



**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 4**

**Módulo de Control de Acceso de Usuarios a la red
de Cargadores Eléctricos**

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

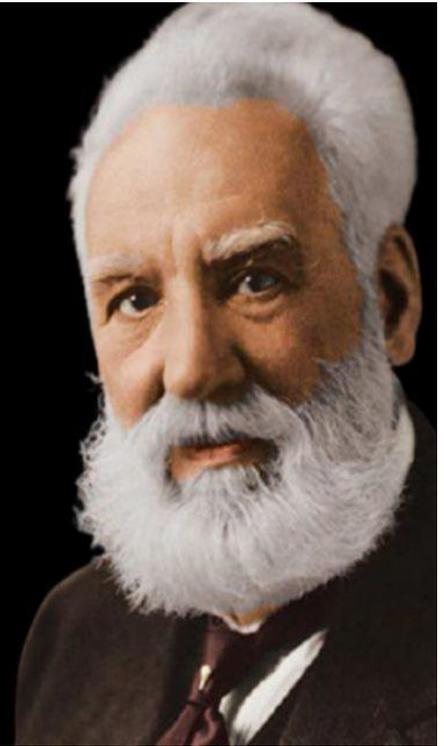
Autora: María Magdalena Velázquez Viego

Tutor: Ing. Andy Suárez Oña

**La Habana, 12 de noviembre de 2022
“Año 64 de la Revolución”**

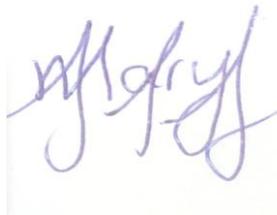
Antes que cualquier otra
cosa, la preparación es la
llave del éxito.

Alexander Graham Bell



Declaración de autoría:

Declaro ser autora de la presente tesis que tiene por título: *Módulo de Control de Acceso de Usuarios a la red de Cargadores Eléctricos*, 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 16 días del mes de noviembre del año 2022.



María Magdalena Velázquez Viego

Autora



Ing. Andy Suárez Oña

Tutor

Dedicatoria:

Dedico este trabajo de tesis primeramente a Dios, por darme la perseverancia suficiente para poder hacer frente a las dificultades presentes en el día a día y darme la oportunidad de llegar hasta aquí; a mis padres, por el constante apoyo que me brindan y los sabios consejos que siempre me imparten, y por enseñarme a luchar por las cosas que queremos; a mi abuela por siempre apoyarme y darme ese cariño que me inspira cada día; a mis hermanos y amigos que de una u otra manera me brindaron su apoyo para poder alcanzar mis objetivos profesionales; y a todos aquellos que hicieron posible este sueño, siempre estarán presentes.

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. Con el propósito de lograr establecer a lo largo y ancho del país, incluido en autopistas, un número considerable de puntos de carga supervisados y controlados, la Universidad de las Ciencias Informáticas se ha encargado de desarrollar un sistema para monitorizar y controlar toda la red de cargadores eléctricos en el país. Este sistema actualmente necesita además poseer un mecanismo de control de acceso de los usuarios a los servicios que el mismo debe brindar. Por lo anteriormente mencionado se decidió investigar sobre el tema y desarrollar un sistema que permita controlar el acceso de los usuarios a la red de cargadores eléctricos en el país.

Palabras clave: cargadores eléctricos, control de acceso, red, servicios, usuarios.

Abstract:

The fight against climate change that affects the world today and the dire economic situation that fossil fuels are going through, have proven to be indicators that seek to promote the development of a sustainable energy market at a global level and, in particular, of electric cars . The electrification of the automotive transport sector is an efficient and intelligent alternative to achieve quality in the productive sector and public transport, in addition to contributing to the energy independence of any country. Currently, the insertion of electric vehicles in Cuba has been one of the objectives set to make the electrification of this sector a fact. With the purpose of establishing a considerable number of supervised and controlled charging points throughout the country, including highways, the University of Informatics Sciences has been in charge of developing a system to monitor and control the entire network of electric chargers in the country. This system currently also needs to have a user access control mechanism to the services that it must provide. Due to the aforementioned, it was decided to investigate the subject and develop a system that allows users to control access to the network of electric chargers in the country.

Keywords: Access control, electric chargers, network, services, users.

Índice

Introducción:.....	1
Capítulo 1. Fundamentación Teórica	6
1.1 Conceptos asociados:	6
1.1.1 Sistemas de control de acceso:	6
1.1.2 Tipos de control de acceso:	7
1.1.3 Mecanismos de control de acceso:.....	8
1.2 Sistemas homólogos a nivel internacional:	10
1.2.1 Plataforma Cosmos:	10
1.2.2 Plataforma EVcharge:.....	12
1.3 Metodología de desarrollo de software:	15
1.3.1 Metodología AUP-UCI:	16
1.4 Lenguajes:	16
1.4.1 Lenguaje de modelado: UML:.....	16
1.4.2 Lenguaje de programación: Python:	17
1.5 Herramientas a utilizar:	18
1.5.1 Herramienta de modelado: Visual Paradigm:	18
1.5.2 Framework: Django:.....	19
1.5.3 Entorno de desarrollo integrado (IDE): PyCharm:.....	20
1.5.4 Gestor de base de datos: PgAdmin:	21
1.4 Conclusiones del capítulo:	21
Capítulo 2. Análisis y Diseño:.....	23
2.1 Especificación de requisitos:	23
2.1.1 Requisitos funcionales:.....	23
2.1.2 Requisitos no funcionales:	23
2.2 Propuesta de solución:.....	24
2.3 Historias de usuario:	25
2.3.1 Descripción de las historias de usuarios:.....	26
2.4 Descripción de la arquitectura:.....	27
2.4.1 Arquitectura Modelo Vista Plantilla (MVT):	27
2.5 Patrones de diseño:	29
2.5.1 Patrones GRASP:.....	30

2.5.2 Patrones GOF:.....	32
2.6 Diagrama de clases:	32
2.7 Conclusiones del capítulo:	36
Capítulo 3. Implementación y evaluación de la solución propuesta:	37
3.1 Estándar de codificación:	37
3.2 Pruebas:.....	39
3.2.1 Niveles de prueba:.....	39
3.2.2 Métodos de prueba:	41
3.3 Pruebas funcionales:.....	41
3.3.1 Prueba de Caja Negra mediante la técnica de partición de equivalencia: .	42
3.3.2 Pruebas unitarias:	43
3.4 Resultados de las pruebas:.....	46
3.5 Conclusiones del capítulo:	48
Conclusiones:.....	50
Recomendaciones:.....	51
Bibliografía Consultada:	52
Anexo 1:	55
Anexo 2:	61

Índice de figuras

Figura 1: Componentes de la infraestructura de carga de vehículos eléctricos (Elaboración propia)	3
Figura 2: Propuesta de solución (Elaboración propia).....	25
Figura 3: Modelo Vista Template (Elaboración propia)	29
Figura 4: Diagrama de clases.....	34
Figura 5: Admisión de las pruebas unitarias	46
Figura 6: Iteraciones de pruebas (Elaboración propia).....	48

Índice de tablas

Tabla 1: Sistemas homólogos (Elaboración propia)	13
Tabla 2: Historia de usuario # 1(Elaboración propia)	26
Tabla 3: Historia de usuario # 6 (Elaboración propia)	26
Tabla 4: Caso de prueba: Autenticar usuario (Elaboración propia)	42
Tabla 5: No conformidades (Elaboración propia)	46
Tabla 6: Historia de usuario # 2 (Elaboración propia)	55
Tabla 7: Historia de usuario # 3 (Elaboración propia)	55
Tabla 8: Historia de usuario # 4 (Elaboración propia)	56
Tabla 9: Historia de usuario # 5 (Elaboración propia)	56
Tabla 10: Historia de usuario # 7 (Elaboración propia)	57
Tabla 11: Historia de usuario # 8 (Elaboración propia)	57
Tabla 12: Historia de usuario # 9 (Elaboración propia)	58
Tabla 13: Historia de usuario # 10 (Elaboración propia)	58
Tabla 14: Historia de usuario # 11 (Elaboración propia)	59
Tabla 15: Historia de usuario # 12 (Elaboración propia)	59
Tabla 16: Caso de prueba: Registrar usuario (Elaboración propia):.....	61
Tabla 17: Caso de prueba: Modificar usuario (Elaboración propia).....	63
Tabla 18: Caso de prueba: Eliminar usuario (Elaboración propia)	64
Tabla 19: Caso de prueba: Buscar usuario (Elaboración propia).....	65
Tabla 20: Caso de prueba: Cerrar sesión (Elaboración propia)	65
Tabla 21: Caso de prueba: Consultar ubicación de puntos de carga (Elaboración propia)	66
Tabla 22: Caso de prueba: Mostrar distancia (Elaboración propia).....	66
Tabla 23: Caso de prueba: Mostrar estado de los cargadores (Elaboración propia).67	
Tabla 24: Caso de prueba: Mostrar informe de actividades (Elaboración propia).....	68
Tabla 25: Caso de prueba: Editar perfil de usuario (Elaboración propia)	69

Introducción:

En Cuba, el sector del transporte automotor consume anualmente 992 mil toneladas de combustible: 74 % diésel y el 26 % gasolina. La electrificación de este sector es una alternativa estratégica que contribuye con la seguridad e independencia energética del país, al reducir su dependencia de los derivados del petróleo. Al mismo tiempo, aumenta la eficiencia energética, la disponibilidad técnica de los medios de transporte e impacta positivamente en la calidad del transporte público de pasajeros y del sector productivo. Además, permite una disminución considerable de las emisiones de gases de efecto invernadero en las ciudades, un mejor uso de las fuentes renovables de energía y una operación más eficiente del sistema electroenergético nacional. (Álvarez, 2022)

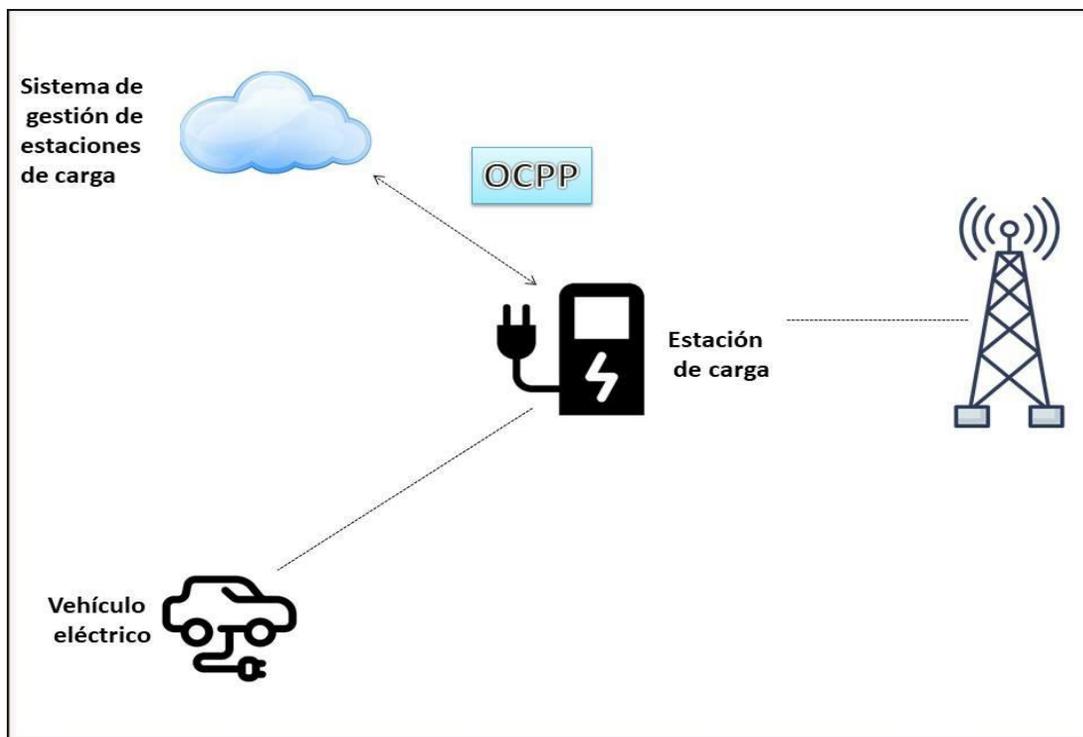
La adquisición de vehículos eléctricos en la Empresa Eléctrica de La Habana, Empresa de Telecomunicaciones de Cuba S.A (ETECSA por sus siglas en español), Ómnibus Metropolitanos y Aguas de La Habana, con excelentes indicadores de ahorro de combustible, refiere que la inserción gradual de vehículos eléctricos 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.

