Facultad 4

Simulador de un punto de carga para la plataforma de carga pública en Cuba

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

Autor: Arianna Correoso Azahares
Tutor: Ing. Adolfo Yasser Santana Rojas

La Habana, 8 de diciembre de 2023
"Año 65 de la Revolución

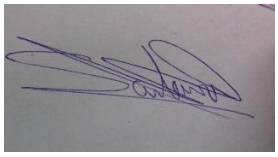
DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título ***"Simulador de un punto de carga para la plataforma de carga pública en Cuba"*****,** concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara único autor de su contenido. Para que así conste firma la presente a los 8 días del mes de diciembre del año 2023.

Arianna Correoso Azahares

_____
Firma del Autor

Ing. Adolfo Yasser Santana Rojas



_____
Firma del Tutor

DATOS DE CONTACTO

Adolfo Yasser Santana. Graduado de ingeniero en Ciencias Informáticas en el año 2015. Especialista del centro VERTEX. Arquitecto de software.

AGRADECIMIENTOS

Tutor: Gracias por su ayuda durante el último semestre. Le agradezco su paciencia, apoyo incondicional por escucharme y ayudarme siempre que se lo he pedido.
Adolfo Yasser Santana Rojas
Profesores: Por su incansable labor, por todos los conocimientos que nos legaron, por su paciencia, por su compromiso.
Andy Suárez Oña, Luis Manuel Valera Pérez
Familia por esta siempre conmigo en incondicionalmente.

DEDICATORIA

Sea este anhelado desenlace un premio a nuestros padres, nuestras familias, nuestros amigos y todos aquellos, que por pequeño que fuera su aporte, hicieron de este día una certeza, especialmente en aquellos momentos donde entre dificultades y penas era difícil vislumbrarlo. Sea homenaje a todos aquellos profesores con los que contrajimos tan fuerte deuda moral por ese tiempo que dedicaron a formar a los profesionales y ciudadanos que somos hoy, alejando con amor cuanto de ignorancia pesaba en nosotros, antes de que la universidad se convirtiera en nuestro hogar.

RESUMEN

El desarrollo de software, hardware, redes de comunicación, seguridad informática aportan a la electromovilidad en múltiples ámbitos, desde la seguridad en la conducción, como en la optimización de rutas, nuevos modelos de servicios y modelos de negocios, entre muchas otras aplicaciones que sin duda surgirán en el futuro. La electromovilidad se ha vuelto una de las opciones para disminuir los contaminantes atmosféricos. Cuba no es ajeno a esta situación por lo que lleva a cabo la electrificación de trasportes públicos por el ministerio de trasporte.

La Universidad de las Ciencias Informáticas tiene como objetivo informatizar la sociedad e industria para ello el Centro de Tecnologías Interactivas (VERTEX) se dio a la tarea de desarrolla una plataforma de electromovilidad, sin embargo, la carencia de cargadores eléctricos físicos imposibilita la validación de la comunicación entre el sistema central y el punto de carga mediante el protocolo OCPP. Este trabajo tiene como objetivo crea un simulador de un punto de carga para dicha plataforma, permitiendo la simulación de la interacción del sistema central y el punto de carga mediante el protocolo.

Palabras clave: electromovilidad, Protocolo OCPP, cargadores eléctricos

ABSTRACT

The development of software, hardware, communication networks, computer security contribute to electromobility in multiple areas, from driving safety to route optimization, new service models and business models, among many other applications that without doubt will arise in the future. Electromobility has become one of the options to reduce atmospheric pollutants. Cuba is no stranger to this situation, which is why it carries out the electrification of public transportation by the Ministry of Transportation. The University of Computer Sciences aims to computerize society and industry. For this purpose, the Center for Interactive Technologies (VERTEX) took on the task of developing an electromobility platform, however, the lack of physical electric chargers makes it impossible to validate the communication between the central system and the charging point using the OCPP protocol. This work aims to create a charging point simulator for said platform, allowing the simulation of the interaction of the central system and the charging point through the protocol.

Keywords: electromobility, OCPP Protocol, electric chargers

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ILUSTRACIÓN

INTRODUCCIÓN

El agotamiento a nivel internacional de los combustibles fósiles y el aumento de la emisión de gases contaminantes a la atmósfera ha propiciado que los fabricantes de vehículos desarrollen nuevas tecnologías en busca de la reducción de los efectos nocivos que provocan los vehículos de combustión interna al medio ambiente, así como de la dependencia de los combustibles tradicionales; conocidos como vehículos eléctricos.

Aunque las emisiones de gases de efecto invernadero en Cuba no son tan elevadas en comparación a otros países, potenciar la movilidad sostenible ha sido una de las estrategias en los últimos años y ya se dan los primeros pasos en ese sentido. Por ejemplo, se venden, aunque a precios poco accesibles para la mayoría de la población, vehículos eléctricos, lo cuales ya circulan por las ciudades. También se comienzan a emplear servicios con transporte eléctrico ya sea para la movilidad de pasajeros en algunas áreas como las cercanías de Fontanar en La Habana o para empresas estatales como ETECSA y Aguas de La Habana (Milanés, 2023).

Dando paso a la movilidad eléctrica o electromovilidad en Cuba, dicho término se refiere al uso de sistemas de impulso o tracción que utilizan energía eléctrica aplicados a distintos medios de transporte (Chile, 2018).

Por la razón antes expuesta en la Universidad de Ciencias Informáticas (UCI) en el Centro de Tecnologías Interactivas (VERTEX) desarrolla una plataforma de electromovilidad, la cual va más allá de la simple recarga de vehículos, incorporando la gestión remota carga de vehículos, servicios de movilidad compartida, análisis de datos, optimización de carga, almacenamiento de energía y la mejora continua de la experiencia del usuario contribuyendo a una red eléctrica más estable lo cual resuelve la necesidad que tiene el país de facilitar a los propietarios de vehículos eléctricos puedan cargarlos fuera del hogar o de las empresas pues tiene pocas estaciones de carga y no tienen control de la comunicación y la integración del sistema central y los puntos de carga.

Dentro del ecosistema de carga pública existe un actor reconocido a nivel global: el operador de un punto de carga, el cual es el encargado de gestionar múltiples puntos de carga y garantiza que todos puedan utilizarse de forma continua para que los conductores de vehículos eléctricos carguen sus coches sin ningún problema. Dichos puntos de carga se comunican e intercambia datos mediante un catalizador con el sistema central: un protocolo abierto para la comunicación de punto de carga.

Sin embargo, la carencia de puntos de carga y la poca interacción con cargadores eléctricos físicos a partir de no tener un escenario real para llevar a cabo la validación de las funcionalidades correspondientes al protocolo de comunicación de punto de carga (OCPP) empleado para la interacción con el

sistema central ha conllevado a un gasto de tiempo, recurso y dinero en el desarrollo de la plataforma de electromovilidad.

Dada la situación anteriormente expuesta se plantea el siguiente:

Problema investigativo: ¿Cómo validar el comportamiento correspondiente del protocolo de comunicación propuesto en la plataforma de electromovilidad?

Para dar solución al problema anterior se propone como **objetivo general:** desarrollar un simulador de un punto de carga mediante el protocolo OCPP.

Se definió como **objeto de estudio**: simulación de un punto de carga en la plataforma de electromovilidad.

Enmarcado en el **campo de acción**: plataforma de electromovilidad para carga pública de Cuba.

Para dar cumplimiento al objetivo propuesto se han derivado un conjunto de objetivos específicos a cumplir:

- Elaborar el marco teórico conceptual relacionado con los aspectos teóricos que sustentan la investigación.
- Realizar el análisis y diseño de la propuesta de solución.
- Desarrollar la propuesta de solución.
- Validar la implementación a través de los métodos definidos en la investigación.

Para obtener los conocimientos necesarios que hagan posible el cumplimiento del objetivo trazado, se lleva a cabo una investigación en las que se utilizan algunos de los métodos científicos existentes:

Métodos teóricos:

Modelación: es empleada en la representación mediante diagramas de las características, procesos y componentes de la solución propuesta, así como la relación existente entre ellos.

Analítico-sintético: se emplearon en el proceso de análisis documental y revisión bibliográfica, con el objetivo de extraer las ideas esenciales que permitirán fundamentar desde el punto de vista teórico la investigación, así como la propuesta que se realiza.

Métodos empíricos:

Estudio documental: en la consulta de la literatura especializada en las temáticas a fines de la investigación.

Observación: sirvió para valorar las tecnologías y productos existentes, además de comparar con sistemas homólogos y así poder definir la propuesta, dando a conocer de manera minuciosa lo que se desea, lo que hace falta realizar y como se realizará.

Estructura de la Tesis

El presente trabajo de diploma está compuesto por tres capítulos, que incluyen los procedimientos desarrollados en relación con el trabajo investigativo, así como la propuesta de solución y validación de la investigación.

Capítulo 1. Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio: se abordan los principales elementos teóricos vinculados a la investigación, se realiza el estudio del estado del arte y finaliza con el análisis de las tecnologías y herramientas a utilizar.

Capítulo 2. Diseño de la solución propuesta: se describe la solución propuesta y las funcionalidades del sistema a desarrollar, así como el diseño arquitectónico y los restantes artefactos relacionados con la etapa de análisis y diseño.

Capítulo 3. Implementación y validación de la solución propuesta: se definen los artefactos relacionados con la etapa de implementación. También se aplican las pruebas con el fin de comprobar el correcto desarrollo de las funcionalidades implementadas.

Al concluir estos tres capítulos, se da el resultado obtenido y las diferentes recomendaciones obtenidas basándose en los resultados de dicha investigación. También se da a conocer la bibliografía de la cual se pudieron ampliar los conocimientos para dicha investigación.

CAPÍTULO 1: FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS SOBRE EL OBJETO DE ESTUDIO

En el presente capítulo se abordan los elementos teóricos relacionados con la simulación de un punto de carga en la plataforma de electromovilidad. Son descritos los elementos fundamentales de la investigación y se reflejan las características esenciales de las diferentes soluciones informáticas existentes en el mercado; así como las tendencias actuales en el desarrollo de este tipo de sistemas y las tecnologías más comúnmente utilizadas. Finalmente, se exponen los elementos más relevantes de la metodología, lenguajes y herramientas definidas para el desarrollo.

1.1 Conceptos asociados a la investigación

A continuación, se presentan un grupo de conceptos que se hacen necesarios para la comprensión de la presente investigación.

1.1.1 Plataforma de electromovilidad

Es un conjunto integrado de tecnologías, infraestructuras y servicios diseñados para respaldar y promover la movilidad eléctrica; es un ecosistema integral que abarca desde la fabricación de vehículos eléctricos hasta la gestión eficiente de la energía y la experiencia del usuario. Va más allá de la simple recarga de vehículos, incorporando la gestión remota de vehículos, servicios de movilidad compartida, análisis de datos, optimización de carga, almacenamiento de energía y la mejora continua de la experiencia del usuario contribuyendo a una red eléctrica más estable (US Energy Solutions, 2023).

