COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

Aplicación para gestión óptima de batería en una instalación fotovoltaica de autoconsumo

Autor: Carlota Buldú Izquierdo

Director: Jose Luis Sancha Gonzalo

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

“Aplicación para la gestión optima de una batería en una instalación fotovoltaica de autoconsumo”

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2024/2025 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: BULDU IZQUIERDO CARLOTA - 29218420W

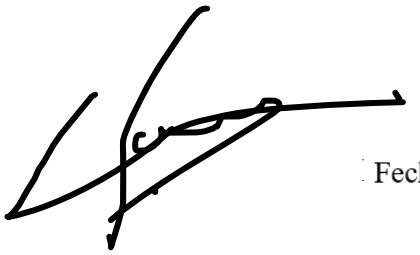
Firmado digitalmente por BULDU IZQUIERDO CARLOTA - 29218420W
Fecha: 2025.08.27 09:35:23 +02'00'

Fecha: 27 / 08 / 2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.:



Fecha: 27 / 08 / 2025

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

Aplicación para gestión óptima de batería en una instalación fotovoltaica de autoconsumo

Autor: Carlota Buldú Izquierdo

Director: Jose Luis Sancha Gonzalo

Madrid

APLICACIÓN PARA LA GESTIÓN ÓPTIMA DE BATERÍA EN UNA INSTALACIÓN FOTOVOLTAICA DE AUTOCONSUMO

Autor: Buldú Izquierdo Carlota.
Director: **Jose Luis Sancha Gonzalo.**
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En este Proyecto se ha desarrollado una aplicación para la gestión óptima de baterías en una instalación fotovoltaica residencial. La lógica implementada decide cuando cargar o descargar la batería o intervenir con la red para maximizar el ahorro económico. Los resultados muestran un ahorro significativo en los costes energéticos y una mejora en el autoconsumo.

Palabras clave: baterías, autoconsumo, ahorro

1. Introducción

El sistema energético español ha entrado en una fase de transición hacia un modelo más sostenible, descentralizado y basado en energías renovables. El autoconsumo fotovoltaico se ha convertido en una de las principales soluciones en esta transformación, con un crecimiento del 131% en nuevas instalaciones en 2023, y más de 7GW de capacidad acumulada, situando a España como líder europeo en expansión solar distribuida (Industria Química, 2025). El objetivo es conseguir para 2030 alcanzar los 14GW de autoconsumo, según el Plan Nacional Integrado de Energía y Clima (PNIEC) 2023-2030, representando un 11% de la demanda eléctrica total (MITECO, 2024).

Sin embargo, se plantean nuevos retos para los usuarios que quieren maximizar el aprovechamiento de su instalación fotovoltaica, como pueden ser la alta variabilidad de la generación solar y los precios volátiles del mercado eléctrico. Las baterías de almacenamiento son una solución a estos potenciales retos, permitiendo almacenar el excedente energético durante las horas de mayor generación solar para así utilizarlo cuando más interese. De este modo, aumentamos el autoconsumo y reducimos la independencia de la red (IDAE, 2024).

Este trabajo propone el diseño e implementación de un sistema de control que simule la operación de una batería en condiciones reales, teniendo en cuenta los datos de generación, consumo y pecios de red. Se presenta el potencial ahorro económico y se contribuye al desarrollo de soluciones que favorecen la eficiencia y sostenibilidad en España. No obstante, también hay que tener en cuenta la rentabilidad de estas baterías, que depende de la eficacia de su gestión.

2. Definición y Descripción del Proyecto

Este Proyecto consiste en desarrollar una herramienta de simulación energética programada en Java, dada su versatilidad y disponibilidad de librerías orientadas a la gestión de datos. La aplicación consiste en analizar hora a hora el comportamiento de la batería ante diferentes situaciones de generación solar y consumo eléctrico, teniendo en cuenta los precios de la red.

El modelo cuenta con las restricciones técnicas habituales, como el límite mínimo y máximo del estado de carga de la batería y la eficiencia del sistema de carga y descarga.

Las entradas necesarias para la simulación son: el perfil horario de generación solar fotovoltaica, el consumo del usuario, los precios de compra y venta por hora y parámetros técnicos como la capacidad de la batería, el SOC inicial, y el rendimiento del sistema. Este diseño se alinea con las recomendaciones del IDAE para el análisis de un sistema como este.

La lógica se organiza mediante estructuras con condiciones que evalúan, hora a hora, la diferencia entre la generación fotovoltaica y el consumo del usuario, y teniendo en cuenta factores como el precio de compra de energía de la red o de venta, toman decisiones sobre el uso de la batería.

El sistema contempla tres escenarios básicos por cada hora:

1. **Consumo mayor que generación**, donde se decide si se extrae energía de la batería o si se consume de la red
2. **Generación mayor que consumo,** donde se evalúa si almacenar el excedente en la batería o si venderlo a la red.
3. **Igualdad o equilibrio,** donde el sistema no realiza cambios

Para facilitar la toma de decisiones, en el tramo de consumo mayor que generación, se clasifican los precios de compra de energía de la red en tres grupos: bajo, medio y alto, realizándose unas operaciones u otras dependiendo del grupo al que pertenezca el precio de la red en esa hora. Cuando el precio es bajo, se aprovecha para cargar la batería o para no consumir la energía de la batería, para aprovecharla en otro momento, y abastecer al exceso de consumo desde la red. Si el precio es alto se consumirá de la batería y si es medio, dependerá del estado de la batería.

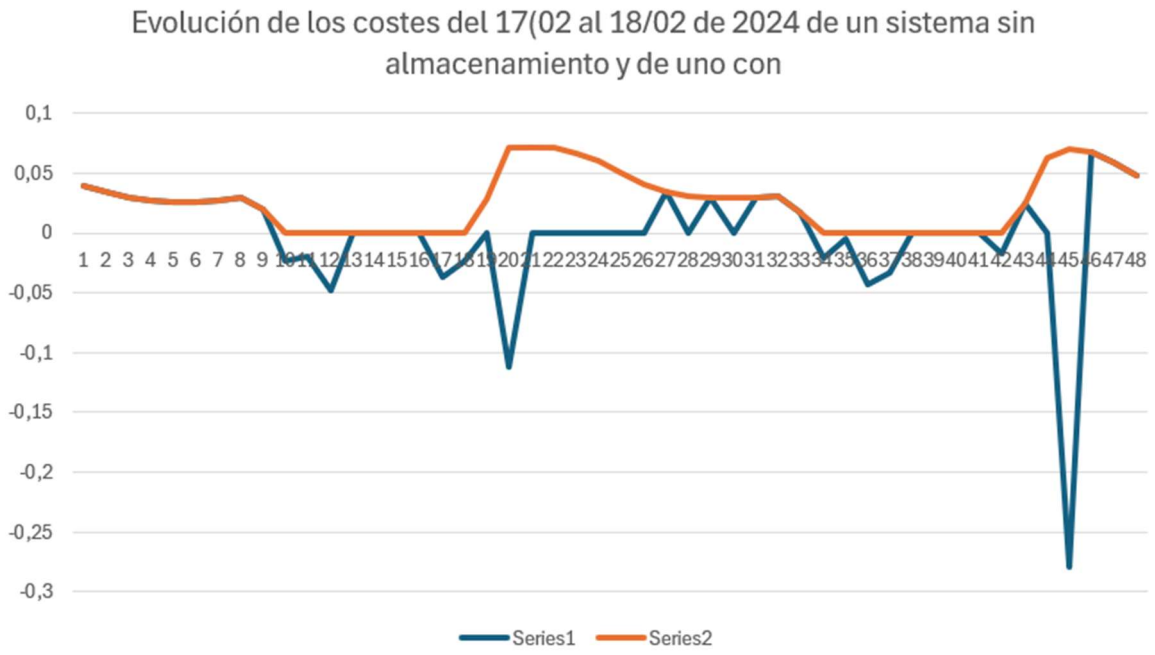
En cuanto a los precios de venta de la red en el tramo de generación mayor que consumo, se carga la batería lo máximo posible y se vende cuando sea necesario, pero siempre en el precio de venta más alto, para maximizar beneficios.

Evolución del estado de carga del 10/04/2024 al 11/04/2024

Esta gráfica, sacada de la aplicación diseñada muestra la evolución del estado de carga de la batería con 13,5 kWh de capacidad nominal. Aquí se puede observar cómo se desarrolla la carga y descarga según la oferta y demanda de energía de la red, entre otros factores.

El objetivo de esta aplicación es maximizar el rendimiento económico de una instalación fotovoltaica equipada con almacenamiento energético. Se quiere demostrar que la gestión optimizada de una batería en un sistema de este tipo permite obtener mayor beneficio económico que operando sin almacenamiento y sin sistema fotovoltaico.

3. Análisis financiero

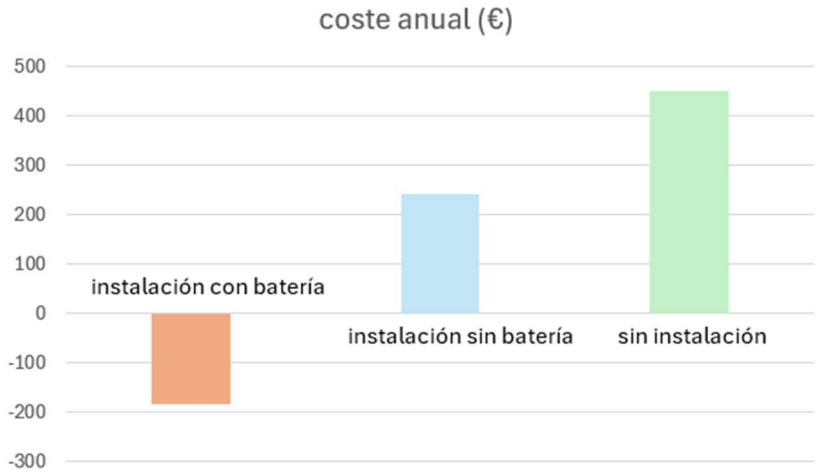


Evolución de los costes de un sistema sin almacenamiento y de uno con almacenamiento

En la figura se muestra la evolución de los costes energéticos entre el 17/02 y el 18/02 de 2024 para dos configuraciones del sistema: Serie 1 corresponde al sistema con almacenamiento y Serie 2 al sistema sin almacenamiento. Aunque se trata de un caso particular, los resultados evidencian un claro ahorro económico al incorporar la batería, ya que los costes medios pasan de 1,248859 € sin almacenamiento a –0,064663 € con almacenamiento. La curva de Serie 1 presenta valores notablemente más bajos e incluso negativos en varias horas, lo que indica situaciones en las que el sistema obtiene beneficios por la venta de excedentes. Por el contrario, la Serie 2 se mantiene en valores positivos la mayor parte del tiempo, reflejando un gasto constante en la compra de energía de la red. Estos resultados confirman la ventaja económica de disponer de almacenamiento para maximizar el aprovechamiento de la generación fotovoltaica.

4. Resultados

El análisis de resultados muestra que la instalación de un sistema fotovoltaico en el ámbito residencial supone una reducción significativa del gasto energético anual, especialmente cuando se incorpora almacenamiento. En ausencia de sistema, el coste anual estimado asciende a 449,95 €, mientras que con un sistema fotovoltaico sin batería se reduce a 241,48€, gracias al autoconsumo instantáneo y al mecanismo de compensación simplificada regulado por el Real Decreto 244/2019. La incorporación de una batería permite alcanzar un escenario de balance económico positivo, con –184,47€, al maximizar el autoconsumo y vender excedentes en periodos de precios altos (IDAE, 2023). Estos resultados coinciden con estudios que indican ahorros de hasta un 85 % en la factura anual y superan los 1 000 € en instalaciones domésticas bien dimensionadas (PV Magazine España, 2024; Fundación Renovables, 2023). Aunque no se ha considerado la amortización de la inversión inicial, los datos evidencian la rentabilidad y el impacto positivo que el autoconsumo con almacenamiento puede tener en la reducción de la dependencia de la red y en la transición hacia un modelo energético más sostenible.



Coste anual de todos los tipos de sistemas

En esta gráfica se puede ver visualmente la diferencia entre tener un sistema fotovoltaico con almacenamiento y sin él.

5. Conclusiones

El crecimiento de la energía solar fotovoltaica convierte a esta aplicación en una herramienta ideal para fomentar su uso, optimizar el autoconsumo y reducir la dependencia de la red. Su escalabilidad permite adaptarla a distintos perfiles, desde viviendas hasta empresas, maximizando el beneficio económico y la eficiencia energética. Además, al priorizar el uso de energía renovable, contribuye de forma directa a la reducción de la huella de carbono y al cumplimiento de los objetivos de sostenibilidad.

6. Referencias

- **Fundación Renovables**. (2023). *Análisis del potencial de autoconsumo residencial en España*. https://fundacionrenovables.org
- **Industria Química.** (2025, 24 de enero). *El autoconsumo sigue avanzando, superando ya el umbral de los 8 GW instalados* (datos UNEF). https://www.industriaquimica.es/noticias/20250124/-autoconsumo-sigue-avanzando-superando-ya-umbral-8gw-instalados
- **Instituto para la Diversificación y Ahorro de la Energía** (IDAE). (2023). *Efectos de la temperatura en el rendimiento fotovoltaico*. https://www.idae.es
- **IDAE.** (2024, 9 de julio). *Guía de Autoconsumo Colectivo (v2.1)* [PDF] — incluye referencias al uso de almacenamiento asociado. https://www.idae.es/sites/default/files/documentos/publicaciones_idae/Guia-Autoconsumo-Colectivo/20240709_Guia_Autoconsumo_Colectivo_v2.1.pdf
- **Ministerio para la Transición Ecológica y el Reto Demográfico (MITECO).** (2024, 24 de septiembre). *Plan Nacional Integrado de Energía y Clima (PNIEC) 2023-2030. Actualización 2024* [PDF]. https://miteco-prod.adobecqms.net/content/dam/miteco/es/energia/files-1/pniec-2023-2030/PNIEC_2024_240924.pdf
  (Página resumen: https://www.miteco.gob.es/es/energia/estrategia-normativa/pniec-23-30.html
- **PV Magazine España**. (2024, 17 de junio). *Autoconsumo doméstico en 2024: los españoles ahorran más de 1 000 euros al año con una instalación media de 11 paneles*. https://www.pv-magazine.es/comunicados/autoconsumo-domestico-en-2024-los-espanoles-ahorran-mas-de-1-000-euros-al-ano-en-su-factura-con-una-instalacion-media-de-11-paneles

.

APPLICATION FOR THE OPTIMAL MANAGEMENT OF A BATTERY IN A SELF-CONSUMPTION PHOTOVOLTAIC INSTALLATION

Author: Buldú Izquierdo, Carlota.
Supervisor: **Jose Luis Sancha Gonzalo.**
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

This project has developed an application for the optimal management of batteries in a residential photovoltaic installation. The implemented logic determines when to charge or discharge the battery or interact with the grid to maximize economic savings. The results show significant reductions in energy costs and improvements in self-consumption.

Keywords: batteries, self-consumption, savings

1. Introduction

The Spanish energy system has entered a transition phase toward a more sustainable, decentralized model based on renewable energy. Photovoltaic self-consumption has become one of the main solutions in this transformation, with a 131% increase in new installations in 2023 and over 7 GW of accumulated capacity, positioning Spain as the European leader in distributed solar expansion (Industria Química, 2025). The goal is to reach 14 GW of self-consumption by 2030, according to the Integrated National Energy and Climate Plan (PNIEC) 2023–2030, representing 11% of total electricity demand (MITECO, 2024).

However, new challenges arise for users seeking to maximize the use of their photovoltaic systems, such as the high variability of solar generation and volatile electricity market prices. Storage batteries are a solution to these challenges, allowing excess energy generated during peak solar hours to be stored for later use when it is most advantageous. This increases self-consumption and reduces dependency on the grid (IDAE, 2024).

This work proposes the design and implementation of a control system that simulates battery operation under real conditions, taking into account generation, consumption, and grid price data. The potential economic savings are presented, contributing to the development of solutions that enhance efficiency and sustainability in Spain. However, it is important to note that the profitability of these batteries depends largely on the effectiveness of their management.

2. Project Definition and Description

This project involves developing an energy simulation tool programmed in Java, chosen for its versatility and the availability of libraries focused on data management. The application analyzes, hour by hour, the behavior of the battery under different scenarios of solar generation and electricity consumption, considering grid prices.

The model incorporates typical technical constraints, such as minimum and maximum battery state of charge (SoC) limits and the efficiency of the charging and discharging system.

The inputs required for the simulation include: hourly profiles of solar generation, user consumption, hourly purchase and sale prices, and technical parameters such as battery capacity, initial SoC, and system efficiency. This design is aligned with recommendations from IDAE for analyzing such systems.

The logic is organized using conditional structures that, hour by hour, evaluate the difference between photovoltaic generation and the user's consumption. Based on factors such as grid purchase or sale prices, the system makes decisions about battery usage.

The system considers three basic scenarios for each hour:

1. Consumption greater than generation: Decides whether to draw energy from the battery or from the grid.
2. Generation greater than consumption: Evaluates whether to store the surplus in the battery or sell it to the grid.
3. Balance/equality: No changes are made.

For decision-making in the "consumption greater than generation" case, grid purchase prices are classified into three categories: low, medium, and high. Different operations are performed depending on the category. When the price is low, the system takes advantage to charge the battery or avoid discharging it, instead covering the excess consumption from the grid. When the price is high, the battery is used to supply energy. If the price is medium, the decision depends on the current battery status.

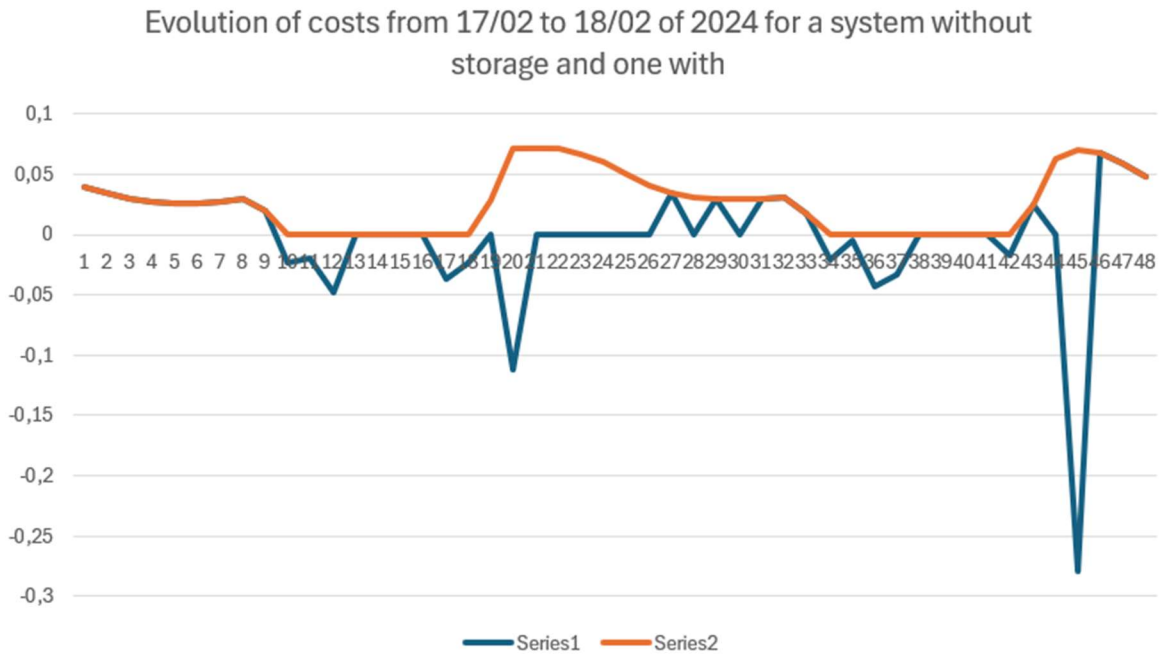
In the "generation greater than consumption" case, the battery is charged as much as possible, and energy is sold when necessary, prioritizing the highest selling price to maximize profits.

State of Charge Evolution (10/04/2024 to 11/04/2024)

This graph, taken from the designed application, shows the evolution of the battery's state of charge with a nominal capacity of 13.5 kWh. It illustrates how charging and discharging occur based on grid supply and demand, among other factors.

The objective of this application is to maximize the economic performance of a photovoltaic installation with energy storage. It aims to demonstrate that optimized battery management in such a system provides greater economic benefits than operating without storage or without a photovoltaic system.

3. Financial Analysis



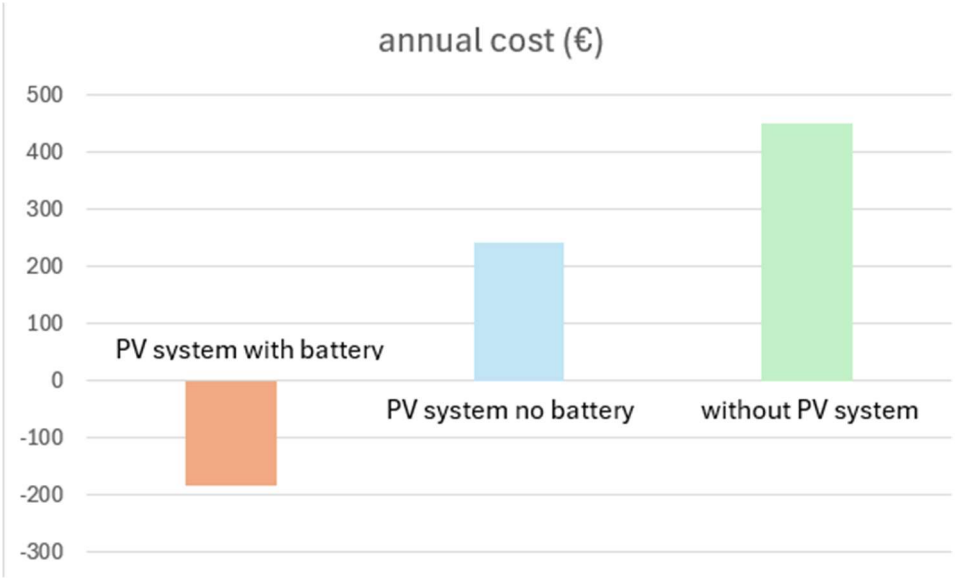
Evolution of the costs of a system without storage and one with storage

The figure shows the evolution of energy costs between 17/02 and 18/02/2024 for two system configurations: Series 1 corresponds to the system with storage and Series 2 to the system without storage. Although this is a particular case, the results clearly show economic savings when a battery is incorporated. Average costs drop from €1.248859 without storage to -€0.064663 with storage. The Series 1 curve remains notably lower and even negative at several hours, indicating situations where the system generates revenue from selling surpluses. In contrast, Series 2 stays mostly in positive values, reflecting constant expenditure on grid energy purchases. These results confirm the economic advantage of using storage to maximize the utilization of photovoltaic generation.

4. Results

The results analysis shows that installing a residential photovoltaic system significantly reduces annual energy expenditure, especially when storage is added. Without a system, the estimated annual cost is €449,95, while with a photovoltaic system without a battery, it drops to €274.81 thanks to instant self-consumption and the simplified compensation mechanism regulated by Royal Decree 244/2019. Adding a battery achieves a positive economic balance of –€36.46 by maximizing self-consumption and selling surpluses during high-price periods (IDAE, 2023).

These findings align with studies reporting savings of up to 85% on annual bills and over €1,000 in well-sized residential installations (PV Magazine España, 2024; Fundación Renovables, 2023). While initial investment amortization is not considered, the data clearly shows the profitability and positive impact of self-consumption with storage in reducing grid dependency and contributing to the transition to a more sustainable energy model.



Annual cost of all system types

5. Conclusions

The growth of photovoltaic solar energy makes this application an ideal tool to encourage its adoption, optimize self-consumption, and reduce grid dependency. Its scalability allows it to adapt to different profiles, from households to businesses, maximizing both economic benefits and energy efficiency. Furthermore, by prioritizing the use of renewable energy, it directly contributes to reducing the carbon footprint and achieving sustainability goals.

6. References

- **Fundación Renovables**. (2023). *Análisis del potencial de autoconsumo residencial en España*. https://fundacionrenovables.org
- **Industria Química.** (2025, 24 de enero). *El autoconsumo sigue avanzando, superando ya el umbral de los 8 GW instalados* (datos UNEF). https://www.industriaquimica.es/noticias/20250124/-autoconsumo-sigue-avanzando-superando-ya-umbral-8gw-instalados
- **Instituto para la Diversificación y Ahorro de la Energía** (IDAE). (2023). *Efectos de la temperatura en el rendimiento fotovoltaico*. https://www.idae.es
- **IDAE.** (2024, 9 de julio). *Guía de Autoconsumo Colectivo (v2.1)* [PDF] — incluye referencias al uso de almacenamiento asociado. https://www.idae.es/sites/default/files/documentos/publicaciones_idae/Guia-Autoconsumo-Colectivo/20240709_Guia_Autoconsumo_Colectivo_v2.1.pdf
- **Ministerio para la Transición Ecológica y el Reto Demográfico (MITECO).** (2024, 24 de septiembre). *Plan Nacional Integrado de Energía y Clima (PNIEC) 2023-2030. Actualización 2024* [PDF]. https://miteco-prod.adobecqms.net/content/dam/miteco/es/energia/files-1/pniec-2023-2030/PNIEC_2024_240924.pdf
  (Página resumen: https://www.miteco.gob.es/es/energia/estrategia-normativa/pniec-23-30.html
- **PV Magazine España**. (2024, 17 de junio). *Autoconsumo doméstico en 2024: los españoles ahorran más de 1 000 euros al año con una instalación media de 11 paneles*. https://www.pv-magazine.es/comunicados/autoconsumo-domestico-en-2024-los-espanoles-ahorran-mas-de-1-000-euros-al-ano-en-su-factura-con-una-instalacion-media-de-11-paneles

Índice de la memoria

Contenido

Índice de ilustraciones

UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Capítulo 1. INTRODUCCIÓN

En las últimas décadas el sistema energético global se ha transformado radicalmente, debido a necesidad de reducir las emisiones de gases de efecto invernadero y mejorar la eficiencia energética. La manera para avanzar hacia un modelo energético más sostenible, descentralizado y resiliente es mediante las energías renovables.

En este proyecto se ha desarrollado una aplicación para gestionar de forma inteligente una batería en un sistema fotovoltaico en un domicilio para así poder proporcionar un beneficio económico y a la vez sostenible para los usuarios.

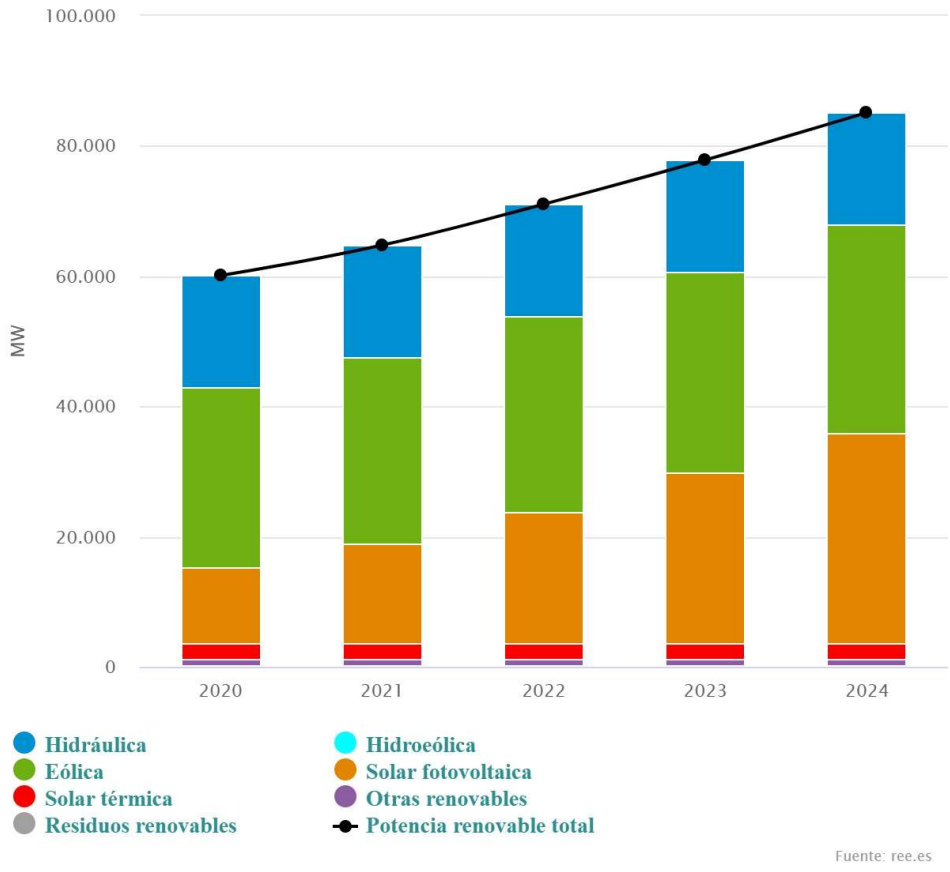


Ilustración 1. Evolución de la potencia instalada renovable

En la ilustración 1 se muestra que la energía solar fotovoltaica ha adquirido un protagonismo creciente en España, siendo la energía renovable la que más ha crecido desde 2020 a 2024, pasando de 11.373 MW a 32. 350 MW instalados. Según el Informe del Sistema Eléctrico Español de Red Eléctrica de España, en 2023 la potencia instalada de energía solar fotovoltaica alcanzó los 25.579 MW, representando un 20,6% de la potencia total renovable en el país (REE, 2024). Este aumento ha sido impulsado por la bajada de precios de los paneles y los incentivos al autoconsumo.

Sin embargo, es importante mencionar que la energía solar presenta inconvenientes, como su variabilidad, Su producción depende de las condiciones meteorológicas y por las noches se frena. Las baterías son una solución clave a este problema, ya que permite almacenar los excedentes para utilizarlos cuando más convenga. Esto incrementa el autoconsumo, reduce la energía demandada de la red y aporta una mayor autonomía energética al usuario (Salvis-E, 2024; Sunnova, 2024).

1.1 MOTIVACIÓN DEL PROYECTO

El contexto energético actual ha demostrado la necesidad de transitar hacia un modelo más eficiente y sostenible. El apagón eléctrico de 2025 en la península ibérica es la evidencia de que el sistema actual es vulnerable, y que existe una fuerte dependencia hacia la red eléctrica convencional (Cadena SER, 2025a).

La energía renovable, concretamente la fotovoltaica, representa una solución para reducir esta dependencia, especialmente si se combina con sistemas de almacenamiento energético como las baterías. Esto no solo permitirá al usuario maximizar su autoconsumo y mantener un suministro energético incluso en situaciones de fallo de la red (Sunnova, 2024), sino que también ofrecen un ahorro energético a largo plazo (PV Magazine España, 2025; EDP Energía, 2023) y una contribución significativa a la reducción de emisiones (Salvis-E, 2024; Vallesolar, 2023).

Este proyecto surge con la motivación de demostrar que una herramienta efectiva para que los ciudadanos consigamos independencia energética, ahorro económico y una participación en disminuir la huella de carbono es integrando baterías en sistemas fotovoltaicos en domicilios.

1.2 OBJETIVO DEL TRABAJO

El objetivo de este trabajo es diseñar y validar una aplicación que permita gestionar inteligentemente el uso de sistemas fotovoltaicos residenciales con almacenamiento. La finalidad de esta aplicación es optimizar el ahorro energético, reducir la dependencia de la red y contribuir a la disminución de emisiones contaminantes.

Uno de los principales objetivos de esta aplicación es conseguir que haya un ahorro en el gasto energético de los usuarios. A través de la programación de una lógica de carga y

descarga eficiente, que analice cuando se debe priorizar el uso de la energía almacenada y maximice el autoconsumo instantáneo, buscando reducir la compra de electricidad de la red. Estudios realizados en España han demostrado que el uso de baterías en sistemas fotovoltaicos puede aumentar el ahorro anual por vivienda más de 280 € y alcanzar una diferencia acumulada de hasta 5.000 € en diez años comparado a instalaciones sin almacenamiento (PV Magazine España, 2025). Con el uso de las baterías se permite una mejor adaptación a la tarifa PVPC y a los tramos horarios, lo que también potencia el beneficio económico (EDP Energía, 2023).

