universidad de león

Escuela de Ingenierías

Industrial, Informática y Aeroespacial

MÁSTER EN INGENIERÍA INDUSTRIAL

Trabajo de Fin de Máster

DESARROLLO DE APLICACIÓN PARA LA GESTIÓN DE RECARGA SIMULTÁNEA DE VEHÍCULOS ELÉCTRICOS

APPLICATION DEVELOPMENT FOR SIMULTANEOUS VEHICLE CHARGING MANAGEMENT

Autor: Marcos Castro de Prado
Tutor: Carlos López Díaz

(Junio, 2022)

1. ÍNDICE GENERAL

2. MEMORIA

ÍNDICE DE CONTENIDO. MEMORIA.

ÍNDICE DE FIGURAS. MEMORIA.

ÍNDICE DE TABLAS. MEMORIA.

2.1. Objeto

El presente Proyecto tiene por objeto el desarrollo de una aplicación de usuario orientada a la gestión de la recarga vehicular efectuada bajo el estándar internacional CSS (Combined Charging System) o Combo, que se recoge en la Norma IEC 62196 y, de manera más específica, el modo 4 (DC) descrito en el mismo (también conocido como *carga rápida*).

La razón fundamental responde al aprovechamiento de la oportunidad de negocio que supone la entrada a un nuevo nicho de mercado ante el surgimiento y estandarización de los vehículos eléctricos y su recarga, especialmente en países desarrollados.

De cara a ofrecer una solución que resulte atractiva para un mayor rango de usuarios potenciales, la aplicación presenta dos modos fundamentales de operación: Administrador y Usuario (minorista). Cada uno de éstos ofrece funcionalidades específicas para facilitar la gestión de la recarga de vehículos eléctricos al consumidor final de acuerdo con sus necesidades particulares.

2.2. Alcance

Se desea llevar a cabo el desarrollo de una aplicación gráfica para la gestión de diferentes funcionalidades relacionadas con la recarga de vehículos eléctricos, de conformidad con el modo 4 (DC) recogido en el estándar CSS, que faciliten el estudio y los procesos de mejora continua en los campos de aplicación en los que opere el usuario final.

El programa, que ha sido denominada comercialmente chargEV, ofrece dos modos fundamentales de operación:

- Administrador:

  El modo Administrador ha sido diseñado específicamente para aquellos usuarios que realicen actividades económicas en las que interviene la recarga simultánea de múltiples vehículos eléctricos (tales como electrolineras, estaciones de recarga situadas en aparcamientos de toda índole, hoteles, etc.).

  Bajo este modo, chargEV permite al Administrador planificar y simular el proceso de recarga múltiple en función de las particularidades de su negocio.

El resultado de cada simulación aporta información de utilidad como:

- La potencia eléctrica nominal total que se precisa contratar de cara a no sufrir penalizaciones.
- Distribución de la potencia agregada instantánea consumida, disponible y en exceso a lo largo del horario de apertura del negocio de manera numérica, textual y gráfica.
- Determinación de la posibilidad de permitir la recarga a un nuevo vehículo (cliente) en función de la disponibilidad de conectores, el porcentaje mínimo permitido de carga para ese modelo, la potencia total contratada y otros parámetros que pueden ser configurados de manera dinámica por el Administrador.

- Usuario (minorista):

  El modo Usuario ha sido diseñado con el objetivo fundamental de agregar y permitir la consulta de datos históricos correspondientes a procesos de carga realizados en el pasado.

  Puesto que no es objeto del presente Proyecto el diseño ni la elaboración de hardware, el modo Usuario incorpora una herramienta simuladora del proceso de recarga para el modelo de vehículo eléctrico configurado por el usuario. La misma servirá como fuente de información de entrada para el resto de utilidades relacionadas con la consulta de datos históricos:

  - Representación de energía consumida y coste económico asociado a cargas pasadas durante períodos de tiempo que el usuario desee.
  - Agrupación y representación de energía consumida por mes durante los últimos 5 (cinco) años para facilitar la detección de patrones de consumo.

  En aquellos casos en los que el usuario final no disponga de un vehículo eléctrico, este modo también puede resultar útil como una herramienta de estudio de mercado que facilite la toma de decisión en aquellas situaciones en las que el

usuario final desee estudiar qué modelo de vehículo eléctrico se ajusta mejor a sus necesidades.

2.3. Antecedentes

En la última década, el resurgimiento del interés por los vehículos eléctricos como solución estructural al consumo global de combustibles fósiles derivado del sector transporte ha servido como motor para el desarrollo y la implementación de estándares que normalicen los procesos de recarga, los conectores y hardware involucrado y las comunicaciones entre el vehículo y el poste (en aquellos casos que aplique).

Como consecuencia, la asociación denominada CharIN (de la que son miembros negocios como Audi, BMW, Daimler, Porsche, Volkswagen, GM, FCA o Bosch, entre otros) desarrolló entre 2013 y 2020 el estándar conocido como CSS (Combined Charging System) para ser implantado en América y Europa. Debido a las diferencias en la tensión nominal de red entre ambos sistemas eléctricos, CSS fue dividido en CSS1 (también llamado Combo 1) y CSS2 (Combo 2), respectivamente.

La norma IEC 61851 representa la guía de referencia que describe los requerimientos generales relacionados con aplicaciones de recarga vehicular en materia de compatibilidad electromagnética (EMC), estaciones de carga AC y DC y las comunicaciones en tiempo real entre el vehículo (cliente) y la estación de carga (servidor) que resultan necesarias para la carga DC (modo 4) o, como es denominada comúnmente, carga rápida.

Por su parte, la norma IEC 62196 (también conocida como CSS o Combo), constituye el estándar internacional que describe todos los aspectos relacionados con el conjunto de conectores eléctricos y los modos de recarga de vehículos eléctricos.

Las figuras inmediatamente inferiores esquematizan los conectores asociados a los sub-estándares CSS1 y CSS2:

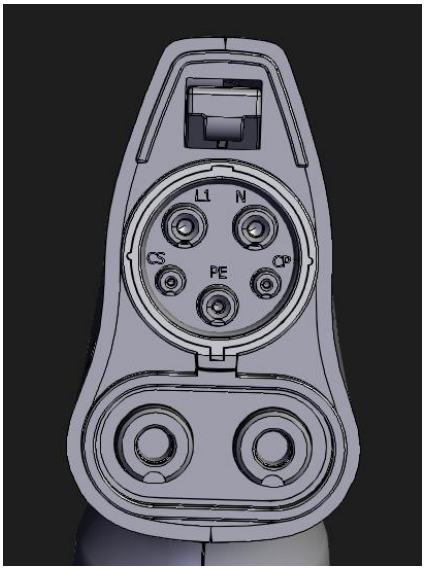


Figura 2.1. Conector CSS1. AC monofásico (arriba) y DC (abajo). (Fuente: Diseño propio).

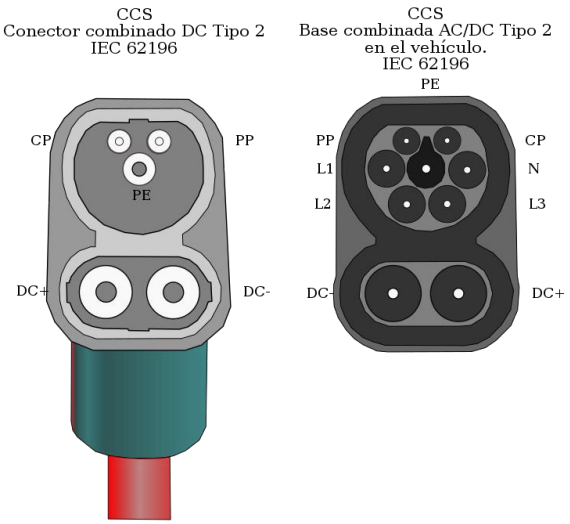


Figura 2.2. Conector CSS2. Izquierda: DC. Derecha: AC/DC. (Fuente: Ajzh2074).

Nótese las similitudes entre ambos conectores: en la parte superior se encuentran los contactos para los modos de carga AC, mientras que la parte inferior está reservada para los dos conectores utilizados para la transmisión de potencia en la recarga DC.

De acuerdo a las especificaciones de la Norma IEC 62196, se consideran 4 (cuatro) modos de recarga con voltajes operativos limitados a máximos de:

- 690VAC 50-60Hz con un límite de corriente nominal máximo de 250A.

- 600VDC con un límite de corriente nominal máximo de 400A.

Los modos de recarga eléctrica vehicular que se describen son:

- Modo 1 (AC):

  Carga lenta desde una base de enchufe doméstico, no industrial, estilo Schuko.

  A pesar de estar limitada por hardware a 16A, típicamente se requiere modificar las instalaciones domésticas para soportar dicha intensidad durante periodos de tiempo prolongados.

Figura 2.3. Modo 1 (AC). Enchufe no dedicado. (Fuente: Schneider Electric).

- Modo 2 (AC):

  Carga lenta desde una base de enchufe doméstico con protección eléctrica en cable, estilo Schuko.

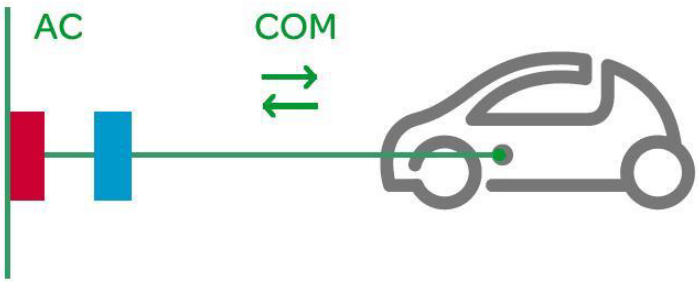


Figura 2.4. Modo 2 (AC). Enchufe no dedicado con protección de cable. (Fuente: Schneider Electric).

- Modo 3 (AC):

  Carga semirrápida o rápida desde enchufe y circuito dedicado para aplicaciones vehiculares (conector CSS o Combo, contactos AC) y funciones adicionales incorporadas para el control y la protección eléctrica.

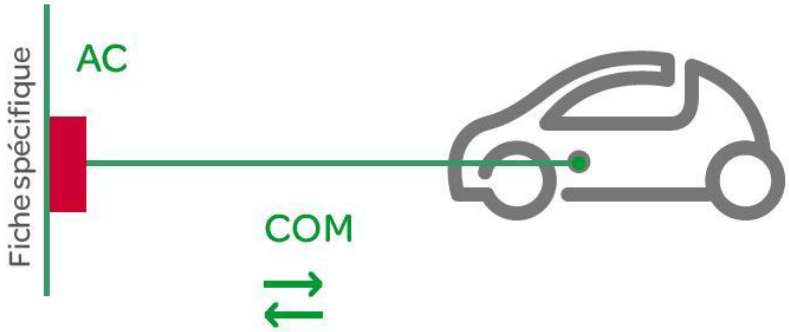


Figura 2.5. Modo 3 (AC). Fijo, enchufe y circuito dedicado. (Fuente: Schneider Electric).

- Modo 4 (DC):

  Carga rápida desde enchufe y circuito específico para aplicaciones vehiculares (conector CSS o Combo, contactos DC). Puede ser llevada a cabo desde una estación de recarga conectada a red con rectificadores o desde un sistema de almacenamiento.

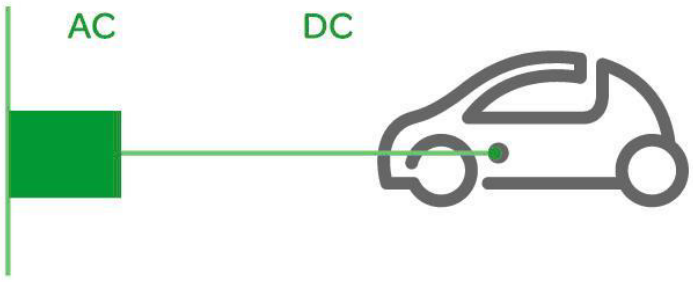


Figura 2.6. Modo 4 (DC). Fijo, enchufe y circuito dedicado. (Fuente: Schneider Electric).

En líneas generales, la norma distingue carga semirrápida como aquella en la que la potencia nominal se encuentra en un rango comprendido entre 15kW y 40kW.

De los modos recogidos en el estándar CSS, el que más se asemeja al modo de uso asociado al repostaje de combustible en vehículos propulsados por motores de combustión es el modo 4 debido, principalmente, a los tiempos de espera reducidos (para un modelo de vehículo eléctrico medio la carga bruta –hasta un SoC del 80%- puede requerir de 30 minutos, aproximadamente).

Es, precisamente, esta característica la que hace a las estaciones de carga que ofrezcan valores de potencia que satisfagan la recarga rápida (DC) simultánea para varios vehículos eléctricos las potenciales beneficiadas de la creación de un nuevo nicho de mercado.

Consecuentemente, se ha decidido centrar el desarrollo de la aplicación objeto del presente Proyecto en dicho ámbito, ofreciendo un servicio en el que los gestores de dichos parques de carga encuentren valor añadido.

Uno de los lenguajes de programación más extendidos y completos a la hora de desarrollar aplicaciones gráficas que deben ejecutar acceso dinámico a bases de datos y archivos locales es Python.

De este modo, las características generales de la aplicación objeto del presente Proyecto, sus requerimientos y puntos de partida son:

i. Desarrollo específico para Windows-based OS.
ii. Desarrollo basado en Python como lenguaje de programación.
iii. Utilización de la librería PyQt5, junto a su componente adicional QtCharts para la representación de gráficas de consumo.
iv. Almacenamiento de configuraciones de usuario en base de datos MySQL.
v. Almacenamiento de curvas de carga potencia-SoC en tablas en formato .CSV.

2.4. Normas y referencias

2.4.1. Disposiciones legales y normas aplicadas

- Norm IEC 61851:2020. Electric vehicle conductive charging system.
- Norm IEC 62196:2022. Plugs, socket-outlets, vehicle connectors and vehicle inlets – Conductive charging of electric vehicles.

- Norma UNE 50132:1994. Numeración de las divisiones y subdivisiones en los documentos escritos.
- Norma UNE 157001:2014. Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico.

2.4.2. Recursos informáticos utilizados

- Entorno de programación:
  - PyCharm (versión 2022.1.1, Community Edition).
  - Python (versión 3.10).
- Lenguaje:
  - Python.
- Librerías y dependencias no nativas:
  - PyQt5 (entorno gráfico).
  - mysql.connector (acceso cliente a base de datos MySQL).
- Base de datos:
  - MySQL Workbench (versión 8.0 CE).

2.4.3. Bibliografía

- Bennett, J. & Ho, D. (2013). *Project Management for Engineers*. Word Scientific Publishing Company.
- Burton, N. (2013). *History of Electric Cars*.The Crowood Press Ltd.
- CharIN Association. (2015). *Combined Charging System Specification (CSS)*.
- Patel, N. (2021). *Electric Vehicles: Modern Technologies and Trends*. Springer.
- PMI. (2017). *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*. 6ª edición. PMI.
- Schwalbe, K. (2006). *Introduction to Project Management*. Course Technology Inc.

2.5. Definiciones y abreviaturas

Las abreviaturas utilizadas durante la redacción del presente Proyecto, así como sus definiciones correspondientes quedan recogidas, siguiendo orden alfabético, en la tabla inferior como se muestra:

Tabla 2.1. Definiciones y abreviaturas. (Fuente: Propio).

Abreviatura	Definición
AC	Alternating Current
API	Application Programming Interface
AENOR	Asociación Española de Normalización
BB.DD.	Base(s) de datos
CSS	Combined Charging System (IEC 61851 e IEC 62196)
CSV	Comma Separated Values
DC	Direct Current
EIA	U.S. Energy Information Administration
EMC	Electromagnetic Compatibility
EN	European Norm
EV	Electric Vehicle
FCA	Fiat Chrisler Automobiles
GM	General Motors
GUI	Graphical User Interface
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
ISO	International Standardization Organization
NEN	NEderlandse Norm
OS	Operating System
PMBOK	Project Management Body of Knowledge
PMI	Project Management Institute
SoC	State of Charge (porcentaje de carga)

UE	Unión Europea
UNE	Una Norma Española

2.6. Requisitos de diseño

La aplicación desarrollada en el marco del presente Proyecto queda compuesta por 3 (tres) componentes fundamentales. A saber:

- Estructura lógica de ejecución.
- Interfaz gráfica de usuario.
- Recursos.

El uso previsible de la misma, en base a los requerimientos de diseño que se describen en los subapartados siguientes, consiste en el cálculo de simulaciones de recarga eléctrica en la que se consideran múltiples vehículos conectados simultáneamente dentro de un horario configurable por el Administrador, así como otras funcionalidades específicas para el usuario minorista que posee un vehículo eléctrico soportado por el sistema.

2.6.1. Descripción del proyecto

De conformidad con la introducción expuesta en párrafos anteriores, el presente Proyecto consiste, fundamentalmente, en el desarrollo de una aplicación para la gestión de recarga simultánea de vehículos eléctricos.

El programa en cuestión ha sido desarrollado para cubrir la necesidad de gestión y planificación de los horarios de recarga vehicular en aquellos negocios que integran, como principal fuente de ingresos operativos o de manera residual, estaciones con múltiples puestos de carga.

No obstante, la aplicación también dispone de un modo Usuario que será comercializado de manera gratuita (en una fase inicial) con funcionalidades y servicios centrados en

facilitar la consulta de datos históricos y el control de costes asociados a la recarga del vehículo eléctrico.

Los subapartados siguientes describen, de manera detallada, las características de cada uno de los 3 (tres) componentes principales en los que se ha dividido el desarrollo y funcionamiento de la aplicación chargEV.

2.6.1.1. Descripción de los recursos

La aplicación chargEV utiliza una serie de recursos informáticos de distinta naturaleza para un gran rango de aplicaciones, entre las que destacan:

- Base de datos.
- Curvas de carga.
- Grafismos y logotipos.

2.6.1.1.1. Base de datos

Se ha embebido una base de datos MySQL bajo el nombre *chargev* para el almacenamiento de información relevante relacionada con ambos modos Administrador y Usuario, quedando conformada por las siguientes tablas:

- *chargev.administrator*:

  Esta tabla tiene por objeto almacenar los datos de configuración del perfil del administrador del sistema y, por ende, únicamente es utilizada bajo el modo de uso correspondiente.

  La tabla siguiente lista las columnas o campos que constituyen la tabla *chargev.administrator*, así como sus tipos de dato y usos previsibles:

Tabla 2.2. Base de datos. Campos de la tabla *chargev.administrator*. (Fuente: Propio).

Campo	Tipo de dato	Descripción
Tabla: chargev.administrator		
admin_id	INT	Número natural identificador del administrador
company_name	VARCHAR(45)	Nombre de la empresa (hasta 45 caracteres)
opening_time	TIME	Horario de apertura del negocio (formato hh:mm)
closing_time	TIME	Horario de cierre del negocio (formato hh:mm)
max_power	FLOAT	Potencia nominal contratada (en kW)

Al usuario de la aplicación, en modo Administrador, se le permite configurar parámetros que definan el perfil de su empresa como el nombre de la misma, horarios en los que el negocio lleva a cabo sus operaciones y la potencial nominal contratada con la distribuidora eléctrica.

- *chargev.user*:

  Esta segunda tabla sirve de almacenamiento para la configuración del perfil del usuario minorista, siendo únicamente utilizada en el modo Usuario.

  La tabla siguiente lista las columnas o campos que construyen la tabla *chargev.user*, así como sus tipos de dato y usos previsibles:

Tabla 2.3. Base de datos. Campos de la tabla *chargev.user*. (Fuente: Propio).

Campo	Tipo de dato	Descripción
Tabla: chargev.user		
user_id	INT	Número natural identificador del usuario
name	VARCHAR(40)	Nombre del usuario (hasta 40 caracteres)
surname	VARCHAR(60)	Apellidos del usuario (hasta 60 caracteres)
ev_model	VARCHAR(40)	Modelo de vehículo eléctrico (hasta 40 caracteres)
manufacturer*	VARCHAR(45)	Fabricante (hasta 45 caracteres)
charger_type	VARCHAR(45)	Tipo de poste de carga, por potencia (hasta 45 caracteres)

*Nota: El campo *manufacturer* (fabricante) no es introducido de manera manual. La aplicación chargEV determina el fabricante del vehículo eléctrico configurado por el usuario de manera automática.

Al Usuario (minorista) se le permite configurar su nombre, apellidos, modelo de vehículo eléctrico y potencia nominal (de entre los valores estandarizados 50kW, 175kW y 350kW).

- *chargev.ev_models*:

  Esta tabla actúa como diccionario de consulta sobre los distintos modelos de vehículo eléctrico y sus correspondientes fabricantes que soporta la aplicación.

  Conforme se comercialicen futuras versiones de la misma, esta arquitectura permite la inclusión de nuevas curvas de carga y modelos soportados de manera sencilla.

  La tabla siguiente lista las columnas o campos que construyen la tabla *chargev.ev_models*, así como sus tipos de dato y usos previsibles:

Tabla 2.4. Base de datos. Campos de la tabla *chargev.ev_models*. (Fuente: Propio).

Campo	Tipo de dato	Descripción
Tabla: chargev.ev_models		
id	INT	Número natural identificador del modelo en la BB.DD.
manufacturer	VARCHAR(45)	Fabricante (hasta 45 caracteres)
model	VARCHAR(45)	Modelo de vehículo eléctrico (hasta 45 caracteres)
soc_min	FLOAT	Mínimo % de carga permitido por el vehículo

La tabla inmediatamente anterior es utilizada por la aplicación chargEV a modo de consulta para varios desplegables gráficos de selección de modelo de vehículo eléctrico ubicados en distintas ventanas y funcionalidades dentro de la interfaz gráfica. A saber, principalmente:

- Configuración del perfil en modo Usuario para la selección de modelo de vehículo eléctrico.

- Configuración de la herramienta simuladora de carga simultánea en el modo Administrador.

- *chargev.historic_data*:

  En modo Usuario, los parámetros correspondientes a toda simulación de proceso de recarga vehicular son almacenados en la presente tabla de manera inmediata tras ser finalizado.

  La tabla siguiente lista las columnas o campos que construyen la tabla *chargev.historic_data*, así como sus tipos de dato y usos previsibles:

Tabla 2.5. Base de datos. Campos de la tabla *chargev.historic_data*. (Fuente: Propio).

Campo	Tipo de dato	Descripción
Tabla: chargev.historic_data		
start	DATETIME	Fecha y hora de comienzo de la carga
stop	DATETIME	Fecha y hora de término de la carga
energy	FLOAT	Energía consumida durante la carga (en kWh)
cost	FLOAT	Coste económico de la carga (en €)

Al término de cada proceso de simulación de carga en modo Usuario, la aplicación chargEV inserta una nueva línea en la tabla *chargev.historic_data* con los campos recogidos en el listado inmediatamente anterior: fecha y hora de comienzo y término de ese proceso de carga, energía total consumida durante la misma (en kWh) y coste económico en el que se ha incurrido.

Esto permite el almacenamiento progresivo de datos de entrada para otras funcionalidades dentro de la aplicación, mientras que aporta un elevado valor añadido a la misma: conforme más usuarios usen chargEV y durante más tiempo, más información sobre procesos de carga pasados se almacena en la base de datos, lo que permite (a) desarrollar nuevas funcionalidades que precisen de un mayor volumen de información de entrada y (b) el usuario se beneficia en un mayor grado del uso de chargEV debido a una mayor sencillez en la detección de patrones fiables de consumo propios.