Dentro del ecosistema de electromovilidad existen varios actores, a la vez, deben interactuar entre sí para permitir el correcto funcionamiento de la red de carga. Se identifica que la interacción entre el punto de carga y el sistema central es clave para el desarrollo de la interoperabilidad en la electromovilidad. Los puntos de carga deben poder comunicarse, a través de protocolos, con el sistema central.

1.1.2 Simulación de un punto de carga en una plataforma de electromovilidad

Simular se define como representar algo, fingiendo o imitando lo que no es (Real Academia Española, 2023; Real Academia Española, 2023).

La simulación se define como la acción y efecto de simular; también es la alteración aparente de la causa, la índole o el objeto verdadero de un acto o contrato (Real Academia Española, 2023).

La simulación no es un concepto nuevo en el desarrollo de software; siempre se ha buscado la manera de evaluar sistemas complejos; es una herramienta idónea para ensayar, comprender el

funcionamiento de determinados sistemas o anticiparse a problemas. Estos softwares de simulación facilitan conocer qué tipo de respuestas pueden ofrecer determinados sistemas ante diferentes situaciones, sin ningún tipo de riesgo físico ni para los humanos ni para las máquinas, recreando entornos flexibles, ahorrando tiempo y dinero (C, 2018).

El uso de una herramienta de simulación como un punto de carga dentro de una plataforma de electromovilidad es un modelo que permite estudiar el comportamiento y el rendimiento de un sistema central; se utiliza para realizar pruebas y validación del sistema central. Permite simular escenarios de carga, comportamiento de los puntos de carga y la interacción con el sistema central para asegurarse de que el sistema funciona de manera más eficiente y confiable.

1.1.3 Punto de carga

Los puntos de cargas permiten recargar de forma segura cualquier tipo de coches eléctricos o híbridos enchufables. Un punto de recarga es una interfaz capaz de recargar un vehículo eléctrico, asociado a una plaza de aparcamiento. Incluye al menos una toma para enchufe y/o un cable adjunto con conector (LugEnergy., 2023).

1.1.4 Sistema central

El sistema central o sistema de control de carga es la pieza central de una infraestructura de recarga inteligente y sostenible para la carga de vehículos eléctricos. Este posee varias características e interfaces en un *hardware* compacto y modular. Su *software* escalable optimiza el funcionamiento de los postes de carga aumentando su disponibilidad y permitiendo una facturación precisa de los procesos de carga (Contact, 2022).

1.1.5 Protocolo OCPP

Es un protocolo libre/abierto para puntos de recarga o dicho de otra forma, protocolo *open-source* para gestionar, a través de un web manager o sistema de gestión en la nube (*CPMS, Charge Point Management System*), estaciones de recarga de vehículos eléctricos, conocidos con el término electrolineras, cuya función es idéntica al de un surtidor convencional de combustible con la diferencia de que suministra energía eléctrica a vehículos que dispongan de sistema de baterías, ya sean híbridos (motor de combustión + motor eléctrico) o puramente eléctricos (Vazquez, 2022).

OCPP es un protocolo estándar para la comunicación entre puntos de carga y un sistema central donde un operador de punto de carga se encarga de gestionar un conjunto de puntos de carga puede, entre otras cosas, supervisar el estado de los puntos de carga, autorizar a quién se permite cargar o realizar acciones a distancia como detener una transacción en curso (service@wallbox).

El protocolo OCPP es respaldado por la *Open Charge Alliance* (OCA), una organización sin ánimo de lucro que se formó con el propósito de promover la adopción de estándares abiertos para la carga de

vehículos eléctricos. La OCA reunió a diversos actores de la industria, incluyendo fabricantes de estaciones de carga, proveedores de servicios de carga, fabricantes de vehículos eléctricos y otros interesados en el desarrollo de un ecosistema de carga eléctrica más abierto y colaborativo.
En la Ilustración 1, se muestra la evolución del protocolo desde su surgimiento.

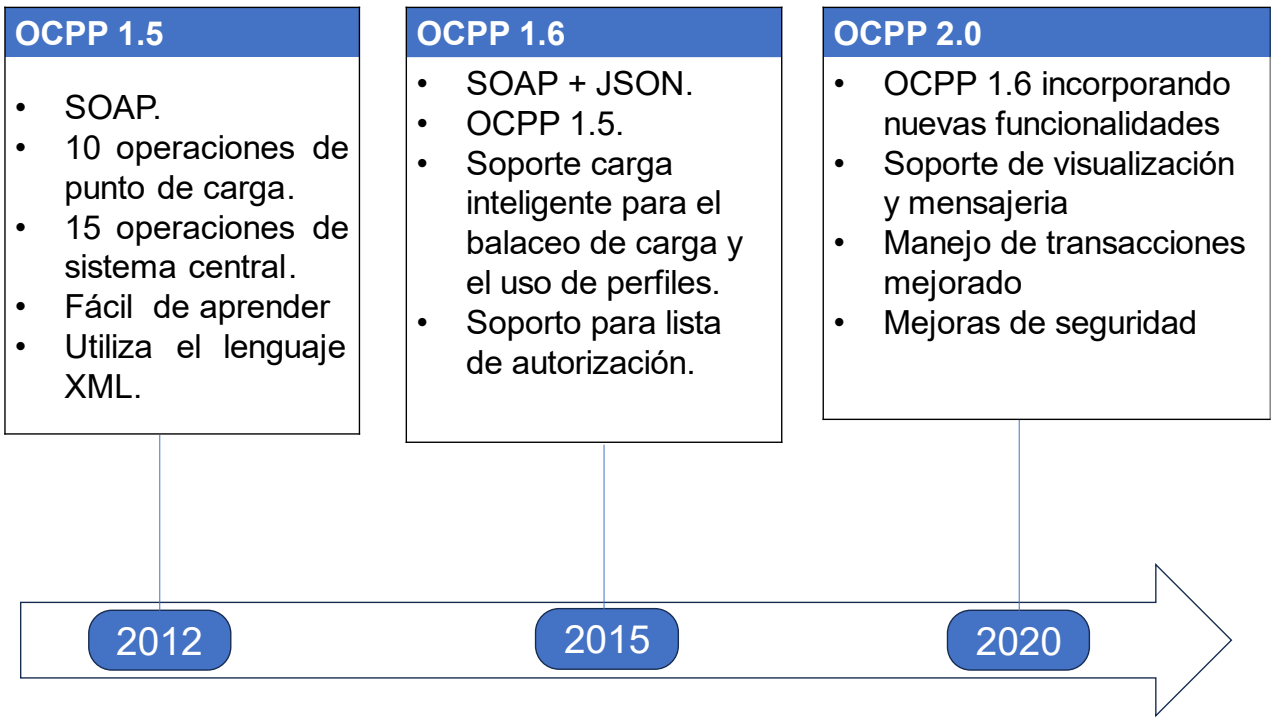


Ilustración 1.Evolución del Protocolo OCPP (Elaboración propia)

En la investigación se va a emplear el protocolo OCPP en su versión 1.6 por políticas del centro de desarrollo y debido a que esta versión está más estandarizada que las otras versiones porque introduce nuevas características para adaptarse al mercado como son:

- La carga inteligente.
- Emplea JSON sobre *Websockets.*
- Brinda mejores posibilidades de diagnóstico.
- Dispone más estados de puntos de carga.

Por lo que, la versión 1.6 de OCPP está basado en la versión 1.5 con algunas características nuevas y mejoras textuales, aclaraciones y correcciones para todas las ambigüedades conocidas (Alliance, 2016).

En la Figura 1 se muestra la representación del protocolo en la versión 1.6.

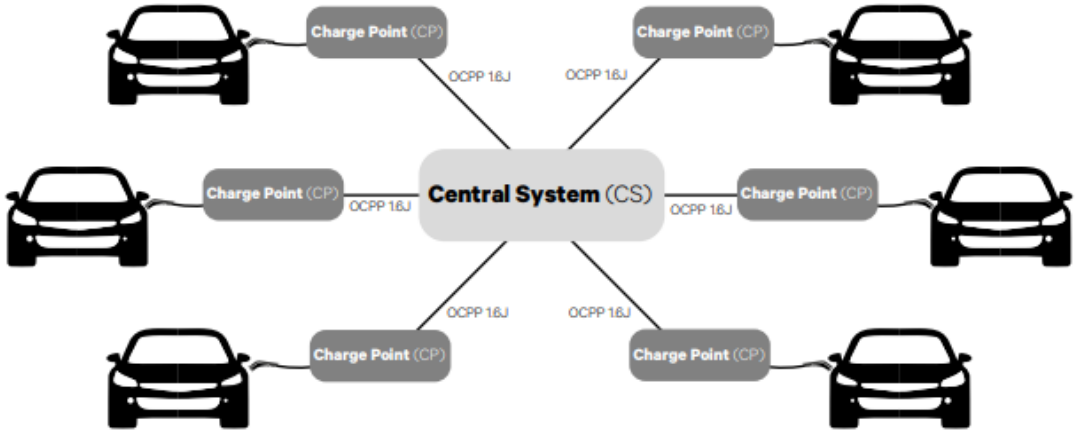


Figura 1. Protocolo OCPP en la versión 1.6 (Open Charge Point Protocolo. Guía de Activación)

1.1.6 Características del Protocolo OCPP en su versión 1.6

Ofrece un formato de mensaje estándar y un protocolo de transmisión para intercambiar información entre estaciones de carga y redes, incluidas características como iniciar y detener la carga, consultar el estado de la batería y procesar pagos. El protocolo también incluye varias funciones de seguridad, como firmas digitales y autenticación, para garantizar una transmisión de información precisa y segura entre la infraestructura de carga (EVBBC, 2023).