Además, el mecanismo de carga utiliza un comportamiento por defecto plug and play¹ local definido por el fabricante, que no verifica si el cliente está autorizado a utilizar el cargador, ni limita la cantidad de energía a servir. Estas deficiencias atentan contra el propósito de establecer a lo largo y ancho del país, incluido autopistas, un número considerable de puntos de carga supervisados y controlados remotamente bajo el concepto de electrolineras.

¹ Esta expresión quiere decir “conectar y usar” (sin la necesidad de configuración).

Al haberse identificado la ausencia en Cuba de un actor reconocido dentro de los ecosistemas de carga público a nivel internacional, y consecuentemente la imposición del desarrollo de un ecosistema de carga público capaz de dar respuesta a una demanda que será creciente con los años, se propuso el desarrollo de un sistema de gestión de estaciones de carga (CSMS por sus siglas en inglés), capaz de proveer servicios de electromovilidad para el ecosistema de carga público en Cuba. Este proyecto fue asumido por la Universidad de las Ciencias Informáticas (UCI por sus siglas en español), la cual ya cuenta con una versión funcional de un sistema que permite monitorizar y controlar toda la red de cargadores eléctricos existentes conectados a él, empleando el Protocolo Abierto de Punto de Carga (OCPP por sus siglas en inglés).

Este sistema central es un software que recibe y controla la información relativa a las sesiones de carga, las reservas y las actualizaciones. En la figura 1 (véase la figura que sigue a continuación de este párrafo) se puede observar cómo es la comunicación de este sistema con las estaciones de carga a través del protocolo. Un componente crítico en la figura es el Sistema de Gestión de Estaciones de Carga o CSMS por sus siglas en inglés (que siendo más específico sería el sistema central desarrollado por la UCI). El CSMS permite a los operadores de los puntos de recarga gestionar y supervisar las operaciones en las instalación de recarga como la fijación de tarifas de carga, la generación de informes y la gestión remota de los cargadores. Este sistema se comunica con la Estación de Carga o EVSE a través del protocolo OCPP, el cual funciona con todos los protocolos de carga para vehículos eléctricos: CHAdeMO, CCS (Sistema de Carga Combinado) o GB/T.



*Figura 1: Componentes de la infraestructura de carga de vehículos eléctricos
(Elaboración propia)*

Este sistema central autoriza a los usuarios basándose en un código que le provee el protocolo, denominado idTag, el cual puede introducirse en el poste de recarga de varias maneras, ya sea con un código de barras, con un dispositivo con Comunicación de Campo Cercano (NFC por sus siglas en inglés), con una tarjeta, etc. Sin embargo, esta forma de autorización no controla adecuadamente el acceso de los usuarios a dicho sistema, careciendo así de un mecanismo de autenticación para el acceso de usuarios y del uso de un protocolo de cifrado y autenticación para las conexiones a los servicios web (HTTPS), lo cual constituyen limitaciones del sistema que necesitan ser resueltas para mayor usabilidad del mismo.

Dado esta problemática se tiene como **problema de investigación**:

¿Cómo controlar el acceso de los usuarios a la red de cargadores eléctricos en Cuba?

Se identifica, como **objeto de estudio**: el control de acceso de los usuarios a la red de cargadores eléctricos en Cuba.

El **objetivo general** de este proyecto es desarrollar un módulo que permita controlar el acceso de los usuarios a la información de la red de cargadores eléctricos en Cuba. Además se establece como **campo de acción**: los mecanismos de control de acceso para los usuarios a dicha red.

Los **objetivos específicos** son:

- ✓ Realizar un análisis detallado sobre los modelos de control de acceso existentes para poder seleccionar el más adecuado en relación a nuestro escenario.
- ✓ Analizar las herramientas existentes en el mercado que tengan las capacidades de brindar un control de acceso hacia otros sistemas informáticos.
- ✓ Definir los componentes que se utilizarán, de acuerdo al modelo de control de acceso seleccionado y el escenario en donde se desarrollará este proyecto.
- ✓ Diseñar los artefactos e implementar la propuesta de solución para resolver dicha problemática.
- ✓ Integrar el sistema de control de acceso al sistema central de cargadores eléctricos.
- ✓ 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 control de acceso de usuarios a redes de cargadores eléctricos a nivel internacional que pueden ser aplicadas en la solución, y analizar también las fuentes bibliográficas que relacionan elementos relevantes para la investigación.

Métodos Empíricos:

- ✓ Observación: se utilizó para observar las distintas herramientas que permiten el control de acceso de usuarios a la red de cargadores eléctricos, 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 comprensión 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 Conceptos asociados:

Los conceptos asociados son de vital importancia para una mejor investigación y comprensión del módulo que se desea desarrollar, para ello se tuvo en cuenta los conceptos de sistemas de control de acceso, los tipos de control de acceso que existen y los mecanismos de control de acceso utilizados por dichos sistemas.

1.1.1 Sistemas de control de acceso:

Un sistema de control de acceso² se puede entender desde dos ámbitos fundamentales. Desde la vertiente física podría definirse como un dispositivo, el cual puede estar formado por un software + un hardware, que va a permitir o restringir el acceso o la entrada de un cliente o de un trabajador utilizando un mecanismo o sistema de identificación. (TD Sistemas, 2018)

Por otra parte y desde el ámbito de la seguridad informática podría decirse que no son más que aquellas herramientas o aplicaciones que tienen como objetivo gestionar quién está autorizado para acceder a determinados sistemas informáticos y

² Un concepto generalizado de control de acceso hace alusión al método que permite probar que los usuarios son quienes dicen ser. El control de acceso es sumamente importante para que todos los usuarios tengan el acceso correspondiente a datos y recursos de sistema. Más que nada, en una serie de restricciones que se van aplicando de acuerdo a los datos y/o recursos a los cuales se desea acceder. Se basa en los procesos de autenticación y autorización. (Fernández, 2021)

a los recursos que contienen. Este último concepto está estrechamente ligado a lo que se quiere lograr, ya que dichos sistemas permitirán restringir o permitir el acceso a sistemas informáticos, bases de datos y otros servicios de información. Además detectan accesos no autorizados y poner a su vez en marcha mecanismos para evitarlos. (Tablado, 2021)

Es fundamental saber como funcionan los sistemas de control de acceso. Estos se basan en tres principios básicos: la identificación, la autenticación y la autorización. Primeramente proceden a la identificación del usuario. Para eso utilizan diversos métodos como pueden ser las huellas dactilares, las tarjetas de identificación o el reconocimiento por voz, entre muchos otros. Luego se aplica la autenticación, la cual detecta si la persona que está intentando acceder se encuentra en la base de datos y si cuenta con los permisos necesarios, lo que no es más que verificar la identidad del usuario. Una vez que el sistema ha identificado y verificado la identidad de dicho usuario, el mismo procederá a autorizar o denegar su acceso a las instalaciones o a los sistemas informáticos.

Por tanto, se quiere desarrollar un sistema de control de acceso visto desde la perspectiva de seguridad informática, que permita restringir o permitir el acceso de los usuarios a la información de la red de cargadores eléctricos. Es necesario conocer además los tipos de sistemas de control de acceso que existen, para poder llegar a la selección del más apropiado.

1.1.2 Tipos de control de acceso:

Existen dos tipos de control de acceso esenciales para estos sistemas: (TD Sistemas, 2018)

- ✓ Sistemas de Control de Acceso Autónomos.
- ✓ Sistemas de Control de Acceso en Red.

Sistemas de Control de Acceso Autónomos:

Son sistemas que permiten básicamente controlar una o más puertas, sin la necesidad de estar conectados a un PC o a un sistema central. Este tipo de sistema no guarda registro de eventos, lo cual es la principal limitante. Existen algunos controles de acceso autónomos que tampoco pueden limitar el acceso por horarios o por grupos de puertas, esto depende de cuan robusta sea la marca. Se conoce que los más sencillos solo usan el método de identificación (ya sea clave, proximidad o biometría) como una "llave" electrónica. (TD Sistemas, 2018)

Sistemas de Control de Acceso en Red:

Estos sistemas son integrados a través de un PC local o remoto, en la cual se hace uso de un software de control de acceso que permite tener un registro de todas las operaciones realizadas sobre el sistema, como por ejemplo, el registro de las personas que entran o salen del centro. Esto es bastante provechoso para quienes interactúan con el mismo. Dichos sistemas pueden ir desde aplicaciones sencillas hasta sistemas muy complejos y sofisticados, (se pueden realizar combinaciones complejas), dependiendo de cómo las partes interesadas en su desarrollo lo requieran, o de las funcionalidades que requieren adaptarse a cada necesidad. (TD Sistemas, 2018)

Teniendo en cuenta los objetivos de este proyecto, como es poder acceder a la información de una red de cargadores eléctricos, se llega a la conclusión de que un sistema de control de acceso en red es el adecuado para resolver nuestra problemática. Sin embargo es necesario conocer que existen diversos mecanismos para aplicar el control de acceso, independientemente del tipo de sistema que sea. Estos serán detallados a continuación.

1.1.3 Mecanismos de control de acceso:

Basándonos en las restricciones y control de acceso a sistemas, podemos encontrar tres tipos principales de mecanismos de control de acceso: Control de Acceso Discrecional (DAC), Control de Acceso Basado en Roles (RBAC) y Control de Acceso Obligatorio (MAC).

Control de Acceso Discrecional (DAC):

El control de acceso discrecional es un mecanismo de acceso que está presente en la mayoría de sistemas operativos. Consiste en restringir el acceso a objetos basándose en la identidad de los sujetos que pretenden operar o acceder sobre ellos. Es el propio dueño del objeto quien determina qué usuarios y con qué privilegios acceden a sus objetos. Esto lo que significa es que dichos accesos serán concedidos a los usuarios en base a las reglas que el propio dueño de los datos especifica. De esto modo cualquier acción de lectura, escritura o ejecución que no se ajuste a los permisos dados por el propietario, será denegado. (López, 2019)

Control de Acceso Basado en Roles (RBAC):

Es un mecanismo de roles que aporta mayor versatilidad que el discrecional, el cual consiste en definir perfiles (roles) a los que se les atribuyen una serie de características que aplican sobre los permisos y acciones que pueden llevar a cabo, incluyendo además el control sobre otros perfiles. Con este mecanismo se segmenta y se organiza de manera eficaz el acceso a los objetos y las tareas. Se puede afirmar que este tipo de control de acceso es de los más utilizados. (López, 2019)

Control de Acceso Obligatorio (MAC):

Este mecanismo de acceso se complementa con anteriores, añadiendo una capa más de seguridad para el control de acceso y de privilegios. Se basa en etiquetar cualquier elemento del sistema y determinar las diferentes políticas de control de acceso, de modo que, cualquier operación que realice un sujeto sobre un objeto será comprobado con las etiquetas y se aplicarán las políticas MAC establecidas para determinar si la operación está permitida, no importando aún si se han cumplido otros controles de seguridad. Es responsabilidad de un usuario privilegiado establecer las políticas centrales MAC que gobernarán los controles a aplicar según el etiquetado establecido. (López, 2019)

Dado que se pretende interactuar con una multitud de usuarios se ha considerado utilizar el mecanismo de control de acceso basado en roles ya que aporta mayor versatilidad para este tipo de ambiente y además es uno de los más utilizados por su facilidad.

