



UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA DE INGENIERÍA Y DISEÑO INDUSTRIAL
Grado en Ingeneriería Eléctrica

TRABAJO DE FIN DE GRADO

**Aplicación web para la estimación del
coste de una instalación solar conectada a
red**

Autor: Ionut Cristian Morariu

Tutor: Oscar Perpiñán Lamigueiro
Departamento de Ingeniería Eléctrica,
Electrónica, Automática y Física aplicada

Madrid, 17 de junio de 2020

Índice general

1. Introducción	1
1.1. Objetivos	1
1.2. Análisis previo de soluciones	2
1.3. Aspectos técnicos	3
1.3.1. Backend	3
1.3.2. Frontend	4
2. Estado del arte	5
2.1. Situación actual de la generación fotovoltaica	5
2.2. Soluciones existentes y sus carencias	6
3. Parte teórica y desarrollo del código	10
3.1. Naturaleza de la radiación solar	11
3.1.1. Radiación fuera de la atmósfera terrestre	12
3.1.2. Cálculo de componentes de radiación solar	14
3.2. Radiación en superficies inclinadas	16
3.2.1. Estimación de irradiancia a partir de irradiación diaria	16
3.2.2. Transformación al plano del generador	18
3.2.3. Pérdidas por ángulo de incidencia y suciedad	21
3.3. Cálculo de la energía producida por el generador	24
3.3.1. Definición de un SFCR	24
3.3.2. Configuración de los elementos del sistema	24
3.3.3. Funcionamiento de una célula solar	25
3.3.4. Comportamiento térmico de un módulo	26
3.3.5. Cálculo final de potencias y energías	35
4. Ejemplo práctico de aplicación	39
4.1. Obtención de datos del usuario	39
4.1.1. Latitud y longitud a partir de la dirección	40
4.1.2. Datos de la superficie destinada a la instalación	40
4.2. Aplicación de los datos al proceso de cálculo	41
4.2.1. Valores medios mensuales de radiación global	42
4.2.2. Irradiancia extra-terrestre diaria	43
4.2.3. Separación de la radiación global horizontal en sus componentes	43

4.2.4. Irradiancia en la superficie inclinada	46
4.2.5. Pérdidas por ángulo de incidencia y suciedad	50
4.2.6. Configuración del módulo solar y su comportamiento	53
4.3. Cálculo final de las potencias y energías	63
5. Detalles de la programación	66
5.1. Descripción de tecnologías	66
5.1.1. Servidor	66
5.1.2. Cliente	68
5.2. Explicaciones de código	69
5.2.1. Emplazamiento del usuario	69
5.2.2. Área, inclinación, orientación y nivel de suciedad de la superficie de instalación	71
5.2.3. Obtención de la información de radiación global en el plano horizontal . . .	71
5.3. Instrucciones	76
5.3.1. Obtener la inclinación y orientación de la superficie	76
A. Código completo	84
A.1. Código del servidor	84
A.1.1. DataController.js	84
A.1.2. Helpers.js	92
A.1.3. SolarData.js	99
A.1.4. StationsData.js	100
A.1.5. Router.js	101
A.1.6. App.js	102
A.1.7. Start.js	102
A.2. Código de cliente	104
A.2.1. dataAquisition.js	104
A.2.2. GenerateLinks.js	115
A.2.3. index.html	117
A.2.4. Results.html	123
A.2.5. renderResults.js	126

Capítulo 1

Introducción

1.1. Objetivos

El objetivo de este proyecto es el desarrollo de una aplicación o página web de fácil acceso para todos los usuarios, con la finalidad de ofrecer una estimación inicial del coste y la posible generación de una instalación fotovoltaica doméstica de conexión a red.

Durante el resto del documento, si fuera necesario, se hará referencia a la aplicación desarrollada en este proyecto en el nombre de SolarCalc.

Para poder llevar a cabo esta estimación, el usuario introducirá unos serie de datos acerca de su emplazamiento y edificación en la que desea situar la instalación, y la aplicación hará todos los cálculos necesarios para ofrecer una aproximación lo más cercana al resultado final teniendo en cuenta todas la variables que puedan intervenir.

La aplicación también ofrecerá otros datos de posible interés para el usuario como: el número de paneles que se pueden instalar, la potencia de dichos paneles y la potencia del inversor.

La idea de esta aplicación surge de una conversación que tuve con un conocido cuando estaba planificando la construcción de su nueva vivienda, en la cual quería realizar una instalación fotovoltaica para reducir el gasto en la factura de electricidad.

En su búsqueda no encontró ningún servicio que fuera lo suficientemente sencillo de entender para una persona sin ningún tipo de conocimiento previo acerca de la generación fotovoltaica, que le aportase la posibilidad de poder personalizar los cálculos a su emplazamiento y planos de construcción.

Otro de los puntos clave de la aplicación es que sea de código abierto y gratuita para los usuarios, usando fuentes de información disponibles para cualquier interesado. Todos los pasos y operaciones se podrán obtener, analizar y reutilizar de manera gratuita desde un repositorio de Github¹.

¹Github.com: Plataforma online de almacenamiento de código y documentación de fuentes abiertas.

Los objetivos detallados de esta aplicación son los siguientes:

- Diseñar una interfaz de usuario amigable y sencilla de usar para que la pueda utilizar un gran número de personas sin necesidad de conocimientos sobre energía fotovoltaica.
- Obtención de los datos de radiación en el emplazamiento indicado por el usuario mediante el uso de API² externas.
- Realizar todos los cálculos necesarios para ofrecer una estimación competente de los siguientes datos:
 - Número de paneles que se pueden instalar.
 - Potencia máxima a instalar.
 - Potencia del inversor.
 - Energía eléctrica producida en un año.

1.2. Análisis previo de soluciones

Antes de comenzar el desarrollo del proyecto, se llevó a cabo una revisión de las soluciones existentes de estimaciones de instalaciones fotovoltaicas existentes en el mercado para decidir si tenía cabida una aplicación como la que se iba a desarrollar.

Algunas de las soluciones encontradas fueron:

1. PVsyst - Photovoltaic Software

El software PVsyst, desarrollado por la empresa suiza con el mismo nombre es quizás el más conocido dentro del ámbito del estudio y la estimación de instalaciones fotovoltaicas. Ofrece una amplia capacidad de personalización de todos los componentes de la instalación.

2. CalculationSolar.com

Es el primer resultado de Google al buscar el término “calculadora de instalaciones fotovoltaicas”, por tanto será una de las primeras aplicaciones que una persona que desea realizar una instalación en su vivienda visite.

3. SISIFO

Es una herramienta web diseñada y desarrollada por el Grupo de Sistemas Fotovoltaicos del Instituto de Energía Solar de la Universidad Politécnica de Madrid. Ha sido y es la herramienta interna utilizada por los ingenieros de dicho grupo.

4. PVGIS

Aplicación web desarrollada por el European Commission Joint Research Center desde 2001. Su enfoque es asistir en el cálculo y estimación de instalaciones fotovoltaicas, ya sean conectadas a red, de seguimiento o de autoconsumo.

²Application Programming Interface: conjunto de funciones y procedimientos que ofrece la posibilidad de un software a interaccionar con otro.

5. System Advisor Model

System Advisor Model (SAM), desarrollado por el Laboratorio Nacional de Energías Renovables, perteneciente al Departamento de Energía del gobierno americano, es una software técnico-económico gratuito que ayuda a la toma de decisiones en el amplio campo de las energías renovables. Ofrece un conjunto de soluciones muy completas no solamente relacionadas con la energía fotovoltaica, sino también termosolar, eólica, geotermal o biomasa, entre otras.

6. solaR

solaR es un paquete de código para el entorno de R, desarrollado por Oscar Perpiñán, que a pesar de tener objetivos diferentes a los planteados por la aplicación desarrollada en este proyecto, ha servido como base para validar todos los cálculos que se han realizado en ella.

En el apartado 2.2 se lleva a cabo un desarrollo mas detallado de las características de las soluciones mencionadas así como sus diferencias con la propuesta de este proyecto.

1.3. Aspectos técnicos

Para construir cualquier página web se deben desarrollar dos sistemas diferentes llamados Backend y Frontend. El Backend es la parte de cálculo y tratamiento de peticiones, comúnmente llamado server o servidor. Por otro lado se encuentra el Frontend que es la parte de cara al usuario, la que se encarga de recoger los datos introducidos por este y enviarlos al servidor.

1.3.1. Backend

Para el Backend se ha empleado una tecnología basada en Javascript llamada NodeJS³, con la ayuda de las librerías ExpressJS⁴ y Mongoose⁵.

La base de datos que se ha utilizado para almacenar los datos necesarios ha sido MongoDB. En el servidor se realizan varias tareas relacionadas con los cálculos necesarios. Algunas de estas tareas son:

- Obtención de los datos de irradiación global media en el plano horizontal para el emplazamiento indicado
- Proceso completo de cálculo para pasar de la irradiación en el plano horizontal al plano inclinado y orientado según los datos introducidos por el usuario
- Proceso de obtención de los datos relacionados con el perfil horario de temperatura en el emplazamiento indicado
- Gestión de las diferentes rutas que constituyen la API.

³NodeJS: Entorno de ejecución basado en el motor de Chrome llamado V8. <https://nodejs.org/en/>

⁴ExpressJS: Framework web para el entorno de NodeJS. <https://expressjs.com/es/>

⁵Mongoose: Capa intermedia de interacción para las BBDD MongoDB <https://mongoosejs.com/>

1.3.2. Frontend

Para el Frontend de la página se han utilizado las tres tecnologías necesarias para poder desarrollar una pagina web: HTML5, CSS3, Javascript.

Esta parte de la página es la encargada de recoger los datos del usuario y enviarlos al servidor para que se realicen los cálculos. Una vez realizados dichos cálculos, la página mostrará la información relevante al usuario, junto con algunos unos gráficos adicionales.

Tanto el backend como el frontend están almacenados en un servidor de Linux remoto activo 24/7.

Todas las tareas mencionadas tanto en la parte de Backend como en la parte de Frontend se describirán en detalle en la sección 3, junto con todos los cálculos en los que se ha basado.

Capítulo 2

Estado del arte

2.1. Situación actual de la generación fotovoltaica

Según el informe anual de la UNEF¹ en 2019[1] la evolución de las energías renovables superó incluso las expectativas más optimistas alcanzando valores próximos a los 100 GW.

Los aspectos que más han influido en estas cifras han sido, entre otros, la reducción drástica del coste de producción de dichas tecnologías. De hecho, la energía fotovoltaica es ya más barata que la generada por plantas de combustibles fósiles en términos de LCOE². Según menciona Bloomberg Energy Finance, la fotovoltaica seguirá reduciendo sus costes un 34 % hasta 2030.

Sumado a la reducción del coste, el aumento de compraventa de energía a largo plazo - que sigue en alza desde 2018, alcanzando los 14 GW - es otro de los factores influyentes en el crecimiento del sector. Así lo es también la reducción del precio de la energía en las subastas, alcanzando valores tan bajos como 20\$/MWh.

Concretamente en Europa, el crecimiento anual de la capacidad solar instalada ha sido de un 23 %, con Alemania como líder sumando otros 2,95 GW respecto a la capacidad del año anterior. En segundo y tercer lugar se encuentran Turquía y los Países Bajos.

Bajando un nivel más, nos encontramos con el mercado español, que, según estimaciones del mismo informe de la UNEF, la potencia total instalada experimentó un crecimiento significativo, llegando hasta el valor de 262 MW, sumando la potencia instalada tanto de generación centralizada como la de autoconsumo.

Para este año 2020 se estima que la instalación de energía fotovoltaica alcance el umbral de los 20 GW. Si se cumplen las expectativas, la capacidad podría llegar a alcanzar los 200 GW para 2023.

¹UNEF: Unión Española fotovoltaica

²LCOE: Levelized Cost of energy: Medición del coste medio de generación de energía de una planta a lo largo de su vida útil

Un papel importante lo juegan las autoridades tanto nacionales como a nivel europeo, que apuestan por las fuentes de energías limpias. Para ser exactos, el año 2018 fue uno de los más relevantes en materia de política energética europea desde que se aprobó el tercer paquete de energía en 2009. De las ocho propuestas que se aprobaron, destaca por su importancia para el sector fotovoltaico la directiva 2018/2001, en la que se recoge el derecho básico al autoconsumo, individual o colectivo, al almacenamiento y sobretodo a la venta de excedentes.

En el panorama español, tras varios años de parálisis debida a la compleja situación política de los últimos años, la energía fotovoltaica volvió a recuperar algo de impulso a final del año 2019. Durante el año 2018 y especialmente el 2019, se han incrementado sustancialmente las instalaciones de fotovoltaica, en gran parte gracias a las expectativas generadas por la eliminación del denominado impuesto al sol y la progresiva suspensión de trabas administrativas todas ellas propiciadas por el nuevo marco legal establecido por la administración pública mediante los Reales Decretos 15/2019 y 244/2019.

Según datos proporcionados por la Red Eléctrica Española³, la potencia instalada en España ha experimentado un crecimiento de 1,2GW en los últimos 4 años, pasando de 4.6 GW en 2016 a 5.8 GW en 2019.

Se trata del mayor ritmo de crecimiento desde 2008, cuando se instalaron cerca de 2.7 GW de nueva potencia. Es una buena noticia, pero no exenta de dificultades debidas sobre todo a la unidireccionalidad de la red eléctrica de nuestro territorio. Sin ir más lejos, el operador técnico del sistema, REE ha denegado la instalación de 26,3 GW de nueva potencia debido a la imposibilidad de los nudos para gestionar la energía producida por dicha capacidad.

Esto no significa que dicha energía no se vaya a instalar, sino que habrá que esperar a la nueva planificación energética prevista para el periodo 2021-2026 en la que ya están trabajando tanto la REE como las comunidades autónomas para brindar más oportunidades para los sistemas de conexión a red.

En definitiva, como podemos observar, las energías renovables, y en especial la fotovoltaica están dando mucho que hablar y cada vez es un tema más tratado por el público general y por tanto, es un aspecto a tener en cuenta a la hora de construir nuevas edificaciones o mejorar la eficiencia energética de las existentes.

Surge por tanto la necesidad de aplicaciones y soluciones para la estimación de instalaciones fotovoltaicas que sean intuitivas y fáciles de usar para aprovechar el gran interés que está mostrando el público general.

2.2. Soluciones existentes y sus carencias

Como ya se ha mencionado en el apartado 1.2 existen multitud soluciones para la estimación o simulación de instalaciones fotovoltaicas. Sin embargo, es posible que la aplicación descrita en este proyecto siga teniendo cabida porque ofrece ciertas funciones que, en cierta medida, no existen en las soluciones que se han estudiado.

A continuación se describirán con detalle las soluciones mencionadas anteriormente, junto con sus capacidades y carencias.

³<https://www.ree.es/es/datos/publicaciones/series-estadísticas-nacionales>

1. PVsyst - Photovoltaic Software [2]

El software PVsyst, desarrollado por la empresa suiza con el mismo nombre es quizá el más conocido dentro del ámbito del estudio y la estimación de instalaciones fotovoltaicas. Ofrece una amplia capacidad de personalización de todos los componentes de la instalación.

PVsyst es capaz de calcular con amplio detalle el diseño y dimensionado del sistema, zonas de sombra, envejecimiento del material, almacenamiento de energía, entre otras opciones.

PVsyst se diferencia de la aplicación que se desarrolla en este proyecto en algunos puntos importantes como:

- Es de pago, con una licencia anual de aproximadamente €952 mientras que mi solución es gratuita.
- Es un programa que se debe instalar en un ordenador de Windows (No funciona en Linux u OSX).
- Es poco intuitivo para un usuario con bajos conocimientos de instalaciones fotovoltaicas.

En resumen, PVsyst es un software mucho mas completo y complejo que la solución que yo propongo, y que va enfocada a un público con una base sólida sobre energía fotovoltaica.

2. CalculationSolar.com [3]

Como ya se ha mencionado en la introducción, CalculationSolar.com es el primer resultado que aparece en el motor de búsqueda de Google cuando se introduce el término “*calculadora de instalaciones fotovoltaicas*” y por tanto será una de las primeras opciones que un usuario que está interesado en realizar una instalación fotovoltaica considere.

A diferencia de PVsyst, CalculationSolar.com es una solución en la web y gratuita, por lo que la barrera de acceso es mas baja. Nada mas entrar a la pagina lo primero con lo que nos encontramos es con un formulario sencillo sobre el emplazamiento y la configuración de la instalación. La información que se solicita es sencilla e intuitiva, incluso se ofrece la posibilidad de hacer clic en un mapa interactivo para determinar las coordenadas.

Lo siguiente es introducir la información acerca de las necesidades de potencia, dado que esta calculadora esta enfocada a las instalaciones de autoconsumo sin conexión a red.

Una vez introducidos estos datos, el programa realiza los cálculos y nos ofrece un resultado bastante detallado del campo fotovoltaico, el regulador de carga, la batería y el inversor que mejor encajaría con nuestro requisitos.

La principal diferencia con la aplicación que yo he desarrollado es requisitos versus limitaciones. Es decir, CalculationSolar.com te permite seleccionar tus requisitos de potencia y te indica el generador que vas a necesitar para poder hacer frente a dicha carga. En cambio, mi solución indica la potencia y energía que se puede llegar a generar con las limitaciones arquitectónicas impuestas por el usuario.

3. SISIFO [4]

La solución propuesta por el Grupo de Energías Fotovoltaicas del Instituto de Energía Solar de la Universidad Politécnica de Madrid es también un solución web para el calculo de instalaciones solares tanto de bombeo de agua como conectadas a red.

El sistema de input de información del usuario es muy completo. Paso por paso le permite a éste introducir desde los datos geográficos y meteorológicos hasta hasta los valores de

perdidas en los diferentes cableados del sistema. No obstante, los valores por defecto son muy válidos así que incluso un usuario con poco conocimiento sobre energía podría llegar a obtener unos resultados fiables.

Una vez realizada la simulación, la aplicación aporta multitud de resultados detallados tales como Irradiaciones en el plano horizontal e inclinado, temperaturas y energía producida.

Esta aplicación es la más parecida, salvando las distancias, a la que se desarrolla en este proyecto. Sin embargo, es posible que puede llegar a abrumar en cierta medida a un usuario que desea solamente conocer cual es la potencia o energía que puede llegar a producir en su casa, para saber si le va a resultar rentable la inversión.