En el OCPP 1.6 ofrece un conjunto de características y de mensajes asociados agrupados en perfiles, los cuales, según las funcionalidades requeridas para los desarrolladores pueden optar por implementar al menos uno de los siguientes perfiles:

Perfil	Descripción
Core	Establece las funcionalidades esenciales para la comunicación, gestión de carga y notificaciones entre los puntos de carga de vehículos eléctricos y el sistema central.
Firmware Management	Se centra en la gestión y actualización segura del firmware de los dispositivos de carga de vehículos eléctricos. Proporciona funcionalidades para la planificación y ejecución

	de actualizaciones, verificación de integridad y manejo de errores durante el proceso de actualización.
Local Auth List Management	Permite la gestión de una lista de autorización local en los puntos de carga. Proporciona funcionalidades para agregar, eliminar y gestionar usuarios autorizados, así como para actualizar y sincronizar la lista con el sistema central.
Remote Trigger	proporciona un mecanismo estandarizado para la activación/desactivación remota del proceso de carga en puntos de carga compatibles, lo que es fundamental para la gestión eficiente y segura de la infraestructura de carga de vehículos eléctricos.
Reservation	Se enfoca en la gestión de reservas de carga de vehículos eléctricos. Proporciona funcionalidades para crear, modificar y cancelar reservas, así como para notificar a los usuarios sobre el estado de sus reservas.
Smart Charging	Se enfoca en la gestión inteligente de la carga de vehículos eléctricos. Proporciona funcionalidades avanzadas para la planificación, optimización e integración con la red eléctrica, con el objetivo de maximizar la eficiencia y minimizar los costos de la carga de vehículos eléctricos.

(HACER TABLA DE LOS PERFILES DEFINIDOS POR EL OCPP.16J)

Es necesario destacar, que de los anteriores perfiles el Core es el básico y necesario de implementar, debido a que las funcionalidades que provee están relacionadas con el inicio y el proceso transacción de datos, las cuales son:

- *Authorize*: permite la autorización de los usuarios para iniciar la carga.
- *Start/Stop transaction*: permite iniciar y detener el proceso de carga.
- *Heartbeat*: permite la comunicación periódica entre la estación de carga y el sistema central de gestión para verificar que la conexión sigue activa.
- *Meter values*: permite la transmisión de datos de medición de la energía consumida durante la carga.
- *Status Notification*: permite la notificación del estado de la estación de carga al sistema central de gestión.

1.2 Análisis de las soluciones existentes

Existen varios simuladores que permiten comprobar la correcta instalación y funcionamiento de los puntos de recarga. A continuación, se detallan una serie de ellos que han podido aportar ideas ya sea por su cercanía con el tema y funcionalidades de la aplicación o por el diseño innovador:

- *Charge Point Emulator OCPP 1.6.*
- *Simple OCPP 1.6 Chargebox Simulator.*
- OCPP 1.6 Simulador es un simulador de estación de carga que ejecuta OCPP versión 1.6 en SOAP o *WebSocket.*

Para llevar a cabo la comparación de las soluciones homólogas estudiadas se tuvieron en cuenta los siguientes aspectos:

- Tipo de licencia.
- Perfiles que cubre.
- Las funcionalidades definidas.

A continuación, se muestra una tabla comparativa sobre para los diferentes simuladores:

Sistemas	**Tipo de licencia**	**Plataforma que emplea para la integración**	**Perfiles que cubre**	**Funcionalidades**
Charge Point Emulator OCPP 1.6	Privado	EV Space	Core	Definir la conexión con el sistema central. Especificar la identificación

				de la etiqueta que activará la caja de carga: *IdTag*.[1] Enviar eventos de mensajes de *chargebox*: *Connect* *Authorize* *Start/Stop transaction* *Heartbeat* *Meter values* *Status Notification*
Simple OCPP 1.6 Chargebox Simulator	Privado	EV Charging for Application Developers	Core	Definir la estación central con la que conectar. Especificar la identificación de la etiqueta que activará la caja de carga: *IdTag* Enviar eventos de mensajes de *chargebox:* *Connect* *Authorize* *Start/Stop transaction* *Heartbeat* *Meter values* *Status Notification*
OCPP 1.6 Simulador	Privado		Core	Definir la estación central con la que conectar. Especificar la identificación de la etiqueta que activará la caja de carga: *IdTag*. Enviar eventos de mensajes

[1] IsTag es el identificador de la tarjeta RFID que utilizan los conductores para autoidentificarse en el punto de carga o en la plataforma.

				de *chargebox*: *Connect* *Authorize* *Start/Stop transaction* *Heartbeat* *Meter values* *Status Notification*

Tabla 1. Comparación de los sistemas homólogos

Realizado el estudio de las soluciones existentes abordadas anteriormente se concluye que las mismas no constituyen una solución factible al problema de la investigación. Estos simuladores son de uso privado para los clientes y el costo para acceder sus servicios es muy alto. Sin embargo, es importante destacar que las soluciones analizadas estandarizan el uso del protocolo OCPP en su versión 1.6 y aportan funcionalidades definidas por el perfil Core para tener en cuenta en el desarrollo de la propuesta de solución.

1.3 Entorno de trabajo

La correcta selección del entorno de trabajo para el desarrollo de la solución es vital en el proceso. En lo adelante se analizan las metodologías existentes y se realiza una selección de la más adecuada, así como de las tecnologías y herramientas a emplear.

1.3.1 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. El objetivo de las distintas metodologías es el de intentar organizar los equipos de trabajo para que estos desarrollen las funciones de un programa de la mejor manera posible (Universidades, Metodologías de desarrollo software, 2020).

La metodología seleccionada para el desarrollo de la propuesta de solución es la variación de Proceso Unificado Ágil (AUP, por sus siglas en inglés) para la UCI por políticas del centro de desarrollo.

Esta metodología cuenta con las fases de Inicio, Ejecución y Cierre. Además, define las disciplinas para la fase de Ejecución: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación y Pruebas de aceptación. Para la disciplina de Requisitos define 4 escenarios.

En la presente investigación se seleccionará el escenario 4, el cual permite a través de historias de usuarios un encapsulamiento de los requisitos del sistema. Este escenario se aplica a negocios bien definidos y que no sean muy extensos, además el cliente estará siempre acompañando al equipo de desarrollo para convertir los detalles de los requisitos y así poder implementarlos, validarlos y probarlos. Intenta evitar los torcidos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados. Permite usar cualquier conjunto de herramienta que se desee, principalmente las que sean más adecuadas a la hora de realizar el trabajo que a menudo son herramientas simples o incluso de código abierto.

1.3.2 Lenguaje de programación

Un lenguaje de programación es un conjunto de instrucciones y sintaxis utilizadas que permite la implementación de un *software* ya sea de tipo móvil, web o una aplicación (Universidades, 2020).

Golang v1.18

Golang, también conocido como Go, es un lenguaje de programación de código abierto desarrollado por Google. Se caracteriza por su simplicidad, eficiencia y enfoque en la productividad del desarrollador. A lo largo de los años, ha ganado popularidad en la comunidad de programación debido a sus ventajas y su capacidad para abordar desafíos en el desarrollo de software convirtiéndolo en un lenguaje altamente atractivo para los programadores (Apuntes de programacion, 2023).

1.3.3 Biblioteca estándar

Una biblioteca es un conjunto de recursos prefabricados, que pueden ser utilizados por el programador para realizar determinadas operaciones. Las declaraciones de las funciones utilizadas en estas bibliotecas, junto con algunas macros y constantes predefinidas que facilitan su utilización, se agrupan en ficheros de nombres conocidos. En el desarrollo de la solución se utiliza la siguiente biblioteca: ocpp-go realizada por Lorenzodonini es definida por el protocolo OCPP en su versión 1.6 en el lenguaje de programación GO permite la implementación de perfil Core. Actualmente, con ella se han hecho estudios y validado su correcto funcionamiento mediante el proyecto sectorial que tiene el centro desde hace dos años (lorenzodonini, s.f.).

1.3.4 Editor de texto avanzado

Para el desarrollo se seleccionó *Visual Studio Code (VSCode*) en su versión 1.65, el cual es un editor de código fuente desarrollado por Microsoft. Es un software libre y multiplataforma. Cuenta con un soporte para depuración de código y dispone de un sin número de extensiones. Estas extensiones dan la posibilidad al usuario de escribir y ejecutar códigos de cualquier tipo de lenguaje de programación. Es capaz de detectar errores de forma automática antes de ejecutar el código o la depuración. Algo importante y una ventaja con respecto a un IDE completo que incluye todos los

componentes en un solo paquete, es que, con *VSCode* puedes instalar únicamente las herramientas de desarrollo requeridas, y personalizarlo de acuerdo con tus necesidades (Flores, 2020).

1.3.5 Lenguaje de modelado

Se utilizará el Lenguaje Unificado de Modelado (UML) para visualizar, especificar, construir y documentar los artefactos de un sistema, incluyendo su estructura y diseño. Utiliza un conjunto de símbolos y notaciones para representar gráficamente los diversos componentes que forman parte de la arquitectura de software. Permite el modelo de procesos de negocios y el modelado de requisitos apoyándose en el análisis orientado a objetos (Amazon.com, Inc. o sus afiliados, 2023).

1.3.6 Herramienta para el modelado de procesos

Para el modelado de procesos se utilizó el *Visual Paradigm 4* en su versión 8.0 esta permite ayudar a los equipos de desarrollo de software a capturar los requisitos correctos y transformarlos en diseños precisos. Ayuda a los desarrolladores a crear el software adecuado según los requisitos lo que genera un resultado con mayor calidad. Es una herramienta CASE (*Computer Aided Software Engineering*, Ingeniería de *Software* Asistida por Computadora) para el desarrollo que utilizan el lenguaje UML (Lenguaje Unificado de Modelado) en su versión 2.2 ofreciendo confiabilidad y estabilidad en el proceso de desarrollo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar códigos desde diagramas y la generación de documentación.

Conclusiones del capítulo

- La investigación de los conceptos asociados al objeto de estudio proporcionó que la versión del protocolo a implementar es la 1.6 JSON por ser la más estandarizada y es la definida por la plataforma de electromovilidad en desarrollo.
- El análisis de soluciones similares permitió definir que el principal perfil asociado al protocolo a implementar es el Core, el cual proporciona funcionalidades asociadas a la notificación de estado, proceso de transferencia de energía, envío de mediciones del punto de carga.
- La selección de las herramientas y del lenguaje de programación permiten definir el ambiente de desarrollo necesario para la implementación de la propuesta de solución.

CAPÍTULO 2: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO

El presente capítulo aborda los principales aspectos relacionados con las características de la propuesta de solución. Se encapsulan los principales requisitos funcionales y no funcionales con los que debe cumplir la solución propuesta, así como se define el estilo arquitectónico y los patrones de diseño para lograr buenas prácticas en el diseño y posterior implementación del sistema.

2.1 Descripción de la propuesta de solución

Después de haber realizado un análisis de los principales conceptos y herramientas que serán utilizados durante la presente investigación y dada las necesidades planteadas en la situación problemática, la solución propuesta constituye una simulación de un punto de carga el cual se va a conectar a un sistema central mediante el protocolo OCPP en su versión 1.6 para una plataforma de carga pública. Dicha simulación debe ser separada en dos procesos: inicialización y transferencia de carga.