1.2 Sistemas homólogos a nivel internacional:

En el mundo existen algunos sistemas o herramientas que nos permiten dar solución a diversos problemas relacionados con el control de acceso de usuarios. Sin embargo se seleccionaron dos sistemas homólogos que cumplen con las características y funcionalidades necesarias para darle solución a la problemática planteada, y que además están relacionados con el proceso de carga de vehículos eléctricos en estaciones de carga. Para el análisis fue tomado en cuenta como característica esencial el protocolo utilizado para la comunicación, que debe ser el protocolo OCPP, pues es el protocolo con el cual trabaja el sistema de gestión de estaciones de carga desarrollado por la UCI. A continuación, se expone el análisis de los sistemas.

1.2.1 Plataforma Cosmos:

La empresa española CIRCUTOR ha desarrollado soluciones integrales para la eficiencia energética aplicables en un gran número de sectores: como por ejemplo en: generación, industria, sector terciario e incluso doméstico. Una de sus grandes soluciones ha sido la plataforma Cosmos, que no es más que una plataforma basada en la nube, que utiliza el protocolo OCPP 1.6J, el cual recopila datos de un conjunto específico de cargadores de VE y lo ayuda a crear y administrar su propia red de recarga.

A través de esta plataforma los usuarios utilizan las credenciales de acceso web para verificar facturas, monitorizar puntos de recarga o crear nuevos usuarios. Esta última funcionalidad es dependiendo del tipo de perfil de usuario proporcionado. Hay solo dos tipos de perfil de usuario:

- ✓ Profesional: permite gestionar cambios dentro de una compañía, como por ejemplo: creación de informes, agregar o eliminar usuarios, etc.
- ✓ Avanzado: permite ver y monitorizar todas las funciones dentro de una compañía, como pueden ser: descarga de facturas, control de puntos de recarga y agregar o eliminar clientes.

Es necesario que los puntos de recarga usen el protocolo OCPP1.6-J para así poder establecer comunicación con la plataforma. El cliente debe tener al menos un identificador registrado y se pueden agregar tantos como sea necesario. Para el registro de un cliente es necesario el identificador de la tarjeta RFID del mismo, por lo que debe poseer una tarjeta RFID.

Algunas de las funcionalidades principales que provee Cosmos son:

-Clientes:

- ✓ Agregar un cliente: es necesario el identificador de la tarjeta RFID.
- ✓ Editar un cliente: Modificar propiedades de un cliente.
- ✓ Eliminar: Eliminar el cliente marcado.
- ✓ Actividad del cliente: muestra todas las transacciones de recarga realizadas por un cliente, como por ejemplo: punto de recarga utilizado por el cliente, dónde tiene lugar la recarga, energía entregada al vehículo kWh.
- ✓ Mostrar facturas VE: muestra un listado de informes enviados a un cliente.

Además permite realizar otras funcionalidades relacionadas con las compañías, los puntos de recarga, los equipos y los informes que genera. (Circutor, S.A, 2019)

Acceso y pago con tarjeta RFID:

Es necesario instalar en el equipo un lector de tarjeta RFID, el cual permite que solo puedan realizar la recarga los usuarios con la tarjeta activada. En el caso de aparcamientos en multipropiedad, el administrador de la finca puede suministrar tarjetas para realizar la gestión de consumos y los importes. (CIRCUTOR, 2021)

1.2.2 Plataforma EVcharge:

EVCharge es una plataforma de gestión para la movilidad eléctrica diseñada y gestionada por la empresa española Etecnic, especializada en ingeniería eléctrica y movilidad sostenible. La misma gestiona tanto cargadores como clientes, usuarios, energía, tarifas y facturación con facilidad y en tiempo real.

Puede hacer uso de ella cualquier operador, organismo o empresa que disponga de varios puntos de recarga y que desee tener el control total de todas las operaciones. Posee varios módulos disponibles para gestión, como por ejemplo: tarifas, facturación, gestión de potencia contratada, entre otros. Además tiene la potencia suficiente para administrar redes de recarga. (EVcharge, 2022)

Entre las principales características de esta plataforma están:

- ✓ Gestiona más de 150 dominios, más de 370 cargadores y más de 2 millones de usuarios.
- ✓ Está configurada en múltiples idiomas.
- ✓ Es una plataforma multiusuario con acceso seguro y gestión de roles.
- ✓ Gestiona y controla todas las transacciones realizadas por el usuario.
- ✓ Comercialización de servicios a sus usuarios.
- ✓ Gestiona solicitudes de Operadores de Sistemas de Distribución (DSO's).
- ✓ Vinculación del consumo del punto de recarga con el usuario a través de las tarjetas RFID.
- ✓ Es adaptable a las necesidades del cliente, tanto en total de puntos de recarga como en cantidad de gestión de usuarios, vehículos y procesos.

Esta plataforma posee además una versión móvil muy popular y usada por los usuarios, la cual pueden utilizar todos los usuarios de forma gratuita. Entre sus principales características están:

- ✓ Registro fácil y rápido: permite que se pueda acceder a través de las cuentas de Google, Facebook o Apple para un inicio rápido.

- ✓ Detalles de carga: Te muestra las estadísticas de la carga así como de cargas anteriores.
- ✓ Pago seguro: permite añadir una o varias tarjetas con facilidad y elegir la que más nos convenga en cada carga.
- ✓ Fácil búsqueda: permite utilizar el mapa o listado de cargadores de proximidad para localizar y navegar hasta el cargador que se desea ir.

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

Tabla 1: Sistemas homólogos (Elaboración propia)

Elementos críticos	Cosmos	EVcharge
Modalidad	WEB	WEB, Aplicación móvil
Protocolo de comunicación	OCCP (1.6J)	OCCP
Tipo de control de acceso	Sistema de control de acceso en red.	Sistema de control de acceso en red.
Mecanismo de control de acceso	Control de Acceso Basado en Roles (RBAC). Posee dos roles fundamentales: Profesional y Avanzado.	Control de Acceso Basado en Roles (RBAC).
Método de identificación de usuarios	Contraseña	Contraseña para la modalidad web. En el caso de la aplicación móvil permite identificarse mediante las cuentas en Google, Facebook o Apple.

Forma de pago	Tarjeta RFID	Tarjetas RFID o Tarjetas Moneder, tarjetas de Crédito, Pago en caja a través de API.
---------------	--------------	--

Al analizar detalladamente estos sistemas se puede concluir que no serían la solución óptima para resolver la problemática planteada, a pesar de que aseguran el control de acceso de los usuarios y poseen funcionalidades que podrían ser útiles. Esto se debe a que usar aplicaciones o sistemas desarrollados por terceros podría traer grandes riesgos para la seguridad informática de los datos, aumentando las probabilidades de que se cometan fraudes y el robo de identidad por parte de atacantes. Desafortunadamente, se desconoce si los desarrolladores adoptaron un enfoque responsable en lo que respecta al almacenamiento y la recopilación de datos, lo que hace que los usuarios expongan su información personal y que dicha información termine en manos no confiables. Además, es posible que los ciberdelincuentes no solo roben los datos y credenciales personales del usuario, sino que también obtengan acceso, como por ejemplo, a su vehículo eléctrico, pudiendo generar amenazas físicas. Tampoco tienen la opción de enviar comentarios, lo que hace imposible dar a conocer algún problema o solicitar más información sobre la política de privacidad de la aplicación. Por otra parte se sabe que para registrar un cliente en la plataforma Cosmos es necesario que el usuario posea una tarjeta RFID, lo que significa que solo podrían utilizar esta plataforma los usuarios que posean este tipo de tarjeta, limitando su uso para los demás usuarios. Sin embargo vale reconocer que la plataforma Cosmos posee varias funcionalidades que serían útiles para nuestro sistema, como son: el registro, modificación y eliminación de usuarios y la gestión de todas las actividades realizadas por el cliente. Por tanto es necesario buscar una solución que logre proteger la seguridad de los datos del usuario y que pueda ser usada por cualquier usuario.

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 ágiles. 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. (Santander Universidades, 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. (Santander Universidades, 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)

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. (IONOS DigitalGuide, 2018)

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. (Santander Universidades, 2021)

Podemos destacar entre sus características fundamentales las siguientes:

- ✓ Programación orientada a objetos (POO): Python, al igual que otros populares lenguajes como es el caso de Java o C++, es un lenguaje orientado a objetos, mediante el cual podemos expresarnos de forma similar como lo haríamos en la vida real. Esto se debe a que permite representar conceptos cotidianos en un programa.
- ✓ Lenguaje interpretado: Python es un lenguaje donde no es necesario compilar, ya que es interpretado y no compilado. Los intérpretes que se utilizan con este lenguaje se encargan de ejecutar dichos programas a través de scripts propios.

- ✓ **Multiplataforma:** Este lenguaje puede ser ejecutado en casi cualquier sistema operativo siempre que se cuente con un intérprete adecuado para ello. Por ejemplo en: Linux, Windows, UNIX, Mac OS, etc.
- ✓ **Lenguaje open source (de código abierto):** Este lenguaje no requiere de licencias de pago para empezar a trabajar con él, o lo que es lo mismo, es totalmente gratuito.
- ✓ **Ampliamente respaldado:** Python posee características y funcionalidades que lo hacen muy interesante, generando así una comunidad de usuarios muy grande a su alrededor, lo cual puede ser de utilidad a la hora de buscar información o pedir ayuda para desarrollar cualquier tipo de programa o algoritmo.
- ✓ **Es polivalente:** Python se utiliza para infinidad de proyectos y aplicaciones diferentes, como por ejemplo: aprendizaje automático (Machine Learning), inteligencia artificial (IA), big data y análisis de datos, operaciones matemáticas, visualización de datos, programación de apps, desarrollo web, desarrollo de videojuegos y gestión financiera. (Miteris, 2021)

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. (Capterra, 2018)

Entre las funciones principales de esta herramienta están:

- ✓ Creación de diagramas: nos permite crear muchos tipos de diagramas: de negocios, técnicos, generales, así como de flujo, UML, mapas mentales y muchos otros.
- ✓ Editor para arrastrar y soltar, por lo que se ahorra tiempo en cada paso para crear, por ejemplo, diagramas de cualquier tipo.

1.5.2 Framework: Django:

Uno de los frameworks más conocidos y usados actualmente, sobre todo para sistemas web es Django, y no es de extrañar ya que es un framework web diseñado con el fin de realizar aplicaciones de cualquier complejidad, en unos tiempos muy razonables. Está escrito en Python, uno de los lenguajes más sencillos que existe (una de las razones por la que fue seleccionado este framework, pues es el lenguaje en que se va a programar).

