



**Aplicación móvil para el acceso a la red de Cargadores  
Eléctricos**

**Trabajo de diploma para optar por el título de Ingeniero  
en Ciencias Informáticas**

**Autor:**

Dairon Hernández Blanco

**Tutores:**

Frank Geiler Vega Duverger

M. Sc. Yadira Ramírez Rodríguez

**La Habana, 2022**

## *Declaración de Autoría*


### **Declaración de Autoría**

Declaramos ser autores de la presente tesis que tiene por título: y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los  2  días del mes de  12  del año  2022 .



Firma del Autor

Dairon Hernández Blanco



Firma del Tutor

M. Sc. Yadira Ramírez Rodríguez



Firma del Tutor

Ing. Frank Geiler Vega Duverger

### **Agradecimientos**

## *Dedicatoria*

*A mis padres, Frank Hernández González y Damaris Blanco Pereira, a mi hermana Yudit Hernández Blanco por su consagración, su amor, su interminable confianza y por ser unos ejemplos a seguir.*

*A mi abuela que la tengo presente en mi corazón. En general a mi familia que de una forma u otra se evidenció su afecto para que yo siguiera en la carrera.*

*A mis tutores Frank Geiler y Yadira Ramírez por toda su ayuda y paciencia. A Adolfo Santana y Andy Suárez.*

*A mis amigos: Alejandro Monaga, José Javier, Dionis, a Samira por la paciencia que me tuvo. A Adrian, Carlos, Elier, Roilan, Rolando, Yoidel y Henry que de subirle la dificultad a la uci en muchos momentos me levantaron el ánimo.*

*A Damaris, Yarieli y Ilenny por todo su apoyo por su inigualable compañía.*

*A la fábrica, Julio y Carlos Pardo por toda la ayuda.*

*A Dr.Arian y Dr.Juanci, al Pamuá por hacerme sentir parte de su familia.*

*A mis amigos de la infancia y a todas las grandes amistades que he conocido en estos últimos años y se quedaron gracias por su preocupación .*

*A Dayron y Daniel Hill que a pesar de la distancia siempre estamos en contacto les deseo el éxito que me desean a mí.*

*A U91 por todo lo que me dio en su momento a pesar de que lo bueno no dura para siempre, se agradece.*

*A todo aquel que de alguna forma ha contribuido a mi formación durante la carrera, les doy gracias por apoyarme en las buenas y malas circunstancias.*

**Dedicatoria**

*A mis padres, los cuales me ayudaron incondicionalmente a ser lo que soy, me enseñaron a creer que se podía cuando ya estaba todo perdido.*

*A todos mis seres queridos, a mis grandes amigos que han compartido momentos inolvidables en el transcurso de la carrera.*

*Gracias a todos.*

### Resumen

El impacto de las nuevas tecnologías móviles dentro de la sociedad se está dando de una manera extraordinaria, actualmente sin estas herramientas móviles no se tendrían comunicaciones ni se estaría al día de los acontecimientos de la sociedad, por ende, se han encargado de hacer que sean esenciales en la vida, a medida que transcurre el tiempo aparecen nuevas versiones o actualizaciones móviles que cumplen con las expectativas de las personas. Dentro de los sistemas operativos asociados a esta tecnología resalta android por sus características. Por tal motivo se decidió desarrollar una aplicación para dispositivos móviles que funcione como cliente de un sistema de mapeo, control y monitorización para cargadores eléctricos. Para registrar el proceso de desarrollo de la aplicación se utilizó como metodología de desarrollo de software AUP-UCI, como lenguaje de modelado UML, como herramienta de modelado Visual Paradigm, como lenguaje de programación Java y como entorno de desarrollo android studio. Como resultado se obtuvo la aplicación cliente para dispositivos móviles de cargadores eléctricos, la cual permite visualizar, reservar y cancelar los puntos de cargas.

**Palabras clave:** *android; aplicación móvil; cargadores eléctricos; cliente; tecnología.*

### **Abstract**

The impact of the new mobile technologies within society is occurring in an extraordinary way, currently without these mobile tools communications will not be used or they would be up to date with the events of society, therefore, they have been in charge of making them essential in life, as time goes by, new versions or mobile updates appear that meet people's expectations. Within the operating systems associated with this technology, Android stands out for its characteristics. For this reason, an application for mobile devices was developed that works as a client of a mapping, control and monitoring system for electric chargers. To govern the development process of the application, the AUP-UCI software development methodology was obtained, as a UML modeling language, as a Visual Paradigm modeling tool, as a Java programming language and as an android studio development environment. As a result, the client application for mobile devices of electric chargers was obtained, which allows viewing, reserving and canceling charging points.

**Keywords:** *android; app; electrical chargers; client; Mobile technology.*

## Índice de Contenido

Dedicatoria	V
Resumen	VI
Abstract	VII
INTRODUCCIÓN	1
CAPÍTULO 1: FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS	5
Introducción	5
1.1 Conceptos asociados a la investigación	5
1.2 Estudio de Aplicaciones Homólogas	7
1.3 Metodología de desarrollo de software	9
1.3.1 Descripción de la metodología AUP variación UCI	9
1.4 Herramienta de modelado	11
1.5 Sistema Operativo	11
1.6 Entorno de desarrollo	13
1.7 Lenguaje de Programación	15
1.8 Lenguaje etiquetado	15
Conclusiones parciales del capítulo	16
CAPÍTULO 2: DESCRIPCIÓN DE LA APLICACIÓN MÓVIL PROPUESTA	17
Introducción	17
2.1 Descripción de la propuesta de solución	17
2.2 Requisitos del sistema	18
2.2.1 Definición de los requisitos funcionales	18
2.2.2 Requisitos no funcionales del sistema	19
2.3 Descripción de Historias de Usuarios	20
2.4 Diseño Arquitectónico	29
2.4.1 Patrón arquitectónico Modelo Vista Presentador	29
2.5 Patrones de diseño	30
2.5.1 Patrones de Software para Asignar Responsabilidades	31
2.5.2 Patrones GoF	34
2.6 Estándares de Codificación	35
Conclusiones parciales del capítulo	37
CAPÍTULO 3: VALIDACIÓN DE LA APLICACIÓN MÓVIL PROPUESTA	39

## *Índice de Contenido*

Introducción	39
3.1 Pruebas	39
3.1.1 Niveles de prueba	39
3.1.2 Métodos de prueba	40
3.2 Pruebas funcionales	41
3.2.2 Pruebas de Aceptación	42
Conclusiones del capítulo	45
CONCLUSIONES GENERALES	46
RECOMENDACIONES	47
REFERENCIAS BIBLIOGRÁFICAS	48
ANEXOS	<b>¡Error! Marcador no definido.</b>



**Índice de Ilustraciones**

Ilustración 1 Cuota de mercado de sistemas operativos móviles en todo el mundo .....	12
Ilustración 2: Cuota de mercado de sistemas operativos móviles en Cuba.....	12
Ilustración 3: Propuesta de solución. (Elaboración Propia) .....	18
Ilustración 4: Patrón arquitectónico Modelo Vista Presentador .....	30
Ilustración 5: Patrón Creador (Elaboración Propia) .....	32
Ilustración 6 Patrón Decorador. (Elaboración Propia) .....	35
Ilustración 7: Patrón Observador. (Elaboración Propia) .....	35
Ilustración 9: Identificadores. (Elaboración Propia).....	36
Ilustración 10:Resumen de las no conformidades encontradas (Elaboración Propia) .....	45

## Índice de Tablas

Tabla 1: Descripción de los RF para el desarrollo de la aplicación móvil .....	18
Tabla 2: Localizar la existencia de puntos de carga en una ubicación determinada .....	20
Tabla 3: Reservar un conector de un punto de carga. ....	21
Tabla 4: Filtrar la información de los puntos de carga de acuerdo a disponibilidad. ....	23
Tabla 5: Mostrar la dirección de la ubicación del punto de carga.....	24
Tabla 6: Mostrar la cantidad de conectores de un punto de carga.....	25
Tabla 7: Cancelar reservación automáticamente luego de pasado el tiempo de iniciar el proceso de carga del vehículo. ....	26
Tabla 8: Detener proceso de carga. ....	27
Tabla 9: Mostrar ubicación del punto de carga reservado en un mapa.....	28
Tabla 10: Caso de Prueba Filtrar la información de los puntos de carga de acuerdo a disponibilidad.....	42
Tabla 11: Caso de Prueba Mostrar la dirección de la ubicación del punto de carga .....	42

## INTRODUCCIÓN

El número de usuarios de teléfonos inteligentes a nivel mundial supera los 3.000 millones y se prevé que siga creciendo de forma paulatina durante los próximos años. A través de esta vía se resuelven gran parte de las problemáticas que afectan a la población, facilitando las diversas tareas y gracias a estas aplicaciones desarrolladas podemos optimizar y desarrollar labores que antiguamente se les hacían difíciles o muy complicadas. Uno de estos problemas que afectan diariamente a estos usuarios es la ubicación, disponibilidad y reservación de puntos de cargas para vehículos eléctricos o híbridos enchufables que usen cargadores eléctricos, los cuales han desarrollado y ha aumentado su crecimiento a nivel mundial ya que es una fuente confiable para el cuidado del medio ambiente. El desarrollo de dichas aplicaciones se reconoce como una alternativa irreversible para resolver dicha problemática.

En Cuba la carga de vehículos eléctricos en los hogares no es algo posible debido a que la incipiente introducción de este tipo de autos solo está relacionada a el sistema empresarial (Aguas de la Habana, Etecsa, Ómnibus Urbanos. La carga de estos solo se realiza en puntos específicos de estas empresas y solo accesibles a su parque automotor.

Ante el auge de este tipo de vehículos a nivel mundial, el país ha trazado una política de creación de puntos de cargas en todo el país para permitir así la expansión de la posibilidad de la carga de dichos vehículos. La no adquisición de la planificación del conductor, para poder saber en qué zona de su viaje habrá un punto de carga para poder reservar este servicio, la disponibilidad de los conectores de los puntos de cargas provistos a visitar o si ya este reservado poder cancelarlo y planificar otra reservación. Cuba cuenta con la ausencia de una aplicación de acceso público para el uso de estos servicios.

Dada la situación problemática anteriormente descrita se define el siguiente **problema investigativo**: ¿Cómo contribuir a la búsqueda, reserva y gestión de los puntos de cargas eléctricos para vehículos eléctricos o híbridos mediante el uso de dispositivos móviles?

Una vez detectado el problema investigativo se establece como **objeto de estudio** las aplicaciones móviles para la monitorización, mapeo, reservación de puntos de carga eléctricos para vehículos eléctricos o híbridos. El objeto de estudio está enmarcado en el **campo de acción** de la monitorización, mapeo, reservación de punto de carga para vehículos en aplicaciones móviles para sistema operativo Android.

Para dar solución a la problemática planteada se propone como **objetivo general** desarrollar una aplicación Android para dispositivos móviles que permite la búsqueda, reserva y gestión de los puntos de cargas eléctricos para vehículos eléctricos o híbridos.