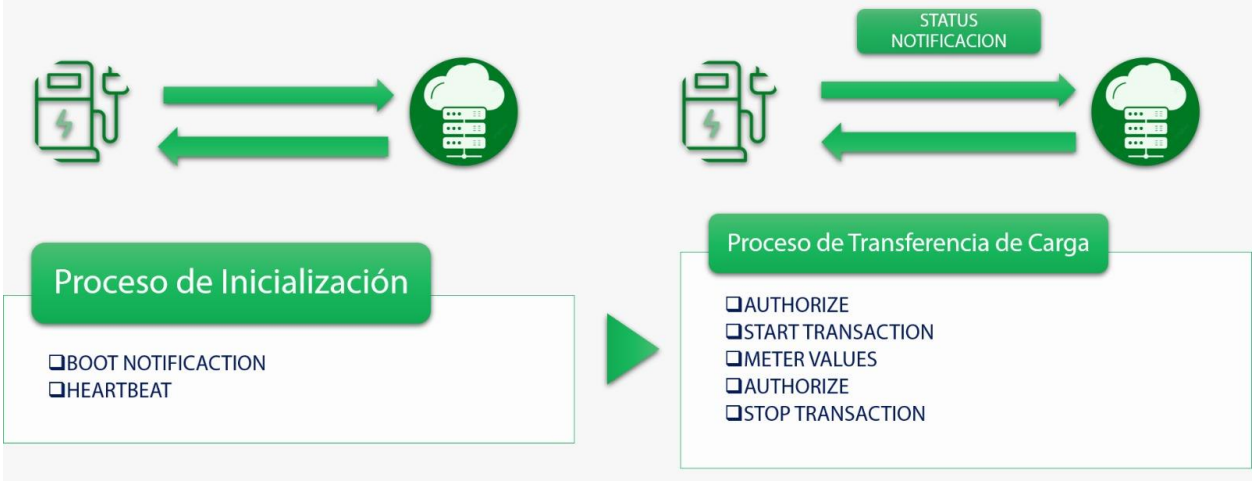


Ilustración 2. Representación de la propuesta de solución (Elaboración propia)

A continuación, se visualiza el proceso de inicialización definido por el protocolo. Dicho proceso inicialización se divide en notificación de inicio de carga y envío de estado de un punto de carga; el cual se lleva a cabo de la siguiente manera:

1. El punto de recarga envía notificación de arranque o reinicio al sistema central, dicha notificación tendrá la configuración de este.
2. El sistema central recibe información de punto de carga y envía si aceptará arranque del punto de recarga mediante un estado que puede ser aceptado o pendiente.

3. En caso de que el sistema central responde con un estado aceptado, el punto de carga ajustará el intervalo de tiempo de un punto de carga.
4. El punto de carga envía en tiempo configurable al sistema central un latido para informar que está activo.
5. El sistema central al recibir el latido envía la hora actual al punto de carga.

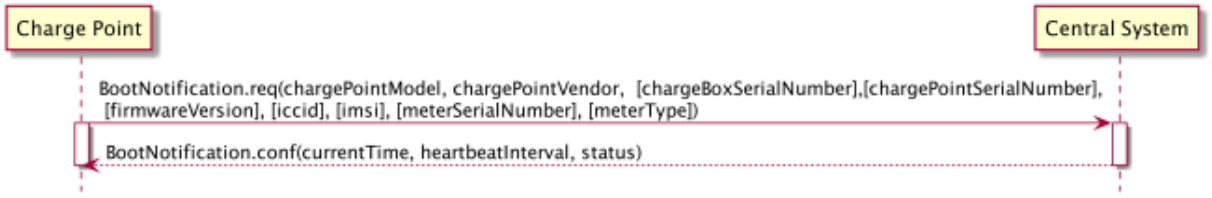


Figura 2. Proceso de notificación de inicio de carga (Open Charge Point Protocol 1.6)

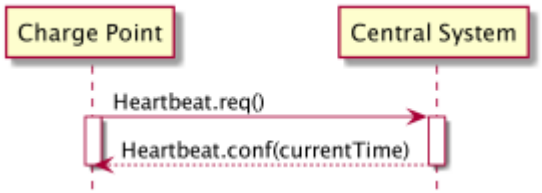


Figure 3. Confirmación de conexión de un punto de carga (Open Charge Point Protocol 1.6)

En la Figura 3 se visualiza el proceso de transacción de carga definido por el protocolo. El proceso de transacción de carga se lleva a cabo de la siguiente manera:

1. El punto de recarga solicita permiso para iniciar carga para que el sistema central lo valide.
2. El sistema central, en este caso, acepta las credenciales y comunica al cargador que el usuario tiene permiso para realizar la carga, mediante un código de estado.
3. Tras recibir la respuesta afirmativa, el punto solicita permiso para iniciar la recarga, y en este caso la central lo autoriza.
4. Durante este periodo de tiempo, podría utilizar '*MeterValues*' para comunicar por intervalos el estado de la carga a la central, aunque se omite el proceso en este ejemplo.
5. Una vez que la carga finaliza, el punto vuelve a solicitar autorización, y la central lo autoriza.
6. El punto solicita permiso para detener la carga, la central registra la petición, y permite que se detenga.
7. El punto queda disponible de nuevo.

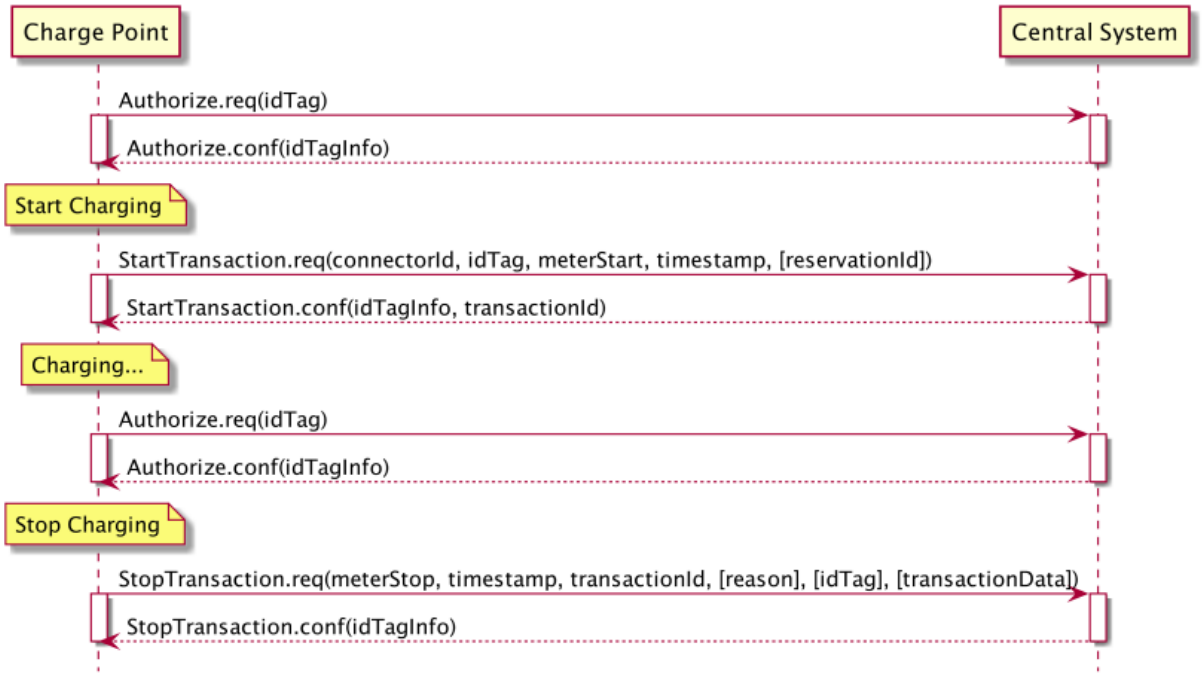


Figura 4. Proceso de transacción de carga (Open Charge Point Protocol 1.6)

2.2 Ingeniería de requisitos

Los requerimientos o requisitos para un sistema son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Tales requerimientos reflejan las necesidades de los clientes o/y usuarios por un sistema que atienda cierto propósito, como sería controlar un dispositivo, colocar un pedido o buscar información (Proceso de Desarrollo y Gestion de Proyectos de software, 2012).

2.2.1 Especificación de requisitos

La definición de requisitos especifica las condiciones y restricciones que debe cumplir el sistema, detallando las necesidades para la satisfacción del equipo de desarrollo. Los requisitos estarán clasificados en funcionales y no funcionales (Copyright 1library, 2023)..

A continuación, se especifican los requisitos funcionales y no funcionales definidos para el desarrollo del simulador.

2.2.2 Requisitos funcionales

Los requisitos funcionales son el resultado del sistema y sus componentes cuando el usuario realiza una tarea o función descrita como un conjunto de entradas, comportamientos y salidas sobre los mismos. Fueron evaluados con el fin de establecer la prioridad y complejidad según el producto de

trabajo de la evaluación de requisitos propuesto por la metodología AUP-UCI. Para determinar la prioridad se asumieron los criterios de urgencia del usuario del sistema (Rivero, 2012).

En la siguiente tabla se muestra la descripción de los requisitos funcionales que se tuvieron en cuenta para la realizar el sistema:

Requerimientos funcionales			
No	**Nombre**	**Descripción**	**Prioridad**
RF1	Solicitar autorización para iniciar proceso de carga.	El punto de carga envía un *idTag* al sistema central para autorizar la carga.	Alta
RF2	Procesar respuesta para autorización del sistema central.	El punto de carga recibe un mensaje que da respuesta si autoriza o no al *idTag* y el valor *IdTagInfo* de la respuesta como se describe en Caché de autorización.	Media
RF3	Establecer conexión con el sistema central.	Conectar el punto de carga mediante *WebSockets* al sistema central.	Alta.
RF4	Iniciar de transacción de carga.	El punto de carga envía al sistema central que inicio transacción de carga con el *connectorId, idTag, meterStart, timestamp*.	Media
RF5	Procesar inicio de transacción carga remota del sistema central.	El punto de carga recibe *transactionId*, *idTagInfo* del sistema central.	Media
RF6	Detener transacción de carga.	El punto de carga envía al sistema central detención de transacción de carga con *meterStop, timestamp, transactionId*.	Media
RF7	Procesar detención de transacción carga remota del sistema central.	El punto de carga recibe mensaje detención de transacción de carga remota del sistema central o un error.	Media
RF8	Inicialización del punto de carga	El punto de carga un punto de recarga envía una notificación arrancar al sistema central.	Media
RF9	Procesar inicialización del punto de carga	El punto de carga recibe la respuesta del sistema central para arrancar el punto carga.	Media

RF10	Enviar valores de medición al sistema central	El punto de carga envía mediciones pertenecientes al proceso de carga al sistema central.	Media
RF11	Enviar cambio de estado o error del conector al sistema central	Un punto de carga envía una notificación al sistema central para informarle sobre un cambio de estado o un error dentro del punto de carga.	Media

Tabla 2. Requisitos funcionales (Elaboración Propia)

2.2.**3 Requisitos no funcionales**

Los requisitos no funcionales, son requisitos que no están directamente relacionados con los servicios específicos que el sistema brinda a sus usuarios. Estos requisitos no funcionales generalmente especifican o limitan las características del sistema en su conjunto. Pueden estar relacionados con propiedades emergentes del simulador.