2.6.1.1.2. Curvas de carga

La potencia instantánea transmitida desde la estación de carga hacia las baterías del vehículo eléctrico en un momento dado depende de múltiples factores:

- La potencia máxima nominal que el poste de recarga es capaz de suministrar: los valores estandarizados y recogidos en el CSS son 50kW, 175kW y 350kW.
- El modelo de vehículo eléctrico.
- El porcentaje de carga (SoC) de las baterías en ese momento.

Típicamente, los fabricantes proporcionan las curvas de carga de sus distintos modelos de vehículo eléctrico representando la potencia instantánea como variable dependiente (eje de ordenadas) respecto al porcentaje de carga de las baterías –SoC- (eje de abscisas).
La figura inmediatamente inferior representa, a modo de ejemplo, la curva de carga incluida por el fabricante Volkswagen en la documentación y manuales de usuario para su modelo e-Golf :

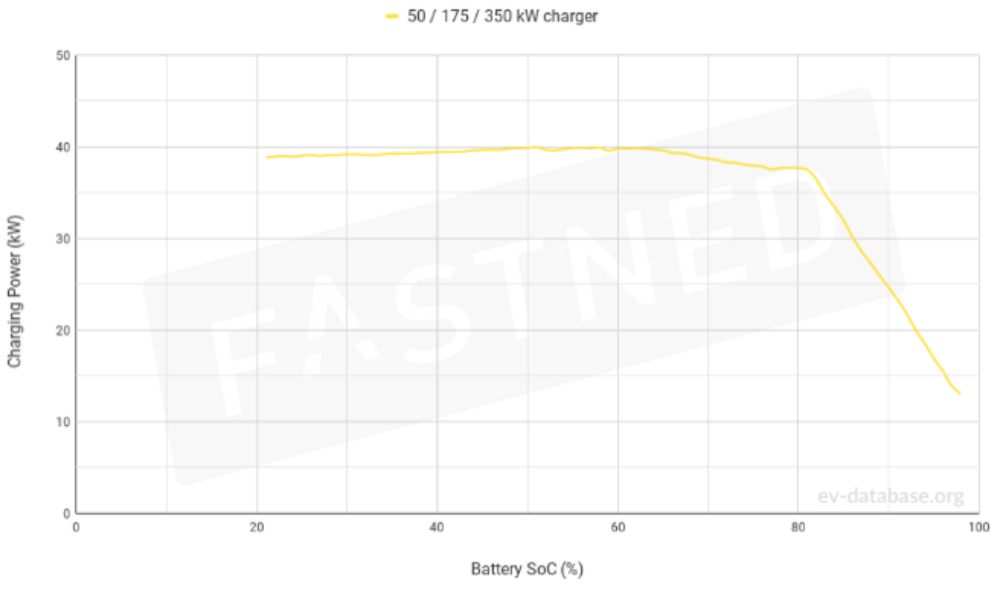


Figura 2.7. Curva de carga para e-Golf. Potencia – SoC. (Fuente: FASTNED).

En este caso, el modelo e-Golf presenta una curva de carga idéntica para los 3 (tres) valores de potencia estándar de carga puesto que la potencia máxima que su sistema de baterías puede admitir (40kW) es inferior al mínimo de los valores estandarizados para los postes (50kW), de conformidad con la Norma IEC 62196.

De este modo, se han desarrollado tablas en formato .CSV como modo de almacenamiento de los parámetros de carga para cada una de las curvas que se asocian a un modelo de vehículo eléctrico en particular. En dichas tablas se incluyen los valores de potencia (en kW) y SoC (en tanto por ciento) en períodos de 1 (un) segundo, comenzando en el valor mínimo de SoC permitido para cada vehículo en particular y terminando en un SoC del 100% (baterías completamente cargadas).

El motivo fundamental en la elección del formato .CSV es su facilidad de uso y lectura desde aplicaciones terceras. El lenguaje de programación Python permite el acceso a archivos con este formato de manera intuitiva y sencilla por medio de su librería nativa *csv*.

Para el desarrollo de la primera versión de la aplicación chargEV, las tablas contenedoras de las curvas de carga han sido incluidas como archivos locales en el propio repositorio del proyecto, bajo el path:

./Resources/ChargingCurves/

No obstante, en futuras versiones comerciales de la aplicación se recomienda ejecutar su migración a una base de datos, integrándolas como tablas de la misma.

Bajo la versión de chargEV desarrollada y descrita en el presente Proyecto, los siguientes modelos de vehículo eléctrico son soportados:

- e-Up (Volkswagen).
- e-Golf (Volkswagen).
- Ioniq (Hyundai).
- Kona (Hyundai).

La aplicación chargEV soporta los 3 (tres) valores de potencial nominal de carga de las estaciones, de conformidad con el estándar CSS: 50kW, 175kW y 350kW.

Consiguientemente, combinando los modelos de vehículo eléctrico listados arriba con los posibles valores de potencia máxima de carga de los postes que se indican, las siguientes tablas .CSV han sido creadas para satisfacer todos los escenarios de carga:

Tabla 2.6. Tablas .CSV para curvas de carga. (Fuente: Propio).

Modelo	**Potencia máxima baterías [kW]**	**Potencia cargador [kW]**	**Nombre de tabla .CSV**
eUP	40.0	50.0	eUP_50kW_175kW_350kW.csv
		175.0	
		350.0	
eGolf	40.0	50.0	eGolf_50kW_175kW_350kW.csv
		175.0	
		350.0	
Ioniq	37.5	50.0	Ioniq_50kW_175kW_350kW.csv
		175.0	
		350.0	
Kona	75.0	50.0	Kona_50kW.csv
		175.0	Kona_175kW_350kW.csv
		350.0	

Nótese cómo, para los modelos de vehículo eléctrico soportados por la versión actual de la aplicación chargEV, únicamente el Hyundai Kona requiere de múltiples curvas de carga puesto que es el único modelo cuya potencia máxima admisible (75.0kW) excede la potencia estandarizada de carga de 50.0kW, lo que implica que la potencia absorbida máxima quedaría limitada a dicho valor en aquellas situaciones en las que la estación de carga sea de potencial nominal 50.0kW.

Para más información sobre cómo se estructuran estos archivos, véase el Anexo 3.3. Curvas de carga (tablas).

2.6.1.1.3. Grafismos y logotipos

La interfaz gráfica utiliza grafismos, logotipos e iconos para una mejor representación y hacer que su uso resulte más intuitivo en diferentes funcionalidades y apartados, tanto en el modo Administrador como Usuario.

Estos grafismos se almacenan, en la presente versión de chargEV, en un directorio local que se encuentra en el mismo repositorio que las tablas .CSV descritas en el subapartado inmediatamente anterior:

./Resources/Icons/

A lo largo de los párrafos siguientes se describen los diferentes logotipos diseñados específicamente para el desarrollo de chargEV, así como su uso dentro de la interfaz gráfica.

- Grafismos de estado de la simulación:

  En total, se han diseñado 3 (tres) iconos de color negro, verde y rojo para la determinación del estado en el que se encuentra una determinada simulación de proceso de carga.

  El color negro indica inactividad, el verde se utiliza para señalar una carga en curso y el rojo, por su parte, en situaciones de alarma o error cuando alguno de los parámetros de carga especificados por el usuario de la aplicación no son correctos.

Figura 2.8. Iconos de estado de la carga. (Fuente: Propio).

- Logotipo de la Universidad de León (España):

  Puesto que el presente Proyecto se corresponde al Trabajo de Fin de Máster en el programa de Máster en Ingeniería Industrial de la Universidad de León (España), existe un apartado gráfico en el que se puede consultar información sobre el desarrollo de chargEV.

  Para más información, véase el Subapartado 2.6.1.2. Descripción de la interfaz gráfica de usuario (GUI).

Figura 2.9. Logotipo de la Universidad de León. (Fuente: Universidad de León).

- Logotipo de la aplicación chargEV:

  El logotipo de la aplicación chargEV combina su nombre con el escudo de la Escuela de Ingenierías Industrial e Informática de la Universidad de León, como se muestra en la figura.

  Este icono es utilizado en todo momento en la parte superior derecha de la interfaz gráfica, así como logo del programa en Windows.

Figura 2.10. Logotipo de la aplicación chargEV. (Fuente: Propio).

2.6.1.2. Descripción de la estructura lógica de ejecución

La estructura lógica de ejecución queda conformada por todos aquellos componentes de programa que realizan funciones específicas dentro del código.

Python es un lenguaje de programación orientado a objetos, lo que lo convierte en una buena herramienta para el desarrollo de software que precise el acompañamiento de interfaces gráficas.

De este modo, la estructura de programa puede subdividirse, atendiendo a la naturaleza de la tarea que satisfagan, en las siguientes unidades principales:

- Librerías.
- Lógica propia asociada al diseño de la interfaz gráfica de usuario.
- Lógica propia asociada a la ejecución de tareas.

Cada uno de los grupos listados consta de sus correspondientes clases que, a su vez, integran métodos específicos para cada utilidad.

2.6.1.2.1. Librerías

Los siguientes párrafos describen qué librerías han sido utilizadas, tanto nativas como desarrollos de terceros, durante la implementación y puesta en marcha de la versión de la aplicación chargEV que se describe en el presente Proyecto.
La tabla inmediatamente inferior recoge el listado de las mismas:

Tabla 2.7. Librerías utilizadas. (Fuente: Propio).

Librería	Descripción
csv	Acceso de lectura y escritura a archivos .CSV
datetime	Funciones relativas a formatos de fecha y hora
mysql.connector	Acceso cliente a bases de datos MySQL
psutil	Utilidades de procesos y sistemas
PyQt5	Librería gráfica Qt adaptada a Python (versión 5)

- *csv*:

  La librería *csv* proporciona, en Python, funciones que resultan de utilidad al acceder (tanto para lectura como escritura) a archivos .CSV.

  Para el caso que aplica al desarrollo de chargEV, la presente librería resulta de gran utilidad para la extracción de datos de las tablas .CSV que se han creado como contenedoras de los datos referidos a las curvas de carga de los diferentes modelos

de vehículos eléctricos en función de la potencia máxima de carga soportada por la estación que ha sido configurada por el Usuario.

- *datetime*:

  La librería *datetime* proporciona, en Python, funciones que resultan de utilidad al trabajar con tipos de datos relacionados con diferentes formatos de fecha y hora.

  Para el caso que aplica al desarrollo de chargEV, la presente librería resulta de gran utilidad en las herramientas siguientes:

  - Calculadora de recarga simultánea de múltiples vehículos eléctricos (modo Administrador):

    Las cargas configuradas por el administrador del sistema para cada uno de los conectores que se encuentran disponibles en su negocio son iniciadas y terminadas dentro de un rango temporal que debe ceñirse a las especificaciones y horario en el que la empresa se encuentra abierta al público.

  - Simulador de recarga individual de vehículo eléctrico minorista (modo Usuario):

    Al término de cada proceso de carga simulado por el usuario mediante esta herramienta los parámetros de la misma son guardados en la tabla de la base de datos dispuesta para tal fin junto a la fecha y hora de inicio y finalización correspondientes.

  - Visor de datos históricos (modo Usuario):

    Al usuario se le permite seleccionar el rango de fechas entre el que desea llevar a cabo la consulta de los consumos de energía y costes económicos en los que ha incurrido, ajustando la información extraída de la base de datos y su representación gráfica de manera dinámica.

- Distribución mensual de consumo energético (modo Usuario):
  Los consumos de energía son organizados y representados gráficamente de manera mensual en gráfico de barras apiladas durante los últimos cinco años.

- *mysql.connector*:

  La librería *mysql.connector* proporciona, en Python, el comportamiento de cliente SQL en accesos a bases de datos MySQL como la que se ha implementado para el almacenamiento de información considerada de interés para garantizar el correcto funcionamiento de la aplicación.

  Para más información sobre la estructura escogida para la base de datos y la información que almacena cada una de las tablas que la componen, véase el Subapartado 2.6.1.1.1.

- *psutil*:

  La librería *psutil* proporciona, en Python, funciones que resultan de utilidad para llevar a cabo acciones relacionadas con el acceso de lectura y escritura a los procesos que se encuentran corriendo en el sistema donde se lanza la aplicación, etc.

- *PyQt5*:

  La librería *PyQt* ha sido diseñada como adaptación de la genérica librería gráfica *Qt* para desarrollos en lenguaje de programación Python, permitiendo integrar todas las funcionalidades de la misma.

  La totalidad de la interfaz gráfica de usuario para chargEV, tal y como se describe en el Subapartado 2.6.1.3 del presente Proyecto ha sido desarrollada utilizando componentes, clases y métodos pertenecientes a los distintos paquetes que incorpora *PyQt5*.

  La tabla inmediatamente inferior recopila los principales paquetes utilizados para el diseño y distribución de los elementos que conforman los diferentes menús, submenús y herramientas gráficas de la aplicación chargEV.

Tabla 2.8. Paquetes de librería *PyQt5* utilizados en el desarrollo de chargEV. (Fuente: Propio).

Paquete	Elementos utilizados
Librería: PyQt5	
QtGui	Elementos principales (ventanas, etc.)
QtWidgets	Botones, etiquetas, desplegables, cajas, hojas, etc.
QtChart	Gráficos de línea, barras, etc.
QtCore	Elementos de núcleo de interfaz

2.6.1.2.2. Lógica propia asociada al diseño de la interfaz gráfica de usuario

Continuando con la línea desarrollada en el subapartado inmediatamente anterior, la interfaz gráfica de usuario de la aplicación chargEV ha sido implementada mediante elementos y componentes de la librería PyQt5.

Los siguientes párrafos se centran en describir cómo se ha llevado a efecto la integración de los mismos.

La aplicación chargEV consiste en seis clases principales, coincidentes con las diferentes ventanas de programa como sigue:

Tabla 2.9. Clases principales. (Fuente: Propio).

Clase	Elemento	Ventana de programa
Main	QMainWindow	Ventana principal
SetupMainWindow		
SetupAboutWindow	QDialog	Ventana de información del sistema
SetupEditAdminConfigWindow	QDialog	Ventana de configuración de perfil de Administrador
SetupEditUserConfigWindow	QDialog	Ventana de configuración de perfil de Usuario
SetupAutoCloseWindow	QMessageBox	Ventana de cierre de aplicación

La clase *Main* corresponde a la ventana principal de la aplicación y, por ende, contiene el hilo principal de ejecución de tareas y desarrollo gráfico. Cuando la aplicación chargEV es

iniciada, esta clase se somete a un proceso de inicialización en el que se invoca a la clase *SetupMainWindow* junto a otras funcionalidades.

La clase *SetupMainWindow* es la encargada de construir todos las herramientas, campos, grupos y, en definitiva, los elementos gráficos que conforman la ventana principal. El listado que se muestra recoge los principales pasos:

1. Título de la ventana principal.
2. Geometría mínima y por defecto de la ventana principal.
3. Icono de programa en el sistema operativo en el que se ejecuta.
4. Construcción de la barra de menús.
    4.1. Construcción de los submenús para cada menú principal.
    4.2. Enlace de submenús a métodos y funciones relacionadas.
5. Construcción de la red de elementos principal.
6. Inicialización de temporizador de control para la ventana principal.
7. Construcción gráfica de cajas principales.
    7.1. Caja de información general.
        7.1.1. Grupo Administrador.
        7.1.2. Grupo Usuario.
        7.1.3. Grupo de consumos (mensual y anual).
        7.1.4. Grupo de precio de energía.
    7.2. Caja de herramientas.
        7.2.1. Hoja de Administrador.
            7.2.1.1. Herramienta de recarga simultánea de vehículos eléctricos.
        7.2.2. Hoja de Usuario.
            7.2.2.1. Herramienta de simulación de recarga individual de vehículo eléctrico.
            7.2.2.2. Herramienta de consulta de datos históricos.
            7.2.2.3. Herramienta de distribución de consumos de energía mensuales.

Una vez construido el esqueleto gráfico de la ventana principal, el hilo principal que representa entra en un bucle infinito en el que se atiende los comandos que el usuario lleva

a efecto por medio de ratón y/o teclado. Dichas acciones, junto a otras que se procesan en segundo plano, forman parte de la estructura de ejecución de tareas descrita en el Subapartado 2.6.1.2.3 del presente Proyecto.

En caso de que el usuario de chargEV desee consultar la información del sistema y abra, haciendo uso del menú correspondiente en la barra situada en la parte superior de la ventana principal, la ventana de información del sistema, se inicializa la clase *SetupAboutWindow*.

La clase *SetupAboutWindow*, y de manera más específica su constructor, construye la ventana correspondiente llevando a cabo los pasos que se describen en el listado siguiente:

1. Configuración del estilo de la ventana de información del sistema:
    1.1. Título.
    1.2. Color de fondo.
    1.3. Fuentes.
2. Construcción de los elementos a mostrar:
    2.1. Etiquetas.
    2.2. Icono de la Universidad de León (ULe).

Del mismo modo, cuando se precisa configurar los datos de perfil de Administrador o de Usuario las ventanas correspondientes pueden ser lanzadas mediante ratón o teclado (acceso directo). Estas acciones inicializan las clases *SetupEditAdminConfigWindow* y/o *SetupEditUserConfigWindow*, según aplique.

El proceso de inicialización de la ventana de configuración de perfil del Administrador es ejecutado por el método constructor de la clase *SetupEditAdminConfigWindow* como sigue:

1. Configuración del estilo de la ventana de edición de perfil de Administrador.
    1.1. Título.
    1.2. Color de fondo.
    1.3. Geometría fija del marco de la ventana.
    1.4. Fuentes.

2. Construcción de los elementos a mostrar:
    2.1. Etiquetas.
    2.2. Campos de entrada de texto (nombre de empresa y potencia contratada en kW).
    2.3. Desplegables.
    2.4. Selectores de horas de apertura y cierre del negocio.
    2.5. Botón para guardar cambios.

Por su parte, el proceso de inicialización de la ventana de configuración del perfil de Usuario es ejecutado por el método constructor de la clase *SetupEditUserConfigWindow* como sigue:

1. Configuración del estilo de la ventana de edición de perfil de Usuario:
    1.1. Título.
    1.2. Color de fondo.
    1.3. Geometría fija del marco de la ventana.
    1.4. Fuentes.
2. Construcción de los elementos a mostrar:
    2.1. Etiquetas.
    2.2. Campos de entrada de texto (nombre y apellidos).
    2.3. Desplegables (modelo de vehículo eléctrico y potencia estandarizada de cargador).
    2.4. Botón para guardar cambios.

Por último, la ventana de confirmación de cierre de la aplicación chargEV es construida por el método de inicialización de la clase *SetupAutoCloseWindow* de diferente manera en función de la naturaleza del evento que causa el lanzamiento de dicha ventana: comando por ratón o teclado del propio usuario o tras cinco minutos de inactividad.

2.6.1.2.3. Lógica propia asociada a la ejecución de tareas

La ejecución de tareas se intercala, como es natural en programas con interfaces gráficas de usuario, con la propia lógica de construcción y configuración de elementos gráficos.

No obstante, si bien es cierto que existen múltiples ventanas con alcances y propósitos diferentes dentro de la aplicación chargEV, como norma general el siguiente orden de prioridad aplica:

1. Lógica de construcción y configuración de elementos gráficos.
2. Lógica de ejecución de tareas.

Manteniendo la jerarquía de operación que se muestra en el listado inmediatamente anterior se garantiza que la ejecución de tareas está asociada, necesariamente, a acciones que son iniciadas en última instancia por el usuario de la aplicación (de no existir elementos gráficos éste no puede iniciar tareas).
Siguiendo con la línea descriptiva planteada en el Subapartado 2.6.1.2.3 del presente Proyecto, la tabla inferior agrupa los métodos y funciones principales para la ejecución de tareas en función de la clase a la que pertenecen.

Tabla 2.10. Clases y métodos desarrollados. (Fuente: Propio).

Método	**Funcionalidad**
Clase: Main	
__init__()	Constructor
edit_admin_data()	Editar configuración de Administrador
edit_user_data()	Editar configuración de Usuario
chargev_about()	Gestión de ventana de información
idle_mouse()	Detección de inactividad
auto_close()	Gestión del auto cierre de la aplicación
auto_close_timeout()	Temporización para auto cierre
eventFilter(source, event)	Filtro de eventos del sistema operativo
sim_admin_start_calc()	Inicialización del cálculo de recarga simultánea
sim_admin_fill_data_into_array()	Gestión de datos de cálculo de los diferentes conectores
sim_start_charge()	Inicialización de la simulación de recarga individual
sim_charge_ongoing()	Gestión de la simulación de recarga individual en proceso

Método	Funcionalidad
sim_kws_to_kwh(kws)	Conversión de kWs a kWh
sim_stop_charge()	Finalización de la simulación de recarga individual
sim_update_gui()	Actualización de interfaz gráfica durante simulación
update_admin_data()	Actualización de perfil de Administrador en interfaz
update_user_data()	Actualización de perfil de Usuario en interfaz
update_monthly_data()	Actualización de consumo mensual en interfaz
update_annual_data()	Actualización de consumo anual en interfaz
historic_update_chart_data()	Actualización de gráfico en visor de datos históricos
hist_increase_counter_date()	Incremento de fecha para visor de datos históricos
hist_check_max_values()	Monitorización de valores históricos máximos
hist_monthly_data()	Gestión de consumo en distribución mensual
Clase: SetupMainWindow	
__init__(main)	Constructor
Clase: SetupAboutWindow	
__init__()	Constructor
Clase: SetupEditAdminConfigWindow	
__init__(main)	Constructor
sabe_config_data()	Guardado de configuración de perfil de Administrador
Clase: SetupEditUserConfigWindow	
__init__(main)	Constructor
sabe_config_data()	Guardado de configuración de perfil de Usuario
Clase: SetupAutoCloseWindow	
__init__(main)	Constructor
Global	
close_application()	Cierre de la aplicación chargEV

Los subapartados siguientes se centran en describir en detalle qué tareas lógicas realiza cada una de las clases y métodos correspondientes que se listan en la tabla.

2.6.1.2.3.1. Clase: Main

La clase *Main* representa el hilo principal de ejecución de la aplicación chargEV (heredando a su vez los métodos de la clase *QMainWindow* nativa de la librería *PyQt5*) y, como tal, contiene aquellos métodos considerados principales dentro del programa, como sigue:

- ***__init__()***:

  Este método es el constructor para la clase, es decir, su inicializador.

  En primer lugar, la clase *QMainWindow*, de la que hereda sus métodos. Seguidamente, todas aquellas variables globales de aplicación son inicializadas para asegurar un correcto inicio del programa. Dichas variables se refieren a elementos tanto gráficos como asociados a la ejecución de tareas lógicas.

  Se configura un temporizador de sistema que atiende la interfaz gráfica cada 200.0ms de manera recurrente. Esta acción se lleva a efecto por medio de llamadas al métido *sim_update_gui()*.

```
# Update system data
self.TimerUpdateSystem = QTimer()
self.TimerUpdateSystem.timeout.connect(self.sim_update_gui)
self.TimerUpdateSystem.start(200)
```

Figura 2.11. Temporizador para atención a interfaz gráfica. (Fuente: Código chargEV).