4. PVGIS [5]

PVGIS es la solución desarrollada por el JRC (Centro de Investigación Conjunta), que forma parte del EU Science Hub. Consiste en una aplicación web que nos permite estimar la energía que puede llegar a producir una instalación fotovoltaica en función de los parámetros introducidos por el usuario.

Tras elegir la localización como primer paso, el programa nos abre la posibilidad de elegir el tipo de instalación, entre conectada a red, con seguimiento, o autónomo.

En este caso, para poder compararlo con la aplicación de este proyecto, se va a realizar la estimación de un sistema conectado a red. La principal diferencia se presenta cuando el programa solicita al usuario la potencia FV pico instalada, así como la tecnología y las perdidas porcentuales del sistema.

Una vez introducidos estos datos, la aplicación nos ofrece los resultados de producción de energía mensual del sistema.

A diferencia de esto, SolarCalc, solicita al usuario el dato de la superficie disponible, para estimar la potencia máxima y por consiguiente, la energía máxima.

5. System Advisor Model [6]

Esta aplicación desarrollada por el Laboratorio de Energías Renovables del Departamento de Energía de los Estados Unidos. Es un conjunto de soluciones enfocadas a facilitar la toma de decisiones relacionadas con el campo de las Energías Renovables. Entre sus funcionalidades se incluyen programas de cálculo de Fotovoltaica, Termosolar, Eólica, Geotermal o Biomasa.

Es una herramienta muy completa a la vez que compleja, con un enfoque centrado en el apartado económico y la rentabilidad. En cuanto al cálculo y la estimación de una instalación fotovoltaica, ofrece una amplia posibilidad de personalización de cada uno de los campos que afecta a dicho cálculo. Cada uno de los parámetros de la radiación, el módulo, el inversor, las sombras e incluso de la inversión y amortización de la instalación son totalmente ajustables.

Con ésta aplicación sucede como con alguna de las mencionadas anteriormente, que puede llevar a confusión a un usuario medio sin conocimiento relacionados con la fotovoltaica. Estos usuarios son el público objetivo de la aplicación de SolarCalc, desarrollada en este proyecto.

6. solaR [7]

Este paquete para R, desarrollado por Oscar Perpiñán, permite llevar a cabo estudios tanto del rendimiento de los sistemas fotovoltaicos como de la radiación solar. Incluye una serie de clases, métodos y funciones para calcular aspectos como la geometría solar, radiación solar

incidente sobre un generador, realizar el paso de un generador horizontal a un generador inclinado y orientado y simular el rendimiento de diferentes aplicaciones de la energía fotovoltaica.

A pesar de no ser una aplicación de estimación de instalaciones solares como tal, ha sido, junto con el libro [8], la base de contraste y referencia de los cálculos que se han sido necesario llevar a cabo para poder desarrollar la aplicación de SolarCalc.

Capítulo 3

Parte teórica y desarrollo del código

El proceso de cálculo que se va a seguir para la estimación completa de la instalación fotovoltaica conectada a red es el que se detalla en el libro de Oscar Perpiñán, tutor de este trabajo, denominado Energía Solar Fotovoltaica [8]. También se harán menciones a las presentaciones que se encuentran en el mismo enlace que el libro.

A lo largo de este capítulo, se utilizará el término **aplicación** para referirse a todo el conjunto de código relacionado tanto con proceso de obtención de todos los datos necesario como el de mostrar la información relevante al usuario.

A continuación se muestra un diagrama que resume los pasos del proceso que se ha seguido para llegar a los resultados deseados.

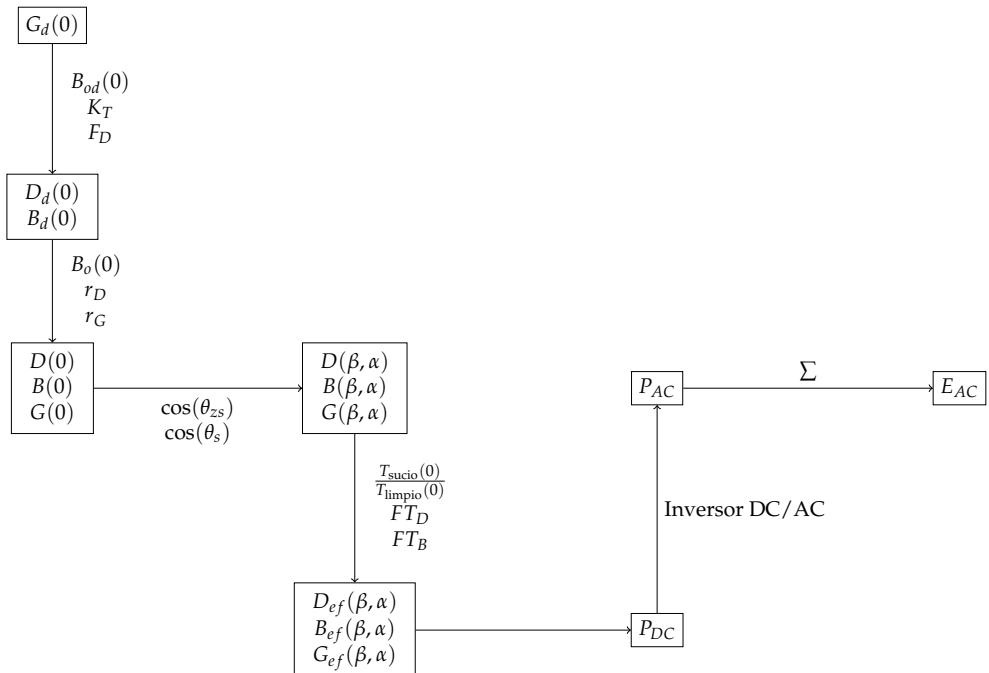


Figura 3.1: Procedimiento de cálculo

Los pasos que se muestran en el diagrama son:

1. Obtener la irradiación global diaria en el plano horizontal en el emplazamiento del usuario
2. Separar dicha irradiación en sus componentes de irradiación difusa y directa.
3. Convertir la irradiación diaria a un perfil horario de irradiación global, directa y difusa en en plano horizontal.
4. Pasar del perfil horario en el plano horizontal a un perfil con la inclinación y orientación indicada por el usuario.
5. Aplicar las pérdidas relacionadas con el ángulo de incidencia y el nivel de suciedad.
6. Calcular la potencia en corriente continua entregada por el generador.
7. Convertir la potencia en corriente alterna a través de inversor.
8. Realizar un sumatorio de las potencias en corriente alterna para obtener la energía entregada.

3.1. Naturaleza de la radiación solar

Para el cálculo de la radiación solar que finalmente incide en una área concreta localizada en la superficie terrestre debemos distinguir tres componentes diferenciados, comúnmente denominados:

- **Radiación Directa, B :** representa la porción de radiación procedente en linea directa del Sol.
- **Radiación Difusa, D :** representa toda la fracción de radiación procedente de todo el cielo, excepto del sol. Es decir, incluye todos aquellos rayos dispersados por la atmósfera. Por tanto, será dependiente de condiciones climatológicas como nubosidades, niebla o lluvia.
- **Radiación del albedo, R :** es aquella fracción de la radiación procedente de la reflexión con el suelo. Habitualmente, supone una contribución muy pequeña, que en algunos casos, como el de este proyecto, puede ser despreciada. Normalmente se toma en consideración en áreas con altos indices de reflexión como aquellos con temporadas largas de nieve.

La suma de las tres componentes constituye la denominada radiación global:

$$G = B + D + R \quad (3.1)$$

El Capítulo 3: Radiación solar, del libro mencionado anteriormente [8], describe el proceso que se ha de seguir para obtener una estimación de las componentes directa y difusa a partir del dato de radiación global, dado que es el que comúnmente se puede obtener de una localización determinada.

3.1.1. Radiación fuera de la atmósfera terrestre

Lo primero que se menciona en dicho proceso es la obtención de la irradiancia denominada extra-terrestre o extra-atmosférica, que es la radiación que llega a la atmósfera, directamente desde el sol, que no sufre ninguna perdida por interaccionar con algún medio. Como la relación entre el tamaño de nuestro planeta y la distancia entre el Sol y la Tierra es muy reducida, es posible asumir que el valor de dicha irradiancia es constante, siendo este valor $B_0 = 1367 \frac{W}{m^2}$, según varias mediciones llevadas a cabo.

Como la órbita que describe la Tierra alrededor del Sol no es totalmente circular, sino que tiene cierta excentricidad, a la hora de calcular la irradiancia incidente en una superficie tangente a la atmósfera en una latitud concreta, debemos aplicar un factor de corrección de la excentricidad, como se muestra en la ecuación:

$$B_0(0) = B_0 \epsilon_0 \cos \theta_{zs} \quad (3.2)$$

Para calcular esta irradiación diaria, debemos calcular primero los diferentes componentes que forman parte de esta:

- $B_0 = 1367 \frac{W}{m^2}$
- Factor de corrección por excentricidad: $\epsilon_0 = 1 + 0,033 \cdot \cos(2\pi d_n / 365)$

En código se representa de la siguiente manera:

```
1 const exct = 1 + 0.033 * Math.cos((2 * Math.PI * elem.normalDay) / 365)
```

Extracto de código 3.1: Factor de corrección por excentricidad

- Ecuación de Cooper para la declinación: $\delta = 23,45 \text{ deg } \sin\left(\frac{2\pi(d_n+284)}{365}\right)$

En código se representa de la siguiente manera:

```
1 const decl = 23.45 * Math.sin((2 * Math.PI * (elem.normalDay + 284)) / 365)
```

Extracto de código 3.2: Ecuación de Cooper para declinación

- Cenit Solar: $\cos(\theta_{zs}) = \cos(\delta) \cos(\omega) \cos(\phi) + \sin(\delta) \sin(\phi)$

Para el calculo del cenit solar, hay que tener en cuenta que varía en función de la hora del día, así que se deberá calcular un valor para cada una de las 24h del día, con la diferencia de que en el cálculo, las horas irán de -12 a 12 en lugar de 0 a 23 y además, tendremos que convertir las horas a ángulos multiplicando por 15° . De tal manera, que en código, el calculo del cenit solar, tendrá ésta representación:

```

1   for (let h = -12; h < 12; h++) {
2     const cosZenit =
3       Math.cos(deg2rad(elem.decl)) * Math.cos(deg2rad(h * 15)) *
4       Math.cos(deg2rad(newData.latitude)) +
5       Math.sin(deg2rad(elem.decl)) * Math.sin(deg2rad(newData.latitude))
6     const zenithVal = rad2deg(Math.acos(cosZenit))
7   }

```

Extracto de código 3.3: Cálculo del cenit solar

Como se puede observar, la irradiancia extra-terrestre solo requiere de consideraciones geométricas. Así, integrando la ecuación 3.2, podemos calcular la irradiación diaria extra-terrestre con la ecuación obtenida:

$$B_{0d}(0) = -\frac{T}{\pi} B_0 \epsilon_0 (\omega_s \sin \phi \sin \delta + \cos \phi \cos \delta \sin \omega_s) \quad (3.3)$$

Se puede llegar a demostrar que la media mensual de la irradiación diaria coincide en valor con la irradiación diaria de un día denominado día promedio. Por tanto, podemos calcular el valor medio mensual de la irradiación diaria extra-atmosférica con el valor de la declinación de uno de los doce días promedio.

Estos doce días promedios son:

Mes	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic
d_n	17	45	74	105	135	161	199	230	261	292	322	347

Cuadro 3.1: Días promedio

Aplicando estos cálculos a la ecuación 3.3 podemos calcular el valor de la irradiancia extra-terrestre diaria $B_{0d}(0)$:

```

1   const B0d0 =
2     -(24 / Math.PI) *
3     B0 *
4     elem.exct *
5     (elem.ws * Math.sin(deg2rad(newData.latitude)) * Math.sin(deg2rad(elem.decl)) +
6      Math.cos(deg2rad(elem.decl)) * Math.cos(deg2rad(newData.latitude)) *
7      Math.sin(elem.ws))

```

Extracto de código 3.4: Ecuación para B0d0

donde:

- B_0 : valor constante de 1367 W/m^2
- exct : excentricidad
- ω_s : ángulo amanecer
- decl : declinación

3.1.2. Cálculo de componentes de radiación solar

Para poder calcular la energía producida por un generador fotovoltaico, se necesita conocer la radiación solar que incide sobre la superficie de dicho generador. En éste caso las características del generador vienen indicadas por el usuario de la aplicación, por tanto, debemos recoger dicha información. La información necesaria para conocer la radiación que incide en la superficie son:

- **Latitud y longitud del emplazamiento:** La ruta que se ha tomado para obtener estos datos, es la de pedirle al usuario su dirección, o una dirección cercana a su localización, y utilizar la API de Google Maps¹ para convertir dicha dirección en las coordenadas de latitud y longitud que se necesitan para poder extraer la irradiación mencionada anteriormente.
- **Orientación, inclinación y nivel de suciedad de la superficie:** Estos valores son recogidos directamente de los campos de la página web, sin necesitar ningún trato especial.

En el anexo de programación se explica más detalladamente como se ha realizado la recogida de estos datos y su posterior envío a la API de Google.

Obtención de la irradiación global media en el plano horizontal

Como se puede observar en el diagrama 3.1, el primer paso del cálculo de una instalación fotovoltaica es el de conocer la irradiación global media en el plano horizontal para el emplazamiento donde se va a realizar el cálculo.

Anteriormente se obtuvieron las coordenadas de latitud y longitud del emplazamiento introducido por el usuario.

A continuación, se va a obtener la irradiación para las dichas coordenadas utilizando un servicio de ADRASE², un proyecto realizado por el CIEMAT. Este servicio ofrece de manera gratuita y de uso libre, datos correspondientes a valores medios mensuales de irradiación global en el plano horizontal para toda la geografía española.

Los datos están disponibles a través de un mapa interactivo y de unos enlaces personalizados que incluyen la valores de latitud y longitud para los que se desea obtener dicha información. En esos enlaces se ofrecen los datos de irradiación global mínima, media y máxima en el plano horizontal. Con el objetivo de no saturar la página de ADRASE y evitar errores de funcionamiento de la aplicación debidos a la posibilidad de que la página desaparezca en un futuro, se ha realizado una descarga de los datos de la página, con un intervalo de 0.1° tanto en longitud como en

¹API Google Maps: Enlace interactivo al que se le pueden enviar los datos de una dirección y devuelve las coordenadas de latitud y longitud de un emplazamiento.

²ADRASE: Acceso a Datos de Irradiación Solar en España <http://www.adrase.com/>

latitud y se han guardado en una base de datos. De esta forma también se reducen los tiempos de cálculo en gran medida al tener un acceso casi instantáneo a los valores medios de irradiación. Una explicación detallada de como se ha realizado dicha descarga se incluye en el la sección 5.2.3 del Anexo I.

Una vez obtenida la radiación global media para los doce meses, en el emplazamiento indicado, se puede comenzar el proceso de obtención de las componentes de ésta.

Obtención de las componentes de radiación global

En 1960, Liu y Jordan [9] expusieron una forma de caracterizar la radiación solar en un lugar, mediante el índice de claridad K_T . Éste índice relaciona la radiación global y la extra-terrestre, ambas en el plano horizontal. La expresión de el indice de claridad diario es:

$$K_{Td} = \frac{G_d(0)}{B_{0d}(0)} \quad (3.4)$$

mientras que el indice de claridad mensual es la relación entre las medias mensuales de irradiación diaria:

$$K_{Tm} = \frac{G_{d,m}(0)}{B_{0d,m}(0)} \quad (3.5)$$

En el código, el índice de claridad media se calcula con la siguiente expresión:

```

1  newData.meanValues.forEach((elem, index) => {
2      const Ktd = elem.meanGR / elem.B0d0
3  })

```

Extracto de código 3.5: Índice de claridad diario

Para cada uno de los valores medios mensuales, dividimos el valor de radiación global media entre el $B_{0d}(0)$ calculado anteriormente.

Habiendo calculado el índice de claridad, podemos utilizar su valor para calcular la Fracción de radiación difusa en el plano horizontal ($F_D = \frac{D(0)}{G(0)}$). Para ello, podemos utilizar la ecuación de Page para aproximar el dicho valor conociendo el índice de claridad.

La ecuación que define éste valor es:

$$F_{Dm} = 1 - 1,13K_{Tm} \quad (3.6)$$

En código, la expresión es:

```

1  newData.meanValues.forEach((elem, index) => {
2      const Fd = 1 - 1.13 * elem.Ktd
3  })

```

Extracto de código 3.6: Fracción de difusa

Donde elem.Ktd es el índice de claridad calculado en el paso anterior.

Al conocer, mediante la ecuación de Page el valor de la fracción de radiación difusa, podemos aplicar dicha relación para calcular los valores de radiación directa y difusa en el plano horizontal, a partir de la radiación global. Las dos ecuaciones necesarias para calcular estos valores son:

$$D_d(0) = F_D G_d(0) \quad (3.7)$$

$$B_d(0) = G_d(0) - D_d(0) \quad (3.8)$$

En código, las expresiones toma la siguiente forma:

```

1  newData.meanValues.forEach((elem, index) => {
2      const Dd0 = elem.Fd * elem.meanGR
3      const Bd0 = elem.meanGR - Dd0
4  })

```

Extracto de código 3.7: Radiación directa y difusa

Para cada uno de los doce valores de radiación global media, aplicamos la fracción calculada anteriormente para obtener los dos valores de radiación en el plano horizontal.

3.2. Radiación en superficies inclinadas

Habiendo ya obtenido las valores de irradiación diaria en el plano horizontal, y divididos en sus componentes directa y difusa. A continuación, para poder llevar a cabo las transformaciones al plano inclinado, debemos estimar los valores horarios de irradiancia difusa, directa y global en el plano horizontal. Con estas estimaciones podemos calcular los valores correspondientes al plano inclinado.

Integrando los valores de irradiancia se obtienen las estimaciones de irradiación diaria difusa, directa y global en el plano inclinado. Al realizar ésta transformación, la irradiancia/irradiación recibe el adjetivo de *incidente*.