A continuación, se enumeran los requisitos no funcionales definidos para el desarrollo del sistema.

Seguridad

- RnF1: El protocolo OCPP en su versión 1.6J utiliza webSockets y define que se debe de establecer un canal seguro de comunicación de un punto de carga al sistema central mediante la encriptación de información utilizando los protocolos *Transport Layer Security* (TLS) y *Secure Sockets Layer* (SSL).

Compatibilidad

- RnF2: Mediante el protocolo el simulador debe de establecer comunicación e intercambiar datos de forma bidimensional con el sistema central.

Usabilidad

- RnF3: La ejecución del simulador debe emplear un fichero en formato JSON que este compuesto por las siguientes propiedades: identificador de la tarjeta RFID (RFID_CARD), identificador del punto de carga (CLIENT_ID), dirección del sistema central (CENTRAL_SYSTEM_URL), intentos de reconexión al sistema central (RETRY_COUNT) .
- RnF4: El simulador debe proveer valores por defectos para su invocación ante la no existencia del fichero de configuración o las existencias de propiedades con valores no válidos.

Mantenibilidad

- RnF5: El simulador debe de ser reutilizable por los siguientes sistemas operativos: Windows, Linux.

2.4 Descripción de Historias de Usuarios

Las historias de usuario es la técnica que propone el escenario 4 de la metodología AUP-UCI para encapsular los requisitos funcionales del sistema, la misma que se utiliza para la especificación de requisitos. Con el empleo de esta técnica se describió qué se espera como salida de la implementación, y cómo se ve beneficiado el usuario final. Se expresa en lenguaje natural y sencillo. A continuación, se muestran las historias de usuario correspondientes a los requisitos funcionales del sistema.

Historia de usuario	
Número: 1	Nombre: Solicitar autorización para iniciar proceso de carga.
Usuario: Punto de carga	
Prioridad en negocio: Alta	Riesgo en desarrollo: medio
Puntos estimados: 5	Iteración asignada: 2
Programador responsable: Arianna Correoso	
Descripción: El punto de carga envía un *idTag* al sistema central para autorizar la carga.	
Observaciones: El punto de carga tiene que estar conectado al sistema central para poder realizar la acción.	

Tabla 3. Historia de usuario 1

Historia de usuario	
Número: 2	Nombre: Procesar respuesta para autorización del sistema central.
Usuario: Punto de carga	
Prioridad en negocio: Media	Riesgo en desarrollo: medio
Puntos estimados: 6	Iteración asignada: 1
Programador responsable: Arianna Correoso Azahares	
Descripción: El punto de carga recibe un mensaje que da respuesta si autoriza o no al idTag y el valor IdTagInfo de la respuesta como se describe en Caché de autorización.	
Observaciones: El punto de carga recibió el *idTag*	

Tabla 4. Historia de usuario 2

Historia de usuario	
Número: 3	Nombre: Establecer conexión con el sistema central.
Usuario: Punto de carga	
Prioridad en negocio: Alta	Riesgo en desarrollo: Baja
Puntos estimados: 5	Iteración asignada: 2
Programador responsable: Arianna Correoso Azahares	
Descripción: Conectar el punto de carga mediante *WebSockets* al sistema central.	
Observaciones:	

Tabla 5. Historia de usuario 3

Historia de usuario	
Número: 4	Nombre: Iniciar de transacción de carga.
Usuario: Punto de carga	
Prioridad en negocio: Media	Riesgo en desarrollo: Medio
Puntos estimados: 4	Iteración asignada: 2
Programador responsable: Arianna Correoso Azahares	
Descripción: El punto de carga envía al sistema central que inicio transacción de carga con el *connectorId*, *idTag, meterStart, timestamp*.	
Observaciones: El punto de carga autorizó inicio de transacción de carga.	

Tabla 6. Historia de usuario 4

Historia de usuario	
Número: 5	Nombre: Procesar inicio de transacción carga remota del sistema central.
Usuario: Punto de carga	
Prioridad en negocio: Media	Riesgo en desarrollo: Medio
Puntos estimados: 5	Iteración asignada: 3
Programador responsable: Arianna Correoso Azahares	
Descripción: El punto de carga recibe *transactionId*, *idTagInfo* del sistema central.	

Observaciones:

Tabla 7. Historia de usuario 5

Historia de usuario	
Número: 6	Nombre: Detener transacción remota de carga.
Usuario: Punto de carga	
Prioridad en negocio: Media	Riesgo en desarrollo: Media
Puntos estimados: 5	Iteración asignada: 2
Programador responsable: Arianna Correoso Azahares	
Descripción: El punto de carga envía al sistema central detención de transacción de carga remota con *meterStop, timestamp, transactionId*.	
Observaciones: El punto de carga realiza una transacción de carga remota y el punto de carga debió haber autorizado la detención de transacción de carga.	

Tabla 8. Historia de usuario 6

Historia de usuario	
Número: 7	Nombre: Procesar detención de transacción carga remota del sistema central.
Usuario: Punto de carga	
Prioridad en negocio: Media	Riesgo en desarrollo: Baja
Puntos estimados: 5	Iteración asignada: 2
Programador responsable: Arianna Correoso Azahares	
Descripción: El punto de carga recibe mensaje de detención de transacción de carga del sistema central o un error.	
Observaciones: Se debe de haber ejecutado inicio de detención de carga remota.	

Tabla 9. Historia de usuario 7

Historia de usuario	
Número: 8	Nombre: Inicialización del punto de carga

Usuario: Punto de carga	
Prioridad en negocio: Media	Riesgo en desarrollo: Baja
Puntos estimados: 5	Iteración asignada: 2
Programador responsable: Arianna Correoso Azahares	
Descripción: El punto de carga envía una notificación de arranque al sistema central después de un intervalo de tiempo configurable.	
Observaciones:	

Tabla 10. Historia de usuario 8

Historia de usuario	
Número: 9	Nombre: Procesar inicialización del punto de carga
Usuario: Punto de carga	
Prioridad en negocio: Media	Riesgo en desarrollo: Media
Puntos estimados: 5	Iteración asignada: 2
Programador responsable: Arianna Correoso Azahares	
Descripción: El punto de carga recibe la respuesta del sistema central para arrancar el punto carga.	
Observaciones:	

Tabla 11. Historia de usuario 9

Historia de usuario	
Número: 10	Nombre: Enviar valores de medición al sistema central
Usuario: Punto de carga	
Prioridad en negocio: Media	Riesgo en desarrollo: Media
Puntos estimados: 4	Iteración asignada: 1
Programador responsable: Arianna Correoso Azahares	
Descripción: El punto de carga envía una solicitud para descargar los valores del medidor. La solicitud contendrá para cada muestra: *connectorId*, *meterValue*.	

Observaciones:

Tabla 12. Historia de usuario 10

Historia de usuario	
Número: 11	Nombre: Enviar cambio de estado o error de un conector al sistema central
Usuario: Punto de carga	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 4	Iteración asignada: 5
Programador responsable: Arianna Correoso	
Descripción: Un punto de recarga envía una notificación al Sistema Central para informarle sobre un cambio de estado o un error dentro del Punto de Recarga. La notificación contiene: *connectorId, errorCode*, *estatus*.	
Observaciones: El punto de carga inicio transacción o detener transacción remota en ese proceso.	

Tabla 13. Historia de usuario 11

2.5 Diseño Arquitectónico

Cliente-Servidor es uno de los estilos arquitectónicos comerciados más conocidos, el cual está compuesto por dos mecanismos, el distribuidor y el consumidor. El distribuidor es un servidor que brinda una serie de servicios o recursos los cuales son consumidos por el cliente en este caso los puntos de cargas y los usuarios registrados en el servidor. Existe un servidor y múltiples clientes que se conectan al servidor para recuperar todos los recursos necesarios para funcionar. En este sentido, el cliente solo es una capa para representar los datos y se detonan acciones para modificar el estado del servidor, mientras que el servidor es el que hace todo el trabajo pesado, se encarga de monitorear todos los puntos de cargas conectados a él, y así mismo a todos los usuarios almacenados en este sistema (Blancarte, 2021).

La siguiente imagen representa el patrón arquitectónico donde los clientes son los simuladores de un punto de carga y se conecta al servidor sistema central mediante protocolo OCPP en su versión 1.6 utilizando *WebSockets*.

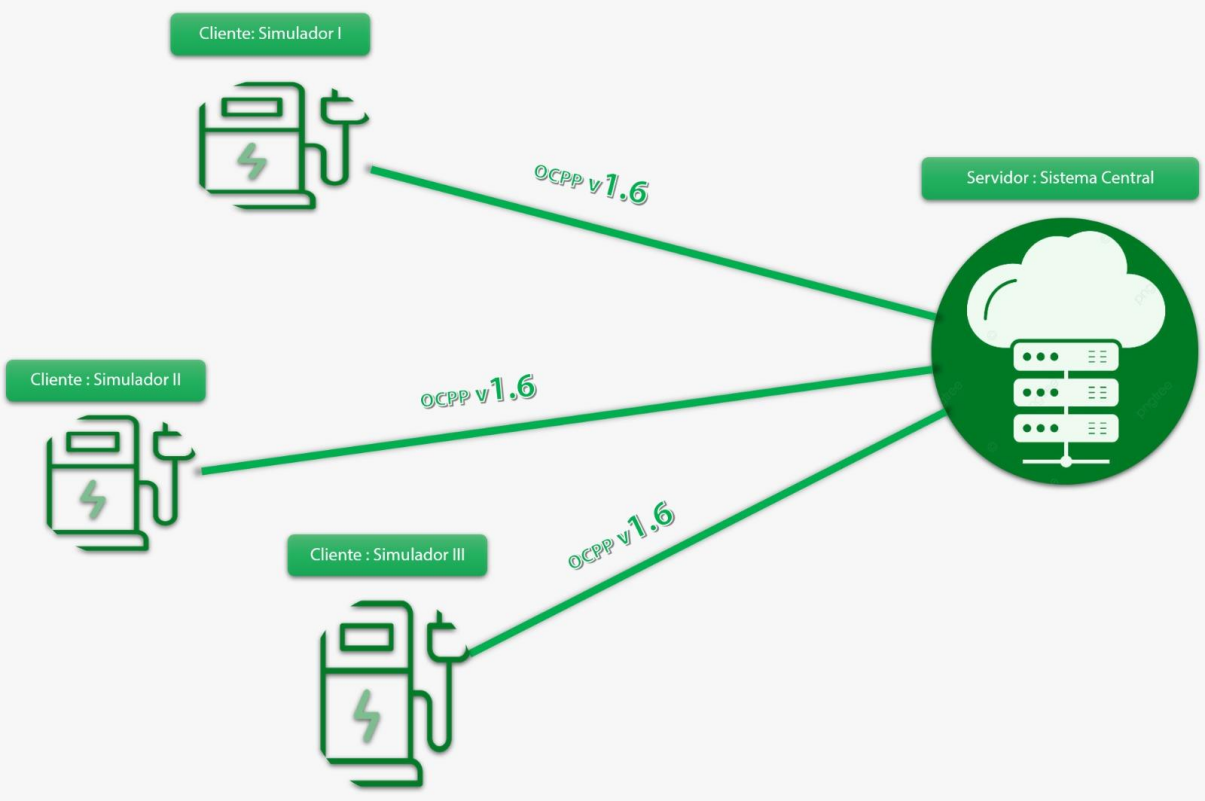


Ilustración 3. Patrón Arquitectónico Cliente Servidor (Elaboración propia)

2.6 Patrones de diseño

Los patrones de diseño o *design patterns*, son una solución general, reutilizable y aplicable a diferentes problemas de diseño de software. Se trata de plantillas que identifican problemas en el sistema y proporcionan soluciones apropiadas a problemas generales a los que se han enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error.