Además del beneficio económico y operativo, la propuesta de este proyecto también tiene un impacto medioambiental. Las emisiones de CO_2 pueden reducirse si se utiliza la energía solar en lugar de electricidad generada a partir de fuentes fósiles. A esto se le suma que con el uso de baterías se permite aprovechar excedentes solares que sin ellas se perderían, aumentando la proporción de energía limpia consumida en un domicilio (Salvis-E, 2024). Según estimaciones recientes, se puede reducir entre un 70% y un 85% las emisiones de CO_2 con una instalación media de autoconsumo con batería (Vallesolar, 2023).

Esta aplicación se desarrolla mediante una lógica y estrategias de gestión energética, que busca maximizar contribuir con el medio ambiente y beneficiar al usuario mediante un ahorro económico.

Capítulo 2. FUNDAMENTOS Y CONTEXTO DEL SISTEMA

2.1 INTRODUCCIÓN GENERAL

Este capítulo explica las normativas, el desarrollo y el funcionamiento de los conceptos que forman la aplicación propuesta. Mientras que el primer capítulo presenta la motivación y objetivos, en este se profundiza en los elementos esenciales que se utilizan en el diseño, simulación y lógica del sistema, contextualizados en el marco español.

2.2 FUNDAMENTOS TÉCNICOS Y CONTEXTO DEL SISTEMA

Un sistema fotovoltaico típico residencial en España incluye paneles solares, inversor, contador bidireccional y, opcionalmente, almacenamiento con baterías. La corriente continua generada se convierte en alterna para el hogar o para inyectar excedentes a la red. Este proceso está regulado por el Real Decreto 244/2019, que define los modos de autoconsumo con y sin vertido, los requisitos técnicos y económicos. Al cierre de 2024, la potencia fotovoltaica instalada alcanzó 31,7 GW en España, situándose como uno de los principales motores del mix renovable nacional.

2.2.1 FUNCIONAMIENTO BÁSICO DE UN SISTEMA FOTOVOLTAICO DOMÉSTICO

El proceso comienza con la captación de la radiación solar por parte de los módulos fotovoltaicos, los cuales convierten la energía solar en electricidad en corriente continua (CC). Esta electricidad es posteriormente transformada a corriente alterna (CA) mediante un inversor, para poder alimentar los dispositivos eléctricos del domicilio, ya que la mayoría de las instalaciones residenciales operan en CA.

El contador bidireccional, obligatorio en instalaciones conectadas a la red, permite registrar tanto la electricidad consumida desde la red como los excedentes de generación vertidos a la misma. De este modo, si en algún momento el sistema ha de vender energía a la red, lo que permite al usuario acogerse a modalidades como el autoconsumo con compensación

simplificada, regulada por el Real Decreto 244/2019. Este decreto establece los procedimientos administrativos, técnicos y económicos para el autoconsumo, promoviendo un marco favorable para su expansión (BOE, 2019).

Un componente opcional, pero cada vez más presente, es el sistema de almacenamiento mediante baterías, que permite almacenar la energía solar que no se haya utilizado y permite tomar decisiones sobre cuando utilizarla. Esta incorporación es necesaria para el uso de esta aplicación, para no solo mejorar el nivel de autonomía energética, sino también para generar ahorros económicos.

2.2.2 ALMACENAMIENTO ENERGÉTICO: PRINCIPIOS, EFICIENCIA, SOC, DOD

Para poder integrar eficazmente el almacenamiento en una lógica de gestión energética, es imprescindible tener en cuenta una serie de parámetros técnicos clave que definen el comportamiento y las limitaciones físicas de las baterías, y que tienen un impacto directo en el rendimiento, la durabilidad y la rentabilidad del sistema.

2.2.2.1 SoC (State of Charge) y DoD (Depth of Discharge)

El State of Charge (SoC) indica el nivel actual de carga de la batería, expresado como un porcentaje del total de su capacidad nominal. Por ejemplo, un SoC del 80 % indica que la batería tiene almacenado el 80 % de su capacidad total. Para que sea más visual para el usuario, la aplicación muestra en KWh el estado de la batería, de esta forma se sabe exactamente cuanta energía hay en la batería.

El Depth of Discharge (DoD), por el contrario, indica el porcentaje de energía descargada respecto a la capacidad total. Una descarga completa de la batería (hasta el 0 %) equivaldría a un DoD del 100 %. Ambos parámetros están íntimamente relacionados: un SoC del 30 % implica un DoD del 70 %, y viceversa. Este parámetro no lo mencionamos tanto como el anterior, al estar tan relacionados.

Estos valores son esenciales para controlar el comportamiento de carga y descarga en el sistema. Diversos estudios técnicos y fabricantes recomiendan mantener el SoC dentro de márgenes seguros, típicamente entre el 20 % y el 80 %, para prolongar la vida útil de la batería y evitar degradaciones prematuras (Motor.es, s.f). Esta es precisamente la lógica que se aplica en el algoritmo de control de este proyecto, donde se establecen umbrales operativos que impiden que la batería baje del 20 % de carga (nivel de reserva de seguridad) o supere el 80 % (nivel de carga máximo operativo). Esta estrategia busca equilibrar el aprovechamiento energético con la durabilidad del sistema.

2.2.2.2 Eficiencia round-trip

La eficiencia round-trip mide la relación entre la energía que se extrae de la batería y la que se utilizó para cargarla. Por ejemplo, si se introducen 100 kWh en una batería y se recuperan

90 kWh, la eficiencia round-trip es del 90 %. En baterías de ion-litio, este valor suele oscilar entre el 85 % y el 95 % dependiendo del modelo y el régimen de uso (Moa & Go, 2023). Esta pérdida de energía es debido a que hay parte que se disipa y no se utiliza para lo que se desea.

Este dato es vital para calcular con precisión los beneficios económicos del almacenamiento, ya que una menor eficiencia implica mayores pérdidas. En el algoritmo desarrollado para el presente proyecto se asume una eficiencia constante del 90 %, y se incorpora en los cálculos del balance energético horario y para estimar correctamente el coste energético real de las cargas y descargas.

Autodescarga y degradación por ciclos

Las baterías sufren procesos de autodescarga, es decir, pérdidas de energía almacenada sin que haya consumo asociado. Aunque este efecto es más pronunciado en tecnologías antiguas, en las baterías de ion-litio la tasa de autodescarga ronda el 2-3 % mensual, lo que sigue siendo relevante en aplicaciones que requieren precisión y planificación (Battery University, s.f).

Por otra parte, cada ciclo completo de carga y descarga provoca un pequeño desgaste en la batería. Con el tiempo, esta degradación por ciclos reduce la capacidad útil de almacenamiento. La vida útil se suele medir en número de ciclos completos hasta que la batería retiene un 80 % de su capacidad inicial. Para baterías domésticas, esto puede situarse en torno a 6.000–8.000 ciclos, lo que equivale a unos 15 años de operación bajo condiciones óptimas (Moa & Go, 2023).

En este proyecto no se ha tenido en cuenta este concepto, ya que complicaba los cálculos. Más adelante se menciona por qué es un necesario incluir la autodescarga y degradación por ciclos y se presenta una posible solución a este problema.

2.2.3 GESTIÓN ENERGÉTICA: DEFINICIÓN, IMPORTANCIA, INDICADORES CLAVE

La gestión energética es el conjunto de estrategias, procesos y herramientas destinadas a optimizar el uso de la energía en un sistema, garantizando un suministro fiable, económico y ambientalmente sostenible. En el contexto de un sistema fotovoltaico doméstico con almacenamiento, la gestión energética cobra un papel esencial para maximizar el aprovechamiento de la energía generada, reducir la dependencia de la red y mejorar la rentabilidad económica del sistema. La principal estrategia es priorizar el autoconsumo y almacenar la energía sobrante, para crear una lógica que gestione esta energía.

Las funciones principales de la gestión energética en este proyecto incluyen la monitorización en tiempo real del consumo y generación, la toma de decisiones sobre cuándo cargar o descargar la batería, y la planificación de estrategias de uso en función de variables como los precios de la electricidad, las previsiones meteorológicas o el patrón de uso

energético del hogar. También es clave la priorización de fuentes (solar, batería o red) siempre en función del coste, para reducirlos al máximo.

En España, esta gestión energética resulta especialmente relevante debido al impulso del autoconsumo y a los nuevos modelos tarifarios horarios establecidos por el Gobierno y las comercializadoras eléctricas. El sistema eléctrico español ha avanzado hacia una mayor penetración de renovables y digitalización, lo que exige a los consumidores una mayor conciencia y control sobre su consumo energético.

El dato central sobre el que pivota la gestión energética en esta aplicación es la diferencia entre generación y consumo en cada hora del día. Esta variable permite saber si se está produciendo más energía de la que se necesita (excedente) o si hay una demanda no cubierta por la generación (déficit). En el caso de excedente, la aplicación decide si almacenar la energía en la batería o verterla a la red, según el estado de carga y el precio de venta en esa hora. En cambio, si hay un déficit, se evalúa si es preferible descargar la batería o recurrir a la red, teniendo en cuenta el precio de compra en ese momento. En cada hora se calculan los costes, según se haya tomado una decisión u otra y así se compara y se analiza que haya menos gastos utilizando esta lógica (Gamarra & Guerrero, 2020).

2.2.4 INTRODUCCIÓN A LOS EMS Y SOFTWARE DE CONTROL ENERGÉTICO

Los Sistemas de Gestión Energética (EMS) son plataformas digitales que combinan hardware y software para supervisar, controlar y optimizar el uso de la energía en tiempo real. Estos sistemas integran datos de generación (solar), consumo, estado de carga de baterías, precios horarios y condiciones ambientales para tomar decisiones automáticas que priorizan la eficiencia energética, la sostenibilidad y el ahorro económico (SotySolar, 2024).

La aplicación desarrollada en este proyecto funciona como un EMS doméstico, adaptado para instalaciones fotovoltaicas con almacenamiento en viviendas. Cada hora evalúa la diferencia entre generación y consumo, gestiona la carga o descarga de la batería según el SoC actual, compara los precios de compra y venta, y decide si importar, exportar o almacenar energía con lógica optimizada. Al estar diseñada específicamente para el contexto residencial español con perfiles reales de generación, consumo y mercado eléctrico, se diferencia de EMS comerciales genéricos por su carácter personalizado y enfocado en usuarios finales.

Según la Fundación Naturgy, la integración de EMS potencia la eficiencia energética, la flexibilidad de la red y la resiliencia frente a eventos extremos, convirtiéndose en una base estratégica para el modelo de smart grid y la digitalización del sistema eléctrico español (Fundación Naturgy, 2021)

2.3 *MARCO LEGAL Y NORMATIVO APLICABLE*

2.3.1 NORMATIVA ESPAÑOLA SOBRE AUTOCONSUMO (RD 244/2019)

El Real Decreto 244/2019 establece el marco regulatorio para el autoconsumo eléctrico en España, tanto individuales como colectivas. Define cómo pueden integrarse instalaciones fotovoltaicas con y sin vertido de excedentes a la red, los requisitos técnicos (como la medición bidireccional), los trámites administrativos (como registros en los sistemas informáticos del operador de red). Permite el autoconsumo con excedentes acogidos a compensación, y la participación en comunidades energéticas (RD 244/2019, artículo 4.5). Según fuentes de la Unión Española Fotovoltaica (UNEF), gracias a esta normativa se alcanzaron más de 7GW de potencia instalada en autoconsumo individual a finales de 2023 (Energías Renovables, 2024).

Esta norma es clave para este proyecto, ya que establece los requisitos para la puesta en marcha de las instalaciones. Este articulo contiene la información del procedimiento para la conexión de estas instalaciones a la red eléctrica.

2.3.2 NORMATIVAS TÉCNICAS APLICABLES A INSTALACIONES FOTOVOLTAICAS (UNE, RITE)

Las normas UNE aplicables garantizan la calidad, eficiencia y seguridad de los componentes eléctricos: paneles, inversores, protección contra sobretensiones, y conexión con la red interior de vivienda. Mientras que, el Reglamento de Instalaciones Térmicas en Edificios (RITE) regula la correcta instalación y seguridad de sistemas térmicos como agua caliente sanitaria o climatización si se integran con sistemas fotovoltaicos. Aunque el foco principal sea eléctrico, el cumplimiento conjunto de estas normas es esencial para cualquier instalación segura y legal.

La adecuación técnica a estas normas se refleja en las especificaciones que tu aplicación debe considerar al modelar el sistema: potencia máxima, eficiencia técnica real, pérdidas por cableado y gestión.

2.3.3 REGULACIÓN DEL ALMACENAMIENTO ENERGÉTICO

Después del gran apagón de abril de 2025, el Real Decreto-ley 7/2025 declaró el almacenamiento como infraestructura estratégica. Se incorpora el almacenamiento como elemento flexible del sistema, simplifica trámites y habilita su inclusión legal en instalaciones renovables. Además, extiende el radio del autoconsumo colectivo de 2 km a 5 km, ampliando así el alcance de comunidades energéticas urbanas o interconectadas (Cadena SER Andalucía, 2025).

2.3.4 INCENTIVOS Y SUBVENCIONES AL AUTOCONSUMO

España ha destinado hasta 1.320 millones de euros en apoyo al autoconsumo, almacenamiento y sistemas térmicos mediante programas del Plan de Recuperación y fondos FEDER gestionados por IDAE (MITECO). En comunidades como Andalucía, se han concedido subvenciones de hasta el 85 % del coste total: unos 1.000 €/kWp para paneles y 490 €/kWh para baterías domésticas, incluso en municipios menores de 5.000 habitantes bajo el programa DUS 5000 (La Moncloa, 2022).

En Andalucía, por ejemplo, se lanzaron convocatorias donde comunidades energéticas y ayuntamientos pueden cubrir hasta el 85 % del coste total, con ayudas de hasta 1.000 €/kWp para paneles y 490 €/kWh para baterías (MITECO, 2025).

Estos incentivos reducen significativamente el coste de adopción residencial y comunitaria del autoconsumo con almacenamiento, lo cual refuerza la viabilidad técnica y comercial de esta aplicación. Permite a los usuarios simular no solo el ahorro energético, sino también el acceso a subvenciones en función de su ubicación y tipo de instalación.

2.4 ESTADO DEL ARTE

2.4.1 CONTEXTO ENERGÉTICO Y TRANSICIÓN RENOVABLE EN ESPAÑA Y EUROPA

El contexto energético en España se encuentra actualmente en una fase de transición acelerada hacia un modelo más sostenible, enmarcado dentro de los objetivos europeos del Pacto Verde y el Plan Nacional Integrado de Energía y Clima (PNIEC) 2021–2030. Esta transformación está marcada por la descarbonización del mix energético, la apuesta por la electrificación y la sustitución progresiva de fuentes fósiles por energías renovables, entre las que destaca, por su crecimiento y potencial, la energía solar fotovoltaica.

Según datos de Red Eléctrica de España (REE, 2024a), al cierre del año 2024 la potencia solar fotovoltaica instalada en el país superó los 31,7 GW, como se ha mencionado anteriormente, consolidándose como la fuente renovable con mayor ritmo de crecimiento en el sistema eléctrico nacional. Este avance sostenido se debe, en gran medida, a la reducción de los costes de componentes como módulos e inversores, al incremento del precio de la electricidad en el mercado mayorista y, especialmente, al despliegue de políticas públicas que impulsan el autoconsumo tanto a nivel estatal como en las comunidades autónomas.

La evolución del sector no solo responde a la mejora tecnológica, sino también al auge de nuevos modelos energéticos. Entre ellos destacan las comunidades energéticas locales y el autoconsumo compartido. Esta transformación se ve reforzada por la creciente digitalización del sector, así como por el desarrollo de sistemas de gestión energética inteligentes que optimizan el uso de la energía renovable en tiempo real. Como resultado, el consumidor

adopta un nuevo rol: el de prosumer, es decir, productor y consumidor de su propia energía, contribuyendo activamente a la sostenibilidad del sistema eléctrico (IDAE, 2023).

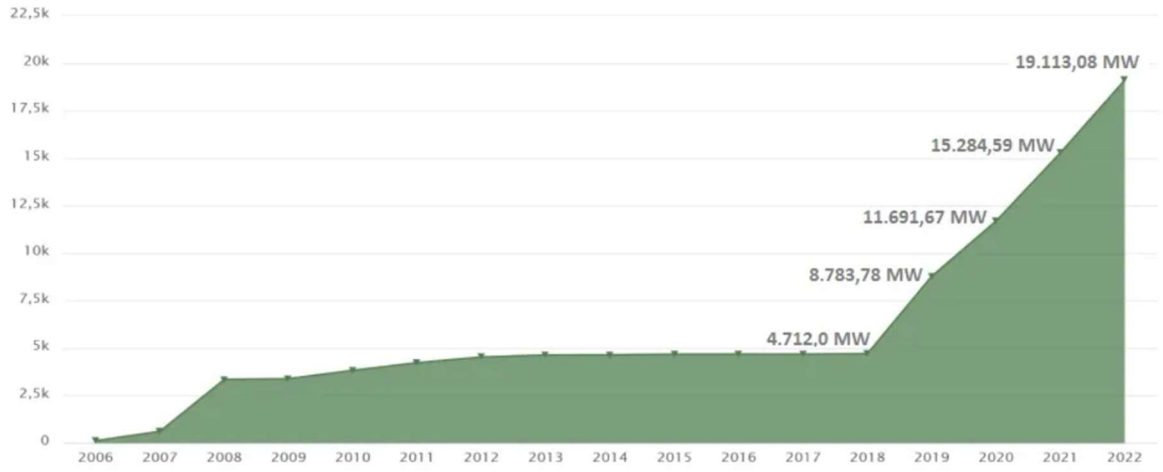


Ilustración 2. Potencia solar fotovoltaica instalada hasta 2022

Fuente: (Energías Renovables, 2023)

Esta gráfica muestra el crecimiento de la producción de energía solar fotovoltaica de una manera más visual, reflejando la evolución sostenida de esta tecnología renovable en España durante los últimos años. Entre 2018 y 2022, la producción solar se multiplicó por más de tres, pasando de 7.089 GWh a 27.101 GWh, según datos de Red Eléctrica de España (REE, 2024a). Este crecimiento se vio impulsado por varios factores: la eliminación del llamado "impuesto al sol" en 2018, la aprobación del Real Decreto 244/2019 que regula el autoconsumo, la reducción significativa de costes en los paneles solares, y un contexto de precios eléctricos al alza que incentivó la inversión en instalaciones propias. Aquí se ve el cambio más drástico en la producción de energía solar fotovoltaica.

Además, el despliegue de ayudas públicas, especialmente a través de fondos europeos como los programas NextGenerationEU, ha sido clave para facilitar la adopción masiva de sistemas solares residenciales, industriales y comunitarios. Esta tendencia se consolida en el nuevo Plan Nacional Integrado de Energía y Clima, donde se establece como objetivo alcanzar los 76 GW de potencia fotovoltaica instalada en 2030, más del triple de la cifra actual (IDAE, 2024). Se espera que la solar fotovoltaica juegue un papel central en la estrategia nacional para alcanzar el 74 % de generación eléctrica renovable en 2030.

2.4.2 TECNOLOGÍAS DE ALMACENAMIENTO ENERGÉTICO

En el contexto de la transición energética en España, el almacenamiento eléctrico ha avanzado desde una opción marginal hasta convertirse en un elemento imprescindible para garantizar la eficiencia, resiliencia e integración de las energías renovables en el sistema eléctrico nacional. Debido a la tecnología de almacenamiento, la creciente implantación de la fotovoltaica residencial ha encontrado un apoyo clave para estabilizar la producción, mejorar el autoconsumo y reducir la dependencia de la red.

2.4.2.1 Baterías de ion-litio

Las baterías de ion-litio (Li-ion) se han convertido en la opción predominante, debido a su elevada densidad energética, eficiencia de carga/descarga, larga vida útil y costes decrecientes gracias a su producción a gran escala. Estas tecnologías de almacenamiento tienen un papel clave en la transición hacia un sistema energético más flexible, sostenible y descentralizado.

En el ámbito doméstico e industrial, las baterías de ion-litio permiten almacenar excedentes de energía generada —principalmente de origen renovable, como la solar fotovoltaica— y utilizarlos en momentos de baja producción o alta demanda, reduciendo así la dependencia de la red eléctrica y mejorando el autoconsumo. Siendo esta la funcionalidad necesaria para poder cumplir los objetivos de la aplicación. Su tiempo de respuesta es rápido, lo que las hace también adecuadas para servicios auxiliares a la red, como la regulación de frecuencia o la gestión de picos de demanda. Estos suelen coincidir con tarifas eléctricas más altas, por lo que gestionarlos puede resultar en ahorros significativos para los consumidores, y también facilita la integración de fuentes de energía intermitentes, al permitir un mayor equilibrio entre oferta y demanda (CIRCE, 2022).

En España, este tipo de almacenamiento está en pleno crecimiento, impulsado por políticas de apoyo como las ayudas del Plan de Recuperación, Transformación y Resiliencia (Componente 7), que financia instalaciones de autoconsumo y almacenamiento (MITECO, 2024). Se están desarrollando también plantas de almacenamiento a gran escala, como la batería de 20 MWh de Iberdrola en Olmedilla (Castilla-La Mancha), destinada a apoyar la integración de fotovoltaica en la red (Iberdrola, 2023).

2.4.2.2 Economía circular y segunda vida de las baterías

La economía circular maximiza el valor de los productos durante su ciclo de vida y reduce la generación de residuos. En el caso de las baterías de ion-litio, una de las estrategias más prometedoras es su reutilización tras su primera vida útil en vehículos eléctricos, cuando aún conservan entre el 70% y el 80% de su capacidad (Endesa, 2022a).

Estas "baterías de segunda vida" se están aplicando en sistemas estacionarios de almacenamiento energético. El proyecto de Endesa en Melilla, es un ejemplo en España, donde se ha instalado un sistema de almacenamiento con baterías procedentes de vehículos Nissan Leaf, con una capacidad de 4 MW y 1,7 MWh, que permite garantizar el suministro eléctrico en caso de desconexión de la red principal (Endesa, 2022b).

La economía circular tiene varias ventajas: disminuye la demanda de materias primas críticas (litio, cobalto, níquel), alarga la vida útil de componentes costosos y reduce el impacto ambiental asociado a su reciclaje prematuro. No obstante, también presenta desafíos, ya que la batería no se encuentra en su estado óptimo: el rendimiento es inferior al de baterías nuevas, requieren procesos de selección y reacondicionamiento, y todavía existe cierta incertidumbre regulatoria en torno a su certificación y uso (Blog Enerlink, 2023**)**.

La segunda vida de baterías podría ser especialmente útil en soluciones distribuidas de autoconsumo residencial, donde los requisitos de potencia y durabilidad son menos exigentes que en el sector de la automoción o de industria. Además, se alinea con los objetivos de sostenibilidad, reducción de huella de carbono y economía circular, muy valorados en proyectos innovadores.

2.4.3 APLICACIONES COMERCIALES ACTUALES

Actualmente existen diversas soluciones comerciales que integran baterías de ion-litio para el almacenamiento doméstico y comercial. Algunas de las más relevantes son:

2.4.3.1 Tesla Powerwall y Powerpack

Tesla ha liderado el mercado de almacenamiento residencial con su sistema Powerwall, una batería de ion-litio de 13,5 kWh destinada a hogares con autoconsumo fotovoltaico. También ofrece versiones de mayor capacidad para empresas o comunidades energéticas (Tesla, 2023). En España, numerosas instalaciones ya combinan fotovoltaica y Powerwall, gestionadas mediante la aplicación Tesla Energy.

2.4.3.2 SonnenBatterie

Sonnen (Grupo Shell) es otra referencia internacional en soluciones modulares de almacenamiento inteligente. Sus sistemas permiten maximizar el autoconsumo, gestionar la carga del vehículo eléctrico, e incluso formar parte de comunidades energéticas virtuales, como la "sonnenCommunity" (Sonnen, 2023). En Europa ya operan agregadores que combinan miles de estas baterías para ofrecer servicios a la red.

2.4.3.3 Huawei LUNA2000

Huawei ha introducido en el mercado europeo sus baterías residenciales modulares LUNA2000, disponibles en configuraciones de 5 a 30 kWh. Son compatibles con sus inversores solares híbridos y permiten una integración eficiente con instalaciones fotovoltaicas, especialmente en domicilios unifamiliares (Huawei, 2023). Su sistema de gestión inteligente optimiza el uso según patrones de consumo y generación.

Estas soluciones representan modelos de negocio escalables, basados en la combinación de generación renovable, almacenamiento, inteligencia artificial y digitalización, como la aplicación diseñada en este proyecto.

2.4.4 OPORTUNIDAD Y JUSTIFICACIÓN DE LA APLICACIÓN DESARROLLADA

El rápido desarrollo de la energía solar fotovoltaica en España ha generado un contexto especialmente favorable para la implementación de herramientas de gestión energética avanzadas. En los últimos años, el país ha experimentado un crecimiento exponencial tanto en potencia instalada como en producción. La situación en España es muy favorable para que las aplicaciones de gestión energética triunfen y proporcionen beneficios a los usuarios, en términos de eficiencia, ahorro económico y sostenibilidad.

Capítulo 3. DISEÑO DEL SISTEMA PROPUESTO

3.1 REQUISITOS FUNCIONALES DE LA APLICACIÓN

Estos requisitos definen las funcionalidades esenciales que se han incorporado a la aplicación para que se puedan cumplir los objetivos. En este caso el objetivo global es ofrecer una herramienta que permita simular y optimizar el uso de una batería en un sistema fotovoltaico doméstico, pero existen requisitos más concretos para conseguir el correcto funcionamiento de la aplicación.

Los principales requisitos funcionales necesarios para el desarrollo de la aplicación son los siguientes:

Se ha realizado una simulación horaria para reflejar el funcionamiento realista de un sistema fotovoltaico con batería. Se tienen en cuenta los datos horarios del consumo eléctrico, la generación solar fotovoltaica, el precio de excedentes y el precio de la red. De este modo se calcula en cada hora los valores necesarios para la lógica.

El cálculo horario del estado de carga (SoC) es imprescindible, ya que para ver si la lógica funciona correctamente, hay que analizar el estado de carga de la batería en cada hora, y esto solo se puede hacer calculándola en cada hora dependiendo del escenario en el que se encuentre la batería. Más adelante se describirán los diferentes cálculos realizados para obtener el SoC dependiendo del consumo eléctrico y la generación solar, de los estados anteriores, y de los precios.

La finalidad de tener un sistema fotovoltaico es aprovechar al máximo el autoconsumo, por eso esta es la prioridad de la lógica, en segundo lugar, el uso de la batería y por último la compra de energía de la red. Teniendo claras las prioridades es cuando se empieza a pensar más en profundidad y en detalle el funcionamiento de la lógica.

El uso de datos reales es fundamental en este proyecto, debido a que todos los cálculos, comparaciones y análisis se realizan con información horaria precisa. Por eso en la programación de esta lógica se permite la entrada de datos externos mediante archivos, como perfiles de generación solar fotovoltaica, consumo eléctrico y precios de la red y de venta de excedentes. Estos datos se obtienen de Red Eléctrica Española, por lo tanto, se garantiza que son datos obtenidos de fuentes oficiales y originales. La calidad y resolución de los datos empleados influye directamente la fiabilidad de los resultados (PV Magazine España, 2025)

Debido a que esta aplicación está orientada a usuarios particulares, se facilita la comparación entre distintos escenarios personalizados, con diferentes capacidades de batería y configuraciones fotovoltaicas. De este modo se crea una herramienta adaptable a cualquier vivienda. La aplicación muestra una tabla con todos los valores generados utilizando una batería, pero también analiza los costes y ahorro económico sin baterías y se comparan con un domicilio sin sistema fotovoltaico. De este modo se podrá observar el claro beneficio que tiene utilizar baterías en un sistema fotovoltaico a largo plazo. (Perfecta Energía, 2024).

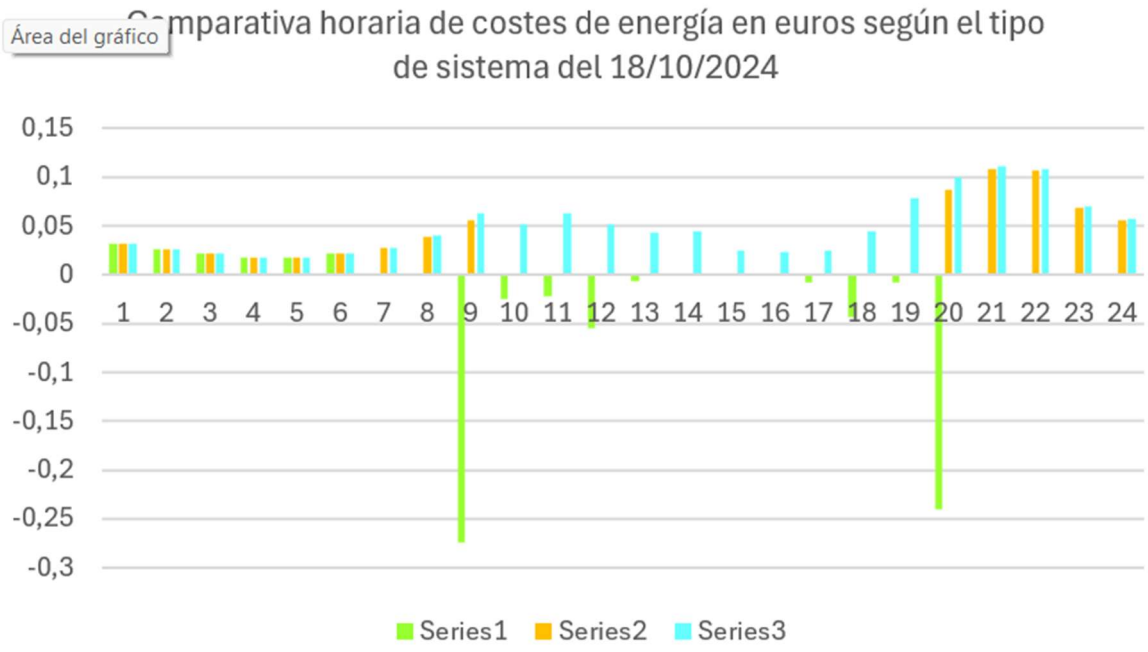


Ilustración 3. Comparativa horaria de costes de energía según el tipo de sistema

Esta gráfica muestra los costes horarios del 18 de octubre de 2024, siendo Series 1 la diferencia entre el coste de energía de un sistema fotovoltaico con batería y el beneficio obtenido por la venta de excedentes, Series 2 el coste de energía de un sistema fotovoltaico sin batería y Series 3 el coste energético sin un sistema fotovoltaico, se puede observar una gran diferencia en los costes.

Se puede observar como el sistema sin batería ni generación fotovoltaica (Series 3) presenta el coste horario más elevado, ya que todo el consumo eléctrico ha de consumirse de la red, ya que no se cuenta con el beneficio del autoconsumo. Esta situación muestra la dependencia energética de la red eléctrica.

La serie 2 muestra una clara diferencia en el coste de la energía con respecto a la serie 1 debido a que la energía generada por el sistema fotovoltaico es directamente consumida. La desventaja de no tener una batería frente a tenerla es que cuando se genera más energía de la que necesitas consumir, se pierde la diferencia, debido a que hay donde almacenarla.

También es importante mencionar, que no se puede seguir una estrategia de gestión para utilizar los excedentes en las horas más necesarias.

Por eso, es la serie 1 la que menos gastos por energía consumida tiene de todos los tipos de sistemas, por todo lo mencionado anteriormente. Puede ocurrir que en determinados momentos el coste de la serie 1 sea superior al de la serie 2, esto es debido al 10% de energía que se pierde al cargar o descargar la batería, entonces para que la lógica funcione se debe cargar un 10% más que si se consumiese directamente para abastecer al consumo únicamente.

Se puede observar momentos en los que la serie 1 tiene costes negativos. Esto indica que el sistema ha vendido excedentes a la red y se ha obtenido un beneficio neto, Estas horas corresponden a periodos de alta generación solar y bajos consumos, sobre todo en las horas centrales del día. Los precios de la energía en el mercado regulado alcanzan sus valores más altos durante las "horas punta", entre las 10:00-14:00 y entre las 18:00-22:00 en días laborales (Selectra, s.f.). Los precios son más elevados en estas franjas horarias, debido a la alta demanda. Es muy relevante que durante estas horas haya una lata generación solar, ya que permite que con el autoconsumo y la energía almacenada en la batería se cubra la mayoría del consumo necesario, demostrando la importancia de los sistemas fotovoltaicos (REE, 2024a).