A estos valores les aplicaremos las pérdidas por suciedad, añadiéndole el adjetivo de *efectiva* a la irradiancia e irradiación. Será esta irradiación incidente efectiva la que se utilizará para el cálculo de la energía producida por el generador.

3.2.1. Estimación de irradiancia a partir de irradiación diaria

Partiendo de los valores calculados anteriormente de irradiación diaria difusa, directa y global en el plano horizontal, podemos realizar la transformación al plano inclinado estimando el perfil de irradiancia correspondiente a cada valor de irradiación.

Como la variación solar durante una hora es relativamente baja, podemos suponer que el valor medio de irradiancia durante esa hora coincide numéricamente con irradiación horaria ($\frac{kWh}{m^2}$).

Por otro lado, mediante análisis de series temporales se ha demostrado que la relación entre la irradiancia y la irradiación difusa es equivalente a la existente entre la irradiancia y la irradiación extra-atmosférica.

$$r_D = \frac{D(0)}{D_d(0)} = \frac{B_0(0)}{B_{0d}(0)} \quad (3.9)$$

El factor r_D se puede calcular mediante la siguiente expresión:

$$r_D = \frac{\pi}{T} \cdot \frac{\cos(\omega) - \cos(\omega_s)}{\omega_s \cdot \cos(\omega_s) - \sin(\omega_s)} \quad (3.10)$$

Donde:

- T: Duración del día en horas
- ω_s : ángulo de amanecer, expresado en radianes
- ω : instante central de la hora correspondiente

La implementación en código es la siguiente

```

1 const rd = []
2 for (let h = -12; h < 12; h++) {
3     const hRad = Math.cos(deg2rad(h * 15))
4     const rdval = (Math.PI / 24) * ((hRad - Math.cos(elem.ws)) / (elem.ws *
5         Math.cos(elem.ws) - Math.sin(elem.ws)))
6     rd.push(rdval)
7 }
```

Extracto de código 3.8: Cálculo de rD

Para cada uno de los valores horarios de -12 a 12, se calcula el angulo horario multiplicando por 15 grados y se introduce en la ecuación, obteniendo 24 valores por cada uno de los meses.

El mismo análisis muestra también la relación entre la irradiancia e irradiación global asimilable a una función dependiente de la hora solar.

$$r_G = \frac{G(0)}{G_d(0)} r_D \cdot (a + b(\omega)) \quad (3.11)$$

siendo:

$$a = 0,409 - 0,5016 \cdot \sin(\omega_s + \frac{\pi}{3}) \quad (3.12)$$

$$a = 0,6609 - 0,4767 \cdot \sin(\omega_s + \frac{\pi}{3}) \quad (3.13)$$

donde ω_s es negativa y está expresada en radianes.

En código, los expresiones anteriores se representan de la siguiente forma:

```

1 const rg = []
2 const a = 0.409 - 0.5016 * Math.sin(elem.ws + Math.PI / 3)
3 const b = 0.6609 + 0.4767 * Math.sin(elem.ws + Math.PI / 3)
4 let i = 0
5 for (let h = -12; h < 12; h++) {
6     const hRad = Math.cos(deg2rad(h * 15))
7     const rgval = rd[i] * (a + b * hRad)
```

```

8     rg.push(rgval)
9     i++
10    }

```

Extracto de código 3.9: Cálculo de rG

3.2.2. Transformación al plano del generador

Para obtener los valores de irradiancia en el plano inclinado, lo primero que debemos calcular son los valores horarios de radiación global, directa y difusa en el plano horizontal, aplicando las dos relaciones, r_D y r_G que se han calculado en la sección anterior.

Para ello, emplearemos las siguientes ecuaciones:

$$D_h = r_D \cdot D_d(0) \quad (3.14)$$

$$G_h = r_G \cdot G(0) \quad (3.15)$$

$$B_h = G_h - D_h \quad (3.16)$$

En código, las expresiones serían las siguientes:

```

1  const hourlyValues = []
2  let hour = -12
3  const dawn = rad2deg(elem.ws) / 15
4  elem.dawn = dawn
5  for (let i = 0; i < 24; i++) {
6      const cosHour = Math.cos(deg2rad(hour * 15))
7      const cosWs = Math.cos(elem.ws)
8      if (cosHour > cosWs) {
9          const Dh = elem.rd[i] * elem.Dd0
10         const Gh = elem.rg[i] * elem.meanGR
11         const Bh = Gh - Dh
12         hourlyValues.push({
13             Bh, Dh, Gh, cosHour, cosWs,
14         })
15     } else {
16         hourlyValues.push({
17             Bh: 0, Dh: 0, Gh: 0, cosHour, cosWs,
18         })
19     }
20  hour++

```

```

21     }
22     elem.hourlyValues = hourlyValues

```

Extracto de código 3.10: Cálculo de rG

Se puede observar que para las horas en las que el coseno es menor que el coseno del ángulo de amanecer, se toman valores nulos para las radiaciones. Dado que en esas horas no existe radiación solar, aunque en algunos casos el cálculo pueda dar valores distintos de 0.

Una vez obtenidos estos valores horarios de irradiancia en el plano horizontal, el siguiente paso consiste en llevarlo al plano del generador.

El planteamiento de ésta transformación es el siguiente:

- **Irradiancia Directa** $B(\beta, \alpha)$: ecuación basada en geometría solar (ángulo cenital) y del generador (ángulo de incidencia).

$$B(\beta, \alpha) = B(0) \cdot \frac{\max(0, \cos(\theta_s))}{\cos(\theta_{zs})} \quad (3.17)$$

- **Irradiancia Difusa** $D(\beta, \alpha)$: Utilizando el modelo de cielo anisotrópico (Pág 31 [8]), se distinguen dos componentes de la irradiancia difusa, denominados *circunsolar* e *isotrópica*.

$$D(\beta, \alpha) = D^I(\beta, \alpha) + D^C(\beta, \alpha) \quad (3.18)$$

$$D^I(\beta, \alpha) = D(0) \cdot (1 - k_1) \cdot \frac{1 + \cos(\beta)}{2} \quad (3.19)$$

$$D^C(\beta, \alpha) = D(0) \cdot k_1 \cdot \frac{\max(0, \cos(\theta_s))}{\cos(\theta_{zs})} \quad (3.20)$$

$$k_1 = \frac{B_h}{B_0(0)} \quad (3.21)$$

- **Irradiancia de Albedo** $R(\beta, \alpha)$: Se considera nula por tener un porcentaje de aportación muy reducido.

Estos componentes sumados, forman la expresión de la irradiancia global en el plano del generador.

$$G(\beta, \alpha) = D(\beta, \alpha) + B(\beta, \alpha) + R(\beta, \alpha) \quad (3.22)$$

Ángulo de incidencia

En la ecuación de la irradiancia directa, uno de los elementos que influyen en el cálculo es el angulo de incidencia (Pág. 14 [8]), cuya expresión es:

$$\begin{aligned}
\cos(\theta_s) = & \text{sign}(\phi) \cdot [\sin(\beta) \cos(\alpha) \cos(\delta) \sin(\phi) - \\
& - \sin(\beta) \cos(\alpha) \cos(\phi) \sin(\delta)] + \\
& + \sin(\beta) \sin(\alpha) \cos(\delta) \sin(\omega) + \\
& + \cos(\beta) \cos(\delta) \cos(\omega) \cos(\phi) + \\
& + \cos(\beta) \sin(\delta) \sin(\phi)
\end{aligned} \quad (3.23)$$

Donde:

- β : Inclinación del generador
- α : Orientación del generador
- ϕ : Latitud del emplazamiento
- δ : Declinación
- ω : Hora solar

Traducida a código, la expresión será:

```

1   for (let h = -12; h < 12; h++) {
2       const betaRad = deg2rad(newData.angle)
3       const alphaRad = deg2rad(newData.orientation)
4       const hRad = deg2rad(h * 15)
5       const latRad = deg2rad(newData.latitude)
6       const declRad = deg2rad(elem.decl)
7       const anglIncidenciaVal =
8           Math.sign(newData.latitude) *
9           (Math.sin(betaRad) * Math.cos(alphaRad) * Math.cos(declRad) *
10            Math.cos(hRad) * Math.sin(latRad) - Math.sin(betaRad) *
11            Math.cos(alphaRad) * Math.cos(latRad) * Math.sin(declRad)) +
12            Math.sin(betaRad) * Math.sin(alphaRad) * Math.cos(declRad) *
13            Math.sin(hRad) + Math.cos(betaRad) * Math.cos(declRad) *
14            Math.cos(hRad) * Math.cos(latRad) + Math.cos(betaRad) *
15            Math.sin(declRad) * Math.sin(latRad)
16       const anglIncidencia = rad2deg(Math.acos(anglIncidenciaVal))
17       const anglIncidencia = rad2deg(Math.acos(anglIncidenciaVal))
18       incid.push(anglIncidencia)
19       elem.incid = incid
20   }

```

Extracto de código 3.11: Cálculo del ángulo de incidencia

Cálculo de las componentes

Teniendo el valor del ángulo de incidencia, podemos proceder a calcular las dos componentes de irradiancia para el plano del generador.

```

1 newData.meanValues.forEach((elem, index) => {
2     for (let i = 0; i < 24; i++) {
3         const zenithRad = deg2rad(elem.zenit[i])
4         const incidRad = deg2rad(elem.incid[i])
5         const betaRad = deg2rad(newData.angle)
6         const numerator = Math.max(0, Math.cos(incidRad))
7         const denominator = Math.cos(zenithRad)
8         const Btilt = elem.hourlyValues[i].Bh * (numerator / denominator)
9         const k1 = elem.hourlyValues[i].Bh / elem.B00[i]
10        const DcTilt = elem.hourlyValues[i].Dh * k1 * (numerator / denominator)
11        const DiTilt = elem.hourlyValues[i].Dh * (1 - k1) * ((1 + Math.cos(betaRad)) / 2)
12        const Dtilt = DcTilt + DiTilt
13        const Gtilt = Btilt + Dtilt
14        elem.hourlyValues[i] =
15            ...elem.hourlyValues[i],
16            Btilt,Dtilt,DcTilt,DiTilt,Gtilt,cosZenit: Math.cos(zenitRad),
17        }
18    }
19 })

```

Extracto de código 3.12: Cálculo del ángulo de incidencia

Con estos cálculos obtenemos la irradiancia global incidente, es decir, en el plano del generador. El siguiente es añadirle el apellido de *efectiva* aplicándole las pérdidas por ángulo de incidencia y suciedad.

3.2.3. Pérdidas por ángulo de incidencia y suciedad

Al tratarse de sistemas estáticos, la radiación que incide en el módulo se puede ver desviada de la perpendicular e incidir con ángulos diferentes. Esta desviación, cuantificada por el ángulo de incidencia [eq. 3.23], es causa de pérdidas por reflexión.

Por otro lado, la suciedad acumulada en la superficie del módulo altera las propiedades angulares del mismo y reduce la capacidad de transmisión del vidrio.

Estos dos fenómenos reducen la irradiancia que es aprovechada por el módulo. Para el caso de la radiación directa, la expresión de irradiancia efectiva es la siguiente:

$$B_{ef}(\beta, \alpha) = B(\beta, \alpha) \cdot \frac{T_{sucio}(0)}{T_{limpio}(0)} \cdot (1 - FT_B(\theta_s)) \quad (3.24)$$

donde $FT_B(\theta_s)$ es el factor de pérdidas angulares para la irradiancia directa, calculable mediante la ecuación:

$$FT_B(\theta_s) = \frac{\exp(-\frac{\cos(\theta_s)}{a_r}) - \exp(-\frac{1}{a_r})}{1 - \exp(-\frac{1}{a_r})} \quad (3.25)$$

Los valores del coeficiente de pérdidas angulares deben ser determinados de forma experimental. En la tabla quedan recogidos algunos valores característicos de un módulo de silicio monocristalino convencional para los diferentes grados de suciedad. Además en la tabla también se recogen los valores de la transmitancia al interior del módulo en incidencia normal respecto a uno limpio $\frac{T_{sucio}(0)}{T_{limpio}(0)}$.

Grado de suciedad	$\frac{T_{sucio}(0)}{T_{limpio}(0)}$	a_r	c_2
Limpio	1	0,17	-0.069
Bajo	0,98	0,20	-0.054
Medio	0,97	0,21	-0.049
Alto	0,92	0,22	-0.023

Cuadro 3.2: Coeficiente de pérdidas angulares y transmitancia relativa en incidencia normal para diferentes niveles de suciedad

El cálculo del coeficiente $FT_B(\theta_s)$, en código, toma este aspecto:

```

1 const FTB = (Math.exp(-Math.cos(tiltRad) / ar) -
2   Math.exp(-1 / ar)) / (1 - Math.exp(-1 / ar))
3 const Btilt = elem.hourlyValues[i].Btilt * TDirtTClean * (1 - FTB)

```

Extracto de código 3.13: Cálculo del coeficiente $FT_B(\theta_s)$

A la hora de aplicar dichas pérdidas a la componente difusa, debemos calcular cada una de las dos partes por separado. Para la componente de isotrópica existe otra expresión que depende del ángulo de inclinación del generador, del coeficiente de pérdidas angulares y de dos coeficientes de ajuste c_1 y c_2 . El primero de ellos toma un valor constante $c_1 = \frac{4}{3\pi}$. El segundo depende linealmente de a_r según se recoge en la tabla 3.2.3. En ésta expresión el ángulo β está en radianes.

$$FT_D \simeq \exp\left[-\frac{1}{a_r} \cdot \left(c_1 \cdot \left(\sin \beta + \frac{\pi - \beta - \sin \beta}{1 + \cos \beta}\right) + c_2 \cdot \left(\sin \beta + \frac{\pi - \beta - \sin \beta}{1 + \cos \beta}\right)^2\right)\right] \quad (3.26)$$

Para estas componentes el cálculo de irradiancia efectiva es similar al de la irradiancia directa. Para la componente difusa circunsolar emplearemos el factor de pérdidas angulares de la irradiancia efectiva:

$$D_{ef}^I(\beta, \alpha) = D^I(\beta, \alpha) \cdot \frac{T_{sucio}(0)}{T_{limpio}(0)} \cdot (1 - FT_D(\beta)) \quad (3.27)$$

$$D_{ef}^C(\beta, \alpha) = D^C(\beta, \alpha) \cdot \frac{T_{sucio}(0)}{T_{limpio}(0)} \cdot (1 - FT_B(\beta)) \quad (3.28)$$

En código las ecuaciones toman esta forma:

```

1   const expFTD =
2     -(1 / ar) *
3       (c1 * (Math.sin(betaRad) + (Math.PI - betaRad - Math.sin(betaRad)) / (1 +
4         Math.cos(betaRad))) +
5           c2 * Math.pow(Math.sin(betaRad) + (Math.PI - betaRad -
6             Math.sin(betaRad)) / (1 + Math.cos(betaRad)), 2))
7   const FTD = Math.exp(expFTD)
8
9   const DiTilt = elem.hourlyValues[i].DiTilt * TDirtTClean * (1 - FTD)
10  const DcTilt = elem.hourlyValues[i].DcTilt * TDirtTClean * (1 - FTB)

```

Extracto de código 3.14: Cálculo de las componentes efectivas

Antes de pasar a calcular la energía producida por el generador configurado, se va a llevar a cabo una filtración de datos para eliminar los valores horarios entre el atardecer y el amanecer. Para ello, aplicamos este algoritmo a los datos calculados anteriormente

```

1 newData.meanValues.forEach((elem, index) => {
2   const significantValues = []
3   elem.hourlyValues.forEach((value, index) => {
4     if (Math.abs(value.hour) < Math.abs(elem.dawn)) {
5       significantValues.push(value)
6     }
7   })
8   elem.significantValues = significantValues
9 })

```

Extracto de código 3.15: Selección de los valores clave

Como se observa, los valores horarios menores al amanecer, en valor absoluto, no se introducen en la lista de valores significativos.

3.3. Cálculo de la energía producida por el generador

Una vez calculados los valores de radiación en el plano del generador, el siguiente paso será proceder a estimar la energía que es capaz de producir dicho generador conectado a red.

En el caso de la aplicación desarrollada en el proyecto, el objetivo es estimar la energía máxima que será capaz de producir un sistema fotovoltaico conectado a red.

3.3.1. Definición de un SFCR

Un Sistema Fotovoltaico Conectado a Red (SFCR) es un sistema cuya función es producir energía eléctrica en unas condiciones específicas, de tal manera que ésta pueda ser inyectada en la red eléctrica convencional. Para ello, un SFCR se compone del propio generador fotovoltaico, un inversor DC/AC y un conjunto de protecciones eléctricas. Tal y como se muestra en la figura 3.2.

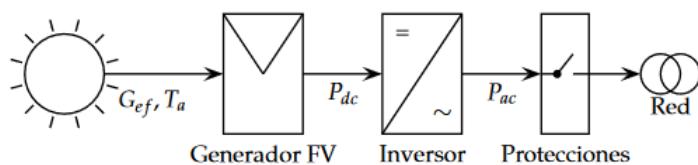


Figura 3.2: Esquema de un SFCR (Figura 6.1 pág 65 [8])

La energía producida por el sistema es consumida total o parcialmente por la unidad doméstica donde está instalada. La energía sobrante es inyectada a la red de distribución, y a cambio el propietario es compensado económicamente a través de sistemas de retribución tales como: la retribución con prima o el balance neto.

A diferencia de un sistema autónomo o de bombeo, el diseño de SFCR no necesita considerar un consumo mínimo a satisfacer. Con este mecanismo, el objetivo es que la producción anual del sistema sea la máxima posible.

3.3.2. Configuración de los elementos del sistema

A la hora de estimar la energía producida por un SFCR, debemos definir las características nominales de los componentes que forman parte de este, como se muestra en la figura 3.2. Con el objetivo de reducir al máximo la complejidad de la aplicación de cara al usuario final (cuyos conocimientos de fotovoltaica, de media, serán reducidos) se han fijado unos valores estándar para el módulo fotovoltaico, la configuración del generador y el inversor.