La aplicación chargEV precisa de acceso a la base de datos MySQL, por lo que el siguiente paso en la inicialización del programa es (a) verificar que la misma se encuentra accesible y (b) crear una conexión activa con ella. Esta conexión será utilizada para ejecutar *queries* SQL desde los diferentes menús, submenús y herramientas de usuario.

```
# CREATE CONNECTION TO MySQL DB
try:
    self.DB = mysql.connector.connect(host=MYSQL_HOST, user=MYSQL_USER, passwd=MYSQL_PASS, database=MYSQL_DB)
    self.CursorDB = self.DB.cursor()
except mysql.connector.Error as error:
    print("ERROR: No se ha podido establecer conexión con DB")
```

Figura 2.12. Verificación de conexión a base de datos. (Fuente: Código chargEV).

Se declara el objeto *gui* a partir de la clase *SetupMainWindow*, cuya finalidad es generar y localizar todos y cada uno de los elementos gráficos pertenecientes a la ventana principal.

```
# SET UP MAIN WINDOW
self.gui = SetupMainWindow(self)
```

Figura 2.13. Objeto *gui* de clase *SetupMainWindow*. (Fuente: Código chargEV).

Finalmente, dos temporizadores adicionales son configurados para detectar la inactividad del ratón en la pantalla (método *idle_mouse()*) y la cuenta atrás de 60.0s que se deja al usuario para continuar utilizando chargEV una vez los cinco minutos de inactividad han pasado (método *auto_close_timeout()*) y, por ende, la ventana de cierre automático correspondiente ha sido abierta.

```
# SET UP AUTO-CLOSE WINDOW
self.AutoCloseWindow = None
self.TimerIdleMouse = QTimer()
self.IdleMouseCounter = 0
self.TimerIdleMouse.timeout.connect(self.idle_mouse)
self.TimerIdleMouse.start(1000)
self.TimerAutoClose = QTimer()
self.AutoCloseCounter = 0
self.TimerAutoClose.timeout.connect(self.auto_close_timeout)
self.TimerAutoClose.setInterval(1000)
```

Figura 2.14. Método constructor *__init__()* de clase *Main*. (Fuente: Código chargEV).

- ***edit_admin_data()***:

  El presente método inicializa el objeto *EditAdminConfigWindow* como clase *SetupEditAdminConfigWindow*, lo que lanza la ventana de configuración de perfil de Administrador.

Para más información, véase el Subapartado 2.6.1.2.3.4.

```
# EDIT ADMIN DATA
def edit_admin_data(self):
    self.EditAdminConfigWindow = SetupEditAdminConfigWindow(self)
    self.EditAdminConfigWindow.show()
```

Figura 2.15. Método *edit_admin_data()*. (Fuente: Código chargEV).

- ***edit_user_data()***:

  El presente método inicializa el objeto *EditUserConfigWindow* como clase *SetupEditUserConfigWindow*, lo que lanza la ventana de configuración de perfil de Usuario.

  Para más información, véase el Subapartado 2.6.1.2.3.5.

```
# EDIT USER DATA
def edit_user_data(self):
    self.EditConfigWindow = SetupEditConfigWindow(self)
    self.EditConfigWindow.show()
```

Figura 2.16. Métido *edit_user_data()*. (Fuente: Código chargEV).

- ***chargev_about()***:

  El presente método inicializa el objeto *AboutWindow* como clase *SetupAboutWindow*, lo que lanza la ventana de información del sistema.

  Para más información, véase el Subapartado 2.6.1.2.3.3.

```
# TOOLING INFORMATION
def chargev_about(self):
    self.AboutWindow = SetupAboutWindow()
    self.AboutWindow.show()
```

Figura 2.17. Método *chargev_about()*. (Fuente: Código chargEV).

- ***idle_mouse()***:

El presente método es ejecutado de manera recurrente por un temporizador de sistema cada segundo, actuando como un contador que inicia la ventana de cierre automático cuando se alcanzan cinco minutos consecutivos de inactividad de ratón en pantalla mediante la llamada al método *auto_close()*.

```
# DETECT IDLE MOUSE
def idle_mouse(self):
    # Increase counter by 1s
    self.IdleMouseCounter += 1
    # Initialize auto-close window counter to timeout and reset timer if not idle mouse
    self.AutoCloseCounter = TIMEOUT_AUTO_CLOSE_WINDOW
    self.TimerAutoClose.stop()
    # Avoid timeout if a charge is ongoing
    if self.SimUserState == SIM_STATE_CHARGING:
        self.IdleMouseCounter = 0
    # Launch auto-close window after timeout of idle mouse
    if self.IdleMouseCounter >= TIMEOUT_IDLE_MOUSE:
        self.auto_close()
```

Figura 2.18. Método *idle_mouse()*. (Fuente: Código chargEV).

- ***auto_close()***:

  El presente método inicializa el objeto *AutoCloseWindow* como clase *SetupAutoCloseWindow*, lo que lanza la ventana de cierre automático, además de controlar el cierre del programa chargEV.

  Para más información, véase el Subapartado 2.6.1.2.3.6.

```
# AUTO-CLOSE WINDOW
def auto_close(self):
    self.AutoCloseWindow = SetupAutoCloseWindow(self)
```

Figura 2.19. Método *auto_closet()*. (Fuente: Código chargEV).

- ***auto_close_timeout()***:

  El presente método se ejecuta cada segundo a partir del momento en el que se detectan más de cinco minutos consecutivos de inactividad por parte del usuario de la aplicación, siendo el proveedor de datos de temporización para el método inmediatamente anterior (*auto_close_timeout()*).

```
# AUTO-CLOSE WINDOW WAITING TIME (WHEN IDLE)
def auto_close_timeout(self):
    # Decrease auto-close counter by 1s
    self.AutoCloseCounter -= 1
    # Check timeout
    if self.AutoCloseCounter < 1:
        QApplication.quit()
        return 1
```

Figura 2.20. Método *auto_close_timeout()*. (Fuente: Código chargEV).

- ***eventFilter(source, event)***:

  El presente método contiene nombre clave reservado, lo que significa que el interpretador de Python sabe que dicha función se refiere al filtrado de eventos del sistema.

  Para el caso de uso de la aplicación chargEV se utiliza para el filtrado de todos aquellos eventos de sistema salvo la inactividad de movimiento de ratón por pantalla.

```
# FILTER FOR EVENTS
def eventFilter(self, source, event):
    if event.type() == QEvent.MouseMove:
        # Restart idle time counter
        self.IdleMouseCounter = 0
    return QMainWindow.eventFilter(self, source, event)
```

Figura 2.21. Método reservado *eventFilter(source, event)*. (Fuente: Código chargEV).

- ***sim_admin_start_calc()***:

  El presente método es el encargado de llevar a cabo el proceso de cálculo de recarga simultánea de la aplicación correspondiente en modo Administrador, siendo ejecutado en el momento en el que el usuario del programa decide dar comienzo mediante el botón con la etiqueta Calcular.

  Entre las acciones necesarias para ello destacan la inicialización de los rangos para ejes de abscisas y ordenadas en los gráficos correspondientes, la agrupación de la

información de entrada aportada por el administrador del sistema para cada conector en arrays, que facilitarán su procesamiento considerablemente (véase el método *sim_admin_fill_data_into_array()*), informar por pantalla del inicio del cálculo, la obtención de consumos de potencia instantánea para cada conector durante el rango de tiempo en el que el negocio se encuentra abierto, el cálculo de potencias agregadas (consumo, disponible y en exceso) para cada instante de tiempo, aportar los datos a las consecuentes tablas y gráficos para su representación en la ventana principal y, por último, informar al administrador del sistema mediante la caja de texto del resultado, así como las recomendaciones que resulten de aplicación.

La figura inferior recoge parte del código desarrollado en el método que se describe.

```
# Get aggregated power values with their corresponding timestamps
# Time-stamps are converted to minutes for scanning purposes
initial_timestamp_m = int(QDateTime(QDate(2022, 1, 1), QTime(self.SimAdminOpeningTime.hour, self.SimAdminOpeningTime.minute)).toMSecsSinceEpoch() / (60 * 1000))
final_timestamp_m = int(QDateTime(QDate(2022, 1, 1), QTime(self.SimAdminClosingTime.hour, self.SimAdminClosingTime.minute)).toMSecsSinceEpoch() / (60 * 1000))

aggregated_entry = 0
excess = False
for time in range(initial_timestamp_m, final_timestamp_m):
    aggregated_power = 0.0

    timestamp_ms = time * (60 * 1000)

    for slot in range(SIM_ADMIN_AMOUNT_OF_SLOTS):
        # If slot is in use
        if self.SimAdminEvModels[slot] != 'Vacio':
            index = 0
            for i in self.SimAdminIndSlotsTimestamp[slot]:
                slot_timestamp_ms = QDateTime(QDate(2022, 1, 1), i).toMSecsSinceEpoch()
                if timestamp_ms == slot_timestamp_ms:
                    # Calculate aggregated power
                    aggregated_power += float(self.SimAdminIndSlotsPower[slot][index])
                    break

                index += 1

    # Calculate aggregated power
    if aggregated_power > self.SimAdminMaxPower:
        aggregated_available_power = 0.0
        aggregated_excess_power = aggregated_power - self.SimAdminMaxPower
        excess = True
    else:
        aggregated_available_power = self.SimAdminMaxPower - aggregated_power
        aggregated_excess_power = 0.0

    # Fill aggregated lists
    self.SimAdminTotalPower.append(aggregated_power)
    self.SimAdminAvailablePower.append(aggregated_available_power)
    self.SimAdminExcessPower.append(aggregated_excess_power)
    self.SimAdminAggregatedTimestamp.append(timestamp_ms)

    # Port data into aggregated chart
    point_total_power = QPointF(timestamp_ms, aggregated_power)
    self.SimAdminTotalPowerSeries.append(point_total_power)

    point_available_power = QPointF(timestamp_ms, aggregated_available_power)
    self.SimAdminAvailablePowerSeries.append(point_available_power)

    point_excess_power = QPointF(timestamp_ms, aggregated_excess_power)
    self.SimAdminExcessPowerSeries.append(point_excess_power)

    # Increase aggregated list index
    aggregated_entry += 1
```

Figura 2.22. Método *sim_admin_start_calc()* (parte). (Fuente: Código chargEV).

Autor: Marcos Castro de Prado.

- ***sim_admin_fill_data_into_array()***:

  El presente método es llamado en una de las fases iniciales del cálculo de la recarga simultánea para múltiples vehículos eléctricos, de la que se encarga el método *sim_admin_start_calc()*.

  Su finalidad principal es la de recoger los datos de entrada aportados por el usuario para la configuración del cálculo en cada uno de los diez conectores, entre los que destacan:

  - Modelo de vehículo eléctrico (en caso de estar ocupado).
  - Hora de inicio de la carga.
  - Porcentaje de carga inicial (SoC).

  Y agregarlos en arrays que faciliten su procesamiento posterior.

  La figura inferior recoge parte del código desarrollado en el método que se describe.

```
# SIMULATOR. ADMINISTRATOR. CALCULATION. GATHER DATA INTO ARRAYS
def sim_admin_fill_data_into_array(self):
    # EV models
    self.SimAdminEvModels.append(self.SimAdminEvModelSlot01.currentText())
    self.SimAdminEvModels.append(self.SimAdminEvModelSlot02.currentText())
    self.SimAdminEvModels.append(self.SimAdminEvModelSlot03.currentText())
    self.SimAdminEvModels.append(self.SimAdminEvModelSlot04.currentText())
    self.SimAdminEvModels.append(self.SimAdminEvModelSlot05.currentText())
    self.SimAdminEvModels.append(self.SimAdminEvModelSlot06.currentText())
    self.SimAdminEvModels.append(self.SimAdminEvModelSlot07.currentText())
    self.SimAdminEvModels.append(self.SimAdminEvModelSlot08.currentText())
    self.SimAdminEvModels.append(self.SimAdminEvModelSlot09.currentText())
    self.SimAdminEvModels.append(self.SimAdminEvModelSlot10.currentText())

    # Calculate amount of slots in use
    for slot in self.SimAdminEvModels:
        if slot != 'Vacío':
            self.SimAdminUsedSlots += 1

    if self.SimAdminUsedSlots <= 0:
        # Log error
        self.SimAdminInfoTextEdit.setTextColor(COLOR_YELLOW)
        self.SimAdminInfoTextEdit.append(TEXT_WARNING)
        self.SimAdminInfoTextEdit.setTextColor(COLOR_BLACK)
        self.SimAdminInfoTextEdit.append(
            "No se ha configurado ningún vehículo. Introduzca alguno.")
        # Interrupt calculation. No vehicles were set
        return -1

    # Charging curve file-paths
    for slot in range(SIM_ADMIN_AMOUNT_OF_SLOTS):
        self.SimAdminEvChargingCurveFiles.append(EV_MODEL_TO_CHARGING_CURVE_50KW.get(self.SimAdminEvModels[slot]))
```

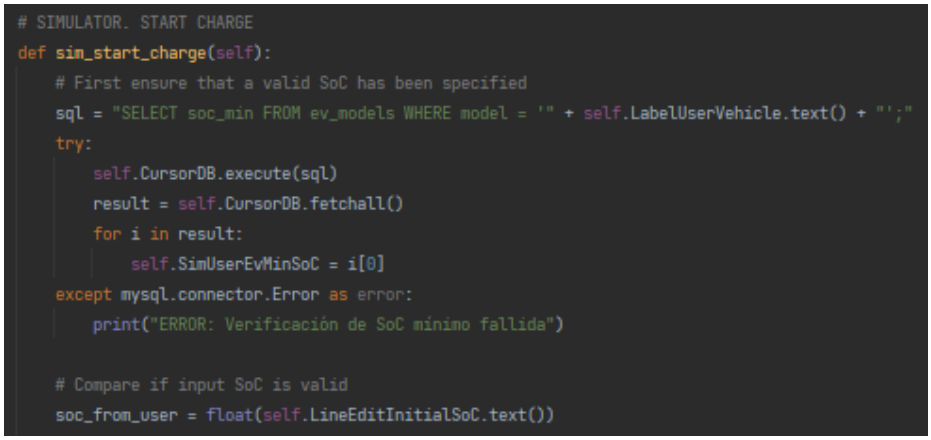
Figura 2.23. Método *sim_fill_data_into_array()* (parte). (Fuente: Código chargEV).

Autor: Marcos Castro de Prado.

- ***sim_start_charge()***:

  El presente método tiene por objeto inicializar aquellas variables clave utilizadas durante el proceso de simulación de una recarga de vehículo eléctrico individual (herramienta correspondiente al modo Usuario).

  En primer lugar, existe una serie de verificaciones que se llevan a cabo sobre el valor de SoC inicial que especifica el usuario. Para ello se ejecuta una consulta a la base de datos que permita conocer a chargEV si el valor indicado es superior al mínimo permitido para su modelo de vehículo eléctrico y, lógicamente, inferior a 100.0% (de lo contrario el vehículo ya se encontraría totalmente cargado).



```
# SIMULATOR. START CHARGE
def sim_start_charge(self):
    # First ensure that a valid SoC has been specified
    sql = "SELECT soc_min FROM ev_models WHERE model = '" + self.LabelUserVehicle.text() + "';"
    try:
        self.CursorDB.execute(sql)
        result = self.CursorDB.fetchall()
        for i in result:
            self.SimUserEvMinSoC = i[0]
    except mysql.connector.Error as error:
        print("ERROR: Verificación de SoC mínimo fallida")

    # Compare if input SoC is valid
    soc_from_user = float(self.LineEditInitialSoC.text())
```

Figura 2.24. Verificación de validez de SoC inicial. (Fuente: Código chargEV).

  En caso de no cumplir con el mínimo exigido, se informa al usuario por pantalla y se cancela la simulación.

  De lo contrario, se llevan a cabo las siguientes acciones (entre otras): inicialización de variables, acceso a la tabla .CSV contenedora de la curva de carga que aplique, configuración de la velocidad de simulación especificada por el usuario, etc.

- ***sim_charge_ongoing()***:

  El presente método es el encargado de ejecutar el proceso de simulación de recarga individual de vehículo eléctrico una vez éste ha sido iniciado por la función *sim_start_charge()*.

Para ello, se ejecuta cada segundo mientras la carga esté activa. En cada llamada a este método, se abre la tabla .CSV que contiene la curva de carga y se obtiene el valor de potencia instantáneo que se precise en función del punto de la curva en el que se encuentre para posteriormente convertirlo a las unidades correspondientes (kWh para energía y euros para coste económico).

```
# SIMULATOR. ONGOING CHARGE
def sim_charge_ongoing(self):
    # Open .CSV file
    try:
        with open(self.SimUserEvChargingCurveFile, mode='r') as csv_file:
            # Reading handler
            csv_reader = csv.DictReader(csv_file)

            # Get data
            line_count = 0
            for row in csv_reader:
                # Get data
                row_soc = float(row['soc_pcnt'])

                # First time, determine starting point based on initial SoC (only once)
                if not self.SimUserCsvStartingRow:
                    if row_soc >= float(self.LineEditInitialSoC.text()):
                        self.SimUserCurrentChargeCsvRowPointer = line_count
                        self.SimUserCurrentChargeCsvRowPointerInitial = line_count
                        self.SimUserCsvStartingRow = True

                # Normal operation (only process one line at a time)
                if line_count == self.SimUserCurrentChargeCsvRowPointer:
                    energy_kws = float(row['power_kW'])
                    actual_soc = float(row['soc_pcnt'])

                # Increase line counter
                line_count += 1
```

Figura 2.25. Acceso a tabla .CSV de curva de carga. (Fuente: Código chargEV).

De igual modo, se actualiza hoja de la herramienta de modo Usuario en la ventana principal y se monitoriza en todo momento si la carga es detenida de manera manual por el usuario mediante el botón con la etiqueta Detener o, en su defecto, automáticamente en caso de que el SoC alcance el 100.0%.

```
        # Re-calculate charging data
        self.SimUserCurrentChargeEnergy += self.sim_kws_to_kwh(energy_kws)
        self.SimUserCurrentChargePrice = self.SimUserCurrentChargeEnergy * SIM_COST_PER_KWH
        if actual_soc > 100.00:
            self.SimUserCurrentChargeSoC = 100.00
        else:
            self.SimUserCurrentChargeSoC = actual_soc
        self.SimUserCurrentChargeElapsedTime = (datetime.now() - self.SimUserCurrentChargeDateTimeStart) * self.SimUserSpeed
        elapsed_time_s = int(self.SimUserCurrentChargeElapsedTime.seconds % 60)
        elapsed_time_min = int((self.SimUserCurrentChargeElapsedTime.seconds / 60) % 60)
        elapsed_time_h = int(self.SimUserCurrentChargeElapsedTime.seconds / 3600)

        # Increase row read pointer for next cycle
        self.SimUserCurrentChargeCsvRowPointer += 1

        # Update objects
        self.LabelEnergyCount.setText("{:.2f}".format(self.SimUserCurrentChargeEnergy))
        self.LabelPriceCount.setText("{:.2f}".format(self.SimUserCurrentChargePrice))
        self.LabelSoCActual.setText("{:.2f}".format(self.SimUserCurrentChargeSoC))
        self.LabelElapsedTime.setText(
            "{:02d}:{:02d}:{:02d}".format(elapsed_time_h, elapsed_time_min, elapsed_time_s))

        # Update charts
        self.SimChartCumEnergySeries.append(
            QPointF(self.SimUserCurrentChargeSoC, self.SimUserCurrentChargeEnergy))
        self.SimChartCumPriceSeries.append(
            QPointF(self.SimUserCurrentChargeSoC, self.SimUserCurrentChargePrice))
        self.SimChartInstPowerSeries.append(QPointF(self.SimUserCurrentChargeSoC, energy_kws))

        # Terminate if the vehicle is 100.0% charged without stop button
        if actual_soc >= 100.0:
            self.sim_stop_charge()
    except all:
        print("ERROR: Fallo al abrir el archivo .CSV")
```

Figura 2.26. Control de simulación de carga en curso. (Fuente: Código chargEV).

- ***sim_kws_to_kwh(kws)*** :

  El presente método se ocupa de la conversión de valores de energía en kWs a kWh para su posterior utilización en otras herramientas o aplicaciones dentro de chargEV.

```
# SIMULATOR. CONVERT [kWs] to [kWh] (adjusted to the indicated simulator speed)
def sim_kws_to_kwh(self, kws):
    return kws / 3600.0
```

Figura 2.27. Método *sim_kws_to_kwh(kws)*. (Fuente: Código charge).

- ***sim_stop_charge()***:

El presente método tiene por objeto ejecutar todas aquellas tareas y subrutinas relacionadas con la finalización de cada proceso de simulación de carga individual en modo Usuario.

Puede ser llamado de manera manual por el propio usuario de la aplicación al presionar el botón con la etiqueta Detener o automáticamente en caso de que la simulación alcance el 100.0% de SoC.

Entre las acciones que se llevan a efecto, destacan el almacenamiento de los parámetros de la carga (energía acumulada, coste económico total y horas de inicio y finalización) y el hacer conocedor al usuario del final de la misma por medio de la caja de texto que se dispone para tal fin en la hoja correspondiente dentro de la ventana principal.

```
# SIMULATOR. STOP CHARGE
def sim_stop_charge(self):
    # Store finalized charge in DB
    sql = "INSERT INTO historic_data (start, stop, energy, cost) VALUES ('{}', current_timestamp(), {:.2f}, {:.2f});".format(
        self.SimUserCurrentChargeDateTimeStart, self.SimUserCurrentChargeEnergy, self.SimUserCurrentChargePrice)
    try:
        self.CursorDB.execute(sql)
        self.DB.commit()
    except mysql.connector.Error as error:
        print("ERROR: Datos de carga finalizada no pudieron ser guardados")

    # Log stop
    self.TextEditSimulatorLog.setTextColor(COLOR_GREEN)
    self.TextEditSimulatorLog.append(TEXT_INFO)
    self.TextEditSimulatorLog.setTextColor(COLOR_BLACK)
    self.TextEditSimulatorLog.append("Carga finalizada. SoC final: {:.2f}%.".format(self.SimUserCurrentChargeSoC))
    charged_soc = self.SimUserCurrentChargeSoC - float(self.LineEditInitialSoC.text())
    self.TextEditSimulatorLog.append("Incremento en SoC: {:.2f}%.".format(charged_soc))
    self.TextEditSimulatorLog.append("Energía consumida: {:.2f}kWh.".format(self.SimUserCurrentChargeEnergy))
    self.TextEditSimulatorLog.append("Coste total: €{:.2f}.".format(self.SimUserCurrentChargePrice))
    self.TextEditSimulatorLog.append("Duración de la carga: {} [hh:mm:ss]".format(self.LabelElapsedTime.text()))
```

Figura 2.28. Método *sim_stop_charge()* (parte). (Fuente: Código chargEV).

