UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 4

Servicio Roaming para la plataforma de carga pública de vehículos eléctricos de Cuba

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

Autor: Oscar O. Coello Perdomo

Tutor: Ing. Adolfo Yasser Santana Rojas

Ing. Julio Alberto Leyva Durán

La Habana, 27 de noviembre de 2023
"Año 64 de la Revolución"


"El futuro mostrará los
resultados y juzgará a cada uno
de acuerdo a sus logros."

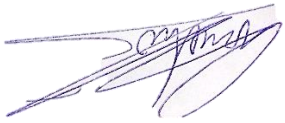
Declaración de autoría:

Declaro ser autor de la presente tesis que tiene por título: *Servicio Roaming para la plataforma de carga pública de vehículos eléctricos de Cuba*, y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo. Para que así conste firmamos la presente a los días 27 del mes de noviembre del año 2023.

_____

Oscar O. Coello Perdomo

Autor



_____

Ing. Adolfo Yasser Santana Rojas

Tutor

Dedicatoria:

Quiero expresar mi profunda gratitud a todas las personas que hicieron posible la realización de esta tesis. En primer lugar, agradezco a mis tutores de tesis por su invaluable orientación, paciencia y apoyo a lo largo de este proceso académico. Mi sincero agradecimiento también al centro VERTEX por brindarme los recursos y el ambiente propicio para llevar a cabo esta investigación. No puedo dejar de mencionar el apoyo incondicional de mi familia y amigos, quienes han sido mi pilar fundamental durante esta etapa de mi vida académica. Por último, pero no menos importante, agradezco a todos los participantes y fuentes que generosamente contribuyeron con su tiempo y conocimientos, haciendo posible este estudio. Su colaboración y respaldo fueron fundamentales para el éxito de este trabajo. Estaré eternamente agradecido/a por su invaluable ayuda.

Resumen:

La lucha contra el cambio climático que hoy afecta al mundo y la pésima situación económica por la que atraviesan los combustibles fósiles, han demostrado ser indicadores que buscan impulsar a nivel mundial el desarrollo de un mercado energético sostenible y, en particular, de los coches eléctricos. La electrificación del sector del transporte automotor es una alternativa eficiente e inteligente para lograr la calidad en el sector productivo y el transporte público, además de contribuir a la independencia energética de cualquier país. Actualmente la inserción de vehículos eléctricos en Cuba ha sido uno de los objetivos planteados para lograr que la electrificación de este sector sea un hecho. Proporcionar un servicio que permita la interoperabilidad entre los diferentes agentes involucrados, como los Operadores de Puntos de Carga (CPO) y los Proveedores de Servicios de Movilidad Eléctrica (EMSP), en el contexto de un sistema de electromovilidad. La investigación se enfoca en la creación de un entorno eficaz y colaborativo que fomente la adopción y uso de los vehículos eléctricos, contribuyendo así al avance hacia un sistema de transporte más sostenible y eficiente en Cuba.

Palabras clave: cargadores eléctricos, interoperabilidad, electromovilidad

Abstract:

The fight against climate change, which currently impacts the world, and the dire economic situation faced by fossil fuels have proven to be indicators urging the global development of a sustainable energy market, particularly focusing on electric cars. The electrification of the automotive transportation sector stands as an efficient and intelligent alternative to enhance the quality of both the productive sector and public transportation, while also contributing to the energy independence of any country. Presently, the integration of electric vehicles in Cuba has been one of the outlined objectives to achieve the electrification of this sector. The main goal is to provide a service that enables interoperability among different stakeholders, such as Charging Point Operators (CPO) and Electric Mobility Service Providers (EMSP), within the context of an electromobility system. The research is concentrated on establishing an effective and collaborative environment that encourages the adoption and use of electric vehicles, thereby contributing to progress towards a more sustainable and efficient transportation system in Cuba.

Keywords: electric chargers, interoperability, electromobility

Índice

Índice de figuras

Índice de tablas

Introducción:

Las Tecnologías de la Información y la Comunicación TIC's impactan en todos los ámbitos de la vida humana por las características que aportan a la sociedad del conocimiento, por todas las posibilidades que brindan al desarrollo de nuevas formas de organizarse, comunicarse, educar, enseñar y aprender, y con ello la transformación y evolución de la misma sociedad. Las TIC's han incrementado el grado de significancia y concepción educativa, estableciendo nuevos modelos de comunicación, además de generar espacios de formación, información, debate, reflexión, entre otros, rompiendo con las barreras del tradicionalismo en el aula. Las TIC's son aquellas tecnologías que -instrumentándose en plataformas materiales e inalámbricas- son inmateriales, interactivas, instantáneas, innovadoras, digitales, interconectadas, diversas, con elevados parámetros de calidad de imagen y sonido, y con influencia de procedimientos sobre procesos y productos. El impacto que han tenido las TIC's en la sociedad ha sido enorme, y aunque existen fenómenos negativos producidos por el impacto de las TIC's en la sociedad, es necesario tratar los aportes que ellas ofrecen no sólo en cuestiones prácticas, sino en capacidades y valores.(Páez 2022)

En 2020, el sector del transporte automotor del país consumía 992 000 toneladas de combustible, de las cuales el 74% eran de diésel y el 26% de gasolina. Una de las principales ventajas de los vehículos eléctricos radica precisamente en la reducción del consumo de combustibles fósiles y su dependencia de la importación. En el caso de los ómnibus híbridos no enchufables, se ha logrado un ahorro que oscila entre el 57% y el 66%. Su uso permite reducir las emisiones de gases con efecto invernadero e incrementar la eficiencia energética y la disponibilidad técnica de los medios de transporte para la población. La adquisición de vehículos eléctricos en la Empresa Eléctrica de La Habana, Empresa de Telecomunicaciones de Cuba S.A (ETECSA por sus siglas en español), Ómnibus Metropolitanos y Aguas de La Habana, con excelentes indicadores de ahorro de combustible, refiere que la inserción gradual de vehículos eléctricos en el país es un hecho. Sin embargo, el nivel de explotación de los vehículos eléctricos en Cuba está limitado a la mitad de la autonomía diseñada por el fabricante, pues los únicos puntos de carga están situados en las empresas

poseedoras de estas flotas. Así, debe reservarse carga suficiente para poder regresar, lo que limita la capacidad de gestión con estos vehículos para viajes largos.(Álvarez 2022)

Uno de los desafíos más apremiantes en el campo de la electromovilidad es la ausencia de un mecanismo efectivo que permita la interoperabilidad entre los diversos agentes involucrados en este ecosistema en constante desarrollo. En este contexto, el Centro de Tecnologías Interactivas VERTEX, adscrito a la Facultad 4 de la UCI, está desempeñando un papel fundamental al enfocarse en el desarrollo de una plataforma de electromovilidad. Esta plataforma busca superar la falta de un estándar común que actualmente limita la comunicación y colaboración fluida entre los Operadores de Puntos de Carga (CPO) y los Proveedores de Servicios de Movilidad Eléctrica (EMSP), dos actores esenciales en la promoción y funcionamiento de la electromovilidad. El trabajo innovador realizado en VERTEX no solo impulsa la movilidad sostenible, sino que también abre las puertas a un futuro más interconectado y colaborativo en el sector de la electromovilidad, allanando el camino hacia una infraestructura más eficiente y amigable con el medio ambiente.

Dada esta problemática, se plantea como **problema de investigación**:

¿Cómo proporcionar la interoperabilidad entre servicios externos y la plataforma de eletromovilidad de carga pública en Cuba? Se identifica, como **objeto de estudio**: servicios de interoperabilidad para la plataforma de electromovilidad de carga público en Cuba, enmarcado como **campo de acción**: servicio de interoperabilidad en el contexto de la electromovilidad y para contribuir a la solución del problema de investigación se determina como **objetivo general** de este proyecto selección e implementación de un estándar de interoperabilidad para la plataforma de carga pública en Cuba

Los **objetivos específicos** son:

- ✓ Realizar un análisis detallado de la situación actual de la infraestructura de carga de vehículos eléctricos en Cuba: Este análisis debe considerar la cantidad y ubicación de las estaciones de carga existentes, su estado de

conservación y capacidad de carga, y la demanda actual y potencial de vehículos eléctricos en el país.

- ✓ Analizar las tecnologías que mejor se adapten a las necesidades del protocolo y la plataforma.
- ✓ Definir los componentes que se utilizarán, de acuerdo a la tecnología seleccionada y el escenario en donde se desarrollará este proyecto.
- ✓ Diseñar los artefactos e implementar la propuesta de solución para resolver dicha problemática.
- ✓ Validar la solución informática propuesta aplicando diferentes pruebas.

Para facilitar el desarrollo de la investigación y además alcanzar la solución deseada se utilizaron diferentes métodos, como son:

Métodos Teóricos:

- ✓ Analítico-Sintético: La utilización de este método permitió identificar y analizar las diversas funcionalidades de las plataformas de electromovilidad a nivel internacional que pueden aplicarse en la solución, además de analizar las fuentes bibliográficas que abordan elementos relevantes para la investigación.

Métodos Empíricos:

- ✓ Observación: se utilizó para observar las distintas herramientas que permitan el diseño, implementación y evaluación del protocolo, que hizo posible realizar un análisis y comparación de las mismas, para así establecer los elementos fundamentales que debía cumplir la propuesta solución.

El presente documento se estructura en tres capítulos:

- ✓ Capítulo 1. Fundamentación teórica: este capítulo está dedicado a definir los conceptos más importantes asociados a la problemática, para la posterior compresión de la investigación, además, se realiza una valoración de las principales soluciones existentes a nivel internacional enfocadas a resolver la problemática planteada, y se explica las principales herramientas, metodologías y leguajes asociadas al tema en cuestión.

- ✓ Capítulo 2. Análisis y Diseño: el capítulo define la propuesta de solución que se propone, se describen los requisitos funcionales y no funcionales del sistema. Además, se detallan los puntos fundamentales dentro del diseño y planificación de la propuesta de solución, tales como la arquitectura del sistema y los patrones de diseño.
- ✓ Capítulo 3. Implementación y evaluación de la solución propuesta: En este capítulo se describe la implementación y posterior validación realizada al producto obtenido como solución. Se describen las pruebas realizadas al sistema para así poder validar si funciona de manera correcta y si cumple con las especificaciones planteadas.

Capítulo 1. Fundamentación Teórica

En este capítulo se expone la fundamentación teórica de este trabajo, en el cual se definirán un conjunto de conceptos muy importantes para la comprensión de la investigación, incluyéndose además en el mismo el estudio de otras soluciones informáticas existentes en la actualidad, de las metodologías, modelos de desarrollo de software y herramientas a utilizar.

1.1 Terminología y definiciones:

Antes de adentrarnos en el análisis y diseño del protocolo, es fundamental establecer un conjunto de términos técnicos y específicos que son esenciales para esta área de estudio. A continuación, se presentan las terminologías clave junto con sus definiciones correspondientes:

1.1.1 Protocolos estandarizados para el roaming de vehículos eléctricos (EV):

En La Universidad Tecnológica de Eindhoven en Países Bajos se realizó un análisis comparativo detallado de todos los protocolos como parte del proyecto evRoaming4EU, titulado 'Análisis comparativo de protocolos estandarizados para la itinerancia de vehículos eléctricos', realizado en 2020. Los autores son Mart van der Kam, quien trabaja en la estandarización de protocolos de carga de vehículos eléctricos en el mismo, y Rudi Bekkers, Profesor Titular y Catedrático de Estandarización y Propiedad Intelectual en la misma. El informe no solo presenta una comparación de los principales protocolos de roaming existentes en Europa, sino que también ofrece información detallada sobre cómo lograr la interoperabilidad desde una perspectiva de estandarización, a través de una combinación de investigación de escritorio y entrevistas con las partes interesadas.

Definición y relación entre EV roaming e interoperabilidad

La relación entre roaming e interoperabilidad en el contexto de la electromovilidad es fundamental para garantizar una experiencia de carga sin problemas para los conductores de vehículos eléctricos. El roaming en la electromovilidad se refiere a la capacidad de los conductores de vehículos eléctricos para cargar sus vehículos en estaciones de carga que no pertenecen a su proveedor de servicios de carga habitual.

Por otro lado, la interoperabilidad se refiere a la capacidad de diferentes sistemas, dispositivos y redes de carga de vehículos eléctricos para comunicarse, compartir datos y operar de manera conjunta. La interoperabilidad es esencial para garantizar que los conductores de vehículos eléctricos puedan acceder a estaciones de carga de diferentes operadores sin problemas, utilizando una única cuenta de cliente gestionada por su proveedor de servicios de movilidad eléctrica (EMSP) preferido. Esto se logra a través de estándares y protocolos de comunicación abiertos, que facilitan la conexión y comunicación entre los operadores y proveedores de servicios de recarga, permitiendo la itinerancia automatizada entre ellos.(Kan, Bekkers 2020)

El roaming implica al menos lo siguiente:

- ✓ Un acuerdo contractual entre MSP y CPOs, ya sea de forma directa (bilateral) o indirecta (a través de un centro de roaming o una casa de compensación).
- ✓ El punto de carga debe tener una conexión a Internet.
- ✓ Un lector de tarjetas RFID o una función para la activación remota.
- ✓ Protocolos de comunicación interoperables.

Selección de protocolos a investigar:

Comparación de los principales protocolos de roaming de vehículos eléctricos (EV) existentes en Europa. Se utilizaron los siguientes criterios para la selección de protocolos: (1) que el protocolo sea ampliamente utilizado en Europa, (2) que la documentación completa del protocolo esté en una forma final y sea públicamente accesible y (3) que el protocolo pueda, en principio, ser implementado por cualquier parte, utilizando los criterios mencionados anteriormente, la selección incluye los siguientes protocolos:

- ✓ Protocolo de Cámara de Compensación Abierta (OCHP).
- ✓ Protocolo de Intercambio Abierto (OICP).
- ✓ Protocolo de Interoperación de eMovilidad (eMIP).
- ✓ Interfaz de Puntos de Carga Abierta (OCPI).