Estos valores se muestran en las siguientes tablas:

- **Configuración del módulo** El módulo elegido es uno disponible comercialmente, fabricado por JINKO SOLAR, modelo JKM320PP³. El área del modulo es de $1,957 \times 0,922m^2$ y tiene una potencia nominal de $320W_p$
- **Inversor:**

³Módulos solar JINKO SOLAR, modelo JKM320PP <https://autosolar.es/pdf/Ficha-Tecnica-Jinko-Solar-305-320W.pdf>

G^*	1000 $\frac{W}{m^2}$
V_{oc}^*	46,4 V
I_{sc}^*	9,05 A
V_{mpp}^*	37,4 V
I_{mpp}^*	8,56 A
N_{cs}	12
N_{cp}	6
TONC	45 °
T_c	25 °
V_t	0,025V
m	1,3

Cuadro 3.3: Configuración módulo standard

k_0	0,01
k_1	0,025
k_2	0,05
Potencia	40 kW
V_{min}	420 V
V_{max}	750

Cuadro 3.4: Configuración del inversor

- **Generador** La configuración del generador será de **12** módulos en serie y **11** módulos en paralelo.

3.3.3. Funcionamiento de una célula solar

Para poder llevar a cabo la estimación de la energía producida por el generador, debemos conocer como influye factores como la radiación o la temperatura en una célula solar y en los valores de tensión y corriente que se alcanzan en dichas condiciones.

Existen cuatro puntos clave que definen una célula solar, la corriente de cortocircuito (I_{sc}), la tensión de circuito abierto (V_{oc}) y la corriente y tensión en el punto de máxima potencia

Punto de máxima potencia

Por otro lado, existe un punto en la curva de funcionamiento del generador conocido como punto de máximo potencia, donde, como su propio nombre indica, los valores de tensión y corriente

son tales que la potencia que está entregando el generador es máxima, y por tanto, es interesante calcular dicho punto.

Factor de forma y eficiencia

Existe un valor que relaciona los valores más significativos de un módulo fotovoltaico, éste se conoce como factor de forma y su expresión es la siguiente:

$$FF = \frac{I_{mpp} \cdot V_{mpp}}{I_{sc} \cdot V_{oc}} \quad (3.29)$$

Este factor toma valores entre 0,7 y 0,8 variando poco de unas células a otras. Conociendo los valores de I_{sc} y V_{oc} es posible calcular la potencia en el punto de máxima potencia, dado que $P_{mpp} = FF \cdot I_{sc} \cdot V_{oc}$.

Por otra parte, la calidad de una célula se puede cuantificar según la ecuación de la eficiencia de conversión:

$$\eta = \frac{I_{mpp} \cdot V_{mpp}}{P_L} \quad (3.30)$$

donde P_L representa la potencia luminosa que incide en la célula.

Influencia de la temperatura y la radiación

Es importante tener en cuenta, a la hora de entender el funcionamiento de una célula solar, la influencia de los dos principales factores externos: la temperatura ambiente y la iluminación incidente o radiación.

El aumento de la temperatura tiene una influencia notable en la tensión de circuito abierto de la célula según el valor dV_{oc}/dT_c . donde T_c es la temperatura de la célula, dependiente de la temperatura ambiente y la radiación incidente. La forma de calcular ésta temperatura depende de las características constructivas del módulo que encapsula a la célula. Si el fabricante no nos ofrece información sobre este valor, para células de silicio cristalino se común tomar el siguiente valor:

$$\frac{dV_{oc}}{dT_c} = -2,3 \frac{mV}{^{\circ}C} \quad (3.31)$$

Sin embargo, el efecto que tiene la irradiancia sobre la tensión de circuito abierto es mucho menor que el de la temperatura. En el caso de éste proyecto, no se tendrá en cuenta dicha influencia.

Sin embargo, como se verá más adelante, en la sección 3.3.4, la radiación afecta a la corriente de cortocircuito, siendo de hecho el factor más significativo en la ecuación.

3.3.4. Comportamiento térmico de un módulo

La mayoría de las ecuaciones que definen el comportamiento de un módulo fotovoltaico se establecen en lo que se conocen como condiciones estándar de funcionamiento. En estas condiciones, la temperatura de la célula es de 25°C. Sin embargo, la temperatura de operación de la célula es diferente y depende directamente de la radiación que recibe el módulo en cada momento.

El módulo recibe una cantidad de radiación dada, absorbiendo la fracción de ésta que no se refleja al exterior. De dicha fracción parte es transformada en energía eléctrica mientras que el resto se entrega en forma de calor al entorno.

Para simplificar podemos asumir que el incremento de la temperatura de la célula respecto a la ambiente depende linealmente de la irradiancia incidente en ésta. El coeficiente de proporcionalidad depende de muchos factores, tales como el modo de instalación del módulo, la velocidad del viento, la humedad del ambiente o las características constructivas del cristal que cubre las células. Todos estos factores quedan recogidos en un valor único representado por la temperatura de operación nominal de la célula (NOCT o TONC), definida como aquella que alcanza una célula cuando su módulo trabaja en las siguientes condiciones:

- Irradiancia: $G = 800 \frac{W}{m^2}$.
- Espectro: el correspondiente $AM = 1,5$.
- Incidencia normal.
- Temperatura ambiente: $T_a = 20^\circ C$.
- Velocidad del viento: $v_v = 1 \frac{m}{s}$.

Este valor influye en la temperatura de la célula, como se ha mencionado anteriormente. La expresión que relaciona la temperatura de la célula T_c con la temperatura de operación nominal TONC es:

$$T_c = T_a + \frac{G_{ef} \cdot (TONC - 20)}{800} \quad (3.32)$$

Como se puede observar, en la temperatura de la célula influyen tanto la temperatura ambiente como la radiación global eficaz. Al trabajar con valores horarios, es necesario conocer el valor de la temperatura ambiente en cada hora del día.

Cálculo del perfil horario de temperaturas

El procedimiento para calcular los valores de temperatura ambiente horarios para cada uno de los meses en el emplazamiento indicado por el usuario no es trivial pues normalmente no suele haber un registro histórico de temperaturas. Para ello vamos a utilizar el proceso descrito en el documento [10].

En dicho documento se describe un proceso para desarrollar un perfil horario de temperaturas partiendo de una temperatura máxima y mínima.

Los pasos a seguir son:

1. **Obtención de la temperatura mínima y máxima:** Lo primero que necesitamos para poder realizar el perfil de temperatura es conocer los valores de temperatura mínima y máxima para cada uno de los meses. Para ello utilizaremos la API que nos proporciona AEMET a través de su plataforma de OPENDATA⁴. Para ello buscamos en su base de datos la estación climatológica más cercana a las coordenadas del usuario y solicitamos la información necesaria, en este caso las temperaturas mensuales de un año entero.

⁴AEMET OPENDATA: Base de datos abierta de información climatológica, entre otras <https://opendata.aemet.es>

2. Cálculo del ángulo de amanecer: Una vez obtenidos los valores de temperatura mensuales, el siguiente paso es calcular el ángulo de amanecer para cada uno de los 12 días normales, como se ha hecho en pasos anteriores mediante la declinación. Para ello empleamos las dos ecuaciones:

$$\delta = 23,45 \cdot \sin\left(\frac{2\pi \cdot (d_N + 284)}{365}\right) \quad (3.33)$$

$$\begin{aligned} \cos(\omega_s) &= -\tan(\delta) \cdot \tan(lat) \\ \omega_s &= \arccos(\omega_s) \end{aligned} \quad (3.34)$$

donde:

- d_N : Día normal correspondiente 3.1.1
- lat: Latitud del emplazamiento

3. Calcular los valores de T_m y T_r : En el documento mencionado, el primer paso para llegar al perfil de temperaturas, una vez obtenidos los valores de T_{max} y T_{min} es calcular los dos valores intermedios:

$$T_m = \frac{T_{max} + T_{min}}{2} \quad (3.35)$$

$$T_r = \frac{T_{max} - T_{min}}{2} \quad (3.36)$$

4. Calcular la matriz de T_a : Una vez obtenidos los dos valores intermedios de T_m y T_r , el siguiente paso es calcular la matriz de valores de temperatura ambiental T_a para cada hora. Para ello el documento nos indica que debemos calcular tres valores que denominamos a_1 , a_2 , a_3 . Utilizaremos estos tres valores para calcular T_1, T_2, T_3 , eligiendo uno u otro en función del ángulo solar. Las expresiones de estos tres valores se indican a continuación:

$$a_1 = \frac{12\pi \cdot (\omega_s - \omega)}{21\pi + 12 \cdot \omega_s} \quad (3.37)$$

$$a_2 = \frac{\pi \cdot (3\pi - 12\omega)}{3\pi - 12\omega} \quad (3.38)$$

$$a_3 = \frac{\pi \cdot (24\pi + 12 \cdot (\omega_s - \omega))}{21 \cdot \pi + 12 \cdot \omega_s} \quad (3.39)$$

donde ω es el ángulo solar y ω_s es el ángulo amanecer.

Con estos tres valores podemos calcular la otra terna de T_1, T_2 y T_3 con las siguientes ecuaciones:

$$T_1 = T_m - Tr \cdot \cos(a_1) \quad (3.40)$$

$$T_2 = T_m + Tr \cdot \cos(a_2) \quad (3.41)$$

$$T_3 = T_m - Tr \cdot \cos(a_3) \quad (3.42)$$

5. Selección del valor clave: en la matriz de temperatura ambiente introduciremos uno de los tres valores en función de si el ángulo solar es menor al del amanecer, si está entre en amanecer y $\frac{\pi}{4}$ o si es menor que $\frac{\pi}{4}$.

En código, todo este proceso toma la siguiente forma:

```

1  const wp = Math.PI / 4;
2  const Tm = (Tmax + Tmin) / 2;
3  const Tr = (Tmax - Tmin) / 2;
4  const Ta = [];
5  const decl = 23.45 * Math.sin((2 * Math.PI * (normalDays[index] + 284)) / 365);
6  const cosWs = -Math.tan(deg2rad(decl)) * Math.tan(deg2rad(latitude));
7  const ws = -Math.acos(cosWs);
8  for (let h = -12; h < 12; h++) {
9    const w = Math.cos(deg2rad(h * 15));
10   const a1 = (Math.PI * 12 * (ws - w)) / (21 * Math.PI + 12 * ws);
11   const a2 = (Math.PI * (3 * Math.PI - 12 * w)) / (3 * Math.PI - 12 * ws);
12   const a3 = (Math.PI * (24 * Math.PI + 12 * (ws - w))) / (21 * Math.PI + 12 * ws);
13   const T1 = Tm - Tr * Math.cos(a1);
14   const T2 = Tm + Tr * Math.cos(a2);
15   const T3 = Tm - Tr * Math.cos(a3);
16   if (w <= ws) {
17     Ta.push(T1);
18   } else if (w > ws && w <= wp) {
19     Ta.push(T2);
20   } else if (w > wp) {
21     Ta.push(T3);
22   }
23 }

```

Extracto de código 3.16: Cálculo de la temperatura ambiente

Habiendo obtenido la matriz de temperatura ambiente, el siguiente paso es utilizar estos valores para calcular la temperatura de la célula T_c con la ecuación 3.32 necesaria para calcular la tensión de circuito abierto V_{oc} . En código sería:

```

1  const cellTempProfiles = tempProfiles.map(({ hourlyTa, Tmax, Tmin }, indexA) => {
2    const meanValue = radiationData.meanValues[indexA];
3    const TCPprofile = hourlyTa.map((temp, indexB) => {
4      const Gef = meanValue.hourlyValues[indexB].Gtilt * 1000;
5      const Tc = temp + (Gef * (moduleData.TONC - 20)) / 800;
6      return Tc;
7    });
8    return { TCPprofile, month: index + 1, Tmax, Tmin };
9  });

```

Extracto de código 3.17: Cálculo de la temperatura de célula

Como se observa en el código, para cada uno de los valores horarios utilizamos la G_{ef} en el plano inclinado (en el código aparece como *Gtilt*) y la temperatura ambiente calculada anteriormente (en el código es *temp*) para calcular la T_c .

Cálculo de V_{oc} y I_{sc}

Conociendo ya los valores horarios de temperatura de la célula, procedemos a calcular la tensión de circuito abierto V_{oc} utilizando la expresión 3.43.

$$V_{oc}(T_c) = V_{oc}^* + (T_c - T_c^*) \cdot \frac{dV_{oc}}{dT_c} \cdot N_{cs} \quad (3.43)$$

En código quedaría de la siguiente forma:

```

1 const VocProfile = cellTempProfile.map(({ TCPProfile }, index) => {
2     const VocArray = TCPProfile.map((temp) => {
3         const Voc = moduleData.VocStar + (temp - moduleData.Tc)
4             * (-2.3 / 1000) * moduleData.Ncs;
5         return Voc;
6     });
7     return VocArray
8 });

```

Extracto de código 3.18: Cálculo de la tensión de circuito abierto

Por último, podemos calcular el valor de la corriente de cortocircuito I_{sc} que depende linealmente del valor de irradiancia según se indica en la expresión 3.44 que en código resulta de la siguiente forma:

$$I_{sc} = G_{ef} \cdot \frac{I_{sc}^*}{G^*} \quad (3.44)$$

```

1 const IscProfile = radiationData.meanValues.map(({ hourlyValues }) => {
2     const IscArray = hourlyValues.map(({ Gtilt }) => {
3         const IscValue = Gtilt * 1000 * (moduleData.IscStar / moduleData.Gstar);
4         return IscValue;
5     });
6     return IscArray;
7 });

```

Extracto de código 3.19: Cálculo de la corriente de cortocircuito

Factor de forma variable

Una vez obtenidos los valores de V_{oc} y I_{sc} , el siguiente paso ha de ser calcular los valores de tensión y corriente en el punto de máxima potencia, pues es donde el generador estará entregando su máxima potencia, como su propio nombre indica, y por tanto es un punto de interés para el cálculo.

Existen dos metodologías de cálculo de dicho punto, uno de ellos significantemente más sencillo que el otro. Éste consiste en suponer que el Factor de Forma, definido en la expresión 3.29 es constante.

Si suponemos que FF es constante, se podrían extraer los valores de tensión y corriente en el punto de máxima potencia ya que si

$$FF = FF^* \quad (3.45)$$

entonces

$$\frac{I_{mpp} \cdot V_{vmpp}}{I_{sc} \cdot V_{oc}} = \frac{I_{mpp}^* \cdot V_{vmpp}^*}{I_{sc}^* \cdot V_{oc}^*} \quad (3.46)$$

pudiendo así obtener los valores de I_{mpp} y V_{vmpp} .

Sin embargo, hacer esta suposición nos aleja en cierta manera de ofrecer una estimación lo más próxima a la realidad posible.

Por tanto, para intentar aproximarnos lo máximo posible a una estimación acertada, eliminaremos dicha suposición y tendremos en cuenta la variación del factor de forma. Para ello, utilizaremos el siguiente proceso:

1. **Cálculo de la tensión térmica, V_t , a temperatura de la célula:** Calcularemos el valor de V_t a 25 °C con la expresión:

$$V_{tn} = \frac{V_t \cdot (273 + 25)}{300} \quad (3.47)$$

```
1 const Vtn = (moduleData.Vt * (273 + 25)) / 300;
```

Extracto de código 3.20: Cálculo de V_{tn}

2. **Cálculo de R_s^* :** El segundo paso consiste en calcular el valor de resistencia en serie con los valores STC:

$$R_s^* = \frac{\frac{V_{oc}^*}{N_{cs}} - \frac{V_{mpp}^*}{N_{cs}} + m \cdot V_{tn} \cdot \ln \left(1 - \frac{I_{sc}^*}{I_{mpp}^*} \right)}{\frac{I_{mpp}^*}{N_{cp}}} \quad (3.48)$$

En código quedaría:

```
1 const RsStar =
2   (moduleData.VocStar / moduleData.Ncs -
3    moduleData.VmppStar / moduleData.Ncs +
4    moduleData.m * Vtn * Math.log(1 - moduleData.ImppStar / moduleData.IscStar)) /
5    (moduleData.ImppStar / moduleData.Ncp);
```

Extracto de código 3.21: Cálculo de R_s^*

3. **Cálculo de r_s :** Utilizando el valor R_s^* calculado en el paso anterior junto con los valores de V_{oc} y I_{sc} calculados anteriormente podemos calcular r_s que se utilizará más adelante en proceso. La ecuación que lo define viene dada por

$$r_s = R_s^* \cdot \left(\frac{N_{cs}}{N_{cp}} \cdot \frac{I_{sc}}{V_{oc}} \right) \quad (3.49)$$

El equivalente en código es

```

1 const rs = VocProfile.map((VocArray, arrIndex) => {
2     return VocArray.map((VocValue, valIndex) => {
3         return RsStar * ((moduleData.Ncs / moduleData.Ncp)
4             * (IscProfile[arrIndex][valIndex] / VocValue));
5     });
6 });

```

Extracto de código 3.22: Cálculo de r_s

Como se puede observar en la línea 4 del código, el valor de I_{sc} viene acompañado por dos índices entre corchete, esto es porque en todo momento estamos trabajando con valores horarios de cada día promedio correspondiente a cada uno de los 12 meses del año, por tanto, estamos trabajando una matriz de 12x24, de ahí el uso de los dos índices.