Para dar cumplimiento al objetivo general, se desglosan los siguientes **objetivos específicos**:

1. Construir el marco teórico referencial de la investigación, relacionado con el desarrollo de aplicaciones móviles para el uso de cargadores eléctricos.
2. Caracterizar la metodología de desarrollo de software que guiará la presente investigación, así como las tecnologías que formarán parte del ambiente de desarrollo de la aplicación móvil que se propone como solución al problema que da paso a la presente investigación.
3. Elaborar los artefactos ingenieriles necesarios como representación abstracta de la solución, que sirva de base para la implementación.
4. Implementar la aplicación móvil para la búsqueda, reserva y gestión de los puntos de cargas eléctricos para vehículos eléctricos o híbridos.
5. Validar la aplicación móvil resultante a partir de pruebas de software.

Para validar metodológicamente la investigación se utilizaron los siguientes **métodos científicos**, clasificados en teóricos y empíricos:

### **Métodos teóricos:**

- Analítico-sintético: permitió establecer el componente teórico de la investigación, segregando la misma en distintas partes para un claro entendimiento y extrayendo la esencia de los elementos más relevantes

del estudio realizado para el desarrollo de aplicaciones móviles de apoyo para el uso de cargadores eléctricos.

- **Histórico-lógico:** permitió realizar un análisis de sistemas similares existentes, tanto nacional como internacionalmente, así como la evolución de los mismos, con el fin de evaluar su viabilidad en el contexto de la investigación.
- **Modelación:** se utilizó para la realización de los diagramas necesarios en el proceso de desarrollo de software, haciendo una representación abstracta de la aplicación móvil propuesta, facilitando así el desarrollo de la misma.

### **Métodos empíricos:**

- **Entrevista:** se utilizó para, a través de la comunicación verbal, obtener información directamente sobre las necesidades existentes para el uso de estos puntos de cargas. De igual modo, fue útil a la hora de definir los distintos requisitos para elaborar la solución propuesta, y obtener la información necesaria para el desarrollo de la investigación.

El presente documento consta de tres capítulos, conclusiones generales, referencias bibliográficas y anexos. La **estructura capitular** se realiza según el nivel de detalle que requiere el contenido abordado en cada uno de los capítulos, definidos de la siguiente forma:

- **Capítulo I: Fundamentos o referentes teórico-metodológicos.** Este capítulo constituye la base teórica de la investigación, se abordan los principales conceptos a tratar durante el desarrollo de la misma, así como un análisis de sistemas similares existentes a nivel nacional e internacional. Además, se caracteriza la metodología utilizada y las tecnologías y herramientas para el desarrollo de la aplicación móvil propuesta.
- **Capítulo II: Descripción de la aplicación móvil propuesta.** En este capítulo se plasma la especificación de los requisitos de software. Además, se describe la arquitectura a utilizar, así como los patrones de diseño y

estándares de codificación que serán aplicados durante la implementación de la solución. Por último, se describe el diagrama de componentes, como resultado de la implementación, para reflejar las relaciones y las dependencias entre estos.

- **Capítulo III: Validación de la aplicación móvil propuesta.** En este capítulo se evalúa el nivel de calidad y fiabilidad de los resultados obtenidos en el desarrollo de la propuesta de solución. Dicha valoración se lleva a cabo a partir del establecimiento de métricas y una estrategia de pruebas, con el fin de verificar y revelar la calidad de la aplicación móvil antes de su entrega al cliente.

## CAPÍTULO 1: FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS

### Introducción

En este capítulo se abordan los principales conceptos asociados a la investigación para establecer un marco de discernimiento a partir del cual podrán ser evaluados los resultados alcanzados. De igual modo, se realiza un estudio de aplicaciones móviles con sistema operativo *Android*<sup>1</sup>, para verificar su viabilidad de uso en el contexto del problema que da paso a la presente investigación. Por último, se dan a conocer las tecnologías y la metodología para el desarrollo de la aplicación móvil propuesta.

### 1.1 Conceptos asociados a la investigación

Luego de la lectura de la documentación relativa con la presente investigación, se hace necesario el establecimiento de los siguientes conceptos asociados a la misma, para lograr una comprensión total del presente trabajo, que a su vez brindan soporte teórico-metodológico y conceptual a la misma.

- **Dispositivo móvil:** es un aparato de pequeño tamaño, con algunas capacidades de procesamiento, con conexión permanente o intermitente a una red, con memoria limitada, que ha sido diseñado específicamente para una función, pero que puede llevar a cabo otras más generales (Arturo Baz Alfonso, 2013).
- **Aplicación:** es un programa de computadora que se utiliza como herramientas para una operación o tarea específica. Para la informática, una aplicación es uno de diversos tipos de programas de computación diseñados especialmente para cumplimentar una función o actuar como herramienta para acciones puntuales del usuario (Bembibre, 2009).
- **Aplicación móvil:** Software para dispositivos móviles que realiza una serie de funciones concretas (Usano Carrasco, 2015).

---

<sup>1</sup> Android es el nombre de un sistema operativo que se emplea en dispositivos móviles, por lo general con pantalla táctil, aunque el software también se usa en automóviles, televisores y otras máquinas.

- **Cargadores eléctricos:** Son los equipos que permiten recargar o almacenar energía en las baterías de los vehículos eléctricos e híbridos enchufables para permitir el posterior uso de ella. Los vehículos eléctricos se pueden cargar con distintos tipos de tecnologías de cargadores, ya sean en corriente alterna (AC) o corriente directa (DC).
- **Tipo de conector en un cargador:** El tipo de conector se refiere al tipo de enchufe, los cuales tienen distintos pines y geometrías que cumplen a su vez el rol de intercomunicar el vehículo eléctrico con el cargador. En la actualidad existen múltiples tipos de conectores que se ofrecen en el mercado, liderando la oferta aquellos procedentes de estándares europeos, americanos, japoneses y chinos.
- **Modo de carga:** La carga de un vehículo eléctrico puede hacerse según cuatro (4) modos distintos según lo estipula la norma internacional IEC 61851-1. El concepto de modo de carga se ha asimilado operacionalmente respecto de la conectividad y comunicación entre el cargador eléctrico y el vehículo eléctrico. Modos de carga 1 y 2 tienen nulo o bajo nivel de comunicación, por lo tanto, un vehículo eléctrico podría no cargarse según la electrónica programada para resguardo de las baterías. Modos de carga 3 y 4 tienen alto nivel de comunicación, siendo los más recomendados. La infografía a continuación describe los cuatro (4) modos de carga y cuáles de ellos ocurren en corriente alterna (AC) o corriente continua (DC).
- **OCPP V(1.6):** Permite la carga inteligente, una característica muy deseable para el equilibrio de la carga y otras ventajas. La carga inteligente implica un sistema en el que los elementos de la red de vehículos eléctricos, incluidos los VE, las estaciones de carga y los operadores de carga, comparten conexiones de datos y acceden a detalles específicos. Todas las versiones de OCPP utilizan también una plataforma abierta para conectar los EVSE con el sistema back-end basado en la nube para facilitar la comunicación.



## 1.2 Estudio de Aplicaciones Homólogas

A continuación, se muestran varias tecnologías existentes a nivel internacional que tienen como funcionalidad la búsqueda, reserva y gestión de los puntos de cargas eléctricos para vehículos.

### **Electrify America**

Electrify America ha estado aumentando rápidamente su red de carga de vehículos eléctricos durante años. Su objetivo es construir 1.700 estaciones de carga rápida (un total de alrededor de 10.000 cargadores individuales) en los Estados Unidos para 2025. Al momento de escribir, Electrify America tiene 670 estaciones de carga (aproximadamente 2,900 cargadores rápidos CCS individuales) repartidas por todo el país. Está priorizando la energía limpia y construyendo los cargadores más rápidos posibles, ya que piensa primero en los propietarios de vehículos eléctricos. (Electrify America, 2022)

### **Chargeway**

Chargeway es una aplicación simple pero excelente que se utiliza principalmente como planificador de viajes. Ya sea que esté en su viaje diario o en un viaje por carretera de varios cientos de millas, necesita una forma de encontrar exactamente dónde están los cargadores en funcionamiento en su ruta y si tiene suficiente alcance para alcanzarlos; ahí es donde entra Chargeway.

Primero, agregue su vehículo eléctrico para encontrar los cargadores apropiados. La app solo te mostrará los cargadores que funcionan con tu coche, aunque te permite buscar otras estaciones de carga a tu gusto. El Planificador de viajes es la mejor parte de la aplicación, ya que le permite configurar su ubicación actual, destino y el porcentaje de batería actual de su vehículo. (Chargeway, 2022)

### **Plugshare**

PlugShare es una de las aplicaciones más destacadas en la App Store para encontrar cargadores EV. Lo ayuda a ubicar una amplia gama de cargadores de vehículos de múltiples redes, incluidos los grandes nombres como Electrify America, Tesla Superchargers, ChargePoint y muchos más.

Cuando se registra por primera vez en una cuenta de PlugShare, se le indica que agregue el vehículo eléctrico que posee. Esta es una característica excelente, ya que filtrará automáticamente los cargadores que funcionan con su EV sin que tenga que hacer nada. Los filtros también funcionan bien y le brindan opciones para reducir su búsqueda a una velocidad de carga específica.

PlugShare le permite agregar múltiples viajes a su perfil para mostrarle qué cargadores están disponibles en su ruta al trabajo, la casa de un amigo o su próximo viaje por carretera. Esta es también una gran característica para que los futuros propietarios de vehículos eléctricos vean si cargar un vehículo totalmente eléctrico en su ruta de trabajo es viable para ellos. (htt)

### **ChargeHub**

ChargeHub tiene una interfaz de usuario fantástica y lo ayuda a ubicar las principales redes de carga de vehículos eléctricos en su área. Cuando abre la aplicación por primera vez, le da la bienvenida con una vista de mapa de todos los cargadores disponibles a su alrededor con cuatro pestañas en la parte inferior de la pantalla: Mapa, Viajes, Comunidad y Perfil. Esta interfaz de usuario limpia ayuda a los nuevos usuarios a navegar por la aplicación fácilmente.

La sección Viajes es fácil de usar, lo que le permite agregar un cargador a su ruta y usar la aplicación ChargeHub para la navegación. La aplicación tiene una buena cantidad de filtros que puede usar sin que parezca abrumador, incluidos los importantes, como el tipo de conector, el kW mínimo, el precio y la red.

La sección Perfil también es bastante sencilla, lo que le permite agregar su marca y modelo de EV, ver sus cargadores favoritos, viajes guardados, etc. ChargeHub incluso incluye una guía de carga para nuevos propietarios de EV para comprender mejor los diferentes tipos de cargadores, conectores, el pros y contras de varias redes, y mucho más. Esta es una excelente señal de una aplicación que piensa primero en sus usuarios. (Chargehub, s.f.)

### **Resultados del análisis de las soluciones similares**

El objetivo de este trabajo es el desarrollo de una aplicación que contribuya al proceso de búsqueda, gestión y reservación de puntos de cargas eléctricos. De ahí que el análisis de las soluciones presentadas anteriormente, permitió corroborar que, en el caso de las aplicaciones, no es posible operar en las mismas sin conexión a Internet. Por otro lado, a pesar de que la mayoría posee facilidad de interfaces de usuario no satisfacen.