Tres de los protocolos mencionados anteriormente utilizan el término "abierto" en su nombre. En el contexto de los protocolos de roaming de vehículos eléctricos (EV), este término puede hacer referencia a un acceso abierto a la infraestructura de carga, pero

también puede implicar que el protocolo sea un estándar abierto. El Comité de la Organización Mundial del Comercio sobre Obstáculos Técnicos al Comercio WTO TBT (por sus siglas en inglés) formuló las siguientes seis condiciones para los procesos de normalización internacional: (Kan, Bekkers 2020)

- ✓ Transparencia (en cuanto a la documentación sobre propuestas de estándares y estándares finales).
- ✓ Apertura (membresía abierta en cada etapa del proceso de normalización).
- ✓ Imparcialidad y consenso (sin privilegios ni favores a intereses de una parte particular).
- ✓ Efectividad y relevancia (facilitar el comercio internacional).
- ✓ Coherencia (evitar la duplicación o superposición con el trabajo de otros organismos de normalización).
- ✓ Atender las preocupaciones de los países en desarrollo (los países en desarrollo no deben ser excluidos de factor del proceso).

El grado en que los protocolos cumplen con estos criterios de apertura difiere, como se puede apreciar en la siguiente tabla que muestra una evaluación sobre cómo la gobernanza del protocolo obtiene una puntuación en los criterios de la OMC TBT para estándares abiertos, '+' indica alto, '0' indica medio y '-' indica bajo.

	OCHP	OICP	eMIP	OCPI
Transparencia	+	+	0	+
Apertura	0	0	-	Actual: +/0 Futuro: +
Imparcialidad y consenso	0	-	-	+/0
Efectividad y relevancia	+	+	+	+
Coherencia	+	+	+	+
Dimensión de desarrollo	0	0/-	0/-	0

Tabla #1: Criterio de apertura (Elaboración propia)

A partir de la investigación realizada y la comparación de OCPI con otros protocolos de roaming varios puntos caben a destacar en la decisión de seleccionar este protocolo como son:

- ✓ Desarrollo basado en la comunidad: OCPI se destaca como el único protocolo en el que el desarrollo se basa en la comunidad, lo que significa que múltiples partes pueden contribuir activamente a su mejora y evolución.
- ✓ Independencia de la organización gestora: A diferencia de otros protocolos, OCPI no es gestionado por una organización que también opera el centro de roaming asociado, lo que garantiza una mayor imparcialidad en el acceso y uso de la infraestructura de carga.
- ✓ Agnóstico en cuanto a modelos de negocio: OCPI es agnóstico en términos de modelos de negocio, lo que significa que es flexible y admite una variedad de enfoques comerciales, beneficiando a una amplia gama de actores en el mercado.
- ✓ No requiere patentes ni regalías: OCPI no exige el uso de invenciones patentadas que requieran el pago de regalías, lo que reduce los costos y barreras para su implementación.
- ✓ Posibilidad de proponer nuevas funcionalidades: OCPI permite la introducción de nuevas funcionalidades, lo que facilita la adaptación del protocolo a las necesidades cambiantes del mercado.(Kan, Bekkers 2020) (*Protocolo de carga de vehículos eléctricos y estandarización de protocolos* 2023)

En comparación con otros protocolos como OICP y eMIP, OCPI se destaca como un ejemplo más destacado de un protocolo abierto debido a su comunidad de desarrollo inclusiva, su mayor tasa de adopción percibida, sus recursos adicionales y su mayor visibilidad en el mercado. Estos factores contribuyen a que OCPI sea visto como una opción preferida en el ámbito de la infraestructura de carga para vehículos eléctricos.

1.1.2 Roles del mercado de carga de vehículos eléctricos

Según la especificación del protocolo OCPI en el mercado de carga de vehículos eléctricos, un 'rol' se refiere a la función o posición desempeñada por un actor o entidad en el ecosistema de carga de vehículos eléctricos. Cada rol tiene responsabilidades y

tareas específicas que contribuyen al funcionamiento eficiente de la infraestructura de carga y a la prestación de servicios a los usuarios de vehículos eléctricos. Los roles en el mercado de carga de vehículos eléctricos pueden incluir:

- ✓ CPO (Charging Point Operator - Operador de Puntos de Carga): Opera una red de puntos de carga.
- ✓ eMSP (e-Mobility Service Provider - Proveedor de Servicios de Movilidad Eléctrica): Ofrece a los conductores de vehículos eléctricos acceso a servicios de carga.
- ✓ Hub (Centro de Conexión): Puede conectar uno o varios CPOs con uno o varios eMSPs.
- ✓ NAP (National Access Point - Punto de Acceso Nacional): Proporciona una base de datos nacional con todas las ubicaciones de carga (públicas). La información se puede enviar y recuperar desde el NAP. Esto lo diferencia de un proveedor de servicios de navegación (NSP) típico.
- ✓ NSP (Navigation Service Provider - Proveedor de Servicios de Navegación): Proporciona a los conductores de vehículos eléctricos información de ubicación de los puntos de carga. Por lo general, solo está interesado en información de ubicación.
- ✓ Roaming Hub (Centro de Roaming): Véase: Hub (Centro de Conexión).
- ✓ SCSP (Smart Charging Service Provider - Proveedor de Servicios de Carga Inteligente): Ofrece servicios de carga inteligente a otras partes. Puede utilizar una variedad de entradas diferentes para calcular perfiles de carga inteligente.

Algunos de estos roles pueden combinarse en una sola empresa. Una plataforma puede proporcionar servicios a múltiples CPO o eMSP, pero también a ambos, eMSP y CPO.(*OCPI - Open Charge Point Interface* 2023)

OCPI 2.0 y OCPI 2.1.1 tenían una definición muy estricta de roles: solo CPO y eMSP. Pero esto es raro en el mundo real, casi no hay partes que sean estrictamente CPO o eMSP y tengan su propia plataforma. En el mundo real, muchas partes brindan servicio a CPO que no están ejecutando su propia plataforma. Muchos CPO también son eMSP. Con OCPI 2.1.1 y versiones anteriores, eso significaba tener que configurar una

conexión OCPI por rol. OCPI 2.2 introdujo más roles y abstrae el rol de la propia conexión OCPI se describen en términos de Plataformas que se conectan a Plataformas, o Plataformas que se conectan a través de Hubs a otras Plataformas. La Plataforma en sí no es un rol. La Plataforma proporciona servicios para uno o más roles. Según la especificación del protocolo en su versión 2.2. el protocolo OCPI es un conjunto de especificaciones técnicas de código abierto que definen un estándar para la interconexión de estaciones de carga eléctrica y servicios de movilidad eléctrica. Este protocolo permite la comunicación en tiempo real entre los puntos de carga eléctrica y los servicios de movilidad eléctrica, lo que facilita la gestión y el acceso a servicios de carga eléctrica. Está basado en el intercambio de mensajes en formato notación de objetos de JavaScript JSON (por sus siglas en inglés) a través de transferencia de estado representacional API RESTful (por sus siglas en inglés). Esto significa que los puntos de carga eléctrica y los servicios de movilidad eléctrica se comunican a través de solicitudes HTTP, lo que permite una comunicación en tiempo real y una integración fácil con otros sistemas y plataformas.(*OCPI - Open Charge Point Interface* 2023)

Define diferentes módulos o componentes que permiten la gestión de la información sobre los puntos de carga eléctrica y los servicios de movilidad eléctrica en su versión 2.2. Estos módulos incluyen:

Figura #1: Módulos de OCPI v2.2 (Elaboración Propia)

1.2 Sistemas homólogos

NewMotion, con sede en los Países Bajos, es una empresa líder en la movilidad eléctrica. Fundada en 2009, se ha destacado por ofrecer soluciones de carga de vehículos eléctricos en toda Europa. Su amplia red de estaciones de carga, aplicaciones móviles y enfoque en la interoperabilidad a través de OCPI han contribuido a la expansión de la movilidad eléctrica en la región. NewMotion trabaja para facilitar la transición hacia una movilidad más sostenible en Europa, estas son las principales características de esta plataforma con respecto a su uso de OCPI:

- ✓ Interoperabilidad de red: NewMotion implementa OCPI como un estándar de comunicación abierto que permite la interoperabilidad entre su propia red de carga y las redes de carga de otros operadores que también utilizan OCPI. Esto significa que los usuarios de NewMotion pueden acceder a estaciones de carga de otros operadores que participan en la red OCPI.
- ✓ Itinerancia de carga: Gracias a OCPI, los usuarios de NewMotion pueden cargar sus vehículos eléctricos en estaciones de carga de otros operadores que son parte de la red OCPI sin necesidad de registrarse en múltiples redes. Esto simplifica significativamente la experiencia de carga y brinda a los conductores más opciones y flexibilidad.
- ✓ Comunicación en tiempo real: OCPI permite que las estaciones de carga compartan información en tiempo real sobre la disponibilidad de cargadores, los precios de la carga y la ubicación de las estaciones. Los usuarios de NewMotion pueden acceder a esta información a través de la aplicación móvil y los servicios en línea de la empresa para tomar decisiones informadas sobre la carga de sus vehículos.
- ✓ Aplicación móvil y servicios en línea de NewMotion: NewMotion ofrece una aplicación móvil y servicios en línea que permiten a los conductores de vehículos eléctricos encontrar estaciones de carga cercanas, verificar la disponibilidad de cargadores, iniciar sesiones de carga y realizar pagos. Estos

servicios están diseñados para funcionar de manera integrada con las redes de carga que utilizan OCPI.

- ✓ Facilita la expansión internacional: OCPI es un estándar global, lo que facilita la expansión internacional de los servicios de carga de NewMotion. La implementación de OCPI permite que NewMotion colabore con otros operadores de carga de vehículos eléctricos en diferentes países y regiones.
- ✓ Gestión de flotas y soluciones comerciales: NewMotion también ofrece soluciones de carga para flotas y empresas. A través de OCPI, pueden proporcionar una gestión centralizada de estaciones de carga, informes de uso detallados y opciones de facturación para flotas de vehículos eléctricos comerciales.
- ✓ Compatibilidad con múltiples vehículos eléctricos: Las estaciones de carga de NewMotion son compatibles con una variedad de vehículos eléctricos y enchufes, lo que garantiza que puedan ser utilizadas por conductores de diferentes marcas y modelos de vehículos eléctricos.

En resumen, NewMotion utiliza OCPI como un estándar abierto para facilitar la interoperabilidad y la itinerancia de carga, lo que mejora la experiencia de carga para los usuarios de vehículos eléctricos. La implementación de OCPI permite que los conductores accedan y utilicen estaciones de carga en múltiples redes sin complicaciones y fomenta la expansión de los servicios de carga de NewMotion a nivel nacional e internacional.

EVBox es otra de las principales empresas en la industria de la movilidad eléctrica, y se ha destacado por proporcionar soluciones integrales de carga de vehículos eléctricos en todo el mundo. Fundada en 2010 y con sede en los Países Bajos, EVBox se ha expandido rápidamente y ha desempeñado un papel importante en la adopción de la movilidad eléctrica en diferentes mercados, estas son las principales características de esta plataforma con respecto a su uso de OCPI:

- ✓ Interoperabilidad de red: EVBox utiliza OCPI como un estándar abierto para habilitar la interoperabilidad entre su propia red de carga y otras redes de carga de vehículos eléctricos. Esto significa que los usuarios de EVBox pueden

acceder a estaciones de carga de otros operadores que también implementan OCPI, lo que amplía significativamente la accesibilidad de las estaciones de carga.

- ✓ Itinerancia de carga: Gracias a OCPI, los conductores de vehículos eléctricos pueden usar la misma tarjeta o aplicación para cargar sus vehículos en estaciones de carga tanto de EVBox como de otros operadores que son parte de la red OCPI. Esto simplifica la experiencia de carga, ya que los usuarios no necesitan registrarse en múltiples redes de carga.
- ✓ Comunicación y transparencia: OCPI permite que las estaciones de carga compartan información relevante en tiempo real, como la disponibilidad de estaciones, el precio de la carga y la ubicación de las estaciones. Esto brinda a los conductores información actualizada para tomar decisiones informadas sobre dónde y cuándo cargar sus vehículos.
- ✓ Aplicación móvil y servicios en línea de EVBox: EVBox integra la funcionalidad de OCPI en su aplicación móvil y servicios en línea. Esto significa que los usuarios de EVBox pueden usar estas herramientas para encontrar estaciones de carga cercanas, verificar la disponibilidad, iniciar sesiones de carga y realizar pagos en estaciones de carga de otros operadores que son parte de la red OCPI.
- ✓ Facilita la expansión internacional: OCPI es un estándar global, lo que facilita la expansión internacional de los servicios de carga de EVBox. Pueden colaborar y conectarse con otros operadores en diferentes regiones del mundo que también utilizan OCPI.
- ✓ Gestión centralizada: EVBox puede gestionar su red de carga y las interacciones con otras redes OCPI de manera centralizada, lo que simplifica la administración y el monitoreo de su infraestructura de carga.

En resumen, EVBox utiliza OCPI para permitir que los usuarios de vehículos eléctricos accedan y utilicen estaciones de carga más allá de su propia red, mejorando así la conveniencia y la accesibilidad de la carga. La implementación de OCPI permite una

experiencia de carga más fluida y facilita la expansión de los servicios de carga de EVBox a nivel nacional e internacional.

A continuación, se muestra una tabla con los elementos esenciales de los sistemas detallados anteriormente:

	NewMotion	EVBox
Versión de OCPI	OCPI 2.2	OCPI 2.2
Módulos OCPI	- Localizaciones - Sesiones - Comandos - Tarifas - CDRs (Registros de Carga) - Credenciales - Versiones - Tokens	- Localizaciones - Sesiones - Comandos - Tarifas - CDRs (Registros de Carga) - Credenciales - Versiones - Tokens
Roles	- Proveedor de Servicios de Movilidad Eléctrica (EMSP) - Operador de Punto de Carga (CPO) - Proveedor de Servicios de Navegación (NSP)	- Proveedor de Servicios de Movilidad Eléctrica (EMSP) - Operador de Punto de Carga (CPO) - Centro de conexión (Hub)
Amplitud de red	Más de 200,000 puntos de carga	Más de 190,000 puntos de carga

Tabla #2: Sistemas homólogos (Elaboración propia)

Al analizar detalladamente estos sistemas se puede concluir que no serían la solución óptima para resolver la problemática planteada, hay razones clave por las cuales

podría ser complicado utilizar estas plataformas como una solución de carga de vehículos eléctricos en Cuba:

- ✓ Ubicación en Europa: Tanto NewMotion como EVBox tienen una presencia y operaciones centradas en Europa. Esto significa que su infraestructura de carga y servicios se han desarrollado principalmente en ese continente. Dado que Cuba se encuentra fuera de su área principal de servicio, podría limitar la disponibilidad de estaciones de carga y la viabilidad de sus soluciones en la ubicación.
- ✓ Interoperabilidad y OCPI: Aunque ambas empresas utilizan OCPI para la interoperabilidad, su enfoque principal ha sido dentro de Europa. La adopción de OCPI en otras regiones puede ser limitada, lo que podría dificultar la integración y la itinerancia de carga en Cuba, donde la infraestructura de carga para vehículos eléctricos podría ser escasa o inexistente.
- ✓ Regulaciones y Requisitos Locales: La implementación de una solución de carga de vehículos eléctricos en Cuba podría estar sujeta a regulaciones y requisitos locales específicos. Estos requisitos pueden variar en comparación con los de Europa, lo que podría requerir adaptaciones y consideraciones adicionales en el diseño de la solución.
- ✓ Disponibilidad de Energía: La disponibilidad de energía eléctrica en Cuba también puede ser un factor limitante. La infraestructura eléctrica y las capacidades de suministro pueden variar, lo que afectaría la capacidad de proporcionar carga de alta calidad y eficiente para los vehículos eléctricos.