Entre las características principales que posee, y que además lo convierte en una buena opción para implementar aplicaciones, están las siguientes:

- ✓ Es rápido en el desarrollo de una solución: Increíblemente, si se tiene prisa por terminar un proyecto o se quiere reducir costes (sobre todo tiempo), se puede utilizar este framework ya que permite construir una aplicación muy buena en poco tiempo y terminarla tan pronto como sea posible.
- ✓ Es muy escalable: Se puede utilizar para un desarrollo sencillo, hasta uno mucho más complejo. Esto quiere decir que se puede pasar desde muy poco a una aplicación enorme perfectamente, que funcione rápido y sea estable.
- ✓ Es altamente seguro: Django implementa por defecto algunas medidas de seguridad, incluyendo mecanismos para proteger bases de datos, formularios y JavaScript. Por ejemplo, entre las más clásicas están: que no haya Inyección SQL, que no haya Falsificación de Petición en Sitios Cruzados (CSRF por sus siglas en inglés) y que no haya Secuestro de Clic (Clickjacking) por JavaScript. Todo esto lo maneja de manera muy sencilla.
- ✓ Es increíblemente versátil: Con el tiempo Django ha ganado una popularidad tremenda que se puede usar para el propósito que se desee.

- ✓ Viene cargado de funciones y paquetes útiles: Django tiene implementado, se puede decir, que cualquier cosa que se necesite realizar, sólo hay que adaptarla a nuestras necesidades. Por ejemplo, tiene docenas de capacidades adicionales para manejar actividades de desarrollo web, como es la autenticación de usuarios, la administración de contenidos y otras tareas de forma inmediata. Además tiene módulos de la comunidad, paquetes Python, y aplicaciones que trae por defecto, que son muy útiles.
- ✓ Posee una increíble interfaz para acceso a la base de datos: Su Mapeo Objeto Relacional (ORM por sus siglas en inglés), que es su interfaz para acceso a la base de datos, nos permite hacer consultas maravillosamente, por lo que es una herramienta muy buena. (Bueno, 2018)

1.5.3 Entorno de desarrollo integrado (IDE): PyCharm:

Para poder facilitarle al programador el desarrollo de un software, es necesario hacer uso de un Entorno de Desarrollo Integrado, conocido por sus siglas como IDE. PyCharm por ejemplo es, por así decirlo, un poderoso entorno de desarrollo integrado, considerado por muchos como el IDE excepcional para programar en Python.

Aunque es un IDE con una versión de pago, posee una versión gratuita, de la cual se puede decir que cubre a la perfección las necesidades para desarrollar proyectos. Algunas de sus características importantes, que lo hace excepcional es que:

- ✓ Dispone de un depurador de código, lo cual resulta muy útil para examinar el código y depurarlo sin necesidad de salir de él.
- ✓ Posee un inspector de código en tiempo real, que nos advierte de posibles errores en la sintaxis y a su vez nos brinda sugerencias para solucionar dichos errores.
- ✓ Plegado de código, lo que nos permite evitar distracciones y que sea cómodo navegar por códigos extensos.
- ✓ Sistema de autocompletado que le da un toque de inteligencia al IDE.

- ✓ Todo viene incluido en un mismo paquete, donde una vez realizado el proceso de configuración inicial, se está listo para empezar a programar. (Zeokat, 2018)

1.5.4 Gestor de base de datos: PgAdmin:

PgAdmin es una herramienta indispensable creada con el fin de gestionar y administrar PostgreSQL, la cual es actualmente la base de datos de código abierto más avanzada del mundo. La misma cuenta con varias versiones pero actualmente la más utilizada es PgAdmin 4, que será la utilizada para este proyecto.

Dicha versión posee características que la hacen especiales, por ejemplo: está diseñada para funcionar tanto en escritorio como en un servidor web, posee un completo Panel de control, que nos permite monitorizar el estado del servidor y de las bases de datos, y un espectacular cambio visual, incluyendo un conjunto de iconos actualizado y fuentes incrustadas. Además posee mejoras de velocidad significativas, sobre todo en el momento de inicio y en la herramienta de consulta (Query Tool). (Morales, 2018)

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 autenticación de usuarios permitió identificar elementos fundamentales para definir posteriormente la propuesta de solución, seleccionando los sistemas de control de acceso en red como variante principal y el mecanismo de control de acceso basado en roles para aplicarlo a los mismos, auxiliándose de la modalidad Web.
- ✓ La identificación de las características propuestas para la solución, el análisis de las tecnologías que distinguen a soluciones similares para la autenticación de usuarios, 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, y como lenguaje de programación Python con el framework Django.

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: Registrar usuario.

RF2: Modificar usuario.

RF3: Eliminar usuario.

RF4: Listar usuario.

RF5: Buscar usuario.

RF6: Autenticar usuario.

RF7: Cerrar sesión.

RF8: Consultar la ubicación de los puntos de carga existentes.

RF9: Mostrar la distancia de un punto de carga a otro.

RF10: Mostrar estado de los cargadores.

RF11: Mostrar informe de las actividades del cliente.

RF12: Editar mi perfil de usuario.

2.1.2 Requisitos no funcionales:

Usabilidad:

RNF1: El sistema debe proporcionar mensajes de error que sean informativos y orientados a usuario final.

Seguridad

RNF2: Se debe restringir el acceso a los recursos en dependencia de los permisos del usuario autenticado.

RNF3: Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos.

Software

RNF4: El sistema debe funcionar con los navegadores web Google Chrome y Mozilla Firefox en sus versiones 78.0 o superiores.

Diseño e implementación:

RNF5: El sistema debe utilizar como gestor de base de datos PostgreSQL en su versión 11.0.

RNF6: El sistema debe emplear como lenguaje de programación Python en su versión 3.9.

Apariencia o Interfaz externa:

RNF7: El sistema debe utilizar en el diseño de sus páginas color blanco y azul con tonalidades claras y oscuras.

2.2 Propuesta de solución:

En función de dar cumplimiento al objetivo planteado y ajustándose a las necesidades presentes se determina realizar un módulo de control de acceso basado en un sistema web que permita controlar el acceso de los usuarios a la información de la red de cargadores eléctricos. Para ello se utilizará el método de identificación por contraseña. Una vez autenticado y autorizado el usuario podrá acceder a los recursos o funcionalidades que le sean permitidos, estos recursos no son más que

ciertas informaciones. Esta información puede ser: cantidad de cargadores que hay en un área determinada, cantidad de cargadores disponibles para su uso, etc. Este sistema contará mínimo con dos roles o actores fundamentales: el cliente que sería el interesado en utilizar y saber toda la información sobre los cargadores eléctricos, y el administrador que sería el encargado de la correcta gestión de los mismos.

Para mayor comprensión se ha elaborado la figura que se muestra a continuación:

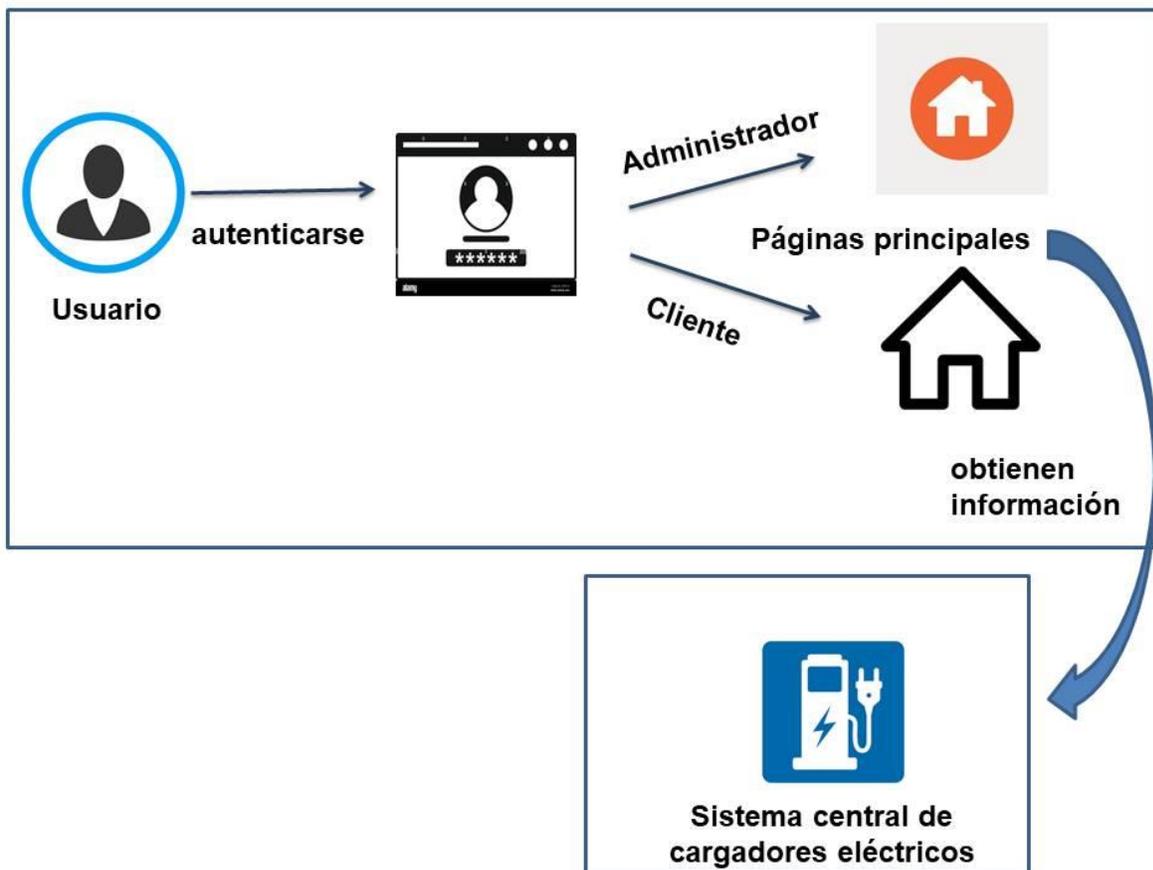


Figura 2: Propuesta de solución (Elaboración propia)

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 RF1 y RF6 para ser desarrollados a continuación. Las demás pueden ser encontradas en el Anexo 1.

Tabla 2: Historia de usuario # 1 (Elaboración propia)

Historia de usuario	
Número: 1	Nombre: Registrar usuario.
Usuario: Administrador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alta
Tiempo estimado: 1 semana	Iteración asignada: 1
Programador responsable: María Magdalena Velázquez Viego	
Descripción: El administrador del sistema tendrá la opción de registrar usuarios, al ingresar a esta opción se desplegará un formulario con los siguientes campos a introducir: Nombre del usuario, Contraseña, Nombre, Apellidos, Dirección de correo electrónico, Activo, Es staff, Estado de superusuario, Carnét de identidad, Facultad, Rol (Cliente o Administrador). Una vez completado los campos, se verificará que no haya errores y si no los hay se procederá a guardar los datos a través del botón: "Guardar registro". Si los hay se detallarán para que puedan ser corregidos.	
Observaciones: N/A	

Tabla 3: Historia de usuario # 6 (Elaboración propia)

Historia de usuario

Número: 6	Nombre: Autenticar usuario.
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	Riesgo en desarrollo: Baja
Tiempo estimado: 1.5 semana	Iteración asignada: 6
Programador responsable: María Magdalena Velázquez Viego	
Descripción: Una vez creado el usuario (ver HU # 1), le permite iniciar sesión y acceder a las funcionalidades que le fueron asignadas según el rol que desempeñe el mismo. Los datos a introducir son: usuario y contraseña. En caso de error muestra un mensaje con el error detectado.	
Observaciones: N/A	

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)

Existen diferentes patrones arquitectónicos, sin embargo para la propuesta de solución se ha seleccionado una redefinición por así decirlo del patrón Modelo Vista Controlador, que Django llama Modelo-Vista-Plantilla (MVT por sus siglas en inglés).