A pesar de esto algunas funcionalidades fueron usadas como guía para el desarrollo de la aplicación anteriormente propuesta como la información y su reservación de los puntos de carga.

A raíz del análisis realizado, ninguno de los sistemas estudiados satisfacen las necesidades. Esto evidencia la necesidad de desarrollar una aplicación móvil para el uso de la red de cargadores eléctricos de nuestro país.

### **1.3 Metodología de desarrollo de software**

Las metodologías de desarrollo de software son un conjunto de técnicas y métodos organizativos que se aplican para diseñar soluciones de software informático. Se utilizan en el ámbito de la programación, entre otros, con el objetivo de trabajar en equipo de manera organizada. Estas metodologías han ido evolucionando a lo largo del tiempo, pasando de ser un mero trámite de organización a ser una base importantísima a la hora de desarrollar software de una manera productiva y eficaz (Santander Universidades, 2020). La Universidad de las Ciencias Informáticas define como política a seguir en los proyectos productivos la utilización de la metodología AUP (Proceso Unificado Ágil) adaptada al ciclo de vida definido para la actividad productiva de la universidad, por lo que se decide su utilización para guiar el proceso de desarrollo de la propuesta de solución.

#### **1.3.1 Descripción de la metodología AUP variación UCI**

La variación de la metodología AUP para la UCI (AUP-UCI) se crea porque, no existía una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto exigiéndose así que el proceso sea

configurable. Por lo que se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Entre las principales diferencias presentes en esta versión se encuentran (Metodología de desarrollo para la Actividad productiva de la UCI., 2015):

- AUP define 4 fases de desarrollo (Inicio, Elaboración, Construcción, Transición), y su versión para la UCI mantiene la fase de Inicio, pero se modifica el objetivo, agrupa las otras 3 fases en una llamada Ejecución e incorpora una llamada Cierre.
- AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), la metodología definida en la UCI tiene 8 disciplinas.
- Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran cada uno de ellos disciplinas. Específicamente en la disciplina Requisitos plantean 4 posibles escenarios para la identificación y descripción de requisitos. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional.

En esta variante se definen 4 **escenarios** (Metodología de desarrollo para la Actividad productiva de la UCI., 2015):

1. **Escenario No1:** Proyectos que modelen el negocio con CUN (Caso de Uso del Negocio) solo pueden modelar el sistema con CUS (Casos de Uso del Sistema).
2. **Escenario No2:** Proyectos que modelen el negocio con MC (Modelo Conceptual) solo pueden modelar el sistema con CUS (Casos de Uso del Sistema).
3. **Escenario No3:** Proyectos que modelen el negocio con DPN (Descripción de Proceso de Negocio) solo pueden modelar el sistema con DRP (Descripción de Requisitos por Proceso).

4. **Escenario No4:** Proyectos que no modelen negocio solo pueden modelar el sistema con HU (Historias de usuario).

Una vez evaluada la propuesta de solución, se decidió seleccionar el escenario 4 debido a que no es un proyecto de larga durabilidad, con el negocio bien definido y siempre acompañado del cliente en cada una de las decisiones e iteraciones del sistema.

### 1.4 Herramienta de modelado

La ingeniería de *software* asistida por computadora (*CASE* del inglés *Computer Aided Software Engineering*) es el dominio de las herramientas de *software* que se utilizan para diseñar e implementar aplicaciones, son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un *Software* (Salas, 2013).

- **Visual Paradigm:** es una herramienta *CASE* que sirve para la notación de modelado de procesos de negocio, así como la generación de código e ingeniería inversa de diagramas a código. Soporta 13 tipos de diagramas, diagrama de clases, diagrama de uso de casos, diagrama de secuencias, diagrama de comunicaciones, diagrama de actividades. Entre las ventajas se consideran (Cortéz, 2018):
  - Disponibilidad multiplataforma.
  - Diseño centrado en casos de uso y enfocado al negocio. que genera *software* de mayor calidad.
  - Capacidades de ingeniería directa e inversa.
  - Licencia gratuita y comercial.

### 1.5 Sistema Operativo

En los últimos años el mercado de los dispositivos móviles ha mostrado un crecimiento notable en todo el mundo, de ahí que exista una gran cantidad de sistemas operativos para estos dispositivos. Según (Zaldivar, 2020) *el mercado de aplicaciones móviles tiene hoy en día un crecimiento importante año tras año. En la*

actualidad el mercado abarca dos grandes ecosistemas móviles: Android y iOS”. La **Ilustración 1** muestra la utilización de los sistemas operativos en los últimos años en todo el mundo, mientras que la **Ilustración 2** muestra la misma comparativa en Cuba. En ambos casos, el *ranking* es liderado por Android.

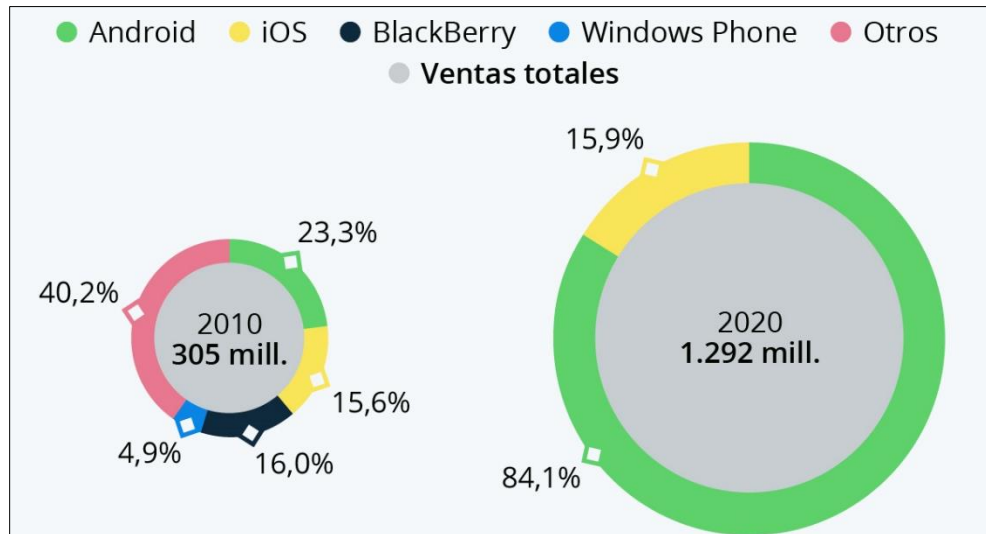


Ilustración 1 Cuota de mercado de sistemas operativos móviles en todo el mundo (Enero 2020 - Julio 2022) Tomado de (Roa, 2021).

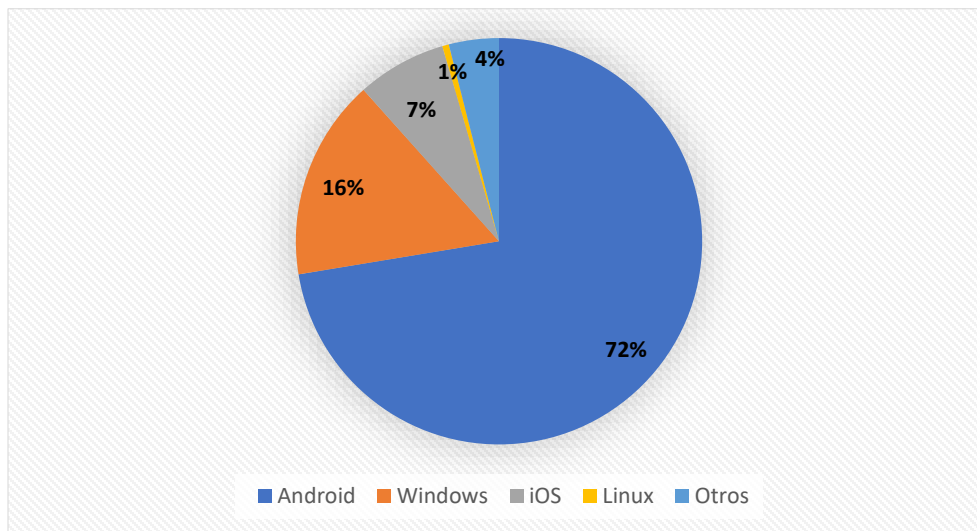


Ilustración 2: Cuota de mercado de sistemas operativos móviles en Cuba (hasta octubre 2022) [Tomado de (GlobalStats, 2022)].

Según el estudio realizado sobre las diferentes plataformas que dominan el mercado móvil, se decide desarrollar la aplicación móvil para el uso de cargadores eléctricos

por parte de vehículos eléctricos e híbridos basada en *Android*. La decisión se basa al ser éste el sistema operativo para dispositivos móvil, según (GlobalStats, 2022), más utilizado en el contexto nacional.

*Android* es un sistema operativo móvil diseñado para dispositivos móviles con pantalla táctil como teléfonos inteligentes o *Tablet*, pero que también se puede encontrar en otros dispositivos como relojes inteligentes, televisores o incluso en los sistemas multimedia de algunos modelos de coches. Es desarrollado por *Google* y basado en el *Kernel de Linux* y otros *softwares* de código abierto y que se ha convertido en el principal responsable de la popularización de muchos dispositivos inteligentes por el hecho de facilitar el uso de una gran cantidad de aplicaciones de forma sencilla (Adeva, 2022).

### 1.6 Entorno de desarrollo

Un entorno de desarrollo integrado (en inglés *Integrated Development Environment* o *IDE*) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones (Entorno de Desarrollo Integrado para C++ , 2013).

- **Android Studio v2021.3.1:** Es una actualización más ligera que en ocasiones anteriores, con novedades centradas en *Jetpack Compose* y el emulador de *Wear OS*. Por supuesto, también tenemos otras mejoras y correcciones bajo el capó y nueva versión de *IntelliJ*. Basado en *IntelliJ 2021.3*. Se incluyen las mejoras de *IntelliJ 2021.3* como la mejorada búsqueda de usos, el depurador de *Kotlin* o la inspección de condiciones de constantes de *Kotlin*. Dispositivos virtuales gestionados por *Gradle*. *Gradle* se encargará de automatizar tus tests descargando el SDK y configurando los dispositivos y ejecución de los tests y otras mejoras relacionadas.