La propuesta de solución de carga de vehículos eléctricos en Cuba se basa en OCPI y consta de solo tres módulos, es esencial considerar cuidadosamente cómo se adaptaría a las condiciones locales y las limitaciones mencionadas anteriormente. Podría ser necesario realizar ajustes significativos y colaborar con las autoridades locales para garantizar que la implementación sea factible y exitosa en ese entorno específico.

1.3 Metodología de desarrollo de software

Al desarrollar productos o soluciones informáticas es necesario organizar el trabajo de la forma más adecuada y ordenada posible. Es por eso que recurrimos a las metodologías de desarrollo de software, que no son más que un conjunto de técnicas y métodos organizativos, los cuales deben aplicarse para el diseño de soluciones de software informático. Ellas a su vez intentan organizar los equipos de trabajo, con el fin de que estos desarrollen las funciones de un programa de la mejor manera posible. Estas también mejoran el resultado final de las aplicaciones que se desean desarrollar. Estas metodologías de software se clasifican en dos grandes grupos: las metodologías ágiles y las metodologías tradicionales. Las metodologías de desarrollo de software tradicionales tienen entre sus principales características la definición total y de forma rígida de los requisitos de software al inicio de los proyectos de ingeniería de software. Estas metodologías no se adaptan nada bien a los cambios, es por eso, que muchos proyectos prefieren utilizar las metodologías agiles. Existen varios tipos de metodologías tradicionales, como, por ejemplo: incrementales, en cascada, en espiral, prototipadas, etc. Otra de las características de estas metodologías es la forma en la que organizan el trabajo, de manera lineal, donde una etapa sucede detrás de la otra y si no se ha terminado esa etapa no se puede comenzar la siguiente.(*Metodologías de desarrollo de software* 2020)

Por otra parte, las metodologías ágiles de desarrollo de software tienen una alta flexibilidad y agilidad, lo que conlleva a que sean más utilizadas que las tradicionales. En esta metodología el cliente forma parte del equipo de desarrollo, el cual puede ir aportando nuevos requerimientos o correcciones. Otra de las características que la diferencian de las tradicionales es que si están preparadas para cambios en el proyecto. Se basan además en la metodología incremental, lo que quiere decir que en cada ciclo de desarrollo se van agregando funcionalidades nuevas a la aplicación.

Entre las principales metodologías ágiles que existen se encuentran: Kanban, Scrum, Lean, la conocida como XP o Programación Extrema y el Proceso Unificado Ágil o AUP por sus siglas en inglés. Esta última metodología ha dado paso a otras, como es

en el caso de la metodología de desarrollo para la actividad productiva de la UCI (AUP-UCI), que es una variación de la misma.(*Metodologías de desarrollo de software* 2020)

1.3.1 Metodología AUP-UCI

Para el desarrollo de la solución planteada se empleará la metodología de desarrollo para la actividad productiva de la UCI, conocida también como metodología AUP-UCI. Dicha metodología es una variación de la metodología AUP, la cual se adapta al ciclo de vida definido para la actividad productiva de la UCI. Esta metodología es la empleada en los proyectos productivos de la universidad.

La metodología AUP propone 4 fases principales: Inicio, Elaboración, Construcción y Transición, sin embargo, para los proyectos de la UCI se decidió establecer tres fases: Inicio, Ejecución y Cierre, donde la segunda fase constituye la unión de las tres últimas fases que propone AUP (Elaboración, Construcción y Transición). Además, esta metodología define cuatro escenarios para modelar el sistema en los proyectos, pero este trabajo solo utilizará el escenario 4 debido a las características de este proyecto, pues ha sido evaluado el negocio a informatizar y se ha obtenido un negocio muy bien definido, en el cual el cliente estará siempre acompañando al equipo de desarrollo. Este escenario es recomendado en proyectos no muy extensos como es el caso de este proyecto, y hace uso de las Historias de Usuarios (HU). Es por eso, que para la encapsulación de los requisitos funcionales se hará uso de esta técnica (HU). Esta metodología se guía para dar cumplimiento a las buenas prácticas en el modelo de calidad CMMI (que significa Integración de los Modelos de Madurez de Capacidades).(Sánchez 2015)

La aplicación de la metodología AUP-UCI puede conllevar numerosas ventajas. Entre estas ventajas, se destaca la capacidad de identificar de manera precisa las funcionalidades necesarias, la flexibilidad para adaptarse a cambios, un enfoque de desarrollo incremental, una comunicación efectiva a lo largo del proceso, una documentación detallada y una validación continua.

1.4 Lenguajes

A continuación, se exponen los tipos de lenguajes utilizados para el desarrollo de la solución planteada y algunas de sus principales características que los hacen útiles para la solución.

1.4.1 Lenguaje de modelado: UML

Para obtener un correcto modelo para cualquier proyecto y garantizar así una arquitectura de información estructurada, es bastante útil hacer uso de un lenguaje de modelado. Es por eso que haremos uso del lenguaje de modelado unificado (UML por sus siglas en inglés), que no es más que un estándar para representar visualmente objetos, estados y procesos dentro de un sistema. Se utiliza principalmente en el desarrollo de software orientado a objetos, y al ampliarse, también es adecuado para visualizar procesos empresariales. Los diagramas UML se utilizan para representar varios componentes del sistema. Por ejemplo: clases, relaciones entre objetos, actividades e interacciones entre objetos e interfaces.(*UML: Lenguaje Unificado de Modelado Orientado a Objetos* 2018)

En resumen, UML es un lenguaje poderoso para modelar y diseñar sistemas complejos. Proporciona una forma clara y efectiva de comunicar, diseñar, implementar y documentar el sistema, lo que contribuye a un desarrollo exitoso y una comprensión compartida entre los equipos de desarrollo y las partes interesadas.

1.4.2 Lenguaje de programación: Python

Python es un lenguaje de programación de alto nivel, el cual es utilizado para desarrollar aplicaciones de todo tipo. Es un lenguaje interpretado o de script, con una sintaxis muy limpia y que favorece un código legible, ya que tiene bastante similitud con el lenguaje humano. Es de código abierto lo que lo hace gratuito, permitiendo desarrollar software sin límites. Es además muy útil ya que permite trabajar con inteligencia artificial, aprendizaje automático, big data, y data science entre otros campos.(*Metodologías de desarrollo de software* 2020)

Python es una elección sólida para la implementación de OCPI en la carga de vehículos eléctricos debido a su facilidad de uso, amplia comunidad, bibliotecas

disponibles y capacidad de integración. Puede ayudar a acelerar el desarrollo y garantizar una implementación efectiva y eficiente.

- ✓ Facilidad de Aprendizaje y Uso: Python se destaca por su sintaxis clara y legible, lo que facilita su aprendizaje y uso para programadores principiantes y experimentados. Esto puede acelerar el desarrollo y reducir la curva de aprendizaje para los miembros del equipo.
- ✓ Amplia Comunidad y Bibliotecas: Python cuenta con una comunidad activa y una amplia variedad de bibliotecas y frameworks que facilitan el desarrollo de aplicaciones complejas. Puedes encontrar bibliotecas para tareas específicas relacionadas con OCPI, como el manejo de comunicaciones y la manipulación de datos.
- ✓ Portabilidad: Python es un lenguaje multiplataforma, lo que significa que puedes escribir código en una plataforma y ejecutarlo en otras sin necesidad de realizar modificaciones importantes. Esto es útil en el contexto de la implementación de OCPI, ya que podría haber diferentes sistemas operativos y plataformas involucradas.
- ✓ Eficiencia de Desarrollo: Python es conocido por su productividad y velocidad de desarrollo. Puedes escribir y probar código de manera rápida, lo que es beneficioso cuando se requiere una implementación ágil de OCPI.
- ✓ Soporte para Integración: Python ofrece una amplia gama de herramientas y bibliotecas para la integración con otros sistemas y servicios. Puedes conectar fácilmente tu implementación de OCPI con sistemas de gestión de estaciones de carga, bases de datos, sistemas de facturación, y más.
- ✓ Escalabilidad: Aunque Python es un lenguaje interpretado, hay opciones disponibles para mejorar el rendimiento y la escalabilidad, como la compilación just-in-time (JIT) y la paralelización. Esto puede ser importante en implementaciones de OCPI que deben manejar un gran volumen de transacciones y comunicaciones.
- ✓ Comunidad de Desarrollo Activa: Python tiene una comunidad de desarrollo activa que constantemente trabaja en mejoras y actualizaciones. Esto garantiza

que el lenguaje se mantenga relevante y actualizado a medida que las necesidades evolucionan.

- ✓ Código Abierto y Gratuito: Python es un lenguaje de código abierto y gratuito, lo que significa que no hay costos de licencia asociados con su uso. Esto puede ser ventajoso desde una perspectiva de costos.

1.5 Herramientas a utilizar

Es importante conocer las herramientas que serán utilizadas para el desarrollo de cualquier proyecto pues nos permitirán un mejor desarrollo e implementación del mismo. A continuación, se exponen las herramientas a utilizar para este proyecto.

1.5.1 Herramienta de modelado: Visual Paradigm

Una de las herramientas de modelado más utilizadas actualmente es Visual Paradigm. Dicha herramienta es de gran ayuda para los equipos de desarrollo de software ya que permite capturar los requisitos correctos y transformarlos en diseños precisos, propiciando así la creación de un software adecuado según los requisitos. Además, es una herramienta con mucha aceptación por parte de los usuarios que la utilizan.(*Visual Paradigm* 2018)

Visual Paradigm es una herramienta de modelado y diseño de software que es útil en la implementación de proyectos de desarrollo de software, incluyendo la implementación de OCPI en la carga de vehículos eléctricos, debido a su capacidad de modelado visual, soporte para UML, colaboración en equipo, generación de código, documentación integrada y más. Simplifica el proceso de diseño y desarrollo y mejora la colaboración en proyectos de implementación de OCPI.

1.5.2 Framework: Flask

Flask es un popular marco de desarrollo web en Python que es ampliamente utilizado para crear aplicaciones web y API RESTful de manera eficiente y sencilla. Este marco se caracteriza por su simplicidad y agilidad, lo que lo convierte en una elección atractiva para una variedad de proyectos, incluyendo la implementación de la Norma de Interfaz de Carga de Vehículos Eléctricos (OCPI).

Flask es apreciado por su capacidad para adaptarse a las necesidades del desarrollador y la aplicabilidad en proyectos de diferentes tamaños y complejidades. Algunas de las razones por las que Flask es considerado un marco adecuado para la implementación de OCPI:

- ✓ Simplicidad y Agilidad: Flask se destaca por su enfoque minimalista y su facilidad de uso. Esto permite a los desarrolladores crear rápidamente API RESTful funcionales sin una curva de aprendizaje pronunciada.
- ✓ Documentación Abundante: Flask cuenta con una documentación extensa y una comunidad activa. Esto facilita a los desarrolladores acceder a recursos y ejemplos que son específicos para la creación de API RESTful con Flask.
- ✓ Flexibilidad y Adaptabilidad: Flask brinda la flexibilidad de estructurar una API de acuerdo con los requisitos específicos de OCPI y seleccionar componentes adicionales según sea necesario.
- ✓ Pruebas Unitarias y de Integración: Flask ofrece herramientas que simplifican las pruebas unitarias y de integración, permitiendo a los desarrolladores garantizar que su API OCPI funcione de manera confiable.
- ✓ Escalabilidad y Despliegue Sencillo: Flask es adecuado para proyectos de diversos tamaños y, en caso de necesitar escalabilidad, es posible optimizar y expandir la API.
- ✓ Despliegue Versátil: Flask se puede desplegar en diversos servidores web y plataformas en la nube, facilitando la implementación y la gestión de la API OCPI.

En resumen, Flask es una elección sólida para la implementación de OCPI debido a su simplicidad, adaptabilidad y comunidad activa. Este marco permite a los desarrolladores crear API RESTful de manera eficiente, satisfaciendo las necesidades de los usuarios y socios de OCPI, al tiempo que mantiene la flexibilidad y la agilidad en el proceso de desarrollo.

1.5.3 Entorno de desarrollo integrado (IDE): Visual Studio Code

Visual Studio Code (VS Code) es un entorno de desarrollo integrado (IDE) de código abierto desarrollado por Microsoft. Aunque el nombre puede sugerir una relación con el entorno de desarrollo completo llamado "Visual Studio", VS Code es en realidad un editor de código más ligero y altamente personalizable que está diseñado para ser una herramienta versátil para programadores y desarrolladores.(*Visual Studio Code* 2023)

Visual Studio Code es una herramienta versátil, gratuita, y altamente personalizable que puede ser una elección sólida para el desarrollo de software en proyectos, incluyendo la implementación de OCPI en la carga de vehículos eléctricos. Ofrece una amplia variedad de características y extensiones que pueden ser valiosas para simplificar el proceso de desarrollo y mejorar la productividad del equipo.

- ✓ Gratuito y de Código Abierto: Visual Studio Code es gratuito y de código abierto, lo que significa que no hay costos de licencia asociados con su uso. Esto puede ser ventajoso desde una perspectiva de costos en proyectos de desarrollo.
- ✓ Amplia Compatibilidad: VS Code es compatible con una amplia variedad de lenguajes de programación, incluyendo Python y otros que pueden ser relevantes para la implementación de OCPI. Esto permite trabajar con múltiples tecnologías en un solo entorno.
- ✓ Extensible y Personalizable: Visual Studio Code es altamente extensible y personalizable. Se pueden agregar extensiones y complementos específicos para satisfacer las necesidades del proyecto de implementación de OCPI.
- ✓ Integración con Control de Versiones: VS Code ofrece integración con sistemas de control de versiones como Git, lo que facilita el seguimiento de cambios en el código y la colaboración en equipo. Esto es esencial en proyectos de desarrollo colaborativo, como la implementación de OCPI.
- ✓ Depuración y Pruebas: Visual Studio Code incluye potentes herramientas de depuración que facilitan la identificación y corrección de errores en el código. También es compatible con marcos de pruebas, lo que es útil para garantizar la calidad de la implementación de OCPI.