2.6.1 Patrones GRASP (*General Responsibility Assignment Software Patterns*)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Controlador: El objetivo de este patrón es asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas, o sea, delega la responsabilidad en otras clases con las que mantiene un modelo de alta cohesión. Este patrón queda evidenciado en la clase *ChargePoint*.

Bajo acoplamiento: Determina el nivel de dependencia de una clase con respecto a otras. Su uso potencia la reutilización, el mantenimiento y la mitigación de efectos a producirse en una clase al hacer cambios en otra.

Alta cohesión: Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Significa que las clases del sistema tienen asignadas solo las responsabilidades que les corresponde y mantienen una estrecha relación con el resto de las clases.

2.6.2 Patrones GoF (Gang of Four)

Estrategia*:** es un patrón de diseño de tipo comportamiento que define un conjunto de algoritmos, separarlos cada uno de ellos en una clase y hacer uso de objetos intercambiables. En la solución propuesta se evidencia en la clase ***CoreProfileHandler que implementa la interfaz ***ChargePointHandler*** que representa las funcionalidades del perfil Core del OCPP y que son inicializadas por el sistema central.

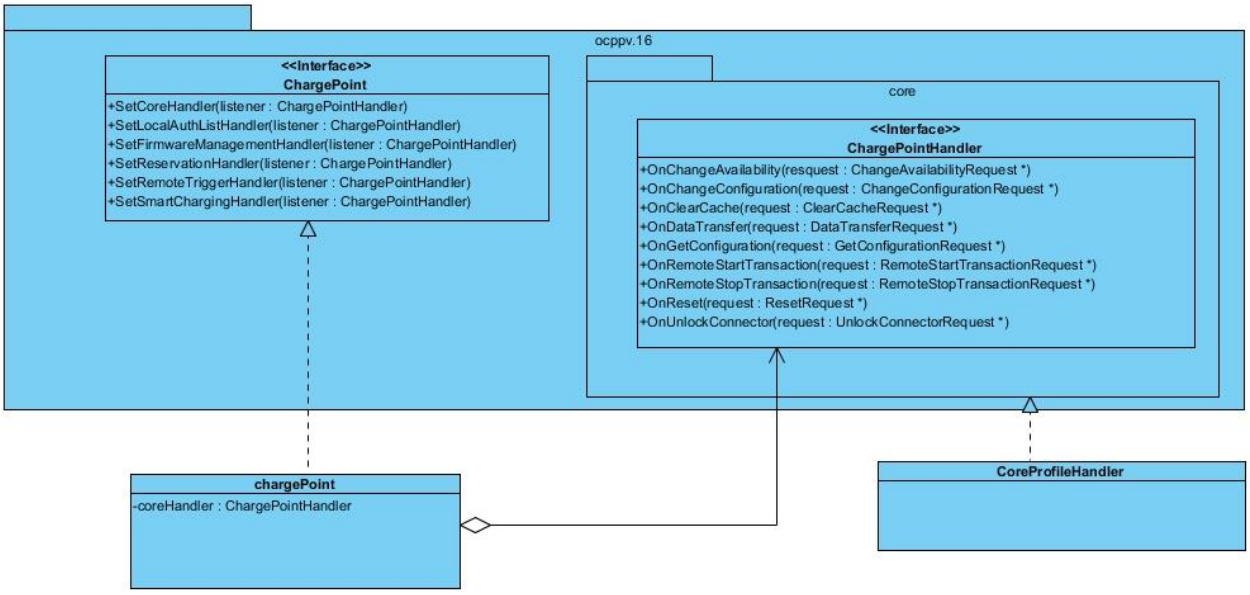


Ilustración 4. Patrón estrategia (Elaboración propia)

Conclusiones del capítulo

- El modelo conceptual de la propuesta de solución permitió representar los conceptos significativos del negocio y la relación que existe entre estos.
- Fueron definidos 11 requisitos funcionales, los cuales fueron descritos haciendo el uso de las historias de usuarios.
- Fueron definidos 10 requisitos no funcionales, los cuales describen las cualidades o características que tendrá el sistema.
- Se definió la arquitectura del sistema, seleccionando el modelo Cliente-Servidor como patrón arquitectónico y apoyándose en patrones de diseño GRASP y GOF, lo que permitió la correcta estructuración del sistema.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

El presente capítulo tiene como principal objetivo abordar la implementación del sistema a partir del diseño desarrollado en el capítulo anterior. Según la metodología seleccionada, la implementación se debe realizar de forma iterativa y ascendente para poder brindar un producto final más completo y funcional para el cliente. También se describirán los estándares de codificación, así como el diseño de casos de prueba y las pruebas unitarias.

3.1 Estándar de codificación

Se entiende como estándar de código a un conjunto de conversiones establecidas previamente. Si bien los programadores deben implementar un estándar de forma prudente y tender siempre el formato más práctico posible, aunque la forma usada va siempre dirigida a la facilidad de comprender el código y devolver ciertas partes realizadas por otros integrantes, así como la depuración de estas (Library, 2019).

La implementación del sistema fue desarrollada utilizando el idioma inglés para los nombres de constantes, variables, estructuras, funciones, clases y métodos de clase. Como estándar de codificación se propone el definido por el lenguaje Go.

- Los nombres de los paquetes son minúsculos, cortos y concisos. Se evita usar guiones bajos o mezclar mayúsculas y minúsculas.

```
import (
    "crypto/tls"
    "crypto/x509"
    "flag"
    "fmt"
    "io/ioutil"
    "math/rand"
    "os"
    "strconv"
    "strings"
    "time"

    "github.com/sirupsen/logrus"

    ocpp16 "github.com/lorenzodonini/ocpp-go/ocpp1.6"
    "github.com/lorenzodonini/ocpp-go/ocpp1.6/core"
    "github.com/lorenzodonini/ocpp-go/ocpp1.6/localauth"
    "github.com/lorenzodonini/ocpp-go/ocpp1.6/types"
    "github.com/lorenzodonini/ocpp-go/ocppj"
    "github.com/lorenzodonini/ocpp-go/ws"
    "github.com/spf13/viper"
)
```

Figura 5. Definición de paquetes

- Los nombres de variables, funciones y métodos tienen la primera letra de cada palabra en mayúscula, excepto la primera palabra.

```
func setupChargePoint(chargePointID string) ocpp16.ChargePoint {
    return ocpp16.NewChargePoint(chargePointID, nil, nil)
}
```

Figura 6. Definición de los nombres de las funciones

- Las funciones, métodos y variables a los que se puede acceder desde otros paquetes, empiezan con mayúscula.
- Las funciones, métodos y variables a los que no se puede acceder desde otros paquetes, empiezan con minúscula.

```
type chargePoint struct {
    client                *ocppj.Client
    coreHandler           core.ChargePointHandler
    localAuthListHandler  localauth.ChargePointHandler
    firmwareHandler       firmware.ChargePointHandler
    reservationHandler    reservation.ChargePointHandler
    remoteTriggerHandler  remotetrigger.ChargePointHandler
    smartChargingHandler  smartcharging.ChargePointHandler
    confirmationHandler   chan ocpp.Response
    errorHandler          chan error
    callbacks             callbackqueue.CallbackQueue
    stopC                 chan struct{}
    errC                  chan error // external error channel
}
```

Figura 7 Definición de valores privados

3.2 Resultados de la implementación

En la figura 7, se pueden observar las respuestas del sistema central al recibir una petición del simulador.

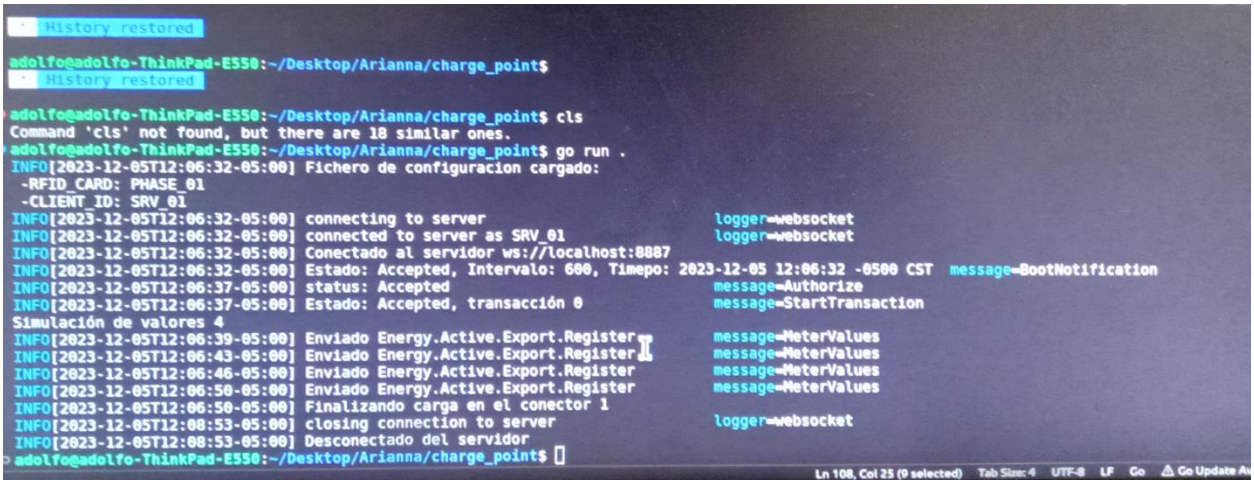


Figura 8. Respuesta del sistema

3.3 Pruebas de software

Al desarrollar cualquier software es algo muy común que se cometan cualquier tipo de errores que conlleven a disímiles errores en las aplicaciones. Por tal motivo, se necesitan realizar varias pruebas de software para poder erradicar la mayoría de estos errores. Estas pruebas se pueden definir como el conjunto de procesos con los cuales se pretende probar un sistema o aplicación con el propósito de

comprobar su correcto funcionamiento. Se debe tener presente que estas se pueden realizar en diferentes momentos del desarrollo de la aplicación, desde su creación hasta su instante de despliegue. También se pueden ejecutar de manera automática determinando en cualquier momento si se tiene una aplicación estable para verificar si un cambio en cualquier parte de este sistema puede afectar en otra parte del sistema (Rivero, 2012).