- **Gradle v7.4:** herramienta de automatización de compilación de código abierto centrada en la flexibilidad y el rendimiento. Los *scripts* de compilación de *Gradle* se escriben con *Groovy* o *Kotlin* DSL. Es compatible con muchos *IDE* principales, incluidos *Android Studio*, *Eclipse*, *IntelliJ IDEA*, *Visual Studio 2019* y *XCode*. También puede invocar a *Gradle* a través de su interfaz de línea de comandos en su terminal o a través de su servidor de integración continua. Posee características como (Gradle, 2022):
  - **Altamente personalizable:** está modelado de una manera que es personalizable y extensible en las formas más fundamentales.
  - **Rápido:** completa las tareas rápidamente al reutilizar los resultados de ejecuciones anteriores, procesar solo los insumos que cambiaron y ejecutar tareas en paralelo.
  - **Potente:** es la herramienta de compilación oficial para *Android* y viene con soporte para muchos lenguajes y tecnologías populares.
- **SDK de Android v10 API 29:** es un conjunto de herramientas y bibliotecas de desarrollo de *software* que se requieren para desarrollar aplicaciones *Android*. Cada vez que *Google* lanza una nueva versión o actualización de *Android*, también se lanza un nuevo *SDK*. Cabe destacar que, si bien también puedes descargar y utilizar el *SDK* de *Android* sin necesidad de usar *Android Studio*, por lo general deberás trabajar con *Android Studio* para cualquier desarrollo de *Android*. Éste comprende todas las herramientas necesarias para codificar programas desde cero e incluso para ponerlos a prueba. Estas herramientas permiten que el proceso de desarrollo fluya sin problemas, desde el desarrollo y la depuración de programas hasta el empaquetado. Es compatible con *Windows*, *macOS* y *Linux*, de modo que puedes desarrollar en cualquier de estas plataformas (Vaati, 2020).
- **Emulador Genymotion v3.3.1:** es un emulador *Android* para *Windows* y también para las computadoras personales que incluye todo lo necesario de serie para funcionar en el ordenador como si fuese un teléfono o *Tablet* con el



sistema operativo de *Google*. Es compatible con la arquitectura de 64 bits, y ofrece multitud de ajustes añadidos a los propios del sistema (Linares, 2021).

### 1.7 Lenguaje de Programación

Es el conjunto de instrucciones a través del cual los humanos interactúan con las computadoras a través de algoritmos e instrucciones escritas en una sintaxis que la computadora entiende e interpreta en lenguaje de máquina. Los lenguajes de programación permiten a las computadoras procesar de forma rápida y eficientemente grandes y complejas cantidades de información. Existen varios lenguajes de programación utilizados en la industria hoy en día. Algunos lenguajes de programación populares incluyen C++, C#, *Visual Basic*, *Go*, *Ruby*, *JavaScript*, *Java* y *Python*, por mencionar algunos (Mendoza, 2020).

- **Java v 8.0:** lenguaje orientado a objetos de propósito general. La principal característica es la de ser un lenguaje compilado e interpretado. Todo programa en Java ha de compilarse y el código que se genera *bytecodes* es interpretado por una máquina virtual. De este modo se consigue la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que si son dependientes de la plataforma. Realiza comprobación estricta de tipos durante la compilación, evitando con ello problemas tales como el desbordamiento de la pila. Pero, es durante la ejecución donde se encuentra el método adecuado según el tipo de la clase receptora del mensaje; aunque siempre es posible forzar un enlace estático declarando un método como final (Introducción al lenguaje de programación Java., 2005).

### 1.8 Lenguaje etiquetado

Un lenguaje de marcas es una señal colocada en un texto con el fin de delimitar una parte del mismo para darle un formato determinado. Algunas de las marcas más empleadas son "<" y ">". A pesar de que no es un lenguaje de programación como tal, puede contener partes de códigos de otros lenguajes de programación (Fernández, 2018).

- **XML v5.0:** significa lenguaje de marcas generalizado (*Extensible Markup Language*). Es un lenguaje usado para estructurar información en un documento o en general en cualquier fichero que contenga texto, como por ejemplo ficheros de configuración de un programa o una tabla de datos. Ha ganado muchísima popularidad en los últimos años debido a ser un estándar abierto y libre. Existen cinco caracteres especiales en XML: los símbolos menores que, <, mayor que, >, las comillas dobles, ”, el apóstrofe' y el carácter &. Los símbolos mayores que y menor que se usan para delimitar las marcas que dan la estructura al documento (Introducción al XML, 2001).

### **Conclusiones parciales del capítulo**

A partir del desarrollo del presente capítulo se arribó a las siguientes conclusiones:

1. El estudio de los conceptos básicos asociados al objeto de estudio posibilitó la adquisición de una mayor comprensión del problema a resolver.
2. El estudio de las soluciones existentes a nivel internacional arrojó como resultado que las aplicaciones estudiadas no brindan una solución completa al problema a resolver, debido fundamentalmente a incompatibilidades, por lo que se decidió desarrollar el Cliente para dispositivos móviles de los puntos de cargas de cargadores eléctricos.
3. El análisis realizado a las diferentes herramientas y tecnologías, así como de la metodología a utilizar, permitió profundizar los conocimientos necesarios para el desarrollo de la solución a la problemática. La caracterización de la metodología AUP-UCI como guía del proceso de desarrollo de software, permitió estandarizar el ciclo de vida del software, dando cumplimiento a las buenas prácticas que define la actividad productiva para la UCI.

## CAPÍTULO 2: DESCRIPCIÓN DE LA APLICACIÓN MÓVIL PROPUESTA

### Introducción

En el presente capítulo se describen las principales características del sistema propuesto, donde se realizan las actividades de la metodología de desarrollo de *software* AUP en su versión UCI, en concordancia con el escenario escogido, así como los diferentes artefactos generados en cada una de ellas. Se definen los requisitos funcionales y no funcionales, se caracteriza la arquitectura de software y los estándares de codificación como buenas prácticas para la implementación de la aplicación móvil propuesta.

### 2.1 Descripción de la propuesta de solución

La aplicación móvil que se pretende desarrollar es una propuesta para el uso de cargadores eléctricos, con el fin de contribuir al mejoramiento del proceso de búsqueda de puntos de cargas y gestión de reservaciones para el uso de estos. Para ello, dicha aplicación contará con las herramientas necesarias para le gestión de reservación, mostrar la información de los conectores, mostrar un listado de los puntos de carga disponibles para el usuario, así como su localización en el mapa. En este sentido les permitirá a los individuos interpretar adecuadamente la información, interactuar y acceder de forma clara y lógica a las secciones de esta aplicación.

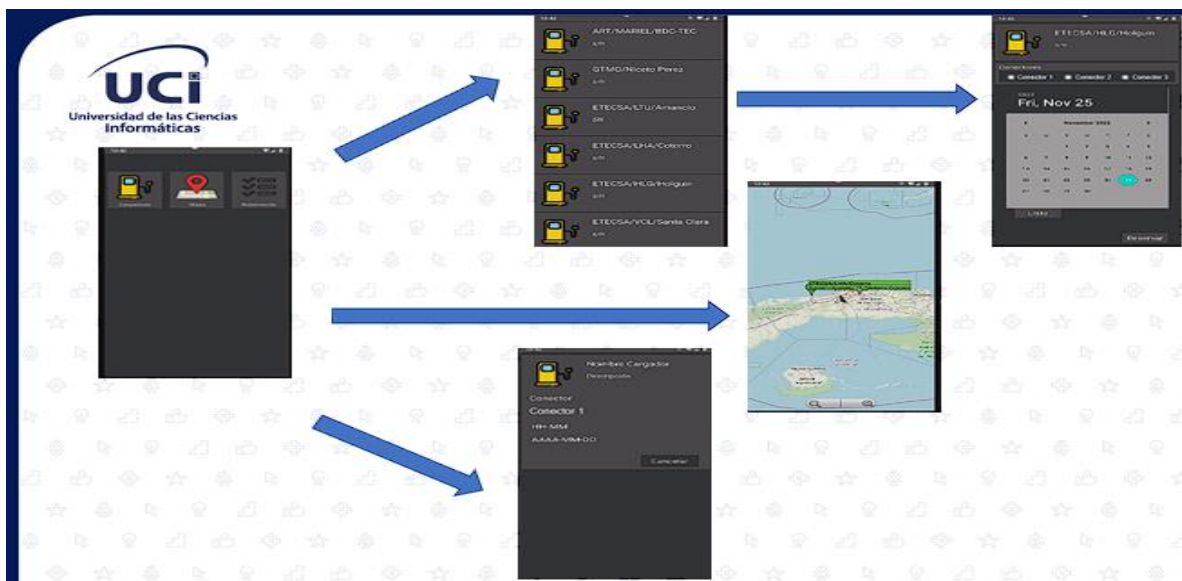


Ilustración 3: Propuesta de solución. (Elaboración Propia)

## 2.2 Requisitos del sistema

Los requisitos para un sistema de software determinan lo que debe hacer el sistema y definen las restricciones en su funcionamiento e implementación, es decir, lo que el software debe hacer y bajo qué circunstancias debe hacerlo. Los requerimientos definen qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Estos requerimientos junto a una interfaz de usuario definen el ámbito de un sistema.

### 2.2.1 Definición de los requisitos funcionales

Tabla 1: Descripción de los RF para el desarrollo de la aplicación móvil

Número	Nombre	Descripción
RF1	Localizar la existencia de puntos de carga en una ubicación determinada (Provincia, Municipio).	Permite mostrar la existencia de puntos de cargas eléctricos mostrados en el mapa (nombre, dirección).
RF2	Filtrar la información de los puntos de carga de acuerdo a disponibilidad	Permite mostrar la información deseada por el usuario (estado).
RF3	Mostrar la dirección de la ubicación del punto de carga.	Permite mostrar la dirección de la ubicación(dirección).
RF4	Mostrar la cantidad de conectores de un punto de carga.	Permite mostrar la cantidad de conectores de cada punto de carga(estado).
RF5	Reservar un conector de un punto de carga.	Permite reservar un conector de un punto de carga.

RF6	Cancelar reservación automáticamente luego de pasado el tiempo de iniciar el proceso de carga del vehículo.	Permite la cancelación automáticamente pasado el tiempo.
RF7	Detener proceso de carga.	Permite detener el proceso de carga de manera manual.
RF8	Mostrar ubicación del punto de carga reservado en un mapa.	Permite ver la ubicación de un punto de carga reservado en un mapa

### 2.2.2 Requisitos no funcionales del sistema

Los requisitos no funcionales son aquellos que están dirigidos a las propiedades emergentes del sistema como son la fiabilidad y seguridad, para tan solo citar algunos ejemplos. Estos requisitos no solo se refieren al software que se desea desarrollar, también existen algunos que restringen el proceso que se debe utilizar para el desarrollo de dicho sistema (Sommerville, 2005).’

#### Espacio

**RNF1:** El dispositivo móvil debe contar con un mínimo de 500mb megas de capacidad interna para almacenar la instalación y datos del mapa.

#### Eficiencia

**RNF2:** En el cliente se requiere un teléfono inteligente con 2048 MB de RAM como mínimo, procesador a 1.0 GHz de velocidad de procesamiento o superior y un adaptador *WiFi*.

#### Software

**RNF3:** El dispositivo debe contar con el sistema *Android* versión 7.0 o superior.

Usabilidad

#### Usabilidad

**RNF4:** Las interfaces deben poseer un diseño sencillo con pocas entradas, permitiendo un balance adecuado entre funcionalidad y simplicidad.

**RNF5:** En el sistema se deben visualizar todos los mensajes en idioma español.

### 2.3 Descripción de Historias de Usuarios

En la presente investigación se definieron dos historias de usuarios, las que representan los requisitos funcionales críticos identificados para la solución propuesta. A continuación, se presentan las tablas.

*Tabla 2: Localizar la existencia de puntos de carga en una ubicación determinada*

<b>Número:</b> HU 1	<b>Nombre del requisito:</b> Localizar la existencia de puntos de carga en una ubicación determinada (Provincia, Municipio).
<b>Programador:</b> Dairon Hernández Blanco	<b>Iteración asignada:</b> 1
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 72 horas
<b>Riesgo en desarrollo:</b> Medio	<b>Tiempo real:</b> 1 semana
<b>Descripción:</b> El usuario deberá poder ver donde se encuentra el punto de carga determinado mostrando provincia y municipio en un cuadro adicional.	