- ✓ Autocompletado y Ayuda en la Escritura de Código: VS Code ofrece características de autocompletado y ayuda en la escritura de código que aumentan la productividad y ayudan a evitar errores de programación.
- ✓ Soporte para Contenedores y Ambientes Virtuales: Se puede utilizar Visual Studio Code para trabajar con contenedores y ambientes virtuales, lo que es relevante en la implementando OCPI en un entorno que utiliza contenedores Docker u otras tecnologías similares.
- ✓ Multiplataforma: VS Code está disponible en múltiples plataformas, incluyendo Windows, macOS y Linux, lo que significa que se puede desarrollar en el entorno de elección sin importar el sistema operativo.
- ✓ Gran Comunidad y Ecosistema: Visual Studio Code tiene una gran comunidad de usuarios y desarrolladores, lo que significa que se puede encontrar ayuda, tutoriales y extensiones fácilmente en línea.

1.5.4 Gestor de base de datos: PostgreSQL

PostgreSQL es un potente sistema de base de datos relacional de objetos de código abierto que utiliza y extiende el lenguaje SQL combinado con muchas características que almacenan y escalan de manera segura las cargas de trabajo de datos más complicadas

PostgreSQL viene con muchas características destinado a ayudar a los desarrolladores a crear aplicaciones, administradores para proteger la integridad de los datos y crear entornos tolerantes a fallas, y ayudarlo a administrar sus datos sin importar cuán grande o pequeño sea el conjunto de datos. Además de ser libre y de código abierto, PostgreSQL es altamente extensible. ¡Por ejemplo, puede definir sus propios tipos de datos, crear funciones personalizadas, incluso escribir código desde diferentes lenguajes de programación sin volver a compilar su base de datos!.(*PostgreSQL* 2023)

PostgreSQL es una elección sólida para la gestión de datos en la implementación de OCPI en la carga de vehículos eléctricos debido a su fiabilidad, extensibilidad, soporte para datos geoespaciales, transacciones ACID, escalabilidad, seguridad avanzada,

amplia comunidad y costo efectivo. Proporciona una base de datos robusta y confiable para almacenar y gestionar datos críticos en este contexto. Se utiliza PostgreSQL en la implementación de OCPI en la carga de vehículos eléctricos debido a sus ventajas significativas como sistema de gestión de bases de datos.

1.5.5 Herramienta de prueba de API: Insomnia

Insomnia es una aplicación cliente de código abierto que se utiliza principalmente para probar y depurar APIs (Interfaces de Programación de Aplicaciones). Es una herramienta útil para desarrolladores de software y equipos de desarrollo web que necesitan interactuar con servicios web y API de terceros, así como para aquellos que trabajan en el desarrollo de sus propias API. Utilizar Insomnia como herramienta de prueba en la implementación de OCPI presenta una serie de ventajas:

- ✓ Facilita la Prueba y Depuración: Insomnia proporciona una interfaz intuitiva que simplifica la creación, prueba y depuración de solicitudes OCPI. Esto agiliza el proceso de desarrollo y garantiza que las solicitudes se realicen y se interpreten correctamente.
- ✓ Documentación Interactiva: Insomnia permite documentar las solicitudes OCPI de manera interactiva. Esto facilita la comprensión de los puntos finales y su uso, lo que es beneficioso para la comunicación con otros miembros del equipo y colaboradores.
- ✓ Gestión de Entornos: La capacidad de gestionar múltiples entornos en Insomnia es especialmente útil. Permite probar la implementación en diferentes configuraciones, como entornos de desarrollo, pruebas y producción, sin tener que reconfigurar manualmente las solicitudes en cada uno.
- ✓ Automatización de Pruebas: Insomnia permite la creación y ejecución de pruebas automatizadas. Esto es fundamental para verificar que la implementación OCPI cumple con los estándares y funciona de manera fiable.
- ✓ Facilita la Colaboración: La posibilidad de compartir colecciones de solicitudes con el equipo de desarrollo simplifica la colaboración. Asegura que todos utilicen

las mismas solicitudes y configuraciones OCPI, lo que mejora la coherencia del proyecto.

- ✓ Autenticación y Seguridad: Insomnia admite diversos métodos de autenticación, lo que es esencial para garantizar la seguridad de los puntos finales OCPI. La configuración de la autenticación en las solicitudes es sencilla y asegura su correcto funcionamiento.

1.4 Conclusiones del capítulo

Una vez desarrollado este capítulo, fue posible alcanzar un mayor entendimiento de todo lo relacionado a la solución, sus principales conceptos, y el estudio del estado del arte sobre las tecnologías similares, por lo que fue posible arribar a las siguientes conclusiones

- ✓ El análisis de los conceptos fundamentales asociados al proceso de roaming permitió identificar elementos fundamentales para definir posteriormente la propuesta de solución, dando a entender por qué el protocolo OCPI es el más adecuado a implementar.
- ✓ La identificación de las características propuestas para la solución, el análisis de las tecnologías que distinguen a soluciones similares, contribuyó a la selección de las herramientas y tecnologías más adecuadas para desarrollar la propuesta de solución, destacando como metodología de desarrollo AUP-UCI, como herramienta de modelado Visual Paradigm usando como lenguaje UML, como lenguaje de programación Python con el framework Flask , como gestor de base de datos PostgreSQL y como aplicación para probar y depurar APIs Insomnia

Capítulo 2. Análisis y Diseño

En este capítulo se abordará la propuesta de solución a la problemática planteada, haciendo énfasis en las actividades realizadas en el proceso de análisis y diseño de dicha propuesta. Se describen también los requisitos funcionales y no funcionales, para así dar cumplimiento a los objetivos planteados con este proyecto. Además, se exponen artefactos importantes como son las historias de usuario para la especificación de los requisitos, entre otros.

2.1 Especificación de requisitos

La especificación de requisitos consiste en la actividad de transcribir la información recopilada durante la actividad de análisis, en un documento que define un conjunto de requisitos. Es una de las actividades más complejas de la ingeniería de requisitos y a su vez muy importante, ya que en ella se definen los requisitos funcionales y no funcionales del software.

2.1.1 Requisitos funcionales

RF1: Obtener información sobre las versiones disponibles.
RF2: Obtener información sobre los puntos finales para las versiones.
RF3: Obtener credenciales.
RF4: Adicionar credenciales.
RF5: Actualizar credenciales.
RF6: Eliminar credenciales.
RF7: Obtener las ubicaciones.
RF8: Obtener una ubicación especifica.
RF9: Obtener una estación de carga especifica.
RF10: Obtener un conector especifico.
RF11: Obtener una ubicación existente.
RF12: Adicionar una nueva ubicación.
RF13: Adicionar una nueva estación de carga.
RF14: Adicionar un nuevo conector.
RF15: Actualizar una ubicación.

RF16: Actualizar una estación de carga.

RF17: Actualizar un conector.

RF18: Actualizar parcialmente una ubicación.

RF19: Actualizar parcialmente una estación de carga.

RF20: Actualizar parcialmente un conector.

2.1.2 Requisitos no funcionales

Rendimiento

RNF1: Tiempo de respuesta: debe proporcionar un tiempo de respuesta promedio de menos de 3 segundos para solicitudes estándar.

Diseño e implementación

RNF2: El sistema debe utilizar como gestor de base de datos PostgreSQL en su versión 11.0.

RNF3: El sistema debe emplear como lenguaje de programación Python en su versión 3.11.

Seguridad

RNF4: Debe admitir para autorización Bearer Token, asegurando que solo los roles autorizados puedan acceder a los recursos.

Confiabilidad

RNF5: Debe devolver mensajes de error significativos y seguir las mejores prácticas de manejo de errores, como el uso de códigos de estado HTTP apropiados.

2.2 Propuesta de solución

En función de dar cumplimiento al objetivo planteado y ajustándose a las necesidades presentes se determina desarrollar una API RESTful basada en el protocolo OCPI, con tres módulos bien definidos (Credenciales, Localizaciones y Versiones), que se sustenta en varios aspectos esenciales.

En primer lugar, la estandarización y la conformidad con el protocolo OCPI son pilares fundamentales de esta propuesta. La implementación de una API siguiendo estas directrices garantiza que cumpla con los estándares definidos por OCPI en su versión

2.2, lo que facilita la interoperabilidad y promueve la adopción del protocolo en el ámbito de la carga de vehículos eléctricos.

La segmentación en módulos lógicos es otro argumento clave. Dividir la API en tres módulos independientes (Credenciales, Localizaciones y Versiones), es una estrategia de diseño sólida. Cada módulo representa una entidad o funcionalidad específica, lo que simplifica el desarrollo, el mantenimiento y la escalabilidad de la API. Además, esta segmentación minimiza el riesgo de errores y fallos inesperados, ya que los cambios en un módulo no afectarán necesariamente a los demás.

La definición de puntos finales específicos para cada módulo es un tercer punto importante. Esto proporciona una estructura organizada y lógica para acceder y manipular los datos relacionados con cada módulo, lo que facilita su uso y comprensión.

El módulo Credenciales se dedica a la gestión de credenciales y la autenticación, resaltando su importancia en términos de seguridad. Al separarlo de los otros módulos, se garantiza una atención especializada en la protección de datos sensibles, lo cual es crucial en cualquier implementación de API.

El módulo Localizaciones se ocupa de la gestión de información sobre estaciones de carga y puntos de recarga, centrándose en operaciones relacionadas con la localización, la disponibilidad y la administración de cargadores eléctricos.

Finalmente, el módulo Versiones se encarga de gestionar las versiones implementadas y los puntos finales necesarios para acceder a estas. Esto facilita a los usuarios de la API seleccionar la versión más adecuada a sus necesidades y acceder a la funcionalidad específica que requieren en función de la versión del protocolo OCPI que utilizan.

2.3 Historias de usuario

Las historias de usuario son la técnica utilizada para especificar los requisitos del software, las cuales son parte de un enfoque ágil. Se podrían definir como descripciones cortas y simples, de una característica contada desde la perspectiva de la persona que desea la nueva capacidad o funcionalidad, el cual puede ser un usuario o cliente del sistema, entre otros. Se escriben en fichas o notas adhesivas, las cuales

se almacenan en una caja y se organizan en paredes o mesas para facilitar la planificación y el debate. Estas cambian fuertemente el enfoque de escribir sobre las características a discutir y además se pueden escribir con distintos niveles de detalle. (SCRUM MÉXICO, 2018)

2.3.1 Descripción de las historias de usuarios

Se realiza la selección del RF4, RF12 y RF15 para ser desarrollados a continuación.

Historia de usuario	
Número: 4	**Nombre:** Obtener una ubicación especifica
Usuario: EMSP	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	
Descripción: El EMSP mediante un punto final estructurado de la siguiente forma {locations_endpoint_url}/{location_id} puede obtener la información en forma de JSON de la ubicación correspondiente a partir del id proporcionado en el punto final. En caso de que la ubicación no exista se mostrara un mensaje “Ubicación no encontrada”. .	
Observaciones: Documentación oficial de OCPI	

Tabla #3: Historia de usuario # 4(Elaboración propia)

Historia de usuario	
Número: 8	**Nombre:** Actualizar una ubicación y Adicionar una ubicación
Usuario: CPO	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	

Descripción: El CPO mediante un punto final estructurado de la siguiente forma {locations_endpoint_url}/{country_code}/{party_id}/{location_id} puede actualizar la información en forma de JSON, en el caso del country_code y el party_id toma la información del punto final y el cuerpo del JSON debe incluir: publish, name, address, city, postal_code, country, latitude, longitude, parking_type, operator_name y time_zone de la ubicación correspondiente a partir del id proporcionado en el punto final y muestra el mensaje “Ubicación {location_id} actualizada correctamente.”. En caso de que la ubicación no exista debe insertarla. No se incluye el campo last_updated ya que esta toma la fecha actual
Observaciones: Documentación oficial de OCPI

Tabla #4: Historia de usuario # 8(Elaboración propia)

2.4 Descripción de la arquitectura

El diseño arquitectónico se enfoca en la organización de un sistema y el diseño de la estructura global del mismo. La arquitectura de software es un elemento muy importante ya que afecta el desempeño y la potencia, así como la capacidad de distribución y mantenimiento de un sistema. Además, los requisitos no funcionales dependen de la arquitectura del sistema, o como se podría decir también, la forma en que dichos componentes se organizan y comunican. La arquitectura de un sistema de software puede basarse en un patrón o un estilo arquitectónico particular, el cual no es más que una descripción de una organización del sistema, como por ejemplo una organización cliente-servidor o una arquitectura por capas.(Sommerville 2011)

2.4.1 Patrón arquitectónico cliente servidor

El patrón arquitectónico cliente-servidor es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Este patrón se utiliza para sistemas distribuidos y se basa en la comunicación a través de una red. La arquitectura cliente-servidor se compone de dos componentes: el proveedor y el consumidor, y la separación entre cliente y servidor es una separación de tipo lógico.(Blancarte 2023)

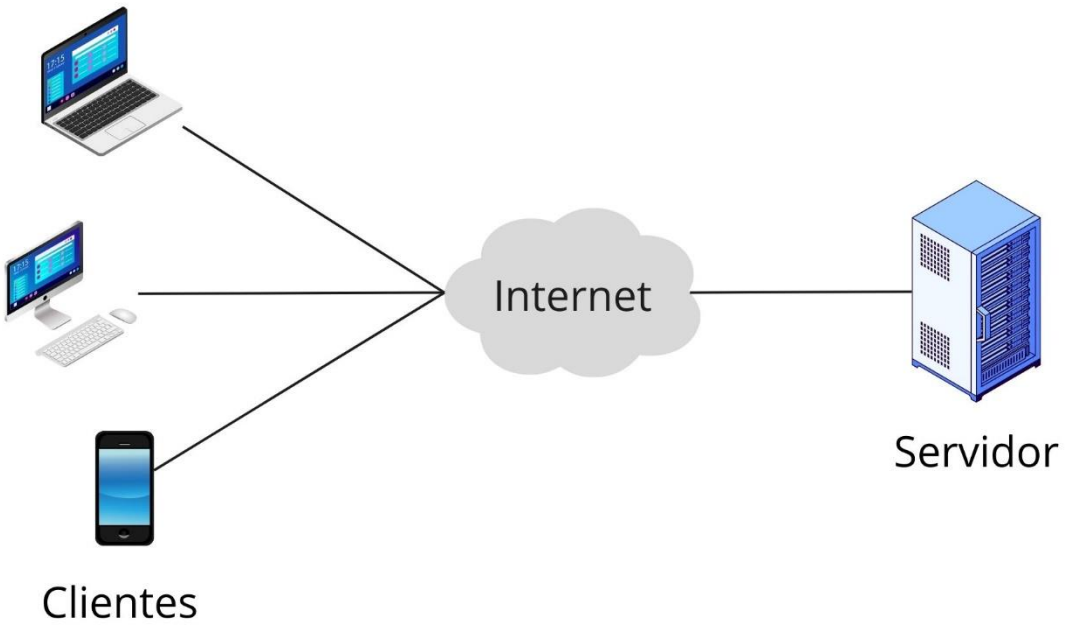


Figura #2: Modelo Cliente Servidor (Elaboración Propia)

2.5 Patrones de diseño

Cuando hablamos de patrón usualmente nos referimos a una descripción del problema y la esencia de su solución, de manera que la solución pueda reutilizarse en diferentes configuraciones. Son una forma de reutilizar el conocimiento y la experiencia de otros diseñadores. Los patrones de diseño se asocian usualmente con el diseño orientado a objetos, y soportan reutilización de concepto de alto nivel. Para su uso es necesario reconocer que cualquier problema de diseño que se enfrente es posible que tenga un patrón asociado para aplicarse. Es por eso que el uso de patrones se considera un proceso de diseño que, con frecuencia implica el desarrollo de un diseño, experimentar un problema, y luego reconocer que puede usarse un patrón.(Sommerville 2011)

El empleo de estos patrones les permite a los desarrolladores utilizar buenas prácticas de programación y ahorrar tiempo y recursos. A continuación, se describen los principales patrones empleados en el diseño e implementación de la solución.