3.2 REQUISITOS TÉCNICOS

El correcto desarrollo de una herramienta de simulación energética requiere establecer una base técnica sólida que garantice su operatividad, eficiencia y adaptabilidad a distintos contextos. Los requisitos técnicos de este proyecto han sido definidos a partir de las necesidades funcionales de la aplicación.

La aplicación ha sido programada en el lenguaje Java, debido a su estabilidad, versatilidad y compatibilidad con distintos sistemas operativos. Se ha utilizado Apache NetBeans como entorno de desarrollo, ya que es una interfaz intuitiva y valida para principiantes y desarrolladores más avanzados. Apache NetBeans cuenta con autocompletado y refactoring, además de tener depuración eficaz y ser un soporte para múltiples lenguajes. Java es conocida por ser una opción popular para el desarrollo de aplicaciones multiplataforma, permitiendo que el programa pueda ejecutarse en sistemas Windows, macOs y Linux, sin tener que realizar modificaciones. (TOPS Infosolutions, 2023).

Desde el punto de vista del rendimiento, la aplicación debe estar optimizada para gestionar grandes volúmenes de información de forma eficiente. Para realizar un análisis anual, se requiere procesar 8.760 registros por variable, por lo tanto, se requiere utilizar estructuras de datos eficientes como arrays dinámicos, listas o mapas.

3.3 ARQUITECTURA GENERAL DEL SISTEMA

El sistema ha sido diseñado bajo una estructura modular y diseñado en clases, es recomendado por la literatura técnica en desarrollo de software orientado a objetos De esta forma se ha mantenido una organización clara y coherente que facilita el desarrollo del programa y el mantenimiento del código. Las partes más significativas de la lógica están separadas por clases y luego dentro de ellas por funciones, esto permite que haya una división clara de responsabilidades y las funciones se puedan utilizar en varios lugares, y sobre todo que haya un orden y mayor legibilidad (Innowise, 2023).

En primer lugar, se ha desarrollado una clase encargada exclusivamente de leer archivos CSV que contiene los datos de entrada. Esta clase recorre los CSV con los datos de entrada y los importar correctamente después de haber sido modificado, a otra clase en la que los almacena en una estructura de datos para poder trabajar con ellos. (Perfecta Energía, 2024)

A continuación, existen clases individuales para todos los datos exportados. Esto se debe a que antes de poder almacenarlos es necesario realizar ciertos ajustes para que correspondan con los datos que queremos. Por ejemplo, el perfil de consumo eléctrico debe convertirse a valores absolutos en función del porcentaje del consumo anual, mientras que la generación solar debe ajustarse al tamaño instalación simulada, estos cálculos se verán más en profundidad en otro capítulo. Las clases de los precios de red y de venta de excedentes se centran en almacenar en otra estructura los precios de mayor a menor y asignarles un grupo, alto, medio o bajo, dependiendo en donde se encuentren en esta lista de precios de orden descendiente (NREL, s.f.). Esto es parte de la lógica detrás del código que se explicará en detalle más adelante.

El núcleo del sistema se encuentra en una clase principal, que contiene la lógica general del algoritmo de simulación. En esta clase se llama a diferentes funciones auxiliares, que están creadas dentro de ella, que calculan el estado de carga de la batería, la energía que se vende a la red o la que se carga. Dentro de estas funciones se determina también que acción ha de realizar la batería, si cargarse, descargarse, no hacer nada, y de qué forma. Este enfoque estructurado facilita la trazabilidad de las decisiones que toma el sistema en cada hora de simulación y permite encontrar errores de una forma más fácil.

Por último, se ha creado una clase final que se encarga de escribir el archivo CSV de salida, donde se recogen todos los resultados de la simulación y todos los cálculos para poder realizar las comparaciones necesarias y obtener los resultados indicados. Este CSV puede ser exportado para generar gráficas y realizar análisis adicionales sobre el comportamiento del sistema.

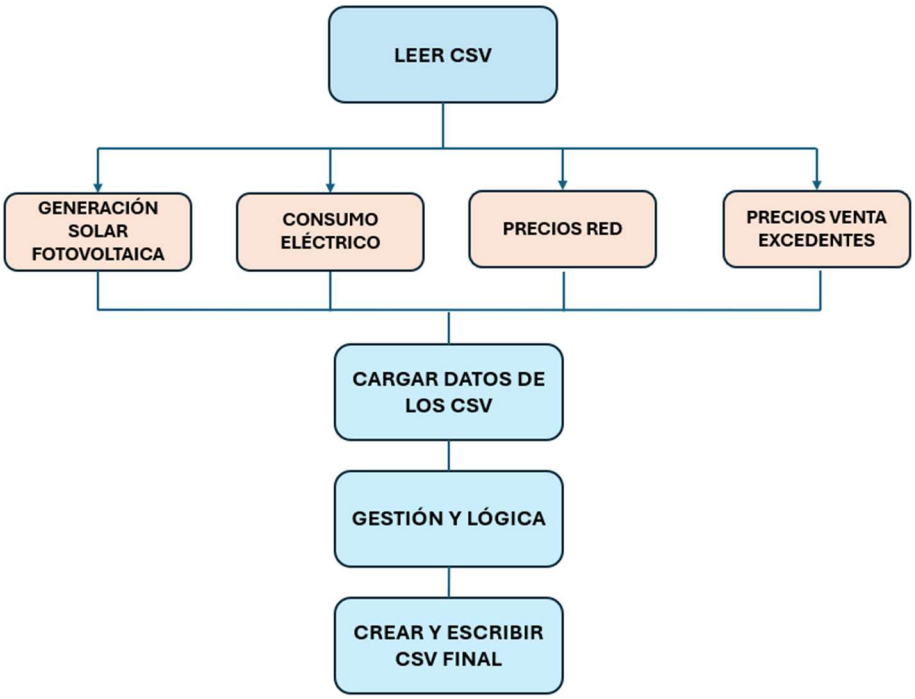


Ilustración 4. Esquema general de la arquitectura del sistema

En la ilustración 3 se muestra un esquema general de la arquitectura del sistema, donde se representan los módulos descritos y su interacción en el programa.

3.4 DISEÑO MODULAR Y LÓGICA GENERAL

La lógica de funcionamiento de este sistema se ha diseñado para que la batería se gestione de forma eficiente y autónoma, priorizando el autoconsumo, minimizando la dependencia con la red eléctrica y optimizando el rendimiento económico en función de los precios horarios. Para ello se han establecido unas condicione que determinan la toma de decisiones en base a los datos reales (EDP Energía, 2023)

En primer lugar, como se ha mencionado antes, se recopila y organiza la información necesaria para cada hora: consumo eléctrico, generación solar fotovoltaica, precio de la electricidad en la red y el precio de venta de excedentes. También hay datos que se le piden al usuario, como la capacidad nominal de la batería y el número de paneles en su instalación que son esenciales para realizar los cálculos del código. Con estos datos lo primero que se hace tras haberlos modificado o ajustado es calcular el balance energético, es decir la diferencia entre el consumo y la generación. Este parámetro, que llamamos diferencia, permite determinar en cada hora si se encuentra en una situación de excedente (generación>consumo) o de déficit (consumo>generación). Esto marca el punto de partida para la lógica (EcoFlow, 2023).

Cuando el consumo eléctrico supera la generación solar fotovoltaica, la prioridad es aprovechar al máximo el autoconsumo. Por tanto, toda la energía generada se utiliza para abastecer al consumo. Si esta energía no es suficiente, se estudia si la batería puede suministrar el resto de la energía necesaria. Sin embargo, esta decisión ha de tomarse con criterio, ya que se incorpora un análisis del precio horario de la red: si el precio es muy bajo es interesante consumir de la red y almacenar la energía para horas con precio de red más alto (PV Magazine España, 2025; REE, 2024a).

Para poder realizar este análisis de precios, se clasifican previamente en tres grupos, alto, medio y bajo, ordenados de mayor a menor. De esta forma se puede realizar comprobaciones para optimizar la carga y descarga de la batería, y tomar la decisión correcta sobre cuando almacenar la energía en la batería. Esta segmentación de precios se utiliza en sistemas reales de gestión de energía doméstica, como los que se analizan en estudios recientes de autoconsumo inteligente (Solarenergía Familias, 2024).

La lógica es diferente en el caso de déficit y de exceso. Para evitar recirculación de energía de la red, optimizar los recursos y mejorar la eficiencia global del sistema, en el momento de transición se realiza un ajuste en el estado de la batería (Rabobank,2025). En el caso en el que se cambia de consumo > generación a generación > consumo el estado de la batería se iguala al 20% de su capacidad nominal, y se ajustan los estados de batería anteriores para que haya coherencia. En el caso opuesto, habrá ocasiones en las que lo óptimo y razonable sea dejar el estado de la batería al 80% de su capacidad y en otras en las que no se modifique.

De manera similar, cuando la generación fotovoltaica excede el consumo eléctrico, la energía sobrante carga la batería, siempre que no supere el 80% de la capacidad nominal de la batería, el umbral máximo de carga. Si esto ocurriese, se buscaría la hora adecuada para vender esa energía sobrante, para obtener un beneficio económico. (Solarenergía Familia, 2023).

3.5 JUSTIFICACIÓN DE VALORES Y UMBRALES

Para garantizar la durabilidad y el correcto funcionamiento del sistema de almacenamiento, se han establecido límites de operación en el estado de carga (SoC)**,** fijando un valor mínimo del 20 % y un valor máximo del 80 % de la capacidad nominal. Estos márgenes se definen con el objetivo de proteger la batería frente a ciclos de carga y descarga completos, que aceleran su degradación debido a reacciones químicas irreversibles y pérdida de capacidad útil (Mathews et al., 2020). Operar fuera de estos rangos puede incrementar la temperatura interna, provocar sobrecargas o sobredescargas y, en consecuencia, reducir significativamente la vida útil del acumulador.

Las condiciones de funcionamiento recomendadas para este tipo de sistemas incluyen mantener la batería dentro de un rango térmico óptimo (normalmente entre 15 °C y 30 °C para tecnologías de ion-litio), evitar corrientes de carga y descarga superiores a las especificadas por el fabricante, y limitar la profundidad de descarga (DoD) según el diseño del sistema. Además, se recomienda realizar una gestión activa mediante un sistema de **ge**stión de batería (BMS) que monitorice tensión, corriente y temperatura, asegurando que los parámetros se mantengan dentro de los límites establecidos (MITECO, 2024).

Aunque la eficiencia de la batería ya ha sido detallada previamente, cabe recordar que en sistemas de ion-litio esta se sitúa típicamente entre el 90 % y el 95 % en condiciones nominales. Por ello, cualquier estrategia de gestión debe tener en cuenta las pérdidas por conversión y autoconsumo, ya que influyen directamente en el balance energético global del sistema (Mathews et al., 2020).

3.6 CONSIDERACIONES DE ESCALABILIDAD Y MEJORAS FUTURAS

El sistema ha sido diseñado con una estructura modular, dándole una elevada escalabilidad, entendida como la capacidad de una aplicación para adaptarse a un aumento de la carga de trabajo o del volumen de datos y poder expandirse de manera funcional, sin perder rendimiento. La lógica permite realizar simulaciones desde un solo día hasta un número indefinido de días, según lo desee el usuario, simplemente aumentando los datos de entrada en los archivos CSV. Esto proporciona una flexibilidad que permite una fácil adaptación del sistema a distintos hogares, periodos de análisis o escenarios energéticos, sin

A pesar de que la aplicación ya proporciona una herramienta útil para simular el comportamiento de un sistema fotovoltaico con batería, existen diversas áreas en las que el sistema puede ser ampliado o mejorado en el futuro, tanto para aumentar su realismo como para maximizar su valor práctico.

La primera mejora consistiría en la integración de predicción dinámica de datos, en función de la ubicación geográfica del usuario. En esta versión, los datos exportados se introducen de forma estática. Sin embargo, sería deseable emplear algoritmos de predicción o bases de datos meteorológicas y tarifarias por región para anticipar la producción solar o las tarifas eléctricas, y así poder tomar las decisiones con antelación sobre el uso de la batería (Hernández et al., 2022). Este enfoque, cada vez más común en los sistemas de gestión energética avanzados, permitiría mejorar la eficiencia del sistema y adaptarlo a distintas ubicaciones sin necesidad de parametrización manual.

Una segunda mejora es incluir el fenómeno de autodescarga de las baterías. Las baterías de ion-litio, aunque son muy eficientes, pierden gradualmente parte de su carga incluso cuando no están en funcionamiento, lo que se conoce como autodescarga. Esta pérdida es aproximadamente del 1 % al 2 % mensual (Battery University, sf), y puede afectar a la capacidad útil a lo largo del año. Por tanto, sería adecuado incorporar esta variable en los cálculos para estimar con mayor precisión la disponibilidad real de energía almacenada y así ser más realistas y conscientes del funcionamiento de las baterías.

Asimismo, actualmente el modelo asume que todas las cargas y descargas se completan dentro de cada intervalo horario. Esta simplificación ignora que las baterías necesitan un tiempo determinado para cargarse y descargarse, dependiendo de su potencia nominal. Por ejemplo, cargar completamente una batería de ion-litio desde cero puede tardar entre 2 y 4 horas dependiendo de cómo se cargue y la capacidad. En futuras versiones, se podría modelar este proceso en mayor detalle para que el análisis sea más realista, dividiendo las ventanas horarias o introduciendo tasas de carga y descarga parciales.

También habría que incluir el envejecimiento progresivo de la batería, siendo representado a través de una variable de "estado de salud" (SoH, por sus siglas en inglés). A lo largo de los ciclos de carga y descarga, la capacidad nominal de la batería se reduce de forma paulatina, afectando directamente a su rendimiento a medio y largo plazo. En la versión actual se asume una capacidad constante, incluir una tasa de degradación permitiría realizar simulaciones más precisas y útiles para el análisis de inversiones a largo plazo.

Otra posible mejora sería permitir la gestión multiusuario o multiinstalación. Actualmente, la aplicación está diseñada para un solo perfil doméstico, pero en entornos residenciales compartidos (como comunidades de vecinos o autoconsumo colectivo), sería útil escalar el sistema para que gestione múltiples flujos energéticos de forma coordinada. Esta idea se alinea con los avances legislativos que promueven el autoconsumo colectivo en España bajo el Real Decreto 244/2019.

Algunas de estas mejoras se han de realizar debido a que se han simplificado varios detalles de la lógica para facilitar la programación, pero el próximo paso sería implementar estas mejoras y realizar una aplicación más completa para seguir ayudando a los usuarios a alcanzar sus objetivos de reducir los costes y de ser independientes de la red, aunque esta versión también proporciona esta solución.

Capítulo 4. LOGÍSTICA DEL SISTEMA DE GESTIÓN ENERGÉTICA

4.1 INTRODUCCIÓN A LA LÓGICA DE CONTROL ENERGÉTICO

4.2 CASO 1: CONSUMO MAYOR QUE GENERACIÓN

4.2.1 LÓGICA GENERAL

En esta parte de la lógica se estudian los momentos en los que el consumo energético del sistema supera la generación solar fotovoltaica disponible, generando un déficit energético. Ante esta situación, el sistema debe decidir cómo cubrir dicha diferencia, priorizando el uso eficiente de la batería y teniendo en consideración el precio horario de la red eléctrica.

Cuando se llama a la función que estudia las horas en las que hay un déficit energético, lo primero que se hace es reiniciar las variables necesarias para el análisis de la hora actual y se inicializan las variables necesarias para los bucles y los cálculos. Lo siguiente es identificar a que grupo de precios pertenece el precio de la red en esa hora (bajo, medio o alto). Este análisis se realiza mediante una estructura de tipo switch, separando en casos cada situación para poder aplicar una estrategia según el grupo de precios.

4.2.1.1 Caso de precio bajo

Cuando el precio de la red es bajo, se cubre toda la diferencia energética directamente desde la red, sin descargar la batería. Esto responde a una estrategia de optimización de costes, reservando la batería para momentos donde el precio de la red sea más elevado. Se ha decidido tomar las decisiones según los precios de la red, para cumplir uno de nuestros objetivos: reducir los costes.

4.2.1.2 Caso de precio medio

En este caso, debido a que tomar decisión basada en el precio puede ser subjetivo, el sistema evalúa si la batería puede cubrir la diferencia sin comprometer su estado mínimo de carga (SoC), que por diseño se ha fijado en el 20 % de la capacidad nominal. Si es así, toda la diferencia se cubre desde la batería. En caso contrario, se utiliza únicamente la energía

disponible entre el estado actual de carga y el 20 %, y el resto se cubre desde la red. Se calculan y registran tanto el consumo energético de la red como el correspondiente coste horario. De este modo, se utiliza la batería siempre que sea posible para no consumir toda la energía de la red, y así poder reducir los costes, pero si no fuese posible, debido a que no es un precio de red alto, es preferible que sea en este caso en el que se consuma de la red y no en uno con el precio de red más elevado.

4.2.1.3 Caso de precio alto

Cuando el precio de la red es alto, se aplica una lógica más compleja debido a que todo lo que no pueda ser compensado con el autoconsumo ha de consumirse de la batería, para no consumir de la red con un precio alto. Si la batería puede cubrir completamente el déficit sin bajar del SoC mínimo, entonces toda esta energía es consumida de la batería. Si esto no es posible, primero se averigua si la función de consumoMayorGeneración, ha sido llamada desde la función principal, se hace mediante un boolean, si es que si, se inicia una búsqueda hacia atrás (mediante un bucle tipo while) para localizar la hora anterior más adecuada para cargar la batería, y llegar a la hora de precio red alto con la batería suficiente para poder abastecer el consumo.

El primer paso para buscar la hora en la que se ha de cargar la batería es encontrar el rango de búsqueda. Para ello se realiza un if que se salga en el momento, que al retroceder se encuentre que hay una transición a un estado de generación mayor que consumo, que el estado de la batería ha alcanzado el 80% de su capacidad nominal, siendo este su estado máximo de carga o que se encuentre una hora en la que ya se haya cargado de la red. La elección de estos momentos es debido a que cada estado tiene su propia lógica y está programado para que funcionen por separado, de esta forma es mucho más sencillo y práctico gestionar los estados de carga. En el momento en el que la batería alcanza un 80% de su capacidad nominal en el estado de consumo mayor que generación es porque ha habido una carga y si ahora estamos retrocediendo para realizar otra, significa que esa carga anterior se ha realizado con criterio y necesidad, por eso otra razón para salir del if es si se encuentra una hora en la que se ha realizado una carga.

Una vez encontrado el límite inferior del rango, se empezará desde el límite superior y se retrocederá hasta el final para obtener la hora con el precio de la red más bajo, es decir, el mejor momento para cargar la red. La batería se cargará al máximo a esa hora para poder abastecer al resto de horas que lo necesiten y llegar a la hora con precio de red alto con suficiente energía para todo el consumo que falte. Más adelante se realizarán los ajustes necesarios para que se utilice la energía necesaria y no haya que vender energía a la red y haya una recirculación de energía (.Solarenergía Familia, 2023)

Si la función de consumoMayorGeneración, no ha sido llamada desde la función principal, entonces s ehacen otros cálculos, esto es así debido a que si no se generaban bucles infinitos y había que simplificar los cálculos. En este caso, se opta por descargar la batería hasta el umbral del 20 % de la capacidad nominal y cubrir el resto de la demanda con energía de la red, registrando tanto el consumo como el coste asociado.

Cuando se han realizado los cálculos en el caso correcto, se avanza a la siguiente hora en la lógica principal para ver si esta hora pertenece al estado de generación mayor que consumo o si pertenece a este estad, y así llamar a la función correcta.

4.2.2 AJUSTE ÚLTIMA HORA

Cuando en el estado de consumo mayor que generación se alcanza la última hora, se llama a la función AjustarBateriaConsMayorGen con el objetivo de preparar al sistema para una transición al estado con excedente energético. Debido a que durante el día ocurren varios escenarios de transición, se quiere lograr la optimización y mayor provecho de la energía, por eso se han creado estas lógicas de ajuste para las últimas horas de cada estado. La estrategia en esta lógica consiste en forzar el estado de la batería de esta última hora a alcanzar su mínimo operativo (20 % de la capacidad nominal). De este modo, se maximiza la capacidad disponible para almacenar energía cuando la generación vuelve a superar el consumo (Namor et al., 2018).

Inicialmente, se comprueba si el estado de la batería en esta última hora es superior al 20%. Si se cumple esta condición, se fuerza el SoC al 20% de la capacidad nominal. Esta modificación, requiere ajustar retrospectivamente los valores de la batería en las horas anteriores, de modo que se mantenga la coherencia en el sistema.

Para ello, se define un rango de operación mediante un bucle while que retrocede en el tiempo mientras se mantenga la condición de déficit energético (consumo > generación) y que el contador este dentro del registro de datos. Dentro de este rango, se buscan dos posibles puntos de interés.

- Una hora anterior donde se haya realizado una carga de batería
- La hora con el precio de venta más alto del rango, si no hay carga registrada

Si se encuentra una hora donde se haya realizado una carga anterior prevalece sobre el precio de venta más alto. La razón es porque probablemente en esa hora de carga se haya cargado de más y por eso se llega a la última hora con una carga superior a la que necesitamos. En esta lógica predomina la mejora en los casos de carga y descarga antes que vender energía, y así reducimos la recirculación de energía (Aurora Solar, 2025).

Una vez identificado el punto de corte, se recalculan los estados de la batería de forma progresiva, dependiendo del precio de la red de cada hora, como en la función ConsumoMayorGeneracion.

- Si el precio de la red es bajo, se asume que toda la diferencia de energía se consume directamente de la red, manteniendo constante el SoC.

- Si el precio es medio o alto, se actualiza el estado anterior como la suma del estado actual y la diferencia de esa hora, asumiendo que la energía procede de la batería

De esta forma en los casos con precio red alto y medio consumimos la diferencia de la batería, ya que vamos a ajustar todos los estados de la batería anteriores para obtener el mayor beneficio y consumir menos de la red.

Al llegar a la hora seleccionada para ajustar la energía sobrante, se evalúa si se debe vender o cargar. Esto se determina comparando el estado anterior con el nuevo valor del estado de la batería calculado.

- Si el estado anterior es mayor, la diferencia se considera energía sobrante que se vende, y se registra como beneficio. Siendo la energía que se vende la diferencia entre el estado de la batería anterior y el estado del batería calculado.

- Si el estado anterior es menor, se interpreta como necesidad de energía adicional, y se carga la batería. La energía que se carga es la diferencia entre el estado de la batería calculado y el estado anterior. Si estamos en la primera hora de la tabla de registro, el estado anterior = 20% capacidad nominal.

En el caso de que el estado de carga (SoC) de la batería en la última hora del intervalo con déficit energético no sea superior al 20 % de su capacidad nominal, no es necesario forzar su ajuste.

Puede darse la situación de que, al recalcular de forma regresiva los estados anteriores de la batería, el proceso haya retrocedido más allá del instante inicial con el que se entró a esta función. Es decir, si el contador actual es menor que el contador original de entrada es necesario rehacer los cálculos hacia adelante para garantizar la coherencia de los datos y el correcto encadenamiento temporal de los estados.

Para ello, se vuelve a llamar recursivamente a la ConsumoMayorGeneracion, que recalcula todos los valores desde el punto al que se ha retrocedido hasta el instante de entrada original. En esta llamada, se incluye un parámetro que indica que no se trata de una llamada desde el bucle principal, con el objetivo de evitar bucles infinitos. Esto es posible gracias al parámetro retroceder, que se utiliza internamente en el código para distinguir entre una llamada desde la función principal y una llamada desde el ajuste.

4.3 CASO 2: GENERACIÓN MAYOR QUE CONSUMO

4.3.1 LÓGICA GENERAL

En esta parte de la lógica, se estudian las horas en las que la generación solar fotovoltaica supera al consumo eléctrico, es decir cuando la diferencia es positiva. En esta lógica se prioriza cargar la batería lo máximo posible, pero cuando no se pueda, buscar la solución óptima para poder seguir cumpliendo los objetivos del programa.

En primer lugar, se inicializan las variables y se ponen a cero cualquier valor que deba calcularse. A continuación, se evalúa si el SoC anterior más la diferencia actual es menor o igual al 80% de la Capacidad Nominal. Si se cumple esta condición, se actualiza el estado de la batería.

Cuando la generación fotovoltaica supera el consumo eléctrico en una determinada hora, es decir, cuando la diferencia entre ambos es positiva, se activa la lógica de "generación mayor que consumo". En primer lugar, se inicializan las variables necesarias y se pone a cero cualquier valor que se deba calcular en esa iteración. A continuación, se evalúa si el estado de carga (SoC) anterior más la diferencia actual es menor o igual al 80 % de la capacidad nominal de la batería. Si se cumple esta condición, se actualiza el estado de la batería sumando la diferencia a su estado anterior, ya que existe capacidad suficiente para almacenar el excedente.

Después de este cálculo, se evalúa si ya existe una variable llamada energíaVentaFutura, que representa energía previamente planificada para su venta. Si esta variable es mayor que cero, se comprueba si el estado de la batería menos esa energía futura es superior al 20 % de la capacidad nominal. Si lo es, se procede a vender dicha energía directamente. Si no se cumple esa condición, se calcula la energía que se puede vender sin que el SoC de la batería caiga por debajo del 20 %. Esta energía se calcula restando ese 20 % al estado actual. Se actualiza el estado de la batería y se conserva el valor restante como energíaVentaFutura para su venta en una hora posterior.

Si, por el contrario, el estado de carga más la diferencia supera el 80 % de capacidad, la batería no puede absorber toda la energía. En este caso, si se está ejecutando desde el bucle principal, se entra en el modo de "retroceso". En este modo, la energía que no puede almacenarse se calcula como el exceso sobre el 80 % y se guarda como energíaVentaFutura. A partir de aquí, se busca la mejor hora futura para vender dicha energía. Para ello, se recorre hacia adelante un rango de horas mientras se mantengan ciertas condiciones: que la diferencia siga siendo positiva, que el beneficio de esa hora sea cero y que aún no se haya alcanzado el final del conjunto de datos.

Una vez delimitado el rango de búsqueda, se analiza dentro de ese rango en qué hora el precio de venta de la energía es más alto. Si esa hora tiene un precio alto, se calcula la energía que puede venderse dejando la batería a un estado mínimo del 50 % de su capacidad, como medida de seguridad. Se actualiza el estado de la batería y se calcula el beneficio correspondiente a esa venta. En caso de que el precio no sea alto (es decir, sea medio o bajo), se evalúa si puede venderse toda la energíaVentaFutura manteniendo al menos un 20 % de carga. Si es posible, se realiza la venta. Si no, se vende únicamente lo que se pueda sin bajar del 20 % del SoC, e incluso se puede optar por ir vendiendo poco a poco en distintas horas sucesivas de precio más alto.

Por último, si no se está en modo retroceso (es decir, si no se ha llamado esta lógica desde el bucle principal), se entiende que se está en un caso de ajuste puntual y, por tanto, se vende toda la energía sobrante en la hora actual, sin buscar un mejor precio en horas futuras. Esto se debe a que retroceder y recalcular en este punto podría descompensar los estados previos ya calculados, y como estos rangos de ajuste suelen ser reducidos, se considera razonable vender de forma inmediata.

En el caso en que la suma del estado anterior de la batería y la diferencia de generación-consumo sea menor o igual al 80 % de la capacidad nominal, se interpreta que la batería puede seguir cargándose sin exceder su límite superior. En este punto se comprueba si existe una variable energíaVentaFutura con valor mayor que cero, lo cual ocurre principalmente cuando se ha ejecutado previamente la lógica de retroceso (es decir, cuando venimos del bucle principal y se ha intentado vender una energía excedente no almacenable). Esa energíaVentaFutura se define originalmente como el exceso de energía que no puede cargarse en la batería, calculado como estado anterior + diferencia - 80 % de la capacidad nominal.

Cuando entramos en esta lógica, se intenta vender dicha energía excedente. Si el estado actual de la batería menos la energíaVentaFutura es mayor o igual al 20 % de la capacidad nominal, se procede a vender toda esa energía. Sin embargo, si no es posible venderla completamente sin reducir el estado de la batería por debajo de ese 20 %, solo se vende la cantidad permitida hasta ese umbral, y el resto se mantiene almacenado en la variable energíaVentaFutura. En ese caso, se actualiza esta variable restándole la cantidad efectivamente vendida.

Así, cuando se vuelve al bucle principal y más adelante se vuelve a activar la lógica de generación mayor que consumo, se detecta nuevamente que la energíaVentaFutura tiene un valor distinto de cero, por lo que se entra otra vez en el if (energíaVentaFutura > 0) para continuar vendiendo esa energía excedente en las siguientes horas, priorizando aquellas en las que el precio de venta sea más alto y las condiciones de seguridad de la batería lo permitan.

4.3.2 AJUSTE ÚLTIMA HORA

La función AjustarBateriaGenMayorCons se llama en la última hora del estado en el que la generación fotovoltaica supera al consumo. Su objetivo es corregir retrospectivamente el estado de la batería y los flujos económicos asociados (energía vendida y beneficio) cuando se identifica un patrón operativo relevante: un ciclo previo en el que la batería alcanza el 80 % de su capacidad nominal y que, tras un periodo de descarga, se producen excedentes de generación.

La lógica comienza fijando el instante actual y retrocediendo en el histórico de registros hasta localizar un punto que cumpla dos condiciones: diferencia positiva (generación mayor que consumo) y estado de batería inferior al 80 % de la capacidad nominal. Este retroceso se detiene si se encuentra exactamente un estado del 80 % o si se alcanza el inicio del registro. El caso del 80 % se interpreta como marcador de un ciclo de carga completo previo que, en escenarios con excedentes, suele ir seguido de periodos de descarga y, potencialmente, de ventas a la red. Esta señal permite delimitar coherentemente el comienzo del tramo que se quiere revisar.

Una vez identificado el 80 %, el algoritmo avanza desde ese punto hasta localizar el instante en el que la batería desciende por debajo del 20 % de su capacidad nominal. Este avance define el tramo principal de descarga (80 % → 20 %). En el instante en que se cruza el umbral del 20 %, se corrige el estado de la batería igualándolo al del registro inmediatamente anterior, evitando así descensos inconsistentes o artefactos numéricos. A continuación, se actualizan los flujos económicos de ese instante: la energía vendida se calcula como la diferencia absoluta entre generación y consumo dividida por 0,9 (asumiendo una eficiencia del 90 %), y el beneficio se obtiene multiplicando dicha energía equivalente por el precio de venta vigente.

Estas correcciones pueden afectar a la coherencia temporal de los estados posteriores. Por ello, si la modificación se produce en un instante anterior al actual, el método fuerza un recálculo secuencial de todos los registros comprendidos entre el punto corregido y el instante presente. Este recálculo garantiza la consistencia dinámica del sistema, propagando los efectos de la corrección a lo largo de la serie temporal para mantener la trazabilidad de estados, flujos de energía y resultados económicos.

4.4 CASO 3 GENERACIÓN IGUAL QUE CONSUMO

En el caso, poco probable, de que durante la ejecución de la lógica de control se produzca una hora en la que la generación fotovoltaica sea exactamente igual al consumo, no se realiza ninguna acción sobre la batería. En esta situación, el sistema establece que el estado de carga de la batería sea exactamente el mismo que el registrado en la hora anterior y procede directamente al análisis de la siguiente hora.

Esta decisión se debe a que, al no existir excedente ni déficit energético, la batería no necesita intervenir ni en modo de carga ni de descarga. De este modo, se evita cualquier operación innecesaria que pudiera generar pérdidas asociadas a la eficiencia de conversión, manteniendo la estabilidad del sistema y optimizando su vida útil.