4. **Cálculo de k_{oc} :** A continuación, utilizando los valores de temperatura ambiente obtenidos con anterioridad junto con la tensión de circuito abierto, calcularemos k_{oc} mediante la expresión:

$$k_{oc} = \frac{V_{oc}/N_{cs}}{m \cdot V_t \cdot \frac{T_c+273}{300}} \quad (3.50)$$

En código tendría este forma:

```

1 const koc = VocProfile.map((VocArray, arrIndex) => {
2     return VocArray.map((VocValue, valIndex) => {
3         return VocValue / moduleData.Ncs / (moduleData.m * ((moduleData.Vt *
4             (cellTempProfile[arrIndex].TCPProfile[valIndex] + 273)) / 300));
5     });
6 });

```

Extracto de código 3.23: Cálculo de k_{oc}

Con éstos cálculos previos, éste método propone localizar el punto de máxima potencia de forma aproximada mediante las ecuaciones:

$$i_{mpp} = 1 - \frac{D_M}{k_{oc}} \quad (3.51)$$

$$v_{mpp} = 1 - \frac{\ln(k_{oc}/D_M)}{k_{oc}} - r_s \cdot i_{mpp} \quad (3.52)$$

donde:

$$D_M = D_{M0} + 2 \cdot r_s \cdot D_{M0}^2 \quad (3.53)$$

$$D_{M0} = \frac{k_{oc} - 1}{k_{oc} - \ln k_{oc}} \quad (3.54)$$

En código el proceso a seguir para llegar a los dos valores de i_{mpp} y v_{mpp} es el siguiente:

```

1 const DMO = koc.map((kocArray, arrIndex) => {
2   return kocArray.map((kocValue, valIndex) => {
3     return (kocValue - 1) / (kocValue - Math.log(kocValue));
4   });
5 });

```

Extracto de código 3.24: Cálculo de D_{M0}

```

1 const DM = DMO.map((DMOArray, arrIndex) => {
2   return DMOArray.map((DMOValue, valIndex) => {
3     return DMOValue + 2 * rs[arrIndex][valIndex] * Math.pow(DMOValue, 2);
4   });
5 });

```

Extracto de código 3.25: Cálculo de D_M

```

1 const impp = DM.map((DMArry, arrIndex) => {
2   return DMArry.map((DMValue, valIndex) => {
3     return 1 - DMValue / koc[arrIndex][valIndex];
4   });
5 });

```

Extracto de código 3.26: Cálculo de i_{mpp}

```

1 const vmp = impp.map((imppArry, arrIndex) => {
2   return imppArry.map((imppValue, valIndex) => {
3     return 1 - Math.log(koc[arrIndex][valIndex] /
4       DM[arrIndex][valIndex]) / koc[arrIndex][valIndex] -
5       rs[arrIndex][valIndex] * imppValue;
6   });
7 });

```

Extracto de código 3.27: Cálculo de v_{mpp}

Por último, multiplicando los valores de i_{mpp} y v_{mpp} por I_{sc} y V_{oc} respectivamente, obtendremos los valores de I_{mpp} y V_{mpp} que serán los que utilizaremos para calcular la potencia entregada por el generador en el punto de máxima potencia.

```

1 const Vmpp = vmpp.map((vmppArray, arrIndex) => {
2   return vmppArray.map((vmppValue, valIndex) => {
3     return vmppValue * VocProfile[arrIndex][valIndex];
4   });
5 });
6 const Impp = impp.map((imppArray, arrIndex) => {
7   return imppArray.map((imppValue, valIndex) => {
8     return imppValue * IscProfile[arrIndex][valIndex];
9   });
10 });
11

```

Extracto de código 3.28: Cálculo de V_{mpp} y I_{mpp}

Una vez calculados los valores de I_{mpp} y V_{mpp} podemos pasar a obtener los valores horarios de la potencia en el MPP, simplemente multiplicando los valores de corriente y tensión: $P_{mpp} = I_{mpp} \cdot V_{mpp}$.

```

1 const calculatePmpp = (ImppProfile, VmppProfile) => {
2   const PmppProfile = ImppProfile.map((ImppArray, dayIndex) => {
3     const PmppArray = ImppArray.map((ImppValue, hourIndex) => {
4       return ImppValue * VmppProfile[dayIndex][hourIndex];
5     });
6     return PmppArray;
7   });
8   return PmppProfile;
9 };
10

```

Extracto de código 3.29: Cálculo de P_{mpp}

3.3.5. Cálculo final de potencias y energías

La potencia que obtenemos es la de un solo módulo. Para conocer la potencia que va a ser capaz de entregar el generador, debemos tener en cuenta su configuración de módulos en serie y en paralelo, y multiplicar estos valores de la siguiente manera:

```

1 const calculateGeneratorPower = (PmppProfile) => {
2   const PdcProfile = PmppProfile.map((PmppArray) => {
3     return PmppArray.map((PmppValue) => {
4       return PmppValue * generatorData.seriesModules * generatorData.parallelModules;
5     });
6   });
7   return PdcProfile;
8 };
9

```

Extracto de código 3.30: Cálculo de la potencia del generador

Con este paso obtenemos la potencia horaria entregada por el generador fotovoltaico. El siguiente paso será pasar esa potencia a través del inversor y calcular la potencia a la salida de este. Para ello debemos llevar a cabo el siguiente desarrollo.

Lo primero, establecemos las expresiones de las potencias normalizadas Siendo P_{inv} la potencia nominal del inversor:

$$p_i = \frac{P_{DC}}{P_{inv}} \quad (3.55)$$

$$p_o = \frac{P_{AC}}{P_{inv}} \quad (3.56)$$

Para calcular la estos dos valores, utilizaremos un modelo empleado en inversores fotovoltaicos:

$$\eta_{inv} = \frac{p_o}{p_o + k_0 + k_1 p_o + k_2 p_o^2} \quad (3.57)$$

donde: k_0 , k_1 y k_2 son tres coeficientes ofrecidos por el fabricante del inversor.

Por otro lado, por definición de rendimiento, tenemos:

$$\eta_{inv} = \frac{p_o}{p_i} \quad (3.58)$$

Si despejamos las dos ecuaciones anteriores:

$$p_i = p_o + k_0 + k_1 p_o + k_2 p_o^2 \quad (3.59)$$

A partir de la expresión anterior obtenemos una ecuación de segundo grado con p_o como incógnita:

$$k_2 p_o^2 + (k_1 + 1)p_o + (k_0 - p_i) = 0 \quad (3.60)$$

Y por último, si conocemos la p_o podemos calcular la potencia en corriente alterna que entrega el generador:

$$P_{AC} = p_o \cdot p_{inv} \quad (3.61)$$

En código, el desarrollo quedaría de la siguiente manera:

1. **Cálculo de p_i :** A partir de la potencia calculada anteriormente y conociendo la potencia del inversor, podemos calcular la p_i de esta manera:

```

1 const calculatePiProfile = (PdcProfile) => {
2   return PdcProfile.map((PdcArray) => {
3     return PdcArray.map((PdcValue) => {
4       return PdcValue / inverterData.power;
5     });
6   });
7 };

```

Extracto de código 3.31: Cálculo de p_i

2. **Cálculo de p_o :** Utilizando p_i podemos despejar la ecuación de segundo grado y calcular el valor de p_o eligiendo la solución que cumpla que $p_o > 0$:

```

1 const calculatePoProfile = (PiProfile) => {
2   return PiProfile.map((PiArray) => {
3     return PiArray.map((PiValue) => {
4       if (PiValue <= 0) {
5         return 0;
6       }
7       const a = inverterData.k2;
8       const b = inverterData.k1 + 1;
9       const c = inverterData.k0 - PiValue;
10      const s1 = (-b + Math.sqrt(b * b - 4 * a * c)) / (2 * a);
11      const s2 = (-b - Math.sqrt(b * b - 4 * a * c)) / (2 * a);
12      return s1 > 0 ? s1 : s2;
13    });
14  });
15};

```

Extracto de código 3.32: Cálculo de p_o

3. **Cálculo de P_{inv} :** Habiendo calculado la p_o , podemos despejar la potencia generada por el generador (P_{inv}) de la ecuación 3.56:

```

1 const calculatePacProfile = (PoProfile) => {
2   return PoProfile.map((PoArray) => {
3     return PoArray.map((PoValue) => {
4       return PoValue * inverterData.power;
5     });
6   });
7 };

```

Extracto de código 3.33: Cálculo de P_{AC}

Finalmente, conociendo las potencias en corriente continua y en alterna, podemos calcular la energía mensual y anual que es capaz de producir el generador configurado. Para ello, en el caso del balance mensual, sumaremos los 24 valores horario que hemos obtenido, y los multiplicaremos por 30.

Por otro lado, para calcular la energía anual, simplemente sumaremos los 12 valores de energía mensual que hemos calculado.

```

1 const calculateMonthlyEac = (PacProfile) => {
2   return PacProfile.map((PacArray) => {
3     return (
4       30 * PacArray.reduce((total, currentValue) => {
5         return total + (currentValue > 0 ? currentValue : 0);
6         EacCell.textContent = MonthlyEac[index];
7       }, 0)))};

```

Extracto de código 3.34: Cálculo de la energía mensual en alterna

```

1 const calculateMonthlyEdc = (PdcProfile) => {
2   return PdcProfile.map((PdcArray) => {
3     return (
4       30 * PdcArray.reduce((total, currentValue) => {
5         return total + (currentValue > 0 ? currentValue : 0);
6       }, 0)
7     );
8   });
9 };

```

Extracto de código 3.35: Cálculo de la energía mensual en continua

```
1 const calculateAnualEnergy = (MonthlyEnergy) => {  
2     return MonthlyEnergy.reduce((total, currentValue) => {  
3         return total + (currentValue > 0 ? currentValue : 0);  
4     }, 0); };
```

Extracto de código 3.36: Cálculo de la energía anual

Capítulo 4

Ejemplo práctico de aplicación

Una vez llevado a cabo el desarrollo teórico de la aplicación y los cálculos necesarios para llegar a los resultados deseados, el siguiente paso será aplicar estos a un emplazamiento concreto, observar los resultados obtenidos y analizarlos para sacar una conclusión.

Éste capítulo también servirá como una guía detallada de uso de la aplicación, explicando la información a introducir en cada paso y como obtenerlos.

La aplicación está disponible a través del enlace <http://solarcalc.app> para utilizarse de manera totalmente gratuita.

4.1. Obtención de datos del usuario

El formulario que se le presenta al usuario para que éste introduzca la información de su emplazamiento es el siguiente:

Datos del emplazamiento

Dirección

Municipio **Código postal**

Datos de la casa

Inclinación del tejado en grados **Área del tejado en metros cuadrados**

Orientación **Nivel de suciedad**

Enviar

Figura 4.1: Formulario web para obtención de datos

Como se puede observar, el formulario consta de dos partes, por un lado está la dirección como tal,

que se utilizará para obtener las coordenadas del emplazamiento y por otro está la configuración física de la superficie destinada a la instalación de los paneles solares.

4.1.1. Latitud y longitud a partir de la dirección

La primera parte del formulario consiste en obtener la información acerca del emplazamiento del usuario, es decir, su latitud y longitud. Estos datos son los que se utilizarán en el código, para obtener la información de la radiación en dicho lugar. El proceso completo para obtener los datos de radiación se describe en 5.2.3

Datos del emplazamiento

Direccion	
<input type="text"/> Calle	
Municipio	Código postal
<input type="text"/> Municipio	<input type="text"/> # Código postal

Figura 4.2: Campos del formulario para la dirección

Para ello, el formulario pide una dirección, que no necesariamente tiene que ser la exacta del emplazamiento, sino que se puede utilizar una dirección genérica cercana al lugar donde se va a realizar la instalación, ya que para una distancia relativamente corta la radiación solar no variará de manera notable.

En la sección 5.2.1 se expone en detalle el proceso para obtener las coordenadas en función de la dirección del usuario.

En este caso práctico la dirección que utilizaremos será la de Guadalajara. Para ésta dirección, las coordenadas que Google nos devuelve son:

- **Latitud:** $40,632^\circ$.
- **Longitud:** $-3,166^\circ$.

4.1.2. Datos de la superficie destinada a la instalación

Una vez obtenidas las coordenadas del emplazamiento, es necesario conocer la inclinación, la orientación y el nivel de suciedad de la superficie, para poder realizar el cambio de la radiación en el plano horizontal obtenida anteriormente a la radiación eficiente incidente.

Datos de la superficie

Inclinación del tejado en grados	Área del tejado en metros cuadrados
<input type="text"/> Inclinación del tejado	<input type="text"/> Área del tejado
Orientación	Nivel de suciedad
<input type="text"/> 0	<input type="text"/> Nivel bajo

Figura 4.3: Campos del formulario para los datos de la superficie

El campo del nivel de suciedad es un desplegable ya que solamente existen 4 niveles, que afectan a los cálculos de una manera específica, como se ha visto en el capítulo teórico, por tanto, debemos limitar la forma en la que el usuario introduce esa información.

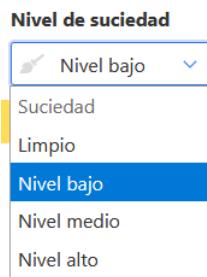


Figura 4.4: Campos del formulario para los datos de la superficie

Para ello el formulario nos ofrece 4 campos de información:

- **Inclinación en grados:** Para obtener la inclinación de la superficie en grados se puede utilizar cualquier smartphone actual dotado con un sensor giroscópico y una aplicación de tipo nivel, algún dispositivo de medición de ángulos como un transportador.
- **Orientación:** Para la obtener la orientación de la superficie se puede utilizar una brújula, ya sea analógica o digital. Se considera que 0 grados es el SUR.
- **Área del tejado:** Una estimación aproximada del área disponible el área que se desea cubrir con paneles solares.
- **Nivel de suciedad:** Este campo es el mas subjetivo y será a criterio del usuario. Se recomienda realizar cálculos con el peor caso para establecer un margen pero, no obstante, si se conoce con seguridad cuál es el nivel de suciedad de la zona, se ha de usar éste.

Un proceso detallado de como obtener los valores de inclinación y orientación se describe en capítulo 5. TODO

4.2. Aplicación de los datos al proceso de cálculo

En la sección anterior hemos recogido los datos que un usuario ha introducido a través del formulario. Estos datos son:

- **Latitud:** $40,632^\circ$.
- **Longitud:** $-3,166^\circ$.
- **Inclinación de la superficie:** 20° .
- **Área de la superficie:** $40m^2$.
- **Orientación de la superficie:** 30° .
- **Nivel de suciedad:** Medio.

A continuación, utilizaremos estos datos para recorrer numéricamente el proceso teórico descrito en el capítulo anterior.

4.2.1. Valores medios mensuales de radiación global

Tal y como se menciona en la sección 3.1.2, el primer paso para poder llevar a cabo el proceso de cálculo de la radiación incidente efectiva es obtener la radiación global media para cada uno de los doce meses, en el emplazamiento indicado por el usuario.

Para las coordenadas introducidas por el usuario, el punto con información mas cercano que tenemos es de latitud $40,57^\circ$ y longitud $-3,16^\circ$.

El proceso de obtención de los siguientes valores se describe en detalle en la sección 5.2.3.

Para este punto, los valores de irradiación media son:

kWh/m^2	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sept	Oct	Nov	Dic
Valor medio	2,0	3,1	4,8	5,7	6,8	8,0	7,8	6,8	5,1	3,5	2,2	1,7

Cuadro 4.1: Irradiación global media mensual



Figura 4.5: Radiación global en el emplazamiento

4.2.2. Irradiancia extra-terrestre diaria

Además, por otro lado, para poder continuar con el proceso de cálculo, es necesario calcular la irradiancia extra-terrestre diaria, utilizando la latitud indicada por el usuario y los días promedio de la tabla 3.1.1. En las ecuaciones de la sección 3.1.1 se indican los pasos a seguir y el resultado es:

kW/m^2	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sept	Oct	Nov	Dic
$B_{0d}(0)$	4,12	5,48	7,47	9,57	11,01	11,59	11,26	10,01	8,05	5,90	4,30	3,58

Cuadro 4.2: Irradiancia extra-terrestre diaria



Figura 4.6: Irradiación extra-terrestre

4.2.3. Separación de la radiación global horizontal en sus componentes

Una vez conocidos los valores mensuales de la radiación global en el plano horizontal, el siguiente paso se centra en separar la radiación global en sus dos componentes, la directa y la difusa, tal y como se explica en la sección 3.1.2.

Como podemos observar en la ecuación 3.4, el primer paso para separar la radiación global en sus dos componentes es calcular el índice de claridad, para que posteriormente calculemos la fracción de radiación difusa.

Si aplicamos dicha ecuación a los resultados anteriores obtenemos los siguientes resultados:

	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sept	Oct	Nov	Dic
K_{Td}	0,485	0,565	0,642	0,595	0,617	0,690	0,692	0,679	0,632	0,592	0,511	0,461

Cuadro 4.3: Índice de claridad

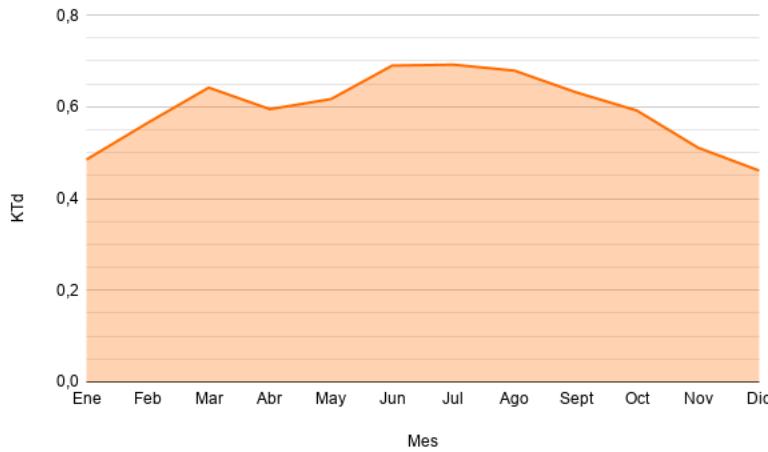


Figura 4.7: Índice de claridad

A continuación, utilizando la ecuación de Page 3.6, con el índice de claridad calculado, podemos calcular la fracción de difusa:

	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sept	Oct	Nov	Dic
F_D	0,452	0,361	0,274	0,327	0,302	0,220	0,218	0,232	0,285	0,330	0,421	0,478

Cuadro 4.4: Fracción de difusa

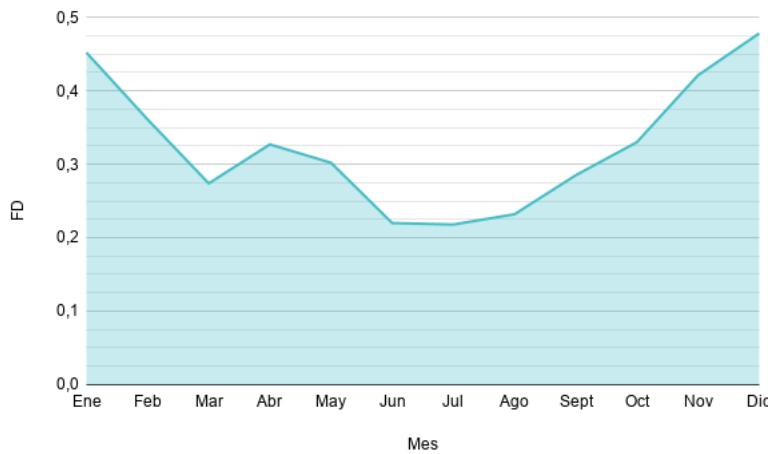


Figura 4.8: Fracción de difusa

Como era de esperar, se observa que la fracción de difusa es más elevada en los meses con mayor nubosidad y menos intensidad solar.