2.5.1 Patrones GRASP:

Los patrones GRASP significan según sus siglas en inglés General Responsability Assignment Software Patterns, es decir, patrones generales de asignación de responsabilidades. Estos ayudan a identificar qué responsabilidad deben tener las clases de nuestro sistema.

GRASP se compone de varios patrones, pero los cinco patrones básicos, altamente importantes y a su vez utilizados en el diseño e implementación de la solución son:

1. Experto: Este patrón nos dice que la responsabilidad de hacer una acción debe ser asignada a la clase que tiene la información necesaria para realizar dicha acción. Tiene entre sus beneficios la reutilización de código y también evita la repetición del mismo, mantiene el encapsulamiento de la información y se distribuye el comportamiento entre las clases, lo que lleva a clases más cohesivas, o lo que es lo mismo, más fáciles de entender y mantener.(Echazarreta 2021)
2. Creador: Este patrón está enfocado en asignar la responsabilidad correcta para crear instancias de un objeto, es decir, realizar las llamadas a los objetos cuando van a ser utilizados. La clase creadora debe ser una clase experta o debe utilizar directamente el objeto, o contener la clase donde se construye el objeto.(Lago 2022)
3. Controlador: Mientras más controladores existan, más fácil es el mantenimiento del código fuente y la reutilización. El patrón Controlador se utiliza para asignar responsabilidades a un objeto que representa la interacción del usuario con el sistema. En resumen, el patrón Controlador busca separar la presentación del negocio en el desarrollo de software y se utiliza para asignar responsabilidades a un objeto que representa la interacción del usuario con el sistema.(Lago 2022)
4. Alta cohesión: Este patrón está estrechamente ligado al bajo acoplamiento. Se refiere a la medida en que una clase realiza solo la tarea para la cual fue diseñada, es decir, mientras más especializada sea una clase, más alta será lo

cohesión. Asigna a las clases responsabilidades que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad.(Lago 2022)

5. Bajo acoplamiento: Este patrón asigna las responsabilidades de forma tal que la relación que exista entre las clases sea simple, o muy poca, es decir, que las clases se comuniquen con el menor número de clases que sea posible. El mismo no puede considerarse independiente de la alta cohesión ya que la existencia de uno, implica al otro. (Lago 2022)

2.5.2 Patrones GOF:

Los patrones de diseño del grupo GOF (Gang of Four), se clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. A continuación, se describe los patrónes GOF utilizado en la solución propuesta:

Patrón de Composición: Permite tratar de la misma manera a objetos simples y a grupos de objetos, lo que permite que los objetos sean más fáciles de mantener y modificar, se utiliza para construir objetos complejos a partir de objetos más simples y pequeños, lo que permite que los objetos sean más fáciles de mantener y modificar, es especialmente interesante para una amplia variedad de programas, como estructuras de árbol binarias.(*Patrón Composite* 2021)

Patrón de Manejo de Estado (State Design Pattern): permite que un objeto altere su comportamiento cuando su estado interno cambia. Este patrón se utiliza para encapsular comportamientos variables para el mismo objeto, basado en su estado interno se utiliza para mejorar la mantenibilidad del código y evitar el uso de declaraciones condicionales, permite que los objetos cambien su comportamiento en tiempo de ejecución sin recurrir a declaraciones condicionales.(Shivets 2021)

2.6 Diagrama de clases:

Los diagramas de clase UML se usan para modelar la estructura estática de las clases de objetos, es decir, para mostrar las clases en un sistema y las asociaciones entre dichas clases. Suele usarse para sistemas orientados a objetos. Estas clases pueden tener tanto atributos como métodos.(Sommerville 2011)

A continuación, se puede observar el diagrama de clases del módulo Locations:

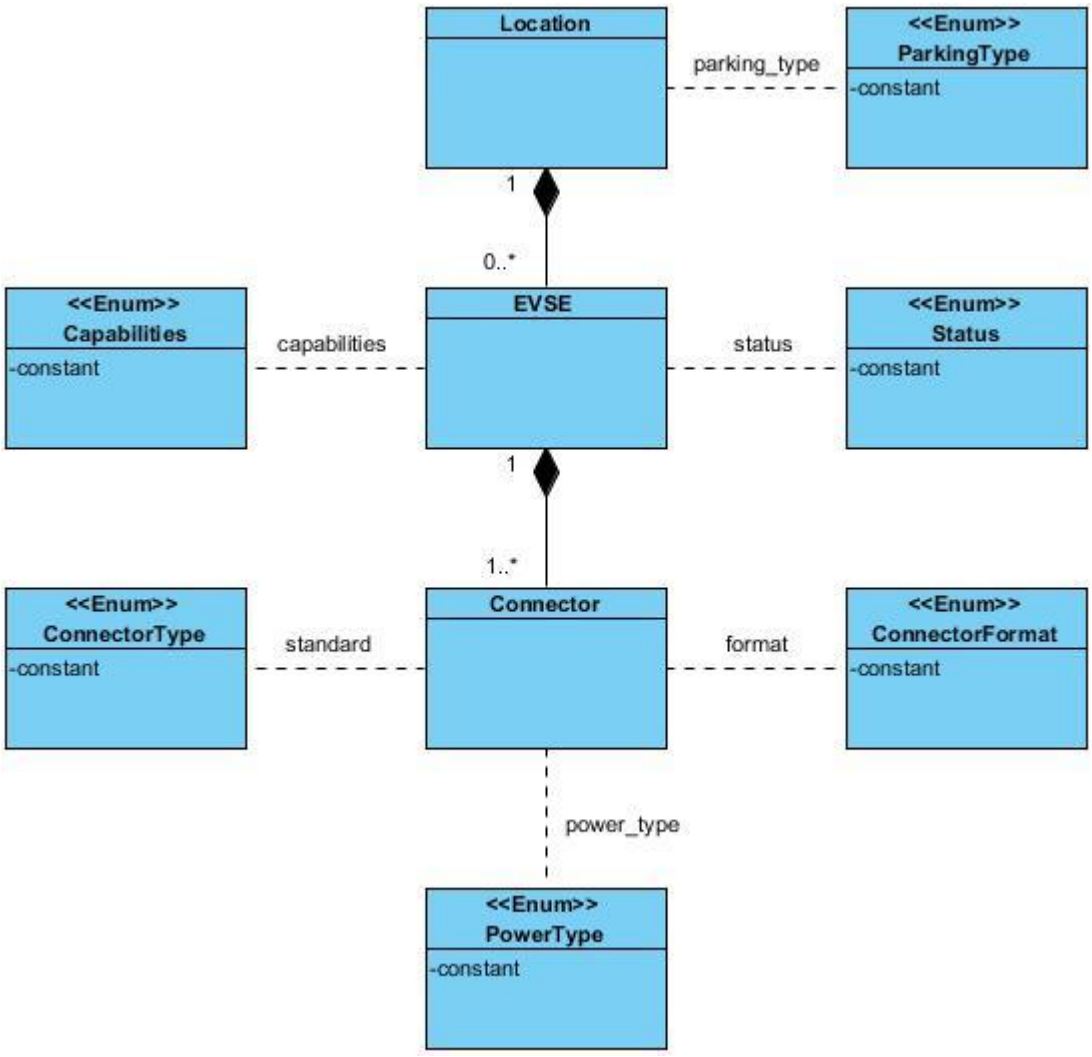


Figura #3: Diagrama de clases Modulo Locations (Elaboración Propia)

Location: La clase "Location" representa un lugar específico donde se encuentra una infraestructura de carga para vehículos eléctricos. Puede haber múltiples ubicaciones, por lo que la relación entre "Location" y "EVSE" se muestra como "1" a "*," lo que significa que una ubicación puede tener cero o más Estaciones de Carga de Vehículos Eléctricos (EVSE, por sus siglas en inglés).

EVSE: La clase "EVSE" representa una estación de carga de vehículos eléctricos en una ubicación específica. Cada ubicación puede tener una o más estaciones de carga. La relación entre "EVSE" y "Connector" se muestra como "1" a "1..n", lo que significa que una estación de carga puede tener uno o más conectores de carga.

Connector: La clase "Connector" representa los conectores de carga en una estación de carga de vehículos eléctricos. Cada estación de carga (EVSE) puede tener uno o más conectores (por ejemplo, diferentes tipos de conectores para vehículos eléctricos).

2.7 Conclusiones del capítulo:

Después de realizado el análisis y diseño durante todo el desarrollo de la propuesta de solución, se puede arribar a que:

- ✓ Fueron definidos 20 requisitos funcionales, los cuales fueron descritos haciendo uso de las Historias de Usuario.
- ✓ Fueron definidos 3 requisitos no funcionales, los cuales describen las cualidades o características que tendrá el sistema.
- ✓ Se definió la propuesta de solución, logrando llegar a un mejor entendimiento de los módulos a desarrollar.
- ✓ Se definió la arquitectura del sistema la cual es modular y 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 evaluación de la solución propuesta:

En este capítulo se abordarán las diferentes pruebas realizadas al software para determinar así el correcto funcionamiento y la calidad de la propuesta que se ha desarrollado. También se describirá el estándar de codificación utilizado y se detallan los diferentes resultados arrojados por cada prueba realizada y la respuesta a dichas no conformidades.

3.1 Estándar de codificación

En el proceso de desarrollo de un software es necesario incorporar principios sólidos para la programación en sus respectivos lenguajes, para lograr así que el código siempre esté a la altura. Es por eso que es necesario hacer uso de los estándares de codificación, que no son más que un conjunto no formal de reglas, que han ido surgiendo en las distintas comunidades de desarrolladores con el paso del tiempo y que pueden incrementar notablemente la calidad de nuestro código si son bien aplicadas. Es por eso que se han aplicado los estándares definidos por las convenciones de escritura de código Python, utilizándose el estándar PEP 8.

Este estándar aborda temas como el nombramiento de las clases, funciones y variables y cubre todos los aspectos de estilos que pueden surgir mientras programamos por lo que el estándar completo es un poco extenso.(*PEP 8* 2020)

Entre las reglas más importantes que define están:

1. Reglas de identación:

PEP 8 recomienda indentar usando 4 espacios, ya que es un número aceptable para la separación visual de los bloques. Aquí tenemos un ejemplo:

Código fuente: Identación de 4 espacios

```
def get_versions():
    try:
        versions = VersionModel.get_versions()
        return jsonify(versions)
    except Exception as ex:
        return jsonify({'message': str(ex)}), 500
```

2. Longitud de líneas:

Aunque permite cierta flexibilidad, recomienda limitar todas las líneas a un máximo de 79 caracteres. Sin embargo, para largos bloques de texto, como comentarios o documentación, recomienda 72 caracteres por línea. Por ejemplo:

Código fuente: Líneas limitadas.

```
def get_evse_details_cpo(loc_id, uid):
    try:
        location = LocationCpoModel.get_location(loc_id)

        if location is None:
            return jsonify({'message': 'Ubicacion no encontrada'}), 404

        evse = LocationCpoModel.get_evse(loc_id, uid)

        if evse is not None:
            return jsonify(evse)
        else:
            return jsonify({'message': 'EVSE no encontrado'}), 404
    except Exception as ex:
        return jsonify({'message': str(ex)}), 500
```

3. Convenciones de nombres:

Para nombrar clases se usa el Formato Capital, donde la primera letra de cada palabra es mayúscula. Clases para uso interno tienen un guion bajo como prefijo. Se recomienda *evitar los caracteres***:** o, O, l y L (letra O y letra L) dado que se pueden confundir con un 0 o un 1 dependiendo del tipo de fuente usada. Los nombres de *variables y funciones* se usan palabras en minúsculas separadas por un guion bajo, aplicándose éstos tanto como sea necesario para mejorar la legibilidad. Las *constantes* deben ser en mayúsculas. Ejemplo: Código fuente: Nombre de clases y de variables.

```
class Version():
     def __init__(self, version_number, url=None) -> None:
        self.version_number = version_number
        self.url = url
```

Código fuente: Nombre de funciones.

```
def get_versions(self):
    try:
        connection = get_connection()
```

```
    versions = []

    with connection.cursor() as cursor:
        cursor.execute("SELECT  version , url FROM versiones")
        resultset = cursor.fetchall()

        for row in resultset:
            version = Version(row[0], row[1])
            versions.append(version.to_JSON())

    connection.close()
    return versions
  except Exception as ex:
    raise Exception(ex)
```

4. Comentarios:

La PEP 8 contempla muchas reglas sobre cómo tratar los comentarios. Entre ellas están: deben de componerse de frases completas; comenzar en mayúscula, con excepción de que comiencen con el nombre de un identificador donde siempre se respetará su tamaño; los comentarios de línea comienzan con al menos 2 espacios y un # y un espacio después. Por ejemplo: Código fuente: Comentarios de una línea.

```
#Resiver_Interface eMSP
@main.route('/emsp/location/<string:country_code>/<string:party_id>/<string:loc_id>', methods=['GET'])
def get_location_details_emsp(country_code, party_id, loc_id):
    try:
        location = LocationEmspModel.get_location(country_code, party_id, loc_id)
        if location is not None:
            return jsonify(location)
        else:
            return jsonify({'message': 'Ubicación no encontrada'}), 404
    except Exception as ex:
        return jsonify({'message': str(ex)}), 500
```

3.2 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 probar un sistema o aplicación con el propósito comprobar su correcto funcionamiento. 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 automática, 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)

3.2.1 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, participando en las mismas

3.2.2 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.

