición del usuario, la distancia, localizaciones, etc.
- **UserController.** Contiene funcionalidades adicionales a la lógica de los usuarios.
- **Singletons.** Contiene las clases estáticas que representan *singletons* de la aplicación.
  - **ActiveUser.** Representa la instancia del usuario que ha iniciado sesión. Esta instancia es única y compartida en toda la aplicación.
- **ResourceHelpers.** Contiene clases estáticas con funcionalidades más orientadas a dar formato y aspectos relacionados con la aplicación, pero menos lógicos.
  - **StringFormat.** Es una clase que da un formato concreto a los objetos de tipo TimeSpan.
  - **TimeHelper.** Sirve para obtener el tiempo restante de un objeto DateTime.
  - **ThemeChanger.** Contiene las funcionalidades para cambiar el tema visual de la aplicación.

### Diagrama de clases

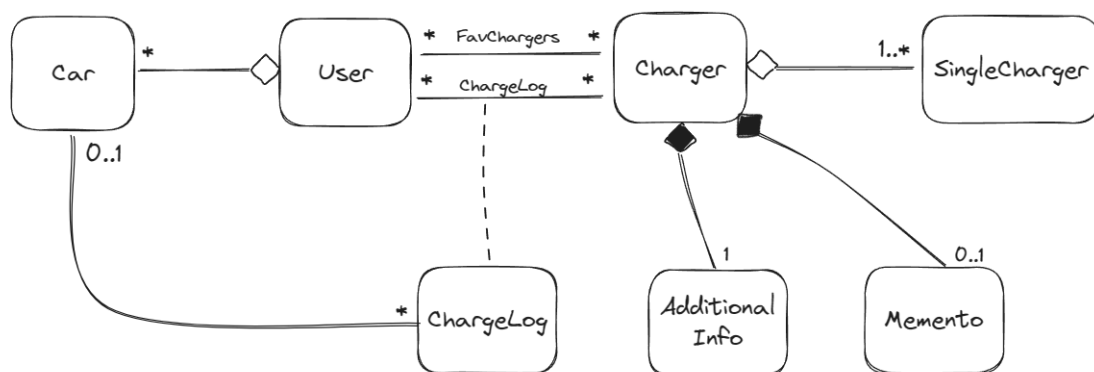


Figura 33: Diagrama de clases de negocio.

### Patrones

En el proyecto se han aplicado diferentes patrones de diseño, y se plantea el uso de otros a futuro para incluir nuevas funcionalidades. El uso de muchos patrones de diseño no ha sido realmente necesario en esta aplicación, ya que no contiene una lógica de negocio muy densa.

### Singleton

Este patrón es un patrón creacional que se basa en que solo exista una instancia de una misma clase, y sea compartida por toda la aplicación. Esto sirve para no crear instancias innecesarias de la misma clase, y así asegurarnos que las operaciones que se realizan sobre esta clase repercuten a todas las demás clases. En la aplicación se ha utilizado para en la clase

ActiveUser, clase que representa al usuario que ha iniciado sesión en la aplicación. Este objeto ActiveUser es compartido a lo largo de toda la aplicación, ya que se consulta en diferentes pantallas. La clase se encarga de gestionar el ciclo de vida del objeto *singleton*, que es del tipo User. Dispone de métodos para la creación del *singleton* (solo posible si no existía previamente uno), la obtención del objeto, la comprobación de si el objeto existe y el borrado del objeto.

[Código en el Anexo 1 – página 73.](#)

Para los cargadores del mapa, para no tener que tener varias listas, o volver a cargar todos los cargadores cada vez que se realiza una actualización, se ha implementado el patrón *singleton* otra vez para crear una instancia del mapa compartida a lo largo de las diferentes interfaces, ya que ahora mismo el mapa que hay en la pantalla principal y el mapa que hay en la pantalla de crear punto de carga no es el mismo objeto, por lo que no están siempre sincronizados de la misma manera, y para realizar dicha sincronización se realizan llamadas redundantes a la API de Google Maps y se hacen intercambios y llamadas a métodos también redundantes.

Este patrón funciona con 2 listas de cargadores, la lista principal donde están los cargadores que son mostrados en el mapa, y una lista que sirve de copia de la principal pero que no recibe los cambios de inmediato. Cuando un cargador es añadido, eliminado o modificado se realizan los cambios sobre la lista principal y cuando se vuelve a la MainPage donde está el mapa, se comprueban las diferencias entre la lista principal y la copia sin los cambios. Si existen diferencias entre las listas, se actualizan dichas diferencias en el mapa y después, a modo de *commit* de una transacción, se copia lista principal en la copia y vuelven a estar sincronizadas.

Esto permite que los cambios y actualizaciones en la lista sean muy ligeros ya que solo hay que volver a cargar los cargadores donde haya diferencias entre la lista principal y la secundaria.

[Código en el Anexo 1 – página 74.](#)

### **Memento**

Este patrón es un patrón de comportamiento cuya finalidad es crear una instantánea del estado de un objeto para más tarde recuperarlo. Esto es muy útil cuando tienes una funcionalidad para modificar un objeto, pero deseas guardar el estado anterior del objeto para en caso de querer restaurarlo, poder recuperar dicho estado anterior. El patrón funciona con una clase adicional que sirve de *Memento*, este objeto no necesariamente tiene todos los atributos del objeto original, si no que puede solo tener algunos objetos y otros son calculados por diferentes factores, o también puede tener atributos que el original no tiene, como la fecha de la creación del memento. Hay diferentes maneras de implementar el patrón memento:

Se puede almacenar en un atributo único, es decir la clase original contiene el objeto memento, que es reemplazado con cada versión del memento que se crea.

Otra posible implementación es mantener una pila de mementos. Esto quiere decir que los mementos se van apilando en una pila, de manera que el más reciente siempre está al principio de la pila, y estos se van llamando en orden inverso de llegada. Cada llamada saca al más reciente de la pila, con lo que obtenemos versiones cada vez más antiguas, similar a lo que obtenemos con el *control + z*.



Otra de las posibles implementaciones es mantener la pila de mementos, pero en un objeto aparte, encargado de gestionar el ciclo de vida de dichos mementos. El funcionamiento es idéntico al del caso anterior, pero tiene algunas implicaciones diferentes, como la posibilidad de mantener varios conjuntos de mementos diferentes por separado, extender la funcionalidad del memento para poder recuperar cambios deshechos (como *control + z* y *control + y*), y la posibilidad de serializarlo de manera diferente a la clase principal.

[Código en el Anexo 1 – página 76.](#)

### Mediador

El patrón mediador es un patrón de comportamiento cuya finalidad es facilitar la comunicación entre elementos independientes mediante el uso de un objeto mediador que es conocido por todos estos. Este patrón es de los más utilizados, aunque no siempre se es consciente de ello ya que es muy transparente. Este patrón está implementado en la mayoría de las interfaces modernas de manera transparente, ya que los elementos de la propia interfaz no se conocen entre ellos, pero utilizan el contenedor como mediador para consultarse información entre ellos. Esto no solo es útil a nivel de interfaces si no que gran parte de la tecnología moderna emplea este concepto.

### A futuro

A futuro para funcionalidades planteadas en el *backlog* se planea implementar más patrones.

Para el estado de los cargadores y los puntos de carga se planea implementar el patrón *state*. Este patrón consiste en almacenar el estado del objeto en un atributo y esto modifica el comportamiento del objeto original. Esto será muy útil para refactorizar la parte de ocupar cargadores y marcarlos como averiados y a la vez permitirá ampliar las funcionalidades de estos casos de uso.

Para notificar cambios en los cargadores cercanos a la ubicación del usuario se planea implementar el patrón *observer*. Este patrón de comportamiento define un mecanismo de suscripción ante eventos sobre un objeto. Esto será muy útil para que los cambios en los estados de los puntos de carga se actualicen de manera cuasi instantánea y no haya necesidad de refrescar el mapa cada cierto tiempo o de realizar consultas innecesarias a la API.

## Capa de persistencia

En esta capa se encuentran las clases que sirven de modelo para la persistencia y las consultas que se realizan a la base de datos. Esta capa no conoce a ninguna capa superior y solo se comunica con la API de Supabase. La idea de separar la capa de persistencia de la lógica de negocio reside en el hecho de que la base de datos va a cambiar. Supabase ofrece muchas ventajas, pero no deja de ser un *host* gratuito con muchas limitaciones, por lo que lo lógico, se continúe la aplicación o no, es plantear que la base de datos va a ser migrada en un momento u otro. Por eso se plantea la persistencia como una capa desacoplada y así será más fácil migrar la persistencia hacia otro *host*.

## Organización

### ❖ BackEnd

- **Connection.** Las conexiones en el *host* de Supabase son conexiones volátiles que por lo general se utilizan solo para una consulta, por lo tanto, se crean muchas consultas. Esta clase estática se encarga de devolver el objeto de la conexión.
- **Models.** Contiene las clases que representan los objetos que van a ser persistidos.
  - **CarModel.** Representa la clase Car de la manera que va a ser persistida.
  - **UserModel.** Representa la clase User de la manera que va a ser persistida.
  - **ChargerModel.** Representa la clase Charger de la manera que va a ser persistida.
  - **SingleChargerModel.** Representa la clase SingleChargerModel de la manera que va a ser persistida.
  - **ChargeLogModel.** Representa la clase ChargeLogModel de la manera que va a ser persistida.
  - **FavChargerModel.** Esta clase no tiene representación en el dominio. Esta clase representa la relación entre User y Charger. Luego en el dominio esta clase se transforma de manera que no necesita representación con una clase propia.
- **Querys.** Contiene las clases estáticas que se encargan de realizar las consultas a la base de datos.
  - **CarQuerys.** La clase que realiza las consultas a la base de datos relacionadas con la clase Car.
  - **UserQuerys.** La clase que realiza las consultas a la base de datos relacionadas con la clase User. También tiene consultas relacionadas con la clase FavChargerModel.
  - **ChargerQuerys.** La clase que realiza las consultas a la base de datos relacionadas con la clase Charger. También tiene consultas relacionadas con la clase FavChargerModel.
  - **SingleChargerQuerys.** La clase que realiza las consultas a la base de datos relacionadas con la clase SingleCharger.
  - **ChargeLogQuerys.** La clase que realiza las consultas a la base de datos relacionadas con la clase ChargeLog.