4.5 *DIAGRAMAS DE FLUJO*

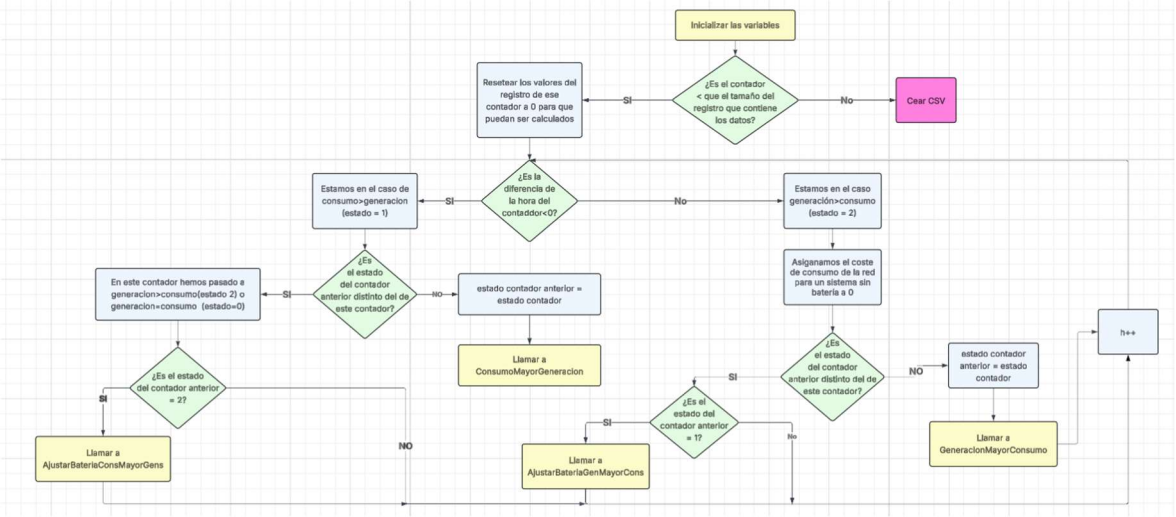


Ilustración 5. Diagrama de flujo de la lógica principal

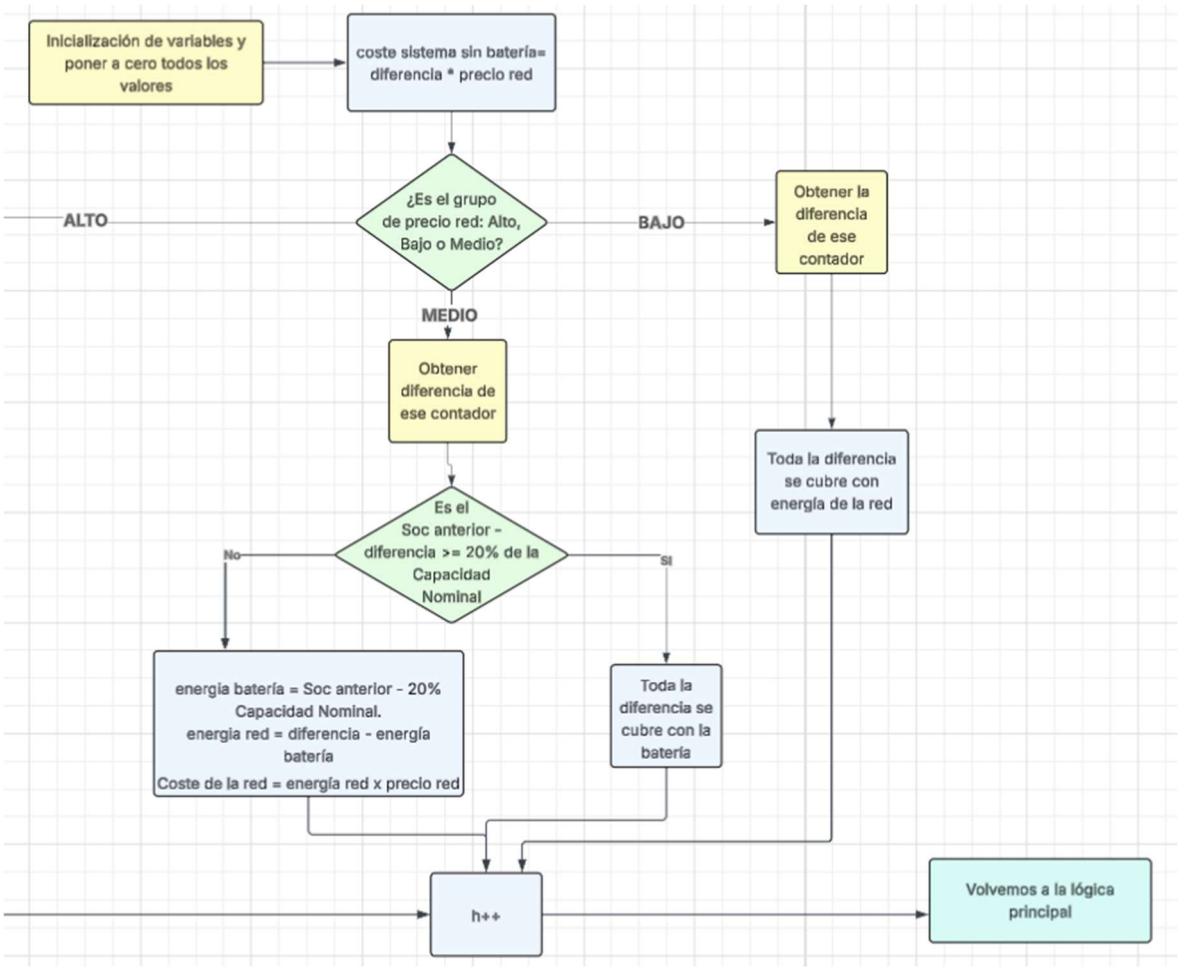


Ilustración 6. Diagrama de flujo de la función de ConsumoMayorGeneraciion de los casos de precio medio y bajo

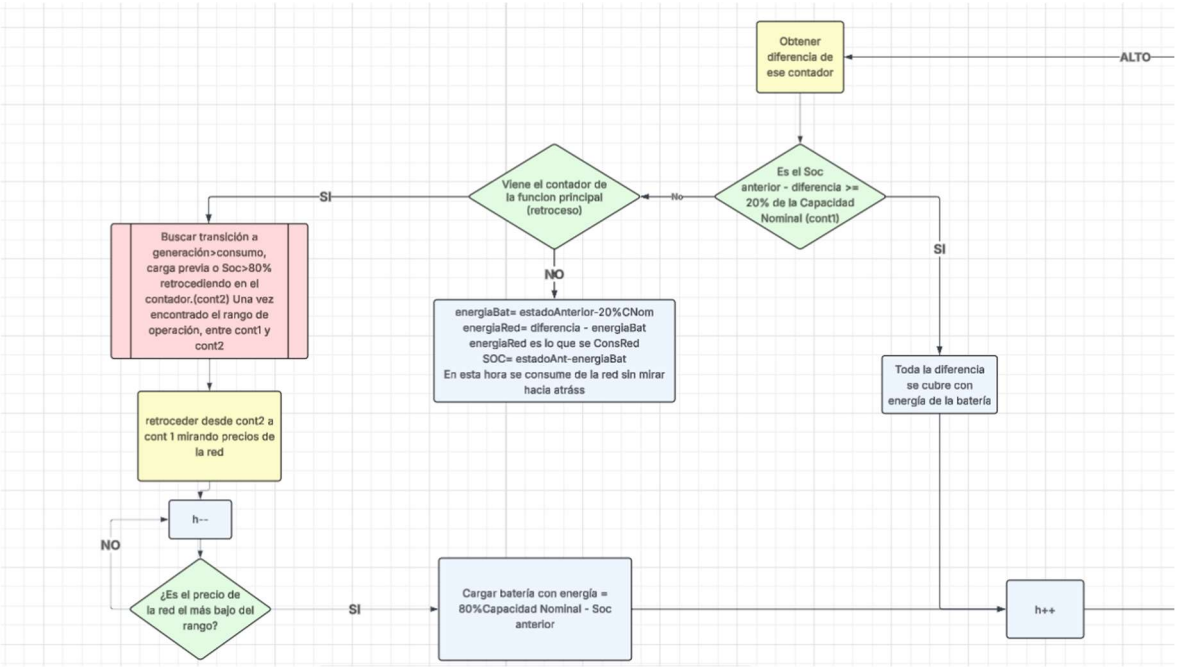


Ilustración 7. Diagrama de flujo de la función de ConsumoMayorGeneraciion del caso de precio alto

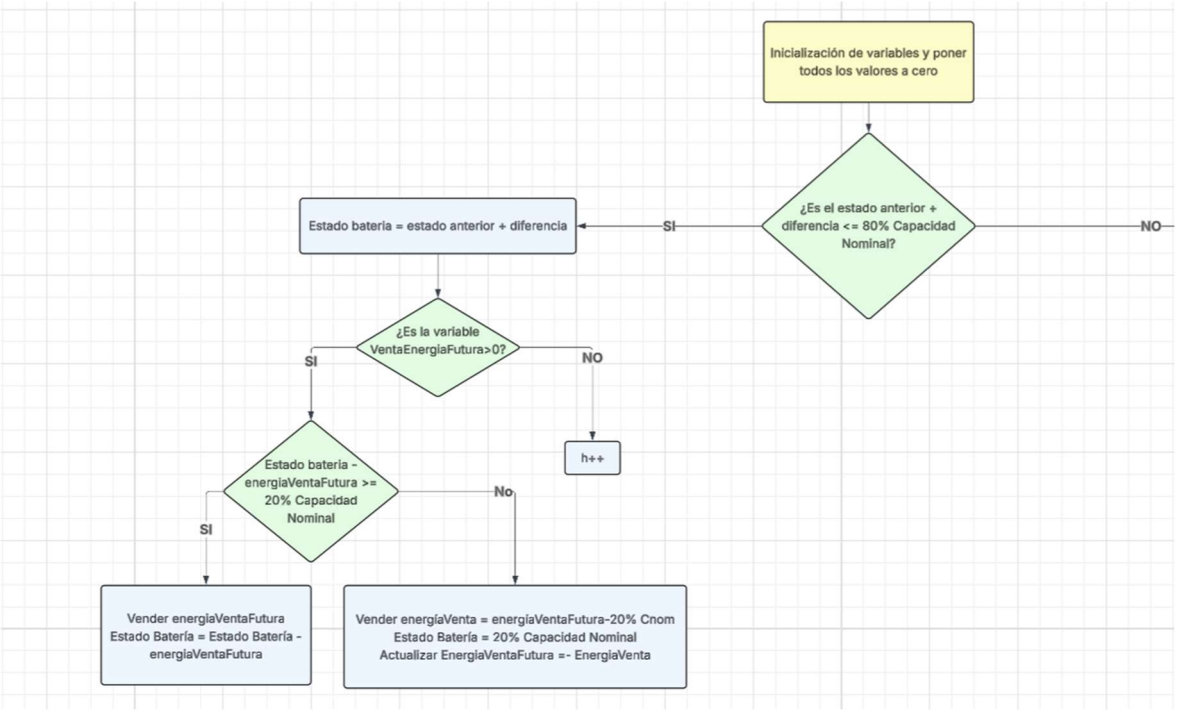


Ilustración 8. Primera parte del diagrama de flujo de la función GeneracionMayorConsumo

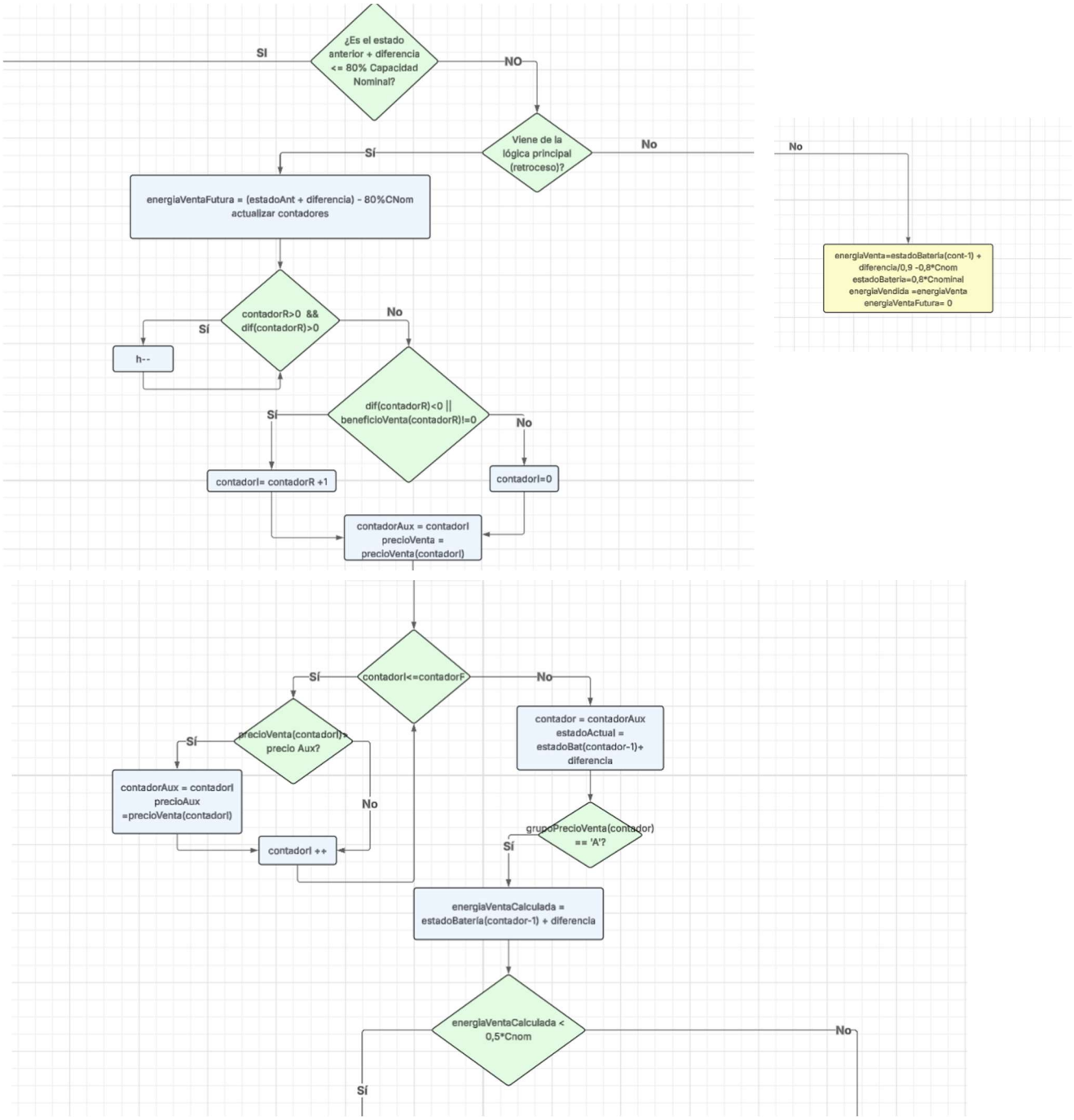


Ilustración 9. Parte I, II y III siguientes partes del diagrama de flujo de la función GeneracionMayorConsumo

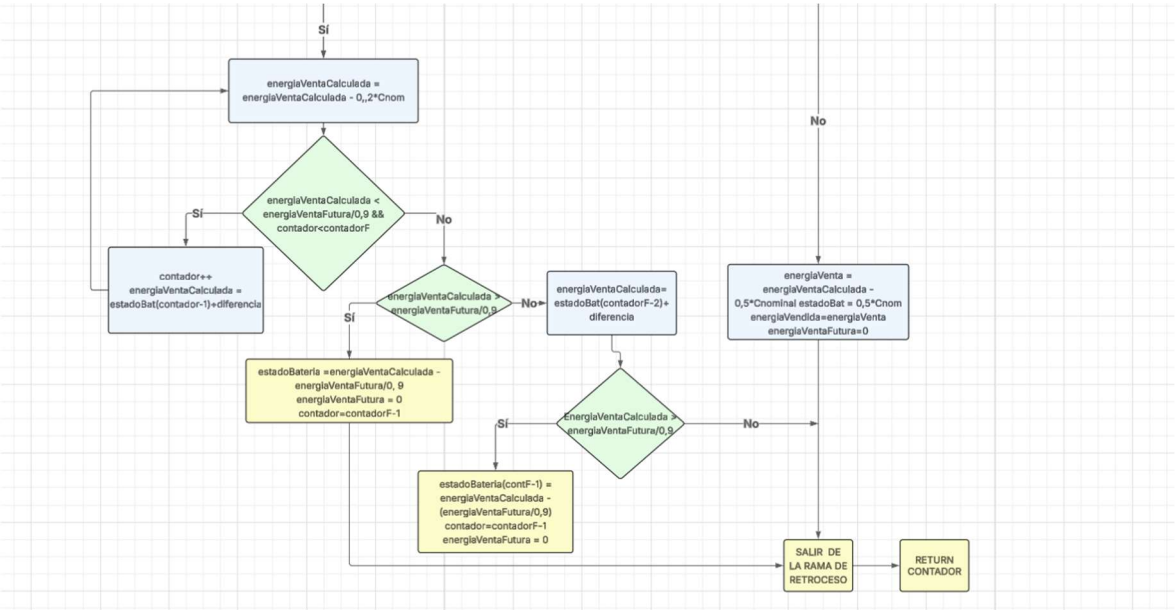


Ilustración 10. Última parte del diagrama de flujo de la función GeneracionMayorConsumo

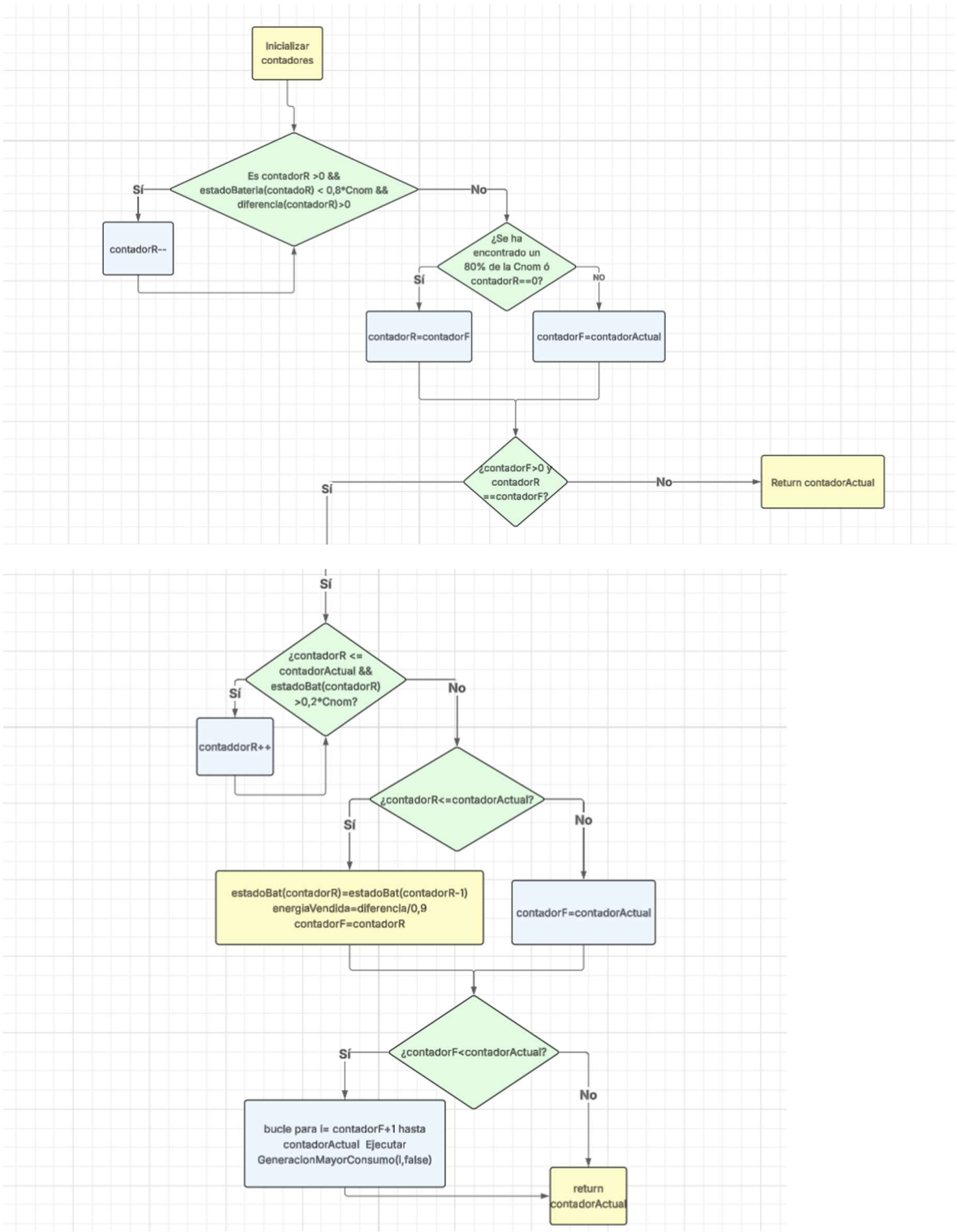


Ilustración 11. Parte I y II del Diagrama de flujo de la función AjustarBateriaGenMayorCons

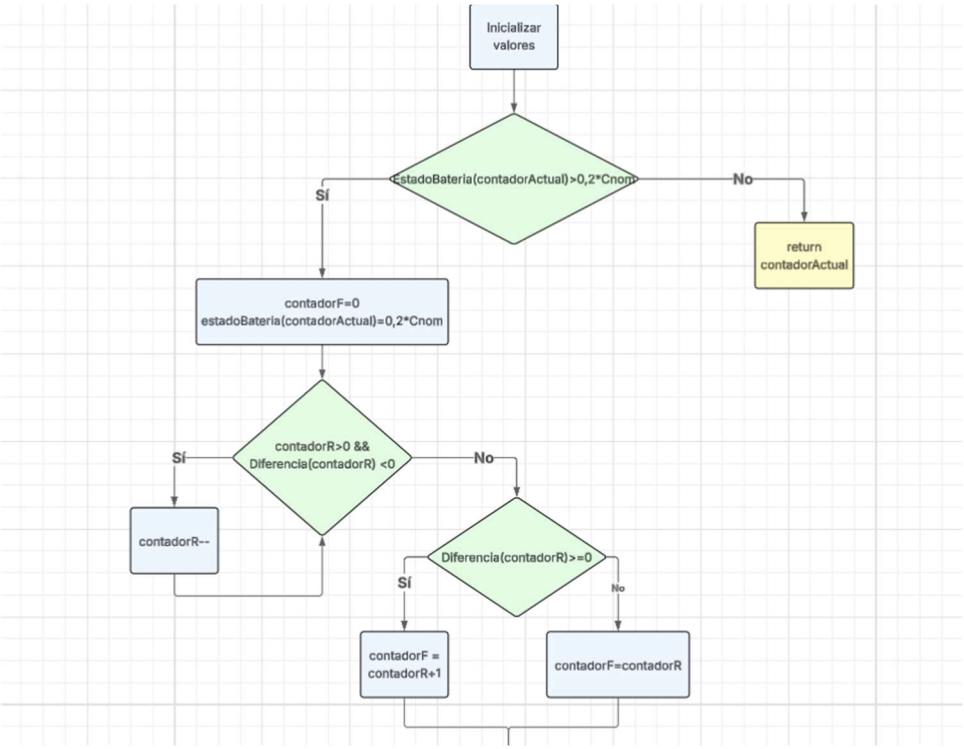


Ilustración 12. Parte I de la función de AjustarBatGenMayorCons

Ilustración

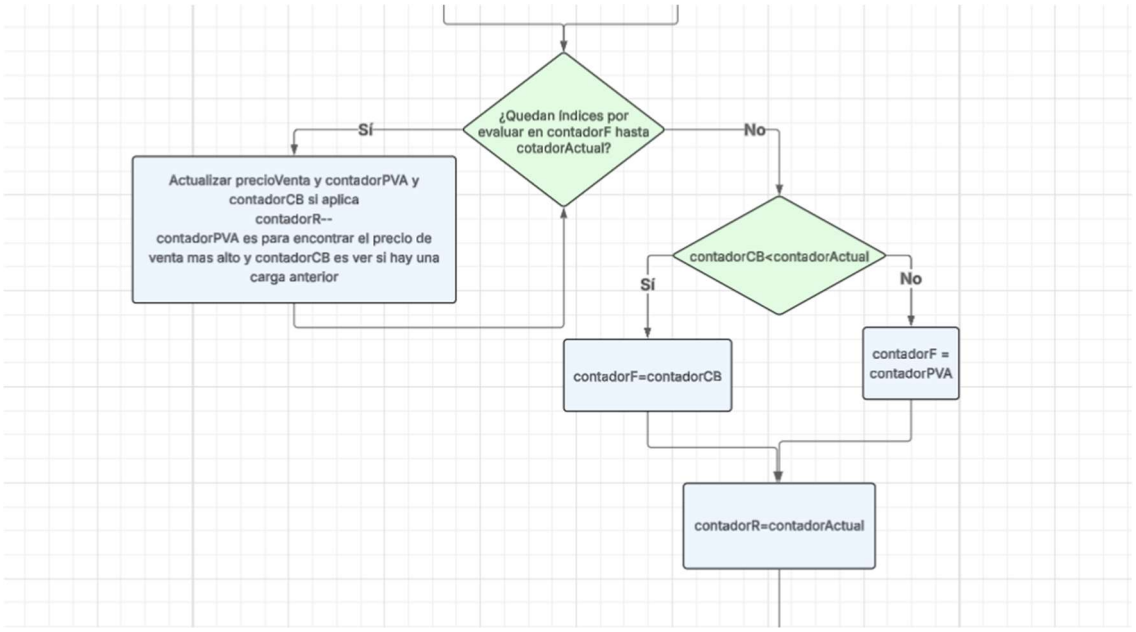


Ilustración 13. Parte II de la función de AjustarBatGenMayorCons

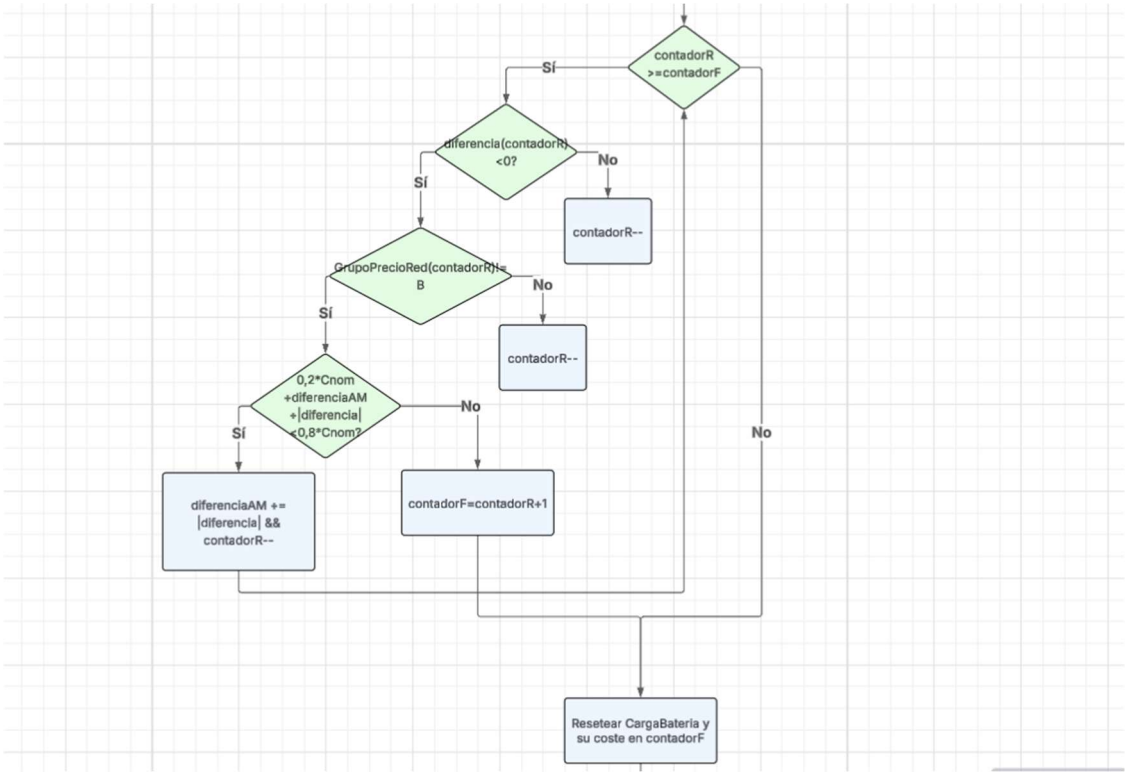


Ilustración 14. Parte III de la función de AjustarBatGenMayorCons

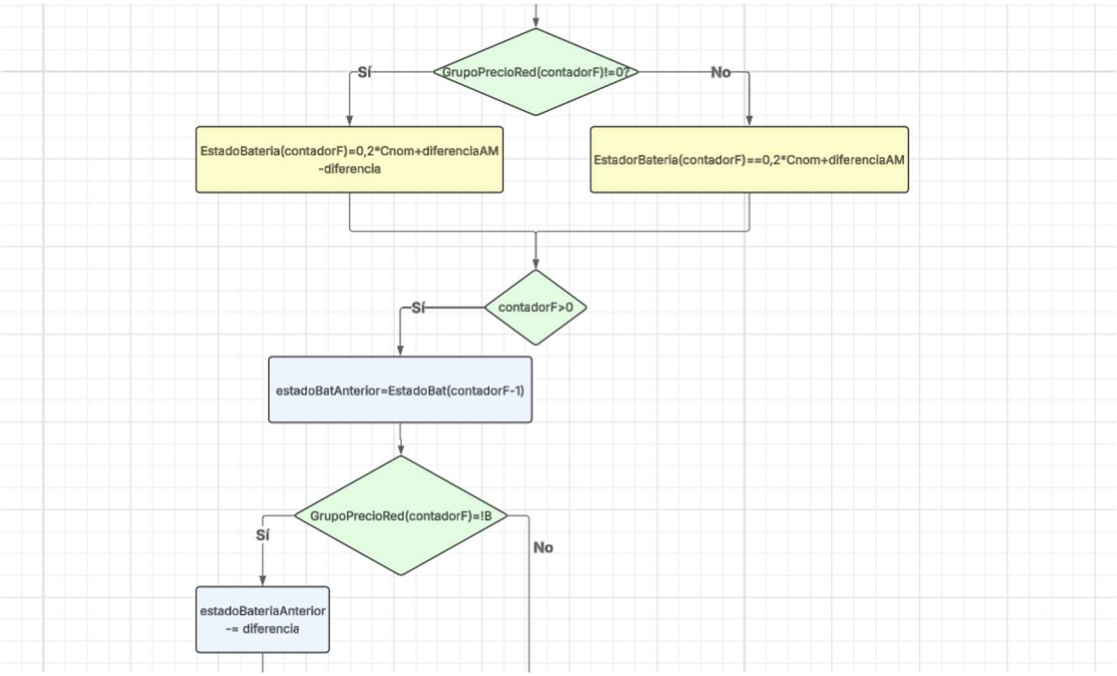


Ilustración 15. Parte IV de la función de AjustarBatGenMayorCons

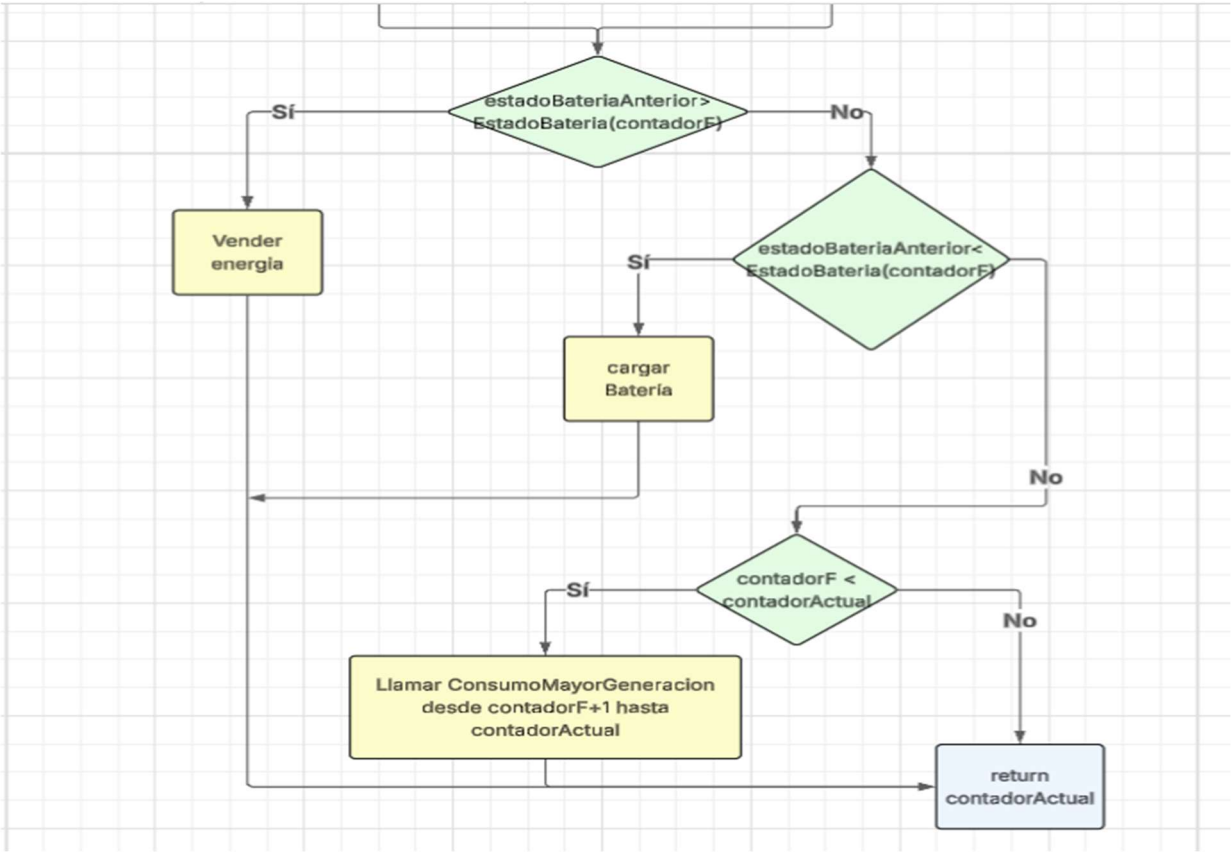


Ilustración 16. Parte V de la función de AjustarBatGenMayorCons

4.6 MODELO DE VISUALIZACIÓN

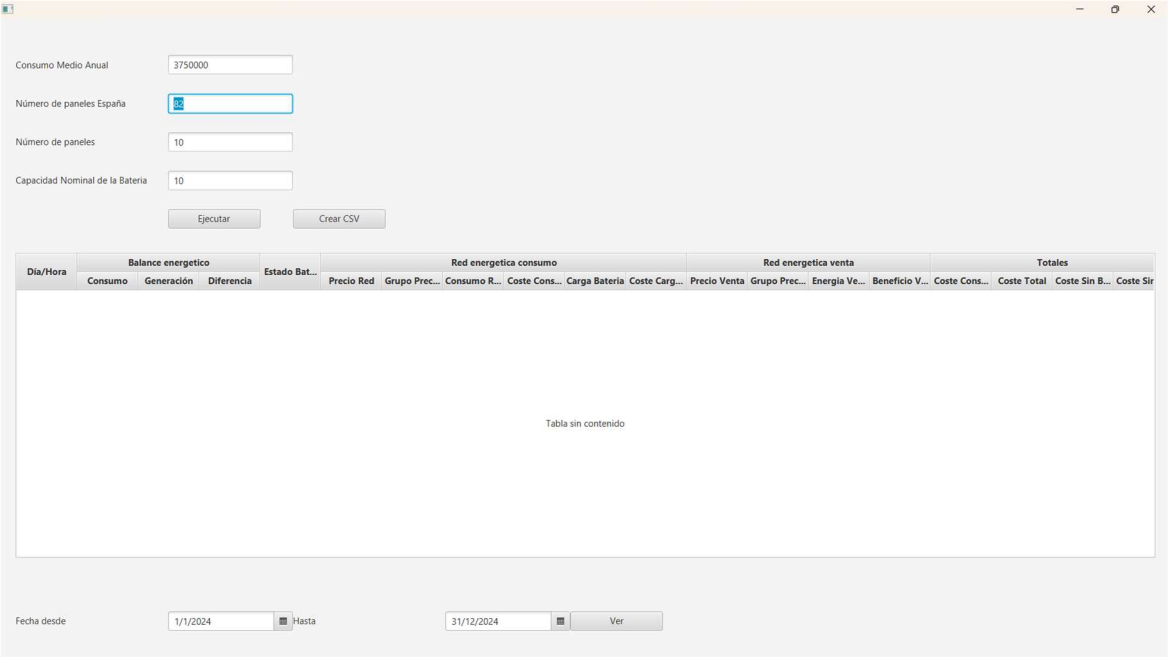


Ilustración 17. Primera imagen de la interfaz con el usuario

Al iniciar la aplicación (Imagen 1), se muestra una interfaz simple en la que el usuario debe introducir dos valores iniciales:

1. Número de paneles (del sistema del usuario).
2. Capacidad Nominal de la Batería (kWh).

En la interfaz, se han añadido el consumo medio anual y el número de paneles en España en millones para poder cambiarlos de manera más accesible si fuese necesario.