2.4.1 Arquitectura Modelo Vista Plantilla (MVT):

El Modelo-Vista-Plantilla o MVT (por sus siglas en inglés) es un patrón de arquitectónico que utiliza el framework Django, escrito en Python, y que no es más que una variación del patrón Modelo Vista Controlador.

Está basado en 3 componentes fundamentales: modelo, vista y template (plantilla). Se podrían definir sus elementos principales como:

Modelo: es la capa de acceso a la base de datos, es decir, es el encargado de los datos: como acceder a ellos, las relaciones entre los mismos, consultas, búsquedas, etc.

Vista: es la encargada de la lógica de negocios. La misma accede al modelo solicitando los datos que delega a la plantilla apropiada, por lo que es el puente entre los modelos y las plantillas. Se diferencia de la vista del Modelo Vista Controlador, en que describe cuál dato ve y no cómo lo ve.

Template: conocida como plantilla, es la representación visual de los datos. Contiene todo lo relacionado a la presentación, como las cosas que son mostradas sobre una página web. (Romario Sarmiento, 2022)

A continuación se muestra en la figura el comportamiento de este patrón:

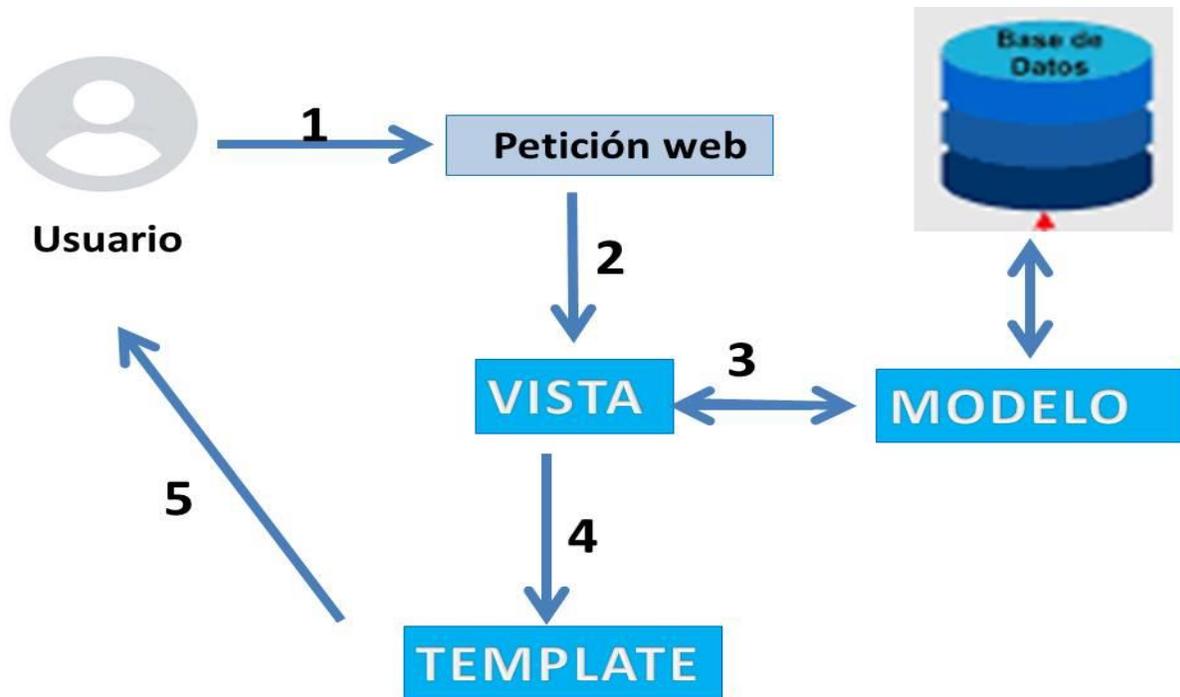


Figura 3: Modelo Vista Template (Elaboración propia)

El flujo de acción de este patrón es el siguiente:

1. El usuario manda una petición a través del navegador, por ejemplo: quiere ver un curso de Python.
2. Esta petición es recibida, y se pasa a la vista.
3. La vista recupera los datos del modelo correspondiente, el cual consulta la base de datos y obtiene todos los datos relacionados con el curso de Python, por ejemplo: videos, información del curso, el título, etc. El modelo responde a la vista con los datos que pidió.
4. Luego, se renderiza el template (plantilla) para que el navegador muestre el HTML resultante, pero integrando los datos dinámicos recuperados del modelo, antes de enviar el HTML final al navegador.
5. Se responde visualmente al usuario a través del navegador.

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)

Con el empleo de estos patrones se le 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 Responsibility 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. Este patrón se evidencia en la clase `UsuarioInfo`, que tiene la información necesaria para que se realice, por ejemplo, la creación de un usuario determinado. (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. Este patrón se evidencia en la clase `UsuarioCreateView`, que a su vez contiene la clase `UsuarioInfo` donde se construye el objeto de tipo usuario, y que realiza llamadas a dichos objetos para ser creados. (Lago, 2022)

3. Controlador: Este patrón es utilizado por la arquitectura MVC, el cual separa la presentación del negocio, en el desarrollo de software. Tiene el fin de crear una o varias clases que se encarguen de tomar los datos que el usuario inserta en la vista y luego enviar la información a las clases donde se utiliza. Este patrón se evidencia en la clase `Home`, sirviendo de intermediario entre la interfaz y el algoritmo que la implementa. (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 la cohesión. Asigna a las clases responsabilidades que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad. Este patrón se utiliza en la clase `UsuarioCreateView` que tiene solo el trabajo de crear usuarios, y que está contenida en métodos internos altamente relacionados siendo así cohesiva. (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. Este patrón se evidencia en todas las aplicaciones web que trabajan con el framework Django, pues cada pieza de las aplicaciones tiene un propósito clave, que puede modificarse sin afectar otras piezas. Por ejemplo, se puede cambiar la URL de cierta parte de la aplicación sin necesidad de verse afectada la implementación subyacente o modificarse el HTML de una página sin tener que tocar el código Python que la renderiza. (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 el patrón GOF utilizado en la solución propuesta:

Decorator (Decorador): Este es un patrón estructural que permite que las clases puedan expandirse dinámicamente mientras el software está funcionando, dicho más fácil, permite añadir responsabilidades extra a objetos concretos de manera dinámica. Su objetivo es hacer que los componentes del software orientado a objetos sean más flexibles y fáciles de reutilizar. Este patrón se evidencia al usarse el decorador `@login_required` (decorador que se usa para proteger las vistas en las aplicaciones web). (IONOS Digital Guide, 2021)

Template method (Método plantilla): Este es un patrón de comportamiento que permite definir un esqueleto común para un algoritmo y dejar los detalles de implementación para los hijos, es decir, permite que las subclasses redefinan algunos pasos de un algoritmo sin cambiar su estructura. Este patrón se evidencia en la clase `AbstractUser`, subclase de la que hereda la clase `User`, modelo original de Django. (Moran, 2018)

Observer (Observador): Este es un patrón de comportamiento que define una dependencia de uno a muchos entre objetos, logrando así que todos los objetos dependientes de un objeto sean notificados y actualizados automáticamente cuando el estado de este objeto cambie. Este patrón se evidencia en la clase `AbstractUser`, en la cual sus objetos son observados, para que, en caso de que haya algún cambio en el estado de alguno de ellos, los objetos dependientes de él en la clase `User` sea notificado. (Moran, 2018)

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 de la propuesta de solución en la siguiente figura:

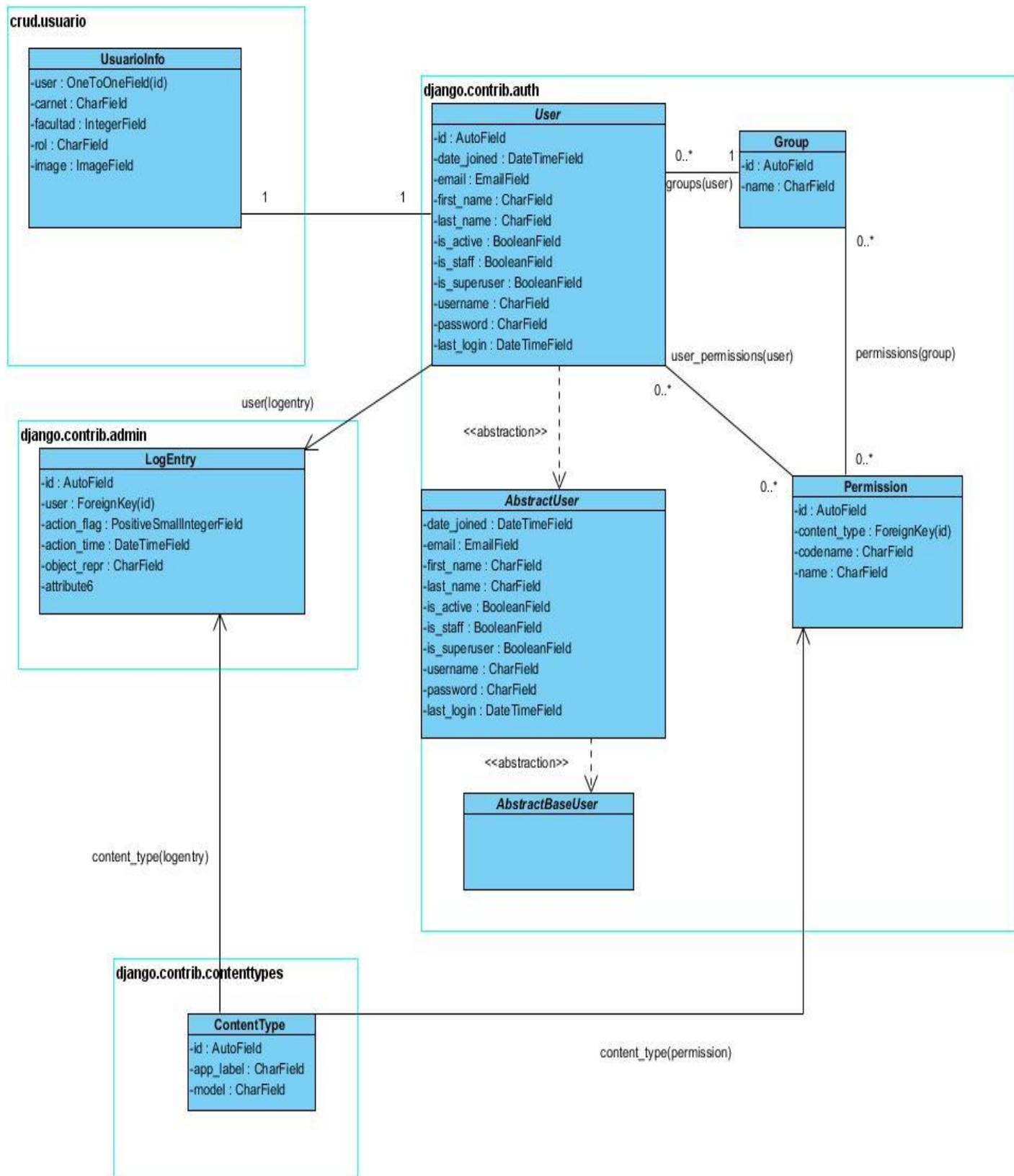


Figura 4: Diagrama de clases

En dicho diagrama de clases, se puede observar cómo están representados los modelos de la aplicación a través de clases, como son:

- ✓ **UsuarioInfo:** es la clase que contiene todos los atributos necesarios para la creación de un usuario. En realidad es un modelo que sirve como un contenedor de los campos extras del modelo `User`, y está relacionado con dicho modelo por el campo `user` de tipo `OneToOneField`.
- ✓ **User:** es la clase o modelo que recibe por defecto la configuración de Django. Hereda de `AbstractUser` que, a su vez, hereda de la clase `AbstractBaseUser`. Normalmente no tiene prácticamente ninguna funcionalidad propia, sino que hereda toda su funcionalidad de `AbstractUser`.
- ✓ **AbstractUser:** es una subclase de la que hereda la clase `User`, que conserva prácticamente toda la funcionalidad del modelo `User` original. Además a su vez hereda de `AbstractBaseUser`.
- ✓ **AbstractBaseUser:** Es la clase que se usa de base para crear el `AbstractUser`. Su funcionamiento es el mínimo y solo cuenta con la función de autenticación. Tienes que indicarle que campo se usará como `username`, para autenticar al usuario.
- ✓ **Group:** es el modelo que tiene que ver con los grupos en Django, que tiene estrecha relación con el modelo `User`. Se usa para organizar los permisos y asignarlos de manera más sencilla.
- ✓ **Permission:** es el modelo que guarda todos los permisos y tiene una relación `ManyToMany` con el modelo `User`.
- ✓ **LogEntry:** es la clase que tiene que ver con la entrada al sitio administrativo de Django. Cada vez que un usuario agrega, elimina o incluso cambia un objeto en el administrador de Django, esa acción se registra en este modelo.
- ✓ **ContentType:** es un modelo especial de Django, el cual se encarga de registrar cada uno de los modelos que existen dentro de nuestra aplicación, ya sea los que nosotros creamos, como los que vienen instalados por defecto.

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 12 requisitos funcionales, los cuales fueron descritos haciendo uso de las Historias de Usuario.
- ✓ Fueron definidos 7 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 del módulo a desarrollar.
- ✓ Se definió la arquitectura del sistema, seleccionando el Modelo Vista Controlador 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. (El Pythonista, 2020)

Entre las reglas más importantes que define están:

1. Reglas de indentación:

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 3.1: Identación de 4 espacios.

```
def deleteuser(request, id):  
    user = get_object_or_404(GeoUser, id=id)
```

```
if user.id == request.user.id:
    messages.add_message(request,
        messages.ERROR, 'No puedes borrar a ti mismo.',
        extra_tags='alert-danger')
    return HttpResponseRedirect(reverse('edituser', args=(user.id,)))
else:
    user.delete()
return redirect(to="listuser")
```

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 3.2: Líneas limitadas.

```
def logoutredirect(request):
    logout(request)
    return redirect('login')
```

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 guión 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 minúscula separadas por un guión bajo, aplicándose éstos tanto como sea necesario para mejorar la legibilidad. Las *constantes* deben ser en mayúsculas. Ejemplo:

Código fuente 3.3: Nombre de clases y de variables.

```
class UsuarioCreateView(CreateView):
    model = Usuario
    form_class = UsuarioForm
    template_name = 'usuario/crear.html'
    success_url = reverse_lazy('listar')
```

Código fuente 3.4: Nombre de funciones.

```
def form_valid(self, form):
    login(self.request, form.get_user())
    return HttpResponseRedirect(self.success_url)
```

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 3.5: Comentarios de una línea.

```
# Vista para listar usuarios.
def listar(request):
    usuario= Usuario.objects.all()
    context= {'usuario':usuario}
    return render(request,'usuario/listar.html',context)
```

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. (Larrea, 2019)

Estos tipos de pruebas incluyen pruebas unitarias, pruebas de regresión, pruebas de aceptación del usuario, además de muchas otras. Las pruebas funcionales se pueden realizar además usando técnicas de Caja Negra, las cuales realizan pruebas sobre la interfaz del programa a probar, es decir, sobre las entradas y salidas de dicho programa.

Para validar las funcionalidades de nuestra aplicación se decidió realizarle dos tipos de pruebas fundamentales: prueba de Caja Negra mediante la técnica de partición de equivalencia y las pruebas unitarias.

3.3.1 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: Autenticar usuario. Para ver los demás casos de prueba consultar [Anexo 2](#).

Tabla 4: Caso de prueba: Autenticar usuario (Elaboración propia)

Sección: Autenticar usuario					
Id del escenario	Escenario	Usuario	Contraseña	Respuesta del sistema	Resultado de la prueba
1.1	Autenticación exitosa	[V,I,N/A]	[V,I,N/A]	Muestra la página principal de acuerdo al rol del usuario.	Se autenticó satisfactoriamente al usuario.
		V	V		
		mary	zxcv1234*		

1.2	Campos vacíos	I	V	Muestra el mensaje: "Rellene este campo".	Se alertó satisfactoriamente la presencia de campos vacíos.
			zxcv1234*		
1.3	Datos incorrectos	V	I	Muestra el mensaje: "El nombre de usuario o la contraseña son incorrectos".	Se alertó satisfactoriamente la presencia de campos incorrectos.
		mary	zxcv1234		

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_user.py haciendo uso del marco de prueba Pytest:

Código fuente 3.6: Pruebas unitarias a través de Pytest.

```
import pytest
from geo.models import GeoUser, Actividades, PtoCarga
from django.contrib.auth import *
```

```

@pytest.mark.django_db
def test_user_creation():
    user= GeoUser.objects.create_user(
        username='dasdas',
        email="",
        password='1234zxcv*',
        is_staff= True,
        is_superuser= True,
        is_active= True,
        first_name= 'Dasda',
        last_name= 'Perez'
    )

    assert user.username == 'dasdas'
    assert user.is_staff
    assert user.first_name == 'Dasda'
    assert user.is_superuser

@pytest.mark.django_db
def test_user_modificarperfil():
    user = GeoUser.objects.create(username='mary', password='1234qwer*',email="")

    GeoUser.objects.filter(username=user.username).update(password='1234qwerasdf*')
    user.refresh_from_db()

    assert user.password == '1234qwerasdf*'

@pytest.mark.django_db
def test_modificarusuario():
    obj = GeoUser.objects.create(username='lia',email='lia@uci.cu',first_name='Lianet')
    GeoUser.objects.filter(username=obj.username).update(email='lianet@uci.cu')
    GeoUser.objects.filter(username=obj.username).update(first_name='Lianet Lucia')
    obj.refresh_from_db()

    assert obj.email== 'lia@uci.cu'
    assert obj.first_name == 'Lianet Lucia'

@pytest.mark.django_db
def test_user_eliminar():
    a = GeoUser.objects.create(id=1,username='mary', password='1234qwer*', email="")
    user= GeoUser.objects.get(username=a.username)
    user.delete()

```

```

exi = GeoUser.objects.all()

assert user.id not in exi

@pytest.mark.django_db
def test_user_listarusuario():
    a=GeoUser.objects.create(id=1,username='mary', password='1234qwer*', email='')
    b= GeoUser.objects.create(id=2, username='alex', password='1238qwer*', email='')
    exi = GeoUser.objects.all()
    user1= GeoUser.objects.get(username='mary')
    user2= GeoUser.objects.get(username='alex')

    assert user1 in exi
    assert user2 in exi

@pytest.mark.django_db
def test_user_mostraractividades():
    a=Actividades.objects.create(id=1,provincia='Pinar del Río', idpto=1,
usuario='mary',dia='2020-10-18 08:00:00+02')
    b= Actividades.objects.create(id=2,provincia='Artemisa', idpto=2,
usuario='mary',dia='2020-10-18 08:00:00+02')
    c = Actividades.objects.create(id=3, provincia='Artemisa', idpto=3,
usuario='mary',dia='2020-10-18 08:00:00+02')
    exi = Actividades.objects.all()
    acti= Actividades.objects.get(id='3')

    assert acti in exi

@pytest.mark.django_db
def test_user_verestado():
    a=PtoCarga.objects.create(id=1,provincia='Pinar del Río', idpto=1, estado='false')
    b= PtoCarga.objects.create(id=2,provincia='Artemisa', idpto=2, estado='true')
    c = PtoCarga.objects.create(id=3, provincia='Artemisa', idpto=3, estado='true')
    exi = PtoCarga.objects.all()
    acti= PtoCarga.objects.get(id='3')

    assert acti in exi
    assert acti.estado== 'true'

@pytest.mark.django_db
def test_autenticar():
    a = GeoUser.objects.create(id=1, username='mary', password='1234qwer*',
email='')
    u= authenticate(username=a.username,password=a.password)
    user= GeoUser.objects.get(id=1)

```

```
assert user.is_active
```

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

```
✓ Tests passed: 8 of 8 tests - 1s 37ms
"C:\Program Files\Python39\python.exe" "C:\Program Files\JetBrains\PyCharm Community Edition 2020.3.4\plugins\python-ce\help
Testing started at 2:40 ...
Launching pytest with arguments in D:\crud\test

===== test session starts =====
platform win32 -- Python 3.9.3, pytest-7.1.3, pluggy-1.0.0 -- C:\Program Files\Python39\python.exe
cachedir: .pytest_cache
django: settings: crud.settings (from ini)
rootdir: D:\crud, configfile: pytest.ini
plugins: django-4.5.2
collecting ... collected 8 items

test_user.py::test_user_creation
test_user.py::test_user_modificarperfil
test_user.py::test_user_eliminar
test_user.py::test_modificarusuario
test_user.py::test_user_listarusuario
test_user.py::test_user_mostraractividades
test_user.py::test_user_verestado
test_user.py::test_autenticar

===== 8 passed in 12.61s =====
```

Figura 5: 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 puede observar en la siguiente tabla.