### Diagrama de clases

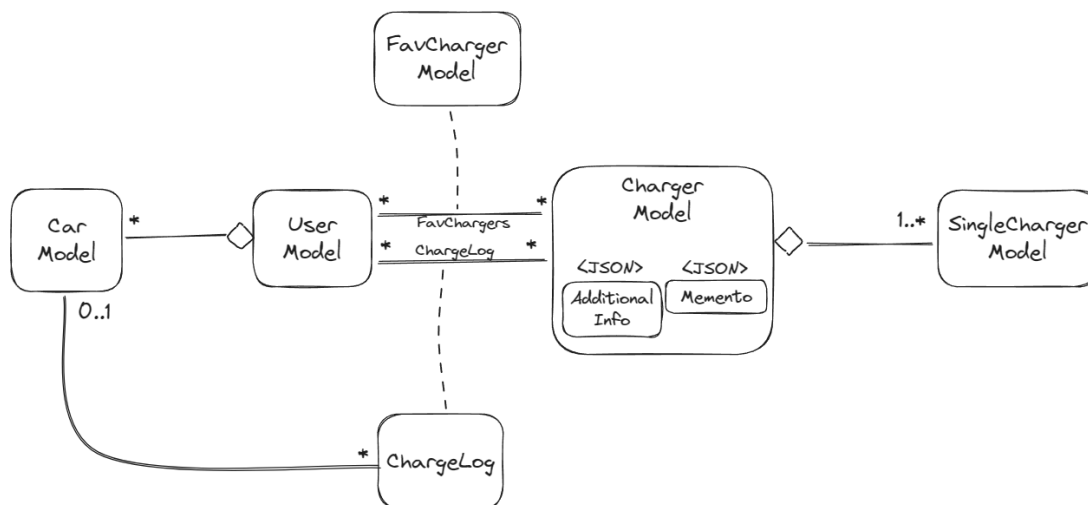


Figura 34: Diagrama de clases de persistencia.

## 4.3 Tecnología utilizada

### .NET MAUI

Como ya se ha mencionado a lo largo de la memoria, el *framework* en el que se basa la aplicación es .NET MAUI, un *framework* creado y mantenido por Microsoft.

MAUI, de las siglas *Multiplatform application user interface*, es un *framework* creado por Microsoft en el año 2022 con la intención de crear aplicaciones nativas con un marco común para dispositivos móviles y escritorio. .NET MAUI ofrece la posibilidad de desarrollar aplicaciones nativas para Android, iOS, macOS y Windows. Adicionalmente existe una integración con Blazor, una tecnología también propia de Microsoft para la creación de aplicaciones web de una sola página (SPA) muy similar a Angular, pero sin la necesidad de escribir en JavaScript y basado en C# al ser de .NET.

.NET MAUI ofrece la posibilidad de crear aplicaciones que comparten lógica entre plataformas, pero en el caso de las interfaces de usuario se tienen que crear diferentes ya que .NET MAUI utiliza las APIs concretas de cada plataforma para compilar los paquetes. En el caso de una interfaz definida en Android se utiliza la API de .NET for Android para compilar el código y así con cada plataforma.

El flujo de ejecución de .NET MAUI consiste en el código escrito en .NET que interactúa con la API de .NET MAUI, que consume directamente la API de la plataforma concreta. En el caso de Android la aplicación se compila directamente desde C# en un lenguaje intermedio que se recompila *Just-in-Time* en un ensamblado nativo cuando se inicia la aplicación.

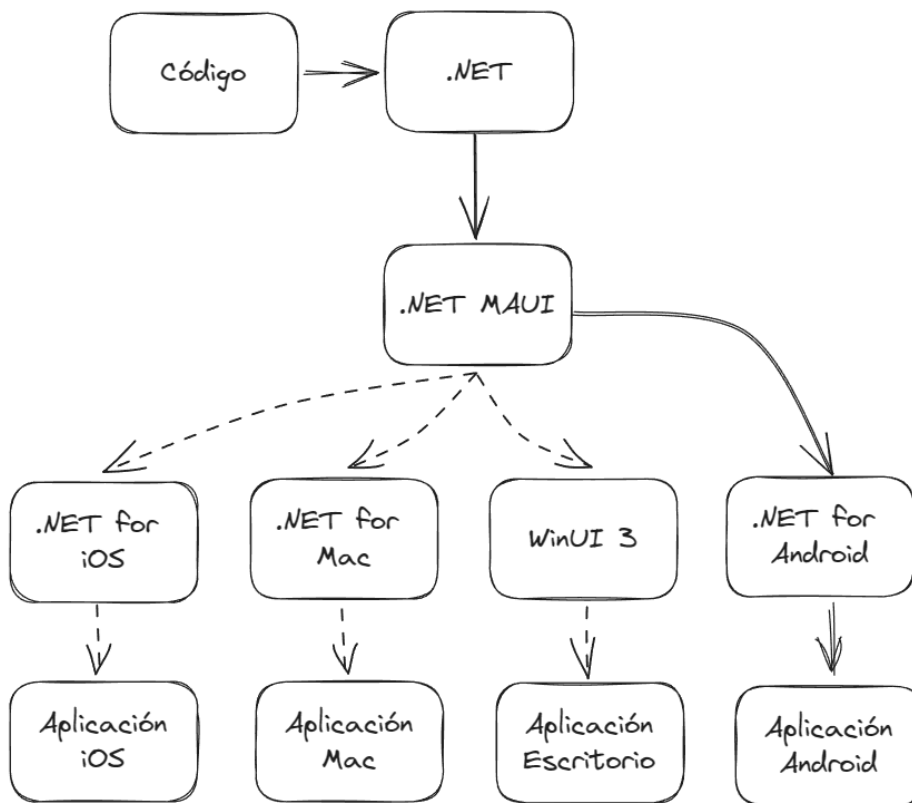


Figura 35: Diagrama de flujo del funcionamiento de .NET MAUI.

.NET MAUI ofrece:

- Un motor de diseño de páginas basado en XAML.
- Diferentes tipos de páginas (*content page, flyout page, navigation page, tabbed page, content view, popup*) para la creación de diferentes interfaces más flexibles.
- API multiplataforma con características nativas de las diferentes plataformas.
- Funcionalidades gráficas compartidas entre plataformas.
- Sistema de proyecto único compartido entre plataformas.
- Recarga activa de código fuente XAML.

## PostgreSQL y Supabase

PostgreSQL es un sistema de base de datos relacional de código abierto. Fue desarrollado en 1986 en la Universidad de Berkeley. A lo largo de los años PostgreSQL se ha consolidado como uno de los grandes SGDBs, recibiendo actualizaciones periódicas. PostgreSQL opera en la mayoría de los sistemas operativos, y cumple los criterios ACID. El lenguaje base de PostgreSQL es el SQL.

Las ventajas de PostgreSQL son:

- Datos tipados.
- Integridad de los datos mediante relaciones.
- Gran concurrencia y eficiencia.
- Alta fiabilidad.
- Seguridad y autenticación.
- Permite integraciones con otros módulos y extensiones.
- Búsqueda por texto.

Supabase es un *host* de bases de datos que permite crear y alojar bases de datos remotas en PostgreSQL. Supabase es la alternativa de código abierto a Firebase, el servicio de *host* de bases de datos de Google. Cuenta con una serie de librerías para implementar y realizar las llamadas cliente desde tu proyecto de una manera mucho más sencilla. Estas librerías soportan JavaScript, Flutter, Python, C#, Swift y Kotlin. Además, se puede acceder la base de datos mediante la API de Supabase.

Algunas de las características de Supabase son:

- Simplicidad de uso.
- Funcionalidades en tiempo real.
- Acceso mediante API REST.
- Autenticación de roles y seguridad.
- Almacenamiento de información en *buckets*.
- Vectores para la indexación y búsqueda de datos.
- Integración de módulos y librerías de código abierto.
- Funciones en PL/pgSQL y TypeScript.

## Google Maps API

La API de Google Maps Platform ofrece un gran abanico de APIs y diferentes funcionalidades. Una de las grandes ventajas es que con un volumen reducido de peticiones el servicio es gratuito e incrementa en función del uso. La principal funcionalidad de este conjunto de APIs es el mapa que ofrece, en el que se puede navegar como en la aplicación de Google Maps y adicionalmente se pueden marcar puntos del mapa, funcionalidad clave para nuestra aplicación. La API del mapa varía en función de la plataforma desde la cual se realiza la llamada, ya que tiene características ligadas a dichas plataformas.

La API más consumida de nuestra aplicación es la API de Maps SDK for Android, que devuelve un mapa para la plataforma de Android actualizado en tiempo real. Otra de las APIs más consumidas por nuestra aplicación es Geolocation API, que devuelve la dirección y otros datos de una localización concreta.

Algunas de las APIs más relevantes de Google Maps Platform son:

- Directions API, que devuelve las direcciones hacia una localización.
- Distance Matrix API, que devuelve tiempo y distancia para una localización.
- Geocoding API, que convierte direcciones en coordenadas y viceversa.
- Places API, que devuelve información sobre lugares.
- Street View Static API, que permite acceder a las funciones de Street View.