**Observaciones: NA.**

**Prototipo:**



Tabla 3: Reservar un conector de un punto de carga.

<b>Número:</b> HU 5	<b>Nombre del requisito:</b> Reservar un conector de un punto de carga.
<b>Programador:</b> Dairon Hernández Blanco	<b>Iteración asignada:</b> 2
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 72 horas


<b>Riesgo en desarrollo:</b>  Alta	<b>Tiempo real:</b> 1 semana
<b>Descripción:</b> El usuario deberá poder reservar un conector de un punto determinado teniendo en cuenta el tipo de conector que va a usar, hora y fecha de su reservación, así como el nombre del punto de carga.	
<b>Botón:</b>  Listo Fecha: botón que permite reservar escogiendo una fecha y envía un toast con la fecha reservada.  Listo Tiempo: botón que permite reservar escogiendo una hora: minutos y envía un toast con la hora reservada.  Reservar: Botón que crea la reservación  Lista dinámica: con los conectores y su estado.  Casillas de verificación: Seleccionar conector.	
<b>Observaciones: NA.</b>	
<b>Prototipo:</b> 	



Tabla 4: Filtrar la información de los puntos de carga de acuerdo a disponibilidad.


<b>Número:</b> HU 2	<b>Nombre del requisito:</b> Filtrar la información de los puntos de carga de acuerdo a disponibilidad
<b>Programador:</b> Dairon Hernández Blanco	<b>Iteración asignada:</b> 2
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 72 horas
<b>Riesgo en desarrollo:</b> Alta	<b>Tiempo real:</b> 1 semana
<p><b>Descripción:</b> El usuario deberá poder ver todos los puntos de carga disponibles</p> <p>Tarjetas: Con la información de los cargadores en cuanto a Nombre y descripción.</p>	
<b>Observaciones:</b> NA.	
<p><b>Prototipo:</b></p>  <p>The screenshot shows a list of six charging stations on a mobile device. Each entry consists of a yellow charging station icon, a location name, and the status 's/n'. The locations are: ART/MARIEL/BDC-TEC, GTMO/Niceto Perez, ETECSA/LTU/Amancio, ETECSA/LHA/Cotterro, ETECSA/HLG/Holguin, and ETECSA/VCL/Santa Clara.</p>	

Tabla 5: Mostrar la dirección de la ubicación del punto de carga.

<b>Número:</b> HU 3	<b>Nombre del requisito:</b> Mostrar la dirección de la ubicación del punto de carga.
<b>Programador:</b> Dairon Hernández Blanco	<b>Iteración asignada:</b> 2
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 72 horas
<b>Riesgo en desarrollo:</b> Alta	<b>Tiempo real:</b> 1 semana
<p><b>Descripción:</b> El usuario deberá poder ver la dirección de la ubicación del punto de carga en el mapa y su dirección con un icono.</p> <p><b>Botón:</b></p> <p>Aumentar Zoom: Acerca el mapa.</p> <p>Alejar Zoom: Aleja el mapa</p>	
<b>Observaciones:</b> NA.	
<b>Prototipo:</b>	

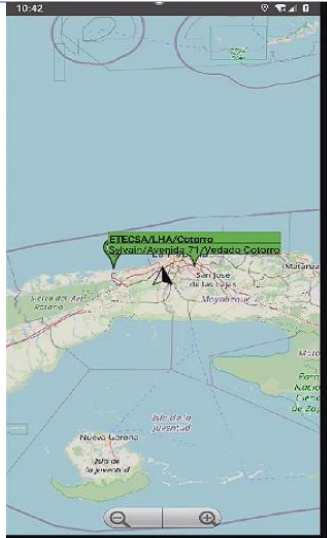


Tabla 6: Mostrar la cantidad de conectores de un punto de carga.

<b>Número:</b> HU 4	<b>Nombre del requisito:</b> Mostrar la cantidad de conectores de un punto de carga.
<b>Programador:</b> Dairon Hernández Blanco	<b>Iteración asignada:</b> 2
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 72 horas
<b>Riesgo en desarrollo:</b> Alta	<b>Tiempo real:</b> 1 semana
<p><b>Descripción:</b> El usuario deberá poder monitorizar la cantidad de cargadores que posea un punto de carga y su estado.</p> <p>Lista Dinámica: Con los conectores y su estado.</p> <p>Casillas de verificación: Seleccionar conector.</p>	

**Observaciones: NA.**

**Prototipo:**



Tabla 7: Cancelar reservación automáticamente luego de pasado el tiempo de iniciar el proceso de carga del vehículo.

<b>Número:</b> HU 6	<b>Nombre del requisito:</b> Cancelar reservación automáticamente luego de pasado el tiempo de iniciar el proceso de carga del vehículo.
<b>Programador:</b> Dairon Hernández Blanco	<b>Iteración asignada:</b> 2
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 72 horas
<b>Riesgo en desarrollo:</b> Alta	<b>Tiempo real:</b> 1 semana

**Descripción:** Se cancelará la reservación automáticamente luego de pasado el tiempo de iniciar el proceso si no se encuentra usándose.

**Observaciones:** NA.

Tabla 8: Detener proceso de carga.

<b>Número:</b> HU 7	<b>Nombre del requisito:</b> Detener proceso de carga.
<b>Programador:</b> Dairon Hernández Blanco	<b>Iteración asignada:</b> 1
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 48 horas
<b>Riesgo en desarrollo:</b> Alta	<b>Tiempo real:</b> 1 semana
<b>Descripción:</b> El usuario deberá poder detener su proceso de carga y ver la información de el cargador, descripción, conector, hora y fecha reservado.	
<b>Botón:</b> Cancelar: Cancela la reservación.	
<b>Observaciones:</b> NA.	
<b>Prototipo:</b>	

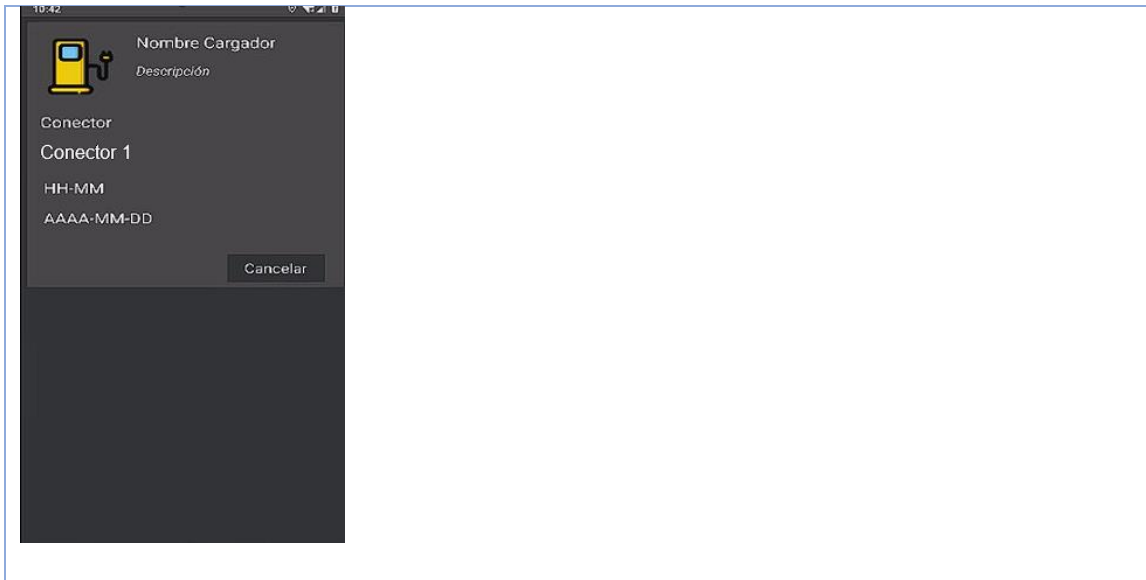


Tabla 9: Mostrar ubicación del punto de carga reservado en un mapa.

<b>Número:</b> HU 8	<b>Nombre del requisito:</b> Mostrar ubicación del punto de carga reservado en un mapa.
<b>Programador:</b> Dairon Hernández Blanco	<b>Iteración asignada:</b> 1
<b>Prioridad:</b> Alta	<b>Tiempo estimado:</b> 72 horas
<b>Riesgo en desarrollo:</b> Alta	<b>Tiempo real:</b> 1 semana
<b>Descripción:</b> El usuario deberá poder ver en el mapa su punto de carga reservado de forma diferente al resto.	
<b>Observaciones:</b> NA.	

### 2.4 Diseño Arquitectónico

La arquitectura es una representación que permite analizar la efectividad del diseño para cumplir con los requerimientos establecidos, considerar alternativas arquitectónicas en una etapa en la que hacer cambios al diseño todavía es relativamente fácil y reducir los riesgos asociados con la construcción del software. Es decir, es la estructura del sistema, lo que comprende a los componentes del software, sus propiedades externas visibles y las relaciones entre ellos (Pressman, 2010).

Teniendo en cuenta la arquitectura antes descrita, en la presente investigación se define utilizar como patrón arquitectónico Modelo Vista Presentador (*MVP*), *que a continuación se detalla.*

#### 2.4.1 Patrón arquitectónico Modelo Vista Presentador

Según el estándar IEEE 1471-2000, se define como la organización fundamental de un sistema incorporando sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. Un patrón de arquitectura busca solucionar problemas de adaptabilidad de requerimientos, rendimiento, modularidad y acoplamiento de los componentes de una aplicación. Los patrones de arquitectura, dentro o entre los niveles arquitectónicos, están relacionados con la interacción de los objetos. Los patrones de arquitectura, al igual que los patrones de diseño, representan llamadas entre objetos, decisiones y criterios arquitectónicos, así como la manera de empaquetado de funcionalidades de una aplicación. Sin embargo, los patrones de arquitectura representan un nivel más alto en el sistema (Tipificación de Dominios de Requerimientos para la Aplicación de Patrones Arquitectónicos, 2016).

El Modelo Vista Presentador es otro patrón de diseño que tiene como objetivo separar la interfaz de usuario de la lógica de las aplicaciones (Serrano, 2013).