- ***sim_update_gui()***:

  El presente método es el encargado de la habilitación y deshabilitación de botones y elementos gráficos de la ventana principal en modo Usuario en función de si existe un proceso de simulación de recarga individual en curso o no.

```
# UPDATE SYSTEM DATA
def sim_update_gui(self):
    # Handle simulator GUI objects
    if self.SimUserState == SIM_STATE_IDLE:
        # IDLE STATE
        # Disable objects
        self.PushButtonStop.setEnabled(False)
        # Enable objects (start button only when a valid initial SoC is provided)
        self.PushButtonStart.setEnabled(True)
        self.LineEditInitialSoC.setEnabled(True)
        self.DropDownListSimSpeed.setEnabled(True)
    elif self.SimUserState == SIM_STATE_CHARGING:
        # CHARGING STATE
        # Disable objects
        self.PushButtonStart.setEnabled(False)
        self.LineEditInitialSoC.setEnabled(False)
        self.DropDownListSimSpeed.setEnabled(False)
        # Enable objects
        self.PushButtonStop.setEnabled(True)
```

Figura 2.29. Método *sim_update_gui()*. (Fuente: Código chargEV).

- ***update_admin_data()***:

  El presente método actualiza la configuración de perfil de Administrador tanto en la tabla correspondiente dentro de la base de datos MySQL como en el grupo Administrador perteneciente a la caja de información general en la parte superior de la ventana principal.

```
# UPDATE SYSTEM DATA
def sim_update_gui(self):
    # Handle simulator GUI objects
    if self.SimUserState == SIM_STATE_IDLE:
        # IDLE STATE
        # Disable objects
        self.PushButtonStop.setEnabled(False)
        # Enable objects (start button only when a valid initial SoC is provided)
        self.PushButtonStart.setEnabled(True)
        self.LineEditInitialSoC.setEnabled(True)
        self.DropDownListSimSpeed.setEnabled(True)
    elif self.SimUserState == SIM_STATE_CHARGING:
        # CHARGING STATE
        # Disable objects
        self.PushButtonStart.setEnabled(False)
        self.LineEditInitialSoC.setEnabled(False)
        self.DropDownListSimSpeed.setEnabled(False)
        # Enable objects
        self.PushButtonStop.setEnabled(True)
```

Figura 2.30. Método *update_admin_data()*. (Fuente: Código chargEV).

- ***update_user_data()***:

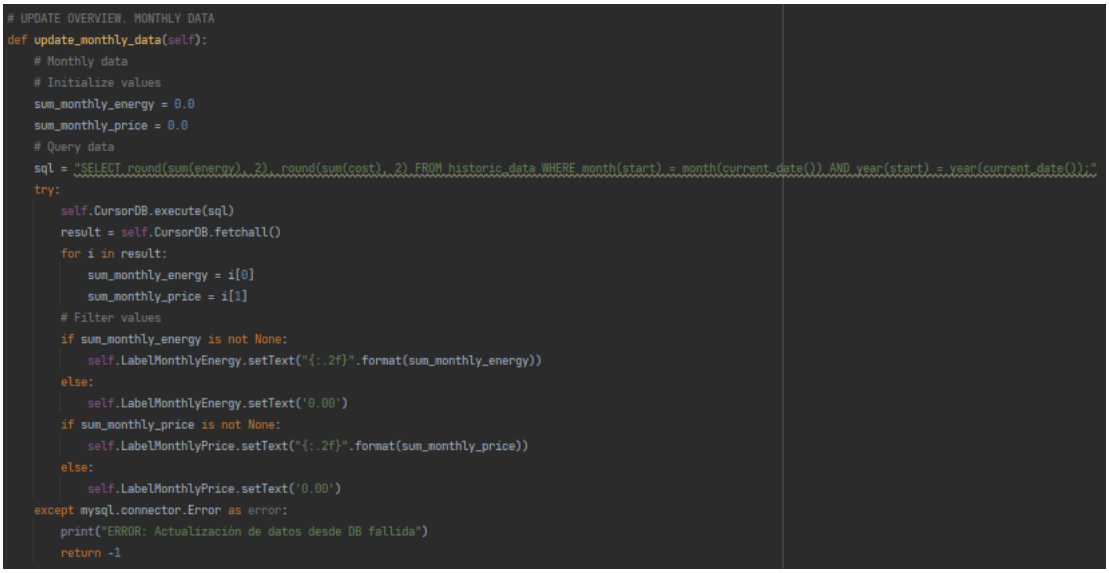
  El presente método actualiza la configuración de perfil de Usuario tanto en la tabla correspondiente dentro de la base de datos MySQL como en el grupo Usuario perteneciente a la caja de información general en la parte superior de la ventana principal.

```
# UPDATE OVERVIEW. USER DATA
def update_user_data(self):
    sql = "SELECT name, surname, ev_model, manufacturer, charger_type FROM user WHERE user_id = 1;"
    try:
        self.CursorDB.execute(sql)
        result = self.CursorDB.fetchall()
        for i in result:
            self.LabelUserName.setText(i[0])
            self.LabelUserSurname.setText(i[1])
            self.LabelUserVehicle.setText(i[2])
            self.LabelUserManufacturer.setText(i[3])
            self.LabelUserChargerType.setText("{:.2f}".format(float(str(i[4]).replace('kW', ''))))
    except mysql.connector.Error as error:
        print("ERROR: Actualización de datos desde DB fallida")
        return -1
```

Figura 2.31. Método *update_user_data()*. (Fuente: Código chargEV).

- ***update_monthly_data()***:

  El presente método actualiza los valores consumidos, tanto en energía como en coste económico, correspondientes al mes en curso en el grupo Consumos mensuales perteneciente a la caja de información general en la parte superior de la ventana principal.



```
# UPDATE OVERVIEW. MONTHLY DATA
def update_monthly_data(self):
    # Monthly data
    # Initialize values
    sum_monthly_energy = 0.0
    sum_monthly_price = 0.0
    # Query data
    sql = "SELECT round(sum(energy), 2), round(sum(cost), 2) FROM historic_data WHERE month(start) = month(current_date()) AND year(start) = year(current_date());"
    try:
        self.CursorDB.execute(sql)
        result = self.CursorDB.fetchall()
        for i in result:
            sum_monthly_energy = i[0]
            sum_monthly_price = i[1]
        # Filter values
        if sum_monthly_energy is not None:
            self.LabelMonthlyEnergy.setText("{:.2f}".format(sum_monthly_energy))
        else:
            self.LabelMonthlyEnergy.setText('0.00')
        if sum_monthly_price is not None:
            self.LabelMonthlyPrice.setText("{:.2f}".format(sum_monthly_price))
        else:
            self.LabelMonthlyPrice.setText('0.00')
    except mysql.connector.Error as error:
        print("ERROR: Actualización de datos desde DB fallida")
        return -1
```

Figura 2.32. Método *update_monthly_data()*. (Fuente: Código chargEV).

- ***update_annual_data()***:

  El presente método actualiza los valores consumidos, tanto en energía como en coste económico, correspondientes al año en curso en el grupo Consumos anuales perteneciente a la caja de información general en la parte superior de la ventana principal.

```python
# UPDATE OVERVIEW. ANNUAL DATA
def update_annual_data(self):
    # Annual data
    # Initialize values
    sum_annual_energy = 0.0
    sum_annual_price = 0.0
    # Query data
    sql = "SELECT round(sum(energy), 2), round(sum(cost), 2) FROM historic_data WHERE year(start) = year(current_date());"
    try:
        self.CursorDB.execute(sql)
        result = self.CursorDB.fetchall()
        for i in result:
            sum_annual_energy = i[0]
            sum_annual_price = i[1]
            pass
        # Filter values
        if sum_annual_energy is not None:
            self.LabelAnnualEnergy.setText("{:.2f}".format(sum_annual_energy))
            pass
        else:
            self.LabelAnnualEnergy.setText('0.00')
            pass
        if sum_annual_price is not None:
            self.LabelAnnualPrice.setText("{:.2f}".format(sum_annual_price))
            pass
        else:
            self.LabelAnnualPrice.setText('0.00')
            pass
    except mysql.connector.Error as error:
        print("ERROR: Actualización de datos desde DB fallida")
        return -1
```

Figura 2.33. Método *update_annual_data()*. (Fuente: Código charge).

- ***historic_update_chart_data()***:

  El presente método es el encargado de mostrar en las gráficas correspondientes a la herramienta visor de datos históricos de modo Usuario las curvas de consumos de energía y costes económicos en los que el usuario ha incurrido debido a procesos de recarga de su vehículo eléctrico en el pasado, ajustándolos al rango de fechas entre los que éste desea efectuar la consulta.

  De igual modo, los ejes de ordenadas también son dinámicos, permitiendo ajustarse en todo momento a diferentes perfiles de usuario.

  Para poder representar estos consumos se precisa llevar a cabo una serie de consultas a la tabla correspondiente de la base de datos MySQL.

  La figura inferior muestra parte de la lógica de control que ejecuta las acciones que se describen.

```python
# CHARTS. REDEFINE ENERGY CHART VIEW
def historic_update_chart_data(self):
    # Reset chart series
    self.HistChartEnergySeries.clear()
    self.HistChartPriceSeries.clear()

    # Reset values
    qdate_time = QDateTime()
    self.CounterYear = self.CounterMonth = self.CounterDay = 0
    energy_item = price_item = 0.0

    # Retrieve date range
    amount_of_days = self.DateEditStart.date().daysTo(self.DateEditEnd.date()) + 1

    # Set initial counter values
    self.CounterYear = self.DateEditStart.date().year()
    self.CounterMonth = self.DateEditStart.date().month()
    self.CounterDay = self.DateEditStart.date().day()

    # If start date is after end date, skip
    if amount_of_days <= 0:
        print("ERROR: Fecha final antes de fecha inicial. El tiempo es unidireccional (sad)")
        return

    # Set axis X range
    self.HistChartAxisX.setRange(QDateTime(self.DateEditStart.date(), QTime(0, 0)),
                                 QDateTime(self.DateEditEnd.date(), QTime(0, 0)))

    # Set axis Y ranges
    self.hist_check_max_values()
    self.HistChartEnergyAxisY.setRange(0.0, self.MaxEnergy)
    self.HistChartPriceAxisY.setRange(0.0, self.MaxPrice)

    # Set initial counter values again after checking max. values
    self.CounterYear = self.DateEditStart.date().year()
    self.CounterMonth = self.DateEditStart.date().month()
    self.CounterDay = self.DateEditStart.date().day()

    # Retrieve daily data
    for day in range(amount_of_days + 1):
        # Skip day 0
        if day == 0:
            continue

        # Get date string
        tmp_date = str(self.CounterYear) + "-" + str(self.CounterMonth) + "-" + str(self.CounterDay)
        qdate_time.setDate(QDate(self.CounterYear, self.CounterMonth, self.CounterDay))
```

Figura 2.34. Método *hist_update_chart_data()*. (Fuente: Código charge).

- ***hist_increase_counter_date()***:

El presente método es ejecutado mediante llamada desde el inmediatamente anterior (*hist_update_chart_data()*), y tiene por finalidad principal el incrementar la fecha de consulta en un día –de acuerdo al calendario para el año correspondiente-.

```
# HISTORICAL DATA. INCREASE COUNTER FOR TRACKING DATE AND FILL AXIS X
def hist_increase_counter_date(self):
    if self.CounterDay == 31:
        self.CounterDay = 1
        if self.CounterMonth == 12:
            self.CounterMonth = 1
            self.CounterYear += 1
        else:
            self.CounterMonth += 1
    elif self.CounterDay == 28 and self.CounterMonth == 2:
        self.CounterDay = 1
        self.CounterMonth += 1
    elif self.CounterDay == 30:
        if self.CounterMonth == 4 or self.CounterMonth == 7 or self.CounterMonth == 9 or self.CounterMonth == 11:
            self.CounterDay = 1
            self.CounterMonth += 1
        else:
            self.CounterDay += 1
    else:
        self.CounterDay += 1
```

Figura 2.35. Método *hist_increase_counter_date()*. (Fuente: Código chargEV).

- ***hist_check_max_values()***:

  El presente método es utilizado para detectar los valores máximos de energía y coste económico dentro del rango de fechas en el que el usuario ha indicado que desea realizar la consulta de datos históricos a fin de determinar de manera dinámica qué rangos de valores resultan apropiados para escalar los ejes de ordenadas de las gráficas de dicha herramienta.

```python
# HISTORICAL DATA. DETECT MAX DAILY ENERGY AND PRICE VALUES FOR ESTABLISHING RANGES OF AXIS Y
def hist_check_max_values(self):
    # Reset max. values
    self.MaxEnergy = self.MaxPrice = 0.0
    qdate_time = QDateTime()
    energy_item = price_item = 0.0

    # Retrieve date range
    amount_of_days = self.DateEditStart.date().daysTo(self.DateEditEnd.date()) + 1

    # Retrieve daily data
    for day in range(amount_of_days + 1):
        # Skip day 0
        if day == 0:
            continue

        # Get date string
        tmp_date = str(self.CounterYear) + "-" + str(self.CounterMonth) + "-" + str(self.CounterDay)
        qdate_time.setDate(QDate(self.CounterYear, self.CounterMonth, self.CounterDay))

        # Launch query
        sql = "SELECT sum(round(energy, 2)), sum(round(cost, 2)) FROM historic_data WHERE date(start) = date('" + str(
            tmp_date) + "');"
        try:
            self.CursorDB.execute(sql)
            result = self.CursorDB.fetchall()
            for i in result:
                energy_item = i[0]
                price_item = i[1]
            if energy_item is not None:
                if energy_item > self.MaxEnergy:
                    self.MaxEnergy = energy_item
            if price_item is not None:
                if price_item > self.MaxPrice:
                    self.MaxPrice = price_item
        except mysql.connector.Error as error:
            print("Lectura de datos por defecto para el gráfico fallida")

        # Increase date for daily checks
        self.hist_increase_counter_date()

    # Increase ranges to make data more readable
    self.MaxEnergy += 10.0
    self.MaxPrice += 10.0
```

Figura 2.36. Método *hist_check_max_values()*. (Fuente: Código chargEV).

- ***hist_monthly_data()***:

  El presente método es el encargado de realizar la consulta de datos de consumo energéticos agregados mensualmente durante los últimos cinco años, así como la representación gráfica de los mismos en la tabla prevista para tal fin dentro de la herramienta correspondiente del modo Usuario.

La figura inferior muestra la lógica de determinación de valores y la asignación de éstos a las variables que han sido enlazadas a la gráfica por la lógica que se describe en el Subapartado 2.6.1.3. del presente Proyecto para tres de los doce meses del año.

```
# HISTORICAL DATA. RETRIEVE MONTHLY ENERGY CONSUMPTIONS
def hist_monthly_data(self):
    for year in range(datetime.now().year - 4, datetime.now().year + 1):
        # January
        sql = "SELECT round(sum(energy), 2) FROM historic_data WHERE month(start) = 1 AND year(start) = {};".format(
            year)
        try:
            self.CursorDB.execute(sql)
            result_jan = self.CursorDB.fetchall()
            for i in result_jan:
                sum_jan = i[0]
                if sum_jan is not None:
                    self.HistChartMonthBarSetJan.append(sum_jan)
                else:
                    self.HistChartMonthBarSetJan.append(0.0)
        except mysql.connector.Error as error:
            print("ERROR: Lectura de datos de enero en año {} fallida".format(str(year)))

        # February
        sql = "SELECT round(sum(energy), 2) FROM historic_data WHERE month(start) = 2 AND year(start) = {};".format(
            year)
        try:
            self.CursorDB.execute(sql)
            result_feb = self.CursorDB.fetchall()
            for i in result_feb:
                sum_feb = i[0]
                if sum_feb is not None:
                    self.HistChartMonthBarSetFeb.append(sum_feb)
                else:
                    self.HistChartMonthBarSetFeb.append(0.0)
        except mysql.connector.Error as error:
            print("ERROR: Lectura de datos de febrero en año {} fallida".format(str(year)))

        # March
        sql = "SELECT round(sum(energy), 2) FROM historic_data WHERE month(start) = 3 AND year(start) = {};".format(
            year)
        try:
            self.CursorDB.execute(sql)
            result_mar = self.CursorDB.fetchall()
            for i in result_mar:
                sum_mar = i[0]
                if sum_mar is not None:
                    self.HistChartMonthBarSetMar.append(sum_mar)
                else:
                    self.HistChartMonthBarSetMar.append(0.0)
        except mysql.connector.Error as error:
            print("ERROR: Lectura de datos de marzo en año {} fallida".format(str(year)))
```

Figura 2.37. Método *hist_monthly_data()* (parte). (Fuente: Código chargEV).

2.6.1.2.3.2. Clase: SetupMainWindow

La clase *SetupMainWindow* es la encargada de, durante su inicialización, configurar, crear y emplazar todos y cada uno de los elementos gráficos que se aprecian en la ventana principal de la aplicación chargEV.

De este modo, únicamente contiene el método constructor (que hereda la clase *Main*):

- ***__init__(main)*:**

  En primer lugar, el presente método parametriza aspectos referidos a la ventana principal como su geometría, título, icono en sistema operativo, estilo de fuentes, etc.

```
def __init__(self, main):
    self.main = main

    # ------------------------------------------------------------------------------- #
    #
    # SET UP MAIN WINDOW
    #
    # ------------------------------------------------------------------------------- #

    main.setWindowTitle(WINDOW_TITLE)
    main.setGeometry(0, 20, 1024, 768)
    main.setMinimumHeight(900)
    main.setMinimumWidth(1150)
    main.move(100, 50)
    main.setWindowIcon(QIcon(RESOURCES_ICONS_CHARGEV))
    palette = QPalette()
    main.setPalette(palette)
    main.setStyleSheet('* {font-family: Times New; font-size: 8pt;}')
    main.setStyleSheet("QMainWindow {background-color: " + TOOL_BACKGROUND_COLOR + ";}")
```

Figura 2.38. Parametrización de ventana principal. (Fuente: Código chargEV).

Posteriormente, se diseñan los elementos gráficos como se describe en el Subapartado 2.6.1.3.1 del presente Proyecto.

2.6.1.2.3.3. Clase: SetupAboutWindow

La clase *SetupAboutWindow* representa la ventana de información del sistema, que puede ser ejecutada por el usuario mediante la acción correspondiente en la barra de menús situada en la parte superior de la ventana principal o mediante acceso directo por teclado pulsando la combinación de teclas Ctrl+I.
Así, únicamente contiene el método constructor:

- ***__init__()***:
  El presente método es el encargado de llevar a cabo la configuración y diseño de todos los elementos gráficos que se describen en profundidad en el Subapartado 2.6.1.3.2.

2.6.1.2.3.4. Clase: SetupEditAdminConfigWindow

La clase *SetupEditAdminWindow* representa la ventana de configuración de perfil de Administrador, que puede ser ejecutada por el usuario mediante la acción correspondiente en la barra de menús situada en la parte superior de la ventana principal o mediante acceso directo por teclado pulsando la combinación de teclas Ctrl+A.
Así, contiene dos métodos:

- ***__init__()***:
  El presente método es el constructor de la clase y, por ende, el encargado de llevar a cabo la configuración y diseño de todos los elementos gráficos que se describen en profundidad en el Subapartado 2.6.1.3.3.

- ***save_config_data()***:
  El presente método es ejecutado cuando el Administrador decide guardar los nuevos datos de su perfil pulsando el botón con la etiqueta Guardar que se encuadra dentro de la ventana de configuración.

```
def __init__(self, main):...

def save_config_data(self):
    # Get data
    company_name = self.EditAdminGetCompanyName.text()
    opening_time = "{:02}:{:02}".format(int(self.EditAdminGetOpeningTime.time().hour()), int(self.EditAdminGetOpeningTime.time().minute()))
    closing_time = "{:02}:{:02}".format(int(self.EditAdminGetClosingTime.time().hour()), int(self.EditAdminGetClosingTime.time().minute()))
    max_power = float(self.EditAdminGetMaxPower.text())

    # Update data on MySQL DB
    sql = "UPDATE {} SET company_name = '{}', opening_time = '{}', closing_time = '{}', max_power = {:.2f} WHERE admin_id = 1;".format(MYSQL_TABLE_ADMIN, company_name, opening_time, closing_time, max_power)
    try:
        self.main.CursorDB.execute(sql)
        self.main.DB.commit()
        # Close window
        self.close()
    except mysql.connector.Error as error:
        print("ERROR: Guardado de datos a DB fallido")

    # Update overview admin data
    self.main.update_admin_data()
```

Figura 2.39. Método *save_config_data()*. (Fuente: Código chargEV).

Como paso final, llama al método *update_admin_data()* para actualizar la nueva información de perfil de Administrador tanto en la base de datos como en la caja de información general de la ventana principal.

2.6.1.2.3.5. Clase: SetupEditUserConfigWindow

La clase *SetupEditUserConfigWindow* representa la ventana de configuración de perfil de Usuario, que puede ser ejecutada por el usuario mediante la acción correspondiente en la barra de menús situada en la parte superior de la ventana principal o mediante acceso directo por teclaro pulstando la combinación de teclas Ctrl+E.
Así, contiene dos métodos:

- ***__init__()***:
  El presente método es el constructor de la clase y, por ende, el encargado de llevar a cabo la configuración y diseño de todos los elementos gráficos que se describen en profundidad en el Subapartado 2.6.1.3.4.

- ***save_config_data()***:
  El presente método es ejecutado cuando el Usuario decide guardar los nuevos datos de su perfil pulsando el botón con la etiqueta Guardar que se encuadra dentro de la ventana de configuración.

```
def save_config_data(self):
    # Get data
    name = self.EditConfigGetName.text()
    surname = self.EditConfigGetSurname.text()
    ev_model = self.DropDownListEv.currentText()
    ev_manufacturer = ''
    charger_type = self.DropDownListCharger.currentText()

    # Get EV manufacturer from 'ev_models' table
    sql = "SELECT manufacturer FROM ev_models WHERE model = '" + ev_model + "';"
    try:
        self.main.CursorDB.execute(sql)
        result = self.main.CursorDB.fetchall()
        for i in result:
            ev_manufacturer = i[0]
        # Update DB
        sql = "UPDATE " + MYSQL_TABLE_USER + " SET name = '" + name + "', surname = '" + surname + "', ev_model = '" + ev_model + "', manufacturer = '" + ev_manufacturer + "', charger_type = '" + charger_type + "' WHERE user_id = 1;"
        try:
            self.main.CursorDB.execute(sql)
            self.main.DB.commit()
            # Close window
            self.close()
        except mysql.connector.Error as error:
            print("ERROR: Guardado de datos a DB fallido")
    except mysql.connector.Error as error:
        print("ERROR: No se pudo comprobar fabricante del modelo de vehículo seleccionado")

    # Update overview user data
    self.main.update_user_data()
```

Figura 2.40. Método *save_config_data()*. (Fuente: Código chargEV).

Como paso final, llama al método *update_admin_data()* para actualizar la nueva información de perfil de Administrador tanto en la base de datos como en la caja de información general de la ventana principal.

2.6.1.2.3.6. Clase: SetupAutoCloseWindow

La clase *SetupAutoCloseWindow* representa la ventana de cierre automático de la aplicación chargEV, que es lanzada por la propia aplicación cuando detecta inactividad por parte del usuario durante un periodo superior a cinco minutos.
De este modo, únicamente contiene el método constructor:

- ***__init__()***:
  El presente método es el constructor de la clase y, por ende, el encargado de llevar a cabo la configuración y diseño de todos los elementos gráficos que se describen en profundidad en el Subapartado 2.6.1.3.5.