Google Maps Platform permite observar métricas en tiempo real sobre las consultas que se realizan a cada API, así como filtrarlas para obtener diferentes estadísticas de uso.

Otra de las funciones de la plataforma es crear mapas personalizados, donde se muestra cierto tipo de información y permite crear estilos para definir la apariencia visual del mapa.

Adicionalmente existe una biblioteca de soluciones ya creadas en la propia plataforma para integrar en tus aplicaciones o utilizar de ejemplo.

## 5 Desarrollo de la solución

En este capítulo se hablará del proceso de desarrollo del proyecto. También se hablará de algunos problemas que han surgido en este proceso.

### 5.1 Problemas

El desarrollo de la aplicación ha sido bastante común, con la única diferencia de que el proyecto ha sido llevado a cabo por un único integrante, lo que conlleva la ventaja de que las tomas de decisiones siempre son unilaterales y la comunicación entre integrantes del equipo es instantánea, pero también conlleva una serie de desventajas que sobrepasan a las ventajas. Uno de dichos problemas encontrados en el desarrollo es la única visión crítica sobre la aplicación. Al ser la misma persona la que define la aplicación, la programa y después la evalúa no siempre se es todo lo crítico que se podría ser.

Otro de los problemas que han surgido en el desarrollo es la falta de tiempo. Al ser un único integrante el encargado de todo el desarrollo de la aplicación, y además con el tiempo reducido. Esto ha repercutido en las funcionalidades incluidas en esta primera versión, ya que inicialmente se plantearon más funcionalidades que han tenido que ser relegadas al backlog para la siguiente versión de la aplicación.

Independientes de los problemas por causas internas al proceso de desarrollo, han surgido otros problemas relacionados con factores externos.

Uno de los mayores defectos de .NET MAUI es la escasa documentación no oficial. Microsoft ofrece una gran documentación oficial, que muchas veces resuelve las dudas con suficiente eficacia, pero dicha documentación oficial en muchas ocasiones no contempla ciertas casuísticas que van surgiendo en el desarrollo de la aplicación. Esto ha supuesto un problema sobre todo a la hora de gestionar los hilos de la aplicación, más concretamente cuando un hilo diferente al hilo principal, encargado de la interfaz gráfica, intenta modificar algún elemento visual. Existe muy poca documentación sobre como trabajar de manera paralela con elementos visuales. El *workaround* planteado para solventar este problema ha sido lanzar un hilo paralelo que se encarga del cálculo de la información necesaria y dicho hilo de manera asíncrona y dicho hilo devuelve el valor de la información en una variable que es usada en el hilo principal y entonces actualizado en la interfaz gráfica.

```
private async void setDistance(){  
    Distance.Text = "Distancia: Calculando";  
  
    var task = await Task.Run(() =>  
        MapController.DistanceFromUser(Charger).Result.ToString("0.##"));  
  
    Distance.Text = "Distancia: " + task + " Km";  
}
```

Otro de los problemas externos que han surgido en el desarrollo, manifestado también por la temprana edad del *framework*, han sido algunos errores del propio *framework*. Como en todos los *frameworks*, surgen errores que deben ser arreglados con parches por el equipo de soporte de la tecnología y no hay más remedio que sortearlos de alguna manera hasta que se resuelvan dichos errores. Uno de esos errores se encuentra en el atributo *Text* de un objeto *Entry* (un cuadro de texto que admite *input*), que muestra visualmente el texto del atributo, pero cuando intentas acceder al atributo y leer el valor, este se muestra como *Null*. Esto pare ser un error que ya apareció en 2021 y supuestamente fue solventado en el parche #3054, pero por algún motivo interno del *framework*, el error vuelve a aparecer. La solución ha sido simple ya que visualmente el texto si se muestra. El valor del texto que se desea almacenar para luego consultarlo se guarda en una variable privada y luego se lee.

Otro ejemplo de error externo, aunque esta vez no es del *framework* si no de una biblioteca externa, es que las notificaciones una vez añadidas al centro de notificaciones del dispositivo, puedes realizar la operación de borrarlas, y efectivamente se borran del objeto que representa el centro de notificaciones, pero, aunque desaparezcan de dicho objeto, las notificaciones pendientes siguen presentes en el sistema y suenan en el momento que se indicó al crearlas. A este problema no se le ha encontrado una solución y está pendiente de que el equipo de soporte de la librería de notificaciones resuelva el error.

## 5.2 Dificultades

Más que un problema, ya que esto no es ningún error ni nada que haya tenido que ser esquivado, es la dificultad de escribir código desacoplado. Las ventajas de tener una arquitectura de capas en las que las capas están lo más desacopladas posibles son muchas, pero esto conlleva una mayor complejidad de desarrollo. Esta dificultad nace de saber trazar la línea de lo que es lógica inherente a la interfaz gráfica o es lógica de negocio. También trasladar toda la lógica de las interfaces que sea común entre las plataformas a la capa de negocio común. Ejemplos del desacoplamiento:

```
public partial class MainPage : ContentPage {  
    ...  
    private async void FindClicked(object sender, EventArgs e) {  
        var charger = await Task.Run(() => MapController.FindNearest(chargers));  
        MapController.MoveMapTo(charger.getLocation(), map);  
        OpenSlide(charger);  
    }  
    ...  
}
```



En este ejemplo se puede observar como la tarea de encontrar el punto de carga más cercano es delegada en la clase estática MapController, una clase estática que contiene las funciones asociadas a la lógica de negocio del mapa. Un ejemplo de funciones de la interfaz gráfica trasladada a la capa lógica por ser común entre plataformas es el método MoveMapTo, un método que centra el mapa sobre la ubicación del usuario. Esta función está incluida en la clase MapController ya que el mapa es un elemento compartido entre plataformas.

```
public partial class MainPage : ContentPage {  
  
    ...  
  
    private void OpenSlide(Charger charger){  
        selectedCharger = charger;  
        if (SelectedChargerLayout.Count > 0)  
            SelectedChargerLayout.RemoveAt(0);  
            SelectedChargerLayout.Add(new ChargerBottomSlideView(charger));  
            SelectedChargerLayout.FadeTo(1, 250);  
            FindButton.TranslateTo(0, -250, 250, default);  
        }  
    ...  
  
}
```

En cambio, en este ejemplo de código extraído de la interfaz de MainPage se puede observar como la lógica de este método solo opera sobre elementos de la interfaz gráfica (SelectedChargerLayout es el contenedor donde se añade el punto de carga seleccionado en el mapa de la MainPage).

Otro de los problemas encontrados ha sido el prolongado tiempo de respuesta de la función de filtrado de puntos de carga. Al realizar la carga masiva de datos, el tiempo de esta función se ha disparado. Para resolver el prolongado tiempo de respuesta de la función de filtrado de puntos de carga, se ha recurrido a la paralelización del método que itera sobre la lista de cargadores y realiza el filtrado en función de la distancia y los cargadores seleccionados.

[Código en el Anexo 1 – página 78.](#)

## 5.3 Puntos destacables

En la clase *AppShell*, que es la página que contiene toda la aplicación, se ha tenido que añadir un fragmento de código para gestionar el texto de encabezado del menú lateral, que es un elemento que no se encuentra en ninguna página si no que es un elemento interno de esta *AppShell*. Este fragmento de código cambia el texto del encabezado por el nombre del usuario en caso de que este haya iniciado sesión y el nombre del botón que te lleva a la página de inicio de sesión, “Iniciar sesión” en caso de que el usuario no haya iniciado sesión y “Cerrar sesión” en caso de que sí. Lo destacable de esto es que esta clase no está diseñada para tener código propio ya que su única función es gestionar el ciclo de vida de la aplicación y contener las pantallas y el menú lateral de la aplicación.

```
public partial class AppShell : Shell
{
    public AppShell()
    {
        InitializeComponent();
    }

    public void ChangeText(bool logged)
    {
        LogText.Text = logged ? "Cerrar sesión" : "Iniciar sesión";
        UserNameText.Text = logged ? ActiveUser.getActiveUser().UserName : "EasyCharge";
    }

    private void LogTapped(object sender, TappedEventArgs e)
    {
        Shell.Current.GoToAsync("//LoginPage");
    }
}
```

## 6 Implantación

En este capítulo se hablará de la carga masiva de datos llevada a cabo previa al despliegue de la aplicación y del despliegue e instalación en un dispositivo real.

### 6.1 Carga de datos

Para la carga masiva de los datos se ha utilizado la API de [Open Charge Map<sup>9</sup>](https://openchargemap.org/), que ofrece una gran cantidad de puntos de carga y su información de manera gratuita. Estos datos pueden ser obtenidos mediante una petición a su API, a la que hay que registrarse en la aplicación para poder realizar dichas peticiones.

Mediante la consulta a la API se obtiene un archivo JSON con la lista de objetos que representan los puntos de carga. Mediante un deserializador de JSON se obtienen unos objetos representados por modelos que conforman el objeto JSON deserializado, que mediante unos mappings se transforma en el modelo que acepta nuestra base de datos.

#### Estructura del modelo JSON y *mappings* a la clase `ChargerModel`

- ❖ **ItemModel**
  - **OperatorInfo**
    - **Title** -> AdditionalInfo.Owner
    - **PhonePrimaryContact** -> ChargerModel.AdditionalInfo.ContactInfo
  - **UsageType**
    - **Title** -> ChargerModel.AdditionalInfo.Place
  - **StatusType**
    - **IsOperational** -> ChargerModel.Broken
  - **NumberOfPoints** -> ChargerModelQuantity
  - **GeneralComments** -> Description
  - **Connections**
    - **PowerKW** -> ChargerModel.AdditionalInfo.ChargingSpeed
    - **ConnectionType**
      - **Title** -> ChargerModel.Ports

El proceso de mapeado contiene operaciones más complejas, como la obtención de los puertos disponibles para el punto de carga a través de los puertos de cada cargador individual. O la creación del número correcto de cargadores individuales para cada punto de carga.