Cálculo de las componentes de la radiación en el plano horizontal

Con estos valores, utilizando las ecuaciones 3.7 y 3.8 podemos obtener las dos componentes de la radiación en el plano horizontal.

kWh/m^2	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sept	Oct	Nov	Dic
$B_d(0)$	1,096	1,980	3,484	3,835	4,745	6,239	6,103	5,220	3,647	2,344	1,272	0,886
$D_d(0)$	0,904	1,120	1,316	1,865	2,055	1,761	1,697	1,580	1,453	1,156	0,928	0,814

Cuadro 4.5: Irradiación directa y difusa en el plano horizontal

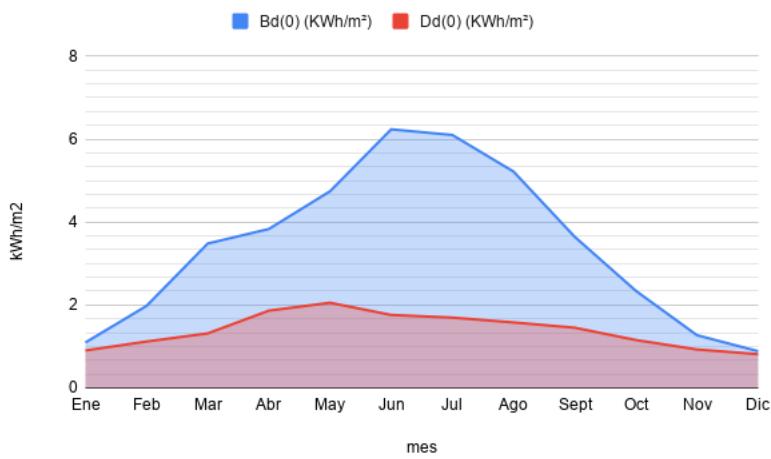


Figura 4.9: Componentes de la irradiación en el plano horizontal

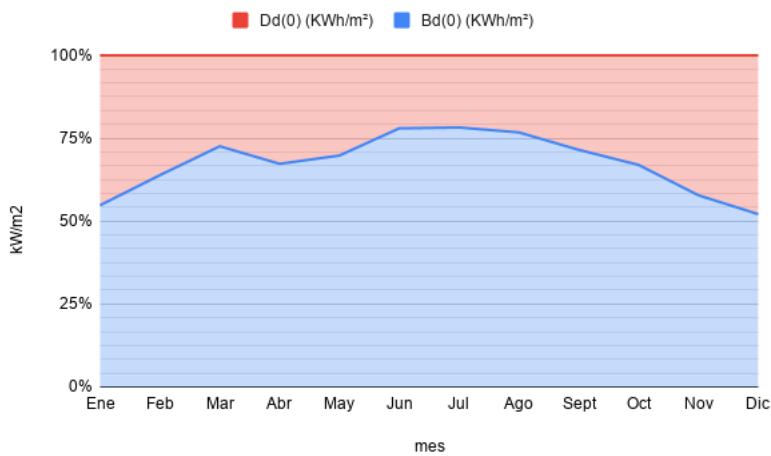


Figura 4.10: Componentes de la irradiación en el plano horizontal

La primera gráfica muestra los valores absolutos de las dos componentes, que la segunda muestra el porcentaje de cada una de las dos componentes. Una vez más, se observa una aportación mayor de la radiación difusa en los meses de invierno.

4.2.4. Irradiancia en la superficie inclinada

Irradiancia en el plano horizontal

Conociendo ya los valores diarios de la radiación directa y difusa en el plano horizontal, el próximo paso es obtener los valores horarios de irradiancia, tanto directa como difusa, en el plano horizontal para poder llevar a cabo posteriormente, la traslación de estos valores al plano inclinado. El proceso se describe detalladamente en las secciones 3.2.1 y 3.2.2.

Al tratarse de 24 valores horarios, para cada uno de los 12 meses, en total tendríamos 288 valores. Por tanto, y con el fin de no saturar este documento con demasiado valores, se van a indicar los resultados de solamente uno de los meses, en concreto de julio. Estos resultados se muestran en la tabla 4.6

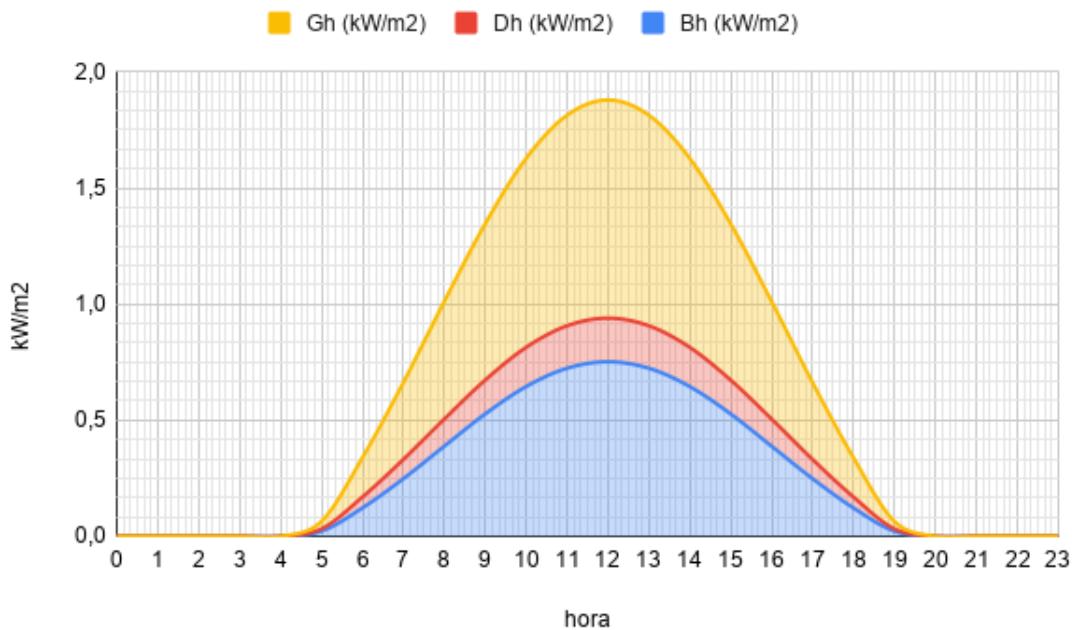


Figura 4.11: Irradiación horaria para el día promedio del mes de julio

hora	r_D	r_G	$B(0)$ (kW/m^2)	$D(0)$ (kW/m^2)	$G(0)$ (kW/m^2)
0	-0,056	-0,027	0	0	0
1	-0,053	-0,026	0	0	0
2	-0,045	-0,024	0	0	0
3	-0,031	-0,018	0	0	0
4	-0,014	-0,009	0	0	0
5	0,006	0,004	0,022	0,010	0,032
6	0,027	0,022	0,122	0,047	0,169
7	0,049	0,042	0,248	0,083	0,331
8	0,069	0,065	0,388	0,117	0,506
9	0,086	0,086	0,527	0,146	0,673
10	0,099	0,104	0,645	0,169	0,814
11	0,108	0,116	0,724	0,183	0,907
12	0,111	0,121	0,752	0,188	0,940
13	0,108	0,116	0,724	0,183	0,907
14	0,099	0,104	0,645	0,169	0,814
15	0,086	0,086	0,527	0,146	0,673
16	0,069	0,065	0,388	0,117	0,506
17	0,049	0,042	0,248	0,083	0,331
18	0,027	0,022	0,122	0,047	0,169
19	0,006	0,004	0,022	0,010	0,032
20	-0,014	-0,009	0	0	0
21	-0,031	-0,018	0	0	0
23	-0,045	-0,024	0	0	0
23	-0,053	-0,026	0	0	0

Cuadro 4.6: Irradiancia directa, difusa y global en el plano horizontal para el día promedio del mes de julio

Traslación de los valores al plano inclinado

Una vez obtenidos los valores horarios en el plano horizontal, podemos trasladar estos al plano inclinado con el proceso descrito de la ecuación 3.17 hasta la ecuación 3.23.

hora	$B(\beta, \alpha)$ (kW/m ²)	$D^C(\beta, \alpha)$ (kW/m ²)	$D^I(\beta, \alpha)$ (kW/m ²)	$D(\beta, \alpha)$ (kW/m ²)	$G(\beta, \alpha)$ (kW/m ²)
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0,006	0,006	0,006
6	0	0	0,027	0,027	0,027
7	0,12	0,018	0,044	0,062	0,184
8	0,280	0,042	0,057	0,099	0,379
9	0,448	0,067	0,065	0,132	0,580
10	0,603	0,091	0,069	0,160	0,763
11	0,721	0,108	0,072	0,180	0,901
12	0,786	0,118	0,072	0,190	0,977
13	0,787	0,119	0,072	0,191	0,978
14	0,724	0,109	0,070	0,179	0,903
15	0,610	0,092	0,065	0,156	0,767
16	0,463	0,069	0,057	0,127	0,589
17	0,304	0,046	0,044	0,090	0,394
18	0,156	0,023	0,027	0,050	0,206
19	0,033	0,005	0,006	0,011	0,044
20	0	0	0	0	0
21	0	0	0	0	0
22	0	0	0	0	0
23	0	0	0	0	0

Cuadro 4.7: Irradiancia directa, difusa y global en el plano inclinado, para el día promedio del mes de julio

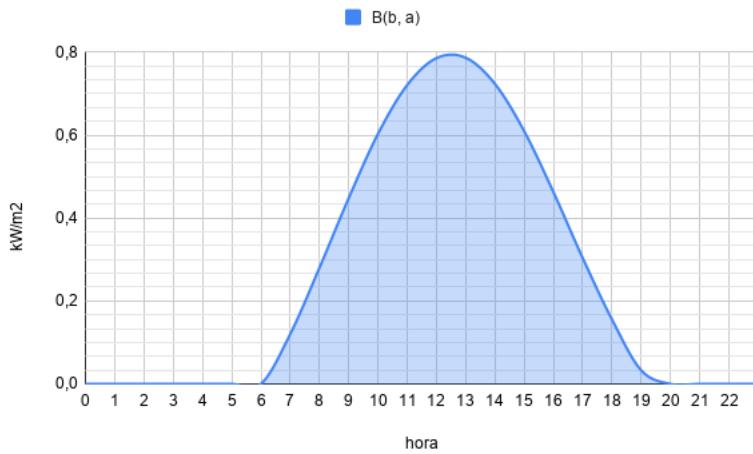


Figura 4.12: Irradiancia directa en el plano inclinado

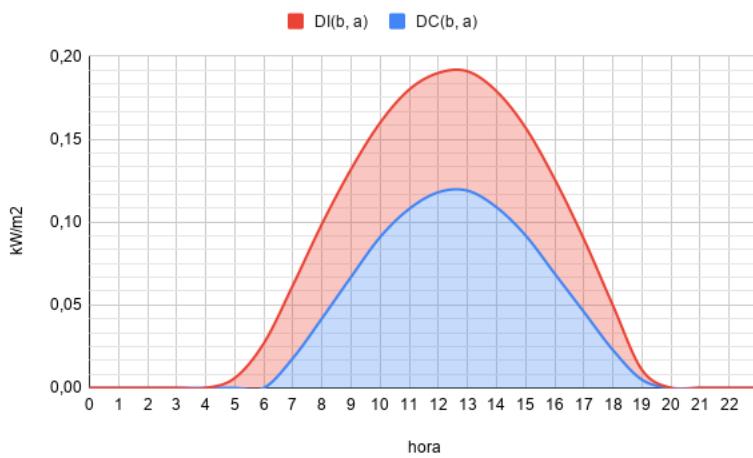


Figura 4.13: Componentes de la irradiancia difusa en el plano inclinado

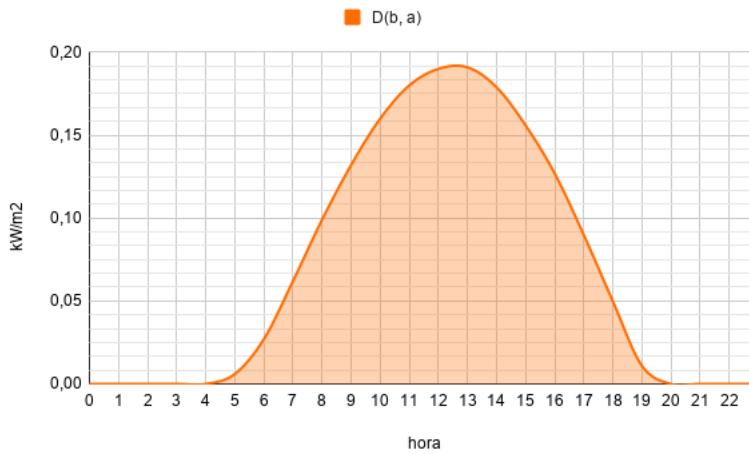


Figura 4.14: Irradiancia difusa en el plano inclinado

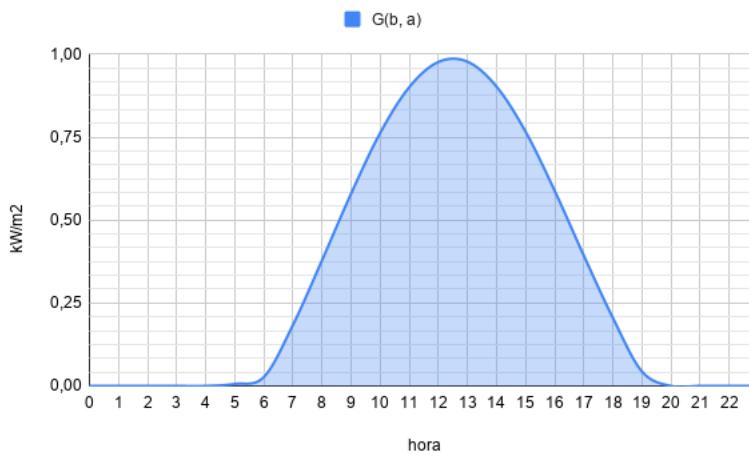


Figura 4.15: Irradiancia global en el plano inclinado

Como se puede observar, a diferencia de la irradiancia en el plano horizontal, estos resultados no son simétricos respecto al mediodía ya que ahora está afectando la orientación de la superficie, que no está orientada totalmente hacia el sur.

4.2.5. Pérdidas por ángulo de incidencia y suciedad

Habiendo calculado los valores de irradiancia horaria en la superficie inclinada, el siguiente paso consiste en aplicar las pérdidas por ángulo de incidencia y suciedad, mediante el proceso descrito en la sección 3.2.3. Los resultados del proceso se muestran en la tabla 4.8.

hora	$B_{ef}(\beta, \alpha)$ (kW/m ²)	$D_{ef}^C(\beta, \alpha)$ (kW/m ²)	$D_{ef}^I(\beta, \alpha)$ (kW/m ²)	$D_{ef}(\beta, \alpha)$ (kW/m ²)	$G_{ef}(\beta, \alpha)$ (kW/m ²)
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0,006	0,006	0,006
6	0	0	0,025	0,025	0,025
7	0,076	0,011	0,041	0,052	0,128
8	0,243	0,037	0,052	0,089	0,3314
9	0,422	0,064	0,059	0,123	0,545
10	0,583	0,088	0,064	0,152	0,735
11	0,704	0,106	0,066	0,172	0,8767
12	0,770	0,116	0,066	0,182	0,952
13	0,771	0,116	0,066	0,182	0,953
14	0,708	0,106	0,064	0,170	0,879
15	0,593	0,089	0,059	0,149	0,742
16	0,443	0,067	0,052	0,119	0,562
17	0,277	0,042	0,041	0,082	0,349
18	0,119	0,018	0,025	0,043	0,162
19	0,010	0,001	0,006	0,007	0,017
20	0	0	0	0	0
21	0	0	0	0	0
22	0	0	0	0	0
23	0	0	0	0	0

Cuadro 4.8: Irradiancia directa, difusa y global efectiva incidente en el plano inclinado, para el día promedio del mes de julio