2.6.1.2.3.7. Global

Dentro de la categoría Global únicamente se ha definido la siguiente función:

- ***close_application()***:

Se encarga de lanzar la ventana de cierre manual de la aplicación chargEV cuando el usuario indica, bien mediante barra de menús o mediante acceso directo por teclado, que desea salir del programa.

```
def close_application():
    message = QMessageBox()
    message.setWindowTitle('Cerrar aplicación')
    message.setIcon(QMessageBox.Question)
    message.setStyleSheet("QMessageBox {background-color: " + TOOL_BACKGROUND_COLOR + ";}")
    message.setText('¿Desea cerrar la aplicación?')
    message.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
    message.setDefaultButton(QMessageBox.No)
    # Launch window
    if message.exec() == QMessageBox.Yes:
        QApplication.quit()
        return 1
```

Figura 2.41. Función *close_application()*. (Fuente: Código chargEV).

2.6.1.3. Descripción de la interfaz gráfica de usuario (GUI)

Para el diseño de la interfaz gráfica, se ha optado por un modelo compacto en el que se contiene toda la información necesaria para la correcta operación –por parte del usuario- de la aplicación.

El presente subapartado tiene por objeto describir, con un elevado nivel de detalle, los diferentes menús, submenús y componentes de cada una de las ventanas de trabajo en las que el usuario puede operar. Por ello, se ha considera interesante dividir la descripción de la interfaz gráfica de usuario como sigue:

- Ventana principal.
- Ventana de información.
- Ventana de edición de perfil de Administrador.
- Ventana de edición de perfil de Usuario.
- Ventana de confirmación de cierre de aplicación.

2.6.1.3.1. Ventana principal

Por defecto, tras ejecutar la aplicación chargEV, la ventana que se muestra al usuario es la principal.

Esta ventana constituye la matriz desde la que se acceden al resto de ventanas y donde se ejecutan las principales acciones y funcionalidades que ofrece chargEV, tanto en modo Administrador como en modo Usuario.

La ventana principal consta de varios elementos:

- Barra de menús.
- Caja de información general.
- Caja de herramientas.

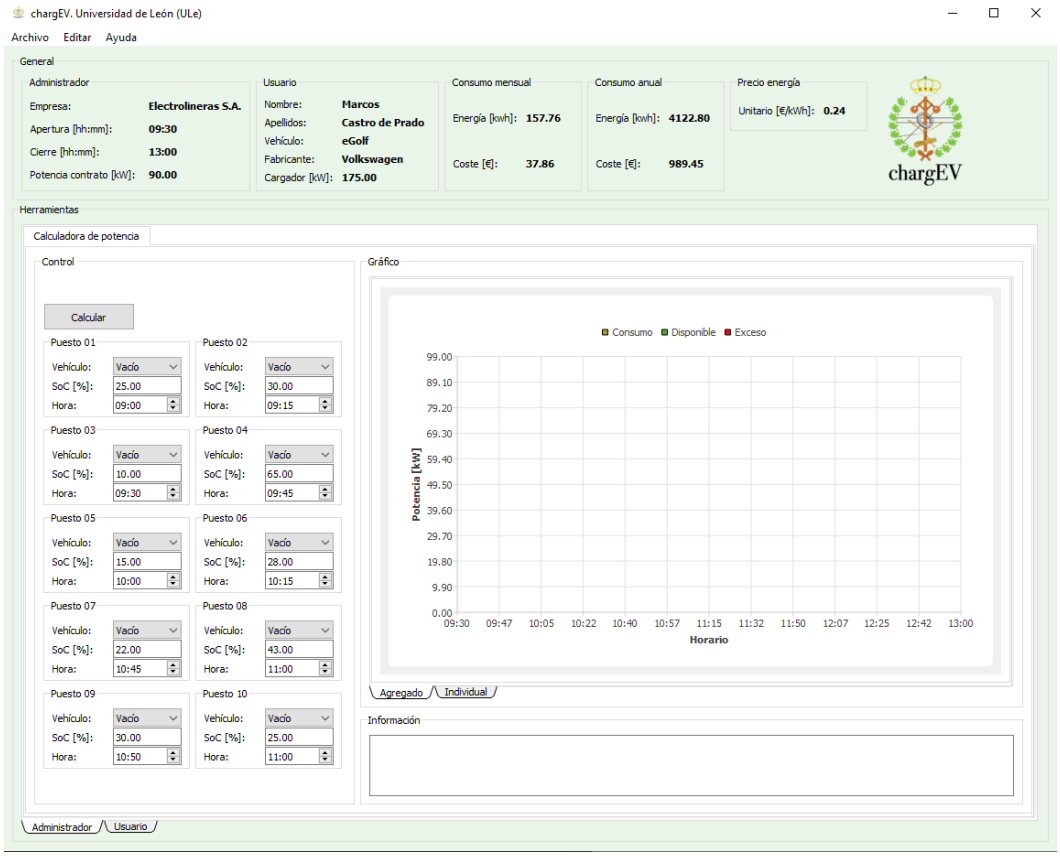


Figura 2.42. Interfaz gráfica chargEV tras ejecutar la aplicación. (Fuente: Propio).

2.6.1.3.1.1. Barra de menús

Figura 2.43. Ventana principal. Barra de menús. (Fuente: Propio).

Tal y como se muestra en la imagen inmediatamente anterior, se han desarrollado 3 (tres) menús principales:

- Archivo:

  El menú Archivo alberga funcionalidades de carácter genérico como Salir de la aplicación.

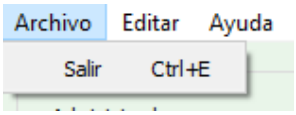


Figura 2.44. Barra de menús. Menú Archivo. (Fuente: Propio).

  Submenú(s):

  - Salir:

    También mediante acceso directo pulsando la combinación de teclas Ctrl+E, abre la ventana de confirmación de cierre de aplicación descrita en el Subapartado 2.6.1.2.5.

- Editar:

  El menú Editar alberga funcionalidades relacionadas con la configuración de los perfiles de Administrador y Usuario.

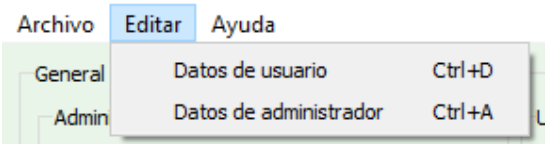


Figura 2.45. Barra de menús. Menú Editar. (Fuente: Propio).

Submenú(s):

- Datos de usuario:
  También mediante acceso directo pulsando la combinación de teclas Ctrl+D, abre la ventana de edición de perfil de Usuario descrita en el Subapartado 2.6.1.2.4.

- Datos de administrador:
  También mediante acceso directo pulsando la combinación de teclas Ctrl+A, abre la ventana de edición de perfil de Administrador descrita en el Subapartado 2.6.1.2.3.

➢ Ayuda:
El menú Ayuda alberga la funcionalidad de abrir la ventana de información del sistema, en la que se indica el autor, tutor, versión del programa y organización donde se ha desarrollado la presente versión de chargEV.

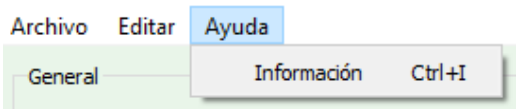


Figura 2.46. Barra de menús. Menú Ayuda. (Fuente: Propio).

Submenú(s):

- Información:

También mediante acceso directo pulsando la combinación de teclas Ctrl+I, abre la ventana de información del sistema descrita en el Subapartado 2.6.1.2.2.

2.6.1.3.1.2. Caja de información general

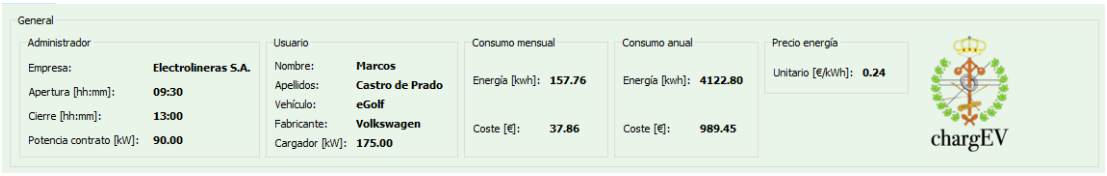


Figura 2.47. Ventana principal. Caja de información general. (Fuente: Propio).

La caja de información general de la ventana principal se muestra en la parte superior de la misma en todo momento, independientemente de si la aplicación chargEV se está utilizando como modo Administrador o Usuario.

En ella se muestra información genérica considerada de utilidad para el resto de herramientas que se integran.

La caja de información general se divide en 4 (cuatro) subgrupos. A saber:

- Administrador:

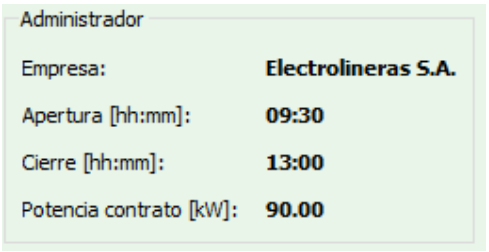


Figura 2.48. Caja de información general. Subgrupo Administrador. (Fuente: Propio).

El subgrupo Administrador muestra los campos que el administrador del sistema ha configurado previamente mediante la ventana de configuración de perfil correspondiente descrita en el Subapartado 2.6.1.2.3.

La información aquí expuesta es extraída directamente desde la tabla *chargev.administrator* correspondiente a la base de datos MySQL. Dicha información queda compuesta por: nombre de la empresa, hora de apertura y cierre del negocio y la potencia máxima contratada con la distribuidora eléctrica.

- Usuario:

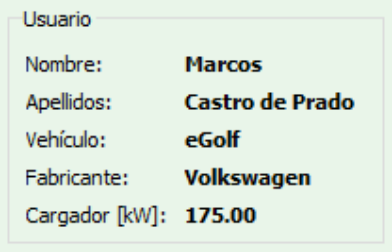


Figura 2.49. Caja de información general. Subgrupo Usuario. (Fuente: Propio).

El subgrupo Usuario muestra los campos que el usuario ha configurado previamente mediante la ventana de configuración de perfil correspondiente descrita en el Subapartado 2.6.1.2.4.

La información aquí expuesta es extraída directamente desde la tabla *chargev.user* correspondiente a la base de datos MySQL. Dicha información queda compuesta por: nombre, apellidos, modelo de vehículo eléctrico –y su fabricante- y la potencia máxima de carga estandarizada soportada por el cargador del que el Usuario dispone.

- Consumos mensual y anual:

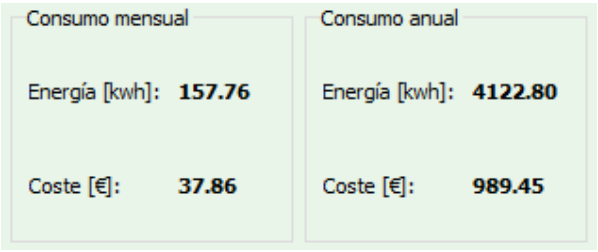


Figura 2.50. Caja de información general. Subgrupo Consumos. (Fuente: Propio).

El subgrupo Consumos mensual y anual muestra, a modo de resumen, los costes totales –tanto energéticos como económicos- en los que el Usuario ha incurrido en el mes y año en curso, respectivamente, durante los procesos de recarga eléctrica que ha realizado sobre el modelo de vehículo eléctrico y poste de recarga que ha configurado en su perfil.

Las simulaciones de los procesos de carga que aportan los datos que permiten popular los campos de consumo de energía y costes económicos son llevadas a cabo por medio de la herramienta simuladora de recarga de vehículo eléctrico en el modo Usuario.

- Precio unitario de energía:

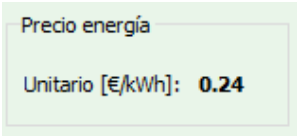


Figura 2.51. Caja de información general. Subgrupo Precio energía. (Fuente: Propio).

El subgrupo Precio energía muestra el precio unitario (en €/kWh) de la energía en cada momento.

En la versión de chargEV descrita por el presente Proyecto dicho precio es asumido como constante a 24 céntimos de euro por kWh consumido debido, fundamentalmente, a la dificultad para obtener (de manera gratuita) dicha información desde alguna API (Application Programming Interface) que aplique al territorio español.

No obstante, la EIA (U.S. Energy Information Administration) sí ofrece una API para desarrollos que utilicen datos estadísticos para los distintos Estados en EE.UU. En caso de una futura comercialización de chargEV en EE.UU., esta funcionalidad puede resultar de interés.

2.6.1.3.1.3. Caja de herramientas

La caja de herramientas se sitúa en la parte inferior de la ventana principal, ocupando el mayor espacio de entre los elementos que la componen.

El diseño gráfico de la misma está basado en un sistema de hojas (similar a cómo funciona el programa de ofimática Excel) repartidas en diversos niveles que permiten al usuario de chargEV cambiar entre sus distintas funcionalidades de manera sencilla y eficiente.

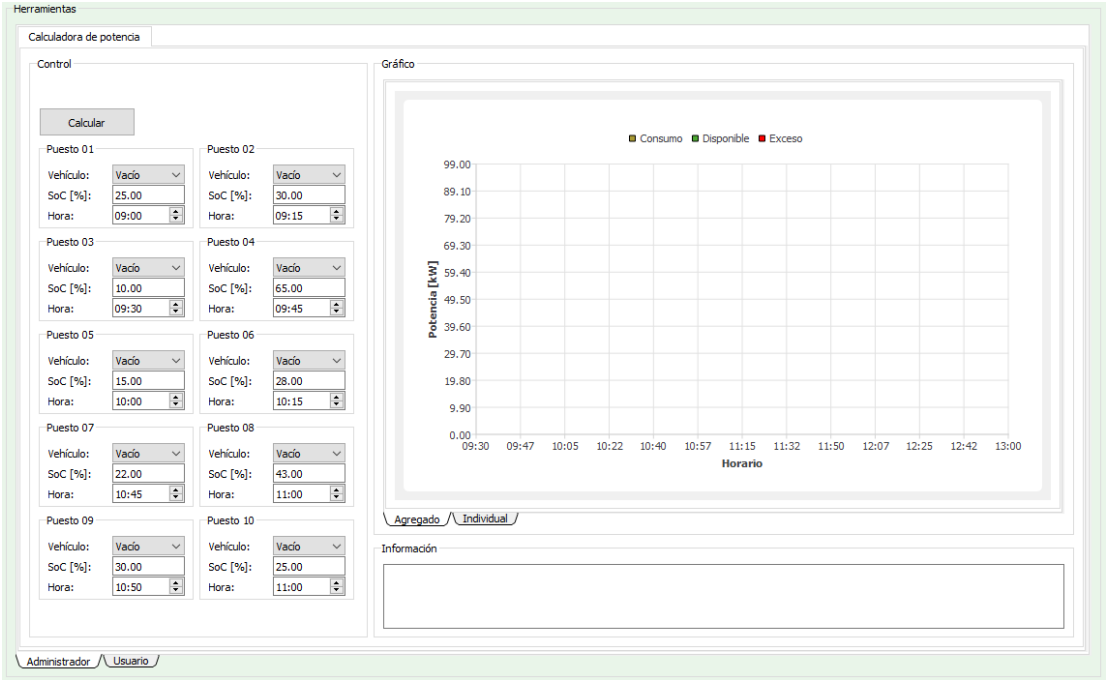


Figura 2.52. Ventana principal. Caja de herramientas. (Fuente: Propio).

Tal y como se muestra en la figura anterior, el modo Administrador (hoja seleccionada abajo a la izquierda) aparece abierto por defecto. Para cambiar al modo Usuario, únicamente basta con seleccionar la hoja contigua con la etiqueta Usuario.

La tabla siguiente muestra cómo se organizan las distintas hojas y qué herramientas contiene cada una de ellas:

Tabla 2.11. Caja de herramientas. Organización de hojas por niveles. (Fuente: Propio).

Herramienta	Descripción
Hoja: Administrador	
Calculadora de recarga simultánea	Simulador de recarga simultánea de múltiples vehículos
Hoja: Usuario	
Simulador de carga individual	Simulador de recarga individual de vehículo eléctrico
Consulta de consumos históricos	Visor gráfico de consumos y costes incurridos
Distribución de consumo mensual	Visor gráfico de consumos energéticos por mes

Cada una de las herramientas que se listan en la tabla para ambos modos (Administrador y Usuario) puede contener sus correspondientes hojas para implementar funcionalidades diferentes de una forma más amigable para el operario de la aplicación.

- Administrador:

  Por defecto, la hoja con la etiqueta Administrador aparece en la ventana principal cuando se ejecuta la aplicación chargEV.

  Herramientas:

  - Calculadora de recarga simultánea:

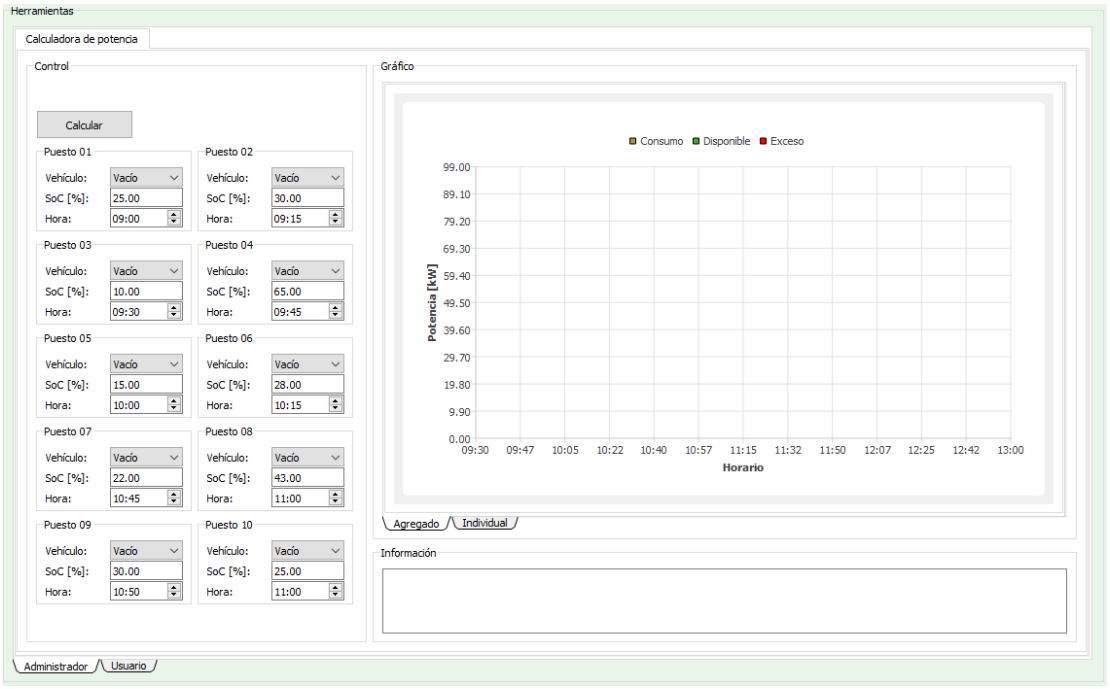


Figura 2.53. Caja de herramientas. Administrador. Calculadora de recarga simultánea. (Fuente: Propio).

La calculadora de recarga simultánea tiene por objeto determinar, tomando como entrada un horario de carga para múltiples conectores dado por el administrador del sistema, la potencia agregada consumida a lo largo del horario de apertura especificado para un negocio específico.

Por ende, su interfaz se divide en los siguientes grupos:

- Control:

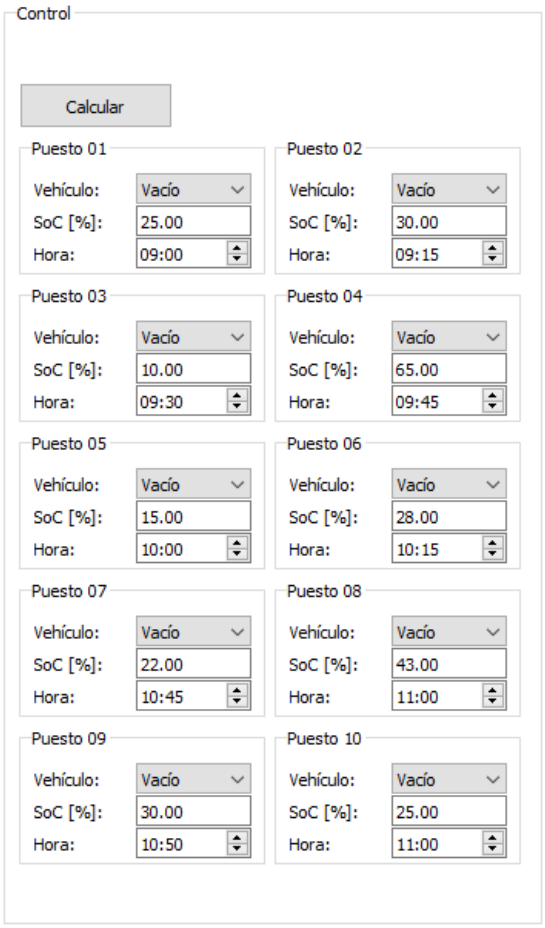


Figura 2.54. Calculadora de recarga simultánea. Grupo Control. (Fuente: Propio).

El grupo Control contiene todos aquellos elementos gráficos cuya funcionalidad es la de gestionar información de entrada para el simulador.

La calculadora emula el comportamiento de una estación múltiple de carga formada por 10 (diez) puestos o conectores en los que pueden cargarse vehículos eléctricos.

Para cada uno de los conectores, el administrador del sistema puede indicar (a) el modelo de vehículo que se va a conectar –en caso de que aplique- (b) su porcentaje de carga (SoC) inicial y (c) la hora de inicio de la recarga eléctrica.

El botón con la etiqueta Calcular inicia la simulación de la recarga múltiple utilizando como datos de entrada la información disponible en cada uno de los conectores.

- Gráfico:

  El grupo Gráfico ha sido diseñado para mostrar, de manera visual, cómo se comportan los valores agregados e individuales de potencia durante el horario en el que el negocio está abierto al público. El mismo es configurado por el administrador del sistema mediante la ventana de configuración del perfil de Administrador descrita en el Subapartado 2.6.1.3.3.

  Queda compuesto por dos hojas: la primera recoge los valores de potencia agregada consumida, disponible y en exceso (respecto al límite máximo de potencia contratada) en cada instante en función del esquema de entrada que ha sido introducido por el Administrador en el grupo Control, mientras que la segunda representa los valores de potencia individual para uno de los 10 (diez) conectores que conforman la estación completa.

  Las figuras inferiores exponen el resultado de un cálculo realizado haciendo uso de la herramienta.

  En la primera se aprecia cómo, para una potencia contratada de 90.0kW, el esquema introducido excede dicho valor entre las 09:30h y las 09:40h. Nótese cómo la gráfica de la potencia disponible en el conjunto de conectores es, lógicamente, inversa a la consumida en aquellos periodos de tiempo en los que no se excede la potencia contratada.

Figura 2.55. Calculadora de recarga simultánea. Grupo Gráfico. Agregado. (Fuente: Propio).

Figura 2.56. Calculadora de recarga simultánea. Grupo Gráfico. Individual. (Fuente: Propio).

Autor: Marcos Castro de Prado.

Por su parte, la segunda figura muestra las potencias consumidas en cada uno de los conectores en el mismo horario que los valores agregados. Nótese cómo, en este caso, únicamente se han configurado vehículos conectados a los puestos número 1, 2, 3, 6 y 7, dejando libres los puestos número 4, 5, 8, 9 y 10.