3.3 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. 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.(Larrea 2019)

3.3.1 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 Actualizar Parcialmente una Ubicación. Para ver los demás casos de prueba consultar Anexo 2.

Sección: Modificación Parcial de Ubicación CPO								
Id del escenario	**Escenario**	**Country_id**	**Party_id**	**Location_id**	**publish**	**name**	**Respuesta del sistema**	**Resultado de la prueba**
10.1	Modificación exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra el mensaje "Ubicación LOC2 actualizada correctamente."	Se modificó satisfactoriamente la ubicación.
		V	V	V	V	V		
		CU	CUB	LOC2	falce	Gent Z		
10.2	Campos vacíos	N/A	N/A	N/A	N/A	N/A	Muestra un mensaje "No se proporcionaron campos para actualizar."	Se alertó satisfactoriamente la presencia de campos vacíos.
10.3	Datos incorrectos	V	V	V	I		Muestra un mensaje indicando cuales el campo incorrecto	Se alertó satisfactoriamente la presencia de campos incorrectos.
		CU	CUB	LOC2	1			
10.4	Ubicación no existente	V	V	V	V	V	Muestra el mensaje "La ubicación no existe con los datos proporcionados"	Se modificó satisfactoriamente cuando no se encuentra la ubicación.
		CU	CUB	LOC4	True	Fred		

Tabla #5: Caso de Prueba: Modificar Parcialmente Ubicación (Elaboración Propia)

3.3.2 Pruebas unitarias:

Las pruebas unitarias son también otro tipo de pruebas necesarias para cualquier software, llevadas a cabo sobre todo dentro de una metodología ágil de trabajo. Estas

nos permitirán comprobar que un fragmento de código funciona correctamente, por lo que utilizan el método de prueba Caja Blanca.

Para la realización de dichas pruebas se utilizó el marco de prueba Pytest, que no es más que la librería utilizada para definir y ejecutar los tests en Python. Es muy útil ya que facilita la creación de test unitarios, test parametrizados, y permite además inyectar en los tests objetos definidos previamente.(Dorta 2020)

En el siguiente código se refleja los resultados arrojados en el archivo de prueba test_modulo_credentials.py haciendo uso del marco de prueba Pytest:

Código fuente: Pruebas unitarias a través de Pytest.

```
import pytest
from modules.credentials.Credentials import CredentialsModel


class TestCredencialesServidor:

    def test_get_credentials(self):

        try:
            result = CredentialsModel.get_credentials()

            assert result is not None


        except Exception as e:
            pytest.fail(f"Test failed: 'id_loc' should not be present in the result, but found in {result}")


    def test_create_credentials_successful(self):
        token = '28be-11e9-b210'
        url = 'https://ocpi.example.com/versions'
        roles = [
            {'role': 'CPO',
             'party_id': 'CUB',
             'country_code': 'CU'}
        ]

        try:
            result = CredentialsModel.create_credentials(token, url, roles)
```

```
            assert result is not None
            assert 'access_token' in result
            assert 'message' in result

        except Exception as e:
            pytest.fail(f"La prueba falló con la excepción: {e}")


    def test_update_credentials_successful(self):
        new_token = '28be-11e9-b210'
        new_url = 'https://ocpi.example.com'
        new_roles = [
            {'role': 'CPO',
             'party_id': 'CUB',
             'country_code': 'CU'}
        ]

        try:
            result = CredentialsModel.update_credentials(new_token, new_url, new_roles)

            assert result is not None
            assert 'message' in result

        except Exception as e:
            pytest.fail(f"La prueba falló con la excepción: {e}")


    def test_delete_credentials_successful(self):
        token = '28be-11e9-b210'
        url = 'https://ocpi.example.com'

        try:
            result = CredentialsModel.delete_credentials(token, url)

            assert result is not None
            assert 'message' in result

        except Exception as e:
            pytest.fail(f"La prueba falló con la excepción: {e}")
```

Además, se puede observar en la siguiente figura los resultados arrojados a través del Visual Studio Code donde fueron ejecutados los tests o pruebas unitarias. La misma muestra la admisión de todos los scripts de pruebas realizados.

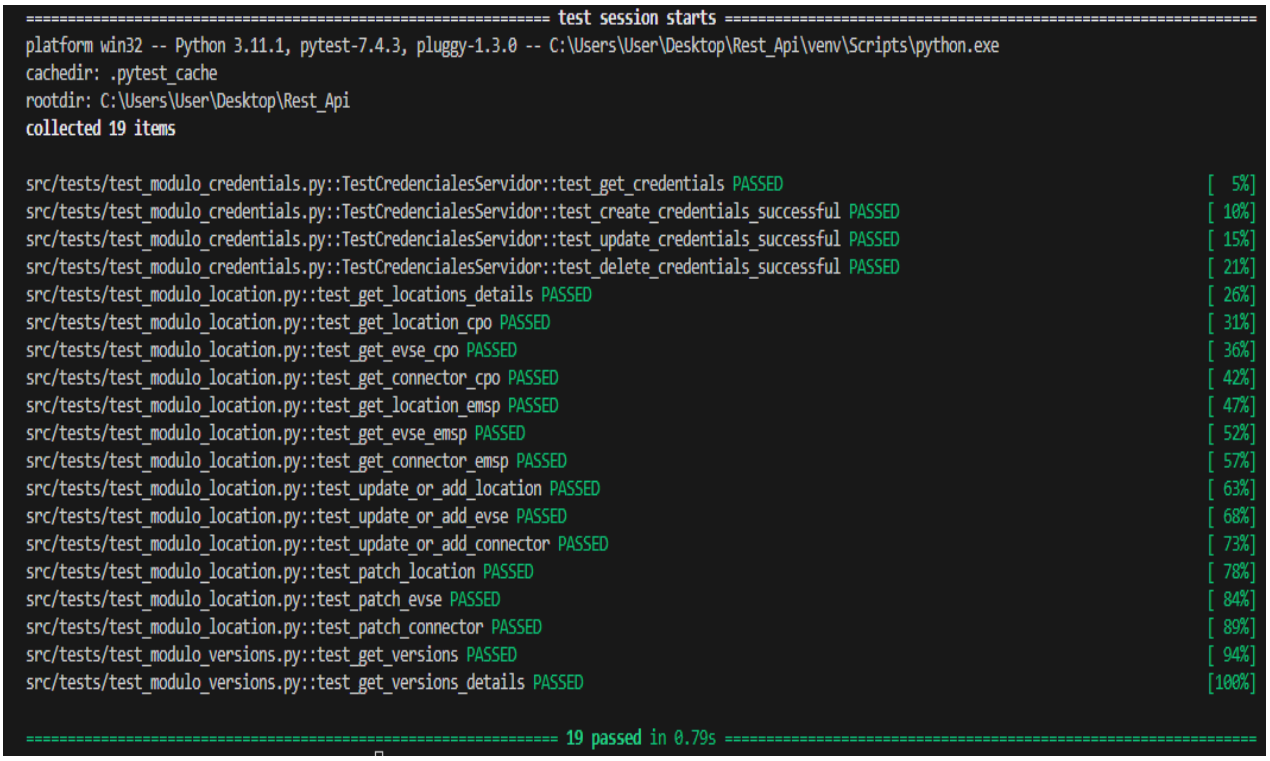


Figura #4: Admisión de las pruebas unitarias

3.4 Resultados de las pruebas:

Al realizarse las pruebas se pudieron identificar una serie de no conformidades.

Estas se pueden observar en la siguiente tabla.

No. NC	Requisito funcional	Descripción	Complejidad	Estado
1	RF 18	Errores al no ofrecer datos para modificar en la solicitud ya que no mostraba el mensaje de alerta.	Baja	Resuelta

2	RF 13	Cuando no existía la ubicación donde se deseaba insertar la estación de carga no mostraba el mensaje de ubicación no encontrada.	Baja	Resuelta
3	RF 4	No mostraba las versiones disponibles.	Alta	Resuelta

Tabla #6: No Conformidades (Elaboración Propia)

En la siguiente figura se muestran los datos correspondientes a cada iteración de prueba por las que transitó la APIRest.

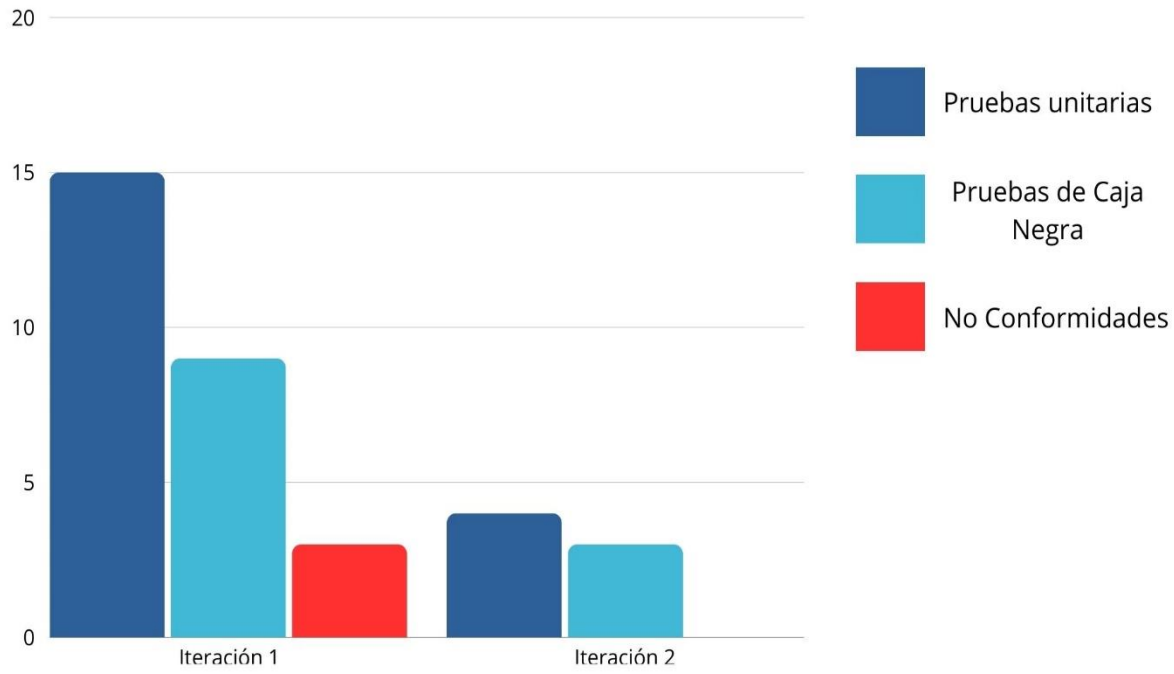


Figura #5: Iteraciones de Pruebas (Elaboración Propia)

En resumen: fueron realizadas dos iteraciones de prueba, de las cuales se obtuvieron un total de 3 no conformidades (NC) significativas: 2 con complejidad baja y otra complejidad alta. En la primera iteración fueron realizados 15 scripts o casos de pruebas unitarias y 9 casos de prueba de Caja Negra, dando como resultado 3 no conformidades. Luego se realizó una segunda iteración donde se desarrollaron 4 casos de pruebas unitarias y 3 casos de prueba de Caja Negra (para un total de 19 pruebas unitarias y 12 pruebas de Caja Negra). Después de concluida esta última iteración se resolvieron las tres NC arrojadas quedando el ciclo cerrado.

3.5 Conclusiones del capítulo:

Una vez realizada toda la fase de pruebas al módulo desarrollado para la detección de defectos, se puede arribar a que:

- ✓ Los diferentes métodos de pruebas aplicados a la solución durante el ciclo de vida del software permitieron comprobar los errores existentes y mejorar la calidad de los resultados.
- ✓ Con las pruebas unitarias se comprobó que el código que da vida a la aplicación funciona correctamente.

Conclusiones:

Las Tecnologías de la Información y la Comunicación (TIC) han demostrado ser un motor clave para el avance de la sociedad en diversas áreas, y su impacto en la movilidad eléctrica no es una excepción. La necesidad de abordar los desafíos en la interoperabilidad de la infraestructura de carga pública para vehículos eléctricos en Cuba se ha convertido en una prioridad evidente. A través de un análisis detallado de la situación actual de la infraestructura de carga de vehículos eléctricos en el país, se ha identificado no solo las limitaciones existentes sino también el potencial significativo para la expansión de la movilidad eléctrica. La implementación de la plataforma de electromovilidad basada en protocolos estandarizados por parte del Centro de Tecnologías Interactivas VERTEX representa un hito significativo en Cuba. Esta iniciativa marca el primer paso hacia la creación de una infraestructura estándar y colaborativa para la carga de vehículos eléctricos en el país. La adopción de una API RESTful basada en el protocolo OCPI, establece un estándar esencial para la comunicación entre Operadores de Puntos de Carga y Proveedores de Servicios de Movilidad Eléctrica. Esta estandarización promueve la interoperabilidad y facilita la expansión y el desarrollo futuro del ecosistema de movilidad eléctrica en Cuba. La conformidad con el protocolo OCPI en su versión 2.2 asegura la compatibilidad y alineación con los estándares internacionales, allanando el camino para una integración fluida con iniciativas globales de movilidad eléctrica. El impacto de esta plataforma no se limita únicamente a la movilidad sostenible; representa un cambio trascendental hacia un futuro más conectado y eficiente. Este avance fomenta la reducción de la dependencia de combustibles fósiles, contribuye a la mitigación de emisiones de gases de efecto invernadero y promueve la eficiencia energética en el sector del transporte en Cuba.

Recomendaciones:

Fomentar la Adopción Generalizada:

Se recomienda implementar campañas de concientización a nivel nacional para fomentar la adopción de vehículos eléctricos y la utilización de la infraestructura de carga pública. Esto incluye educar a la población sobre los beneficios ambientales y económicos de la movilidad eléctrica.