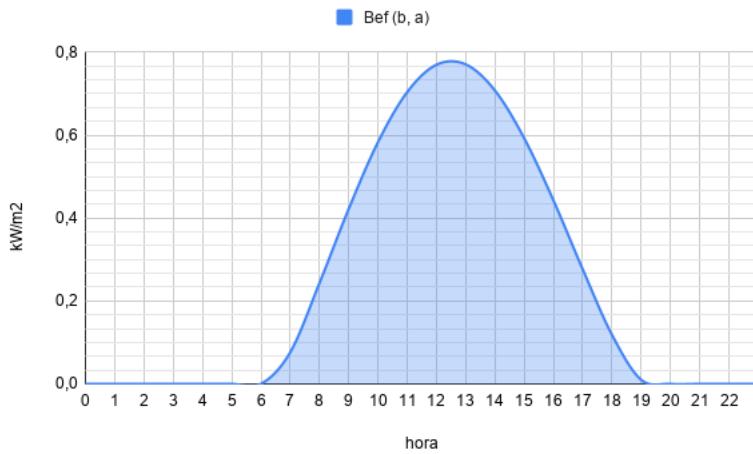


Figura 4.16: Irradiancia directa efectiva en el plano inclinado

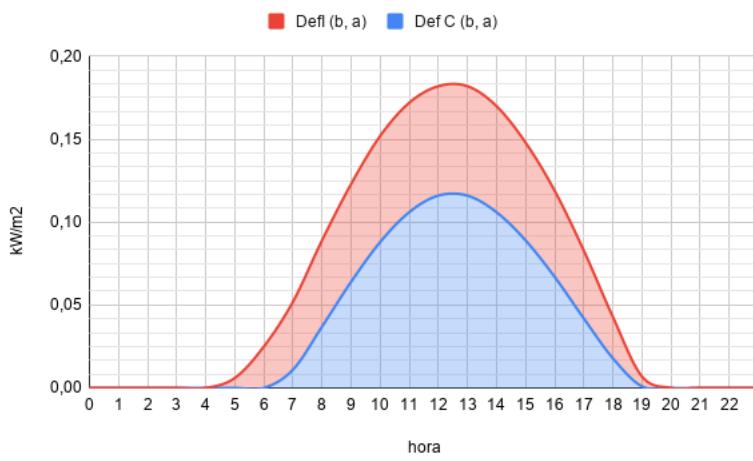


Figura 4.17: Componentes de la irradiancia difusa efectiva en el plano inclinado

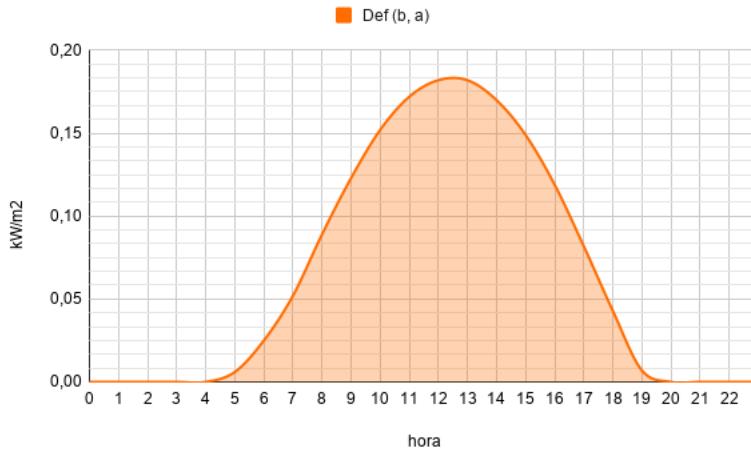


Figura 4.18: Irradiancia difusa efectiva en el plano inclinado

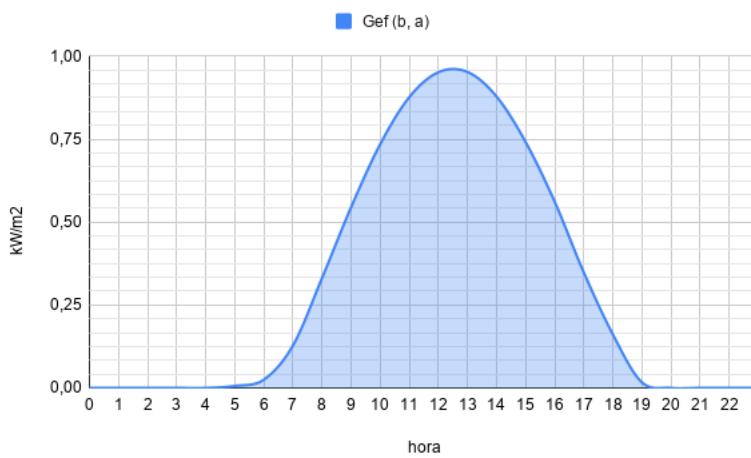


Figura 4.19: Irradiancia global efectiva en el plano inclinado

Como se puede apreciar, los valores eficaces son ligeramente inferiores al aplicar las pérdidas por el nivel de suciedad medio y el ángulo de incidencia.

4.2.6. Configuración del módulo solar y su comportamiento

La primera parte del proceso de cálculo está completada, teniendo ya los valores eficaces de irradiancia horaria para cada uno de los 12 días promedio.

La segunda parte consiste en aplicar dicha radiación a un módulo para estimar la potencia máxima que será capaz de entregar, así como la energía que producirá al cabo de un año.

Para ello, tal y como se describe en el desarrollo teórico, debemos calcular la potencia en el punto de máxima potencia con las condiciones de temperatura y radiación del emplazamiento. Como

ya tenemos los datos de radiación, a continuación calcularemos el perfil horario de temperatura como se describe en la sección de la página 26.

El resultado de ese proceso, para las coordenadas indicadas por el usuario se muestran en las tablas 4.9 y 4.10.

$^{\circ}\text{C}$	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sept	Oct	Nov	Dic
T_{\min}	0,7	2,6	5,8	8,9	11,8	17,5	22,4	18,6	14,1	10,8	6,8	4,9
T_{\max}	11,4	10,6	17,1	20,3	27	31,1	36,4	32,3	26,4	20,3	17,5	14,2

Cuadro 4.9: Temperaturas máximas y mínimas en $^{\circ}\text{C}$ para cada uno de los 12 meses

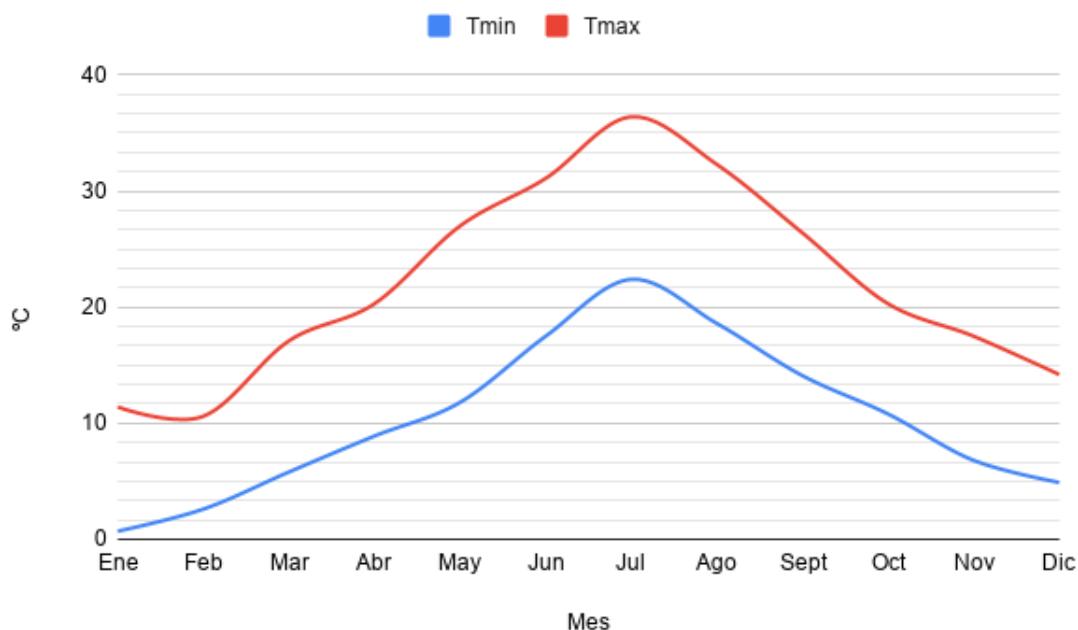


Figura 4.20: Temperaturas máximas y mínimas en $^{\circ}\text{C}$ para cada uno de los 12 meses

hora	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sept	Oct	Nov	Dic
0	1,05	3,14	7,19	11,04	15,46	21,13	25,96	21,43	15,86	11,57	7,21	5,11
1	1,16	3,25	7,36	11,23	15,73	21,37	26,21	21,66	16,06	11,70	7,32	5,19
2	1,56	3,60	7,93	11,84	16,54	22,09	26,95	22,39	16,69	12,14	7,73	5,51
3	2,41	4,29	8,96	12,88	17,91	23,30	28,20	23,64	17,81	12,97	8,59	6,21
4	3,83	5,37	10,46	14,34	19,76	24,91	29,89	25,37	19,43	14,26	10,03	7,44
5	5,77	6,77	12,31	16,05	21,89	26,75	31,81	27,37	21,39	15,88	11,95	9,14
6	7,88	8,23	14,18	17,74	23,94	28,50	33,65	29,34	23,36	17,58	14,04	11,02
7	9,70	9,46	15,71	19,09	25,56	29,88	35,11	30,91	24,96	19,01	15,83	12,66
8	10,88	10,25	16,68	19,93	26,56	30,73	36,01	31,88	25,96	19,90	16,99	13,73
9	11,36	10,57	17,06	20,27	26,96	31,07	36,37	32,26	26,36	20,27	17,46	14,16
10	11,39	10,59	17,08	20,28	26,98	31,08	36,38	32,28	26,38	20,29	17,49	14,19
11	11,35	10,56	17,04	20,23	26,90	31,01	36,31	32,22	26,33	20,25	17,45	14,15
12	11,33	10,54	17,01	20,20	26,86	30,97	36,27	32,18	26,30	20,23	17,43	14,14
13	11,35	10,56	17,04	20,23	26,90	31,01	36,31	32,22	26,33	20,25	17,45	14,15
14	11,39	10,59	17,08	20,28	26,98	31,08	36,38	32,28	26,38	20,29	17,49	14,19
15	11,36	10,57	17,06	20,27	26,96	31,07	36,37	32,26	26,36	20,27	17,46	14,16
16	10,88	10,25	16,68	19,93	26,56	30,73	36,01	31,88	25,96	19,90	16,99	13,73
17	9,70	9,46	15,71	19,09	25,56	29,88	35,11	30,91	24,96	19,01	15,83	12,66
18	7,88	8,23	14,18	17,74	23,94	28,50	33,65	29,34	23,36	17,58	14,04	11,02
19	5,77	6,77	12,31	16,05	21,89	26,75	31,81	27,37	21,39	15,88	11,95	9,14
20	3,83	5,37	10,46	14,34	19,76	24,91	29,89	25,37	19,43	14,26	10,03	7,44
21	2,41	4,29	8,96	12,88	17,91	23,30	28,20	23,64	17,81	12,97	8,59	6,21
22	1,56	3,60	7,93	11,84	16,54	22,09	26,95	22,39	16,69	12,14	7,73	5,51
23	1,16	3,25	7,36	11,23	15,73	21,37	26,21	21,66	16,06	11,70	7,32	5,19

Cuadro 4.10: Perfil de temperaturas en °C en las coordenadas del emplazamiento según el método descrito en [10]

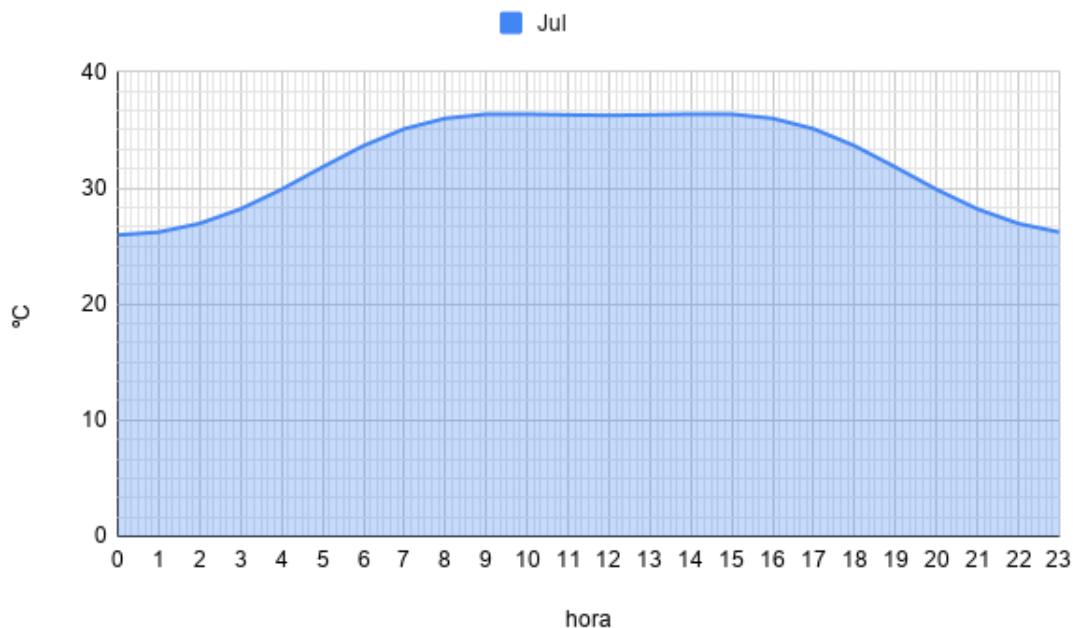


Figura 4.21: Perfil horario de temperaturas para el día promedio del mes de Julio

La temperatura ambiente se necesita para calcular la temperatura de la célula según la ecuación 3.32, que a su vez se utiliza para calcular la tensión de circuito abierto, como se puede ver en la ecuación 3.43.

A continuación, en la tabla 4.11 se muestran los resultados del proceso de cálculo descrito a partir de la página 31 para hallar la tensión de circuito abierto V_{oc} , la intensidad de cortocircuito I_{sc} , así como la tensión, intensidad y potencia en el punto de máxima potencia, todo ello en las condiciones de temperatura y radiación del emplazamiento indicado por el usuario

hora	$V_{oc}(V)$	$I_{sc}(A)$	$V_{mpp}(V)$	$I_{mpp}(A)$	$P_{mpp}(W)$
0	46,601	0	44,793	0	0
1	46,595	0	44,786	0	0
2	46,578	0	44,767	0	0
3	46,550	0	44,734	0	0
4	46,512	0	44,689	0	0
5	46,463	0,052	44,586	0,051	2,287
6	46,405	0,222	44,365	0,220	9,782
7	46,286	1,126	43,417	1,116	48,451
8	46,093	2,935	41,575	2,906	120,800
9	45,902	4,860	39,635	4,806	190,477
10	45,739	6,567	37,925	6,487	246,037
11	45,619	7,842	36,652	7,742	283,740
12	45,554	8,525	35,970	8,413	302,612
13	45,553	8,531	35,963	8,419	302,759
14	45,615	7,868	36,624	7,767	284,452
15	45,732	6,636	37,856	6,556	248,184
16	45,895	5,009	39,495	4,953	195,606
17	46,089	3,192	41,341	3,159	130,596
18	46,290	1,429	43,151	1,416	61,081
19	46,454	0,153	44,484	0,151	6,732
20	46,512	0	44,689	0	0
21	46,550	0	44,734	0	0
22	46,578	0	44,767	0	0
23	46,595	0	44,786	0	0

Cuadro 4.11: Valores horarios de V_{oc} , I_{sc} , I_{mpp} , V_{mpp} y P_{mpp} para el día promedio del mes de julio

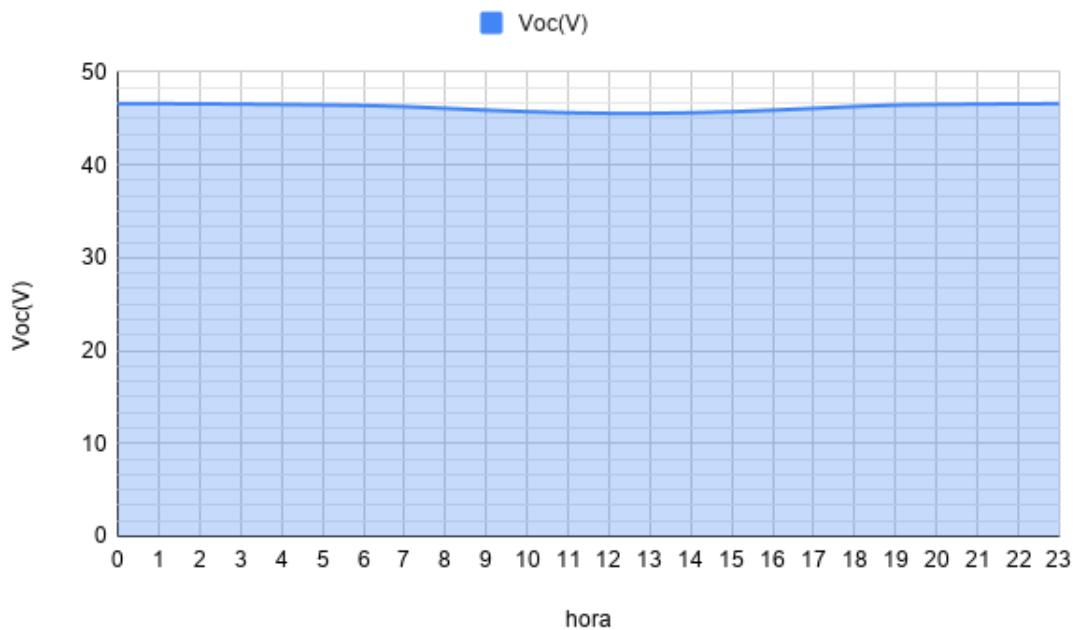


Figura 4.22: Valores horarios de tensión de circuito abierto para el día promedio del mes de Julio

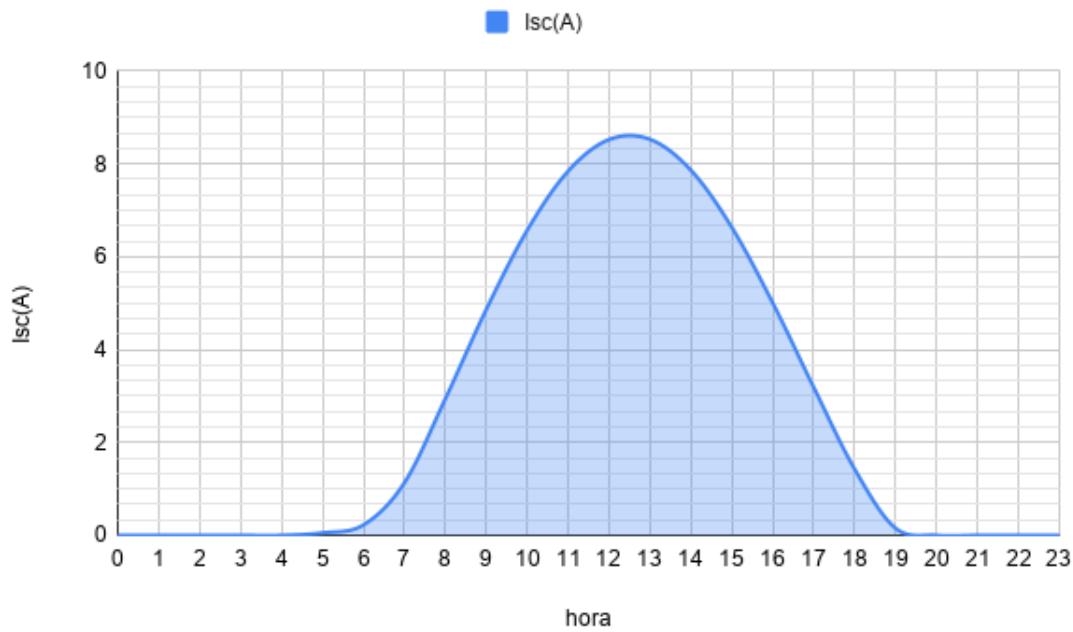


Figura 4.23: Valores horarios de corriente de cortocircuito para el día promedio del mes de Julio