- Información:

  El grupo Información consiste en un cajetín dinámico de texto en el que se muestran mensajes que resulten de aplicación en relación al progreso de cálculo de recarga múltiple, como alarmas, errores o información útil sobre el resultado de la simulación.

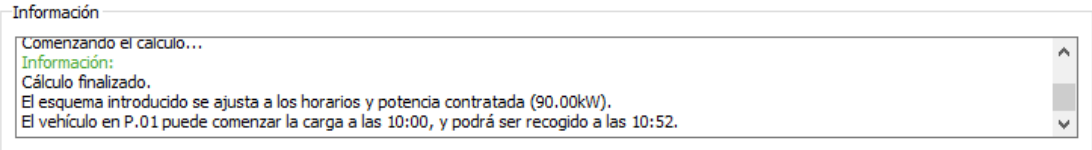


Figura 2.57. Calculadora de recarga simultánea. Grupo Información. (Fuente: Propio).

- Usuario:

  Seleccionando la hoja con la etiqueta Usuario en la parte inferior de la caja de herramientas se accede al modo Usuario, para el que se han desarrollado tres herramientas fundamentales orientadas a facilitar la detección de patrones de consumo y toma de decisiones en relación a los consumos en la recarga del vehículo eléctrico que el mismo haya configurado previamente utilizando la ventalla especialmente diseñada para ello.

  Herramientas:

  - Simulador de carga individual:

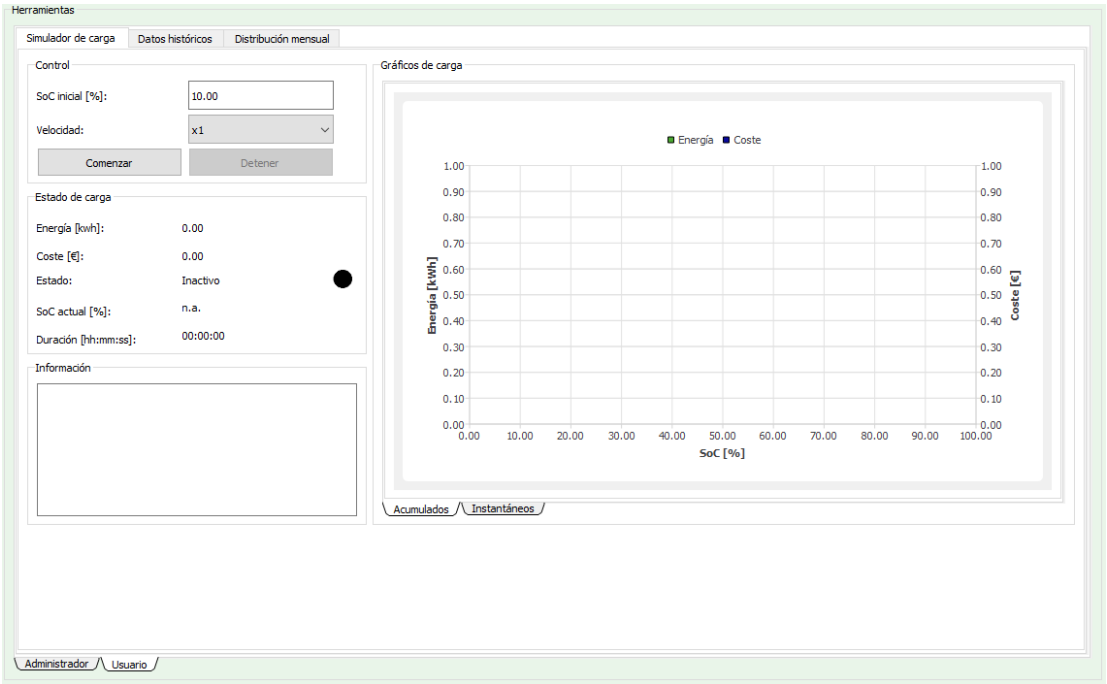


Figura 2.58. Caja de herramientas. Usuario. Simulador de carga individual. (Fuente: Propio).

Dentro del modo Usuario, la herramienta que se abre por defecto es la del simulador de recarga para vehículo eléctrico (véase, en la parte superior de la caja de herramientas, la hora con la etiqueta Simulador de carga).

Esta herramienta permite al usuario de chargEV ejecutar recargas del modelo de vehículo eleéctrico que ha configurado como suyo utilizando la ventana prevista para tal fin que se describe en el Subapartado 2.6.1.3.4.

El Simulador de carga individual se divide, gráficamente, en cuatro grupos de objetos principales. A saber:

- Control:

  El grupo Control es el contenedor de todos aquellos objetos que resultan necesarios para configurar y controlar la simulación de la recarga. Para tal fin, se dispone de los elementos que se muestran en la figura inmediatamente inferior.

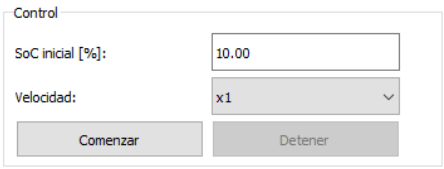


Figura 2.59. Simulador de recarga individual. Grupo Control. (Fuente: Propio).

El primer elemento consiste en una línea de entrada de texto en la que al usuario se le permite configurar el porcentaje de carga inicial de las baterías del vehículo eléctrico.

Dicho SoC resulta indispensable para la simulación por varios motivos: cada modelo de vehículo eléctrico posee un umbral mínimo de carga de sus baterías a partir del cual no es posible efectuar la recarga y, por otro lado, puesto que las curvas de carga no son lineales, es preciso conocer cuál es el SoC de partida para ejecutar un cálculo de los parámetros de carga como potencia, energía y, por ende, coste económico de manera correcta.

El segundo elemento consiste en una lista desplegable que permite al usuario seleccionar la velocidad de simulación de la carga de acuerdo a la figura siguiente:

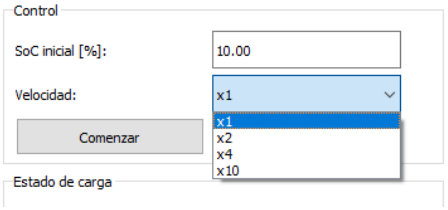


Figura 2.60. Grupo Control. Selector de velocidad de simulación. (Fuente: Propio).

La tabla inmediatamente inferior recoge una breve descripción para cada uno de las posibilidades de velocidad de simulación de recarga que se indican en la figura:

Tabla 2.12. Grupo Control. Selector de velocidad de simulación. (Fuente: Propio).

Campo	Descripción
x1	Velocidad normal (carga en tiempo real)
x2	1.0s real equivale a 2.0s simulados
x4	1.0s real equivale a 4.0s simulados
x10	1.0s real equivale a 10.0s simulados

Por ultimo, en el grupo Control se disponen dos botones con las etiquetas Comenzar y Detener arrancar y finalizar el proceso de simulación de carga, respectivamente. Nótese cómo, cuando un proceso de simulación se encuentra inactivo, el botón Detener se encuentra inhabilitado, mientras que sucede la opuesta durante el periodo de tiempo en el que una simulación se encuentra activa.

- Estado de carga:

  El grupo Estado de carga se incluye como contenedor de todos aquellos campos y variables que, de forma dinámica, son actualizados para informar al usuario de la aplicación chargEV sobre el estado de la simulación en todo momento.

  Los campos que se incluyen en el grupo Estado de carga son los siguientes: consumo de energía acumulado (en kWh), coste económico acumulado (en €), estado de la herramienta (inactivo o cargando), el SoC actual y el tiempo transcurrido desde el inicio de la simulación (en formato hh:mm:ss).

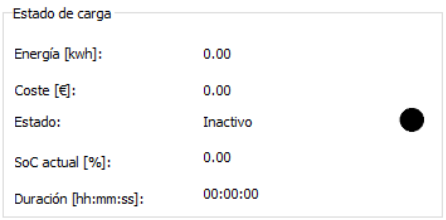


Figura 2.61. Simulador de recarga individual. Grupo Estado de carga. Carga inactiva. (Fuente: Propio).

Figura 2.62. Simulador de recarga individual. Grupo Estado de carga. Carga en curso. (Fuente: Propio).

Las dos figuras que se muestran arriba se corresponden con el grupo Estado de carga en instantes diferentes. La primera de ellas muestra cómo todos los campos numéricos son reseteados a valor nulo, en el campo con la etiqueta Estado se muestra la palabra Inactivo y el icono toma el color negro para indicar que no existe ningún proceso de simulación de recarga vehicular en la herramienta activo en ese momento. Por su parte, la segunda imagen se ha tomado a los dos minutos y treinta y tres segundos de iniciar una simulación de carga que dio comienzo con un SoC inicial del 20.0% (en el instante de la captura éste se había elevado hasta el 25.14%), con un consumo de energía acumulado hasta el momento de 1.56kWh y un coste económico acumulado de 37 céntimos de euro.

- Información:

  El grupo Información tiene por objeto hacer conocedor al usuario de la herramienta de cualquier mensaje de alerta, error o información de utilidad en relación al proceso de simulación de recarga para su vehículo eléctrico que ha configurado en el grupo Control.

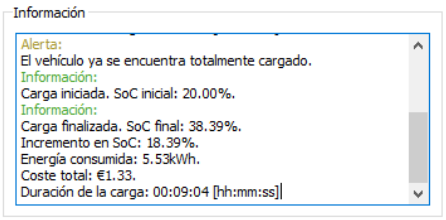


Figura 2.63. Simulador de recarga individual. Grupo Información. (Fuente: Propio).

Nótese cómo los mensajes de información aparecen precedidos por el código Información en color verde, mientras que las alertas hacen lo propio utilizando el código Alerta en color amarillo oscuro.

- Gráfico de carga:

  El grupo Gráfico de carga tiene una finalidad similar a la del grupo Estado de carga: representar el progreso de la simulación de carga que está en curso, aunque de manera más visual.

  Se divide fundamentalmente en dos hojas con las etiquetas Acumulados e Instantáneos.

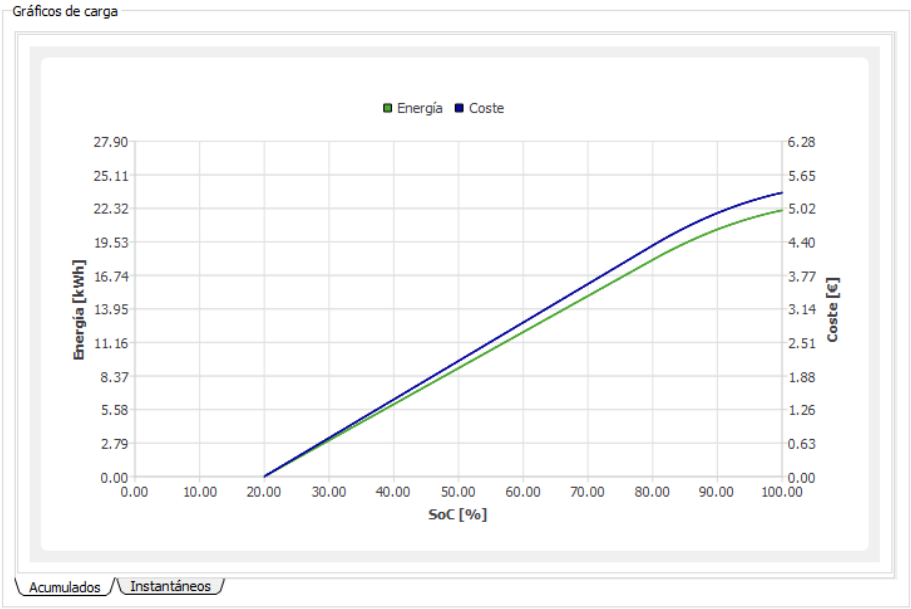


Figura 2.64. Simulador de recarga individual. Grupo Gráfico de carga. Acumulados. (Fuente: Propio).

La hoja de etiqueta Acumulados ofrece la representación, en tiempo real, de las curvas de consumo de energía (en kWh) y de coste económico (en €) acumulados durante la recarga.

Figura 2.65. Simulador de recarga individual. Grupo Gráfico de carga. Instantáneos. (Fuente: Propio).

La hoja de etiqueta Instantáneos ofrece la representación, en tiempo real, de la curva de potencia instantánea (en kW) transferida desde la estación de carga configurada por el usuario en la ventana correspondiente a las baterías de su modelo de vehículo eléctrico.

En el ejemplo mostrado en las imágenes arriba mostradas, la simulación de la recarga ha sido llevada a cabo utilizando como vehículo un e-Golf, potencia máxima de cargador de 175.0kW y utilizando un SoC inicial del 20.0% que, coincidentemente, es el mínimo soportado por dicho modelo de automóvil.

- Visor de datos históricos:

Esta herramienta se incluye bajo el modo Usuario para proporcionarle una serie de servicios que le faciliten la detección de patrones de consumo propios asociados a la recarga de su vehículo eléctrico, así como la ejecución de un mejor control de costes y consumos energéticos y económicos asociados.

Utilizando como datos de entrada los parámetros resultantes de los procesos de simulación de la recarga eléctrica realizados en el pasado y que han sido almacenados por la aplicación chargEV en la base de datos, es posible ejecutar consultas SQL a la misma para, a partir del rango de fechas que seleccione el usuario, graficar los consumos de energía y económicos a lo largo del tiempo de manera dinámica.

La figura siguiente muestra la gráfica que devuelve la herramienta tras solicitar la representación gráfica de los consumos incurridos entre el 13/04/2022 y el 30/06/2022:

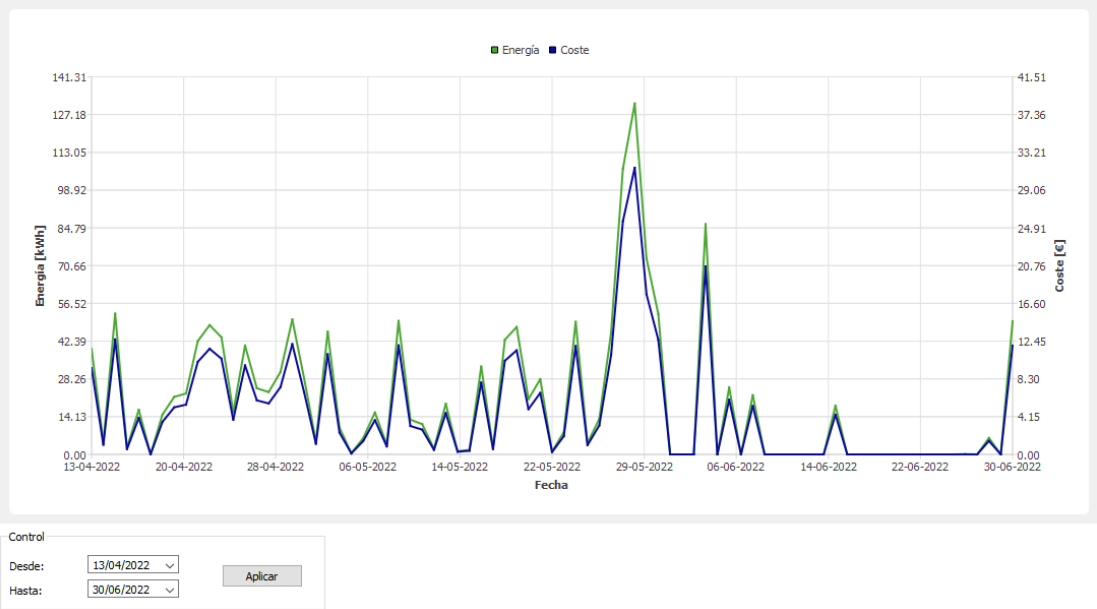


Figura 2.66. Caja de herramientas. Usuario. Visor de datos históricos. (Fuente: Propio).

Nótese cómo, de acuerdo con la leyenda de la gráfica, el consumo de energía se muestra en color verde y se relaciona al eje de ordenadas situado en la

parte izquierda de la gráfica, mientras que el coste económico en el que se ha incurrido lo hace en color azul marino y al eje de ordenadas derecho.

Manteniendo un cierto grado de uniformidad en el diseño gráfico de la interfaz para todas las herramientas y modos que la componen, el Visor de datos históricos dispone de un grupo de Control sencillo formado por dos selectores de fecha (para el inicio y el final del rango que desea consultar) y un botón con la etiqueta Aplicar para informar a la aplicación chargEV de que desea conocer el resultado.

A modo de ejemplo, la siguiente imagen recoge el diseño gráfico que se ha realizado para la selección de las fechas:

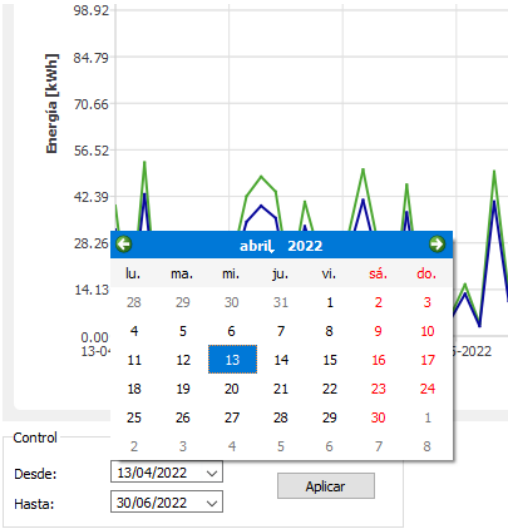


Figura 2.67. Vistor de datos históricos. Grupo Control. Selector de fecha. (Fuente: Propio).

El diseño del selector tiene por objeto fundamental resultar tan intuitivo como sea posible para el usuario de la aplicación, remarcando en color rojo aquellos días que sean fin de semana.

- Distribución de consumos de energía mensuales:

  La tercera herramienta que se ofrece dentro del modo Usuario presenta en su interfaz gráfica la distribución mensual, durante los últimos cinco años,

de los consumos de energía totales en procesos de recarga en los que el usuario minorista ha incurrido agregados por meses.

El ajuste es completamente dinámico, por lo que con la entrada de un nuevo año dicha representación se ajusta de tal modo que la nueva pila de meses se añade en primer lugar, descartando del gráfico los consumos mostrados para el año sexto.

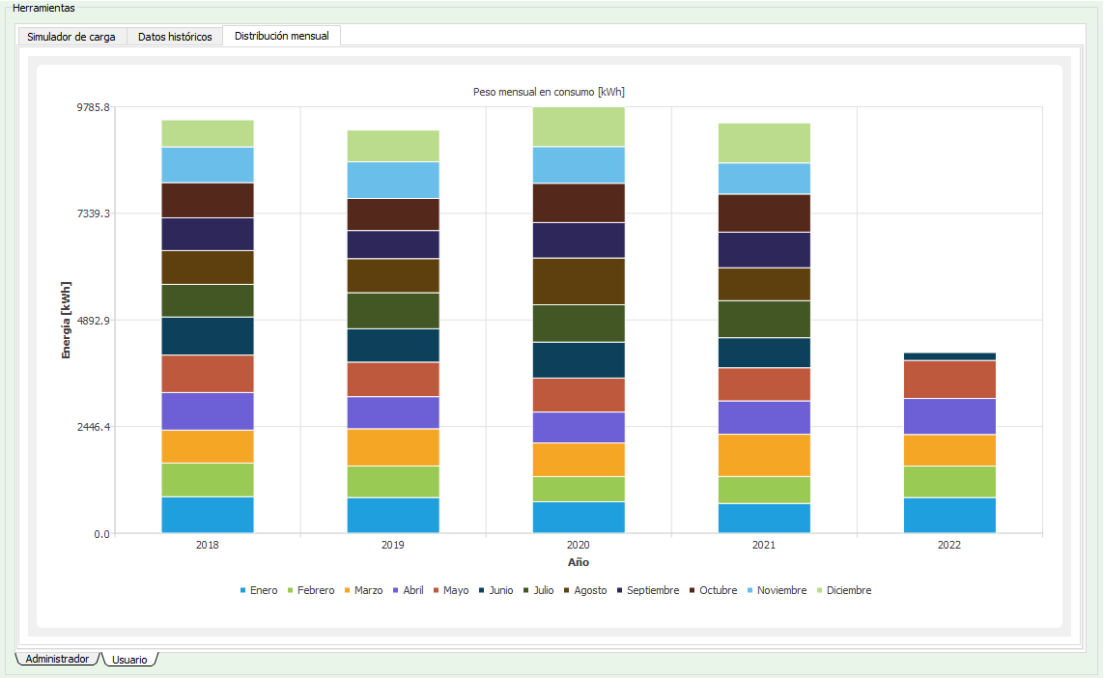


Figura 2.68. Caja de herramientas. Usuario. Distribución de consumos mensuales. (Fuente: Propio).

Se utiliza un gráfico de barras apiladas en las que cada bloque corresponde al consumo de energía total correspondiente a cada mes dentro del año. Asimismo, se ha escogido un código de colores distintivos para cada periodo mensual de manera que se facilite la detección de patrones de consumo al usuario tanto como sea posible.

En el eje de las abscisas se emplazan los últimos cinco años, mientras que el de las ordenadas queda reservado para los valores de energía (en kWh).

2.6.1.3.2. Ventana de información

La ventana de información se lanza cuando el usuario accede, en la barra de menús, al campo Ayuda/Información o presionando el acceso por teclado Ctrl+I.
Esta ventana contiene información sobre el desarrollo de la aplicación chargEV, como:

- Versión de la aplicación chargEV (con el formato vXX.XX).
- Fecha de lanzamiento de la versión en uso.
- Tutor del presente Trabajo de Fin de Máster.
- Autor del presente Trabajo de Fin de Máster.
- Logo de la organización (Universidad de León).

Figura 2.69. Ventana de información. (Fuente: Propio).

2.6.1.3.3. Ventana de edición de perfil de Administrador

La ventana de edición de perfil de Administrador permite configurar al mismo datos relativos a su negocio, de los cuales casi la totalidad resultarán necesarios para el correcto funcionamiento de la herramienta calculadora de recarga simultánea de vehículos eléctricos.

Para abrir dicha ventana basta con seleccionar el campo Editar/Datos de administrador en la barra de menús situada en la parte superior de la ventana principal (o presionar la combinación de acceso directo por teclado Ctrl+A, en su defecto).

Se lanza una ventana como sigue:

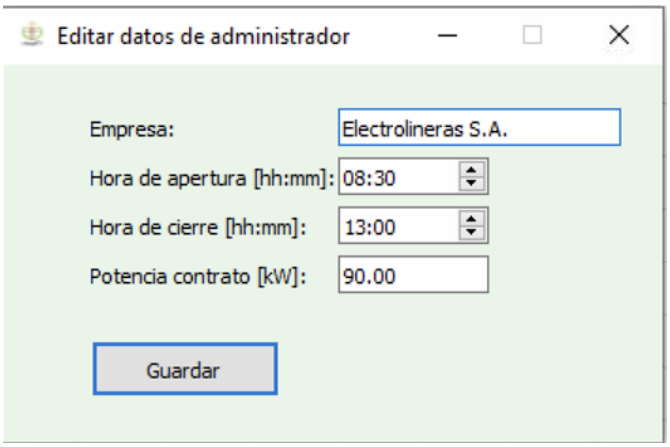


Figura 2.70. Ventana de edición de perfil de Administrador. (Fuente: Propio).

Son cuatro los campos que se habilitan para el administrador del sistema a la hora de configurar su perfil:

- Nombre de la empresa.
- Hora de apertura (selector).
- Hora de cierre (selector).
- Potencia contratada con la distribuidora eléctrica (en kW).