La carga de datos no asegura que el 100% de los puntos de carga estén disponible, lo más probable es que puntos de carga con pocos cargadores no estén registrados en la API de [Open Charge Map](https://openchargemap.org/), pero los usuarios disponen de la opción de agregar los puntos de carga que no existan en la aplicación, y así la base de datos crecerá en volumen y será cada vez más exacta.

<sup>9</sup><https://openchargemap.org/site>

## 6.2 Despliegue

El proceso para desplegar la aplicación en otros dispositivos es sencillo. Desde el Visual Studio, el IDE utilizado, se puede publicar la aplicación en la Microsoft Store o de manera local mediante un archivo .apk o .aab. Para Android lo necesario es el archivo .apk, que permite instalar la aplicación en el dispositivo. Para crear dicho archivo y publicar la aplicación hay que crear un certificado digital de la *truststore*<sup>10</sup> de Visual Studio, dónde se indica el nombre u organización que ha desarrollado la aplicación, datos de la localización de origen y una contraseña.

Una vez el archivo está en el dispositivo Android, se puede instalar igual que cualquier otra apk. El certificado, al no estar registrado y validado, el dispositivo nos avisa de que el autor de la aplicación no es conocido, pero para este caso se puede ignorar e instalar de todas formas.

Cuando se abre la aplicación, comienza en la pantalla de inicio de sesión, donde nos podemos crear una cuenta nueva. Independientemente de si creamos una cuenta o accedemos sin cuenta, al acceder al mapa por primera vez te pide los permisos necesarios para acceder a tu ubicación. A partir de ese punto la aplicación ya es totalmente funcional en el dispositivo.

---

<sup>10</sup>La *truststore* de Visual Studio es un contenedor que almacena los certificados digitales emitidos por autoridades certificadoras confiables para Visual Studio.

## 7 Pruebas

Las pruebas han consistido sobretodo en validar el funcionamiento de la aplicación desde el punto de vista del cliente del proyecto. Se han definido historias de usuario para comprobar el correcto funcionamiento de las funcionalidades implementadas. No todas las funcionalidades han sido probadas ya que al validar ciertas funcionalidades validas ya otras funcionalidades como por ejemplo al buscar el punto de carga más cercano también validas que los puntos son visualizados sobre el mapa. Para la funcionalidad de ocupar cargadores se ha desarrollado un test para validar el correcto funcionamiento del estado del cargador en función de los datos introducidos.

[Código en el Anexo 1 – página 81.](#)

<b>Objetivo</b>	Buscar el punto de carga más cercano
<b>Pasos</b>	<ol style="list-style-type: none"> <li>1. Entrar a la página principal.</li> <li>2. Presionar el botón de buscar el más cercano (la lupa).</li> </ol>
<b>Resultado</b>	Se muestra en el control inferior de la pantalla la información del punto más cercano y el mapa centra su ubicación en él.
<b>Ejecución correcta</b>	Si

*Tabla 9: Pruebas búsqueda punto de carga.*

<b>Objetivo</b>	Crear un punto de carga
<b>Pasos</b>	<ol style="list-style-type: none"> <li>1. Entrar a crear punto de carga.</li> <li>2. Seleccionas la ubicación del nuevo punto de carga.</li> <li>3. Introduces los datos.</li> <li>4. (Opcional) Introduces los datos adicionales.</li> <li>5. Presionas el botón de aceptar y aceptas en el diálogo que aparece.</li> </ol>
<b>Resultado</b>	Aparece un diálogo que nos confirma que el punto de carga ha sido creado.
<b>Ejecución correcta</b>	Si

*Tabla 10: Pruebas creación punto de carga.*

<b>Objetivo</b>	Editar información de un punto de carga.
<b>Pasos</b>	<ol style="list-style-type: none"> <li>1. Accedes a un punto de carga.</li> <li>2. Presionas el botón de editar (el lápiz).</li> <li>3. Modificas la información de los campos.</li> <li>4. Presionas el botón de aceptar y aceptas en el diálogo que aparece.</li> </ol>
<b>Resultado</b>	Aparece un diálogo que nos confirma que el punto de carga ha sido modificado.
<b>Ejecución correcta</b>	Si

*Tabla 11: Pruebas editar información punto de carga.*

<b>Objetivo</b>	Ocupar punto de carga.
<b>Pasos</b>	<ol style="list-style-type: none"> <li>1. Accedes a un punto de carga.</li> <li>2. (Opcional) Le introduces duración.</li> <li>3. Presionas el <i>switch</i> de ocupar cargador.</li> <li>4. Aceptas en el diálogo que aparece.</li> </ol>
<b>Resultado</b>	El cargador se muestra como ocupado. En el caso de tener duración se indica el tiempo restante para el que cargador quede libre.
<b>Ejecución correcta</b>	Si

*Tabla 12: Pruebas ocupar punto de carga.*

<b>Objetivo</b>	Filtrar punto de carga
<b>Pasos</b>	<ol style="list-style-type: none"> <li>1. Accedes a la página principal.</li> <li>2. Despliegas los filtros.</li> <li>3. Introduces los filtros y aceptas.</li> </ol>
<b>Resultado</b>	Solo se muestran los puntos de carga que cumplan los criterios de búsqueda.
<b>Ejecución correcta</b>	Parcialmente. El filtro funciona pero el tiempo de respuesta es demasiado largo.

*Tabla 13: Pruebas filtrar puntos de carga.*

<b>Objetivo</b>	Añadir punto de carga a favoritos.
<b>Pasos</b>	<ol style="list-style-type: none"> <li>1. Accedes a un punto de carga.</li> <li>2. Presionas el botón de añadir a favoritos (la estrella).</li> </ol>
<b>Resultado</b>	La estrella se muestra coloreada y al acceder a la página de cargadores favoritos te aparece el punto de carga.
<b>Ejecución correcta</b>	Si

*Tabla 14: Pruebas añadir puntos de carga.*

<b>Objetivo</b>	Crear notificación.
<b>Pasos</b>	<ol style="list-style-type: none"> <li>1. Accedes a la página de notificaciones.</li> <li>2. Presionas el botón de añadir (el signo de suma).</li> <li>3. Introduces los datos.</li> <li>4. Seleccionas la hora y la periodicidad y aceptas</li> </ol>
<b>Resultado</b>	La notificación se muestra en la página de notificaciones y cuando llega la hora se muestra la notificación en el dispositivo.
<b>Ejecución correcta</b>	Si

*Tabla 15: Pruebas creación de notificaciones.*

## 8 Conclusiones

Son muchas las conclusiones que se pueden obtener de realizar un proyecto de desarrollo de una aplicación a lo largo de 4 meses. Las iré mencionando por orden cronológico.

La primera conclusión a la que llegué, que ya lo sabía antes de comenzar el proyecto, es en la importancia de plantear un plan de elicitación de requisitos. No tiene que ser un plan detallado con fechas, métodos e instrucciones, pero un plan general sobre cómo se van a elicitar los requisitos, con diferentes metodologías y ejercicios para ello viene muy bien. Luego aplicar dichos ejercicios te da una amplia visión de lo que podría ser la aplicación. No hay que centrarse solo en tu punto de vista, también hay que valorar las ideas de los demás y esto fomenta que la aplicación sea más completa, con funcionalidades que uno puede considerar prescindibles pero que en realidad para otros usuarios puede ser una funcionalidad diferenciadora. El uso de estrategias como la lluvia de ideas, o las personas o historias de usuarios son métodos muy útiles a la hora de extraer requisitos, ya que conoces otros puntos de vista y haces el ejercicio de ponerte en el lugar de diferentes posibles de usuarios para identificar sus diferentes necesidades.

La siguiente conclusión a la que llegué, que también sabía desde que cursé la asignatura de Proceso de software, es la importancia de llevar una organización de las tareas y un seguimiento. Como en la conclusión anterior, no es necesario llevar al máximo el nivel de detalle, pero tener una lista de las tareas, con un orden de prioridad y una estimación del esfuerzo requerido hace que el desarrollo del trabajo sea mucho más ordenado y por ende sea mucho más eficaz. El uso de herramientas como tableros *Kanban* para esta función es muy útil y a diferencia de otros planteamientos, emplear un *Kanban* es muy sencillo y no consume prácticamente tiempo. Desde luego las ventajas sobrepasan las desventajas.

Siguiendo la línea de la anterior conclusión, llevar una organización del tiempo es fundamental. No existe un plan perfecto sobre cómo organizar tu tiempo para las diferentes situaciones que se dan en el desarrollo de un proyecto, pero existen diferentes planes que son perfectamente válidos y hay que encontrar uno que funcione para tu situación. En mi caso al estar trabajando jornada completa, realizar varias horas de trabajo diario era complicado, por lo que escogí dedicar más tiempo el fin de semana, en vez de menos tiempo, pero de manera diaria. A mí esto me sirvió para poder llevar una continuidad, ya que trabajar al trabajar poco tiempo cada día, empleaba parte del tiempo en recordar lo que había hecho la sesión anterior, en cambio al ser todo en el mismo día el trabajo era más continuo.