3.3.1 Pruebas funcionales

Las pruebas funcionales están enfocadas principalmente en verificar que las funcionalidades especificadas en el sistema cumplan con las especificaciones previstas, por lo que están basadas en los requisitos funcionales. Estos tipos de pruebas incluyen pruebas unitarias, pruebas de regresión, pruebas de aceptación del usuario, entre otras. Las pruebas funcionales se pueden realizar además usando técnicas de caja blanca, las cuales realizan pruebas sobre la estructura interna y el diseño del programa.

Método de caja blanca.

Las pruebas de caja blanca es una técnica de prueba de software en ingeniería de software que consisten en probar el código y el diseño interno del software para verificar el flujo de entrada-salida y comprobar el diseño, la usabilidad y la seguridad del software. Las pruebas de caja blanca son un término genérico que engloba muchos tipos diferentes de pruebas de software, incluidas las pruebas unitarias. Dado que las pruebas de caja blanca implican probar el código y la programación, llevarlas a cabo suele requerir ciertos conocimientos de programación informática.

3.3.2 Pruebas unitarias

Se centra en el proceso de verificación de la menor unidad del diseño del software: el componente de software o módulo. Se emplea para detectar errores debidos a cálculos incorrectos, comparaciones incorrectas o flujos de control inapropiados. Las pruebas del camino básico y de bucles son técnicas muy efectivas para descubrir una gran cantidad de errores en los caminos.

```
127  func TestBootNotification(t *testing.T) {
128      confirmation, err := cp.BootNotification("model1", "vendor1")
129      if err != nil {
130          t.Fatal(err)
131      } else if confirmation.Status == core.RegistrationStatusRejected {
132          t.Fatal(errors.New("El sistema central ha denegado la conexion"))
133      }
134  }
```

Figure 9. Código fuente de la prueba unitaria TestBootNotification

```
136  func TestEnergyTransfer(t *testing.T) {
137      //Enviar "estado disponible" del cargador
138      statusNotification(0, core.NoError, core.ChargePointStatusAvailable)(t)
139      time.Sleep(5 * time.Second)
140
141      //autorizar la tarjeta RFID
142      authorize(idTag)(t)
143      time.Sleep(1 * time.Second)
144
145      //Enviar estado de "preparacion" del conector 1
146      statusNotification(1, core.NoError, core.ChargePointStatusPreparing)(t)
147      time.Sleep(3 * time.Second)
148
149      //Iniciar transaccion
150      startTransaction(1, idTag, meterValue, types.NewDateTime(time.Now()))(t)
151      time.Sleep(time.Second)
152
153      //enviar estado "cargando" del conector 1
154      statusNotification(1, core.NoError, core.ChargePointStatusCharging)(t)
155
156      //Enviar valores de medicion de
157      meterValues(1, t)
158
159      //Enviar estado de "finalizando" del conector 1
160      statusNotification(1, core.NoError, core.ChargePointStatusFinishing)(t)
161      time.Sleep(2 * time.Second)
162
163      //Detener transaccion
164      stopTransaction(meterValue, types.NewDateTime(time.Now()), transactionID)
165      time.Sleep(5 * time.Second)
166
167      //Enviar estado "disponible" del conector 1
168      statusNotification(1, core.NoError, core.ChargePointStatusAvailable)(t)
169
170  }
```

Figura 9. Código fuente de las pruebas unitarias restantes

En la siguiente figura se muestra el resultado de la realización de las pruebas unitarias realizadas durante el proceso de implementación, en la que se evidencia la admisión de todas las verificaciones.

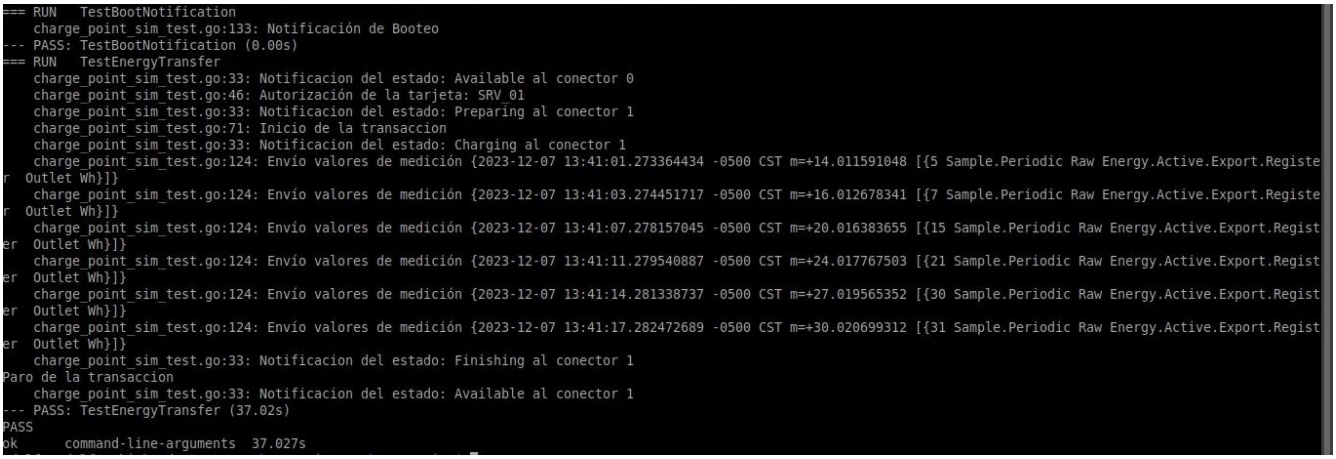


Figura 10. Resultado de las Pruebas unitarias

3.4 Diseño de pruebas

Los casos de prueba incluyen todas las funciones que el programa es capaz de realizar. Deben tener en cuenta el uso de todo tipo de datos de entrada/salida, cada comportamiento esperado, todos los elementos de diseño, y cada clase de defecto. Todos los requisitos deberán ser cubiertos por los casos de prueba, de manera tal que cubra el software a fondo. Se puede decir que son la descripción de las actividades que se van a ejecutar con el fin de validar la aplicación (Duarte, 2015).

Descripción general			
01 - Solicitar autorización para iniciar proceso de carga.			
Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
E.C 1 Enviar Authorize.req	El punto de carga envía un idTag al sistema central para autorizar la carga.	Enviar petición de authorization.	Simulador
E.C 2 Enviar Authorize.conf	El punto de carga recibe un mensaje que da respuesta si autoriza o no al idTag y el valor IdTagInfo de la respuesta como se describe en Caché de autorización.	Recibir confirmación de petición de autorización.	Sistema Central

Tabla 14. Solicitar autorización para iniciar proceso de carga

Descripción general			
02 - Establecer conexión con el sistema central.			
Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
E.C 1 Iniciar conexión con el sistema central	Conectar el punto de carga mediante WebSockets al sistema central.	Devuelve el estado de la conexión.	Simulador

Tabla 15. Establecer conexión con el sistema central

Descripción general			
03- Inicialización de transacción de carga.			
Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
E.C 1 Enviar StartTransaction.req	El punto de carga envía al sistema central que inició la transacción de carga con el connectorId, idTag, meterStart, timestamp.	Enviar petición de inicio de la transacción.	Simulador
E.C 1 Enviar StartTransaction.conf	El punto de carga recibe transactionId, idTagInfo del sistema central.	Recibir confirmación de petición de inicio de la transacción.	Sistema Central

Tabla 16. Inicialización de transacción de carga

Descripción general			
04 - Detener transacción de carga.			
Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
E.C 1 Enviar StopTransaction.req	El punto de carga envía al sistema central detención de transacción de carga con meterStop, timestamp, transactionId.	Enviar petición de parada de la transacción.	Simulador
E.C 1 Enviar StopTransaction.conf	El punto de carga recibe mensaje detención de transacción de carga remota del sistema central o un error.	Recibir confirmación de petición de parada de la transacción.	Sistema Central

Tabla 17. Detener transacción de carga.

Descripción general			
05- Inicialización del punto de carga.			
Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
E.C 1 Enviar BootNotifiacion.req	El punto de carga un punto de recarga envía una notificación de arranque al Sistema Central.	Enviar petición de notificación de inicialización.	Simulador
E.C 1 Enviar BootNotifiacion.conf	El punto de carga recibe la confirmación de que fue aceptada por el sistema central.	Recibir confirmación de petición de inicialización.	Sistema Central

Tabla 18. Enviar valores de medición al sistema central.

Descripción general			
06- Enviar valores de medición al sistema central.			
Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
E.C 1 Enviar valores de medición al sistema central	El punto de carga envía mediciones pertenecientes al proceso de carga al sistema central.	Recolectar el metrado de los conectores a enviar al sistema central.	Simulador

Tabla 19. Enviar valores de medición al sistema central.

Descripción general			
07 - Enviar cambio de estado o error del conector al sistema central.			
Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo Central**

E.C 1 Enviar cambio de estado o error al sistema central	Un Punto de Recarga envía una notificación al sistema central para informarle sobre un cambio de estado o un error dentro del Punto de Recarga referente al conector.	Enviar información al sistema central.	Simulador

Tabla 20. Enviar cambio de estado o error del conector al sistema central.

3.5 Resultados de las pruebas

La realización de pruebas funcionales permitió identificar una serie de no conformidades que se pueden observar en la tabla.