El botón con la etiqueta Guardar, al ser presionado, activa una subrutina que actualiza (a) los nuevos datos de Administrador en la base de datos (tabla *chargev.administrator*) y (b) la caja de información general (grupo Administrador).

2.6.1.3.4. Ventana de edición de perfil de Usuario

La ventana de edición de perfil de Usuario permite configurar al mismo datos relativos a su instalación y particularidades, de los cuales casi la totalidad resultarán necesarios para el correcto funcionamiento de las herramientas y servicios que la aplicación chargEV ofrece bajo el modo Usuario.

Para abrir dicha ventana basta con seleccionar el campo Editar/Datos de usuario en la barra de menús situada en la parte superior de la ventana principal (o presionar la combinación de acceso directo por teclado Ctrl+D, en su defecto).

Se lanza una ventana como sigue:

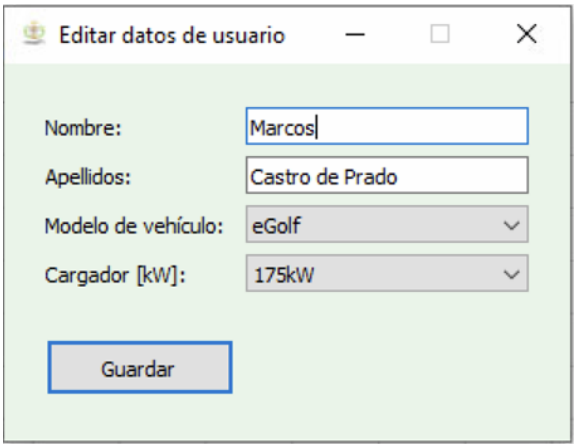


Figura 2.71. Ventana de edición de perfil de Usuario. (Fuente: Propio).

Son cuatro los campos que se habilitan para el usuario a la hora de configurar su perfil:

- Nombre.
- Apellidos.
- Modelo de vehículo eléctrico (lista desplegable).
- Potencia estandarizada de cargador (lista desplegable).

El botón con la etiqueta Guardar, al ser presionado, activa una subrutina que actualiza (a) los nuevos datos de Usuario en la base de datos (tabla *chargev.user*) y (b) la caja de información general (grupo Usuario).

2.6.1.3.5. Ventana de cierre de aplicación

La ventana de confirmación de cierre de aplicación ha sido diseñada para prevenir cierres accidentales del programa o, en su defecto, alertar al usuario de que está a punto de salir de la aplicación chargEV.
Esta ventana posee dos versiones, atendiendo a la naturaleza del evento que la generó:

- Selección, por parte del usuario, del campo Archivo/Salir en la barra de menús situada en la parte superior de la ventana principal (o presionar la combinación de acceso directo por teclado Ctrl+E, en su defecto).

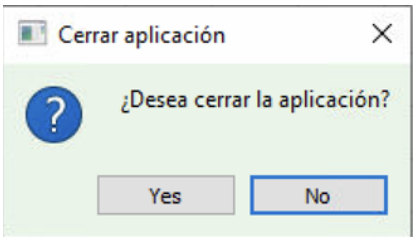


Figura 2.72. Ventana de confirmación de cierre de aplicación. Barra de menús. (Fuente: Propio).

Bajo la situación indicada (el usuario ha sido el que ha iniciado el proceso para cerrar la aplicación chargEV), se lanza la ventana de confirmación que se muestra en la figura inmediatamente superior.

- Inactividad:

  De cara a garantizar un diseño de la aplicación chargEV y un consumo de recursos de CPU limitado, chargEV posee un hijo paralelo al principal que muestrea de manera continua la actividad del ratón en pantalla para, de esta manera, detectar aquellas situaciones en las que el usuario ha dejado de usar la aplicación pero ésta sigue corriendo y, por ende, consumiendo recursos de procesamiento.

  Únicamente existe una excepción que fuerza a la aplicación a ignorar la inactividad por parte del usuario, y es cuando una simulación de carga individual de vehículo eléctrico se encuentra en curso.

En todas las demás situaciones, tras cinco minutos de inactividad chargEV lanzará la siguiente ventana a modo de alerta:

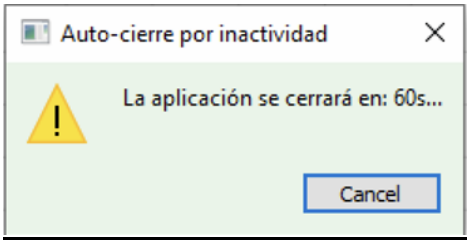


Figura 2.73. Ventana de confirmación de cierre de aplicación. Inactividad. (Fuente: Propio).

Como puede apreciarse en la imagen, al usuario se le da la posibilidad de cancelar el cierre de la aplicación durante sesenta segundos. De no recibir respuesta, chargEV termina su proceso en la CPU de manera automática.

2.7. Análisis de soluciones

Los párrafos siguientes tienen por objeto documentar aquellas soluciones consideradas en la fase inicial de desarrollo del presente Proyecto.

Debido a la naturaleza abstracta del mismo (desarrollo de software), existen múltiples alternativas que podrían llegar a resultar de utilidad. De cada a simplificar la toma de decisiones y el enfoque para el desarrollo de la aplicación, así como acotar las posibles soluciones, los siguientes requerimientos fueron establecidos:

i. La aplicación chargEV deberá ser desarrollada en un lenguaje de programación tipo dinámico, esto es, no es necesario declarar las variables de manera previa a su inicialización.
ii. El lenguaje debe resultar sencillo a la hora de integrar lógica de ejecución con interfaces gráficas entre el código y el usuario.

Durante el estudio de la viabilidad del desarrollo de chargEV, algunos de los enfoques principales fueron los siguientes:

- Desarrollo en lenguaje JavaScript.
- Desarrollo en lenguaje Python.

Como decisión última de manera previa al diseño de la aplicación chargEV se ha decidido la implementación de la misma por medio del lenguaje de programación Python debido a la facilidad de integración, testeo y *debugging* cuando es llevado a cabo mediante la IDE de PyCharm.
Así mismo, la librería gráfica PyQt resulta de una adaptación del sistemas de librerías comúnmente extendido Qt, con una amplia comunidad de desarrolladores que la utilizan a diario y, por ende, la enorme disponibilidad de soluciones y foros disponibles para la consulta de dudas o cuestiones de cualquier ámbito.

2.8. Resultados finales

Los resultados finales se muestran en el Subapartado 2.6.1, donde se aborda la descripción en detalle del presente Proyecto.

2.9. Orden de prioridad entre los documentos

En línea con las recomendaciones de la Norma UNE 157001:2014, se ratifica en el presente Proyecto el orden de prioridad entre los distintos documentos básicos que se indica en la misma, siendo (prioridad descendente):

i. Documento Básico 2. Memoria.
ii. Documento Básico 3. Anexos.

Nótese cómo los documentos básicos Planos, Presupuesto y Pliego de condiciones no resultan de aplicación para el presente trabajo, por lo que han sido excluidos de la lista inmediatamente anterior.

3. ANEXOS

ÍNDICE DE CONTENIDO. ANEXOS.

3.1. CURVAS DE CARGA (TABLAS .CSV)

ÍNDICE DE CONTENIDO. ANEXO 3. CURVAS DE CARGA.

ÍNDICE DE TABLAS. ANEXO 3. CURVAS DE CARGA.

3.1.1. eGolf_50kW_175kW_350kW

El modelo de vehículo eléctrico e-Golf presenta la misma curva de carga para los tres valores de potencia máxima estandarizada de acuerdo a CSS.

La misma queda recogida en la tabla inmediatamente inferior. En ella, la primera columna se corresponde con el instante de tiempo (en segundos), la segunda con la potencia instantánea consumida por el sistema de baterías (en kW) y la tercera con el SoC o porcentaje de carga.

Nota: De cara a facilitar la legibilidad de la misma se ha optado por rellenar una línea con puntos suspensivos en aquellos rangos de valores en los que los valores de potencia se mantenga constante.

Tabla 3.1. Tabla .CSV de carga para vehículo modelo e-Golf (Volkswagen). (Fuente: Propio).

time_s	**power_kW**	**soc_pcnt**
0	40.0	20.0
1	40.0	20.04
2	40.0	20.07
…	…	…
1622	39.98	80.01
1623	39.94	80.05
1624	39.89	80.09
1625	39.84	80.13
1626	39.8	80.16
1627	39.75	80.2
1628	39.7	80.24
1629	39.66	80.27
1630	39.61	80.31
1631	39.57	80.35
1632	39.52	80.38

time_s	power_kW	soc_pcnt
1633	39.47	80.42
1634	39.43	80.46
1635	39.38	80.5
1636	39.33	80.53
1637	39.29	80.57
1638	39.24	80.61
1639	39.2	80.64
1640	39.15	80.68
1641	39.1	80.72
1642	39.06	80.75
1643	39.01	80.79
1644	38.96	80.83
1645	38.92	80.87
1646	38.87	80.9
1647	38.83	80.94
1648	38.78	80.98
1649	38.73	81.01
1650	38.69	81.05
1651	38.64	81.09
1652	38.59	81.12
1653	38.55	81.16
1654	38.5	81.2
1655	38.46	81.24
1656	38.41	81.27
1657	38.36	81.31
1658	38.32	81.35
1659	38.27	81.38
1660	38.22	81.42
1661	38.18	81.46

time_s	power_kW	soc_pcnt
1662	38.13	81.49
1663	38.09	81.53
1664	38.04	81.57
1665	37.99	81.61
1666	37.95	81.64
1667	37.9	81.68
1668	37.85	81.72
1669	37.81	81.75
1670	37.76	81.79
1671	37.72	81.83
1672	37.67	81.86
1673	37.62	81.9
1674	37.58	81.94
1675	37.53	81.98
1676	37.48	82.01
1677	37.44	82.05
1678	37.39	82.09
1679	37.35	82.12
1680	37.3	82.16
1681	37.25	82.2
1682	37.21	82.23
1683	37.16	82.27
1684	37.11	82.31
1685	37.07	82.35
1686	37.02	82.38
1687	36.98	82.42
1688	36.93	82.46
1689	36.88	82.49
1690	36.84	82.53

time_s	power_kW	soc_pcnt
1691	36.79	82.57
1692	36.74	82.6
1693	36.7	82.64
1694	36.65	82.68
1695	36.61	82.72
1696	36.56	82.75
1697	36.51	82.79
1698	36.47	82.83
1699	36.42	82.86
1700	36.37	82.9
1701	36.33	82.94
1702	36.28	82.97
1703	36.24	83.01
1704	36.19	83.05
1705	36.14	83.09
1706	36.1	83.12
1707	36.05	83.16
1708	36	83.2
1709	35.96	83.23
1710	35.91	83.27
1711	35.87	83.31
1712	35.82	83.34
1713	35.77	83.38
1714	35.73	83.42
1715	35.68	83.46
1716	35.63	83.49
1717	35.59	83.53
1718	35.54	83.57
1719	35.5	83.6

time_s	power_kW	soc_pcnt
1720	35.45	83.64
1721	35.4	83.68
1722	35.36	83.71
1723	35.31	83.75
1724	35.26	83.79
1725	35.22	83.83
1726	35.17	83.86
1727	35.13	83.9
1728	35.08	83.94
1729	35.03	83.97
1730	34.99	84.01
1731	34.94	84.05
1732	34.89	84.08
1733	34.85	84.12
1734	34.8	84.16
1735	34.76	84.2
1736	34.71	84.23
1737	34.66	84.27
1738	34.62	84.31
1739	34.57	84.34
1740	34.52	84.38
1741	34.48	84.42
1742	34.43	84.45
1743	34.39	84.49
1744	34.34	84.53
1745	34.29	84.57
1746	34.25	84.6
1747	34.2	84.64
1748	34.15	84.68

time_s	power_kW	soc_pcnt
1749	34.11	84.71
1750	34.06	84.75
1751	34.02	84.79
1752	33.97	84.82
1753	33.92	84.86
1754	33.88	84.9
1755	33.83	84.94
1756	33.78	84.97
1757	33.74	85.01
1758	33.69	85.05
1759	33.65	85.08
1760	33.6	85.12
1761	33.55	85.16
1762	33.51	85.19
1763	33.46	85.23
1764	33.41	85.27
1765	33.37	85.31
1766	33.32	85.34
1767	33.28	85.38
1768	33.23	85.42
1769	33.18	85.45
1770	33.14	85.49
1771	33.09	85.53
1772	33.04	85.56
1773	33	85.6
1774	32.95	85.64
1775	32.91	85.68
1776	32.86	85.71
1777	32.81	85.75

time_s	power_kW	soc_pcnt
1778	32.77	85.79
1779	32.72	85.82
1780	32.67	85.86
1781	32.63	85.9
1782	32.58	85.93
1783	32.54	85.97
1784	32.49	86.01
1785	32.44	86.05
1786	32.4	86.08
1787	32.35	86.12
1788	32.3	86.16
1789	32.26	86.19
1790	32.21	86.23
1791	32.17	86.27
1792	32.12	86.3
1793	32.07	86.34
1794	32.03	86.38
1795	31.98	86.42
1796	31.93	86.45
1797	31.89	86.49
1798	31.84	86.53
1799	31.8	86.56
1800	31.75	86.6
1801	31.7	86.64
1802	31.66	86.67
1803	31.61	86.71
1804	31.56	86.75
1805	31.52	86.79
1806	31.47	86.82

time_s	power_kW	soc_pcnt
1807	31.43	86.86
1808	31.38	86.9
1809	31.33	86.93
1810	31.29	86.97
1811	31.24	87.01
1812	31.19	87.04
1813	31.15	87.08
1814	31.1	87.12
1815	31.06	87.16
1816	31.01	87.19
1817	30.96	87.23
1818	30.92	87.27
1819	30.87	87.3
1820	30.82	87.34
1821	30.78	87.38
1822	30.73	87.41
1823	30.69	87.45
1824	30.64	87.49
1825	30.59	87.53
1826	30.55	87.56
1827	30.5	87.6
1828	30.45	87.64
1829	30.41	87.67
1830	30.36	87.71
1831	30.32	87.75
1832	30.27	87.78
1833	30.22	87.82
1834	30.18	87.86
1835	30.13	87.9

time_s	power_kW	soc_pcnt
1836	30.08	87.93
1837	30.04	87.97
1838	29.99	88.01
1839	29.95	88.04
1840	29.9	88.08
1841	29.85	88.12
1842	29.81	88.15
1843	29.76	88.19
1844	29.71	88.23
1845	29.67	88.27
1846	29.62	88.3
1847	29.58	88.34
1848	29.53	88.38
1849	29.48	88.41
1850	29.44	88.45
1851	29.39	88.49
1852	29.34	88.52
1853	29.3	88.56
1854	29.25	88.6
1855	29.21	88.64
1856	29.16	88.67
1857	29.11	88.71
1858	29.07	88.75
1859	29.02	88.78
1860	28.97	88.82
1861	28.93	88.86
1862	28.88	88.89
1863	28.84	88.93
1864	28.79	88.97

time_s	power_kW	soc_pcnt
1865	28.74	89.01
1866	28.7	89.04
1867	28.65	89.08
1868	28.6	89.12
1869	28.56	89.15
1870	28.51	89.19
1871	28.47	89.23
1872	28.42	89.26
1873	28.37	89.3
1874	28.33	89.34
1875	28.28	89.38
1876	28.23	89.41
1877	28.19	89.45
1878	28.14	89.49
1879	28.1	89.52
1880	28.05	89.56
1881	28	89.6
1882	27.96	89.63
1883	27.91	89.67
1884	27.86	89.71
1885	27.82	89.75
1886	27.77	89.78
1887	27.73	89.82
1888	27.68	89.86
1889	27.63	89.89
1890	27.59	89.93
1891	27.54	89.97
1892	27.49	90
1893	27.45	90.04

time_s	power_kW	soc_pcnt
1894	27.4	90.08
1895	27.36	90.12
1896	27.31	90.15
1897	27.26	90.19
1898	27.22	90.23
1899	27.17	90.26
1900	27.12	90.3
1901	27.08	90.34
1902	27.03	90.37
1903	26.99	90.41
1904	26.94	90.45
1905	26.89	90.49
1906	26.85	90.52
1907	26.8	90.56
1908	26.75	90.6
1909	26.71	90.63
1910	26.66	90.67
1911	26.62	90.71
1912	26.57	90.74
1913	26.52	90.78
1914	26.48	90.82
1915	26.43	90.86
1916	26.38	90.89
1917	26.34	90.93
1918	26.29	90.97
1919	26.25	91
1920	26.2	91.04
1921	26.15	91.08
1922	26.11	91.11

time_s	power_kW	soc_pcnt
1923	26.06	91.15
1924	26.01	91.19
1925	25.97	91.23
1926	25.92	91.26
1927	25.88	91.3
1928	25.83	91.34
1929	25.78	91.37
1930	25.74	91.41
1931	25.69	91.45
1932	25.64	91.48
1933	25.6	91.52
1934	25.55	91.56
1935	25.51	91.6
1936	25.46	91.63
1937	25.41	91.67
1938	25.37	91.71
1939	25.32	91.74
1940	25.27	91.78
1941	25.23	91.82
1942	25.18	91.85
1943	25.14	91.89
1944	25.09	91.93
1945	25.04	91.97
1946	25	92
1947	24.95	92.04
1948	24.9	92.08
1949	24.86	92.11
1950	24.81	92.15
1951	24.77	92.19

time_s	power_kW	soc_pcnt
1952	24.72	92.22
1953	24.67	92.26
1954	24.63	92.3
1955	24.58	92.34
1956	24.53	92.37
1957	24.49	92.41
1958	24.44	92.45
1959	24.4	92.48
1960	24.35	92.52
1961	24.3	92.56
1962	24.26	92.59
1963	24.21	92.63
1964	24.16	92.67
1965	24.12	92.71
1966	24.07	92.74
1967	24.03	92.78
1968	23.98	92.82
1969	23.93	92.85
1970	23.89	92.89
1971	23.84	92.93
1972	23.79	92.96
1973	23.75	93
1974	23.7	93.04
1975	23.66	93.08
1976	23.61	93.11
1977	23.56	93.15
1978	23.52	93.19
1979	23.47	93.22
1980	23.42	93.26

time_s	power_kW	soc_pcnt
1981	23.38	93.3
1982	23.33	93.33
1983	23.29	93.37
1984	23.24	93.41
1985	23.19	93.45
1986	23.15	93.48
1987	23.1	93.52
1988	23.05	93.56
1989	23.01	93.59
1990	22.96	93.63
1991	22.92	93.67
1992	22.87	93.7
1993	22.82	93.74
1994	22.78	93.78
1995	22.73	93.82
1996	22.68	93.85
1997	22.64	93.89
1998	22.59	93.93
1999	22.55	93.96
2000	22.5	94
2001	22.45	94.04
2002	22.41	94.07
2003	22.36	94.11
2004	22.31	94.15
2005	22.27	94.19
2006	22.22	94.22
2007	22.18	94.26
2008	22.13	94.3
2009	22.08	94.33

time_s	power_kW	soc_pcnt
2010	22.04	94.37
2011	21.99	94.41
2012	21.94	94.44
2013	21.9	94.48
2014	21.85	94.52
2015	21.81	94.56
2016	21.76	94.59
2017	21.71	94.63
2018	21.67	94.67
2019	21.62	94.7
2020	21.57	94.74
2021	21.53	94.78
2022	21.48	94.81
2023	21.44	94.85
2024	21.39	94.89
2025	21.34	94.93
2026	21.3	94.96
2027	21.25	95
2028	21.2	95.04
2029	21.16	95.07
2030	21.11	95.11
2031	21.07	95.15
2032	21.02	95.18
2033	20.97	95.22
2034	20.93	95.26
2035	20.88	95.3
2036	20.83	95.33
2037	20.79	95.37
2038	20.74	95.41

time_s	power_kW	soc_pcnt
2039	20.7	95.44
2040	20.65	95.48
2041	20.6	95.52
2042	20.56	95.55
2043	20.51	95.59
2044	20.46	95.63
2045	20.42	95.67
2046	20.37	95.7
2047	20.33	95.74
2048	20.28	95.78
2049	20.23	95.81
2050	20.19	95.85
2051	20.14	95.89
2052	20.09	95.92
2053	20.05	95.96
2054	20	96
2055	19.96	96.04
2056	19.91	96.07
2057	19.86	96.11
2058	19.82	96.15
2059	19.77	96.18
2060	19.72	96.22
2061	19.68	96.26
2062	19.63	96.29
2063	19.59	96.33
2064	19.54	96.37
2065	19.49	96.41
2066	19.45	96.44
2067	19.4	96.48

time_s	power_kW	soc_pcnt
2068	19.35	96.52
2069	19.31	96.55
2070	19.26	96.59
2071	19.22	96.63
2072	19.17	96.66
2073	19.12	96.7
2074	19.08	96.74
2075	19.03	96.78
2076	18.98	96.81
2077	18.94	96.85
2078	18.89	96.89
2079	18.85	96.92
2080	18.8	96.96
2081	18.75	97
2082	18.71	97.03
2083	18.66	97.07
2084	18.61	97.11
2085	18.57	97.15
2086	18.52	97.18
2087	18.48	97.22
2088	18.43	97.26
2089	18.38	97.29
2090	18.34	97.33
2091	18.29	97.37
2092	18.24	97.4
2093	18.2	97.44
2094	18.15	97.48
2095	18.11	97.52
2096	18.06	97.55

time_s	power_kW	soc_pcnt
2097	18.01	97.59
2098	17.97	97.63
2099	17.92	97.66
2100	17.87	97.7
2101	17.83	97.74
2102	17.78	97.77
2103	17.74	97.81
2104	17.69	97.85
2105	17.64	97.89
2106	17.6	97.92
2107	17.55	97.96
2108	17.5	98
2109	17.46	98.03
2110	17.41	98.07
2111	17.37	98.11
2112	17.32	98.14
2113	17.27	98.18
2114	17.23	98.22
2115	17.18	98.26
2116	17.13	98.29
2117	17.09	98.33
2118	17.04	98.37
2119	17	98.4
2120	16.95	98.44
2121	16.9	98.48
2122	16.86	98.51
2123	16.81	98.55
2124	16.76	98.59
2125	16.72	98.63

time_s	power_kW	soc_pcnt
2126	16.67	98.66
2127	16.63	98.7
2128	16.58	98.74
2129	16.53	98.77
2130	16.49	98.81
2131	16.44	98.85
2132	16.39	98.88
2133	16.35	98.92
2134	16.3	98.96
2135	16.26	99
2136	16.21	99.03
2137	16.16	99.07
2138	16.12	99.11
2139	16.07	99.14
2140	16.02	99.18
2141	15.98	99.22
2142	15.93	99.25
2143	15.89	99.29
2144	15.84	99.33
2145	15.79	99.37
2146	15.75	99.4
2147	15.7	99.44
2148	15.65	99.48
2149	15.61	99.51
2150	15.56	99.55
2151	15.52	99.59
2152	15.47	99.62
2153	15.42	99.66
2154	15.38	99.7

time_s	power_kW	soc_pcnt
2155	15.33	99.74
2156	15.28	99.77
2157	15.24	99.81
2158	15.19	99.85
2159	15.15	99.88
2160	15.1	99.92
2161	15.05	99.96
2162	15.01	99.99
2163	14.96	100.03

3.1.2. Ioniq_50kW_175kW_350kW

El modelo de vehículo eléctrico Ioniq presenta la misma curva de carga para los tres valores de potencia máxima estandarizada de acuerdo a CSS.