Otra de las conclusiones obtenidas es la importancia de validar el trabajo, de manera global y a poder ser incluyendo a más personas en el proceso. Muchas veces al finalizar una pantalla o una funcionalidad, validamos y verificamos que es correcta, pero que sea correcta no implica que sea coherente con el resto del proyecto. Es importante realizar una validación amplia que no se centre solo en la pantalla que se acaba de realizar, si no que incluya también las pantallas anteriores hasta llegar a dicha pantalla, o las tareas anteriores previas a la funcionalidad a validar. Con esto te puedes dar cuenta de que el estilo de la pantalla no es coherente con el de las demás, o que la funcionalidad desarrollada se lleva a cabo de una manera diferente a las anteriores, lo que le resta coherencia a la aplicación. También es



importante realizar validaciones con personas externas al proyecto. Ya que muchas veces tu como desarrollador conoces como se debe utilizar la aplicación y el por qué está diseñada de esa manera, pero una persona externa que no conoce el sistema puede encontrar errores de diseño o de planteamiento que tu como desarrollador no eres capaz de ver.

La última de las conclusiones obtenidas en el desarrollo del proyecto es la importancia de plantear un correcto alcance de proyecto. Si desarrollas sin definir el alcance de proyecto, a la hora de cortar el desarrollo puede que la aplicación que has desarrollado no contenga todos los aspectos necesarios para ser un MPV (Mínimo producto viable), y al no contener las funcionalidades importantes o representativas puedes no pasar la validación del cliente en caso de tener un cliente, o si publicas la aplicación como demo puede que no tenga el impacto necesario que debiera tener para poder continuar la aplicación.

Los objetivos planteados para el desarrollo de la aplicación han sido cumplidos. El MVP creado contiene las funcionalidades necesarias para ser un producto funcional. Cumple el objetivo planteado de mostrar los puntos de carga, de una manera sencilla para el usuario, y mostrar información útil para el usuario y contar con funcionalidades adicionales para resultar un producto atractivo y útil para el usuario final. Desde luego el estado actual de la aplicación no es el estado final que deberá tener la aplicación una vez finalizada. Como trabajo futuro quedan las funcionalidades planteadas en el *backlog*, así como mejoras visuales, de rendimiento, de seguridad y otros aspectos que para el MVP no han sido contemplados. El alcance del proyecto ha sido más reducido de lo inicialmente planteado, que era un objetivo ambicioso, debido a la falta de tiempo. Otro de los factores que no han sido del todo desarrollados es el diseño de la interfaz. Se ha hecho lo más sencilla e intuitiva posible pero se nota que el diseño no es todo lo atractivo que podría ser un diseño hecho por una persona cualificada en el diseño de interfaces graficas.

Para englobar las conclusiones en un gran pensamiento me gustaría ultimar que las metodologías, en concreto la ágil, son una herramienta importantísima a la hora de llevar a cabo un proyecto, ya sea en solitario o en equipo. Aplicar estas metodologías te permite llevar un correcto y ordenado desarrollo del proyecto, de una manera más completa y profunda y te otorga una serie de facilidades a la hora de organizar y plantear el proceso de desarrollo.

## 9 Referencias

### Información sobre .NET MAUI

Microsoft (2023). *.NET Multi-platform App UI*.

<https://dotnet.microsoft.com/en-us/apps/maui>

Microsoft (2023). *¿Qué es .NET MAUI?*

<https://learn.microsoft.com/es-es/dotnet/maui/what-is-maui>

Jonathan Dick (2023). *GitHub*.

<https://github.com/dotnet/maui>

Gerald Versluis (2023). *Software development with .NET*.

<https://www.youtube.com/@jversluis>

### Información sobre Kotlin

JetBrains (2023). *Kotlin Programming Language*.

<https://kotlinlang.org/>

Android developers (2023). *Aprende el lenguaje de programación Kotlin*.

<https://developer.android.com/kotlin/learn?hl=es-419>

### Información sobre Angular

Google (2020). *Angular*.

<https://docs.angular.lat/>

Desarrollo web. *Angular*

<https://desarrolloweb.com/home/angular>

Capacitorjs (2022). *Angular with capacitor*.

<https://capacitorjs.com/docs/guides/angular>

### Información sobre Swift

Apple. *Swift documentación oficial*.

<https://developer.apple.com/swift/>

### Información sobre Supabase y PostgreSQL

Supabase Inc (2023). *Supabase documentación oficial*.

<https://supabase.com/changelog>

Joseph Schultz (2023). *GitHub Ssupabase C#*.

<https://github.com/supabase-community/supabase-csharp>

PostgreSQL Global Development Group (2023). *PostgreSQL documentación oficial*.

<https://www.postgresql.org/docs/current/>

## Información sobre puntos de carga

ChargeGuru (2023). *Punto de recarga de coches eléctricos: la guía 2023*.

<https://chargeguru.com/es/2022/12/19/punto-de-recarga-de-coches-electricos-la-guia-2023/>

OpenChargeMap (2023). *Open Charge Map*.

<https://openchargemap.org/site/about>

## Aplicaciones analizadas

[Electromaps \(2023\)](#). *Electromaps*.

<https://www.electromaps.com/es>

Chargemap (2023). *Chargemap*.

<https://es.chargemap.com/>

Iberdrola (2023). *Recarga pública Iberdrola*.

[Puntos de Recarga para Coches Eléctricos - IBERDROLA](#)

## Otros enlaces

ANFAC (2023). *Informe Anual de Vehículo Electrificado – 2022*.

[https://anfac.com/categorias\\_publicaciones/informe-anual/](https://anfac.com/categorias_publicaciones/informe-anual/)

Solución Individual (2017). *APPs vs Web APPs ¿En qué se diferencian?*

<https://www.solucionindividual.com/solucionindividual/nuestro-blog/entry/diferencia-app-vs-webapp.html>

SymLab (2021). *Aplicación Móvil vs Web ¿Cómo elegir?*

[Aplicación Móvil vs Web ¿Cómo elegir? | SymLab](#)

Sage (2018). *¿Qué diferencia hay entre una aplicación móvil y una web con versión móvil?*

<https://www.sage.com/es-es/blog/que-diferencia-hay-entre-una-aplicacion-movil-y-una-web-con-version-movil/>

Vanessa Rosselló (2022). *13 herramientas agile para la gestión ágil de proyectos*.

Easy Charge

<https://www.iebschool.com/blog/herramientas-gestion-agil-proyectos-agile-agile-scrum/>

Ionos (2022). *¿Qué es Jira? Todo lo que necesitas saber sobre el software de gestión de proyectos.*

<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-jira/>

Fernando Medina (2022). *Mejores tecnologías para desarrollo de aplicaciones móviles 2022.*

<https://www.armadilloamarillo.com/blog/mejores-tecnologias-para-desarrollo-de-aplicaciones-moviles-2022/>

Luis Saez (2021). *11 tecnologías para aprender los desarrolladores móviles en 2021.*

<https://es.linkedin.com/pulse/11-tecnolog%C3%ADas-para-aprender-los-desarrolladores-en-2021-saez-irurre>

Tokyo School (2022). *Diseñar interfaces en Android: cómo hacerlo y ser un experto en apps.*

<https://www.tokioschool.com/noticias/disenar-interfaces-android/>

## Imágenes utilizadas

Pexels. *Fotos de stock gratuitas.*

<https://www.pexels.com/es-es/>

Istock. *Fotos.*

<https://www.istockphoto.com/es/fotograf%C3%ADas-de-stock>

El proyecto también se ha apoyado mucho en la documentación de las asignaturas, sobre todo en las asignaturas de Proceso de software, Proyecto de ingeniería de software, Integración e interoperabilidad, Análisis y especificación de requisitos y Desarrollo de software dirigido por modelos.

## Índice de figuras

Figura 1: Persona 1.....	17
Figura 2: Persona 2.....	18
Figura 3: Persona 3.....	19
Figura 4: Gráfica trabajo completado Sprint 0.....	23
Figura 5: Gráfica trabajo pendiente Sprint 0.....	23
Figura 6: Gráfica trabajo completado Sprint 1.....	24
Figura 7: Gráfica trabajo pendiente Sprint 1.....	25
Figura 8: Gráfica trabajo completado Sprint 2.....	26
Figura 9: Gráfica trabajo pendiente Sprint 2.....	26
Figura 10: Gráfica trabajo completado Sprint 3.....	27
Figura 11: Gráfica trabajo pendiente Sprint 3.....	27
Figura 12: Hoja de ruta del proyecto.....	28
Figura 13: Diagrama de los componentes de la aplicación.....	29
Figura 14: MainPage.....	33
Figura 15: MainPage vista de satélite.....	33
Figura 16: MainPage con menú lateral desplegado.....	34
Figura 17: MainPage con punto de carga desplegado.....	34
Figura 18: ChargerPage.....	35
Figura 19: ChargerPage con información adicional desplegada.....	35
Figura 20: CreateChargerPage.....	36
Figura 21: EditChargerPage.....	36
Figura 22: FavChargerPage.....	37
Figura 23: ChargeLogPage.....	37
Figura 24: NotificationsPage.....	38
Figura 25: LoginPage.....	39
Figura 26: RegisterPage.....	39
Figura 27: Mock-ups de la MainPage.....	40
Figura 28: Mock-ups de la ChargerPage, CreateChargerPage y ProfilePage.....	41

Figura 29: Mock-ups tema gris claro.....	42
Figura 30: Mock-ups tema azul.....	42
Figura 31: Mock-ups tema rosa.....	43
Figura 32: Mock-ups tema gris oscuro.....	43
Figura 33: Diagrama de clases de negocio.....	45
Figura 34: Diagrama de clases de persistencia.....	49
Figura 35: Diagrama de flujo del funcionamiento de .NET MAUI.....	50

## Índice de tablas

Tabla 1: Análisis de competidores.....	7
Tabla 2: Perfil del competidor.....	8
Tabla 3: Oferta de productos.....	10
Tabla 4: Comparación de funciones básicas.....	11
Tabla 5: Análisis DAFO.....	12
Tabla 6: Persona 1.....	17
Tabla 7: Persona 2.....	18
Tabla 8: Persona 3.....	19
Tabla 9: Pruebas búsqueda punto de carga.....	60
Tabla 10: Pruebas creación punto de carga.....	60
Tabla 11: Pruebas editar información punto de carga.....	61
Tabla 12: Pruebas ocupar punto de carga.....	61
Tabla 13: Pruebas filtrar puntos de carga.....	62
Tabla 14: Pruebas añadir puntos de carga.....	62
Tabla 15: Pruebas creación de notificaciones.....	62

## **ANEXO 1 - CÓDIGO**

## Consideraciones

En este anexo se muestran las clases referenciadas dentro de la memoria y que no han sido incluidas en ella por tamaño y para hacer la memoria más legible.