Ampliar la Infraestructura de Carga:

Dada la limitación actual de la autonomía de los vehículos eléctricos en Cuba, se sugiere la expansión de la infraestructura de carga pública a lo largo de las principales rutas y ciudades. Esto permitirá un uso más extenso de los vehículos eléctricos y eliminará las restricciones en los viajes de larga distancia.

Colaboración Interinstitucional:

Recomendamos fomentar la colaboración entre entidades gubernamentales, empresas y organizaciones involucradas en la movilidad eléctrica. La cooperación interinstitucional puede acelerar la implementación de estándares y mejorar la eficiencia en la gestión de la infraestructura de carga.

Actualización Continua del Estándar:

Dado que la tecnología evoluciona rápidamente, se aconseja establecer un mecanismo para la actualización continua del estándar de interoperabilidad. Esto garantizará que la plataforma siga siendo compatible con los avances tecnológicos y las necesidades cambiantes del ecosistema de la electromovilidad.

Incentivos y Políticas Gubernamentales:

Implementar políticas gubernamentales que ofrezcan incentivos fiscales y financieros para la adquisición de vehículos eléctricos y la instalación de puntos de carga. Estos incentivos pueden acelerar la transición hacia la movilidad eléctrica y motivar a más actores a participar en la expansión de la infraestructura.

Investigación Continua:

Se sugiere continuar la investigación en el ámbito de la movilidad eléctrica para estar al tanto de las últimas tendencias y tecnologías. Esto garantizará que Cuba siga siendo relevante y competitiva en el panorama internacional de la electromovilidad.

Desarrollo de Capacidades Locales:

Invertir en la capacitación y desarrollo de habilidades locales en el campo de la electromovilidad. El fortalecimiento de recursos humanos locales permitirá una gestión más efectiva de la infraestructura y la implementación de soluciones tecnológicas avanzadas.

Bibliografía

1. ÁLVAREZ, O. M., 2022. Los vehículos eléctricos tienen futuro en Cuba. [en línea]. marzo 2022. Recuperado a partir de: https://periodismodebarrio.org/2022/03/los-vehiculos-electricos-tienen-futuroen-cuba/

2. BLANCARTE, Oscar, 2023. Reactive Programming. [en línea]. 2023. Recuperado a partir de: https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/cliente-servidor

3. DORTA, D. A., 2020. *Framework para agilizar la aplicación de técnicas basadas en Deep Learning*.

4. ECHAZARRETA, M., 2021. Aplica mejores prácticas siguiendo los patrones GRAPS. [en línea]. octubre 2021. Recuperado a partir de: https://www.laraveltip.com/aplica-mejores-practicas-siguiendo-los-patronesgraps/

5. GOMEZ, C., 2020. Niveles de Prueba. [en línea]. agosto 2020. Recuperado a partir de: https://www.diariodeqa.com/post/niveles-de-prueba

6. KAN, Mart y BEKKERS, Rudi, 2020. Comparative Analysis of Standardized Protocols for EV Roaming. [en línea]. mayo 2020. Recuperado a partir de: https://evroaming.org/app/uploads/2020/06/D6.1-Comparative-analysis-of-standardized-protocols-for-EV-roaming.pdf

7. LAGO, N., 2022. Patrones de Diseño de Software. [en línea]. febrero 2022. Recuperado a partir de: https://saasradar.net/patrones-diseno-de-software/

8. LARREA, N. P., 2019. *Análisis de la aplicación de pruebas funcionales y pruebas de usabilidad de software en el desarrollo de sistemas web*.

9. Metodologías de desarrollo de software, 2020 [en línea]. Recuperado a partir de: https://www.becas-santander.com/es/blog/metodologias-desarrollosoftware.html

10. OCPI - Open Charge Point Interface, 2023 [en línea]. Recuperado a partir de: https://github.com/ocpi/ocpi/blob/master/mod_locations.asciidoc

11. PÁEZ, Chichí, 2022. La Influencia de las TIC en la Sociedad Actual. [en línea]. octubre 2022. Recuperado a partir de: https://entorno-empresarial.com/la-influencia-de-las-tics-en-la-sociedad-actual/

12. Patrón Composite, 2021 [en línea]. Recuperado a partir de: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/patron-composite/

13. PEP 8, 2020 [en línea]. Recuperado a partir de: https://elpythonista.com/pep-8

14. PostgreSQL, 2023 [en línea]. Recuperado a partir de: https://www.postgresql.org/docs/current/intro-whatis.html

15. Protocolo de carga de vehículos eléctricos y estandarización de protocolos, 2023 [en línea]. Recuperado a partir de: https://energy5.com/es/protocolo-de-carga-de-vehiculos-electricos-y-estandarizacion-de-protocolos

16. SÁNCHEZ, T. R., 2015. *Metodología de desarrollo para la Actividad productiva de la UCI*. La Habana.

17. SHIVETS, Alexander, 2021. Design Patterns. [en línea]. enero 2021. Recuperado a partir de: https://refactoring.guru/es/design-patterns

18. SOMMERVILLE, I., 2011. *Ingeniería de Software (Novena ed.)*.

19. TURRADO, J., 2020. ¿Qué son las pruebas de software? [en línea]. marzo 2020. Recuperado a partir de:

https://www.campusmvp.es/recursos/post/que-son-las-pruebas-desoftware.aspx

20. UML: Lenguaje Unificado de Modelado Orientado a Objetos, 2018 [en línea]. Recuperado a partir de: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/uml-lenguajeunificado-de-modelado-orientado-a-objetos/

21. *Visual Paradigm*, 2018 [en línea]. Recuperado a partir de: https://www.capterra.es/software/145716/visualparadigm

22. Visual Studio Code, 2023 [en línea]. Recuperado a partir de: https://code.visualstudio.com/docs

Anexo 1:

Historia de usuario	
Número: 1	**Nombre:** Obtener información sobre las versiones disponibles
Usuario: EMSP y CPO	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	
Descripción: La estructura del punto final no está definida en OCPI. Esto está abierto para que cada parte se defina a sí misma quedando definido el punto final {servidor}/ocpi/versions. Este punto final enumera todas las versiones OCPI disponibles y las URL correspondientes a donde se pueden encontrar detalles específicos de la versión, como los puntos finales compatibles.	
Observaciones: Documentación oficial de OCPI	

Tabla #7: Historia de Usuario #1 (Elaboración Propia)

Historia de usuario	
Número: 2	**Nombre:** Obtener información sobre los puntos finales para las versiones
Usuario: EMSP y CPO	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	
Descripción: La estructura del punto final no está definida en OCPI. Esto está abierto para que cada parte se defina a sí misma quedando definido el punto final {servidor}/ocpi/versions/details. Este punto final enumera los puntos finales compatibles y sus URL para una versión específica de OCPI.	
Observaciones: Documentación oficial de OCPI	

Tabla #8: Historia de Usuario #2 (Elaboración Propia)

Historia de usuario	
Número: 3	**Nombre:** Obtener las ubicaciones
Usuario: EMSP	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	
Descripción: El EMSP mediante un punto final estructurado de la siguiente forma {locations_endpoint_url}/[date_from={date_from}]&[date_to={date_to}]&[offset={offset}]& [limit={limit}] puede un listado en forma de JSON de las ubicaciones, solo ubicaciones con (last_updated) entre lo dado {date_from} (incluyendo) y {date_to} (excluyendo) será devuelto. Si se actualiza una estación de carga, también las ubicaciones 'padres last_updated el campo debe actualizarse. Del mismo modo, si se actualiza un conector, las estaciones de carga last_updated y las ubicaciones last_updated los campos deben actualizarse.	
Observaciones: Documentación oficial de OCPI	

Tabla #9: Historia de Usuario #3 (Elaboración Propia)

Historia de usuario	
Número: 5	**Nombre:** Obtener una estación de carga especifica
Usuario: EMSP	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	
Descripción: El EMSP mediante un punto final estructurado de la siguiente forma {locations_endpoint_url}/{location_id}/{uid} puede obtener la información en forma de JSON de la estación de carga correspondiente a partir del location_id y el uid proporcionado en el punto final. En caso de que la estación de carga no exista se mostrara un mensaje “EVSE	

no encontrado” y en caso de que la ubicación no exista muestra el mensaje "Ubicación no encontrada"
Observaciones: Documentación oficial de OCPI

Tabla #10: Historia de Usuario #5 (Elaboración Propia)

Historia de usuario	
Número: 6	**Nombre:** Obtener un conector especifico
Usuario: EMSP	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	
Descripción: El EMSP mediante un punto final estructurado de la siguiente forma {locations_endpoint_url}/{location_id}/{uid}/{id_connector} puede obtener la información en forma de JSON del conector correspondiente a partir del location_id, el uid y el id_connector proporcionado en el punto final. En caso de que el conector no exista se mostrara un mensaje “Conector no encontrado”. En caso de que la estación de carga no exista se mostrara un mensaje “EVSE no encontrado”. En caso de que la ubicación no exista se mostrara un mensaje “Ubicación no encontrada”.	
Observaciones: Documentación oficial de OCPI	

Tabla #11: Historia de Usuario #6 (Elaboración Propia)

Historia de usuario	
Número: 7	**Nombre:** Obtener una ubicación existente
Usuario: CPO	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	

Descripción: El CPO mediante un punto final estructurado de la siguiente forma {locations_endpoint_url}/{country_code}/{party_id}/{location_id} puede obtener la información en forma de JSON de la ubicación correspondiente a partir del country_code el party_id y el location_id proporcionado en el punto final. En caso de que la ubicación no exista se mostrara un mensaje “Ubicación no encontrada”.
Observaciones: Documentación oficial de OCPI

Tabla #12: Historia de Usuario #7 (Elaboración Propia)

Historia de usuario	
Número: 9	**Nombre:** Adicionar una nueva estación de carga y Actualizar una estación de carga.
Usuario: CPO	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	
Descripción: El CPO mediante un punto final estructurado de la siguiente forma {locations_endpoint_url}/{country_code}/{party_id}/{location_id}/{uid} puede actualizar la estación de carga en forma de JSON, en el caso del uid toma la información del punto final y el cuerpo del JSON debe incluir: evse_id, status, capabilities, physical_reference, floor_level, de la estación de carga correspondiente a partir del uid proporcionado en el punto final y muestra el mensaje “EVSE {uid} actualizado correctamente.”. En caso de que la estación de carga no exista debe insertarla. No se incluye el campo last_updated ya que esta toma la fecha actual. En caso de que la ubicación no exista muestra el mensaje "La ubicación no existe con los datos proporcionados".	
Observaciones: Documentación oficial de OCPI	

Tabla #13: Historia de Usuario #9 (Elaboración Propia)

Historia de usuario	
Número: 10	**Nombre:** Adicionar un nuevo conector y Actualizar un conector.

Usuario: CPO	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	
Descripción: El CPO mediante un punto final estructurado de la siguiente forma {locations_endpoint_url}/{country_code}/{party_id}/{location_id}/{uid}/{connector_id} puede actualizar el conector en forma de JSON, en el caso del id toma la información del punto final y el cuerpo del JSON debe incluir: standard, format, power_type, max_voltage, max_amperage, del conector correspondiente a partir del {connector_id} proporcionado en el punto final y muestra el mensaje “Conector {connector_id} actualizado correctamente.”. En caso de que el conector no exista debe insertarlo. No se incluye el campo last_updated ya que esta toma la fecha actual. En caso de que la estación de carga no exista muestra el mensaje "El EVSE no existe con los datos proporcionados". En caso de que la ubicación no exista muestra el mensaje "La ubicación no existe con los datos proporcionados".	
Observaciones: Documentación oficial de OCPI	

Tabla #14: Historia de Usuario #10 (Elaboración Propia)

Historia de usuario	
Número: 11	**Nombre:** Actualizar parcialmente una ubicación
Usuario: CPO	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	
Descripción: El CPO mediante un punto final estructurado de la siguiente forma {locations_endpoint_url}/{country_code}/{party_id}/{location_id} puede actualizar la información en forma de JSON, en el caso del country_code y el party_id toma la información del punto final y el cuerpo del JSON puede incluir: publish, name, address, city, postal_code, country, latitude, longitude, parking_type, operator_name y time_zone de la ubicación	

correspondiente a partir del id proporcionado en el punto final y muestra el mensaje “Ubicación {location_id} actualizada correctamente.”. En caso de que la ubicación no exista muestra el mensaje “La ubicación no existe con los datos proporcionados”. No se incluye el campo last_updated ya que esta toma la fecha actual. En caso de no proporcionar datos muestra el mensaje “No se proporcionaron campos para actualizar”
Observaciones: Documentación oficial de OCPI

Tabla #15: Historia de Usuario #11 (Elaboración Propia)

Historia de usuario	
Número: 12	**Nombre:** Actualizar parcialmente una estación de carga
Usuario: CPO	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	
Descripción: El CPO mediante un punto final estructurado de la siguiente forma {locations_endpoint_url}/{country_code}/{party_id}/{location_id}/{uid} puede actualizar la estación de carga en forma de JSON, en el caso del uid toma la información del punto final y el cuerpo del JSON puede incluir: evse_id, status, capabilities, physical_reference, floor_level, de la estación de carga correspondiente a partir del uid proporcionado en el punto final y muestra el mensaje “EVSE {uid} actualizado correctamente.”. En caso de que la estación de carga o la ubicación no exista muestra el mensaje “Ubicacion o EVSE no enctontrado con los datos proporcionados”. No se incluye el campo last_updated ya que esta toma la fecha actual. En caso de no proporcionar datos muestra el mensaje “No se proporcionaron campos para actualizar”	
Observaciones: Documentación oficial de OCPI	

Tabla #16: Historia de Usuario #12 (Elaboración Propia)

Historia de usuario	
Número: 13	**Nombre:** Actualizar parcialmente un conector

Usuario: CPO	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Tiempo estimado: 1 semana	**Iteración asignada:** 1
Programador responsable: Oscar Coello Perdomo	
Descripción: El CPO mediante un punto final estructurado de la siguiente forma {locations_endpoint_url}/{country_code}/{party_id}/{location_id}/{uid}/{connector_id} puede actualizar el conector en forma de JSON, en el caso del id toma la información del punto final y el cuerpo del JSON puede incluir: standard, format, power_type, max_voltage, max_amperage, del conector correspondiente a partir del {connector_id} proporcionado en el punto final y muestra el mensaje “Conector {connector_id} actualizado correctamente.”. En caso de que el conector no exista muestra el mensaje “Conector {connector_id} no existe”. No se incluye el campo last_updated ya que esta toma la fecha actual. En caso de que la estación de carga o la ubicación no exista muestra el mensaje " Ubicacion o EVSE no enctontrado con los datos proporcionados ". En caso de no proporcionar datos muestra el mensaje “No se proporcionaron campos para actualizar”.	
Observaciones: Documentación oficial de OCPI	