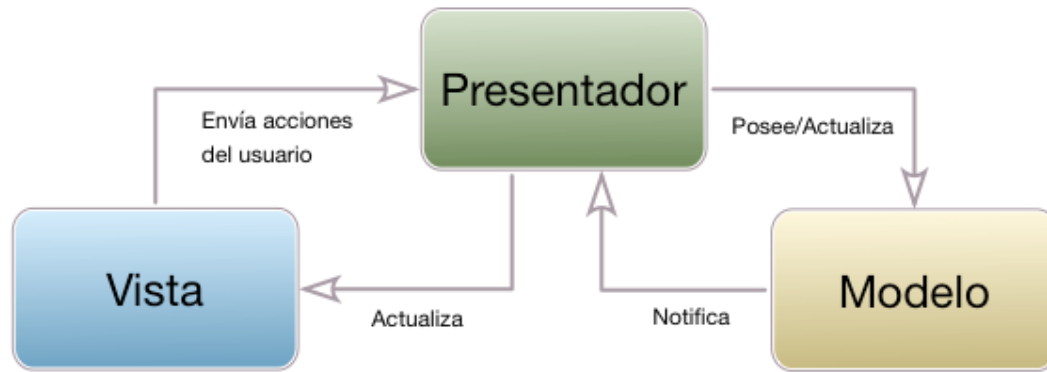


Ilustración 4: Patrón arquitectónico Modelo Vista Presentador

Básicamente este patrón consiste en 3 componentes (Serrano, 2013):

- La vista. Compuesta de las ventanas y controles que forman la interfaz de usuario de la aplicación.
- El modelo. Que es donde se lleva a cabo toda la lógica de negocio.
- El presentador. Escucha los eventos que se producen en la vista y ejecuta las acciones necesarias a través del modelo. Además, puede tener acceso a las vistas a través de las interfaces que la vista debe implementar.

El concepto de este patrón es bastante sencillo. Por un lado, está la vista, que se encarga de mostrar la información al usuario y de interactuar con él para hacer ciertas operaciones. Por otro lado, tenemos el modelo que, ignorante de cómo la información es mostrada al usuario, realiza toda la lógica de las aplicaciones usando las entidades del dominio. Y por último tenemos al presentador que es el que “presenta” a ambos actores sin que haya ningún tipo de dependencia entre ellos. El propósito del presentador tal y como se ha definido en este patrón no establece de manera clara el grado de control que este puede hacer de la vista.

### 2.5 Patrones de diseño

Durante el diseño Orientado a Objetos es frecuente encontrarse repetidamente con ciertos tipos de pruebas, para analizar, compartir y documentar el conocimiento sobre dichos tipos de problemas se han desarrollado los patrones de diseño. Dichos patrones también se aplican en otras partes del desarrollo (análisis) y es recomendable usarlos en cualquier momento para aprovechar el conocimiento y la



experiencia existente y no a partir de cero. Los patrones constituyen la solución base ideal o por lo menos la más recomendable a un problema concreto (Gamma, Helm, Johnson , & Vlissides, 2003).

Un patrón es un conjunto de información que proporciona respuestas a un conjunto de problemas similares. Ayudan a capturar conocimiento y a crear un vocabulario técnico, hacen el diseño orientado a objetos más flexibles, elegantes y en algunos casos reusable. Los patrones pueden relacionarse y aplicarse de forma simultánea en muchos casos (Gamma, Helm, Johnson , & Vlissides, 2003). Existen 2 grandes estructuras fundamentales para agrupar dichos patrones, Patrones de Software para Asignar Responsabilidades GRASP (por sus siglas en inglés *General Responsibility Assignment Software Patterns*) y los Patrones del Grupo de los Cuatro GoF (por sus siglas en inglés *Gang of Four*). A continuación, se definen los patrones de diseño utilizado durante el desarrollo de la aplicación móvil propuesta.

### 2.5.1 Patrones de Software para Asignar Responsabilidades

Los Patrones de Software para Asignar Responsabilidades describen los principios del diseño de objetos y la asignación de responsabilidades, expresados como patrones (Larman, 2003). A continuación, se muestran los patrones GRASP utilizados en la propuesta de solución:

**Patrón Controlador:** sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado (Larman, 2003). Este patrón se evidencia en la solución a través de las clases controladoras.

```

public class MainActivity extends AppCompatActivity {
    private ConstraintLayout chargerListBtn, mapBtn, reservationBtn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        chargerListBtn=findViewById(R.id.cl_chargerlist);
        mapBtn=findViewById(R.id.cl_map);
        reservationBtn=findViewById(R.id.cl_todoList);

        chargerListBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent( packageContext: MainActivity.this, ListChargerActivity.class));
            }
        });
    }
}

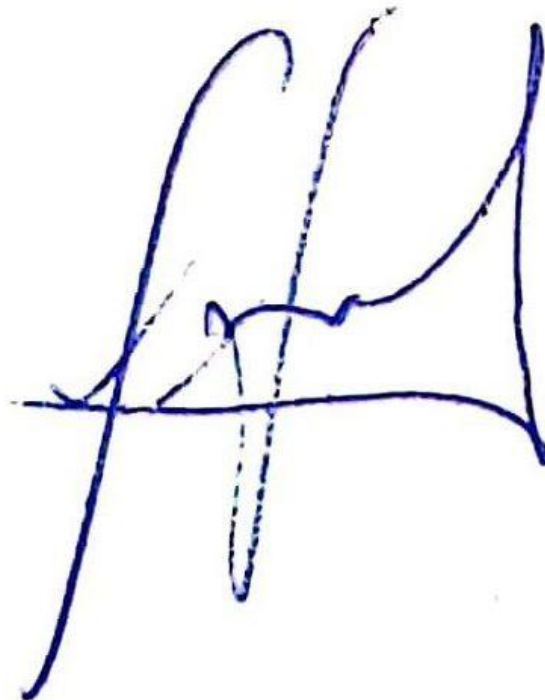
```

Ilustración 5: Patrón Creador (Elaboración Propia)

**Patrón Creador:** consiste en asignarle a una clase la responsabilidad de crear objetos de otra clase. Este patrón es adaptable a las clases controladoras, siendo estas las responsables de crear instancias de diferentes clases (Larman, 2003). Este patrón se evidencia en la solución a través de los métodos *onCreate()* de cada una de las controladoras que engloban la solución, tal como se muestra en la

**Ilustración**

**5.**







**Patrón Alta cohesión:** La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realizan un trabajo enorme.

**Patrón Experto:** El GRASP de experto en información es el principio básico de asignación de responsabilidades. Nos indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento).

**Patrón Bajo acoplamiento:** este patrón se basa en la baja dependencia que debe existir entre las clases, además está estrechamente relacionado con el patrón Alta Cohesión, puesto que a cada clase se le asignan solamente las responsabilidades necesarias de manera que no dependan en gran medida de otras (Larman, 2003).

### 2.5.2 Patrones GoF

Patrones publicados por Gamma, Helm, Johnson y Vlossodes en 1995: patrones de la pandilla de los cuatro (*Gang of Four*). Esta serie de patrones permiten ampliar el lenguaje, aprender nuevos estilos de diseño y además se introducirá más notación (Gamma, Helm, Johnson , & Vlissides, 2003). A continuación, se muestran los patrones utilizados en la propuesta de solución:

**Patrón Decorador:** está diseñado para solucionar problemas donde la jerarquía con subclasificación no puede ser aplicada, o se requiere de un gran impacto en todas las clases de la jerarquía con el fin de poder lograr el comportamiento esperado. Permite al usuario añadir nuevas funcionalidades a un objeto existente sin alterar

su estructura, mediante la adición de nuevas clases que envuelven a la anterior dándole funcionamiento extra (Iturralde, 2016) a continuación en la ilustración 5.

```
import ...

public class ConnectorAdapter extends RecyclerView.Adapter<ChargePointAdapter.ViewHolder> {

    public ConnectorAdapter(ArrayList<Conector> conectores) {

    }

    @NonNull
    @Override
    public ChargePointAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return null;
    }

    @Override
    public void onBindViewHolder(@NonNull ChargePointAdapter.ViewHolder holder, int position) {
```

Ilustración 6 Patrón Decorador. (Elaboración Propia)

**Patrón Observador:** permite observar los cambios producidos por un objeto. Es uno de los principales patrones de diseño utilizados en interfaces gráficas de usuario, ya que permite desacoplar al componente gráfico de la acción a realizar y actualizar la vista (Larman, 2003). Este patrón es empleado al enviar los datos de la vista al presentador, este último hace la petición de un servicio y se queda a la espera de la respuesta, que este servicio debe proveer para así emitir o no un evento. En la ilustración 6 se muestra un ejemplo.

```
void bindData (final ListElement item){

    iconImage.setColorFilter(Color.parseColor(item.getColor()));
    name.setText(item.getName());
    provincia.setText(item.getCity());
    municipio.setText(item.getCity2());
    status.setText(item.getStatus());

}

}
```

Ilustración 7: Patrón Observador. (Elaboración Propia)

### 2.6 Estándares de Codificación

El objetivo de los estándares de codificación de software es inculcar prácticas de programación probadas que conduzcan a un código seguro, confiable, comprobable

y mantenible. Por lo general, esto significa evitar las prácticas de codificación inseguras conocidas o el código que puede causar un comportamiento impredecible (Hamilton, 2020). Para el desarrollo de la investigación se siguieron los estándares presentes en el sitio oficial del lenguaje Dart (dart.dev) entre los cuales se puede mencionar (Dart, 2022). Ejemplo en la siguiente ilustración.

### Identificadores:

- *lowerCamelCase*: colocar mayúscula a la primera letra de cada palabra, excepto la primera que siempre va en minúscula, aunque sea una sigla.
- *lowercase\_with\_underscores*: los nombres usan solo letras minúsculas, incluso para los acrónimos, y separan las palabras con `_`.

```
public class ChargerPoint implements Serializable {
    private String ID;
    private String name;
    private String description;
    // private ArrayList<Conector> connector;

    public ChargerPoint(String ID, String name, String description, ArrayList<Conector> connector) {
        this.ID = ID;
        this.name = name;
        this.description = description;
        // this.connector = connector;
    }
}
```

Ilustración 8: Identificadores. (Elaboración Propia)

### Orden:

Para mantener ordenado el preámbulo de su archivo, debe tener un orden prescrito en el que deben aparecer las directivas. Cada "sección" debe estar separada por una línea en blanco.

### Uso de llaves:

Uso de llaves para todas las estructuras de control de flujo.

### Comentarios:

- Puede usarse un comentario de bloque (`/*...*/`) para comentar temporalmente una sección de código, pero todos los demás comentarios deben usar `//`.

- Para darle formato a los comentarios como oraciones se usa mayúscula en la primera palabra a menos que sea un identificador que distingue entre mayúsculas y minúsculas terminando con un punto (o “!” o “?”, supongo).
- Un comentario de documento es cualquier comentario que aparece antes de una declaración y usa la sintaxis especial ///.

### **Conclusiones parciales del capítulo**

A partir del desarrollo del presente capítulo se arribó a las siguientes conclusiones:

1. Se identificaron los requerimientos de la aplicación, para un total de 5 requisitos no funcionales y 8 requisitos funcionales. Se realizó la descripción textual de cada uno de ellos, lo cual permitió comprender mejor las funcionalidades de la aplicación que se desea desarrollar, el comportamiento de la misma y la secuencia de actividades que debe de realizar el usuario para lograr su objetivo.
2. La especificación de los requisitos que engloban la aplicación propuesta, permitió obtener toda la información relativa a la propuesta de solución y encapsular los requisitos funcionales a través de las Historias de usuario para una mejor comprensión de estos por el equipo de desarrollo.
3. El estudio del diseño arquitectónico partiendo de la arquitectura de software y los patrones de diseño a utilizar, facilitó aislarse del problema en el que se enmarca la presente investigación, obteniéndose así una solución con poca dependencia entre clases, flexible al mantenimiento y a la introducción de cambios.
4. Como resultado principal de este capítulo quedó plasmada la propuesta de solución para la problemática planteada, detallada a través de los requisitos funcionales y no funcionales, las historias de usuario, la arquitectura y el uso de patrones de diseño en el diagrama de clases del diseño.