Para mayor legibilidad del código escrito en este anexo, se ha utilizado la convención de colores utilizada en el IDE Visual Studio 2022 por defecto. Esto se ha utilizado para ofrecer claridad a la hora de leer e interpretar el código, ya que el texto plano sin ningún tipo de color o resalte es complicado de comprender.



## Clase `ActiveUser` (singleton)

Esta clase representa al usuario cuando inicia sesión. Contiene el objeto de la clase `User` como *singleton*. Esta clase gestiona el ciclo de vida de dicho objeto *singleton*.

Contiene los métodos `setUser`, que permite establecer un objeto usuario como `user`. En caso de ya existir un objeto `user` este método lanza una excepción indicando que previamente existía un objeto `user`.

El método `getActiveUser`, que devuelve el objeto `user`. En caso de no existir dicho objeto `user`, este método lanza una excepción indicando que no existe ningún usuario activo.

El método `isLoggedIn`, que comprueba si existe un usuario activo o no.

El método `clear`, que borra el valor del objeto `user`.

```
public static class ActiveUser {  
    private static User user;  
    public static void setUser(User newUser) {  
        if (user is null)  
            user = newUser;  
        else  
            throw new Exception("Ya hay usuario activo");  
    }  
    public static User getActiveUser() {  
        if (user is null)  
            throw new Exception("No hay usuario activo");  
        return user;  
    }  
    public static bool isLoggedIn() { return user != null; }  
    public static void Clear() {  
        user = null;  
    }  
}
```

## Clase MapChargers (*singleton*)

Esta clase representa la instancia única de los puntos de carga del mapa. El mapa utiliza esta lista para crear los *Pins* que se muestran sobre el mapa. Las diferentes pantallas acceden a esta lista y la actualizan, de modo que al volver al mapa este cuenta con la lista actualizada.

Se basa en dos listas de objetos de la clase *Charger*, una clase donde se realizan las actualizaciones y otra que sirve como comparación para ver los cambios que ha habido y solo actualizar los puntos de carga afectados, por optimización del tiempo de respuesta de carga del mapa.

Contiene los métodos *setChargers*, que si existe la lista principal, la copia en la lista secundaria y establece la lista de cargadores en la principal.

El método *getChargers*, que obtiene la lista principal.

El método *getDifChargers*, que obtiene una lista con los puntos de carga que han sido modificados en la lista principal, comparándolos con la lista secundaria que no ha recibido modificaciones.

El método *addChargers*, que añade un cargador a la lista principal de cargadores.

El método *removeChargers*, que elimina un cargador de la lista principal de cargadores.

El método *isEmpty*, que comprueba si la lista principal está vacía o no existe.

El método *commit* que, a modo de finalización de la transacción, copia la lista principal en la secundaria para confirmar que los cambios han sido actualizados en el mapa.

```
public static class MapChargers {  
    private static List<Charger> mapChargers;  
    private static List<Charger> oldMapChargers;  
    public static void setChargers(List<Charger> chargers) {  
        if(mapChargers is not null)  
            oldMapChargers = mapChargers;  
        mapChargers = chargers;  
    }  
    public static List<Charger> getChargers() {  
        return mapChargers;  
    }  
    public static List<Charger> getDifChargers() {
```

## Easy Charge

```
        if(oldMapChargers.Count>mapChargers.Count)
            return oldMapChargers.Except(mapChargers).ToList();
        else
            return mapChargers.Except(oldMapChargers).ToList();
    }

    public static void addChargers(Charger charger) {
        if (mapChargers is not null)
            mapChargers.Add(charger);
    }

    public static void removeChargers(Charger charger) {
        if (mapChargers is not null)
            mapChargers.Remove(charger);
    }

    public static bool isEmpty() {
        return mapChargers == null || mapChargers.Count == 0;
    }

    public static void commit() {
        oldMapChargers = mapChargers.ToList();
    }
}
```

## Clase Charger y clase Memento (Memento)

Esta clase representa los puntos de carga. Cuando estos son modificados por el usuario, se guarda una instantánea del estado del punto de carga previo a la modificación, para más adelante poder recuperarse en caso de ser requerido.

La clase contiene un objeto de tipo Memento, una clase interna a la clase *Charger* que contiene los atributos necesarios para realizar la instantánea.

Relacionados con el patrón memento, la clase contiene los métodos *CreateMemento*, que crea el objeto Memento a partir de la información de la clase.

El método *MementoToJson*, que serializa el objeto Memento a formato JSON, que es la manera en que se serializa dicho atributo.

El método *MementoFromJson*, que deserializa el objeto Memento a partir de su información en formato JSON.

```
public class Charger {  
    ...  
    public Memento? Memento { get; set; }  
    ...  
    public ChargerModel GetModel() => new ChargerModel() {  
        Id = Id,  
        Address = Address,  
        Latitude = Latitude,  
        Longitude = Longitude,  
        Quantity = Quantity,  
        Description = Description,  
        Broken = Broken,  
        AditionallInfo= AditionallInfo,  
        Ports = Ports,  
        Memento = this.MementoToJson()  
    };  
    public static Charger getFromModel(ChargerModel model) => new()  
    {
```

## Easy Charge

```
    Id = model.Id,
    Address = model.Address,
    Latitude = model.Latitude,
    Longitude = model.Longitude,
    Quantity = model.Quantity,
    Description = model.Description,
    Broken = model.Broken,
    AdditionalInfo = model.AdditionalInfo,
    Ports = model.Ports,
    Memento = Charger.MementoFromJson(model.Memento)
};
...
public Memento CreateMemento() {
    Memento snapshot = new Memento() {
        Id = this.Id,
        Address = this.Address,
        Latitude = this.Latitude,
        Longitude = this.Longitude,
        Quantity = this.Quantity,
        Description = this.Description,
        Broken = this.Broken,
        AdditionalInfo = this.AdditionalInfo,
        Ports = this.Ports,
        snapshotDate = DateTime.Now
    };
    return snapshot;
}
```

```
private JObject MementoToJson() {
    if(Memento is not null) {
        JObject snapshot = new JObject();
        snapshot = JObject.FromObject(this.Memento);
        return snapshot;
    } else { return null; }
}

private static Memento MementoFromJson(JObject snapshot) {
    if(snapshot is not null) {
        JsonSerializer serializer = new JsonSerializer();
        return serializer.Deserialize<Memento>(snapshot.CreateReader());
    } else { return null; }
}

...
}

public class Memento {
    public int Id { get; set; }
    public string Address { get; set; }
    public float Latitude { get; set; }
    public float Longitude { get; set; }
    public int Quantity { get; set; }
    public string Description { get; set; }
    public bool Broken { get; set; }
    public JObject AdditionalInfo { get; set; }
    public bool[] Ports { get; set; }
    public DateTime snapshotDate { get; set; }
}
```

## Paralelización filtrado de puntos de carga

La velocidad de respuesta de la funcionalidad de filtrar los puntos de carga, antes del llenado masivo de la base de datos, era aceptable. Pero con el nuevo volumen de datos, el tiempo de respuesta del mismo método era de aproximadamente 3 minutos y además bloqueaba el uso de la aplicación.

Para solventar este problema se ha recurrido a la asincronicidad y al paralelismo. La función *FilterMap*, que es llamada desde el botón de aceptar dentro del control de *SearchBarControl*. Este control invoca la llamada con la lista de cargadores obtenida del *singleton MapChagers*. Entonces se lanzan 3 hilos (se lanzan 3 ya que el emulador de dispositivo Android y la mayoría de teléfonos disponen de 4 hilos de procesamiento y se reserva un hilo para el hilo principal de la interfaz que utiliza .NET MAUI) que dividen la lista de cargadores en 3 y aplican el filtrado de los puntos de carga en paralelo con los parámetros indicados. El hilo principal se queda a la espera del resultado de estos hilos de manera asíncrona, por lo que el usuario mantiene control del hilo principal y puede seguir operando en la aplicación ya que esta no se bloquea. Una vez finalizan los filtrados el hilo principal recoge las listas filtradas y las agrupa nuevamente en una única. Esta lista es pasada a un método que limpia todos los puntos de carga y añade los puntos filtrados al *singleton* de *MapChagers*, y al finalizar realiza el *commit*.

```
public static async void FilterMap(Microsoft.Maui.Controls.Maps.Map map,
    EventHandler<PinClickedEventArgs> handler, Dictionary<string, bool> ports, double
    distance) {

    List<Charger> chargers = MapChagers.getChargers();

    var aux = new List<Charger>();

    Task<List<Charger>>[] tasks = new Task<List<Charger>>[3];

    for (int i = 0; i < 3; i++) {

        var t = i;

        Task<List<Charger>> task = Task.Run(() => filterThread(chargers, distance, ports, t));

        tasks[i] = task;

    }

    for (int i = 0; i < 3; i++) {

        aux.AddRange(await tasks[i]);

    }

}
```