Una vez introducidos los valores deseados, se pulsa el botón Ejecutar, lo que genera los resultados en una tabla y una gráfica en la parte inferior. El botón Crear CSV permite

exportar los resultados para generar gráficas externas o realizar cálculos adicionales con otras herramientas.

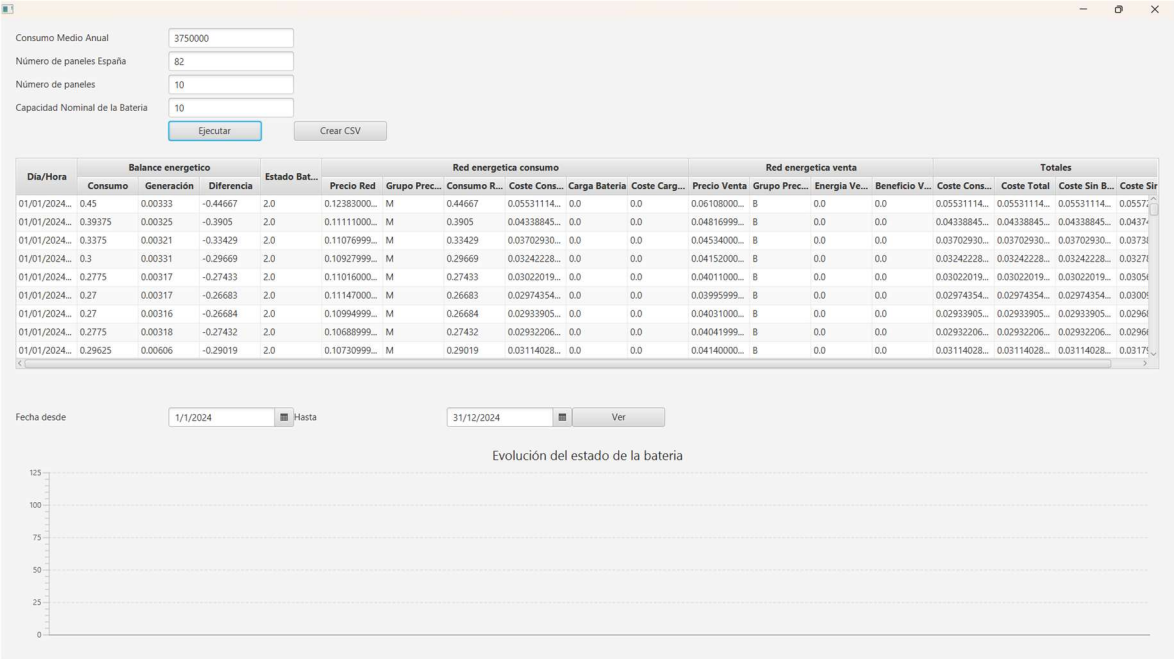


Ilustración 18. Segunda imagen de la interfaz con el usuario

En la Ilustración 19, se aprecia la tabla generada tras la ejecución. Esta tabla muestra el balance energético y el estado de la batería hora a hora, incluyendo parámetros como consumo, generación, diferencia, estado de carga, precios, energía comprada y vendida, y costes o beneficios asociados.

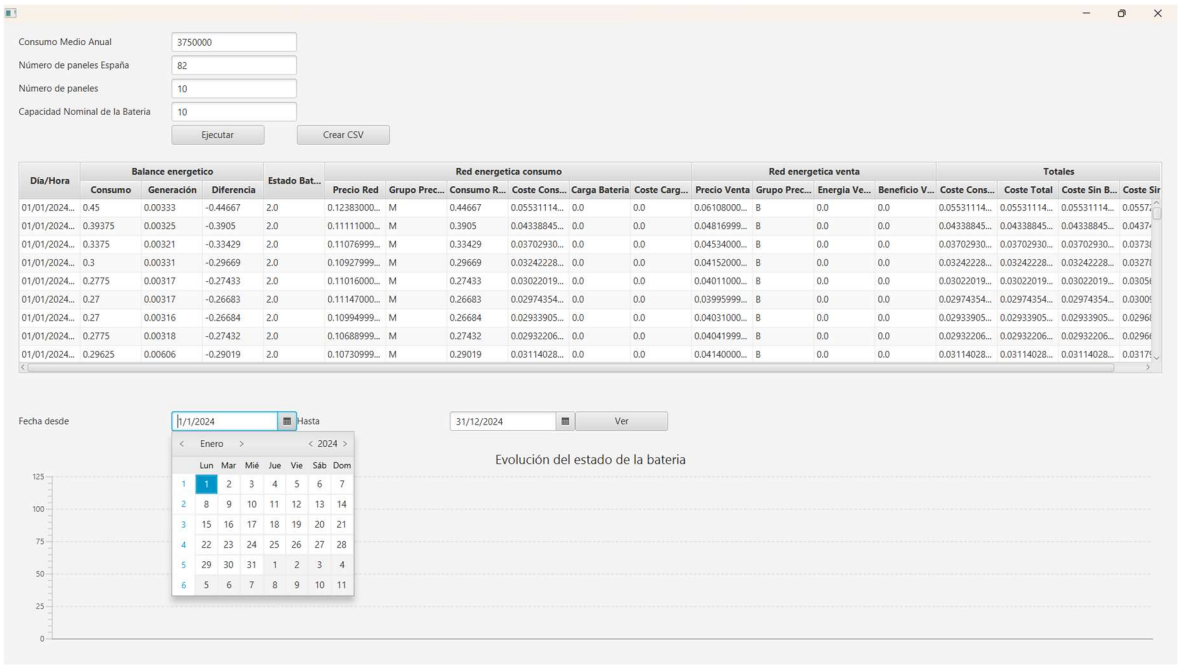


Ilustración 19. Tercera imagen de la interfaz con el usuario

La Ilustración 20 muestra cómo el usuario puede seleccionar cualquier rango de fechas utilizando los calendarios de las opciones Fecha desde y Hasta. La aplicación cuenta con datos de todo el año 2024, de manera que es posible visualizar la evolución del estado de carga de la batería en cualquier periodo de ese año.

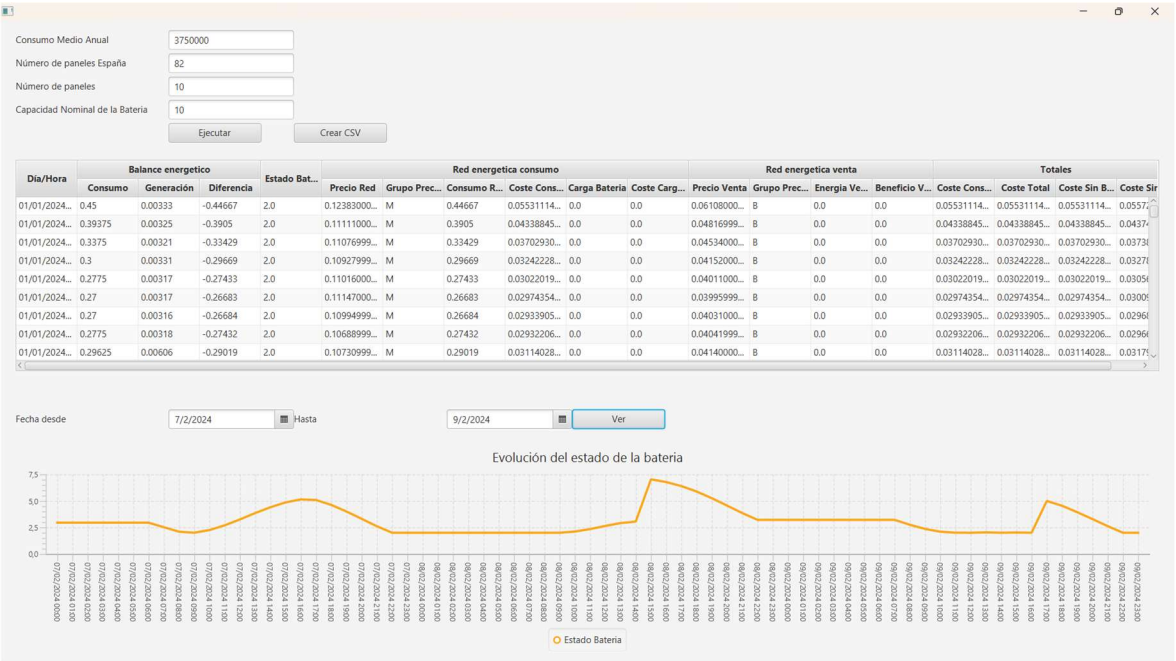


Ilustración 20. Cuarta imagen de la interfaz con el usuario

En la ilustración 21 se ve como en la parte inferior, se muestra la gráfica titulada Evolución del estado de la batería, que representa visualmente el comportamiento del almacenamiento energético, lo que facilita la interpretación de los datos de la tabla. La curva muestra cómo varía el nivel de carga a lo largo del tiempo seleccionado, permitiendo detectar patrones y momentos clave en la operación del sistema.

Así es como se muestra la interfaz con el usuario, aunque la finalidad de este proyecto era solo crear la lógica. Debido a que era necesario poder crear gráficas para ver como evoluciona el estado de la batería y para calcular el coste energético, se ha creado esta interfaz simple para facilitarlo.

Capítulo 5. DATOS DE ENTRADA Y SIMULACIÓN

5.1 DATOS METEOROLÓGICOS Y DE RADIACIÓN

Todos los datos utilizados en esta aplicación provienen de la Red Eléctrica de España (REE). Los datos de generación solar utilizados ofrecen información horaria de producción fotovoltaica agregada a nivel nacional en megavatios (MW) (REE, 2025a). Estos datos se convierten a gigavatios (GW) y se normalizan dividiéndolos por el número estimado de paneles solares instalados en España, con el objetivo de obtener una media representativa de generación por cada panel. A finales de 2024, se estimó que en España había aproximadamente 82 millones de paneles solares instalados, con una potencia media de 395W por panel y una potencia fotovoltaica total instalada de 32.350 MW (REE, 2025).

$$Número\ de\ paneles = \frac{32.350\ MW}{0{,}395\ kW/panel} \approx 81.898.734\ paneles$$

Posteriormente, esta media se multiplica por el número de paneles instalado en el sistema doméstico para estimar la generación horaria del usuario. Aunque este método permite una estimación funcional, lo ideal sería disponer de datos meteorológicos locales para afinar la precisión del modelo.

Variables como la irradiancia solar, la temperatura ambiente, la velocidad del viento y la nubosidad tienen un impacto directo en la producción de energía solar. Debido a que en esta simulación de la aplicación ya se cogen datos de generación, no tenemos en cuenta estas variables, pero su tuviésemos que calcular la generación de manera local si que tendrían relevancia.

La irradiancia es el factor más determinante, ya que mide la cantidad de radiación solar que incide sobre una superficie; a mayor irradiancia, mayor será la generación eléctrica. Una temperatura muy elevada puede afectar al rendimiento de las células fotovoltaicas disminuye cuando la temperatura del panel supera los 25 °C, debido al aumento de la resistencia interna (IDAE, 2023). Esta pérdida de rendimiento sí se ha tenido en cuenta en la batería, que es la que controlamos en esta aplicación.

El viento afecta de manera indirecta; una mayor velocidad del viento puede ayudar a enfriar los paneles, reduciendo la temperatura de operación y mejorando su eficiencia. Las nubes reducen drásticamente la radiación directa, lo que disminuye la producción, aunque parte se compensa con radiación difusa. Las condiciones variables de nubosidad pueden provocar

fluctuaciones rápidas e impredecibles en la generación, lo que dificulta los cálculos. (REE, 2024a).

Debido a todos los cálculos que habría que hacer, una futura mejora del sistema podría incluir datos de estaciones meteorológicas locales, modelos satelitales o APIs de servicios como Copernicus o AEMET para obtener predicciones y registros en tiempo real, de esta forma no habría que tener en cuenta todas estas variables.

5.2 PERFILES DE CONSUMO ELÉCTRICO RESIDENCIAL

Para estimar el consumo eléctrico por hora de un domicilio, se utilizan los perfiles horarios oficiales elaborados por REE, quienes generan estos patrones de consumo a partir de mediciones reales o mediante métodos de estimación cuando no se requiere un contador inteligente (REE, 2025b). Estos perfiles se aplican luego al consumo medio anual de un hogar, estimado en alrededor de 3 487 kWh/año según el IDEA (Iberdrola, 2024), lo que equivale a unos 290 kWh/mes o 9,5 kWh/día (Endesa, 2022a).

Por ejemplo, si un horario tiene asignado un 10 % del consumo, ese día se estimarían 34,87 kWh en ese intervalo específico (3 487 kWh × 0,10), adaptando así el perfil global a la realidad semanal o mensual del hogar. Los datos que proporciona Red Eléctrica son anuales por lo tanto sería utilizando el consumo medio anual.

Este método permite generar curvas de demanda realistas y útiles para la simulación de sistemas que gestionan generación propia, almacenamiento o respuesta a precios variables. Cabe resaltar que la demanda del sector residencial representa aproximadamente el 20 % del consumo de España (Madrid360, 2022), por eso son de gran importancia estas aplicaciones orientadas a la optimización energética.

5.3 PRECIO DE VENTA DE EXCEDENTES FOTOVOLTAICOS

El precio de venta de excedentes en España se regula a partir de un mecanismo de compensación simplificada, obteniéndose los datos de REE (REE, 2024b). Esto permite a los usuarios verter el excedente de energía generada a la red y recibir una compensación económica por cada kilovatio hora (kWh) exportado. Este precio depende de la comercializadora y de la tarifa contratada, aunque generalmente varía entre (Endesa, 2024). Esta compensación solo cubre el término de energía, no afectando a otros conceptos de la factura como peajes o costes fijos. Además, esta venta se realiza a precio horario, por lo que los ingresos dependen también del momento del vertido, lo que justifica la necesidad de una gestión energética optimizada.

5.4 PRECIO DE COMPRA DE ENERGÍA DESDE LA RED

El precio de compra de energía desde la red se basa íntegramente en la tarifa regulada PVPC (Precio Voluntario para el Pequeño Consumidor), correspondiente a la tarifa 2.0 TD (REE, 2024c). Este precio tiene una estructura horaria diferenciada en tres tramos -punta, llano y valle- que reflejan los periodos de mayor y menor demanda eléctrica (TotalEnergies, 2025). La Red Eléctrica de España publica cada día, a las 20:15 h, el precio horario del día siguiente para el término de energía, disponible en su web y en la aplicación eSios/redOS. El enfoque utilizado consiste en usar directamente estos precios, sin requerir que el usuario especifique su tipo de tarifa. Una futura mejora es preguntarle al usuario que tipo de tarifa tiene para obtener resultados más exactos.

5.5 TARIFAS ELÉCTRICAS REALES

En el sistema eléctrico español, los consumidores pueden elegir entre dos modelos de contratación: el mercado regulado y el mercado libre. Cada uno tiene aspectos que afectan al coste, la previsibilidad y la flexibilidad del consumo eléctrico.

En el mercado regulado, la tarifa más común es el PVPC. Esta tarifa está gestionada por las comercializadoras de referencia y varía en función del precio del mercado mayorista de electricidad. Esta tarifa se publica diariamente por la REE para el día siguiente (TotalEnergies, 2025). Esta opción ofrece transparencia y flexibilidad y tiene acceso a Bono Social para consumidores vulnerables, por eso es ampliamente utilizado por usuarios domésticos (Observatorio de Energía, 2024).

Por otro lado, está el mercado libre, que está formado por comercializadoras privadas que diseñan sus propias ofertas y tarifas. Siendo las principales modalidades la tarifa fija, manteniendo el precio del kWh fijo durante todo el año, proporcionando previsibilidad, aunque suele resultar más cara en comparación con la PVPC. La trifa indexada, varía el precio del kWh según el mercado mayorista, con un margen comercial pactado. Esta tarifa puede ser más económica en algunos periodos, pero tiene mayor volatilidad (TarifasGasLuz, 2025). La tarifa plana, mantiene una factura mensual fija independiente del consumo real, es una tarifa cómoda, pero no refleja un consumo eficiente.

Como se ha mencionado en el apartado anterior, en este proyecto se ha utilizado el mercado regulado, debido a su crecimiento en España y por no tener más modalidades como el mercado libre.

5.6 *PRUEBAS REALIZADAS*

Debido a que esta aplicación está diseñada para analizar datos de 365 días, hay que asegurarse de que la lógica, con estos datos, funciona para todos los casos. Para ello, se han analizado diferentes momentos del año en el que los datos varían más. Se han escogido los días 10-13 de febrero, mayo, agosto y noviembre para analizar con cuidado cómo evoluciona la batería, si se vende energía o se compra y si se han realizado correctamente los cálculos.

Ilustración 21. Evolución del estado de la batería del 10-13 de febrero

Ilustración 22. Evolución del estado de la batería del 10-13 de mayo

Ilustración 23.Evolución del estado de la batería del 10-13 de agosto

Ilustración 24. Evolución del estado de la batería del 10-13 de noviembre

Estas imágenes son sacadas de la aplicación, que como se puede observar, se pueden obtener las gráficas de cómo evoluciona el estado de la batería los días que desee el usuario. Una vez comprobado que los valores calculados son correctos se obtiene como varía el estado de la batería y se observa como aumenta y disminuye la carga y en qué hora.

Capítulo 6. RESULTADOS Y ANÁLISIS

6.1 COMPORTAMIENTO DEL SISTEMA BAJO DISTINTOS ESCENARIOS

El comportamiento del sistema fotovoltaico con almacenamiento puede variar de forma significativa en función de dos parámetros clave: la capacidad nominal de la batería y el número de paneles solares instalados. Este análisis permite evaluar cómo influye el dimensionamiento del sistema sobre el grado de autosuficiencia, el aprovechamiento de la generación renovable y la evolución del estado de carga (SoC) de la batería.

6.1.1 CAPACIDAD NOMINAL DE LA BATERÍA

La capacidad nominal de una batería, expresada en kilovatios-hora (kWh), determina cuánta energía puede almacenar en condiciones óptimas. Este valor tiene un impacto directo sobre la cantidad de energía solar que puede aprovecharse para su uso posterior durante la noche o en momentos de baja generación. La elección de la capacidad adecuada depende del perfil de consumo diario del hogar, la frecuencia de los excedentes generados y el nivel de independencia energética deseado.

Los sistemas domésticos suelen incorporar baterías con capacidades nominales entre 5 y 15 kWh (Cambio Energético, s.f). En este trabajo se han seleccionado dos valores representativos para realizar las simulaciones: 7,5 kWh, como ejemplo de una instalación intermedia adecuada para un hogar medio, y 13,5 kWh, representando un sistema de alta capacidad similar al que ofrecen soluciones como la Tesla Powerwall 2. El número de paneles en estos casos va a ser de 10. Esta comparación permite evaluar la influencia del almacenamiento disponible sobre el uso de la energía renovable y la necesidad de recurrir a la red eléctrica.

Ilustración 25. Gráfico de la evolución del estado de la batería el 20/03/2024 con 7,5kWh

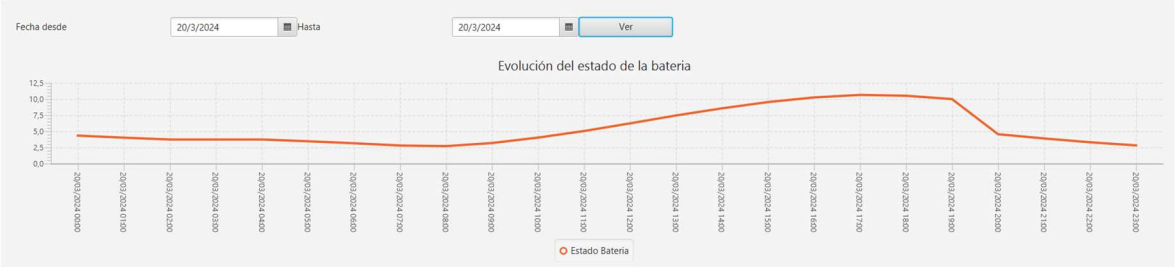


Ilustración 26. Gráfico de la evolución del estado de la batería el 20/03/2024 con 13,5kWh

En las gráficas presentadas se observa claramente la diferencia en el comportamiento del estado de carga (SoC) para dos capacidades nominales: 7,5 kWh y 13,5 kWh. La segunda gráfica, correspondiente a la batería de mayor capacidad, muestra una acumulación superior de carga, lógica dado su mayor almacenamiento, aunque también implica un coste significativamente mayor.

Económicamente, el sistema con 13,5 kWh arroja un balance anual de -75,47 €, lo que representa un mayor beneficio neto en comparación con el de 7,5 kWh, que es de solo -3,83 €. Sin embargo, estas cifras no incluyen el coste proporcional anual de la instalación —como el sistema de montaje, inversor o BMS, ni el sobrecoste inicial asociado a una batería de mayor capacidad.

Este incremento del coste es esperable: el precio de los sistemas de almacenamiento se encarece al aumentar la capacidad. Según datos recientes, el coste instalado de baterías de litio para autoconsumo en el sector residencial europeo se sitúa entre 300 y 400 €/kWh, mientras que en el mercado internacional los precios oscilan entre 280 y 580 USD/kWh, dependiendo de la tecnología y el tamaño de la instalación (GSL Energy, 2025; SolarPower Europe, 2023). Esto significa que, al duplicar la capacidad nominal, se incrementa de forma prácticamente proporcional la inversión inicial, lo que impacta directamente en el periodo de amortización y en la rentabilidad del sistema.

En resumen, aunque una batería de mayor capacidad permite obtener un ahorro anual más notable, como se observa en los datos de -75,47 € frente a –3,83 €, también implica un desembolso inicial considerablemente superior, que debe valorarse cuidadosamente en el análisis económico del sistema fotovoltaico.

6.1.2 NÚMERO DE PANELES SOLARES

La cantidad de paneles solares instalados define la potencia pico del sistema y, por tanto, su capacidad de generación. En el contexto residencial español, una instalación tipo suele constar de entre 5 y 14 módulos, dependiendo de factores como el consumo eléctrico del hogar, la superficie disponible o la orientación del tejado (Climatermia, 2025). En general, con 6 a 15 paneles de 450 Wp se puede cubrir una parte significativa o incluso la totalidad de la demanda eléctrica de una vivienda media.

En este estudio se han definido dos escenarios: uno con 6 paneles solares (equivalente a 2,7 kWp de potencia instalada) y otro con 15 paneles (6,75 kWp), manteniendo una capacidad nominal de 10 kWh, lo que permite comparar el impacto que tiene la capacidad de generación sobre el estado de carga de la batería, los excedentes gestionados y el nivel de autosuficiencia del hogar. Al hacer la simulación variando estos dos datos, se ha comprobado que, a mayor número de paneles, mayor generación fotovoltaica, por lo tanto, se obtiene un mayor ahorro económico. Con 6 paneles solares se ha calculado un coste de 37,84 euros y con 15 paneles solares de -157,87, siendo este el coste simplemente de energía comprada, sin tener en cuenta los términos fijos de potencia contratada, ni alquiler de equipo de medida ni impuestos asociados. Por lo tanto, se ve que la hipótesis se cumple teniendo un coste más elevado con 6 paneles.

El análisis conjunto de estas dos variables ofrece una visión más completa del comportamiento del sistema en diferentes condiciones de uso. Además, proporciona una base sólida para valorar hasta qué punto una mayor capacidad de almacenamiento o una instalación más generosa de paneles puede mejorar el rendimiento energético del sistema fotovoltaico con batería.

6.2 EVOLUCIÓN DEL ESTADO DE CARGA (SOC)

Ilustración 27. Evolución del estado de la batería del 15 al 16 de octubre

La gráfica muestra la evolución del estado de carga (SoC) de una batería de 10 kWh de capacidad nominal, instalada junto con un sistema fotovoltaico de 10 paneles solares, que es el número que se ha elegido por defecto, durante el periodo comprendido entre el 15 y el 16 de octubre de 2024. Teniendo como referencia los umbrales del 20 % de la capacidad nominal (2 kWh) como límite inferior y el 80 % (8 kWh) como límite superior. Durante el día 15, el SoC se mantiene mayoritariamente en valores intermedios, con ligeras variaciones

asociadas a los ciclos de carga y descarga, sin alcanzar en ningún momento el umbral máximo. En la madrugada y primeras horas del día 16, la batería permanece próxima al límite inferior debido a un mayor consumo y a la ausencia de generación solar. A partir de media mañana, el incremento de la irradiación permite un aumento progresivo del SoC, alcanzando su valor máximo cercano al 80 % en la tarde, gracias a la elevada producción de los 10 paneles solares. Posteriormente, al caer la noche, se inicia un nuevo proceso de descarga que reduce gradualmente el nivel de carga, reflejando el suministro de energía al consumo doméstico en ausencia de producción fotovoltaica. Este es un ejemplo de un día aleatorio de cómo evoluciona el estado de carga de la batería.

6.3 *ENERGÍA CONSUMIDA, ALMACENADA Y VERTIDA*

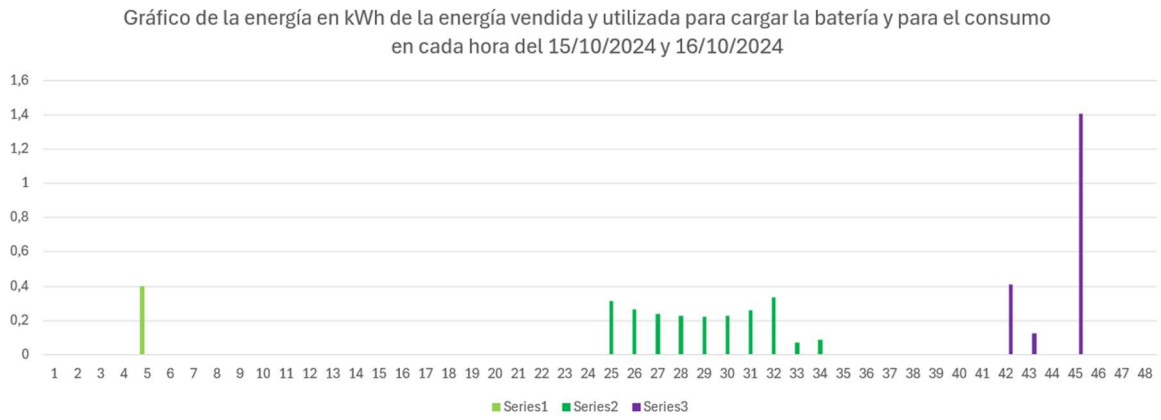


Ilustración 28. Gráfico de la energía vendida, consumida y utilizada para cargar la batería

En la Figura se muestra la evolución horaria de tres flujos energéticos durante los días 15 y 16 de octubre de 2024: energía consumida de la red para cargar la batería (Serie 1), energía consumida de la red para abastecer la demanda (Serie 2) y energía vendida a la red (Serie 3).

La serie 1 es una acción necesaria para que tenga energía suficiente cuando el precio de la red sea elevado, para eso ha de cargarse posteriormente. Se observa que la serie 2 coincide con horas en las que el precio horario de la electricidad es más bajo según el mercado diario español (OMIE, 2024). Este patrón responde a una estrategia de gestión energética orientada a la optimización económica, en la que la batería se carga aprovechando las tarifas reducidas y, cuando resulta más rentable, se utiliza energía de red para el consumo en lugar de descargar la batería.

En cuanto a la Serie 3, los incrementos registrados en las horas 42 a 47 corresponden a momentos en los que, tras un periodo prolongado de “consumo mayor que generación”, la batería alcanza su umbral mínimo de carga o se identifican precios de venta elevados en el mercado. En estas condiciones, la descarga parcial o total de la batería para su venta a la red maximiza el beneficio económico, aprovechando la volatilidad de los precios horarios. Esta práctica está respaldada por investigaciones que demuestran que el uso flexible de sistemas

de almacenamiento energético permite incrementar la rentabilidad y estabilidad del sistema en mercados eléctricos con alta penetración renovable.

6.4 AHORROS ENERGÉTICOS ESTIMADOS

El principal beneficio de instalar un sistema fotovoltaico en el ámbito residencial es la reducción del consumo eléctrico procedente de la red, lo que se traduce en un importante ahorro económico y energético para el usuario. La magnitud de este ahorro depende, principalmente, de si el sistema cuenta o no con almacenamiento mediante batería.