### CAPÍTULO 3: VALIDACIÓN DE LA APLICACIÓN MÓVIL PROPUESTA

#### Introducción

En este capítulo se muestran los resultados obtenidos durante la validación y verificación del desarrollo de la aplicación móvil para el uso de puntos de cargas eléctricos por vehículos eléctricos e híbridos. Inicialmente se parte de la validación los requisitos. Luego se verifica el correcto funcionamiento de dicha aplicación móvil, a través de las pruebas de software para garantizar la calidad del mismo.

#### 3.1 Pruebas

Como seres humanos al desarrollar cualquier software es normal que cometamos errores que puede llevar a defectos en las aplicaciones. Es por eso que son tan necesarias las pruebas de software. Estas se pueden definir como el conjunto de procesos con los cuales se pretende analizar un sistema o aplicación con el propósito comprobar su correcta marcha. Las mismas pueden realizarse en diferentes momentos del desarrollo de la aplicación, desde su creación hasta su puesta en producción. Estas se pueden ejecutar de manera instintiva, determinando en cualquier momento si tenemos una aplicación estable o, por ejemplo, si un cambio realizado en alguna parte ha afectado a otras partes sin que nos hubiésemos dado cuenta. (Turrado, 2020)

##### 3.1.1 Niveles de prueba

Para aplicarle pruebas a un software es necesario conocer que existen diferentes niveles de prueba, los cuales no son más que actividades de prueba que se organizan y se gestionan en conjunto. Estas actividades son realizadas de acuerdo al nivel de desarrollo en que se encuentre el producto. (Gomez, 2020)

Estos niveles son:

1. Pruebas de componente o unitarias: estas pruebas están enfocadas en los componentes, unidades o módulos, o lo que es lo mismo, los elementos más pequeños del software. Son realizadas generalmente por el desarrollador y de forma

automatizada. Las mismas se encargan de verificar que los comportamientos funcionales y no funcionales del componente son los diseñados y especificados. Además, buscan encontrar defectos y a su vez prevenir la propagación de los mismos en otros niveles de prueba.

**2. Pruebas de integración:** estas pruebas están enfocadas en cómo interactúan los distintos componentes o sistemas entre sí. También buscan encontrar defectos, los cuales pueden estar en las interfaces o en los componentes o sistemas. Pueden ser realizadas tanto por los desarrolladores (generalmente cuando es una prueba de integración de componentes) o por los probadores (generalmente cuando es una prueba de integración entre sistemas).

**3. Pruebas de sistema:** estas pruebas se centran en el comportamiento y las capacidades de los sistemas o productos. Entre sus propósitos están: la reducción de riesgos, verificar que los requerimientos funcionales y no funcionales del sistema sean cumplidos y además validar que el sistema funciona como se espera, generando así confianza en la calidad del sistema. Son realizadas por los probadores.

**4. Pruebas de aceptación:** estas pruebas al igual que las de sistema, están enfocadas en el comportamiento de todo el sistema o producto. Tienen como propósito validar que el sistema está completo, que funcionará según se espera, verificar que los comportamientos funcionales y no funcionales son los que se especificaron y además proveer información que permita decidir si el sistema está óptimo o no para salir a producción. En este tipo de pruebas el usuario juega un papel muy importante.

### **3.1.2 Métodos de prueba**

Para poder obtener un buen diseño de casos de prueba con buenas probabilidades de descubrir los defectos del software es necesario emplear métodos de prueba. (Álvarez N. S., 2019) Estos métodos de prueba son:

- **Pruebas de caja negra:** este método de prueba está centrado en los requisitos funcionales del software. Los casos de prueba que diseña pretenden demostrar que

las funcionalidades del software son operativas y que la entrada se acepta de forma adecuada, produciéndose además una salida correcta.

- Pruebas de caja blanca: este es un método de diseño de casos de prueba dirigidas a las funciones internas y a la evaluación del código. Usa la estructura de control del diseño procedimental para así derivar los casos de prueba.

### 3.2 Pruebas funcionales

Las pruebas funcionales son muy importantes ya que están enfocadas principalmente en verificar que las funcionalidades desarrolladas en el sistema cumplan con sus especificaciones, por lo que están basadas en los requisitos funcionales. (Larrea, 2019) Estos tipos de pruebas incluyen pruebas unitarias, pruebas de regresión, pruebas de aceptación del usuario, además de muchas otras. Las pruebas funcionales se pueden realizar además usando técnicas de Caja Negra, las cuales realizan pruebas sobre la interfaz del programa a probar, es decir, sobre las entradas y salidas de dicho programa. Para validar las funcionalidades de nuestra aplicación se decidió realizarle dos tipos de pruebas fundamentales: prueba de Caja Negra mediante la técnica de partición de equivalencia y las pruebas unitarias.

#### 3.2.1 Prueba de Caja Negra mediante la técnica de partición de equivalencia

Para la realización de este tipo de prueba se han diseñado casos de prueba. Estos miden la funcionalidad en un conjunto de acciones o condiciones para verificar el resultado esperado. Para ello hace falta un número de datos que ayuden a la ejecución de estos casos, pueden ser datos válidos o inválidos para el programa, eso depende de lo que se desea hallar: un error o probar una funcionalidad. Para el diseño de los mismos se utilizará la técnica de partición de equivalencia. Esta técnica es un enfoque efectivo para las pruebas, la cual ayuda en la explicación de los errores que cometen con frecuencia los programadores al procesar entradas en los bordes de las particiones. Es además aplicable a entradas de datos realizadas por personas o vía interfaces con otros sistemas. (Sommerville, 2011) A continuación se expone el caso de prueba realizado al requisito:

Tabla 10: Caso de Prueba Filtrar la información de los puntos de carga de acuerdo a disponibilidad

Sección: Filtrar la información de los puntos de carga de acuerdo a disponibilidad			
Id	Escenario	Respuesta del Sistema	Resultado de la prueba
1.1	Conexión con el API-REST con éxito	Muestra el recyclerView con la información de los cargadores eléctricos	Se muestra los datos satisfactoriamente al usuario
1.2	Conexión con el API-REST fracasada	Mensaje Toast: “Fallo de Conexión”	Se alertó satisfactoriamente

Tabla 11: Caso de Prueba Mostrar la dirección de la ubicación del punto de carga

Sección: Mostrar la dirección de la ubicación del punto de carga.			
Id	Escenario	Respuesta del Sistema	Resultado de la prueba
1.1	Mapa Osmdroid	Mostrar la dirección de un punto de carga dando click en este	Se muestra los datos satisfactoriamente al usuario

### 3.2.2 Pruebas de Aceptación

La prueba de software es el proceso de evaluación y verificación de un producto o aplicación de software para saber si hace lo que se supone que debe hacer. Los beneficios de las pruebas incluyen la prevención de errores, la reducción de los costos de desarrollo y la mejora del rendimiento.

Incluso una aplicación simple puede estar sujeta a una gran cantidad y variedad de pruebas. Un plan de gestión de pruebas ayuda a priorizar qué tipos de pruebas proporcionan el mayor valor, dado el tiempo y los recursos disponibles. La eficacia de las pruebas se optimiza ejecutando la menor cantidad de pruebas para encontrar la mayor cantidad de defectos. (IBM, 2022)

### Pruebas Alfa y Beta

Son las metodologías de validación de clientes (tipos de pruebas de aceptación) que ayudan a generar confianza para lanzar el producto y, por lo tanto, resultan en el éxito del producto en el mercado.

Aunque ambos se basan en usuarios reales y en diferentes comentarios del equipo, están impulsados por distintos procesos, estrategias y objetivos. Estos dos tipos de pruebas en conjunto aumentan el éxito y la vida útil de un producto en el mercado.

Se centran principalmente en descubrir los errores de un producto ya probado y dan una imagen clara de cómo los usuarios en tiempo real utilizan realmente el producto. También ayudan a ganar experiencia con el producto antes de su lanzamiento y los valiosos comentarios se implementan de manera efectiva para aumentar la usabilidad del producto. (myservname.com, 2021)

**Pruebas Alfa:** Es la primera versión del programa, la cual es enviada a los verificadores para probarla. Algunos equipos de desarrollo utilizan el término alfa informalmente para referirse a una fase donde un producto todavía es inestable, aguarda todavía a que se eliminen los errores o a la puesta en práctica completa de toda su funcionalidad, pero satisface la mayoría de los requisitos.

**Pruebas Beta:** Representa generalmente la primera versión completa del producto, que es posible que sea inestable pero útil para que las demostraciones internas y las inspecciones previas seleccionen a clientes. Esta etapa comienza a menudo cuando los desarrolladores anuncian una congelación de las características del producto, indicando que no serán agregadas más características a esta versión y que solamente se harán pequeñas ediciones o se corregirán errores. Muchos de estos programas beta, son de uso privado solo permitiendo a un número determinado de usuarios probarlo, y de esta manera mantener un control más eficiente de la calidad y las opiniones de los usuarios que lo están probando.

Este tipo de programas casi siempre incluye instrucciones específicas para reportar estos bugs y recibir ayuda en caso de ser necesario.

Para la realización de estas pruebas se tomaron en cuenta personas, que han desempeñado algún rol en el desarrollo de aplicaciones android y personas ajenas al desarrollo para que ejecutaran las pruebas en busca de errores u inconformidades:

- Ing. Frank Geiler Vega Duverger
- Dairon Hernández Blanco
- Samira Enriquez González
- Dionis Pentón García
- Yoan Delgado Antón

Para la fase de pruebas se ejecutaron 2 iteraciones de las mismas, las cuales arrojaron una cantidad determinadas de errores y no conformidades, las cuales se fueron corrigiendo antes de avanzar a la siguiente iteración; para la última repetición no se encontraron errores o no conformidades por parte de los equipos que realizaron las pruebas.

Como se muestra en la siguiente ilustración 8, durante la 1ra iteración la tenencia de errores fue de “Opciones que no funcionan”, ya que al acceder a 4 de las opciones de la aplicación móvil, estas no mostraron la información solicitada. Además, se detectaron 2 faltas de ortografías. En la primera iteracion se corrigieron las NC detectadas, lo que trajo consigo que en la segunda iteración no se detectaron NC, demostrando así la calidad de los componentes propuestos, y a su vez, el cumplimiento de los requisitos acordados previamente con el cliente.