No. NC	Requisito Funcional	Descripción	Complejidad	Estado
1	RF 03	El simulador no informa que el puerto está siendo utilizado por otro programa ocurriendo un fallo en la conexión con el Sistema Central.	Medio	Resuelta
2	RF 03	El simulador no informa que la dirección IP está siendo utilizado por otro programa ocurriendo un fallo en la conexión con el Sistema Central.	Medio	Resuelta
3	RF 03	El simulador no valida la url empleada por el protocolo *WebSocket* para el Servidor Central.	Medio	Resuelta
4	RF 03	El simulador no permite configurar externamente el puerto del Sistema Central a conectar.	Medio	Resuelta
5	RF 03	El simulador no permite configurar externamente la dirección IP del Sistema Central a conectar.	Medio	Resuelta
6	RF 11	El simulador no procesa los estados del conector *[Faulted, SuspendedEV, SuspendedEVSE].*	Baja	Resuelta
7	RF 09	El simulador no procesa el estado de registro Rejected.	Baja	Resuelta
8	RF 05	El simulador no espera que el estado de confirmación de autorización para iniciar el proceso de Carga.	Medio	Resuelta
9	RF 11	El simulador no envía el metrado siguiendo la configuración establecida por el sistema central.	Baja	Resuelta

Tabla 21. No conformidades

Con la aplicación de la estrategia de pruebas diseñada se obtuvieron resultados satisfactorios, demostrando el correcto funcionamiento de todas las funcionalidades implementadas.

Se implementaron las pruebas internas, definidas como disciplina de la metodología AUP-UCI, que guía la presente investigación. Las pruebas unitarias aplicadas están separadas en dos: inicialización de carga y transferencia de carga las cuales demostraron resultados convincentes. El método de casos de prueba aplicado demostró resultados satisfactorios desde el punto de vista funcional. Las no conformidades encontradas fueron analizadas y corregidas en el tiempo establecido, logrando un correcto comportamiento ante diferentes situaciones (entradas válidas y no válidas).

En la primera iteración se detectaron un total de 5 No conformidades: 4 funcionales y 1 unitaria. En la segunda iteración fueron detectadas 3 No conformidades: 2 funcional y 1 unitaria. En la tercera iteración fueron detectadas 1 No conformidad: 1 funcional. En la cuarta no se detectó ningún tipo de error. A continuación, se muestran las no conformidades detectadas en cada iteración de prueba por las que transitó el simulador.

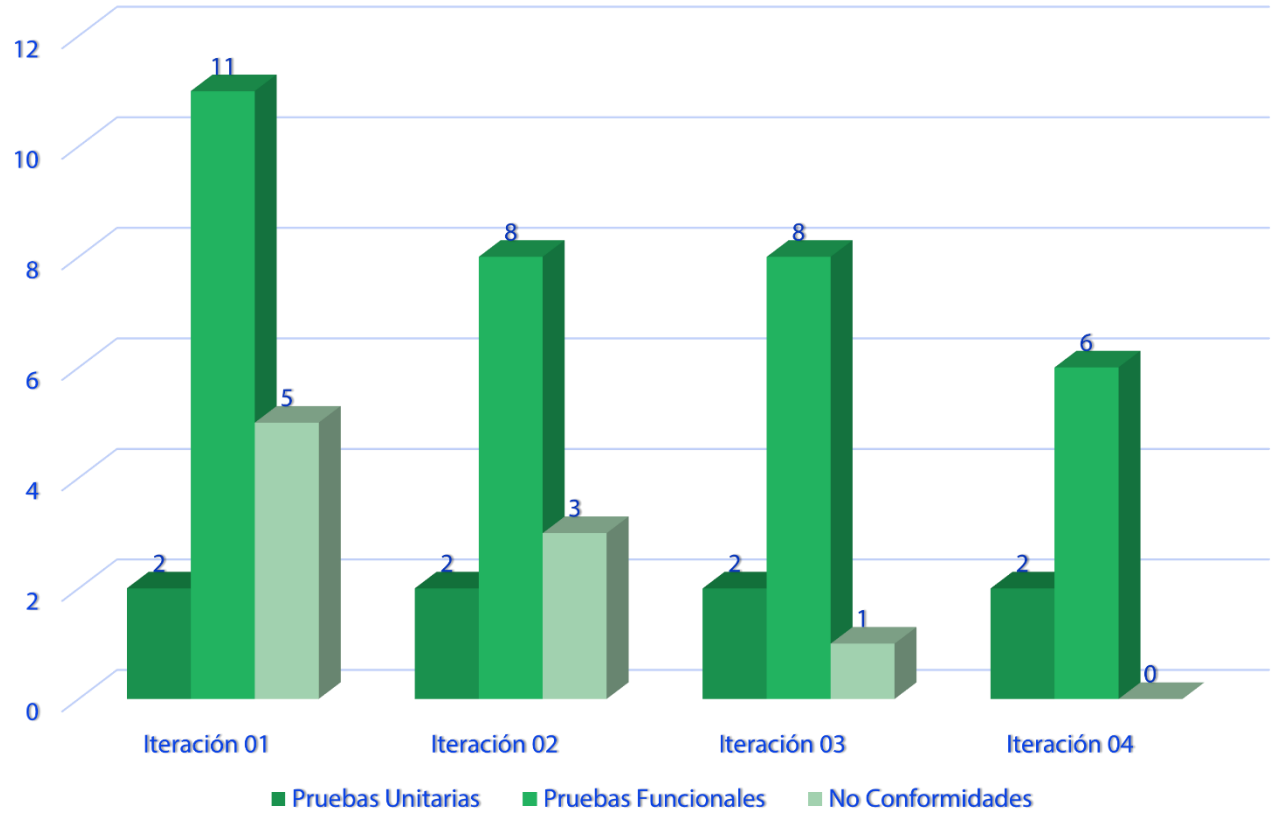


Figura 11. Iteraciones de pruebas

Conclusiones del capítulo

- Con la realización de pruebas funcionales se detectaron una serie de no conformidades que fueron corregidas, asegurando así la calidad del sistema.
- Con las pruebas unitarias lograron validar el correcto funcionamiento del simulador mediante el protocolo OCPP.

CONCLUSIONES FINALES

De la investigación se concluye:

- La elaboración del marco teórico conceptual sobre el protocolo OCPP permitió determinar que de los 6 perfiles que posee el OCPP, solamente el perfil de Core es necesario para la simulación del punto de carga.
- Se obtuvo un simulador para un punto de carga que cumple con el estándar mínimo requerido por la OCA para obtener una certificación básica, la cual consiste en la implementación de un único perfile: Core.
- Las pruebas unitarias permitieron comprobar que el simulador logra establecer una correcta comunicación con el sistema central, no habiendo fallas en el proceso de inicialización y transferencias de carga.

RECOMENDACIONES

Desarrollar los siguientes perfiles en el simulador de OCPP v 1.6 para ampliar sus prestaciones

- Local Auth List Management
- Reservation
- Remote Trigger
- Smart Charging

El perfil *Firmware Management no se recomienda implementar porque dicho perfil actualiza la version*

REFERENCIAS BIBLIOGRÁFICAS

EVBBC. (2023). EVBBC. Obtenido de OCPP 1.6 ofrece un formato de mensaje estándar y un protocolo de transmisión para intercambiar información entre estaciones de carga y redes, incluidas características como iniciar y detener la carga, consultar el estado de la batería y procesar pagos. El

Alliance, O. C. (2016). Open Charge Point Protocol 1.6. Obtenido de Open Charge Point Protocol 1.6: file:///C:/OCPP%20INFO/ocpp-1.6.pdf

Amazon.com, Inc. o sus afiliados. (2023). El lenguaje unificado de modelado: Guía de usuario : aprenda UML directamente de sus creadores. Obtenido de Amazon: https://www.amazon.com/-/es/Grady-Booch/dp/8478290761

Apuntes de programacion. (2023). Obtenido de Intruduccion a goland: https://apuntes.de/golang/introduccion-a-golang/#gsc.tab=0

C, L. B. (2018, septiembre 4). Centro tecnológico . Obtenido de Aplicaciones de sistemas de simulación en la industria: https://itcl.es/blog/para-que-sirven-los-sistemas-de-simulacion/

Chile, M. d. (2018). Plataforma de electromovilidad. Obtenido de https://energia.gob.cl/electromovilidad/introduccion

Contact, P. (2022). Sistemas de control de carga AC para postes de carga y. Obtenido de Sistemas de control de carga AC para postes de carga y: https://www.phoenixcontact.com/es-pc/productos/tecnologia-de-carga-para-la-

Flores. (2020). Qué es Visual Studio Code y qué ventajas ofrece. Obtenido de https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/

Library. (2019). Estandar de Codificación. Obtenido de Library: https://1library.co/article/est%C3%A1ndares-codificaci%C3%B3n-t%C3%ADtulo-sistema-

lorenzodonini. (s.f.). Open Charge Point Protocol implementation in Go . Obtenido de github: https://github.com/lorenzodonini

LugEnergy. (2023). Puntos de recarga en pared para coches eléctricos. Obtenido de LugEnergy.: https://www.lugenergy.com/puntos-de-recarga/puntos-de-recarga-en-pared-wallbox/

Milanés. (2023). Transporte sostenible en Cuba: soluciones ecológicas y autos eléctricos. Obtenido de https://www.cubahora.cu/sociedad/transporte-sostenible-en-cuba-soluciones-ecologicas-y-autos-electricos

Milanés, L. S. (2023, agosto 19). Transporte sostenible en Cuba: soluciones ecológicas y autos eléctricos. Obtenido de Cubahora: https://www.cubahora.cu/sociedad/transporte-sostenible-en-cuba-soluciones-ecologicas-y-autos-electricos

Peño, J. M. (2015). Pruebas de Software. Fundamentos y Técnicas. España.

Real Academia Española. (2023). dle. Obtenido de REA: https://dle.rae.es/simular

Rivero, L. B. (2012, septiembre 11). Proceso de prueba de un software. TINO. Obtenido de https://revista.jovenclub.cu/proceso-de-

Rodríguez, S. A. (2022, octubre 28). ENERGÍAS ALTERNATIVAS EN EL TRANSPORTE ¿Un reto ecológico que es posible concretar en Cuba? Granma.

service@wallbox. (s.f.). Open Charge Point Protoco. GUÍA DE ACTIVACIÓN. Obtenido de wallbox: https://support.wallbox.com/wp-content/uploads/2021/02/ES_OCPP_GUIA_DE_ACTIVACION.pdf

US Energy Solutions. (2023). Ecosistemas de movilidad eléctrica que potencian el transporte sostenible con vehículos eléctricos y la integración de redes inteligentes. Obtenido de Energy5: https://energy5.com/es/ecosistemas-de-movilidad-el%C3%A9ctrica-que-potencian-el-transporte-sostenible-con-veh%C3%ADculos-el%C3%A9ctricos-y-la-integraci%C3%B3n-de-redes-inteligentes

Vazquez, f. A. (2022, enero). Life is On. Obtenido de Life is On.: https://blogespanol.se.com/vehiculo-electrico/2022/01/31/protocolo-