La misma queda recogida en la tabla inmediatamente inferior. En ella, la primera columna se corresponde con el instante de tiempo (en segundos), la segunda con la potencia instantánea consumida por el sistema de baterías (en kW) y la tercera con el SoC o porcentaje de carga.

Nota: De cara a facilitar la legibilidad de la misma se ha optado por rellenar una línea con puntos suspensivos en aquellos rangos de valores en los que los valores de potencia se mantenga constante.

Tabla 3.2. Tabla .CSV de carga para vehículo Ioniq (Hyundai). (Fuente: Propio).

time_s	**power_kW**	**soc_pcnt**
0	37.50	10.00
1	37.50	10.03
2	37.50	10.06
…	…	…
1876	12.50	70.03
1877	12.50	70.06
1878	12.50	70.10
…	…	…
2811	12.50	99.95
2812	12.50	99.98
2813	12.50	100.02

3.1.3. Kona_50kW

El modelo de vehículo eléctrico Kona se equipa con un sistema de baterías capaz de admitir potencias de carga de hasta 75.0kW entre un SoC del 20.0% y 60.0%, por lo que su curva de carga varía en función de la potencia máxima que la estación de carga pueda suministrar.

En el caso de cargador de 50.0kW, éste valor es el que limita la máxima potencia de carga en las primeras fases.

La curva de recarga del vehículo Kona mediante cargador de 50.0kW queda recogida en la tabla inmediatamente inferior. En ella, la primera columna se corresponde con el instante de tiempo (en segundos), la segunda con la potencia instantánea consumida por el sistema de baterías (en kW) y la tercera con el SoC o porcentaje de carga.

Nota: De cara a facilitar la legibilidad de la misma se ha optado por rellenar una línea con puntos suspensivos en aquellos rangos de valores en los que la potencia se mantenga constante.

Tabla 3.3. Tabla .CSV de carga para vehículo Kona (Hyundai) con cargador de 50.0kW. (Fuente: Propio).

time_s	**power_kW**	**soc_pcnt**
0	50	10
1	50	10.024
2	50	10.048
…	…	…
2501	37.5	70.024
2502	37.5	70.048
2503	37.5	70.072
…	…	…
2918	25	80.032
2919	25	80.056
2920	25	80.08
…	…	…

time_s	power_kW	soc_pcnt
3335	24.95	90.04
3336	24.92	90.064
3337	24.89	90.088
3338	24.86	90.112
3339	24.83	90.136
3340	24.8	90.16
3341	24.77	90.184
3342	24.74	90.208
3343	24.71	90.232
3344	24.68	90.256
3345	24.65	90.28
3346	24.62	90.304
3347	24.59	90.328
3348	24.56	90.352
3349	24.53	90.376
3350	24.5	90.4
3351	24.47	90.424
3352	24.44	90.448
3353	24.41	90.472
3354	24.38	90.496
3355	24.35	90.52
3356	24.32	90.544
3357	24.29	90.568
3358	24.26	90.592
3359	24.23	90.616
3360	24.2	90.64
3361	24.17	90.664
3362	24.14	90.688
3363	24.11	90.712
3364	24.08	90.736
3365	24.05	90.76
3366	24.02	90.784

time_s	power_kW	soc_pcnt
3367	23.99	90.808
3368	23.96	90.832
3369	23.93	90.856
3370	23.9	90.88
3371	23.87	90.904
3372	23.84	90.928
3373	23.81	90.952
3374	23.78	90.976
3375	23.75	91
3376	23.72	91.024
3377	23.69	91.048
3378	23.66	91.072
3379	23.63	91.096
3380	23.6	91.12
3381	23.57	91.144
3382	23.54	91.168
3383	23.51	91.192
3384	23.48	91.216
3385	23.45	91.24
3386	23.42	91.264
3387	23.39	91.288
3388	23.36	91.312
3389	23.33	91.336
3390	23.3	91.36
3391	23.27	91.384
3392	23.24	91.408
3393	23.21	91.432
3394	23.18	91.456
3395	23.15	91.48
3396	23.12	91.504
3397	23.09	91.528
3398	23.06	91.552

time_s	power_kW	soc_pcnt
3399	23.03	91.576
3400	23	91.6
3401	22.97	91.624
3402	22.94	91.648
3403	22.91	91.672
3404	22.88	91.696
3405	22.85	91.72
3406	22.82	91.744
3407	22.79	91.768
3408	22.76	91.792
3409	22.73	91.816
3410	22.7	91.84
3411	22.67	91.864
3412	22.64	91.888
3413	22.61	91.912
3414	22.58	91.936
3415	22.55	91.96
3416	22.52	91.984
3417	22.49	92.008
3418	22.46	92.032
3419	22.43	92.056
3420	22.4	92.08
3421	22.37	92.104
3422	22.34	92.128
3423	22.31	92.152
3424	22.28	92.176
3425	22.25	92.2
3426	22.22	92.224
3427	22.19	92.248
3428	22.16	92.272
3429	22.13	92.296
3430	22.1	92.32

time_s	power_kW	soc_pcnt
3431	22.07	92.344
3432	22.04	92.368
3433	22.01	92.392
3434	21.98	92.416
3435	21.95	92.44
3436	21.92	92.464
3437	21.89	92.488
3438	21.86	92.512
3439	21.83	92.536
3440	21.8	92.56
3441	21.77	92.584
3442	21.74	92.608
3443	21.71	92.632
3444	21.68	92.656
3445	21.65	92.68
3446	21.62	92.704
3447	21.59	92.728
3448	21.56	92.752
3449	21.53	92.776
3450	21.5	92.8
3451	21.47	92.824
3452	21.44	92.848
3453	21.41	92.872
3454	21.38	92.896
3455	21.35	92.92
3456	21.32	92.944
3457	21.29	92.968
3458	21.26	92.992
3459	21.23	93.016
3460	21.2	93.04
3461	21.17	93.064
3462	21.14	93.088

time_s	power_kW	soc_pcnt
3463	21.11	93.112
3464	21.08	93.136
3465	21.05	93.16
3466	21.02	93.184
3467	20.99	93.208
3468	20.96	93.232
3469	20.93	93.256
3470	20.9	93.28
3471	20.87	93.304
3472	20.84	93.328
3473	20.81	93.352
3474	20.78	93.376
3475	20.75	93.4
3476	20.72	93.424
3477	20.69	93.448
3478	20.66	93.472
3479	20.63	93.496
3480	20.6	93.52
3481	20.57	93.544
3482	20.54	93.568
3483	20.51	93.592
3484	20.48	93.616
3485	20.45	93.64
3486	20.42	93.664
3487	20.39	93.688
3488	20.36	93.712
3489	20.33	93.736
3490	20.3	93.76
3491	20.27	93.784
3492	20.24	93.808
3493	20.21	93.832
3494	20.18	93.856

time_s	power_kW	soc_pcnt
3495	20.15	93.88
3496	20.12	93.904
3497	20.09	93.928
3498	20.06	93.952
3499	20.03	93.976
3500	20	94
3501	19.97	94.024
3502	19.94	94.048
3503	19.91	94.072
3504	19.88	94.096
3505	19.85	94.12
3506	19.82	94.144
3507	19.79	94.168
3508	19.76	94.192
3509	19.73	94.216
3510	19.7	94.24
3511	19.67	94.264
3512	19.64	94.288
3513	19.61	94.312
3514	19.58	94.336
3515	19.55	94.36
3516	19.52	94.384
3517	19.49	94.408
3518	19.46	94.432
3519	19.43	94.456
3520	19.4	94.48
3521	19.37	94.504
3522	19.34	94.528
3523	19.31	94.552
3524	19.28	94.576
3525	19.25	94.6
3526	19.22	94.624

time_s	power_kW	soc_pcnt
3527	19.19	94.648
3528	19.16	94.672
3529	19.13	94.696
3530	19.1	94.72
3531	19.07	94.744
3532	19.04	94.768
3533	19.01	94.792
3534	18.98	94.816
3535	18.95	94.84
3536	18.92	94.864
3537	18.89	94.888
3538	18.86	94.912
3539	18.83	94.936
3540	18.8	94.96
3541	18.77	94.984
3542	18.74	95.008
3543	18.71	95.032
3544	18.68	95.056
3545	18.65	95.08
3546	18.62	95.104
3547	18.59	95.128
3548	18.56	95.152
3549	18.53	95.176
3550	18.5	95.2
3551	18.47	95.224
3552	18.44	95.248
3553	18.41	95.272
3554	18.38	95.296
3555	18.35	95.32
3556	18.32	95.344
3557	18.29	95.368
3558	18.26	95.392

time_s	power_kW	soc_pcnt
3559	18.23	95.416
3560	18.2	95.44
3561	18.17	95.464
3562	18.14	95.488
3563	18.11	95.512
3564	18.08	95.536
3565	18.05	95.56
3566	18.02	95.584
3567	17.99	95.608
3568	17.96	95.632
3569	17.93	95.656
3570	17.9	95.68
3571	17.87	95.704
3572	17.84	95.728
3573	17.81	95.752
3574	17.78	95.776
3575	17.75	95.8
3576	17.72	95.824
3577	17.69	95.848
3578	17.66	95.872
3579	17.63	95.896
3580	17.6	95.92
3581	17.57	95.944
3582	17.54	95.968
3583	17.51	95.992
3584	17.48	96.016
3585	17.45	96.04
3586	17.42	96.064
3587	17.39	96.088
3588	17.36	96.112
3589	17.33	96.136
3590	17.3	96.16

time_s	power_kW	soc_pcnt
3591	17.27	96.184
3592	17.24	96.208
3593	17.21	96.232
3594	17.18	96.256
3595	17.15	96.28
3596	17.12	96.304
3597	17.09	96.328
3598	17.06	96.352
3599	17.03	96.376
3600	17	96.4
3601	16.97	96.424
3602	16.94	96.448
3603	16.91	96.472
3604	16.88	96.496
3605	16.85	96.52
3606	16.82	96.544
3607	16.79	96.568
3608	16.76	96.592
3609	16.73	96.616
3610	16.7	96.64
3611	16.67	96.664
3612	16.64	96.688
3613	16.61	96.712
3614	16.58	96.736
3615	16.55	96.76
3616	16.52	96.784
3617	16.49	96.808
3618	16.46	96.832
3619	16.43	96.856
3620	16.4	96.88
3621	16.37	96.904
3622	16.34	96.928

time_s	power_kW	soc_pcnt
3623	16.31	96.952
3624	16.28	96.976
3625	16.25	97
3626	16.22	97.024
3627	16.19	97.048
3628	16.16	97.072
3629	16.13	97.096
3630	16.1	97.12
3631	16.07	97.144
3632	16.04	97.168
3633	16.01	97.192
3634	15.98	97.216
3635	15.95	97.24
3636	15.92	97.264
3637	15.89	97.288
3638	15.86	97.312
3639	15.83	97.336
3640	15.8	97.36
3641	15.77	97.384
3642	15.74	97.408
3643	15.71	97.432
3644	15.68	97.456
3645	15.65	97.48
3646	15.62	97.504
3647	15.59	97.528
3648	15.56	97.552
3649	15.53	97.576
3650	15.5	97.6
3651	15.47	97.624
3652	15.44	97.648
3653	15.41	97.672
3654	15.38	97.696

time_s	power_kW	soc_pcnt
3655	15.35	97.72
3656	15.32	97.744
3657	15.29	97.768
3658	15.26	97.792
3659	15.23	97.816
3660	15.2	97.84
3661	15.17	97.864
3662	15.14	97.888
3663	15.11	97.912
3664	15.08	97.936
3665	15.05	97.96
3666	15.02	97.984
3667	14.99	98.008
3668	14.96	98.032
3669	14.93	98.056
3670	14.9	98.08
3671	14.87	98.104
3672	14.84	98.128
3673	14.81	98.152
3674	14.78	98.176
3675	14.75	98.2
3676	14.72	98.224
3677	14.69	98.248
3678	14.66	98.272
3679	14.63	98.296
3680	14.6	98.32
3681	14.57	98.344
3682	14.54	98.368
3683	14.51	98.392
3684	14.48	98.416
3685	14.45	98.44
3686	14.42	98.464

time_s	power_kW	soc_pcnt
3687	14.39	98.488
3688	14.36	98.512
3689	14.33	98.536
3690	14.3	98.56
3691	14.27	98.584
3692	14.24	98.608
3693	14.21	98.632
3694	14.18	98.656
3695	14.15	98.68
3696	14.12	98.704
3697	14.09	98.728
3698	14.06	98.752
3699	14.03	98.776
3700	14	98.8
3701	13.97	98.824
3702	13.94	98.848
3703	13.91	98.872
3704	13.88	98.896
3705	13.85	98.92
3706	13.82	98.944
3707	13.79	98.968
3708	13.76	98.992
3709	13.73	99.016
3710	13.7	99.04
3711	13.67	99.064
3712	13.64	99.088
3713	13.61	99.112
3714	13.58	99.136
3715	13.55	99.16
3716	13.52	99.184
3717	13.49	99.208
3718	13.46	99.232

time_s	power_kW	soc_pcnt
3719	13.43	99.256
3720	13.4	99.28
3721	13.37	99.304
3722	13.34	99.328
3723	13.31	99.352
3724	13.28	99.376
3725	13.25	99.4
3726	13.22	99.424
3727	13.19	99.448
3728	13.16	99.472
3729	13.13	99.496
3730	13.1	99.52
3731	13.07	99.544
3732	13.04	99.568
3733	13.01	99.592
3734	12.98	99.616
3735	12.95	99.64
3736	12.92	99.664
3737	12.89	99.688
3738	12.86	99.712
3739	12.83	99.736
3740	12.8	99.76
3741	12.77	99.784
3742	12.74	99.808
3743	12.71	99.832
3744	12.68	99.856
3745	12.65	99.88
3746	12.62	99.904
3747	12.59	99.928
3748	12.56	99.952
3749	12.53	99.976
3750	12.5	100

3.1.4. Kona_175kW_350kW

En el caso de cargadores de 175.0kW y 350.0kW, la curva de carga resulta idéntica, tal y como queda recogida en la tabla inmediatamente inferior. En ella, la primera columna se corresponde con el instante de tiempo (en segundos), la segunda con la potencia instantánea consumida por el sistema de baterías (en kW) y la tercera con el SoC o porcentaje de carga.

Nota: De cara a facilitar la legibilidad de la misma se ha optado por rellenar una línea con puntos suspensivos en aquellos rangos de valores en los que la potencia se mantenga constante.

Tabla 3.4. Tabla .CSV de carga para Kona (Hyundai) con cargador de 175.0/350.0kW. (Fuente: Propio).

time_s	**power_kW**	**soc_pcnt**
0	75.00	10.00
1	75.00	10.03
2	75.00	10.07
…	…	…
1472	55.00	60.05
1473	55.00	60.08
1474	55.00	60.12
…	…	…
1766	25.00	70.04
1767	25.00	70.08
1768	25.00	70.11
…	…	…
2354	24.95	90.04
2355	24.91	90.07
2356	24.87	90.10
2357	24.83	90.14
2358	24.78	90.17
2359	24.74	90.21

time_s	power_kW	soc_pcnt
2360	24.70	90.24
2361	24.66	90.27
2362	24.61	90.31
2363	24.57	90.34
2364	24.53	90.38
2365	24.49	90.41
2366	24.44	90.44
2367	24.40	90.48
2368	24.36	90.51
2369	24.32	90.55
2370	24.27	90.58
2371	24.23	90.61
2372	24.19	90.65
2373	24.15	90.68
2374	24.10	90.72
2375	24.06	90.75
2376	24.02	90.78
2377	23.98	90.82
2378	23.93	90.85
2379	23.89	90.89
2380	23.85	90.92
2381	23.81	90.95
2382	23.76	90.99
2383	23.72	91.02
2384	23.68	91.06
2385	23.64	91.09
2386	23.59	91.12
2387	23.55	91.16
2388	23.51	91.19
2389	23.47	91.23
2390	23.42	91.26
2391	23.38	91.29

time_s	power_kW	soc_pcnt
2392	23.34	91.33
2393	23.30	91.36
2394	23.25	91.40
2395	23.21	91.43
2396	23.17	91.46
2397	23.13	91.50
2398	23.08	91.53
2399	23.04	91.57
2400	23.00	91.60
2401	22.96	91.63
2402	22.91	91.67
2403	22.87	91.70
2404	22.83	91.74
2405	22.79	91.77
2406	22.74	91.80
2407	22.70	91.84
2408	22.66	91.87
2409	22.62	91.91
2410	22.57	91.94
2411	22.53	91.97
2412	22.49	92.01
2413	22.45	92.04
2414	22.40	92.08
2415	22.36	92.11
2416	22.32	92.14
2417	22.28	92.18
2418	22.23	92.21
2419	22.19	92.25
2420	22.15	92.28
2421	22.11	92.31
2422	22.06	92.35
2423	22.02	92.38

time_s	power_kW	soc_pcnt
2424	21.98	92.42
2425	21.94	92.45
2426	21.89	92.48
2427	21.85	92.52
2428	21.81	92.55
2429	21.77	92.59
2430	21.72	92.62
2431	21.68	92.65
2432	21.64	92.69
2433	21.60	92.72
2434	21.55	92.76
2435	21.51	92.79
2436	21.47	92.82
2437	21.43	92.86
2438	21.38	92.89
2439	21.34	92.93
2440	21.30	92.96
2441	21.26	92.99
2442	21.21	93.03
2443	21.17	93.06
2444	21.13	93.10
2445	21.09	93.13
2446	21.04	93.16
2447	21.00	93.20
2448	20.96	93.23
2449	20.92	93.27
2450	20.87	93.30
2451	20.83	93.33
2452	20.79	93.37
2453	20.75	93.40
2454	20.70	93.44
2455	20.66	93.47

time_s	power_kW	soc_pcnt
2456	20.62	93.50
2457	20.58	93.54
2458	20.53	93.57
2459	20.49	93.61
2460	20.45	93.64
2461	20.41	93.67
2462	20.36	93.71
2463	20.32	93.74
2464	20.28	93.78
2465	20.24	93.81
2466	20.19	93.84
2467	20.15	93.88
2468	20.11	93.91
2469	20.07	93.95
2470	20.02	93.98
2471	19.98	94.01
2472	19.94	94.05
2473	19.90	94.08
2474	19.85	94.12
2475	19.81	94.15
2476	19.77	94.18
2477	19.73	94.22
2478	19.68	94.25
2479	19.64	94.29
2480	19.60	94.32
2481	19.56	94.35
2482	19.51	94.39
2483	19.47	94.42
2484	19.43	94.46
2485	19.39	94.49
2486	19.34	94.52
2487	19.30	94.56

time_s	power_kW	soc_pcnt
2488	19.26	94.59
2489	19.22	94.63
2490	19.17	94.66
2491	19.13	94.69
2492	19.09	94.73
2493	19.05	94.76
2494	19.00	94.80
2495	18.96	94.83
2496	18.92	94.86
2497	18.88	94.90
2498	18.83	94.93
2499	18.79	94.97
2500	18.75	95.00
2501	18.71	95.03
2502	18.66	95.07
2503	18.62	95.10
2504	18.58	95.14
2505	18.54	95.17
2506	18.49	95.20
2507	18.45	95.24
2508	18.41	95.27
2509	18.37	95.31
2510	18.32	95.34
2511	18.28	95.37
2512	18.24	95.41
2513	18.20	95.44
2514	18.15	95.48
2515	18.11	95.51
2516	18.07	95.54
2517	18.03	95.58
2518	17.98	95.61
2519	17.94	95.65

time_s	power_kW	soc_pcnt
2520	17.90	95.68
2521	17.86	95.71
2522	17.81	95.75
2523	17.77	95.78
2524	17.73	95.82
2525	17.69	95.85
2526	17.64	95.88
2527	17.60	95.92
2528	17.56	95.95
2529	17.52	95.99
2530	17.47	96.02
2531	17.43	96.05
2532	17.39	96.09
2533	17.35	96.12
2534	17.30	96.16
2535	17.26	96.19
2536	17.22	96.22
2537	17.18	96.26
2538	17.13	96.29
2539	17.09	96.33
2540	17.05	96.36
2541	17.01	96.39
2542	16.96	96.43
2543	16.92	96.46
2544	16.88	96.50
2545	16.84	96.53
2546	16.79	96.56
2547	16.75	96.60
2548	16.71	96.63
2549	16.67	96.67
2550	16.62	96.70
2551	16.58	96.73

time_s	power_kW	soc_pcnt
2552	16.54	96.77
2553	16.50	96.80
2554	16.45	96.84
2555	16.41	96.87
2556	16.37	96.90
2557	16.33	96.94
2558	16.28	96.97
2559	16.24	97.01
2560	16.20	97.04
2561	16.16	97.07
2562	16.11	97.11
2563	16.07	97.14
2564	16.03	97.18
2565	15.99	97.21
2566	15.94	97.24
2567	15.90	97.28
2568	15.86	97.31
2569	15.82	97.35
2570	15.77	97.38
2571	15.73	97.41
2572	15.69	97.45
2573	15.65	97.48
2574	15.60	97.52
2575	15.56	97.55
2576	15.52	97.58
2577	15.48	97.62
2578	15.43	97.65
2579	15.39	97.69
2580	15.35	97.72
2581	15.31	97.75
2582	15.26	97.79
2583	15.22	97.82

time_s	power_kW	soc_pcnt
2584	15.18	97.86
2585	15.14	97.89
2586	15.09	97.92
2587	15.05	97.96
2588	15.01	97.99
2589	14.97	98.03
2590	14.92	98.06
2591	14.88	98.09
2592	14.84	98.13
2593	14.80	98.16
2594	14.75	98.20
2595	14.71	98.23
2596	14.67	98.26
2597	14.63	98.30
2598	14.58	98.33
2599	14.54	98.37
2600	14.50	98.40
2601	14.46	98.43
2602	14.41	98.47
2603	14.37	98.50
2604	14.33	98.54
2605	14.29	98.57
2606	14.24	98.60
2607	14.20	98.64
2608	14.16	98.67
2609	14.12	98.71
2610	14.07	98.74
2611	14.03	98.77
2612	13.99	98.81
2613	13.95	98.84
2614	13.90	98.88
2615	13.86	98.91

time_s	power_kW	soc_pcnt
2616	13.82	98.94
2617	13.78	98.98
2618	13.73	99.01
2619	13.69	99.05
2620	13.65	99.08
2621	13.61	99.11
2622	13.56	99.15
2623	13.52	99.18
2624	13.48	99.22
2625	13.44	99.25
2626	13.39	99.28
2627	13.35	99.32
2628	13.31	99.35
2629	13.27	99.39
2630	13.22	99.42
2631	13.18	99.45
2632	13.14	99.49
2633	13.10	99.52
2634	13.05	99.56
2635	13.01	99.59
2636	12.97	99.62
2637	12.93	99.66
2638	12.88	99.69
2639	12.84	99.73
2640	12.80	99.76
2641	12.76	99.79
2642	12.71	99.83
2643	12.67	99.86
2644	12.63	99.90
2645	12.59	99.93
2646	12.54	99.96
2647	12.50	100.00