Tabla 5: No conformidades (Elaboración propia)

No. NC	Requisito funcional	Descripción	Complejidad	Estado
1	RF 1	Errores de validación del campo carnét de identidad del formulario ya que aceptaba menos de 11 caracteres.	Baja	Resuelta
2	RF 9	No mostraba en el mapa la línea que une al punto uno con el punto dos, para que se visualizara así de forma correcta la distancia entre ellos.	Media	Resuelta
3	RF 12	No guardaba correctamente la imagen.	Media	Resuelta

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

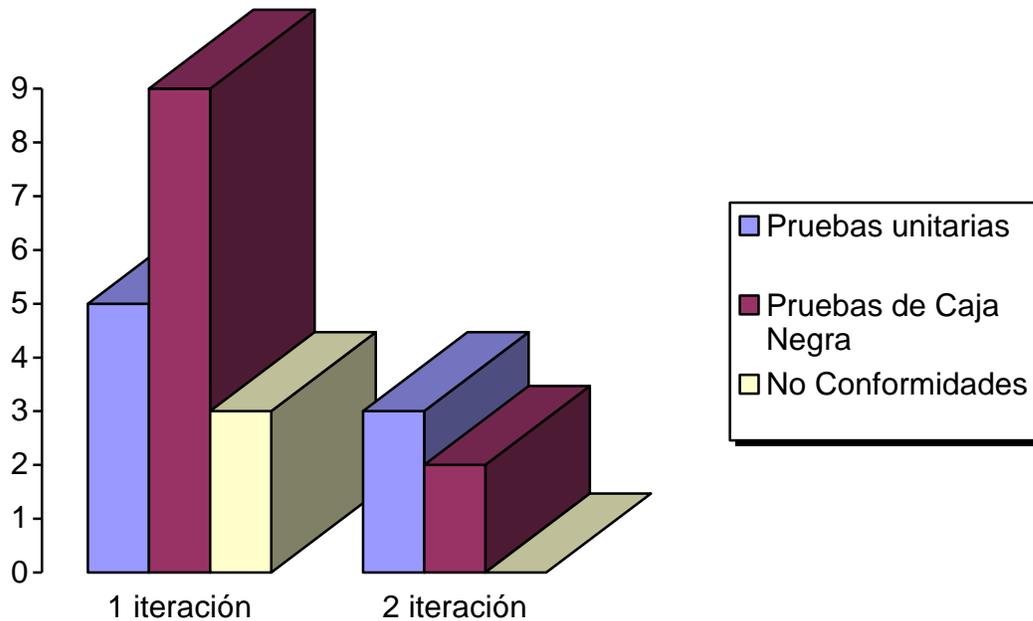


Figura 6: Iteraciones de pruebas (Elaboración propia)

En resumen: fueron realizadas dos iteraciones de prueba, de las cuales se obtuvieron un total de 3 no conformidades (NC) significativas: 2 con complejidad media y otra complejidad baja. En la primera iteración fueron realizados 5 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 3 casos de pruebas unitarias y 2 casos de prueba de Caja Negra (para un total de 8 pruebas unitarias y 11 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:

Considerando los resultados descritos en este informe, la necesidad y el objetivo planteado por la investigación, se arriban a las siguientes conclusiones:

- ✓ Las pruebas funcionales de Caja Negra y las pruebas unitarias aplicadas al módulo comprobaron su correcto funcionamiento y la ausencia de errores, demostrando su capacidad en función del entorno para el cual fue desarrollado.
- ✓ Se obtuvo un módulo de control de acceso que permite o restringe el acceso de los usuarios a la red de cargadores eléctricos, cumpliéndose el objetivo planteado.

Recomendaciones:

Para futuras investigaciones y programas a desarrollar sobre el tema, se recomienda que:

- ✓ Integrar el módulo de control de acceso con una pasarela nacional de pago apenas se encuentre disponible.
- ✓ Valorar la utilización de otras tecnologías, arquitecturas y herramientas en futuras investigaciones, sin dejar de tener en cuenta las ventajas que estas le ofrecen a la solución obtenida.
- ✓ Incluirle otras funcionalidades que le puedan ser útiles a los usuarios que se conecten a esta aplicación.

Bibliografía Consultada:

- Santander Universidades. (21 de diciembre de 2020). Obtenido de <https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html>
- Atareao. (11 de diciembre de 2017). Obtenido de <https://atareao.es/software/ofimatica/pgadmin-y-postgresql/>
- Capterra. (2018). Obtenido de <https://www.capterra.es/software/145716/visual-paradigm>
- IONOS *DigitalGuide*. (26 de octubre de 2018). Obtenido de <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/uml-lenguaje-unificado-de-modelado-orientado-a-objetos/>
- SCRUM MÉXICO. (2 de agosto de 2018). Obtenido de <https://scrum.mx/informate/historias-de-usuario>
- TD Sistemas. (26 de junio de 2018). Obtenido de <https://www.tdsistemas.com/que-es-un-sistema-de-control-de-acceso/>
- El Pythonista. (2 de septiembre de 2020). Obtenido de <https://elpythonista.com/pep-8>
- CIRCUTOR. (8 de abril de 2021). Obtenido de <https://circuitor.com/soporte/formacion/notebooks/vehiculo-electrico/>
- IONOS *Digital Guide*. (19 de febrero de 2021). Obtenido de <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-el-patron-de-diseno-decorator/>
- Miteris. (16 de diciembre de 2021). Obtenido de <https://www.miteris.com/blog/que-es-python-caracteristicas-y-librerias/>
- EVcharge. (4 de abril de 2022). Obtenido de <https://evcharge.net/>
- Álvarez, N. S. (2019). Pruebas de mutación, control sobre variaciones en el código fuente.
- Álvarez, O. M. (28 de marzo de 2022). *Periodismo de Barrio*. Obtenido de <https://periodismodebarrio.org/2022/03/los-vehiculos-electricos-tienen-futuro-en-cuba/>
- Bueno, P. C. (3 de agosto de 2018). *OpenWebinars*. Obtenido de <https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/>

- Circutor, S.A. (2019). *Manual de instrucciones Cosmos*.
- Dorta, D. A. (2020). *Framework para agilizar la aplicación de técnicas basadas en Deep Learning*.
- Durand, S. W. (2017). *Análisis y requerimientos de software: manual autoformativo*.
- Echazarreta, M. (11 de octubre de 2021). *Laravel Tip*. Obtenido de <https://www.laraveltip.com/aplica-mejores-practicas-siguiendo-los-patrones-graps/>
- Fernández, L. (29 de julio de 2021). *RedesZone*. Obtenido de <https://www.redeszone.net/tutoriales/seguridad/control-de-acceso-que-es/>
- Gomez, C. (9 de agosto de 2020). *Diario de QA*. Obtenido de <https://www.diariodeqa.com/post/niveles-de-prueba>
- Lago, N. (24 de febrero de 2022). *Saasradar*. Obtenido de <https://saasradar.net/patrones-diseno-de-software/>
- Larrea, N. P. (2019). *Análisis de la aplicación de pruebas funcionales y pruebas de usabilidad de software en el desarrollo de sistemas web*.
- López, A. (21 de mayo de 2019). *INCIBE-CERT*. Obtenido de <https://www.incibe-cert.es/blog/control-acceso>
- Morales, A. (14 de septiembre de 2018). *MappingGIS*. Obtenido de <https://mappinggis.com/2017/11/descubre-el-nuevo-pgadmin-4-para-trabajar-con-postgis/>
- Moran, N. (2018). *Web Frameworks y patrones de diseño*. Universidad de Los Andes.
- Romario Sarmiento. (2022). *prezi.com*. Obtenido de <https://prezi.com/nd9ydrb01kqv/modelo-vista-template/>
- Sánchez, T. R. (2015). *Metodología de desarrollo para la Actividad productiva de la UCI*. La Habana.
- Santander Universidades. (9 de abril de 2021). Obtenido de <https://www.becas-santander.com/es/blog/python-que-es.html>
- Sommerville, I. (2011). *Ingeniería de Software* (Novena ed.).
- Tablado, F. (7 de julio de 2021). *Grupo Atico34*. Obtenido de <https://protecciondatos-lpdp.com/empresas/control-de-acceso/>

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

Zeokat. (1 de enero de 2018). *Vozidea*. Obtenido de <https://www.vozidea.com/pycharm-el-mejor-ide-para-python>

Anexo 1:

Tabla 6: Historia de usuario # 2 (Elaboración propia)

Historia de usuario	
Número: 2	Nombre: Modificar usuario.
Usuario: Administrador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Tiempo estimado: 0.5 semanas	Iteración asignada: 2
Programador responsable: María Magdalena Velázquez Viego	
Descripción: El administrador del sistema tendrá la opción de modificar usuarios, al ingresar a esta opción se desplegará un formulario con los siguientes campos a modificar: Nombre, Apellidos, Dirección de correo electrónico, Activo, Es staff, Estado de superusuario, Carnét de identidad, Facultad, Rol (Cliente o Administrador). Una vez completado los campos, se verificará que no haya errores y si no los hay se procederá a guardar los datos modificados a través del botón: "Guardar modificación". Si los hay se detallarán para que puedan ser corregidos.	
Observaciones: N/A	

Tabla 7: Historia de usuario # 3 (Elaboración propia)

Historia de usuario	
Número: 3	Nombre: Eliminar usuario.
Usuario: Administrador	
Prioridad en negocio: Alta	Riesgo en desarrollo: Baja
Tiempo estimado: 0.5 semanas	Iteración asignada: 3
Programador responsable: María Magdalena Velázquez Viego	
Descripción: El administrador del sistema tendrá la opción de eliminar usuarios, al ingresar a esta opción se le mostrará un mensaje de confirmación o cancelación. En caso de querer cancelar se selecciona esta opción y el archivo quedará seguro, en caso de querer eliminar se selecciona la opción de eliminar y se procede a la	

eliminación del archivo.
Observaciones: N/A

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

Historia de usuario	
Número: 4	Nombre: Listar usuario.
Usuario: Administrador	
Prioridad en negocio: Media	Riesgo en desarrollo: Baja
Tiempo estimado: 0.5 semanas	Iteración asignada: 4
Programador responsable: María Magdalena Velázquez Viego	
Descripción: El administrador del sistema tendrá la opción de listar los usuarios. Se mostrará una tabla con todos los usuarios registrados en la base de datos.	
Observaciones: N/A	

Tabla 9: Historia de usuario # 5 (Elaboración propia)

Historia de usuario	
Número: 5	Nombre: Buscar usuario.
Usuario: Administrador	
Prioridad en negocio: Baja	Riesgo en desarrollo: Baja
Tiempo estimado: 0.5 semanas	Iteración asignada: 5
Programador responsable: María Magdalena Velázquez Viego	
Descripción: El administrador tendrá la opción de buscar usuarios ya registrados en el sistema. Para eso deberá introducir cualquier criterio de búsqueda válido en la barra de búsqueda que está encima de la tabla con todos los usuarios registrados. Luego de dicha acción se le mostrará el usuario o los usuarios que coincidan con el criterio de búsqueda introducido.	

Observaciones: N/A

Tabla 10: Historia de usuario # 7 (Elaboración propia)

Historia de usuario	
Número: 7	Nombre: Cerrar sesión.
Usuario: Usuario del sistema	
Prioridad en negocio: Media	Riesgo en desarrollo: Baja
Tiempo estimado: 0.5 semanas	Iteración asignada: 7
Programador responsable: María Magdalena Velázquez Viego	
Descripción: El cliente tendrá la opción de cerrar su sesión cuando así lo desee. Una vez que ingrese a esta opción se volverá a la página de autenticación.	
Observaciones: N/A	

Tabla 11: Historia de usuario # 8 (Elaboración propia)

Historia de usuario	
Número: 8	Nombre: Consultar la ubicación de los puntos de carga existentes.
Usuario: Cliente	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alta
Tiempo estimado: 1 semana	Iteración asignada: 8
Programador responsable: María Magdalena Velázquez Viego	
Descripción: El cliente tendrá la opción de consultar la ubicación de los puntos de carga existentes en el país. Al ingresar a esta opción se visualizará un mapa el cual mostrará todas las provincias del país y la ubicación de los diferentes puntos de carga existentes en cada área geográfica. Además al seleccionar uno de los puntos ubicados se mostrará un cartel con algunos detalles sobre el mismo.	
Observaciones: N/A	

Tabla 12: Historia de usuario # 9 (Elaboración propia)

Historia de usuario	
Número: 9	Nombre: Mostrar la distancia de un punto de carga a otro.
Usuario: Cliente	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alta
Tiempo estimado: 1 semana	Iteración asignada: 9
Programador responsable: María Magdalena Velázquez Viego	
<p>Descripción: El cliente tendrá la opción de pedir que se le muestre la distancia que hay de un punto de carga ubicado en el mapa a otro. También puede consultar la distancia entre cualquier punto o área del mapa seleccionada por él. Al ingresar a esta opción se visualizará un mapa el cual mostrará todas las provincias del país y la ubicación de los diferentes puntos de carga existentes en cada área geográfica. El cliente deberá marcar en el mapa haciendo un clic el primer punto y luego de la misma manera seleccionar el segundo punto. Inmediata a esta acción se mostrará encima del mapa la distancia en metros que hay entre esos dos puntos marcados. El cliente debe considerar que esto es simplemente un aproximado, ya que no podrá seleccionar directamente el punto de carga situado, sino acercarse lo más posible a él.</p>	
Observaciones: N/A	