En un hogar sin instalación fotovoltaica, el 100 % de la energía consumida proviene de la red eléctrica. Teniendo en cuenta un consumo medio de entre 2.950 y 3.500 kWh/año (IDAE, 2024), y un precio medio regulado de unos 0,14 €/kWh (Naturgy, s.f), el gasto anual puede situarse en torno a los 600 € a 1000 € por vivienda, sin contar costes fijos ni impuestos.

La instalación de un sistema fotovoltaico sin almacenamiento permite cubrir aproximadamente entre 30 % y 60 % del consumo anual, dependiendo de la orientación de los paneles, la ubicación geográfica y los hábitos de consumo (Efitar ingeria, 2025). En un caso medio, se estima un ahorro del 45 % en el consumo de red, lo que representa unos 260 € a 315 € anuales. El resto del excedente no consumido en el momento puede ser vertido a la red y compensado parcialmente en la factura eléctrica mediante el mecanismo de compensación simplificada, generando un ahorro adicional de entre 50 € y 100 € al año (Repsol, 2025).

Al incorporar una batería doméstica, se logra aprovechar un mayor porcentaje de la energía generada, llegando a tasas de autoconsumo superiores al 70 % o incluso al 85 % si el sistema está bien dimensionado (Fundación Renovables, 2023). En este caso, el ahorro energético anual puede alcanzar hasta los 400 € o 500 €, sumando la energía autoconsumida directamente y la que se almacena para cubrir la demanda nocturna. Además, se reducen aún más los picos de demanda a la red, lo cual contribuye a una mayor eficiencia energética del sistema global.

Por tanto, si se compara el escenario sin sistema frente al sistema completo con batería, el ahorro estimado puede superar el 65 % del gasto energético anual, lo que justifica la rentabilidad progresiva de la inversión y su relevancia en un contexto de transición energética.

6.5 GASTOS DE UN DOMICILIO SIN SISTEMA FOTOVOLTAICO

Con la información de consumo anual y aplicando la tarifa regulada PVPC se ha estimado que el coste anual de suministro eléctrico en ausencia de sistema fotovoltaico asciende a 449,95€. Este cálculo considera el término de energía, basado en los precios horarios del

mercado mayorista, pero no los términos fijos de potencia contratada, ni alquiler de equipo de medida ni impuestos asociados. El PVPC, gestionado por comercializadoras de referencia y regulado por el Gobierno de España, refleja directamente la volatilidad del mercado eléctrico diario, lo que permite estimaciones realistas cuando se dispone de datos horarios de consumo (García Gómez, P.2022). En comparación con el gasto estimado en el apartado anterior, este valor es ligeramente inferior, pero sigue situándose dentro de un rango razonable, por lo que no resulta preocupante.

6.6 GASTOS DE UN DOMICILIO CON SISTEMA, PERO SIN ALMACENAMIENTO

En el escenario de autoconsumo fotovoltaico sin almacenamiento, el coste anual del suministro eléctrico se sitúa en torno a 241,48€. Esto se debe a que una parte relevante del consumo se cubre directamente mediante autoconsumo instantáneo, lo que reduce las compras a la red. Además, gracias al mecanismo de compensación simplificada de excedentes, regulado por el Real Decreto 244/2019, los excedentes de energía vertidos a la red se traducen en descuentos en la factura eléctrica, siempre que la instalación cumpla los requisitos (potencia ≤ 100 kW, energía renovable, sin régimen retributivo adicional) (MITECO, 2024). Este mecanismo establece que el valor económico de los excedentes no puede superar el del consumo de red en el periodo facturado (García Gómez,P. 2022). Estudios recientes muestran que las instalaciones domésticas fotovoltaicas logran ahorros medios anuales de entre un 84 % y más de 1 000 € en la factura energética tradicional (Imagina Energía, 2024; PV Magazine España, 2024), lo cual refrenda la coherencia de estimaciones como la presentada, siempre que la generación solar cubra un porcentaje alto del consumo. Este precio es exactamente como el estimado, por lo tanto, los cálculos se están realizando correctamente.

6.7 GASTOS DE UN DOMICILIO CON SISTEMA Y ALMACENAMIENTO

En el escenario con sistema fotovoltaico y almacenamiento, el resultado económico anual presenta un valor de –184,47 €, lo que implica que los ingresos por venta de excedentes superan el coste de la energía comprada a la red. Este resultado, inferior al estimado en el apartado anterior, es coherente en situaciones donde la generación renovable y la capacidad de almacenamiento permiten maximizar el autoconsumo y vender energía en periodos de precios elevados. No obstante, es importante señalar que este cálculo no contempla la amortización del coste de las baterías ni de la instalación fotovoltaica, por lo que el balance económico global del sistema sería diferente al considerar la inversión inicial.

Además, en la modelización actual no se ha tenido en cuenta la restricción de que los kWh vertidos a la red nunca pueden superar la generación fotovoltaica real, lo que puede haber provocado que el valor obtenido ascienda. Igualmente, debe considerarse la limitación regulatoria de compensación simplificada, según la cual el importe asociado a los excedentes nunca puede superar el coste de la energía consumida de la red; es decir, si el cálculo

resultara en un valor negativo, debe fijarse en cero, dado que en la práctica no se abonan importes adicionales al consumidor. Ambos aspectos deben corregirse y mejorarse en futuras versiones del modelo para garantizar una estimación más realista (IDAE, 2023; MITECO, 2024; García Gómez, P., 2022).

6.8 RESULTADOS DEL SISTEMA DE ALMACENAMIENTO

Para evaluar el comportamiento del sistema de almacenamiento, se han analizado en detalle los meses de mayo y noviembre, representativos de condiciones estacionales con elevada y reducida radiación solar, respectivamente.

La generación fotovoltaica considerada corresponde a un sistema con una batería de 10kWh de capacidad nominal y 10 paneles solares monocristalinos (Mono-Si), con una eficiencia en el rango del 18–22 %, lo que los convierte en la opción más adecuada cuando se dispone de un espacio limitado en la cubierta. Aunque en esta aplicación los perfiles de generación empleados provienen de los datos horarios de Red Eléctrica de España (REE), se ha planteado el uso de paneles monocristalinos como referencia tecnológica estándar. En este contexto, la producción fotovoltaica estimada es de alrededor de 620 kWh en mayo, mientras que en noviembre desciende hasta valores próximos a 270 kWh, reflejando la clara variación estacional de la radiación solar.

En cuanto al uso de la batería, los resultados muestran que en el mes de marzo la batería se carga y descarga de manera completa unas 28 veces, mientras que en noviembre esta cifra asciende a unas 32 veces. Este mayor número de ciclos en noviembre se debe a que, al existir menor generación solar, la batería tiende a descargarse con mayor frecuencia para cubrir la demanda, aunque con menores cantidades de energía acumulada en cada ciclo. Aproximadamente, la batería se descarga una vez al día como mínimo, confirmando su papel esencial en la gestión del autoconsumo.

En lo referente a la energía vertida a la red, la máxima venta de excedentes entre estos dos meses se registra en marzo, con un valor de 5,93086 kWh el día 03/03/2024 a las 21:00 horas, momento en el que la generación solar acumulada y la menor demanda instantánea del hogar permitieron maximizar la inyección a red.

Capítulo 7. DISCUSIÓN

7.1 IMPACTO MEDIOAMBIENTAL DEL AUTOCONSUMO CON BATERÍAS

La implantación de sistemas de autoconsumo con almacenamiento contribuye de manera significativa a la reducción de emisiones de gases de efecto invernadero en España. La energía solar fotovoltaica es una fuente renovable que, durante su operación, no produce CO_2 ni contaminantes atmosféricos, favoreciendo el cumplimiento de los compromisos nacionales e internacionales en materia de descarbonización. Cada kWh generado con energía solar evita aproximadamente 0,5 kg de CO_2 en comparación con fuentes fósiles (Vallesolar, 2023). Ejemplos reales en el territorio español demuestran su impacto: la planta fotovoltaica de El Casar (Guadalajara) evita cerca de 11873 t de CO_2 al año (Wikipedia, 2025), mientras que la cubierta fotovoltaica del Teatro Real de Madrid ha logrado reducir el consumo energético del edificio en un 50 % (El País, 2025). A menor escala, proyectos como la instalación de 33000 paneles en Stellantis Vigo permiten evitar más de 9000 t de CO_2 anuales (Cadena SER, 2025b). Estos datos evidencian que la integración de baterías en sistemas de autoconsumo no solo optimiza el uso de la energía generada, sino que amplifica el beneficio ambiental al maximizar la utilización de electricidad libre de emisiones.

7.2 ESCALABILIDAD A OTROS CONTEXTOS

Aunque el sistema propuesto se ha diseñado para un entorno residencial, su arquitectura modular y la lógica de gestión implementada permiten su adaptación a múltiples contextos. En el ámbito industrial, podría aplicarse para reducir la factura eléctrica en fábricas con elevados consumos diurnos, complementando la generación renovable con almacenamiento para cubrir picos de demanda. En instalaciones públicas, como centros educativos o deportivos, su implementación podría contribuir a la autosuficiencia energética y a la reducción de costes operativos. Asimismo, en entornos aislados o de difícil acceso a la red, como refugios de montaña, estaciones meteorológicas o explotaciones agrícolas, el sistema permitiría garantizar un suministro estable aprovechando al máximo la energía solar disponible. Estas aplicaciones, ya presentes en ejemplos como el Teleférico del Teide o proyectos turísticos en el Somontano, muestran que el sistema no está limitado a un uso doméstico, sino que puede ser una solución eficaz en una amplia variedad de escenarios.

7.3 ALINEACIÓN DEL PROYECTO CON LOS OBJETIVOS DE DESARROLLO SOSTENIBLE

El proyecto de gestión optimizada de baterías en sistemas fotovoltaicos residenciales se alinea con varios Objetivos de Desarrollo Sostenible (ODS) propuestos por la ONU (ONU, 2015).

ODS 7: Energía Asequible y No Contaminante

Este proyecto fomenta el uso eficiente de la energía solar fotovoltaica, permitiendo que los hogares reduzcan su dependencia de fuentes de energía convencionales y optimicen el almacenamiento en baterías. Mediante un sistema inteligente de gestión, se maximiza el autoconsumo y se minimiza la compra de electricidad de la red, promoviendo el acceso a energía más limpia y asequible (IEA, 2022). Estudios recientes han demostrado que la gestión avanzada del almacenamiento puede reducir el costo de la energía en el hogar y mejorar la eficiencia del sistema eléctrico (Luthander et al., 2015).

ODS 9: Industria, Innovación e Infraestructura

El desarrollo de un algoritmo de gestión energética en Java impulsa la digitalización y la innovación en el sector energético (Shahsavari & Akbari, 2018). La optimización del almacenamiento y consumo de energía en entornos residenciales representa una mejora tecnológica que puede ser escalable a infraestructuras mayores, impulsando la modernización de los sistemas de generación distribuida y almacenamiento energético (IRENA, 2020).

ODS 11: Ciudades y Comunidades Sostenibles

Al mejorar la eficiencia en el consumo de energía y reducir la dependencia de la red eléctrica,
este proyecto contribuye al desarrollo de ciudades más sostenibles (UN-Habitat, 2020). La integración de baterías y sistemas fotovoltaicos en los hogares favorece la descentralización
de la energía, disminuyendo la sobrecarga en infraestructuras urbanas y facilitando el desarrollo de redes eléctricas más resilientes e inteligentes (Fares & Webber, 2017).

ODS 12: Producción y Consumo Responsables

El proyecto optimiza el uso de los recursos energéticos al evitar desperdicios y reducir el consumo innecesario de electricidad de la red. La estrategia de gestión
eficiente de baterías ayuda a los hogares a consumir energía de manera más racional, minimizando pérdidas y mejorando la rentabilidad del sistema, lo que contribuye a un modelo energético más sostenible (IEA, 2022).

ODS 13: Acción por el Clima

La optimización del almacenamiento de energía solar y la reducción del consumo de electricidad de fuentes fósiles contribuyen a la mitigación del cambio climático (IPCC, 2021). Al fomentar el uso de energías renovables y reducir la huella de carbono del sector residencial, este proyecto apoya la transición hacia un sistema energético más limpio y resiliente ante el cambio climático (REN21, 2022).

Capítulo 8. CONCLUSIÓN

La realización de este proyecto ha permitido demostrar que un sistema de autoconsumo con batería no solo es técnicamente viable en el contexto residencial español, sino que también supone una inversión con claros beneficios económicos, energéticos y medioambientales. La integración de generación fotovoltaica con almacenamiento mediante baterías de ion-litio optimiza el aprovechamiento de la energía renovable generada, permitiendo desplazar el consumo hacia las horas sin producción y reduciendo de forma significativa la compra de electricidad a la red. Este enfoque, respaldado por estudios como los del Instituto para la Diversificación y Ahorro de la Energía (IDAE, 2023) y la Comisión Nacional de los Mercados y la Competencia (CNMC, 2024), coincide con las tendencias actuales de transición energética, donde el autoconsumo con almacenamiento es clave para alcanzar los objetivos de descarbonización establecidos en el Plan Nacional Integrado de Energía y Clima (PNIEC) 2021-2030.

El proyecto ha permitido validar, a través de simulaciones, que la elección adecuada de la capacidad de la batería y el número de paneles solares influye directamente en el grado de autosuficiencia, el aprovechamiento de excedentes y la estabilidad del suministro. Con escenarios dimensionados de forma realista —y respaldados por valores de mercado procedentes de empresas como Iberdrola (2024) y Endesa (2025)— se ha comprobado que un sistema bien configurado puede cubrir un alto porcentaje de la demanda anual de un hogar, llegando a generar ahorros superiores al 50 % en la factura eléctrica (Efitar Ingeniería, 2025). Además, el almacenamiento contribuye a aliviar la carga sobre la red eléctrica en horas punta, favoreciendo la estabilidad del sistema eléctrico nacional, tal y como apunta Red Eléctrica de España (REE, 2024a).

Desde un punto de vista medioambiental, la reducción de la dependencia de fuentes fósiles para la generación eléctrica conlleva una disminución directa de las emisiones de gases de efecto invernadero. Según la Agencia Internacional de la Energía Renovable (IRENA, 2023), cada kWh generado mediante energía solar fotovoltaica en España evita la emisión de aproximadamente 0,5 kg de CO_2 equivalente respecto a la generación media del sistema. Esto implica que, a lo largo de su vida útil, el sistema diseñado en este proyecto podría evitar la emisión de varias toneladas de CO_2, contribuyendo de manera tangible a los compromisos climáticos internacionales.

En definitiva, este trabajo ha merecido la pena porque ha permitido desarrollar y analizar una solución de autoconsumo con almacenamiento que no solo responde a las necesidades actuales de eficiencia y sostenibilidad, sino que también ofrece un marco escalable y adaptable a futuros escenarios energéticos. La combinación de análisis técnico, validación mediante datos reales y alineación con las políticas energéticas vigentes garantiza que este proyecto pueda servir como referencia para la implementación de sistemas similares, fomentando así un modelo energético más limpio, eficiente y resiliente para los hogares españoles.

Capítulo 9. REFERENCIAS

1. Agencia Internacional de Energías Renovables (IRENA). (2023). *Renewable Energy Statistics 2023*. https://www.irena.org
2. Aurora Solar. (2025, 29 de abril). *¿Qué es el arbitraje energético?* Aurora Solar. https://help.aurorasolar.com/hc/es/articles/28998198908563-Comprender-los-modelos-de-almacenamiento-para-el-arbitraje-energ%C3%A9tico
3. Battery University. (s. f.). *BU-802b: What does elevated self-discharge do?* https://batteryuniversity.com/article/bu-802b-what-does-elevated-self-discharge-do
4. Blog Enerlink. (2023, octubre 17). *Economía circular y reciclaje de baterías de vehículos eléctricos*. https://blog.enerlink.com/economia-circular-y-reciclaje-de-baterias-de-vehiculos-electricos
5. Cambio Energético. (s.f.). *¿Qué consumos puedo alimentar con una batería de litio y durante cuánto tiempo?* https://www.cambioenergetico.com/blog/consumos-bateria-litio/
6. Cadena SER. (2025a, 2 de mayo). *El apagón ha evidenciado que depender exclusivamente de la red eléctrica nos hace muy vulnerables energéticamente*. https://cadenaser.com/andalucia/2025/05/02/el-apagon-ha-evidenciado-que-depender-exclusivamente-de-la-red-electrica-nos-hace-muy-vulnerables-energeticamente-ser-malaga/
7. Cadena SER. (2025b, 26 de marzo). *Stellantis Vigo busca nuevas oportunidades en las energías renovables*. https://cadenaser.com/galicia/2025/03/26/stellantis-vigo-busca-nuevas-oportunidades-en-las-energias-renovables-radio-vigo/
8. Cadena SER Andalucía. (2025, 10 de julio). *RDL 7/2025: un salto cuántico para el autoconsumo en España*. https://cadenaser.com/andalucia/2025/07/10/rdl-72025-un-salto-quantico-para-el-autoconsumo-en-espana-ser-malaga/
9. CIRCE. (2022). *El reto del almacenamiento: ¿Cómo dimensionar baterías en plantas renovables para maximizar el aprovechamiento energético?* https://www.fcirce.es/blog/el-reto-del-almacenamiento-dimensionar-baterias-en-plantas-renovables-para-el-aprovechamiento-energetico
10. Climatermia. (2025). *¿Cuántas placas solares necesito para una casa según los m²?* https://climatermia.com/cuantas-placas-solares-necesito-para-una-casa-segun-los-m2/
11. Comisión Nacional de los Mercados y la Competencia (CNMC). (2024). *Informe sobre el autoconsumo en España* https://www.cnmc.es
12. EcoFlow. (2023). *¿Qué hacer con los excedentes de nuestras placas solares?* https://www.ecoflow.com/es/blog/excedentes-placas-solares
13. EDP Energía. (2023). *¿Es rentable una batería solar para una vivienda?* https://www.edpenergia.es/es/blog/energia-fotovoltaica/rentabilidad-bateria-solar/
14. Efitar Ingeniería. (2025). *¿Cuánto se ahorra con placas solares?* https://efitaringenieria.com/cuanto-se-ahorra-con-placas-solares-actualizado-2025/

15. El País. (2025, 6 de marzo). *El Teatro Real estrena su cubierta fotovoltaica: "Si se hace aquí, se puede hacer en cualquier sitio"*. https://elpais.com/clima-y-medio-ambiente/2025-03-06/el-teatro-real-estrena-su-cubierta-fotovoltaica-si-se-hace-aqui-se-puede-hacer-en-cualquier-sitio.html
16. Energías Renovables. (2023, 9 de agosto). *Estos son los 10 gráficos que explican el sector energético español*. https://www.energias-renovables.com/panorama/estos-son-los-10-gr-ficos-que-20230809
17. Energías Renovables. (2024, 9 de octubre). *El Gobierno saca a consulta pública el Real Decreto que actualizará el autoconsumo*. https://www.energias-renovables.com/autoconsumo/el-gobierno-saca-a-consulta-pnblica-el-20241009
18. Endesa. (2022a, 6 de abril). *¿Cómo calcular el consumo eléctrico de una casa?* https://www.endesa.com/es/blog/blog-de-endesa/luz/calcular-consumo-electrico-casa
19. Endesa. (2022b, 18 de marzo). *Se pone en marcha en la central de Endesa en Melilla un innovador sistema de almacenamiento para dar una "segunda vida" a las baterías usadas de los coches eléctricos*. Endesa. https://www.endesa.com/es/prensa/sala-de-prensa/noticias/eficiencia-energetica/economia-circular/puesta-en-marcha-central-almacenamiento-baterias-coches-electricos-melilla
20. Endesa. (2024). *Compensación de excedentes en autoconsumo con batería virtual*. https://www.endesa.com/es/luz-y-gas/autoconsumo-endesa/compensacion-excedentes
21. Fares, R., & Webber, M. E. (2017). *The impacts of storing solar energy in the home to reduce reliance on the utility.* Nature Energy, 2(1), 17001. https://www.nature.com/articles/nenergy20171
22. Fundación Renovables. (2023). *Análisis del potencial de autoconsumo residencial en España*.: https://fundacionrenovables.org
23. Fundación Naturgy / PwC España. (2021). *El papel del almacenamiento en la transición energética en España*. https://static.smartgridsinfo.es/media/2021/05/almacenamiento-transicion-energetica-espana-baterias-informe-naturgy.pdf
24. Gamarra, C., & Guerrero, J. M. (2020). *Barriers to grid-connected battery systems: Evidence from the Spanish electricity market*. arXiv. https://arxiv.org/abs/2007.00486
25. García Gómez, P. (2022, mayo). *Real Decreto 244/2019 – Normativa autoconsumo – Resumen técnico*. E-ficiencia., de https://e-ficiencia.com/normativa-autoconsumo-resumen-tecnico-rd-244-2019/
26. GSL Energy. (2025). *The real cost of commercial battery energy storage in 2025: What you need to know*. R https://www.gsl-energy.com/the-real-cost-of-commercial-battery-energy-storage-in-2025-what-you-need-to-know.html
27. Huawei, 2023. *Huawei LUNA2000 Energy Storage System*. www.huawei.com
28. Iberdrola, 2023. *Planta de almacenamiento en Olmedilla: integración de energía solar y red* www.iberdrola.com
29. Iberdrola. (2024). *¿Qué batería solar necesito?*. https://www.iberdrola.es/autoconsumo/bateria-solar

30. IDAE. (2023). *Efectos de la temperatura en el rendimiento fotovoltaico*. https://www.idae.es
31. IDAE. (2024, 10 de octubre). *¿Cuántos kWh consume una casa al mes?* Iberdrola. https://www.iberdrola.es/blog/luz/cuantos-kwh-consume-una-casa-al-mes
32. International Energy Agency (IEA). (2022). Renewables 2022 Market Report.: https://www.iea.org/reports/renewables-2022
33. International Renewable Energy Agency (IRENA). (2020). Innovation landscape for a renewable-powered future. IRENA Publications. https://www.irena.org/publications/2019/Feb/Innovation-landscape-for-arenewable-powered-future
34. Innowise. (2023). *Ventajas y desventajas de Java: una guía detallada*. https://innowise.com/es/blog/benefits-and-drawbacks-of-java/
35. La Moncloa. (2022, 17 de junio). *El MITECO amplía en 505 millones € el programa de ayudas al autoconsumo, almacenamiento y renovables térmicas*. https://www.lamoncloa.gob.es/serviciosdeprensa/notasprensa/transicion-ecologica/Paginas/2022/170622-ayudasautoconsumo.aspx
36. Luthander, R., Widén, J., Nilsson, D., & Palm, J. (2015). *Photovoltaic selfconsumption in buildings*: A review. Applied Energy, 142, 80-94. http://liu.diva-portal.org/smash/get/diva2:783948/FULLTEXT01
37. Madrid360. (2022). *Guía de consumo inteligente*. Ayuntamiento de Madrid. https://www.madrid360.es/wp-content/uploads/2022/09/guia_consumo_v2.pdf
38. Mathews, I., Xu, B., He, W., Barreto, V., Buonassisi, T., & Peters, I. M. (2020). *Techno-economic model of a second-life energy storage system for utility-scale solar power considering Li-ion calendar and cycle aging*. *arXiv*. https://arxiv.org/abs/2003.03216
39. Ministerio para la Transición Ecológica y el Reto Demográfico (MITECO). (2024, 24 de septiembre). *Plan Nacional Integrado de Energía y Clima (PNIEC) 2023-2030. Actualización 2024* [PDF]. https://miteco-prod.adobecqms.net/content/dam/miteco/es/energia/files-1/pniec-2023-2030/PNIEC_2024_240924.pdf
(Página resumen: https://www.miteco.gob.es/es/energia/estrategia-normativa/pniec-23-30.html
40. Ministerio para la Transición Ecológica y el Reto Demográfico (MITECO). (2025, 5 de agosto). *El MITECO destina 148 millones a 199 proyectos pioneros con almacenamiento de agrivoltaica, renovables en infraestructuras y bomba de calor*. https://www.miteco.gob.es/es/prensa/ultimas-noticias/2025/agosto/el-miteco-destina-148-millones-a-199-proyectos-pioneros-con-alma.html
41. Moa, E. H. Y., & Go, Y. L. (2023). *Large-scale energy storage system: safety and risk assessment*. *Sustainable Energy Research, 10*, Article 13. https://doi.org/10.1186/s40807-023-00082-z
42. Motor.es. (s.f.). *¿Qué es el SoC (State of Charge)?*. https://www.motor.es/que-es/state-of-charge-soc
43. Namor, E., Sossan, F., Cherkaoui, R., & Paolone, M. (2018). *Control of battery storage systems for the simultaneous provision of multiple services*. arXiv. https://arxiv.org/abs/1803.00978
44. Naturgy. (s.f.). *Precio de la luz*. https://www.naturgy.es/hogar/luz/precio_de_la_luz

45. NREL. (s.f.). *Solar + Storage Analysis*. https://www.nrel.gov/solar/market-research-analysis/solar-plus-storage-analysis.html
46. OMIE. (2024). *Precios horarios del mercado eléctrico*. Operador del Mercado Ibérico de Energía. https://www.omie.es/
47. United Nations (ONU). (2015). *Transformar nuestro mundo: la Agenda 2030 para el Desarrollo Sostenible*. ONU Publicaciones. https://sdgs.un.org/es/goals
48. Perfecta Energía. (2024). *¿Placas solares con batería o sin batería?* https://perfectaenergia.com/placas-solares-con-bateria-o-sin-bateria/
49. Plan Nacional Integrado de Energía y Clima (PNIEC). (2021). *Estrategia 2021-2030*. Ministerio para la Transición Ecológica y el Reto Demográfico. https://www.miteco.gob.es
50. PV Magazine España. (2024, 17 de junio). *Autoconsumo doméstico en 2024: los españoles ahorran más de 1 000 euros al año con una instalación media de 11 paneles*. PV Magazine. https://www.pv-magazine.es/comunicados/autoconsumo-domestico-en-2024-los-espanoles-ahorran-mas-de-1-000-euros-al-ano-en-su-factura-con-una-instalacion-media-de-11-paneles
51. PV Magazine España. (2025). *Un nuevo estudio cuantifica el ahorro que las baterías solares ofrecen a los propietarios*. https://www.pv-magazine.es/comunicados/un-nuevo-estudio-cuantifica-el-ahorro-que-las-baterias-solares-ofrecen-a-los-propietarios/
52. Rabobank. (2025, 18 de marzo). *Battery energy storage systems: The foundations of a resilient energy transition*. https://www.rabobank.com
53. Red Eléctrica de España (REE). (2024a). *Datos de generación en tiempo real*. https://www.ree.es/es/datos
54. Red Eléctrica de España (REE). (2024b, 3 de diciembre**).** *Análisis ESIOS 1739: Datos del 3 de diciembre de 2024 (agrupación por hora)* ESIOS. https://www.esios.ree.es/es/analisis/1739?vis=1&start_date=03-12-2024T00%3A00&end_date=03-12-2024T23%3A55&compare_start_date=02-12-2024T00%3A00&groupby=hour
55. Red Eléctrica de España (REE). (2024c, 3 de diciembre). *Análisis ESIOS 1001: Datos del 3 de diciembre de 2024 (agrupación por hora)* ESIOS. https://www.esios.ree.es/es/analisis/1001?vis=1&start_date=03-12-2024T00%3A00&end_date=03-12-2024T23%3A55&compare_start_date=02-12-2024T00%3A00&groupby=hour
56. Red Eléctrica de España (REE). (2025a, agosto). *Análisis ESIOS 2044: Curvas de oferta — 6 de agosto de 2025* ESIOS. https://www.esios.ree.es/es/analisis/2044?vis=1&start_date=06-08-2025T00%3A00&end_date=06-08-2025T23%3A55&compare_start_date=05-08-2025T00%3A00&groupby=hour
57. Red Eléctrica de España (REE). (2025b, noviembre). *Análisis ESIOS 1006: Consumo eléctrico — 1–30 noviembre de 2025 (agrupación por hora)* ESIOS. https://www.esios.ree.es/es/analisis/1006?vis=1&start_date=18-08-2025T00%3A00&end_date=18-08-2025T23%3A55&compare_start_date=17-08-2025T00%3A00&groupby=hour
58. Renewable Energy Policy Network for the 21st Century (REN21). (2022). *Renewables 2022 Global Status Report. REN21 Publications*.

https://www.ren21.net/reports/global-status-report/
59. Repsol. (2025). *Guía sobre la compensación de excedentes*. https://www.repsol.es/particulares/asesoramiento-consumo/guia-sobre-la-compensacion-de-excedentes/
60. Salvis-E. (2024). *Cómo las baterías para paneles solares ayudan a reducir la huella de carbono*. https://salvis-e.com/como-las-baterias-para-paneles-solares-ayudan-a-reducir-la-huella-de-carbono/
61. Selectra. (s.f.). *Franjas o Tramos Horarios de la Luz: punta, valle y llano*. https://selectra.es/energia/info/que-es/discriminacion-horaria
62. Shahsavari, A., & Akbari, M. (2018). Potential of solar energy in developing countries for reducing energy-related emissions. Renewable and Sustainable Energy Reviews, 90, 275–291.https://iranarze.ir/storage/uploads/2018/07/E8301-IranArze.pdf
63. Solarenergía Familias. (2023). *Autoconsumo con baterías – Almacenamiento*. https://solarfam.com/almacenamiento/
64. Solarenergía Familias. (2024). *¿Qué factores hacen rentable un sistema de almacenamiento de energía?* https://solarfam.com/que-factores-hacen-rentable-un-sistema-de-almacenamiento-de-energia/
65. SolarPower Europe. (2023). *EU market outlook for solar power 2023–2027*. SolarPower Europe. https://www.solarpowereurope.org/insights/outlooks/eu-market-outlook-for-solar-power-2025-mid-year-analysis
66. Sonnen, (2023). *SonnenBatterie y redes energéticas comunitarias*. www.sonnen.de
67. Sonnova. (2024). *¿Vale la pena el almacenamiento de energía en baterías solares?*. https://www.sunnova.com/es/watts-up/vale-la-pena-una-bateria
68. SotySolar. (2024, septiembre 16). *Sistemas de gestión energética para maximizar la eficiencia en tu hogar*. https://sotysolar.es/blog/sistemas-de-gestion-energetica
69. Tesla, 2023. *Powerwall y soluciones de almacenamiento doméstico*. www.tesla.com
70. TOPS Infosolutions. (2023). *Why has Java become so popular?* https://www.tops-int.com/blog/why-has-java-become-so-popular
71. TarifasGasLuz. (2025, 5 de febrero). *¿Qué son las tarifas de luz con precio indexado?* https://tarifasgasluz.com/comparador/precios-indexados
72. TotalEnergies. (2025, 17 de junio). *Conoce los tramos horarios de la tarifa 2.0 TD*. https://www.totalenergies.es/es/pymes/blog/tramos-horarios-20-td
73. UN-Habitat. (2020). World Cities Report 2020: The Value of Sustainable Urbanization. UN-Habitat Publications. https://unhabitat.org/worldcities-report-2020
74. Vallesolar. (2023). *Energía fotovoltaica y reducción de las emisiones de CO_2*. https://vallesolar.es/energia-fotovoltaica-y-reduccion-de-las-emisiones-de-co2/

ANEXO I: CÓDIGO DEL CÁLCULO DE CONSUMO

```
package es.carlota.controlbateria.clases;


import java.util.Date;


/**
 *
 * @author Carlota
 */
public class ConsumoRealHora {
    private Date hora = null;
    private double consMedio = 0;
    private double porcConsumo = 0;


    public ConsumoRealHora() {
    }
    public ConsumoRealHora(Date hora, double consMedio, double
porcConsumo) {
        this.hora = hora;
        this.consMedio = consMedio;
        this.porcConsumo = porcConsumo;
    }
```

```
    public void setHora(Date hora) {
        this.hora = hora;
    }
    public Date getHora() {
        return this.hora;
    }
    public void setPorcentajeConsumo(double porcConsumo) {
        this.porcConsumo = porcConsumo;
    }
    public void setConsMedio(double consMedio) {
        this.consMedio = consMedio;
    }
    public double getConsumo() {
        return Mates.Redondear((consMedio/1000)*porcConsumo,5);
    }

    public double getConsMedio() {
        return consMedio;
    }
}
```