## Easy Charge

```
map.Pins.Clear();
await setChargers(map, aux, handler);
}

static async Task<List<Charger>> filterThread(List<Charger> chargers, double distance,
Dictionary<string, bool> ports, int t) {
    try {
        var i = t;

        List<Charger> aux = chargers.Skip((chargers.Count / 3) * i).Take((chargers.Count /
            3)).ToList();

        List<Charger> res = aux.ToList();
        foreach (Charger charger in aux) {
            if (!charger.ContainsPort(ports)) {
                res.Remove(charger);
            }
            else if (distance != 0 && await DistanceFromUser(charger) > distance) {
                res.Remove(charger);
            }
        }
        return res;
    } catch (Exception e) {
        Console.WriteLine(e);
    }
    return null;
}

public static async Task<List<Charger>> setChargers(Microsoft.Maui.Controls.Maps.Map map,
List<Charger> chargers, EventHandler<PinClickedEventArgs> handler) {

    MapChargers.setChargers(chargers);

    foreach (var charger in chargers) {
        var location = new Location(charger.Latitude, charger.Longitude);
        var pin = new Pin()
        {
```



## Easy Charge

```
        Label = charger.Address,  
        Type = PinType.Place,  
        Location = location,  
  
    };  
    pin.MarkerClicked += handler;  
    map.Pins.Add(pin);  
}  
  
MapChargers.commit();  
return chargers;  
}
```

## Pruebas de aceptación GC05 Ocupar cargador y GC06

### Desocupar cargador

Esta clase contiene las pruebas para todos los posibles estados de un cargador. Para verificar la completitud del test, se han utilizado valores en todos los posibles rangos y combinaciones para los valores del estado del cargador.

```
public static class SingleChargerTest {

    private static List<SingleChargerModel> SingleChargerModels() {

        var result = new List<SingleChargerModel>();

        result.Add(new SingleChargerModel() { Id = 1, Broken = true, Occupied = true, FreedTime = null, });

        result.Add(new SingleChargerModel() { Id = 1, Broken = true, Occupied = true, FreedTime = DateTime.Now.AddDays(-5), });

        result.Add(new SingleChargerModel() { Id = 1, Broken = true, Occupied = true, FreedTime = DateTime.Now.AddMinutes(5) });

        result.Add(new SingleChargerModel() { Id = 1, Broken = true, Occupied = true, FreedTime = DateTime.Now.AddMinutes(55) });

        result.Add(new SingleChargerModel() { Id = 1, Broken = true, Occupied = false, FreedTime = null, });

        result.Add(new SingleChargerModel() { Id = 1, Broken = true, Occupied = false, FreedTime = DateTime.Now.AddDays(-5) });

        result.Add(new SingleChargerModel() { Id = 1, Broken = true, Occupied = false, FreedTime = DateTime.Now.AddMinutes(5) });

        result.Add(new SingleChargerModel() { Id = 1, Broken = true, Occupied = false, FreedTime = DateTime.Now.AddMinutes(55) });

        result.Add(new SingleChargerModel() { Id = 1, Broken = false, Occupied = true, FreedTime = null, });

        result.Add(new SingleChargerModel() { Id = 1, Broken = false, Occupied = true, FreedTime = DateTime.Now.AddDays(-5) });

        result.Add(new SingleChargerModel() { Id = 1, Broken = false, Occupied = true, FreedTime = DateTime.Now.AddMinutes(5) });

        result.Add(new SingleChargerModel() { Id = 1, Broken = false, Occupied = true, FreedTime = DateTime.Now.AddMinutes(55) });

        result.Add(new SingleChargerModel() { Id = 1, Broken = false, Occupied = false, FreedTime = null });
    }
}
```

## Easy Charge

```
result.Add(new SingleChargerModel() { Id = 1, Broken = false, Occupied = false, FreedTime = DateTime.Now.AddDays(-5) });
```

```
result.Add(new SingleChargerModel() { Id = 1, Broken = false, Occupied = false, FreedTime = DateTime.Now.AddMinutes(5) });
```

```
result.Add(new SingleChargerModel() { Id = 1, Broken = false, Occupied = false, FreedTime = DateTime.Now.AddMinutes(55) });
```

```
return result;
```

```
}
```

```
public static bool TestState() {
```

```
var modelList = SingleChargerModels();
```

```
var list = SingleCharger.GetFromModelList(modelList);
```

```
foreach (var single in list) {
```

```
if (single.Broken) {
```

```
if (single.State != 4)
```

```
return false;
```

```
else if (single.FreedTime != null)
```

```
return false;
```

```
else if (single.Occupied)
```

```
return false;
```

```
}
```

```
else {
```

```
if (!single.Occupied) {
```

```
if (single.State != 0)
```

```
return false;
```

```
else if (single.FreedTime != null)
```

```
return false;
```

```
}
```

```
else {
```

```
if (single.FreedTime == null) {
```

```
if (single.State != 3)
```

## Easy Charge

```
        return false;
    }
    else if (single.FreedTime <= DateTime.Now) {
        if (single.State != 0)
            return false;
        else if (single.FreedTime != null)
            return false;
    }
    else if (single.FreedTime > DateTime.Now && single.FreedTime <=
DateTime.Now.AddMinutes(15)) {
        if (single.State != 1)
            return false;
    }
    else if (single.FreedTime > DateTime.Now.AddMinutes(55)) {
        if (single.State != 2)
            return false;
    }
}
}
}
}
return true;
}
}
```

## **ANEXO 2 – DESCRIPCIÓN DE REQUISITOS**

## Gestión de usuarios

### [GU01](#) Nombre en *Shell*.

Cuando el usuario inicie sesión en la aplicación, poder visualizar el nombre de la cuenta en el menú desplegable lateral. Cuando el usuario no inicie sesión aparecerá el nombre de la aplicación 'Easy Charge'.

### [GU02](#) Iniciar sesión.

Pantalla de inicio de sesión y funcionalidad para iniciar sesión en una cuenta ya creada. También se permite acceder al usuario a la aplicación sin tener que iniciar sesión.

### [GU03](#) Registrar usuario.

Pantalla de crear cuenta y la funcionalidad para crear una nueva cuenta de usuario.

### [GU04](#) Cambiar la contraseña.

Pantalla para poder cambiar la contraseña mediante una confirmación introduciendo la contraseña anterior.

### [GU05](#) Cerrar sesión.

Cuando el usuario ha iniciado sesión poder cerrar sesión desde el botón del menú desplegable lateral.

### [GU06](#) Añadir cargadores a favoritos.

Desde la pantalla del punto de carga, con la cuenta de usuario iniciada, poder añadir el punto de carga a la lista de favoritos.

### [GU07](#) Eliminar cargadores favoritos.

Desde la pantalla del punto de carga, con la cuenta de usuario iniciada, poder eliminar el punto de carga a la lista de favoritos.

### [GU08](#) Ver cargadores favoritos.

Pantalla para poder visualizar la lista de cargadores favoritos. También el *ContentView* que representa el punto de carga añadido a favoritos. Desde este *ContentView* poder navegar con un *click* a la página del punto de carga.

### [GU09](#) Crear alarmas/recordatorios.

*Popup* desde el cual se crean alarmas y recordatorios programables en el tiempo. Poder añadir título, descripción, hora y periodicidad a la alarma/recordatorio.

[GU10](#) Visualizar alarmas/recordatorios.

Pantalla desde la cual se pueden visualizar las alarmas y recordatorios programados. También el *ContentView* que representa las alarmas/recordatorios y que muestra su información asociada.

[GU11](#) Eliminar/ pausar las alarmas/recordatorios.

Desde la pantalla en la que se visualizan las alarmas/recordatorios, poder pausar para que no suene pero sin llegar a eliminar. También la opción de eliminar de manera definitiva las alarmas y recordatorios programados.

[GU12](#) Visualizar el historial de recargas.

Pantalla desde la cual visualizar la información de las recargas realizadas por el usuario. También el *ContentView* que representa el histórico de la recarga y que muestra su información asociada.

[GU13](#) Ver estadísticas del coche.

Pantalla que permite visualizar diferentes estadísticas sobre el uso de la batería del vehículo. Por ejemplo la cantidad de kilómetros recorrida entre recargas, el uso medio de la batería, el tiempo que tarda la batería en recargarse, número de recargas en función del tiempo, etc.

[GU14](#) Añadir coche.

Desde el perfil de usuario poder asociar un vehículo al usuario, introduciendo los datos de nombre del vehículo, marca, capacidad de la batería, usos medios de la batería (urbano, extra-urbano y mixto).

[GU15](#) Eliminar coche.

Desde el perfil de usuario poder eliminar un vehículo asociado a un usuario.

[GU16](#) Visualizar perfil de usuario.

Pantalla que contiene la información del usuario, nombre del perfil del usuario, correo asociado, vehículos asociados, descripción.

[GU17](#) Editar perfil de usuario.

Permitir al usuario modificar los datos que no identifican al usuario de su perfil. El nombre de usuario o la contraseña no.

[GU18](#) Añadir foto de perfil.

Poder subir una foto para personalizar el perfil de usuario.

[GU19](#) Añadir fotos del vehículo.

Poder subir fotos de los vehículos asociados al usuario.

## Gestión de cargadores

### [GC01](#) Crear punto de carga.

Crear un punto de carga en la ubicación indicada, introduciendo los datos necesarios, la dirección se calcula a partir de la ubicación, el número de cargadores y los puertos de los que dispone.