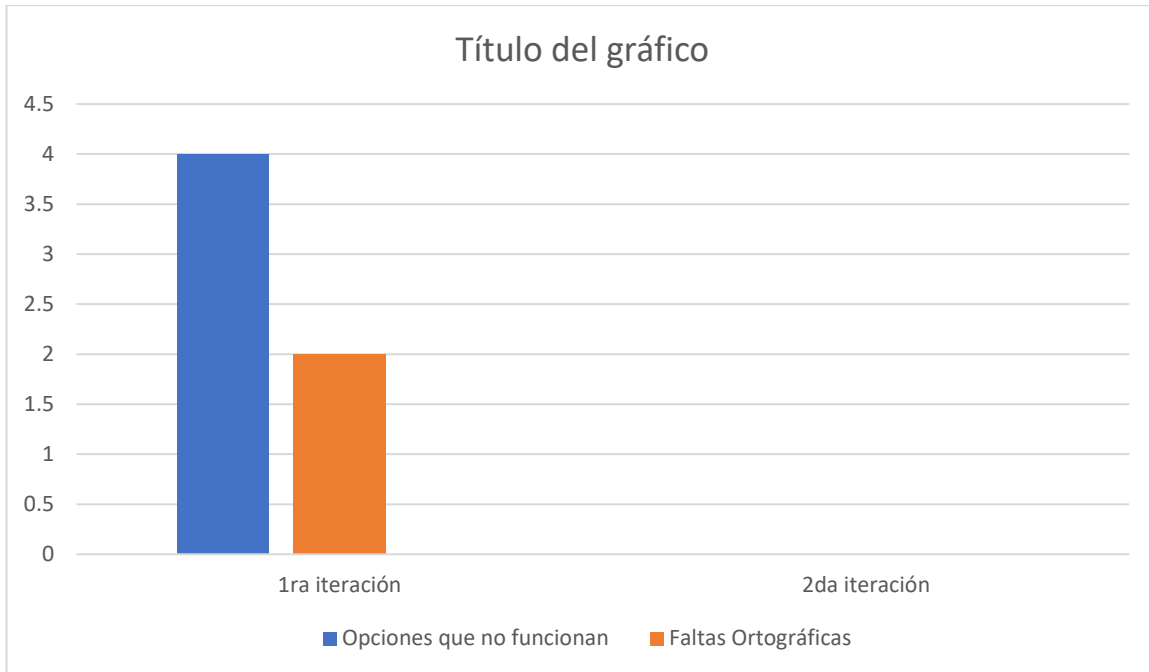


Ilustración 9: Resumen de las no conformidades encontradas (Elaboración Propia)

### Conclusiones del capítulo

El desarrollo de la presente investigación permitió arribar a las siguientes conclusiones:

- Se comprobó cuantitativamente la calidad de los artefactos obtenidos a lo largo de la investigación mediante la aplicación de las técnicas de validación y verificación desarrolladas en la fase de pruebas.
- Se logró la implementación de una aplicación android en fase beta que sirva como enriquecimiento a la red de cargadores eléctricos.

### **CONCLUSIONES GENERALES**

El desarrollo de la presente investigación permitió arribar a las siguientes conclusiones:

- La construcción del marco teórico referencial de la investigación, relacionado con el desarrollo de aplicaciones móviles de apoyo, permitió obtener una comprensión total del presente trabajo de diploma brindando así soporte teórico-metodológico y conceptual al mismo.
- La caracterización de la metodología de desarrollo de software, así como las tecnologías que formarán parte del ambiente de desarrollo, permitió establecer las pautas de un proceso disciplinado sobre el desarrollo de software de forma organizada.
- La elaboración de los artefactos ingenieriles como resultado del desarrollo de la aplicación móvil para su uso, permitieron obtener una representación abstracta de la solución.
- La realización de las pruebas de aceptación y las funcionales demostró que la implementación satisface los requisitos propuestos por el cliente, dando cumplimiento al objetivo trazado en la investigación.



### **RECOMENDACIONES**

Tomando como base la investigación realizada y el análisis de los resultados obtenidos se recomienda:

1. Agregar una funcionalidad de pasarela de pago.
2. Continuar trabajando en el perfeccionamiento de la aplicación propuesta, para lograr la actualización constante de todos los puntos de carga, así como su reserva por parte de los usuarios.
3. Divulgar por los medios existentes la existencia y formas de uso de la aplicación propuesta para un posterior uso.

### REFERENCIAS BIBLIOGRÁFICAS

- 1.(s.f.). Obtenido de <https://www.plugshare.com/es>
- 2.(s.f.). Obtenido de <https://standards.ieee.org/ieee/18/4343/>
3. @es, e. (28 de junio de 2022). Mobility @es. Obtenido de <https://circontrol.com/es/vehiculos-electricos-contra-combustion-que-contamina-mas-realmente/>
- 4.apoyo. (2022). Universidad de Alicante. Obtenido de Universidad de Alicante: <https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html>
- 5.Arquitectura del sistema. (s.f.).
- 6.Bacon, F. R. ( 2015). Metodologías de desarrollo de software.
- 7.Bennett-Cohen, J. (20 de Abril de 2022). makeuseof. Obtenido de <https://www.makeuseof.com/best-electric-vehicle-charging-station-apps/>
- 8.Chargehub. (s.f.). Obtenido de Chargehub: <https://chargehub.com/en/>
- 9.Chargeway. (2022). Obtenido de Chargeway: <https://www.chargeway.net>
- 10.Electrify America. (2022). Obtenido de Electrify America: <https://www.electrifyamerica.com>
- 11.GNU. (21 de 10 de 2022). Obtenido de GNU: <https://www.gnu.org/home.es.html>
- 12.Guevara, J. M. (2019). Fundamentos de programación en Java. España: EME . Obtenido de [https://es.wikipedia.org/wiki/Java\\_\(lenguaje\\_de\\_programación\)](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programación))
- 13.Matthew David, K.-C. (2021). computerweekly. Obtenido de computerweekly: <https://www.computerweekly.com/es/definicion/Desarrollo-de-aplicaciones-moviles#:~:text=El%20desarrollo%20de%20aplicaciones%20móviles%20es%20el%20conjunto%20de%20procesos,inteligentes%20y%20otros%20dispositivos%20portátiles.>
- 14.Nielfa, J. S. (2022). Scoreapps. Obtenido de Scoreapps: <https://scoreapps.com/blog/es/android-studio/>
- 15.Cliente para dispositivos móviles de la Tienda de Aplicaciones Android del Caribe  
Lisvany Alonso Martinez. Obtenido de
- 16.Sistema Android de mapas y utilidades de la Universidad Central “Marta Abreu” de Las Villas

## *Referencias Bibliográficas*

17. Adeva, Roberto. 2022. ADSLZone. [En línea] 11 de marzo de 2022. <https://www.adslzone.net/reportajes/software/que-es-android/>.
18. Análisis de la aplicación de la tecnología móvil en las empresas. Usano, Silvia Carrasco. 2015. Valencia : s.n., 2015.
19. Arturo Baz Alfonso, Irene Ferreira Artime, María Álvarez Rodríguez, Rosana García Baniello. 2013. 20. Dispositivos móviles. [En línea] 1 de octubre de 2013. <https://www.clubensayos.com/Tecnolog%C3%ADa/Dispositivos-m%C3%B3viles/1090904.html>.
21. Bali, Eren. 2022. Udeemy, cursos en línea: aprende de todo y a tu propio ritmo. [En línea] 2022. <https://www.udemy.com/>.
22. Bembibre, Victoria. 2009. Definición ABC. [En línea] enero de 2009. <https://www.definicionabc.com/tecnologia/aplicacion.php>.
23. Castillo, José Dimas Luján. 2019. Desarrollo de aplicaciones Android con Android Studio: Conoce Android Studio. 2019.
24. Iturralde, Oscar Javier Blancarte. 2016. Introducción a los patrones de diseño, un enfoque práctico. México : s.n., 2016.
25. Khan, Sal. 2022. Khan Academy | Práctica, lecciones y cursos en línea gratuitos. [En línea] 2022. <https://es.khanacademy.org/>.
26. Larman, Craig. 2003. UML y Patrones. Madrid : s.n., 2003.
- 27.—. 2016. UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. 3ra . s.l. : Prentice-Hall, 2016.
28. Roa, Monica Mena. 2021. Gráfico: Android e iOS dominan el mercado de los smartphones | Statista. [En línea] 30 de agosto de 2021. <https://es.statista.com/grafico/18920/cuota-de-mercado-mundial-de-smartphones-por-sistema-operativo/>.
29. Rodríguez Sánchez, Tamara. 2015. Metodología de desarrollo para la Actividad productiva UCI. La Habana : Universidad de las Ciencias Informáticas, 2015.

## *Referencias Bibliográficas*

- 30.—. 2015. Metodología de desarrollo para la Actividad productiva UCI. La Habana : s.n., 2015.
- 31.Salas, Maribel Zahuantitla. 2013. Definicion e historia de las herramientas CASE - Fundamentos de Ingeniería de Software. [En línea] 2013.
- 32.<https://sites.google.com/site/fundamentosdeingendoftware/u1-fundamentos-ingenieria-de-software/1-5-definicion-e-historia-de-las-herramientas-case>.
- 33.Serrano, Joaquín Medina. 2013. La Gueb de Joaquín. El patrón Modelo-Vista-Presentador (MVP) a examen. [En línea] 26 de noviembre de 2013.
- 34.[http://joaquin.medina.name/web2008/documentos/informatica/documentacion/logica/patrones/patronMVP/2012\\_06\\_09\\_PatronMVP.html](http://joaquin.medina.name/web2008/documentos/informatica/documentacion/logica/patrones/patronMVP/2012_06_09_PatronMVP.html).
- 35.Álvarez, N. S. (2019). Pruebas de mutación, control sobre variaciones en el código fuente. Álvarez, O. M. (28 de marzo de 2022). Periodismo de Barrio. Obtenido de <https://periodismodebarrio.org/2022/03/los-vehiculos-electricos-tienen-futuroen-cuba/>
- 36.Somerville, Ian. 2011. Ingeniería de Software, un enfoque práctico. México : s.n., 2011.
- 37.Teorías del aprendizaje en el contexto educativo. Yolanda Heredia Escorza, Ana Lorena Sánchez Aradillas. 2013. México : Editorial Digital Tecnológico de Monterrey, 2013.
- 38.Tipificación de Dominios de Requerimientos para la Aplicación de Patrones Arquitectónicos. Johanna M. Suárez, Luz E. Gutiérrez. 2016. 4, Colombia : s.n., 2016, Vol. 27.
- 39.Ucha, Florencia. 2022. Definición ABC. [En línea] noviembre de 2022. <https://www.definicionabc.com/tecnologia/sistema-operativo.php>.
- 40.—. 2010. Definición ABC. [En línea] febrero de 2010. <https://www.definicionabc.com/general/cuestionario.php>.
- 41.Vaati, Esther. 2020. Code Envato Tuts+. [En línea] 2 de julio de 2020. <https://code.tutsplus.com/es/tutorials/the-android-sdk-tutorial--cms-34623>.

## *Referencias Bibliográficas*

42. Veronas, Nico. 2012. Pencil Project. [En línea] 2012. <https://www.neoteo.com/pencil-project-crea-prototipos-para-aplicaciones-y/>.

43. Zaldivar, Jose Antonio Baldres. 2020. Desarrollo de una aplicación multiplataforma mediante el framework Flutter e implementación de servicios de autenticación y base de datos mediante Firebase. [En línea] 13 de mayo de 2020. <https://riunet.upv.es/handle/10251/143049>.

44. Turrado, J. (10 de marzo de 2020). campusMVP.es. Obtenido de <https://www.campusmvp.es/recursos/post/que-son-las-pruebas-desoftware.asp>

45. IBM. (2022). Obtenido de IBM: <https://www.ibm.com/cl-es/topics/software-testing>