ANEXO II: CÓDIGO DEL CÁLCULO DE GENERACIÓN

```
package es.carlota.controlbateria.clases;


import java.util.Date;


/**
 *
 * @author Carlota
 */
public class GeneracionRealHora {
    private Date hora = null;
    private double panelesEspana = 0;
    private double numeroPaneles = 0;
    private double generacionTotal = 0; //en MW


    //CONSTRUCTOR
    public GeneracionRealHora () {
     }
    public GeneracionRealHora (Date hora, double panelesEspana,
double numeroPaneles, double generacionTotal) {
        this.hora = hora;
        this.panelesEspana = panelesEspana;
        this.numeroPaneles = numeroPaneles;
```

```
        this.generacionTotal = generacionTotal;
    }

    //PROPIEDADES
    public void setHora (Date hora){
        this.hora = hora;
    }
    public Date getHora(){
        return this.hora;
    }

    public void setPanelesEspana (double panelesEspana){
        this.panelesEspana = panelesEspana;
    }

    public void setNumeroPaneles (double numeroPaneles){
        this.numeroPaneles = numeroPaneles;
    }

    public void setGeneracionTotal (double generacionTotal){
        this.generacionTotal = generacionTotal;
    }
    public double getGeneracionTotal (){
```

```
        return
Mates.Redondear((this.generacionTotal/(1000*this.panelesEspana))*t
his.numeroPaneles,5);

    }

}
```

ANEXO III: CÓDIGO DEL CÁLCULO DEL PRECIO DE LA RED

```
package es.carlota.controlbateria.clases;
import java.util.Date;
/**
 *
 * @author Carlota
 */
public class PreciosRed {
    private Date hora;
    private double precioRed;
    private String grupo;

    public PreciosRed(){
    }
    public PreciosRed(Date hora, double precioRed){
        this.hora = hora;
        this.precioRed = precioRed;
        this.grupo = "";
    }
```

```
    //PROPIEDADES
    public void setHora (Date hora){
        this.hora = hora;
    }
    public Date getHora(){
        return this.hora;
    }

    //PROPIEDADES
    public void setPrecioRed (double precioRed){
        this.precioRed = precioRed;
    }
    public double getPrecioRed(){
        return this.precioRed/1000;
    }
    //PROPIEDADES
    public void setGrupo (String grupo){
        this.grupo = grupo;
    }
    public String getGrupo(){
        return this.grupo;
    }
}
```

ANEXO VI: CÓDIGO DEL CÁLCULO DEL PRECIO DE VENTA

```
package es.carlota.controlbateria.clases;


import java.util.Date;


/**
 *
 * @author Carlota
 */
public class PreciosVenta {
    private Date hora;
    private double precioVenta;
    private String grupo;


    public PreciosVenta() {
    }
    public PreciosVenta(Date hora, double precioVenta){
        this.hora = hora;
        this.precioVenta = precioVenta;
    }
    //PROPIEDADES
```

```
    public void setHora (Date hora){
        this.hora = hora;
     }
    public Date getHora(){
        return this.hora;
    }
     //PROPIEDADES
    public void setPrecioVenta (double precioVenta){
        this.precioVenta = precioVenta;
     }
    public double getPrecioVenta(){
        return this.precioVenta/1000;
    }
    //PROPIEDADES
    public void setGrupo (String grupo){
        this.grupo = grupo;
     }
    public String getGrupo(){
        return this.grupo;
    }
}
```

ANEXO V: CÓDIGO PARA CREAR EL REGISTRO ENERGÉTICO

```
package es.carlota.controlbateria.clases;


import java.text.SimpleDateFormat;

import java.util.Date;


/**

 *

 * @author Carlota

 */

public class RegistroEnergetico {

    //Crear registro

    private Date hora = null; //Solo guardamos la hora sin minutos
ni segundos

    private String horaCadena = null;

    private double consumo = 0;

    private double generacion = 0;

    private double diferencia = 0;

    private double precioRed = 0;

    private double precioVenta = 0;
```

```
    private double estadoBateria = 0;

    private double cargaBateria = 0;

    private double consumoRed = 0;

    private double costeCargaBateria = 0;

    private double costeConsumoRed = 0;

    private double energiaVendida = 0;

    private double beneficioVenta = 0;

    private double costeSinPanelesNiBateria = 0;

    private double costeSinBateria = 0;

    private String grupoPrecioRed = "";

    private String grupoPrecioVenta = "";


    //CONSTRUCTORES

    public RegistroEnergetico() {

    }

    public RegistroEnergetico(Date hora, double consumo, double
generacion, double precioRed, double precioVenta,

            double estadoBateria, double cargaBateria, double
consumoRed, double costeCargaBateria, double costeConsumoRed,

            double energiaVendida, double beneficioVenta, double
costeSinPanelesNiBateria, double costeSinBateria, String
grupoPrecioRed) {

        this.hora = hora;

        this.consumo = consumo;

        this.generacion = generacion;

        this.diferencia = generacion-consumo; //Calculamos de la
diferencia
```

```
        this.precioRed = precioRed;
        this.precioVenta = precioVenta;
        this.estadoBateria = estadoBateria;
        this.cargaBateria = cargaBateria;
        this.consumoRed = consumoRed;
        this.costeCargaBateria = costeCargaBateria;
        this.costeConsumoRed = costeConsumoRed;
        this.energiaVendida = energiaVendida;
        this.beneficioVenta = beneficioVenta;
        this.costeSinPanelesNiBateria = costeSinPanelesNiBateria;
        this.costeSinBateria = costeSinBateria;
        this.grupoPrecioRed = grupoPrecioRed;
    }


    //PROPUEDADES
    public void setHora(Date hora) {
        SimpleDateFormat formatoHora = new
SimpleDateFormat("dd/MM/yyyy HH:mm");
        this.hora = hora;
        this.horaCadena = formatoHora.format(hora);
    }
    public Date getHora() {
        return this.hora;
    }
    public String getHoraCadena() {
```

```
        return this.horaCadena;

    }

    public void setConsumo(double consumo) {

        this.consumo = consumo;

        this.diferencia    =    Mates.Redondear(this.generacion    -
consumo, 5);

    }

    public double getConsumo() {

        return this.consumo;

    }

    public void setGeneracion(double generacion) {

        this.generacion = generacion;

        this.diferencia     =     Mates.Redondear(generacion     -
this.consumo, 5);

    }

    public double getGeneracion() {

        return this.generacion;

    }

    public double getDiferencia() {

        return this.diferencia;

    }


    public double getPrecioRed() {

        return precioRed;

    }
```

```
public void setPrecioRed(double precioRed) {
    this.precioRed = precioRed;
}

public double getPrecioVenta() {
    return precioVenta;
}

public void setPrecioVenta(double precioVenta) {
    this.precioVenta = precioVenta;
}

public double getEstadoBateria() {
    return estadoBateria;
}

public void setEstadoBateria(double estadoBateria) {
    this.estadoBateria = Mates.Redondear(estadoBateria, 5);
}

public double getCargaBateria() {
    return cargaBateria;
}
```

```
public void setCargaBateria(double cargaBateria) {
    this.cargaBateria = cargaBateria;
}

public double getConsumoRed() {
    return consumoRed;
}

public void setConsumoRed(double consumoRed) {
    this.consumoRed = consumoRed;
}

public double getCosteCargaBateria() {
    return costeCargaBateria;
}

public void setCosteCargaBateria(double costeCargaBateria) {
    this.costeCargaBateria = costeCargaBateria;
}

public double getCosteConsumoRed() {
    return costeConsumoRed;
}
```

```
public void setCosteConsumoRed(double costeConsumoRed) {
    this.costeConsumoRed = costeConsumoRed;
}

public double getEnergiaVendida() {
    return energiaVendida;
}

public void setEnergiaVendida(double energiaVendida) {
    this.energiaVendida = energiaVendida;
}

public double getBeneficioVenta() {
    return beneficioVenta;
}

public void setBeneficioVenta(double beneficioVenta) {
    this.beneficioVenta = beneficioVenta;
}

public double getCosteSinPanelesNiBateria() {
    return costeSinPanelesNiBateria;
}
```

```
    public void setCosteSinPanelesNiBateria(double
costeSinPanelesNiBateria) {
        this.costeSinPanelesNiBateria = costeSinPanelesNiBateria;
    }

    public double getCosteSinBateria() {
        return costeSinBateria;
    }

    public void setCosteSinBateria(double costeSinBateria) {
        this.costeSinBateria = costeSinBateria;
    }

    public String getGrupoPrecioRed() {
        return grupoPrecioRed;
    }

    public void setGrupoPrecioRed(String grupoPrecioRed) {
        this.grupoPrecioRed = grupoPrecioRed;
    }

    public String getGrupoPrecioVenta() {
        return grupoPrecioVenta;
    }
```

```
    public void setGrupoPrecioVenta(String grupoPrecioVenta) {
        this.grupoPrecioVenta = grupoPrecioVenta;
    }

    public double getConsumoRedTotal() {
        return costeConsumoRed+costeCargaBateria;
    }

    public double getCosteTotal() {
        return (costeConsumoRed+costeCargaBateria)-beneficioVenta;
    }
    public String getTitulos() {
        return
"Contador;Dia/Hora;Consumo;Generacion;Diferencia;Precio Red;Grupo
Precio Red;Precio Venta;Grupo Precio Venta;Estado Bateria;Carga
Bateria;Coste Carga Bateria;Consumo Red;Coste Consumo Red;Coste
Consumo Red Total;Energia Vendida;Beneficio Venta;Coste Total;Coste
Sin Bateria;Coste Sin Paneles Ni Bateria";
    }
    public String getDatos(int contador) {
        SimpleDateFormat formatoHora = new
SimpleDateFormat("dd/MM/yyyy HH:mm");
        double costeConsumoRedTotal =
costeConsumoRed+costeCargaBateria;
        double costeTotal = costeConsumoRedTotal-beneficioVenta;
        return
String.format("%d;%s;%f;%f;%f;%f;%s;%f;%s;%f;%f;%f;%f;%f;%f;%f;%f;
%f;%f;%f",contador,formatoHora.format(hora),consumo,
```

```
generacion,diferencia,precioRed,grupoPrecioRed,precioVenta,grupoPr
ecioVenta,estadoBateria,cargaBateria,costeCargaBateria,consumoRed,
costeConsumoRed,costeConsumoRedTotal,energiaVendida,beneficioVenta
,costeTotal,costeSinBateria,costeSinPanelesNiBateria);

    }

}
```

ANEXO VI: CÓDIGO CON LOS VALORES GLOBALES UTILIZADOS EN TODA LA LÓGICA

```
package es.carlota.controlbateria.clases;


/**
 *
 * @author Carlota
 */
public class ValoresGlobales {
    private double consMedio = 3750000;
    private double panelesEspana = 82;
    private double numeroPaneles = 10;
    private double capacidadNominal = 10;

    public double getConsMedio() {
        return consMedio;
    }

    public double getPanelesEspana() {
        return panelesEspana;
    }
```

```
    public double getNumeroPaneles() {
        return numeroPaneles;
    }

    public double getCapacidadNominal() {
        return capacidadNominal;
    }

    public void setConsMedio(double consMedio) {
        this.consMedio = consMedio;
    }

    public void setPanelesEspana(double panelesEspana) {
        this.panelesEspana = panelesEspana;
    }

    public void setNumeroPaneles(double numeroPaneles) {
        this.numeroPaneles = numeroPaneles;
    }

    public void setCapacidadNominal(double capacidadNominal) {
        this.capacidadNominal = capacidadNominal;
    }
}
```

ANEXO VII: CÓDIGO PARA INICIALIZAR LOS VALORES

```
package es.carlota.controlbateria.procesos;


import es.carlota.controlbateria.clases.ConsumoRealHora;
import es.carlota.controlbateria.clases.GeneracionRealHora;
import es.carlota.controlbateria.clases.PreciosRed;
import es.carlota.controlbateria.clases.PreciosVenta;
import es.carlota.controlbateria.clases.ValoresGlobales;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;


/**
 *
 * @author Carlota
 */
public class InicializaValoresCSV {
    ValoresGlobales valoresg;
    private ConsumoRealHora[] consumoReal;
    private GeneracionRealHora[] generacionReal;
    private PreciosRed[] preciosRed;
    private PreciosVenta[] preciosVenta;
```

```
    public InicializaValoresCSV(ValoresGlobales valoresg, String archivoConsumoReal, String archivoGeneracionReal, String archivoPrecioRed, String archivoPrecioVenta) {

        this.valoresg=valoresg;

        ConsumoRealLeer(archivoConsumoReal);

        GeneracionLeer(archivoGeneracionReal);

        PrecioRedLeer(archivoPrecioRed);

        PrecioVentaLeer(archivoPrecioVenta);


        /*

        1) Se ordena de menor a mayor

        2) divide el numero de elementos entre 3

        3) se asigna a X el resultante

        4) se asigna el grupo B a los X primeros elementos

        5) se asigna el grupo A a los X últimos elementos

        6) se asigna el grupo M al resto de elementos

        */

        AsignarGrupoPrecioRed();

        /*

        1) Se ordena de menor a mayor

        2) divide el numero de elementos entre 3

        3) se asigna a X el resultante

        4) se asigna el grupo A a los X últimos elementos

        5) se asigna el grupo B al resto de elementos

        */
```

```
        AsignarGrupoPrecioVenta();


    }


    private void ConsumoRealLeer(String archivoCSV) {

        LeerCSV leer = new LeerCSV(archivoCSV);

        consumoReal = new ConsumoRealHora[leer.lineas.size()];

        for (int contador = 1; contador < consumoReal.length;
contador++) {

            consumoReal[contador-1] = new ConsumoRealHora();

            String[]                    columna                    =
leer.lineas.get(contador).toString().split(";");

            try {

                String cadenaFecha = columna[5].substring(0,10);

                cadenaFecha += " "+columna[5].substring(11,16);

                Date     fecha     =     ConvertirAFecha("yyyy-M-dd
HH:mm",cadenaFecha);

                consumoReal[contador-1].setHora(fecha);

            } catch (ParseException pe) {

                System.out.println(pe.getMessage());

            }

            consumoReal[contador-
1].setPorcentajeConsumo(Float.parseFloat(columna[4]));

            consumoReal[contador-
1].setConsMedio(valoresg.getConsMedio());

        }

    }
```

```
    private void GeneracionLeer(String archivoCSV) {

        int colHora = -1;

        int colGeneracion = -1;

        int iniTitulos = -1;

        LeerCSV leer = new LeerCSV(archivoCSV);

        for (int linea = 0; linea < leer.lineas.size() && colGeneracion==-1 && colHora==-1; linea++) {

            String[] columna = leer.lineas.get(linea).toString().split(";");

            for (int col = 0; col < columna.length && (colGeneracion==-1 || colHora==-1); col++ ) {

                switch(columna[col]) {

                    case "datetime":

                        colHora = col;

                        iniTitulos = linea;

                        break;

                    case "value":

                        colGeneracion = col;

                        break;

                    default:

                        //No hace nada

                }

            }

        }

        generacionReal = new GeneracionRealHora[leer.lineas.size()-iniTitulos];
```

```
        for (int contador = iniTitulos+1; contador < generacionReal.length-iniTitulos; contador++) {

            generacionReal[contador-1] = new GeneracionRealHora();

            String[] columna = leer.lineas.get(contador).toString().split(";");

            try {

                Date fecha = null;

                if (columna[colHora].indexOf("T")> 0) {

                    String cadenaFecha = columna[colHora].substring(0,10);

                    cadenaFecha += " "+columna[colHora].substring(11,16);

                    fecha = ConvertirAFecha("yyyy-M-dd HH:mm",cadenaFecha);

                } else {

                    fecha = ConvertirAFecha("yyyy-M-dd HH:mm",columna[colHora]);

                }

                generacionReal[contador-1].setHora(fecha);

            } catch (ParseException pe) {

                System.out.println(pe.getMessage());

            }

            generacionReal[contador-1].setGeneracionTotal(Float.parseFloat(columna[colGeneracion]));

            generacionReal[contador-1].setNumeroPaneles(valoresg.getNumeroPaneles());

            generacionReal[contador-1].setPanelesEspana(valoresg.getPanelesEspana());

        }
```

```
    }


    private void PrecioRedLeer(String archivoCSV) {

        LeerCSV leer = new LeerCSV(archivoCSV);

        preciosRed = new PreciosRed[leer.lineas.size()];

        for (int contador = 1; contador < preciosRed.length;
contador++) {

            preciosRed[contador-1] = new PreciosRed();

            String[]                    columna                    =
leer.lineas.get(contador).toString().split(";");

            try {

                String cadenaFecha = columna[5].substring(0,10);

                cadenaFecha += " "+columna[5].substring(11,16);

                Date     fecha     =     ConvertirAFecha("yyyy-M-dd
HH:mm",cadenaFecha);

                preciosRed[contador-1].setHora(fecha);

            } catch (ParseException pe) {

                System.out.println(pe.getMessage());

            }

            preciosRed[contador-
1].setPrecioRed(Float.parseFloat(columna[4]));

        }

    }


    private void PrecioVentaLeer(String archivoCSV) {

        LeerCSV leer = new LeerCSV(archivoCSV);
```

```
        preciosVenta = new PreciosVenta[leer.lineas.size()];

        for (int contador = 1; contador < preciosVenta.length;
contador++) {

            preciosVenta[contador-1] = new PreciosVenta();

            String[]                    columna                    =
leer.lineas.get(contador).toString().split(";");

            try {

                String cadenaFecha = columna[5].substring(0,10);

                cadenaFecha += " "+columna[5].substring(11,16);

                Date     fecha     =     ConvertirAFecha("yyyy-M-dd
HH:mm",cadenaFecha);

                preciosVenta[contador-1].setHora(fecha);

            } catch (ParseException pe) {

                System.out.println(pe.getMessage());

            }

            preciosVenta[contador-
1].setPrecioVenta(Float.parseFloat(columna[4]));

        }

    }


    private void AsignarGrupoPrecioRed() {

        PreciosRed aux;

        int numeroElementos = 0;

        int elementosTotal = 0;

        for (int cont1 = 0; cont1 < preciosRed.length; cont1++) {

            if (preciosRed[cont1]!=null) {
```

```
                numeroElementos++;
                for (int cont2 = cont1; cont2 < preciosRed.length;
cont2++) {
                    if (preciosRed[cont2]!=null) {
                        if   (preciosRed[cont1].getPrecioRed()   >
preciosRed[cont2].getPrecioRed()) {
                            aux = preciosRed[cont1];
                            preciosRed[cont1] = preciosRed[cont2];
                            preciosRed[cont2] = aux;
                        }
                    }
                }
            }
        }
        elementosTotal = numeroElementos;
        numeroElementos = elementosTotal / (int) 3;
        for (int cont1 = 0; cont1 < numeroElementos; cont1++) {
            preciosRed[cont1].setGrupo("B");
        }
        elementosTotal=0;
        for (int cont1 = preciosRed.length-1; cont1 >= 0; cont1--)
{
            if (preciosRed[cont1]!=null) {
                if (preciosRed[cont1].getGrupo()==null) {
                    if (elementosTotal < numeroElementos) {
                        preciosRed[cont1].setGrupo("A");
```

```
                    } else {
                        preciosRed[cont1].setGrupo("M");
                    }
                    elementosTotal++;
                }
            }
        }
    }

    private void AsignarGrupoPrecioVenta() {
        PreciosVenta aux;
        int numeroElementos = 0;
        int elementosTotal = 0;
        for (int cont1 = 0; cont1 < preciosVenta.length; cont1++) {
            if (preciosVenta[cont1]!=null) {
                numeroElementos++;
                for (int cont2 = cont1; cont2 < preciosVenta.length;
cont2++) {
                    if (preciosVenta[cont2]!=null) {
                        if (preciosVenta[cont1].getPrecioVenta() >
preciosVenta[cont2].getPrecioVenta()) {
                            aux = preciosVenta[cont1];
                            preciosVenta[cont1]                    =
preciosVenta[cont2];
                            preciosVenta[cont2] = aux;
                        }
```

```
                    }
                }
            }
        }
        elementosTotal = numeroElementos;
        numeroElementos = elementosTotal / (int) 3;
        elementosTotal=0;
        for (int cont1 = preciosVenta.length-1; cont1 >= 0; cont1-
-) {
            if (preciosVenta[cont1]!=null) {
                if (preciosVenta[cont1].getGrupo()==null) {
                    if (elementosTotal < numeroElementos) {
                        preciosVenta[cont1].setGrupo("A");
                    } else {
                        preciosVenta[cont1].setGrupo("B");
                    }
                    elementosTotal++;
                }
            }
        }
    }


    private Date ConvertirAFecha(String formato, String fecha)
throws ParseException {
        SimpleDateFormat formatea = new SimpleDateFormat(formato);
```

```
        return formatea.parse(fecha);
    }

    public ConsumoRealHora[] getConsumoReal() {
        return consumoReal;
    }

    public GeneracionRealHora[] getGeneracionReal() {
        return generacionReal;
    }

    public PreciosRed[] getPreciosRed() {
        return preciosRed;
    }

    public PreciosVenta[] getPreciosVenta() {
        return preciosVenta;
    }
}
```

ANEXO VIII: CÓDIGO PARA LEER EL CSV CON LOS DATOS

```
package es.carlota.controlbateria.procesos;


import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;



/**
 *
 * @author Carlota
 */
public class LeerCSV  {
    ArrayList lineas = new ArrayList();
    public LeerCSV(String nombreArchivo) {
        String texto = "";
        try {
            FileReader fr = new FileReader(nombreArchivo);
            BufferedReader br = new BufferedReader(fr);
            while (br.ready()) {
```

```
                texto = br.readLine();

                this.lineas.add(texto);

            }

            br.close();

            fr.close();

        } catch (IOException ex) {

            System.out.println(ex.getMessage());

        }

    }

}
```