### [GC02](#) Modificar datos del punto de carga.

Poder modificar los datos que no identifican el punto de carga. La dirección o la ubicación no se pueden modificar.

### [GC03](#) Ver punto de carga.

*ContentView* que aparece en la parte inferior de la pantalla principal al clickar en un punto de carga en el mapa y muestra la información mínima esencial, como la dirección, número de cargadores disponibles y breve descripción. Al clickar sobre el *ContentView* se muestra la pantalla con la información completa del punto de carga. Por ejemplo dirección, número de cargadores con sus respectivos estados, operaciones sobre los cargadores, descripción, opción de modificar, añadir a favoritos e información adicional.

### [GC04](#) Información adicional.

Permitir al usuario añadir información adicional sobre el punto de carga (precio, horario, propietario, información de contacto, etc). En la pantalla del punto de carga aparece el botón que muestra dicha información como un popup.

### [GC05](#) Ocupar cargador.

Desde la pantalla del punto de carga, marcar un punto de carga individual como ocupado, por un tiempo fijo o indefinido. Esto se muestra luego al clickar en el mapa sobre un punto, el desplegable muestra la cantidad de cargadores disponibles. Dentro de la pantalla del punto de carga los estados de los cargadores se muestran mediante iconos, un icono de recarga verde el disponible, un icono de recarga rojo los ocupados por tiempo indefinido, un icono de un reloj rojo los puntos de carga que aun quedan más de 15 minutos para su liberación, un icono de un reloj naranja para los cargadores que en menos de 15 minutos se liberan y una señal de prohibido para indicar que el cargador está averiado. El tiempo restante también se muestra en caso de haberlo.

### [GC06](#) Desocupar cargador.

Poder indicar que un punto de carga individual que estaba ocupado ahora está disponible.



[GC07](#) Marcar cargador como averiado.

Poder marcar un punto de carga individual como averiado. También se puede marcar como disponible otra vez un punto marcado como averiado.

[GC08](#) Marcar punto de carga como averiado.

Poder marcar el punto de carga completo como averiado. Esto se mostrará en el desplegable que muestra la información imprescindible en el mapa. Este estado se activará automáticamente si todos los cargadores del punto son marcados como averiados.

[GC09](#) Añadir foto del punto de carga.

Al crear el punto de carga permitir al usuario incluir una foto del lugar. Al modificar la información del punto de carga también se podrá modificar la imagen o incluirla en caso de que no tuviese una previamente.

[GC10](#) Comentarios/chat/foro.

Permitir al usuario añadir comentarios e interactuar con otros usuarios sobre el punto de carga. A modo de foro se podrán publicar mensajes y estos podrán ser visualizados por otros usuarios que accedan al mismo punto de carga y podrán ser respondidos.

[GC11](#) Conectar con la plataforma del punto de carga.

Disponer de un medio para conectar tu cuenta de usuario de la aplicación con las plataformas de los puntos de carga. Adicionalmente en caso de que el punto de carga requiera validación (NFC, mensaje a través de un dispositivo, etc) esto se podrá gestionar desde la aplicación de manera más sencilla.

[GC12](#) Pagar desde la aplicación.

Permitir al usuario realizar los pagos a las plataformas de los puntos de carga, mediante una tarjeta de crédito vinculada a la cuenta, o mediante una cartera virtual que se podrá recargar por diferentes métodos como Paypal o tarjetas de prepago.

## Gestión del mapa

### [GM01](#) Insertar mapa.

Pantalla principal de la aplicación, donde se muestra el mapa sincronizado en tiempo real.

### [GM02](#) Ubicación del usuario.

Mostrar sobre el mapa la ubicación del usuario en tiempo real.

### [GM03](#) Añadir puntos de carga al mapa.

visualizar los puntos de carga sobre el mapa, y *clickando* sobre los marcadores que indican la ubicación de los puntos de carga, desplegar el *ContentView* que muestra la información esencial sobre estos.

### [GM04](#) Cargador más cercano.

Función que localiza rápidamente el cargador más cercano al usuario, centra el mapa en la ubicación de dicho punto de carga y despliega el *ContentView* que muestra la información asociada.

### [GM05](#) Personalizar mapa.

Permitir al usuario seleccionar el modo en el que quiere mostrar el mapa, predeterminado o satélite.

### [GM06](#) Mejoras visuales al mapa.

Sustituir los marcadores de puntos de carga que utiliza Google Maps por defecto y utilizar marcadores personalizados que cambian en función del estado del punto de carga.

### [GM07](#) Filtrar puntos de carga por estado.

Permitir al usuario filtrar los cargadores, por puertos de los que dispone el punto, distancia máxima o estado en el que se encuentra el punto de carga.

## **ANEXO 3 – OBJETIVOS DE DESARROLLO SOSTENIBLE**

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

<b>Objetivos de Desarrollo Sostenibles</b>	<b>Alto</b>	<b>Medio</b>	<b>Bajo</b>	<b>No Procede</b>
❖ <b>ODS 1 Fin de la pobreza.</b>				<b>X</b>
❖ <b>ODS 2 Hambre cero.</b>				<b>X</b>
❖ <b>ODS 3 Salud y bienestar.</b>		<b>X</b>		
❖ <b>ODS 4 Educación de calidad.</b>				<b>X</b>
❖ <b>ODS 5 Igualdad de género.</b>				<b>X</b>
❖ <b>ODS 6 Agua limpia y saneamiento.</b>		<b>X</b>		
❖ <b>ODS 7 Energía asequible y no contaminante.</b>	<b>X</b>			
❖ <b>ODS 8 Trabajo decente y crecimiento económico.</b>				<b>X</b>
❖ <b>ODS 9 Industria, innovación e infraestructuras.</b>	<b>X</b>			
❖ <b>ODS 10 Reducción de las desigualdades.</b>				<b>X</b>
❖ <b>ODS 11 Ciudades y comunidades sostenibles.</b>	<b>X</b>			
❖ <b>ODS 12 Producción y consumo responsables.</b>			<b>X</b>	
❖ <b>ODS 13 Acción por el clima.</b>	<b>X</b>			
❖ <b>ODS 14 Vida submarina.</b>				<b>X</b>
❖ <b>ODS 15 Vida de ecosistemas terrestres.</b>		<b>X</b>		
❖ <b>ODS 16 Paz, justicia e instituciones sólidas.</b>				<b>X</b>
❖ <b>ODS 17 Alianzas para lograr objetivos.</b>				<b>X</b>

*Tabla 16: Impacto en los objetivos de desarrollo sostenible.*

La aplicación fomenta a los propietarios de vehículos eléctricos a hacer un uso más eficiente de los puntos de carga, lo que reduce el gasto y despilfarro energético por su parte. Además fomenta a personas que no tienen un vehículo eléctrico a adquirir uno. Esto Tiene una serie de implicaciones positivas para el medio ambiente ya que la contaminación por las emisiones de co2 en el aire disminuirán a medida que disminuyan los vehículos de combustión interna. Aun hay cuestiones por aclarar en materia de sostenibilidad de los vehículos eléctricos, como la contaminación en la producción de las baterías para estos vehículos, el aumento de consumo eléctrico en caso de que todo el mundo tenga que cargar su vehículo, o la complicación de crear una red eléctrica y la infraestructura suficiente para sostener que todos los vehículos

sean eléctricos. Hay cuestiones sin resolver pero el rumbo que se toma desde Europa es fomentar el uso de estos vehículos y Easy Charge ofrece facilidades para las personas que disponen de este tipo de vehículos.

Los ODS con los que Easy Charge está relacionado directamente son:

- **ODS 3 Salud y bienestar.** Al haber menos contaminación en el aire, la calidad del aire respirado por las personas mejora y esto tiene un impacto positivo en toda la población.
- **ODS 6 Agua limpia y saneamiento.** Los barcos y petroleras contaminan mucho los mares, y cuando sucede un accidente con los barcos que transportan dicho petróleo tienen consecuencias desmesuradas en el ecosistema de la zona, por lo que depender menos de estos combustibles la contaminación del agua disminuirá.
- **ODS 7 Energía asequible y no contaminante.** La obtención de energía eléctrica de manera sostenible es más asequible que la obtención de petróleo de manera sostenible. El consumo energético conlleva contaminación de manera inevitable pero la energía eléctrica es más renovable y menos contaminante.
- **ODS 9 Industria, innovación e infraestructuras.** El fomento del uso de vehículos eléctricos conlleva un crecimiento de la industria y con ello más innovación en el campo y se requerirá más infraestructura que soporte dichos vehículos.
- **ODS 11 Ciudades y comunidades sostenibles.** La aplicación fomenta el uso responsable y a reducir el despilfarro eléctrico, por lo que se genera un movimiento de sostenibilidad entre las personas que utilizan la aplicación y si el movimiento es general esto se traslada a la comunidad.
- **ODS 12 Producción y consumo responsables.** Al igual que con otros ODS, la aplicación fomenta el uso eficiente y a no despilfarrar electricidad. En general a ser responsables.
- **ODS 13 Acción por el clima.** Esta ODS engloba varias ODS con los que Easy Charge está muy relacionada directamente. Al fomentar el uso de vehículos menos contaminantes y del uso de energías renovables, la calidad del clima puede mejorar en gran medida. La contaminación del aire será menor y la del agua también, esto fomentará que la fauna y la flora puedan volver a un estado más naturalizado y reducir el impacto de la huella de carbono.
- **ODS 15 Vida de ecosistemas terrestres.** Así como con el anterior ODS, fomentar el uso de energías renovables conlleva una mejora directa en la calidad de vida de la fauna de la zona. Esto tiene un impacto directo en la calidad de vida de los animales y el ecosistema, que poco a poco podrá volver a naturalizarse.