Tabla 13: Historia de usuario # 10 (Elaboración propia)

Historia de usuario	
Número: 10	Nombre: Mostrar estado de los cargadores.
Usuario: Cliente	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Tiempo estimado: 1 semanas	Iteración asignada: 10

Programador responsable: María Magdalena Velázquez Viego
Descripción: El cliente tendrá la opción de buscar información acerca de los cargadores, sobre todo si están ocupados o disponibles en el momento que se consulte la información. Se le mostrará en una tabla una lista con todos los cargadores, su respectiva dirección y provincia, entre otros datos.
Observaciones: N/A

Tabla 14: Historia de usuario # 11 (Elaboración propia)

Historia de usuario	
Número: 11	Nombre: Mostrar informe de las actividades del cliente.
Usuario: Cliente, Administrador	
Prioridad en negocio: Media	Riesgo en desarrollo: Media
Tiempo estimado: 1 semana	Iteración asignada: 11
Programador responsable: María Magdalena Velázquez Viego	
Descripción: Tanto el administrador como el cliente podrán acceder al informe de las actividades realizadas por los clientes. En el caso del cliente solo podrá ver las realizadas por él, mientras que el administrador podrá ver las de cada cliente individualmente. Este informe tendrá por cada actividad hecha: usuario que realizó la recarga, lugar dónde la ha realizado (provincia), punto de carga utilizado, tipo de recarga, día de la recarga, duración de la misma y cantidad de energía entregada al vehículo.	
Observaciones: N/A	

Tabla 15: Historia de usuario # 12 (Elaboración propia)

Historia de usuario	
Número: 12	Nombre: Editar mi perfil de usuario.
Usuario: Usuario del sistema	
Prioridad en negocio:	Riesgo en desarrollo: Baja

Baja	
Tiempo estimado: 0.5 semanas	Iteración asignada: 12
Programador responsable: María Magdalena Velázquez Viego	
Descripción: Cualquier usuario del sistema, ya sea el cliente o el administrador tendrá la opción de editar su propio perfil de usuario. Esto incluye poner una foto de perfil y cambiar su contraseña cuando así lo desee.	
Observaciones: N/A	

Anexo 2:

Tabla 16: Caso de prueba: Registrar usuario (Elaboración propia):

Sección: Registrar usuario								
Id del escenario	Escenario	Nombre de usuario	Contraseña	Nombre	Apellidos	Carnét de identidad	Respuesta del sistema	Resultado de la prueba
2.1	Registro ok	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra la lista de usuarios con el nuevo usuario ya registrado.	Se registró de forma satisfactoria al usuario.
		V	V	V	V	V		
		maritza	zxcv1234*	Maritza	López Pérez	99012333322		
2.2	Campos vacíos	I	V	V	V	V	Muestra el mensaje: "Rellene este campo".	Se alertó de forma satisfactoria la presencia de campos vacíos.
			zxcv1234*	Maritza	López Pérez	99012333322		
		V	I		V	V	Muestra el mensaje: "Rellene este campo".	Se alertó de forma satisfactoria la presencia de campos vacíos.
		maritza		Maritza	López Pérez	99012333322		
		V	V	I	V	V		
maritza López Pérez	zxcv1234*		López Pérez	99012333322	Muestra el mensaje: "Rellene este	Se alertó de forma satisfactoria la presencia		

							campo”.	de campos vacíos.
		V	V	V	I	V	Muestra el mensaje: “Rellene este campo”.	Se alertó de forma satisfactoria la presencia de campos vacíos.
		maritza	zxcv1234*	Maritza		990123 33322		
		V	V	V	V	I	Muestra el mensaje: “Rellene este campo”.	Se alertó de forma satisfactoria la presencia de campos vacíos.
		maritza	zxcv1234*	Maritza	López Pérez			
2.3	Datos incorrectos	V	V	V	V	I	Muestra el mensaje: “Use al menos 11 caracteres”.	Se alertó de forma satisfactoria la presencia de datos incorrectos.
		maritza	zxcv1234*	Maritza	López Pérez	99		
2.4	Usuario existente	V	V	V	V	V	Muestra el mensaje: “Ha ocurrido un error al querer guardar el usuario”.	Se alertó de forma satisfactoria que ya existía el usuario.
		maritza	zxcv1234*	Maritza	López Pérez	990123 33322		

Tabla 17: Caso de prueba: Modificar usuario (Elaboración propia)

Sección: Modificar usuario						
Id del escenario	Escenario	Nombre	Apellidos	Carnét de identidad	Respuesta del sistema	Resultado de la prueba
3.1	Modificación ok	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra la lista de usuarios con el usuario modificado.	Se modificó de forma satisfactoria al usuario.
		V	V	V		
		Maritza	López Pérez	99012333322		
3.2	Campos vacíos	I	V	V	Muestra el mensaje: "Rellene este campo".	Se alertó de forma satisfactoria la presencia de campos vacíos.
			99012333322	99012333322		
		V	I	V	Muestra el mensaje: "Rellene este campo".	Se alertó de forma satisfactoria la presencia de campos vacíos.
		Maritza		99012333322		
		V	V	I	Muestra el mensaje: "Rellene este campo".	Se alertó de forma satisfactoria la presencia de campos vacíos.
		Maritza	López Pérez			

3.3	Datos incorrectos	V	V	I	Muestra el mensaje: "Use al menos 11 caracteres."	Se alertó de forma satisfactoria la presencia de datos incorrectos.
		Maritza	López Pérez	99		

Tabla 18: Caso de prueba: Eliminar usuario (Elaboración propia)

Sección: Eliminar usuario								
Id del escenario	Escenario	Nombre del usuario	Contraseña	Nombre	Apellidos	Carnét de identidad	Respuesta del sistema	Resultado de la prueba
4.1	Confirmar eliminación	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra el mensaje: "¿Está seguro que quiere eliminar?"	Se mostró de forma satisfactoria un mensaje para confirmar.
		N/A	N/A	N/A	N/A	N/A		
4.2	Eliminar ok	N/A	N/A	N/A	N/A	N/A	Muestra el mensaje: "Su archivo ha sido eliminado".	Se eliminó de forma satisfactoria al usuario.
4.3	Eliminación cancelada	N/A	N/A	N/A	N/A	N/A	Muestra el mensaje: "Su archivo está	Se canceló de forma satisfactoria la acción de

							seguro".	eliminar al usuario.
--	--	--	--	--	--	--	----------	----------------------

Tabla 19: Caso de prueba: Buscar usuario (Elaboración propia)

Sección: Buscar usuario							
Id del escenario	Escenario	Nombre del usuario	Nombre	Apellidos	Carné de identidad	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 en una tabla todos los usuarios coincidentes con la búsqueda.	Se buscó de forma satisfactoria a un usuario.
		V	V	V	V		
		maritza	Maritza	López Pérez	99012333322		
5.2	Búsqueda sin resultados	V	V	V	V	Muestra la tabla vacía.	Se mostró de forma satisfactoria que no está el usuario buscado.
		maritza	Maritza	López Pérez	99012333322		

Tabla 20: Caso de prueba: Cerrar sesión (Elaboración propia)

Sección: Cerrar sesión								
Id del	Escenario	Nombre del	Contraseña	Nombre	Apellidos	Carné de	Respuesta del	Resultado de la

esce- nario		usuario				identidad	sistema	prueba
6.1	Cierre de sesión exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Vuelve a la página de autenticación.	Se cerró de forma satisfactoria la sesión.
		N/A	N/A	N/A	N/A	N/A		

Tabla 21: Caso de prueba: Consultar ubicación de puntos de carga (Elaboración propia)

Sección: Consultar la ubicación de los puntos de carga existentes								
Id del esce- nario	Escenario	Nombre del usuario	Contraseña	Nombre	Apellidos	Carnét de identidad	Respuesta del sistema	Resultado de la prueba
7.1	Consulta exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra un mapa con todos los puntos de carga existentes en el país.	Se consultó de forma satisfactoria la ubicación de los puntos de carga.
		N/A	N/A	N/A	N/A	N/A		

Tabla 22: Caso de prueba: Mostrar distancia (Elaboración propia)

Sección: Mostrar la distancia de un punto de carga a otro								
Id del esce- nario	Escenario	Nombre del usuario	Contraseña	Nombre	Apellidos	Carnét de identidad	Respuesta del sistema	Resultado de la prueba

8.1	Selección punto 1	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra en el mapa la ubicación del primer punto marcado.	Se mostró de forma satisfactoria la ubicación del primer punto.
		N/A	N/A	N/A	N/A	N/A		
8.2	Selección punto 2	N/A	N/A	N/A	N/A	N/A	Muestra en el mapa la ubicación del segundo punto marcado.	Se mostró de forma satisfactoria la ubicación del segundo punto.
8.3	Distancia mostrada	N/A	N/A	N/A	N/A	N/A	Muestra la distancia del punto 1 al punto 2.	Se mostró de forma satisfactoria la distancia entre los dos puntos.

Tabla 23: Caso de prueba: Mostrar estado de los cargadores (Elaboración propia)

Sección: Mostrar estado de los cargadores								
Id del escenario	Escenario	Nombre del usuario	Contraseña	Nombre	Apellidos	Carnét de identidad	Respuesta del sistema	Resultado de la prueba
9.1	Muestra	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra en	Se mostró de

	exitosa	N/A	N/A	N/A	N/A	N/A	una tabla	forma
							todos los	satisfactoria
							cargadores	si estaba o
							registrados	no disponible
							en el	un cargador.
							sistema y si	
							están	
							disponibles	
							o no.	

Tabla 24: Caso de prueba: Mostrar informe de actividades (Elaboración propia)

Sección: Mostrar informe de actividades								
Id del escenario	Escenario	Nombre del usuario	Contraseña	Nombre	Apellidos	Carnét de identidad	Respuesta del sistema	Resultado de la prueba
10.1	Muestra exitosa	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Muestra en una tabla todas las actividades del cliente.	Se mostró de forma satisfactoria las actividades del cliente.
		N/A	N/A	N/A	N/A	N/A		
10.2	Sin actividades	N/A	N/A	N/A	N/A	N/A	Muestra la tabla vacía.	Se mostró de forma satisfactoria que no tiene actividades

								dicho cliente.
--	--	--	--	--	--	--	--	----------------

Tabla 25: Caso de prueba: Editar perfil de usuario (Elaboración propia)

Sección: Editar perfil de usuario						
Id del escenario	Escenario	Nombre del usuario	Contraseña	Imagen de perfil	Respuesta del sistema	Resultado de la prueba
11.1	Modificación ok	[V,I,N/A]	[V,I,N/A]	[V,I,N/A]	Se modifica el usuario y vuelve a la página de autenticación.	Se modificó de forma satisfactoria al usuario.
		V	V	V		
		maritza	zxcv1234*	Avatar.jpg		
11.2	Campos vacíos	I	V	V	Muestra el mensaje: "Rellene este campo".	Se alertó de forma satisfactoria la presencia de campos vacíos.
			zxcv1234*	Avatar.jpg		
		V	I	V	Muestra el mensaje: "Rellene este campo".	Se alertó de forma satisfactoria la presencia de campos vacíos.
		maritza		Avatar.jpg		