ANEXO IX: CÓDIGO CON LA LÓGICA COMPLETA

```
package es.carlota.controlbateria.procesos;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import es.carlota.controlbateria.clases.ConsumoRealHora;
import es.carlota.controlbateria.clases.GeneracionRealHora;
import es.carlota.controlbateria.clases.Mates;
import es.carlota.controlbateria.clases.PreciosRed;
import es.carlota.controlbateria.clases.PreciosVenta;
import es.carlota.controlbateria.clases.RegistroEnergetico;
import es.carlota.controlbateria.clases.ValoresGlobales;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

/**
 *
 * @author Carlota
 */
public class ProcesarRegistro {
    private ValoresGlobales valoresG;
```

```
    private ConsumoRealHora[] consumoReal = null;

    private GeneracionRealHora[] generacionReal = null;

    private PreciosRed[] preciosRed = null; //estan ordenadas por
la fecha que se importó el csv

    private PreciosVenta[] preciosVenta = null; //pasamos los CSV

    private Map<Integer, RegistroEnergetico> mapRegistroE = new
HashMap(); //mapeado, le decimos que tipo va a ser el indice(por el
cual identificamos la linea)-HORA Y DIA y que informacion va a
contener este registro- clase que contiene toda la info=registro
energetico

    private double energiaVentaFutura=0;



    //CONSTRUCTORES

    public ProcesarRegistro(){ //pasamos 4 tablas y despues
generamos el resultante del registro energetico



    }

    public        ProcesarRegistro(ValoresGlobales        valoresG,
ConsumoRealHora[] consumoReal, GeneracionRealHora[] generacionReal,
PreciosRed[] preciosRed, PreciosVenta[] preciosVenta){

        this.valoresG = valoresG;

        this.consumoReal = consumoReal;

        this.generacionReal = generacionReal;

        this.preciosRed = preciosRed;

        this.preciosVenta = preciosVenta;



    }
```

```
    public boolean CrearRegistroEnergetico(){

        boolean procesoOK = true; //para ver que ha funcionado y que se ha creado bien el mapeado y tenemos el registro -> devuelve true, sino false

        Date hora;

        int contador = 0;

        //variable tipo RegistroEnergetico es una clase por eso puedes crear una variable de este tipo

        for (ConsumoRealHora consumo1 : consumoReal) {

            //recorrer tabla ppal, en java el indice del array empieza en 0

            if (consumo1!=null) {

                hora = consumo1.getHora(); //coge las horas de consumo por si no coincidiesen todas las de los 4 csv

                RegistroEnergetico registroE = new RegistroEnergetico();//variable tipo RegistroEnergetico es una clase por eso puedes crear una variable de este tipo

                //hora(indice)

                registroE.setHora(hora); //ASIGNO AL REGISTRO

                //Consumo

                registroE.setConsumo(consumo1.getConsumo());  //no necesito guardarlo en una variable, puedo obtenerla de la tabla

                //Generacion

                registroE.setGeneracion(DevuelveGeneracion(hora));

                //PrecioRed

                PreciosRed valorPrecioRedHora = DevuelvePrecioRed(hora);

                if (valorPrecioRedHora!=null) {
```

```
registroE.setPrecioRed(valorPrecioRedHora.getPrecioRed());


registroE.setGrupoPrecioRed(valorPrecioRedHora.getGrupo());
                } else {
                    registroE.setPrecioRed(0);
                    registroE.setGrupoPrecioRed("");
                }
                //PrecioVenta
                PreciosVenta        valorPrecioVentaHora        =
DevuelvePrecioVenta(hora);
                if (valorPrecioVentaHora!=null) {


registroE.setPrecioVenta(valorPrecioVentaHora.getPrecioVenta());
//el set es el de la clase PreciosVenta


registroE.setGrupoPrecioVenta(valorPrecioVentaHora.getGrupo());
//el set es el de la clase PreciosVenta
                } else {
                    registroE.setPrecioVenta(0);
                    registroE.setGrupoPrecioVenta("");
                }
                //Estado Bateria
                registroE.setEstadoBateria((double)
(valoresG.getCapacidadNominal() * .2));


registroE.setCosteSinPanelesNiBateria(registroE.getPrecioRed()*reg
istroE.getConsumo());
                //GUARDO EL REGISTRO ENERGETICO A LA HORA QUE
CORRESPONDE
```

```
                mapRegistroE.put(contador, registroE); //
                contador++;
            }
        }
        return procesoOK;
    }

    public                          ObservableList<RegistroEnergetico>
RecalculaRegistroEnergetico(){
        int contador = 0;
        int contadorAux = 0;
        int estadoDiferencia=-1;
        int estadoDiferenciaAnt=-1;
        boolean titulos=false;
        while (contador < mapRegistroE.size()) {
            mapRegistroE.get(contador).setEnergiaVendida(0);
            mapRegistroE.get(contador).setBeneficioVenta(0);
            mapRegistroE.get(contador).setCargaBateria(0);
            mapRegistroE.get(contador).setCosteCargaBateria(0);
            mapRegistroE.get(contador).setConsumoRed(0);
            mapRegistroE.get(contador).setCosteConsumoRed(0);
            if (mapRegistroE.get(contador).getDiferencia()<0) {
                estadoDiferencia = 1;
                contadorAux=contador;
```

```
                if (estadoDiferenciaAnt != estadoDiferencia &&
estadoDiferenciaAnt!=-1) {

                    if (estadoDiferenciaAnt==2) {

                        AjustarBateriaGenMayorCons(contador-1);
//Cuando Generacion sea MAYOR que consumo

                    }

                }

                estadoDiferenciaAnt=estadoDiferencia;

                contador = ConsumoMayorGeneracion(contador,true);

            }                       else                       if
(mapRegistroE.get(contador).getDiferencia()>0) {

                mapRegistroE.get(contador).setCosteSinBateria(0);

                //Cuando la generacion es mayor que lo que se
consume

                estadoDiferencia = 2;

                contadorAux=contador;

                if (estadoDiferenciaAnt != estadoDiferencia &&
estadoDiferenciaAnt!=-1) {

                    if (estadoDiferenciaAnt==1) {

                        contadorAux                            =
AjustarBateriaConsMayorGen(contador-1); //Cuando consumo sea MAYOR
que genaracion

                    }

                }

                estadoDiferenciaAnt=estadoDiferencia;

                contador = GeneracionMayorConsumo(contador, true);

            } else {

                mapRegistroE.get(contador).setCosteSinBateria(0);
```

```
                //Cuando consumo y generacion son iguales

                estadoDiferencia = 0;

                estadoDiferenciaAnt = 0;

                if (contador > 0)
mapRegistroE.get(contador).setEstadoBateria(mapRegistroE.get(conta
dor-1).getEstadoBateria());

                contadorAux=contador;

            }

            contador++;

            if (contador==mapRegistroE.size()) {

                if (estadoDiferencia == 1) {

                    contadorAux =
AjustarBateriaConsMayorGen(contador-1);

                } else {

                    contadorAux =
AjustarBateriaGenMayorCons(contador-1);

                }

                if (contadorAux<contador-1) contador =
contadorAux+1;

            }

        }

        ObservableList<RegistroEnergetico> datos =
FXCollections.observableArrayList(mapRegistroE.values());

        return datos;

    }


    public void CrearArchivo() {
```

```
        if (mapRegistroE.size()>0) {

            WriteCSV writeCSV = new
WriteCSV("./resultado.csv",mapRegistroE);

        }

    }


    private int GeneracionMayorConsumo(int contador, boolean
retroceder) {

        int contadorR;

        int contadorI;

        int contadorF;

        int contadorAux;

        double diferencia;

        double estadoAnterior;

        double estadoActual;

        double energiaVenta;

        double precioAux;

        mapRegistroE.get(contador).setEnergiaVendida(0);

        mapRegistroE.get(contador).setBeneficioVenta(0);

        mapRegistroE.get(contador).setCargaBateria(0);

        mapRegistroE.get(contador).setCosteCargaBateria(0);

        mapRegistroE.get(contador).setConsumoRed(0);

        mapRegistroE.get(contador).setCosteConsumoRed(0);

        estadoAnterior =(double)
.2*this.valoresG.getCapacidadNominal();
```

```
        if (contador > 0) estadoAnterior =
mapRegistroE.get(contador-1).getEstadoBateria();

        diferencia =
Math.abs(mapRegistroE.get(contador).getDiferencia());

        if (estadoAnterior+diferencia<=(double)
.8*this.valoresG.getCapacidadNominal()) {


mapRegistroE.get(contador).setEstadoBateria(estadoAnterior+diferen
cia);

            if (energiaVentaFutura > 0) {

                if (mapRegistroE.get(contador).getEstadoBateria()-
(energiaVentaFutura/(double) .9) >= (double)
.2*valoresG.getCapacidadNominal()) {

                    energiaVenta =energiaVentaFutura/(double) .9;


mapRegistroE.get(contador).setEstadoBateria(mapRegistroE.get(conta
dor).getEstadoBateria()-energiaVenta);


mapRegistroE.get(contador).setBeneficioVenta(mapRegistroE.get(cont
ador).getPrecioVenta()*energiaVenta);


mapRegistroE.get(contador).setEnergiaVendida(energiaVenta);

                    energiaVentaFutura=0;

                } else {

                    energiaVenta
=mapRegistroE.get(contador).getEstadoBateria()-(double)
.2*valoresG.getCapacidadNominal();


mapRegistroE.get(contador).setEstadoBateria(mapRegistroE.get(conta
dor).getEstadoBateria()-(energiaVenta*(double).9));


mapRegistroE.get(contador).setBeneficioVenta(mapRegistroE.get(cont
ador).getPrecioVenta()*energiaVenta);
```

```
mapRegistroE.get(contador).setEnergiaVendida(energiaVenta);

                    energiaVentaFutura-=(energiaVenta*(double).9);

                }

            }

        } else {

            if (retroceder) {

                energiaVentaFutura  =  (estadoAnterior+diferencia)-
(double) .8*this.valoresG.getCapacidadNominal();

                contadorR = contador;

                contadorF = contador;

                contadorI = 0;

                contadorAux = 0;

                while        (contadorR        >        0        &&
mapRegistroE.get(contadorR).getDiferencia()>0                    &&
mapRegistroE.get(contadorR).getBeneficioVenta()==0) {

                    contadorR-=1;

                }

                if
(mapRegistroE.get(contadorR).getDiferencia()<=0)    contadorI    =
contadorR+1;

                else                                              if
(mapRegistroE.get(contadorR).getBeneficioVenta()!=0)  contadorI  =
contadorR+1;

                else contadorI=0;

                contadorAux=contadorI;


precioAux=mapRegistroE.get(contadorI).getPrecioVenta();

                while (contadorI <= contadorF) {
```

```
                    if
(mapRegistroE.get(contadorI).getPrecioVenta() > precioAux) {

                        contadorAux=contadorI;


precioAux=mapRegistroE.get(contadorI).getPrecioVenta();

                    }

                    contadorI++;

                }

                contador = contadorAux;

                diferencia                                          =
Math.abs(mapRegistroE.get(contador).getDiferencia());

                estadoActual    =       mapRegistroE.get(contador-
1).getEstadoBateria()+diferencia;

                if
(mapRegistroE.get(contador).getGrupoPrecioVenta().equals("A")) {

                    double          energiaVentaCalculada          =
(mapRegistroE.get(contador-1).getEstadoBateria()+diferencia);

                    if    (energiaVentaCalculada    <    (double)
.5*valoresG.getCapacidadNominal()) {

                        energiaVentaCalculada                       =
Mates.Redondear(energiaVentaCalculada          -          (double)
.2*valoresG.getCapacidadNominal(), 5);

                        while       (energiaVentaCalculada       <
(energiaVentaFutura/ (double) .9) && contador < contadorF) {

                            contador++;

                            diferencia                              =
Math.abs(mapRegistroE.get(contador).getDiferencia());

                            energiaVentaCalculada                   =
(mapRegistroE.get(contador-1).getEstadoBateria()+diferencia);
```

```
                        energiaVentaCalculada                    =
Mates.Redondear(energiaVentaCalculada          -          (double)
.2*valoresG.getCapacidadNominal(), 5);

                    }

                    if        (energiaVentaCalculada        >
(energiaVentaFutura/ (double) .9)) {

                        diferencia                               =
Math.abs(mapRegistroE.get(contador).getDiferencia());

                        energiaVentaCalculada                    =
(mapRegistroE.get(contador-1).getEstadoBateria()+diferencia);


mapRegistroE.get(contador).setEstadoBateria(energiaVentaCalculada-
(energiaVentaFutura/ (double) .9));


mapRegistroE.get(contador).setBeneficioVenta(mapRegistroE.get(cont
ador).getPrecioVenta()*(energiaVentaFutura/ (double) .9));


mapRegistroE.get(contador).setEnergiaVendida(energiaVentaFutura/
(double) .9);

                    } else {

                        diferencia                               =
Math.abs(mapRegistroE.get(contadorF-1).getDiferencia());

                        energiaVentaCalculada                    =
(mapRegistroE.get(contadorF-2).getEstadoBateria()+diferencia);

                        if      (energiaVentaCalculada      >
(energiaVentaFutura/ (double) .9)) {

                            mapRegistroE.get(contadorF-
1).setEstadoBateria(energiaVentaCalculada-(energiaVentaFutura/
(double) .9));

                            mapRegistroE.get(contadorF-
1).setBeneficioVenta(mapRegistroE.get(contadorF-
1).getPrecioVenta()*(energiaVentaFutura/ (double) .9));
```

```
                                    mapRegistroE.get(contadorF-
1).setEnergiaVendida(energiaVentaFutura/ (double) .9);

                                    contador = contadorF-1;

                                }

                            }

                        } else {

                            energiaVenta    =    (energiaVentaCalculada-
((double) .5*valoresG.getCapacidadNominal()))/ (double) .9;


mapRegistroE.get(contador).setEstadoBateria((double)
.5*valoresG.getCapacidadNominal());


mapRegistroE.get(contador).setBeneficioVenta(mapRegistroE.get(cont
ador).getPrecioVenta()*energiaVenta);


mapRegistroE.get(contador).setEnergiaVendida(energiaVenta);

                        }

                        energiaVentaFutura=0;

                    } else {

                        if   (estadoActual-(energiaVentaFutura/(double)
.9) >= (double) .2*valoresG.getCapacidadNominal()) {

                            energiaVenta   =energiaVentaFutura/(double)
.9;


mapRegistroE.get(contador).setEstadoBateria(estadoActual-
energiaVenta);


mapRegistroE.get(contador).setBeneficioVenta(mapRegistroE.get(cont
ador).getPrecioVenta()*energiaVenta);


mapRegistroE.get(contador).setEnergiaVendida(energiaVenta);
```

```
                        energiaVentaFutura=0;

                    } else {

                        energiaVenta        =estadoActual-(double)
.2*valoresG.getCapacidadNominal();


mapRegistroE.get(contador).setEstadoBateria(estadoActual-
(energiaVenta*(double).9));


mapRegistroE.get(contador).setBeneficioVenta(mapRegistroE.get(cont
ador).getPrecioVenta()*energiaVenta);


mapRegistroE.get(contador).setEnergiaVendida(energiaVenta);

                        energiaVentaFutura-
=(energiaVenta*(double).9);

                    }

                }

            } else {

                energiaVenta     =     mapRegistroE.get(contador-
1).getEstadoBateria()+(diferencia/(double)          .9)-((double)
.8*valoresG.getCapacidadNominal());


mapRegistroE.get(contador).setEstadoBateria((double)
.8*valoresG.getCapacidadNominal());


mapRegistroE.get(contador).setBeneficioVenta(mapRegistroE.get(cont
ador).getPrecioVenta()*energiaVenta);


mapRegistroE.get(contador).setEnergiaVendida(energiaVenta);

                energiaVentaFutura=0;

            }

        }
```

```
        return contador;

    }


    private int ConsumoMayorGeneracion(int contador, boolean
retrocede) {

        int contadorR;

        int contadorI;

        int contadorF;

        int contadorAux;

        double diferencia;

        double costeConsumoRed;

        double estadoAnterior;

        double energiaBateria;

        double energiaRed;

        double precioAux;

        mapRegistroE.get(contador).setEnergiaVendida(0);

        mapRegistroE.get(contador).setBeneficioVenta(0);

        mapRegistroE.get(contador).setCargaBateria(0);

        mapRegistroE.get(contador).setCosteCargaBateria(0);

        mapRegistroE.get(contador).setConsumoRed(0);

        mapRegistroE.get(contador).setCosteConsumoRed(0);

        //Cuando el consumo es mayor que lo que se genera

        diferencia                                          =
Math.abs(mapRegistroE.get(contador).getDiferencia());
```

```
mapRegistroE.get(contador).setCosteSinBateria(mapRegistroE.get(con
tador).getPrecioRed()*diferencia);

        switch (mapRegistroE.get(contador).getGrupoPrecioRed()) {

            case "B":

                diferencia                                          =
Math.abs(mapRegistroE.get(contador).getDiferencia());

                costeConsumoRed                                     =
mapRegistroE.get(contador).getPrecioRed() * diferencia;


mapRegistroE.get(contador).setConsumoRed(diferencia);


mapRegistroE.get(contador).setCosteConsumoRed(costeConsumoRed);

                if            (contador            >            0)
mapRegistroE.get(contador).setEstadoBateria(mapRegistroE.get(conta
dor-1).getEstadoBateria());

                break;

            case "M":

                estadoAnterior                              =(double)
.2*this.valoresG.getCapacidadNominal();

                if    (contador    >    0)    estadoAnterior    =
mapRegistroE.get(contador-1).getEstadoBateria();

                diferencia                                          =
Math.abs(mapRegistroE.get(contador).getDiferencia());

                if                (Mates.Redondear(estadoAnterior-
diferencia,5)>=Mates.Redondear((double)
.2*this.valoresG.getCapacidadNominal(),5)) {


mapRegistroE.get(contador).setEstadoBateria(Mates.Redondear(estado
Anterior-diferencia,5));

                } else {
```

```
                    energiaBateria     =     estadoAnterior-(double)
.2*this.valoresG.getCapacidadNominal();

                    energiaRed = (diferencia-energiaBateria);

                    costeConsumoRed                                    =
mapRegistroE.get(contador).getPrecioRed() * energiaRed;


mapRegistroE.get(contador).setConsumoRed(energiaRed);


mapRegistroE.get(contador).setCosteConsumoRed(costeConsumoRed);


mapRegistroE.get(contador).setEstadoBateria(Mates.Redondear(estado
Anterior-energiaBateria,5));

                }

                break;

            case "A":

                estadoAnterior                                =(double)
.2*this.valoresG.getCapacidadNominal();

                if    (contador    >    0)    estadoAnterior    =
mapRegistroE.get(contador-1).getEstadoBateria();

                diferencia                                            =
Math.abs(mapRegistroE.get(contador).getDiferencia());

                if                 (Mates.Redondear(estadoAnterior-
diferencia,5)>=Mates.Redondear((double)
.2*this.valoresG.getCapacidadNominal(),5)) {


mapRegistroE.get(contador).setEstadoBateria(Mates.Redondear(estado
Anterior-diferencia,5));

                } else {

                    if (retrocede) {

                        contadorR = contador-1;
```

```
                        contadorF = contador;

                        contadorI = 0;

                        contadorAux = 0;

                        while      (contadorR      >      0      &&
mapRegistroE.get(contadorR).getDiferencia()<0                    &&
mapRegistroE.get(contadorR).getCosteCargaBateria()==0            &&
mapRegistroE.get(contadorR).getEstadoBateria()<((double)
.8*this.valoresG.getCapacidadNominal())) {

                           contadorR-=1;

                        }

                        if
(mapRegistroE.get(contadorR).getDiferencia()>=0)    contadorI    =
contadorR+1;

                        else                                      if
(mapRegistroE.get(contadorR).getCosteCargaBateria()!=0) contadorI =
contadorR+1;

                        else                                      if
(mapRegistroE.get(contadorR).getEstadoBateria()>=((double)
.8*this.valoresG.getCapacidadNominal())) contadorI = contadorR+1;

                        else contadorI=0;

                        contadorAux=contadorI;


precioAux=mapRegistroE.get(contadorI).getPrecioRed();

                        while (contadorI <= contadorF) {

                           if
(mapRegistroE.get(contadorI).getPrecioRed()<=precioAux) {

                               contadorAux=contadorI;


precioAux=mapRegistroE.get(contadorI).getPrecioRed();

                           }
```

```
                        contadorI++;

                    }

                    contador = contadorAux;

                    diferencia                                  =
Math.abs(mapRegistroE.get(contador).getDiferencia());

                    if
(mapRegistroE.get(contador).getGrupoPrecioRed().equals("B")) {

                        //estadoAnterior       =       (double)
.2*this.valoresG.getCapacidadNominal();

                        //if   (contador>0)   estadoAnterior   =
mapRegistroE.get(contador).getEstadoBateria();

                        estadoAnterior                          =
mapRegistroE.get(contador).getEstadoBateria();

                        energiaRed         =         (((double)
.8*this.valoresG.getCapacidadNominal())-estadoAnterior)/(double)
.9;


mapRegistroE.get(contador).setEstadoBateria((double)
.8*this.valoresG.getCapacidadNominal());


mapRegistroE.get(contador).setCargaBateria(energiaRed);


mapRegistroE.get(contador).setCosteCargaBateria(mapRegistroE.get(c
ontador).getPrecioRed()*energiaRed);


mapRegistroE.get(contador).setCosteConsumoRed(mapRegistroE.get(con
tador).getPrecioRed()*diferencia);


mapRegistroE.get(contador).setConsumoRed(diferencia);

                    } else { //El grupo sera M o A

                        //estadoAnterior       =       (double)
.2*this.valoresG.getCapacidadNominal();
```

```
                            //if   (contador>0)   estadoAnterior   =
mapRegistroE.get(contador-1).getEstadoBateria();

                            estadoAnterior                          =
mapRegistroE.get(contador).getEstadoBateria(); //

                            energiaRed        =        (((double)
.8*this.valoresG.getCapacidadNominal())-estadoAnterior)/(double)
.9;


mapRegistroE.get(contador).setEstadoBateria((double)
.8*this.valoresG.getCapacidadNominal()-diferencia);


mapRegistroE.get(contador).setCargaBateria(energiaRed);


mapRegistroE.get(contador).setCosteCargaBateria(mapRegistroE.get(c
ontador).getPrecioRed()*energiaRed);

                        }

                    } else {

                        energiaBateria   =   estadoAnterior-(double)
.2*this.valoresG.getCapacidadNominal();

                        energiaRed = (diferencia-energiaBateria);

                        costeConsumoRed                             =
mapRegistroE.get(contador).getPrecioRed() * energiaRed;


mapRegistroE.get(contador).setConsumoRed(energiaRed);


mapRegistroE.get(contador).setCosteConsumoRed(costeConsumoRed);


mapRegistroE.get(contador).setEstadoBateria(Mates.Redondear(estado
Anterior-energiaBateria,5));

                    }

                }
```

```
                break;

            }

            return contador;

        }


        //AJUSTA EL ESTADO DE LA BATERIA DEPENDIENDO DEL GRUPO DE DONDE
A SALIDO 1-diferencia < 0 y 2 diferncia > 0

        private int AjustarBateriaConsMayorGen(int contador) {

            int contadorActual = contador;

            int contadorR = contador;

            int contadorF = contador;

            int contadorCB = contador;

            int contadorPVA = contador;

            double   estadoBateriaActual   =   Mates.Redondear((double)
.2*valoresG.getCapacidadNominal(),5);

            double estadoBateriaAnterior = 0;

            double diferencia = 0;

            double diferenciaAM = 0;

            double precioVenta = 0;

            if  (mapRegistroE.get(contadorActual).getEstadoBateria()  >
estadoBateriaActual) {

                contadorF = 0;


mapRegistroE.get(contadorActual).setEstadoBateria(estadoBateriaAct
ual);

                while         (contadorR         >         0         &&
mapRegistroE.get(contadorR).getDiferencia() < 0) {
```

```
                contadorR -= 1;

            }

            if (mapRegistroE.get(contadorR).getDiferencia() >= 0)
contadorF = contadorR + 1;

            else contadorF = contadorR;

            for (contadorR = contadorActual;contadorR >= contadorF;
contadorR--) {

               if (mapRegistroE.get(contadorR).getPrecioVenta() >
precioVenta) {

                   contadorPVA = contadorR;

                   precioVenta                                    =
mapRegistroE.get(contadorR).getPrecioVenta();

               }

               if (mapRegistroE.get(contadorR).getCargaBateria()> 0
&& contadorR < contadorCB ) {

                   contadorCB = contadorR;

               }

            }

            if (contadorCB<contadorActual) { //Hemos encontrado una
carga de bateria

                contadorF = contadorCB;

            } else {

                contadorF = contadorPVA;

            }

            contadorR = contadorActual;

            while (contadorR >= contadorF) {

                if (mapRegistroE.get(contadorR).getDiferencia() <
0) {
```

```
                    diferencia                                          =
Math.abs(mapRegistroE.get(contadorR).getDiferencia());

                    if
(!mapRegistroE.get(contadorR).getGrupoPrecioRed().equals("B")) {

                        if
(estadoBateriaActual+diferenciaAM+diferencia                            <
(double).8*valoresG.getCapacidadNominal()) {

                            diferenciaAM += diferencia;

                        } else {

                            contadorF = contadorR+1;

                        }

                    }

                contadorR--;

            }

            mapRegistroE.get(contadorF).setCargaBateria(0);

            mapRegistroE.get(contadorF).setCosteCargaBateria(0);

            diferencia                                                  =
Math.abs(mapRegistroE.get(contadorF).getDiferencia());

            if
(!mapRegistroE.get(contadorF).getGrupoPrecioRed().equals("B")) {


mapRegistroE.get(contadorF).setEstadoBateria(estadoBateriaActual+d
iferenciaAM-diferencia);

            } else {


mapRegistroE.get(contadorF).setEstadoBateria(estadoBateriaActual+d
iferenciaAM);


mapRegistroE.get(contadorF).setConsumoRed(diferencia);
```

```
mapRegistroE.get(contadorF).setCosteConsumoRed(diferencia*mapRegis
troE.get(contadorF).getPrecioRed());

            }

            if(contadorF > 0) {

                estadoBateriaAnterior = mapRegistroE.get(contadorF-
1).getEstadoBateria();

                if
(!mapRegistroE.get(contadorF).getGrupoPrecioRed().equals("B")) {

                    estadoBateriaAnterior -= diferencia;

                }

                if             (estadoBateriaAnterior             >
mapRegistroE.get(contadorF).getEstadoBateria()) {


mapRegistroE.get(contadorF).setEnergiaVendida(estadoBateriaAnterio
r - mapRegistroE.get(contadorF).getEstadoBateria());


mapRegistroE.get(contadorF).setBeneficioVenta((estadoBateriaAnteri
or                                                              -
mapRegistroE.get(contadorF).getEstadoBateria())*mapRegistroE.get(c
ontadorF).getPrecioVenta());

                } else {

                    if           (estadoBateriaAnterior           <
mapRegistroE.get(contadorF).getEstadoBateria()) {


mapRegistroE.get(contadorF).setCargaBateria(mapRegistroE.get(conta
dorF).getEstadoBateria() - estadoBateriaAnterior );


mapRegistroE.get(contadorF).setCosteCargaBateria((mapRegistroE.get
(contadorF).getEstadoBateria()                                  -
estadoBateriaAnterior)*mapRegistroE.get(contadorF).getPrecioRed())
;

                    }
```

```
                }

            } else {

                diferencia                                          =
mapRegistroE.get(contadorF).getEstadoBateria()-
(double).2*valoresG.getCapacidadNominal(); //compramos esta energia


mapRegistroE.get(contadorF).setCargaBateria(diferencia);


mapRegistroE.get(contadorF).setCosteCargaBateria(diferencia*mapReg
istroE.get(contadorF).getPrecioRed());

            }

        }

        if (contadorF < contadorActual) {  // Si retrocede vuelve a
recalcular desde donde ha retrocedido

            for    (int    contadorN=contadorF+1;    contadorN    <=
contadorActual; contadorN++) {

                contador = ConsumoMayorGeneracion(contadorN,false);
//Recalcula todo desde el retroceso al estado actaul

            }

        }

        return contadorActual;

    }


    private int AjustarBateriaGenMayorCons(int contador) {

        int contadorActual = contador;

        int contadorR = contador;

        int contadorF = contador;

        while(contadorR>0                                           &&
mapRegistroE.get(contadorR).getEstadoBateria()<(double).8*valoresG
```

```
.getCapacidadNominal()                                          &&
mapRegistroE.get(contadorR).getDiferencia()>0){

            contadorR--;

        }


if(mapRegistroE.get(contadorR).getEstadoBateria()==(double).8*valo
resG.getCapacidadNominal() || contadorR == 0) contadorF = contadorR;

        else contadorF = contadorActual;

        if(contadorF > 0){

            if(contadorR == contadorF){ //ha encontrado un 80%

                while(contadorR      <=      contadorActual      &&
mapRegistroE.get(contadorR).getEstadoBateria()>(double).2*valoresG
.getCapacidadNominal()){

                    contadorR++;

                }

                if(contadorR <= contadorActual){

                    double               diferencia              =
Math.abs(mapRegistroE.get(contadorR).getDiferencia());

                    double estadoAnt = mapRegistroE.get(contadorR-
1).getEstadoBateria();


mapRegistroE.get(contadorR).setEstadoBateria(estadoAnt);      //80%
menos perdida del consumo


mapRegistroE.get(contadorR).setEnergiaVendida(diferencia/(double)
.9);


mapRegistroE.get(contadorR).setBeneficioVenta((diferencia/(double)
.9)*mapRegistroE.get(contadorActual).getPrecioVenta());

                    contadorF=contadorR;
```

```
                } else contadorF=contadorActual;


            }

        }

        if (contadorF < contadorActual) {  // Si retrocede vuelve a
recalcular desde donde ha retrocedido

            for    (int    contadorN=contadorF+1;    contadorN    <=
contadorActual; contadorN++) {

                contador = GeneracionMayorConsumo(contadorN,false);
//Recalcula todo desde el retroceso al estado actaul

            }

        }

        return contadorActual;

    }


    //FUNCION QUE DEVUELVE GENERACION

    private  double  DevuelveGeneracion(Date  hora){  //devuelve  un
double de la generacion en la hora  que toca

        double retorno = 0; //devuelve valor inicial 0 como valor
de la generacion si no encuentra la fecha

        boolean encontrada = false; //en un ppio es falsa, no hemos
empezado ni a buscarla

        for(int contador = 0; contador < generacionReal.length &&
!encontrada; contador++){

            if (generacionReal[contador]!=null){

                if
(generacionReal[contador].getHora().equals(hora)){ //equals es para
comparar objetos

                    encontrada = true;
```

```
                    retorno = generacionReal[contador].getGeneracionTotal();
                }
            }
        }
        return retorno;
    }

    //FUNCION QUE DEVUELVE PRECIO RED
    private PreciosRed DevuelvePrecioRed(Date hora){ //devuelve un double de la generacion en la hora  que toca
        PreciosRed retorno = null; //devuelve valor inicial 0 como valor de la generacion si no encuentra la fecha
        boolean encontrada = false; //en un ppio es falsa, no hemos empezado ni a buscarla
        for(int contador = 0; contador < preciosRed.length && !encontrada; contador++){
            if (preciosRed[contador]!=null){
                if (preciosRed[contador].getHora().equals(hora)){
                    encontrada = true;
                    retorno = preciosRed[contador];
                }
            }

        }
        return retorno;
    }
```

```
            //FUNCION QUE DEVUELVE PRECIO VENTA

    private PreciosVenta DevuelvePrecioVenta(Date hora){ //devuelve
un double de la generacion en la hora  que toca

        PreciosVenta retorno = null; //devuelve valor inicial 0 como
valor de la generacion si no encuentra la fecha

        boolean encontrada = false; //en un ppio es falsa, no hemos
empezado ni a buscarla

        for(int contador = 0; contador < preciosVenta.length &&
!encontrada; contador++){

            if (preciosVenta[contador]!=null){

                if
(preciosVenta[contador].getHora().equals(hora)){

                    encontrada = true;

                    retorno       =       preciosVenta[contador];
//preciosVenta es la variable de la tabla

                }

            }

        }

        return retorno;

    }

}
```

ANEXO X: CÓDIGO PARA ESCRIBIR EL CSV CON LOS VALORES FINALES

```
package es.carlota.controlbateria.procesos;


import es.carlota.controlbateria.clases.RegistroEnergetico;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Map;


/**
 *
 * @author Carlota
 */
public class WriteCSV {

    public      WriteCSV(String      nombreFicheroCSV,Map<Integer,
RegistroEnergetico> mapRegistroE){

        String texto = "";

        try {

            if (!mapRegistroE.isEmpty()) {

                File f = new File(nombreFicheroCSV);
```

```
                if (f.exists()) {
                    f.delete();
                }
                FileWriter fw = new FileWriter(nombreFicheroCSV);
                BufferedWriter bw = new BufferedWriter(fw);
                texto=mapRegistroE.get(0).getTitulos();
                bw.write(texto);
                bw.newLine();
                for(int contador1 = 0; contador1 < mapRegistroE.size(); contador1++){

texto=mapRegistroE.get(contador1).getDatos(contador1);
                    bw.write(texto);
                    bw.newLine();
                }
                bw.close();
                fw.close();
            }
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```

ANEXO XI: CÓDIGO PARA CREAR LA INTERFAZ CON EL USUARIO

```
package es.carlota.controlbateria;

import es.carlota.controlbateria.clases.RegistroEnergetico;

import es.carlota.controlbateria.clases.ValoresGlobales;

import es.carlota.controlbateria.procesos.InicializaValoresCSV;

import es.carlota.controlbateria.procesos.ProcesarRegistro;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.time.LocalDate;

import java.time.Month;

import java.time.ZoneId;

import java.util.Date;

import javafx.collections.ObservableList;

import javafx.fxml.FXML;

import javafx.scene.chart.CategoryAxis;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.TextField;

import javafx.scene.chart.LineChart;

import javafx.scene.chart.NumberAxis;
```

```
import javafx.scene.chart.XYChart;
import javafx.scene.control.DatePicker;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.VBox;

public class PrimaryController {
    ValoresGlobales valoresG = new ValoresGlobales();
    ProcesarRegistro procesarRegistro;
    ObservableList<RegistroEnergetico> datos;
    LineChart<String, Number> grafica;

    @FXML
    private VBox caja;

    @FXML
    private TextField txtConsMedio;

    @FXML
    private TextField txtPanelesEspana;

    @FXML
    private TextField txtNumeroPaneles;

    @FXML
```

```
    private TextField txtCapacidadNominal;


    @FXML

    private DatePicker txtFechaDesde;


    @FXML

    private DatePicker txtFechaHasta;


    @FXML

    private TableView tablaDatos;


    @FXML

    private void BotonAccion() throws IOException {

valoresG.setConsMedio(Double.parseDouble(txtConsMedio.getText()));


valoresG.setPanelesEspana(Double.parseDouble(txtPanelesEspana.getT
ext()));


valoresG.setNumeroPaneles(Double.parseDouble(txtNumeroPaneles.getT
ext()));


valoresG.setCapacidadNominal(Double.parseDouble(txtCapacidadNomina
l.getText()));

        InicializaValoresCSV        iniValCSV        =        new
InicializaValoresCSV(valoresG,
"./ConsumoRealHoraA.csv","./GeneracionHoraA.csv","./PrecioRedA.csv
","./PrecioVentaA.csv");
```

```
        procesarRegistro = new ProcesarRegistro(valoresG,
iniValCSV.getConsumoReal(), iniValCSV.getGeneracionReal(),
iniValCSV.getPreciosRed(), iniValCSV.getPreciosVenta());

        procesarRegistro.CrearRegistroEnergetico();

        datos = procesarRegistro.RecalculaRegistroEnergetico();

        tablaDatos.setItems(datos);

        CategoryAxis xAxis = new CategoryAxis();

        NumberAxis yAxis = new NumberAxis();

        grafica = new LineChart<String, Number>(xAxis, yAxis);

        caja.getChildren().add(grafica);

        grafica.setTitle("Evolución del estado de la bateria");

        grafica.setCreateSymbols(false);

    }


    //Convierte del tipo LocalDate a tipo Date

    private Date PasaAFecha(LocalDate localDate) {

        return
Date.from(localDate.atStartOfDay().atZone(ZoneId.systemDefault()).
toInstant());

    }


    //Quita la hora de la fecha dejando dia mes y año

    private Date QuitarHora(Date fechaHora) throws ParseException {

        SimpleDateFormat formato = new SimpleDateFormat("yyyy-MM-
dd");

        String fechaSinHoraString = formato.format(fechaHora);

        return formato.parse(fechaSinHoraString);
```

```
    }


    private void CreaGrafica() throws IOException, ParseException {

        Date fecha1;

        Date fechaD = PasaAFecha(txtFechaDesde.getValue());

        Date fechaH = PasaAFecha(txtFechaHasta.getValue());

        grafica.getData().clear();

        XYChart.Series<String, Number> series1 = new
XYChart.Series<>();

        series1.setName("Estado Bateria");

        for(RegistroEnergetico reg1: datos) {

            //Quito la hora de la fecha para realizar una
comparacion del dia sin tener en cuenta la hora

            fecha1 = QuitarHora(reg1.getHora());

            boolean esPosteriorFechaInicial = fecha1.after(fechaD);

            boolean esIgualFechaInicial = fecha1.equals(fechaD);

            boolean esFechaInicial = esPosteriorFechaInicial ||
esIgualFechaInicial;

            boolean esAnteriorFechaFinal = fecha1.before(fechaH);

            boolean esIgualFechaFinal = fecha1.equals(fechaH);

            boolean esFechaFinal = esAnteriorFechaFinal ||
esIgualFechaFinal;

            if (esFechaInicial && esFechaFinal ) {

                series1.getData().add(new
XYChart.Data<>(reg1.getHoraCadena(),reg1.getEstadoBateria()));

            }

        }
```

```
        grafica.getData().add(series1);

    }

    @FXML

    private void BotonGrafica() throws IOException, ParseException
{

        CreaGrafica();

    }

    @FXML

    private void BotonArchivo() throws IOException {

        procesarRegistro.CrearArchivo();

    }

    @FXML

    public void initialize(){


txtConsMedio.setText(String.format("%.0f",valoresG.getConsMedio())
);


txtPanelesEspana.setText(String.format("%.0f",valoresG.getPanelesE
spana()));


txtNumeroPaneles.setText(String.format("%.0f",valoresG.getNumeroPa
neles()));


txtCapacidadNominal.setText(String.format("%.0f",valoresG.getCapac
idadNominal()));

        LocalDate fi = LocalDate.of(2024, Month.JANUARY, 1);

        txtFechaDesde.setValue(fi);

        LocalDate ff = LocalDate.of(2024, Month.DECEMBER, 31);

        txtFechaHasta.setValue(ff);
```

```
        TableColumn colHora = new TableColumn("Día/Hora");

        colHora.setCellValueFactory(new
PropertyValueFactory<>("horaCadena"));

        TableColumn colConsumo = new TableColumn("Consumo");

        colConsumo.setCellValueFactory(new
PropertyValueFactory<>("consumo"));

        TableColumn colGeneracion = new TableColumn("Generación");

        colGeneracion.setCellValueFactory(new
PropertyValueFactory<>("generacion"));

        TableColumn colDiferencia = new TableColumn("Diferencia");

        colDiferencia.setCellValueFactory(new
PropertyValueFactory<>("diferencia"));

        TableColumn colPrecioRed = new TableColumn("Precio Red");

        colPrecioRed.setCellValueFactory(new
PropertyValueFactory<>("precioRed"));

        TableColumn colGrupoPrecioRed = new TableColumn("Grupo
Precio Red");

        colGrupoPrecioRed.setCellValueFactory(new
PropertyValueFactory<>("grupoPrecioRed"));

        TableColumn colPrecioVenta = new TableColumn("Precio
Venta");

        colPrecioVenta.setCellValueFactory(new
PropertyValueFactory<>("precioVenta"));

        TableColumn colGrupoPrecioVenta = new TableColumn("Grupo
Precio Venta");

        colGrupoPrecioVenta.setCellValueFactory(new
PropertyValueFactory<>("grupoPrecioVenta"));

        TableColumn colEstadoBateria = new TableColumn("Estado
Bateria");
```

```
        colEstadoBateria.setCellValueFactory(new
PropertyValueFactory<>("estadoBateria"));

        TableColumn colCargaBateria = new TableColumn("Carga
Bateria");

        colCargaBateria.setCellValueFactory(new
PropertyValueFactory<>("cargaBateria"));

        TableColumn colCosteCargaBateria = new TableColumn("Coste
Carga Bateria");

        colCosteCargaBateria.setCellValueFactory(new
PropertyValueFactory<>("costeCargaBateria"));

        TableColumn colConsumoRed = new TableColumn("Consumo Red");

        colConsumoRed.setCellValueFactory(new
PropertyValueFactory<>("consumoRed"));

        TableColumn colCosteConsumoRed = new TableColumn("Coste
Consumo Red");

        colCosteConsumoRed.setCellValueFactory(new
PropertyValueFactory<>("costeConsumoRed"));

        TableColumn colCosteConsumoRedTotal = new
TableColumn("Coste Consumo Red Total");

        colCosteConsumoRedTotal.setCellValueFactory(new
PropertyValueFactory<>("consumoRedTotal"));

        TableColumn colEnergiaVendida = new TableColumn("Energia
Vendida");

        colEnergiaVendida.setCellValueFactory(new
PropertyValueFactory<>("energiaVendida"));

        TableColumn colBeneficioVenta = new TableColumn("Beneficio
Venta");

        colBeneficioVenta.setCellValueFactory(new
PropertyValueFactory<>("beneficioVenta"));

        TableColumn colCosteTotal = new TableColumn("Coste Total");
```

```
        colCosteTotal.setCellValueFactory(new
PropertyValueFactory<>("costeTotal"));

        TableColumn colCosteSinBateria = new TableColumn("Coste Sin
Bateria");

        colCosteSinBateria.setCellValueFactory(new
PropertyValueFactory<>("costeSinBateria"));

        TableColumn     colCosteSinPanelesNiBateria     =     new
TableColumn("Coste Sin Paneles Ni Bateria");

        colCosteSinPanelesNiBateria.setCellValueFactory(new
PropertyValueFactory<>("costeSinPanelesNiBateria"));

        TableColumn colBalanceEnergetico = new TableColumn("Balance
energetico");

        colBalanceEnergetico.getColumns().addAll(colConsumo,
colGeneracion,colDiferencia);

        TableColumn colRedEnergeticaConsumo = new TableColumn("Red
energetica consumo");

        colRedEnergeticaConsumo.getColumns().addAll(colPrecioRed,
colGrupoPrecioRed,        colConsumoRed,        colCosteConsumoRed,
colCargaBateria, colCosteCargaBateria);

        TableColumn  colRedEnergeticaVenta  =  new  TableColumn("Red
energetica venta");

        colRedEnergeticaVenta.getColumns().addAll(colPrecioVenta,
colGrupoPrecioVenta, colEnergiaVendida, colBeneficioVenta);

        TableColumn colTotales = new TableColumn("Totales");

        colTotales.getColumns().addAll(colCosteConsumoRedTotal,
colCosteTotal, colCosteSinBateria, colCosteSinPanelesNiBateria);

        tablaDatos.getColumns().addAll(colHora,
colBalanceEnergetico,  colEstadoBateria,   colRedEnergeticaConsumo,
colRedEnergeticaVenta, colTotales);

    }

}
```