Tabla #17: Historia de Usuario #13 (Elaboración Propia)

Anexo 2:

Sección: Crear Credenciales								
Id del escenario	**Escenario**	**Token**	**Url**	**Role**	**Party_id**	**Country_code**	**Respuesta del sistema**	**Resultado de la prueba**
1.1	Creación exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra el token de acceso además del mensaje "Credenciales creadas con éxito" y los puntos finales accesibles.	Se registro satisfactoriamente la credencial.
		V	V	V	V	V		
		ebf3b399-779f-4497-9b9d-ac6ad3cc44d2	https://example.com/ocpi/versions	CPO	EXA	NL		
1.2	Campos vacíos	N/A	N/A	V	N/A	N/A	Muestra un error 404 con el mensaje “Token, URL y roles son campos obligatorios y no pueden estar vacíos”	Se alertó satisfactoriamente la presencia de campos vacíos.
				CPO				
1.3	Datos incorrectos	V		I	I		Muestra un error 404 y muestra un mensaje indicando los errores existentes en los campos	Se alertó satisfactoriamente la presencia de campos incorrectos.
		ebf3b399-779f-4497-9b9d-ac6ad3cc44d2		CPU	ESPO			

Tabla #18: Caso de Prueba Crear Credenciales (Elaboración Propia)

Sección: Modificar Credenciales								
Id del escenario	**Escenario**	**Token**	**Url**	**Role**	**Party_id**	**Country_code**	**Respuesta del sistema**	**Resultado de la prueba**
2.1	Modificación exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra el mensaje "Credenciales actualizadas con éxito"	Se modificó satisfactoriamente la credencial.
		V	V	V	V	V		
		ebf3b399-779f-4497-9b9d-ac6ad3cc44d2	https://example.com/ocpi/versions	CPO	CUB	CU		
2.2	Campos vacíos	N/A	N/A	V		N/A	Muestra un error 404 con el mensaje “Token, URL y roles son campos obligatorios y no pueden estar vacíos”	Se alertó satisfactoriamente la presencia de campos vacíos.
				CPO				
2.3	Datos incorrectos	V		I	I		Muestra un error 404 y muestra un mensaje indicando los errores existentes en los campos	Se alertó satisfactoriamente la presencia de campos incorrectos.
		ebf3b399-779f-4497-9b9d-ac6ad3cc44d2		CPU	ESPO			

Tabla #19: Caso de Prueba Modificar Credenciales (Elaboración Propia)

Sección: Eliminar Credenciales

Id del escenario	Escenario	Token	Url	Respuesta del sistema	Resultado de la prueba
3.1	Eliminación exitosa	[V,I,N/A]	[V,I,N/A]	Muestra el mensaje "Credenciales eliminadas con éxito"	Se eliminó satisfactoriamente la credencial.
		V	V		
		ebf3b399-779f-4497-9b9d-ac6ad3cc44d2	https://example.com/ocpi/versions		
3.2	Campos vacíos	N/A	N/A	Muestra un error 404 con el mensaje “Token y URL son campos obligatorios y no pueden estar vacíos”	Se alertó satisfactoriamente la presencia de campos vacíos.
3.3	Datos incorrectos	V		Muestra un mensaje "Credenciales no encontradas"	Se alertó satisfactoriamente la presencia de campos incorrectos.
		ebf3b399-779f-4497-9b9d-ac6ad3cc44d2			

Tabla #20: Caso de Prueba Eliminar Credenciales (Elaboración Propia)

Sección: Obtener Ubicación CPO						
Id del escenario	**Escenario**	**Country_id**	**Party_id**	**Location_id**	**Respuesta del sistema**	**Resultado de la prueba**
4.1	Búsqueda exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra la ubicación así como las estaciones de carga (EVSE) y conectores asociados a esta	Se encontró satisfactoriamente la ubicación.
		V	V	V		
		CU	CUB	LOC2		
4.2	Campos vacíos	N/A	N/A	N/A	Muestra un error 404 Not Found	Se alertó satisfactoriamente la presencia de campos vacíos.
4.3	Datos incorrectos	V			Muestra un mensaje "Ubicación no encontrada"	Se alertó satisfactoriamente la presencia de campos incorrectos.
		CU	CUB	LOC5		

Tabla #21: Caso de Prueba Obtener Ubicación CPO (Elaboración Propia)

Sección: Obtener Estación de carga CPO							
Id del escenario	**Escenario**	**Country_id**	**Party_id**	**Location_id**	**uid**	**Respuesta del sistema**	**Resultado de la prueba**
5.1	Búsqueda exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra la estación de carga (EVSE) y conectores asociados a esta	Se encontró satisfactoriamente la estación de carga.
		V	V	V	V		
		CU	CUB	LOC2	1234		
5.2	Campos vacíos	N/A	N/A	N/A	N/A	Muestra un error 404 Not Found	Se alertó satisfactoriamente la presencia de campos vacíos.
5.3	Datos incorrectos	V				Muestra un mensaje "Ubicación no encontrada" en caso de que no exista la ubicación y "EVSE no encontrado" en caso de que no se encuentre la estación de carga	Se alertó satisfactoriamente la presencia de campos incorrectos.
		CU	CUB	LOC5	12345		

Tabla #22: Caso de Prueba Obtener Estación de carga CPO (Elaboración Propia)

Sección: Obtener Conector CPO								
Id del escenario	**Escenario**	**Country_id**	**Party_id**	**Location_id**	**uid**	**Connector_id**	**Respuesta del sistema**	**Resultado de la prueba**
6.1	Búsqueda exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra el conector	Se encontró satisfactoriamente el conector.
		V	V	V	V	V		
		CU	CUB	LOC2	1234	1		
6.2	Campos vacíos	N/A	N/A	N/A	N/A	N/A	Muestra un error 404 Not Found	Se alertó satisfactoriamente la presencia de campos vacíos.
6.3	Datos incorrectos	V					Muestra un mensaje "Ubicación no encontrada" en caso de que no exista la ubicación y "EVSE no encontrado" en caso de que no se encuentre la estación de carga y "Conector no encontrado" en caso de no encontrarse el conector	Se alertó satisfactoriamente la presencia de campos incorrectos.
		CU	CUB	LOC5	12345	3		

Tabla #23: Caso de Prueba Obtener Conector CPO (Elaboración Propia)

Sección: Modificar o Insertar Ubicación CPO										
Id del escenario	**Escenario**	**Country_id**	**Party_id**	**Location_id**	**publish**	**name**	**address**	**city**	**Respuesta del sistema**	**Resultado de la prueba**
7.1	Modificación exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A	[V,I,N/A]	Muestra el mensaje "Ubicación LOC2 actualizada correctamente."	Se modificó satisfactoriamente la ubicación.
		V	V	V	V	V	V	V		
		CU	CUB	LOC2	True	Gent	F.Roosevelt laan 3A	Matanzas		
7.2	Campos vacíos	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Muestra un mensaje indicando cual es el campo faltante	Se alertó satisfactoriamente la presencia de campos vacíos.
7.3	Datos incorrectos	V	V	V	I			I	Muestra un mensaje indicando cual es el campo incorrecto	Se alertó satisfactoriamente la presencia de campos incorrectos.
		CU	CUB	LOC2	1			3		
7.4	Dato correctos pero no existentes	V	V	V	V	V	V	V	Inserta la nueva ubicación y muestra el mensaje "Ubicación LOC4 insertada con éxito"	Se insertó satisfactoriamente la ubicación.
		CU	CUB	LOC4	True	Fred	Calle 8 /Neptuno y Vicente # 4532	Habana		

Tabla #24: Caso de Prueba Modificar o Insertar Ubicación CPO (Elaboración Propia)

Sección: Modificar o Insertar Estación de carga CPO										
Id del escenario	**Escenario**	**Country_id**	**Party_id**	**Location_id**	**uid**	**evse_id**	**status**	**capabilities**	**Respuesta del sistema**	**Resultado de la prueba**
8.1	Modificación exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A	[V,I,N/A]	Muestra el mensaje "EVSE 1234 actualizado correctamente"	Se modificó satisfactoriamente la estación de carga.
		V	V	V	V	V	V	V		
		CU	CUB	LOC2	1234	EVSE789	AVALIABLE	RESERVABLE		
8.2	Campos vacíos	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Muestra un mensaje indicando cual es el campo faltante	Se alertó satisfactoriamente la presencia de campos vacíos.
8.3	Datos incorrectos	V	V	V	V	V	I	I	Muestra un mensaje indicando cual es el campo incorrecto	Se alertó satisfactoriamente la presencia de campos incorrectos.
		CU	CUB	LOC2	1234	EVSE789	455	3		
8.4	Datos correctos pero no existentes	V	V	V	V	V	V	V	Inserta la nueva estación de carga y muestra el mensaje "EVSE 12345 insertado con éxito"	Se insertó satisfactoriamente la estación de carga.
		CU	CUB	LOC2	12345	EVSE789	AVALIABLE	RESERVABLE		

Tabla #25: Caso de Prueba Modificar o Insertar Estación de carga CPO (Elaboración Propia)

Sección: Modificar o Insertar Connector CPO										
Id del escenario	**Escenario**	**Country_id**	**Party_id**	**Location_id**	**uid**	**connector_id**	**standard**	**format**	**Respuesta del sistema**	**Resultado de la prueba**
9.1	Modificación exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A	[V,I,N/A]	Muestra el mensaje "Conector 1 actualizado correctamente "	Se modificó satisfactoriamente el conector.
		V	V	V	V	V	V	V		
		CU	CUB	LOC2	1234	1	IEC_62196_T2	Cable		
9.2	Campos vacíos	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Muestra un mensaje indicando cual es el campo faltante	Se alertó satisfactoriamente la presencia de campos vacíos.
9.3	Datos incorrectos	V	V	V	V	V	I	I	Muestra un mensaje indicando cual es el campo incorrecto	Se alertó satisfactoriamente la presencia de campos incorrectos.
		CU	CUB	LOC2	1234	1	jbj	3		
9.4	Datos correctos pero no existentes	V	V	V	V	V	V	V	Inserta el nuevo conector y muestra el mensaje "Conector 3 insertado con éxito"	Se insertó satisfactoriamente la ubicación.
		CU	CUB	LOC2	12345	3	IEC_62196_T2	Soket		

Tabla #26: Caso de Prueba Modificar o Insertar Conector CPO (Elaboración Propia)

Sección: Modificación Parcial de Ubicación CPO								
Id del escenario	**Escenario**	**Country_id**	**Party_id**	**Location_id**	**publish**	**name**	**Respuesta del sistema**	**Resultado de la prueba**
10.1	Modificación exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra el mensaje "Ubicación LOC2 actualizada correctamente."	Se modificó satisfactoriamente la ubicación.
		V	V	V	V	V		
		CU	CUB	LOC2	falce	Gent Z		
10.2	Campos vacíos	N/A	N/A	N/A	N/A	N/A	Muestra un mensaje "No se proporcionaron campos para actualizar."	Se alertó satisfactoriamente la presencia de campos vacíos.
10.3	Datos incorrectos	V	V	V	I		Muestra un mensaje indicando cual es el campo incorrecto	Se alertó satisfactoriamente la presencia de campos incorrectos.
		CU	CUB	LOC2	1			
10.4	Ubicación no existente	V	V	V	V	V	Muestra el mensaje "La ubicación no existe con los datos proporcionados"	Se modificó satisfactoriamente cuando no se encuentra la ubicación.
		CU	CUB	LOC4	True	Fred		

Tabla #27: Caso de Prueba Modificar Parcialmente Ubicación CPO (Elaboración Propia)

Sección: Modificación Parcial de estación de carga CPO								
Id del escenario	**Escenario**	**Country_id**	**Party_id**	**Location_id**	**uid**	**status**	**Respuesta del sistema**	**Resultado de la prueba**
11.1	Modificación exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra el mensaje "EVSE 1234 actualizad0 correctamente."	Se modificó satisfactoriamente la estación de carga.
		V	V	V	V	V		
		CU	CUB	LOC2	1234	REMOVED		
11.2	Campos vacíos	N/A	N/A	N/A	N/A	N/A	Muestra un mensaje indicando cual es el campo faltante	Se alertó satisfactoriamente la presencia de campos vacíos.
11.3	Datos incorrectos	V	V	V	V	I	Muestra un mensaje indicando cual es el campo incorrecto	Se alertó satisfactoriamente la presencia de campos incorrectos.
		CU	CUB	LOC2	1234	34		
11.4	Dato correctos pero no existentes	V	V	V	I	V	Muestra el mensaje "Ubicación o EVSE no encontrado con los datos proporcionados"	Se modificó satisfactoriamente cuando no se encuentra la estación de carga.
		CU	CUB	LOC2	12	REMOVED		

Tabla #28: Caso de Prueba Modificar Parcialmente Ubicación CPO (Elaboración Propia)

Sección: Modificación Parcial de conector CPO									
Id del escenario	**Escenario**	**Country_id**	**Party_id**	**Location_id**	**uid**	**connector_id**	**format**	**Respuesta del sistema**	**Resultado de la prueba**
12.1	Modificación exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra el mensaje "Connector 1 actualizado correctamente."	Se modificó satisfactoriamente el conector.
		V	V	V	V	V	V		
		CU	CUB	LOC2	1234	1	Cable		
12.2	Campos vacíos	N/A	N/A	N/A	N/A	N/A	N/A	Muestra un mensaje indicando cual es el campo faltante	Se alertó satisfactoriamente la presencia de campos vacíos.
12.3	Datos incorrectos	V	V	V	V	V	I	Muestra un mensaje indicando cual es el campo incorrecto	Se alertó satisfactoriamente la presencia de campos incorrectos.
		CU	CUB	LOC2	1234	1	34		
12.4	Dato correctos pero no existentes	V	V	V	V	V	V	Muestra el mensaje "Ubicación, EVSE o conector no encontrado con los datos proporcionados"	Se insertó satisfactoriamente la ubicación.
		CU	CUB	LOC2	1234	5	Cable		

Tabla #29: Caso de Prueba Modificar Parcialmente Ubicación CPO (Elaboración Propia)