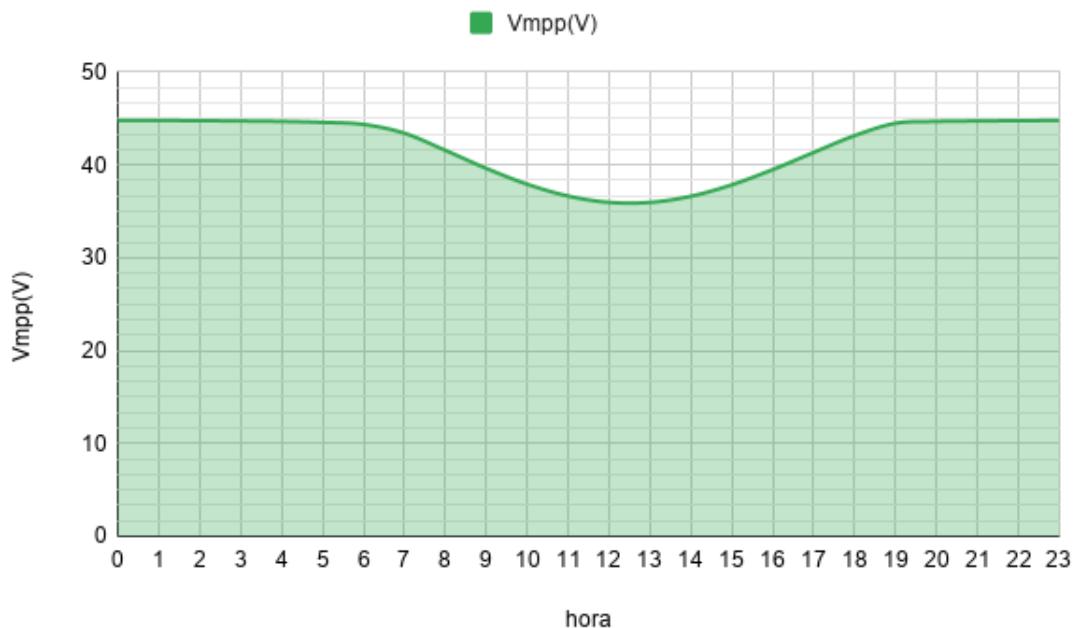


Figura 4.24: Valores horarios de tensión en el punto de máxima potencia para el día promedio del mes de Julio

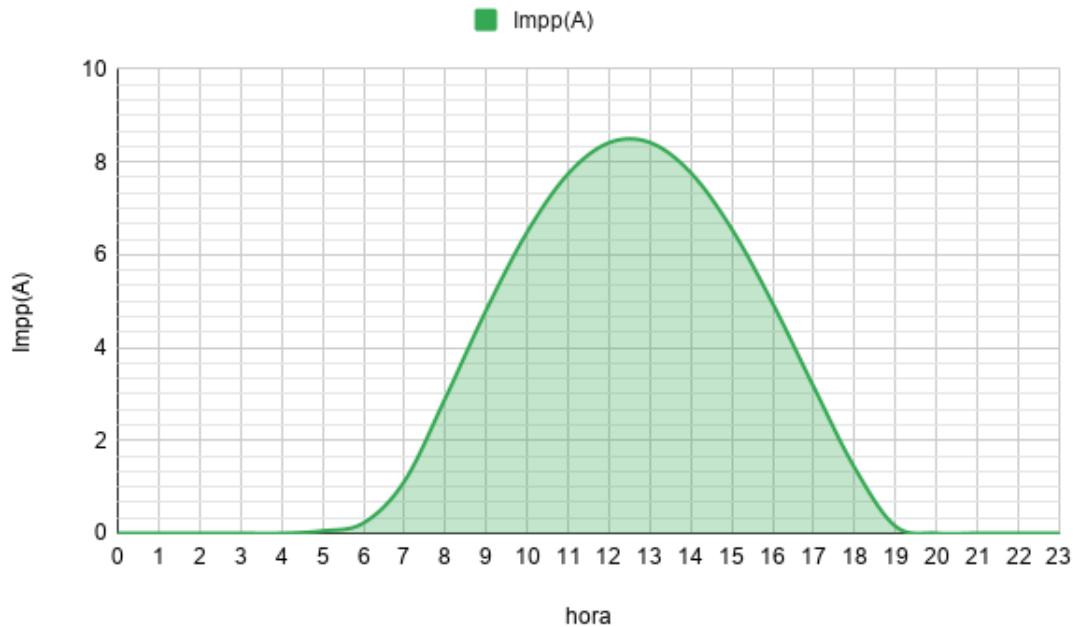


Figura 4.25: Valores horarios de corriente en el punto de máxima potencia para el día promedio del mes de Julio

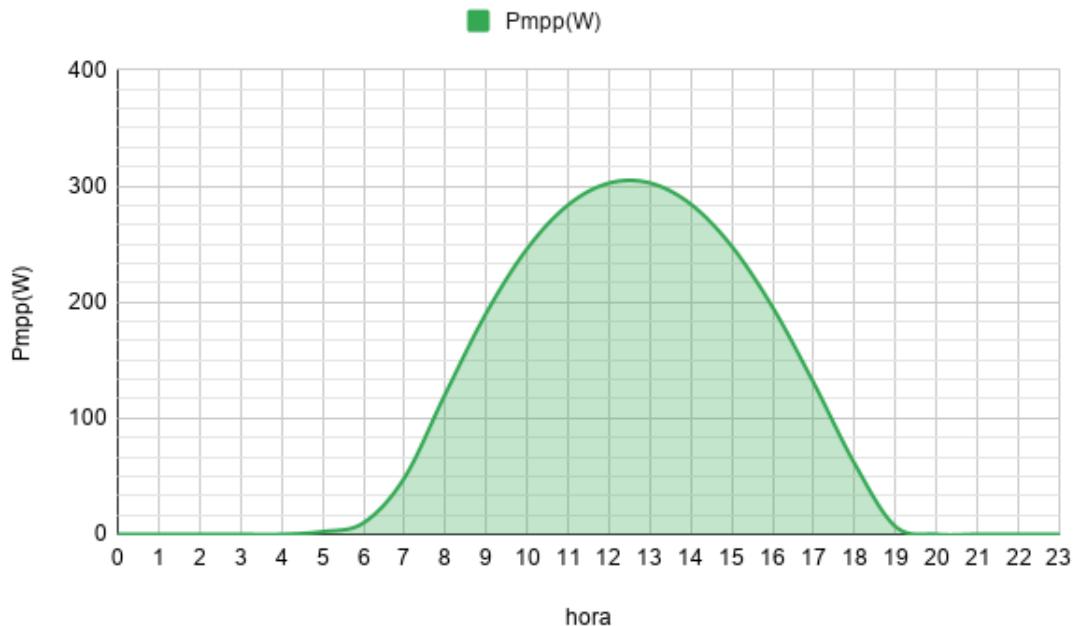


Figura 4.26: Valores horarios de potencia en el punto de máxima potencia para el día promedio del mes de Julio

Una vez obtenidos los valores en el punto de máximo potencia, el siguiente paso consiste en obtener las potencias en continua y en alterna, es decir, antes y después del inversor. Sobretodo, lo que más interesa es la potencia a la salida del inversor, pues es la que se utilizará para calcular la energía que será capaz de generar un módulo. El proceso de éste cálculo se describe en las ecuaciones 3.55 a 3.61

El resultado para el mes de julio en el emplazamiento del usuario se muestra en la tabla 4.12.

hora	$P_{AC}(W)$
0	0.000
1	0.000
2	0.000
3	0.000
4	0.000
5	0.000
6	6,531
7	43,754
8	112,216
9	176,765
10	227,312
11	261,161
12	277,969
13	278,100
14	261,796
15	229,249
16	181,469
17	121,378
18	55,820
19	3,576
20	0.000
21	0.000
22	0.000
23	0.000

Cuadro 4.12: Valores horarios de la potencia en alterna P_{AC} para el día promedio del mes de julio

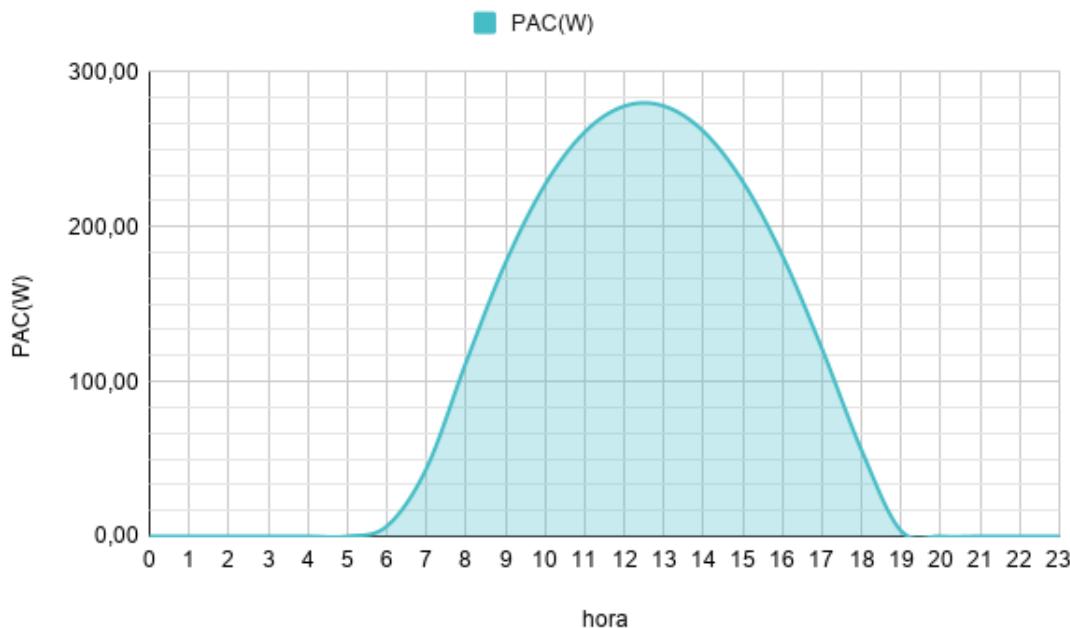


Figura 4.27: Valores horarios de potencia en corriente alterna para el día promedio del mes de Julio

A continuación, utilizaremos los valores de la potencia a la salida del inversor P_{AC} para calcular la energía que será capaz de producir el módulo en cada mes. Estas energías se muestran en la tabla 4.13.

	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic
$E_{mes}(kWh)$	26,19	37,88	51,54	55,33	61,02	68,16	67,11	62,78	52,67	41,33	28,47	22,85

Cuadro 4.13: Energía generada por un módulo en un mes

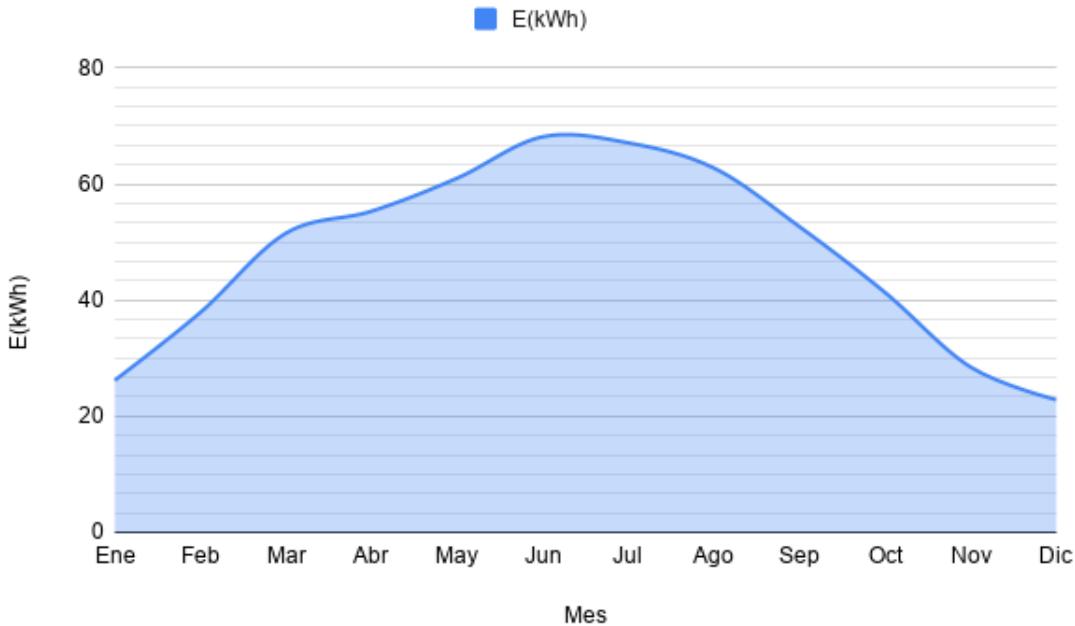


Figura 4.28: Energía generada por un módulo en un mes

4.3. Cálculo final de las potencias y energías

Habiendo hecho un breve repaso por el proceso de cálculo en un emplazamiento concreto, lo último y más importante que debemos estudiar son los resultados finales que se han obtenido, es decir, las potencias máximas mensuales, la productividad y sobretodo la energía, tanto mensual como anual, que es capaz de producir el generador configurado por el usuario.

Hasta el momento, con el fin de simplificar los cálculos, éstos se llevaron a cabo con un solo módulo. A continuación vamos a trasladar todos estos valores a la superficie que el usuario va a destinar para la instalación del generador.

Para ello, debemos obtener una relación entre el área disponible que el usuario ha introducido y el área total del generador configurado (12 módulo en serie y 11 en paralelo). Para ello, utilizamos el número total de módulos del generador base multiplicado por el área del un módulo, que se indica como uno de los valores disponibles en la tabla 3.3.

En esta caso, la relación resultante es:

$$\begin{aligned}
 Area_{generador} &= 12 \cdot 11 \cdot 1,941 m^2 = 256,257 m^2 \\
 Area_{superficie} &= 40 m^2 \\
 R &= \frac{Area_{superficie}}{Area_{generador}} = 0,156
 \end{aligned} \tag{4.1}$$

Por otro lado, podemos obtener el numero de módulos totales que se pueden instalar en el área indicada por el usuario simplemente dividiendo el área indicada por el área de un módulo y redondeando hacia abajo al primero numero entero. De tal manera que:

$$\begin{aligned}
 Area_{modulo} &= 1,941 m^2 \\
 Area_{superficie} &= 40 m^2 \\
 N_{modulos} &= \frac{40}{1,941} = 20,60 \simeq 20
 \end{aligned} \tag{4.2}$$

Por tanto, al conocer el numero de módulos, también podemos conocer la potencia nominal instalada, así como el resto de valores como la energía mensual, la potencia máxima que es capaz de entregar el generador o la productividad.

A continuación se muestran unos gráficos extraídos directamente de la página de resultados:

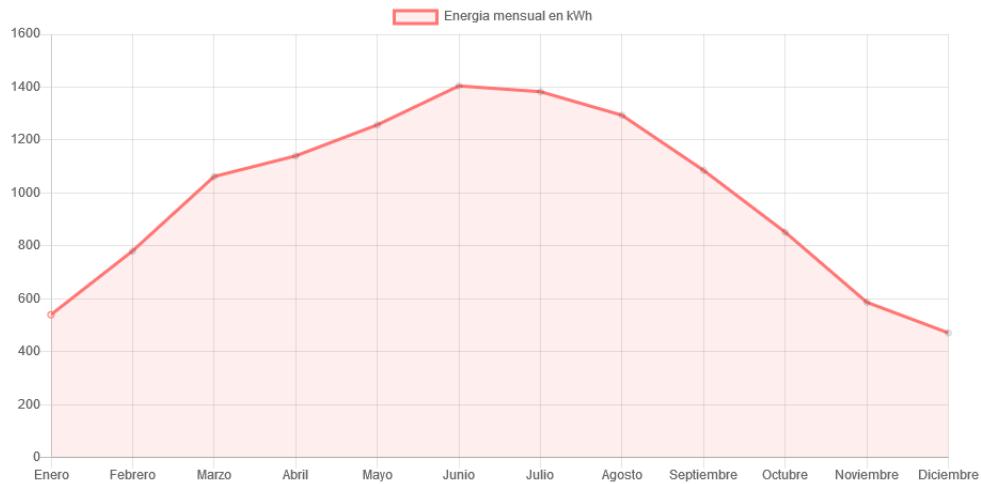


Figura 4.29: Energía mensual producida

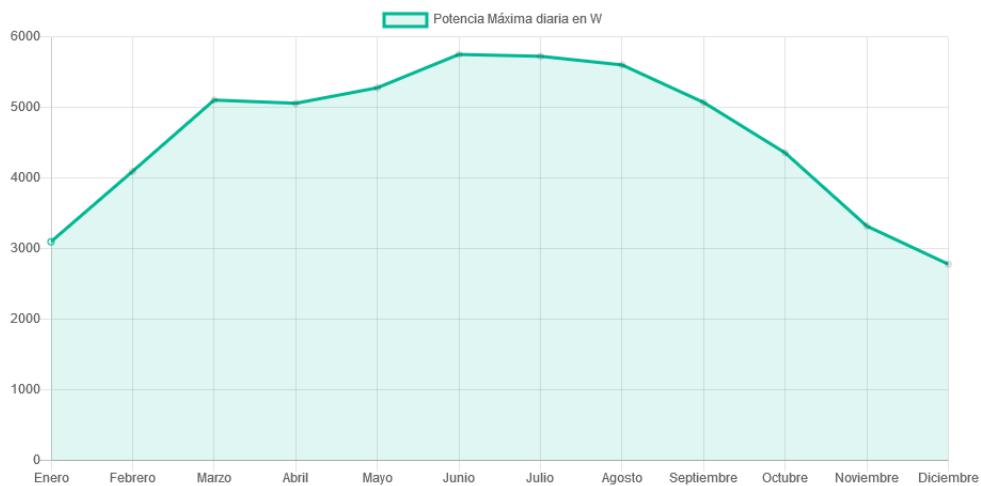


Figura 4.30: Potencia máxima mensual entregada

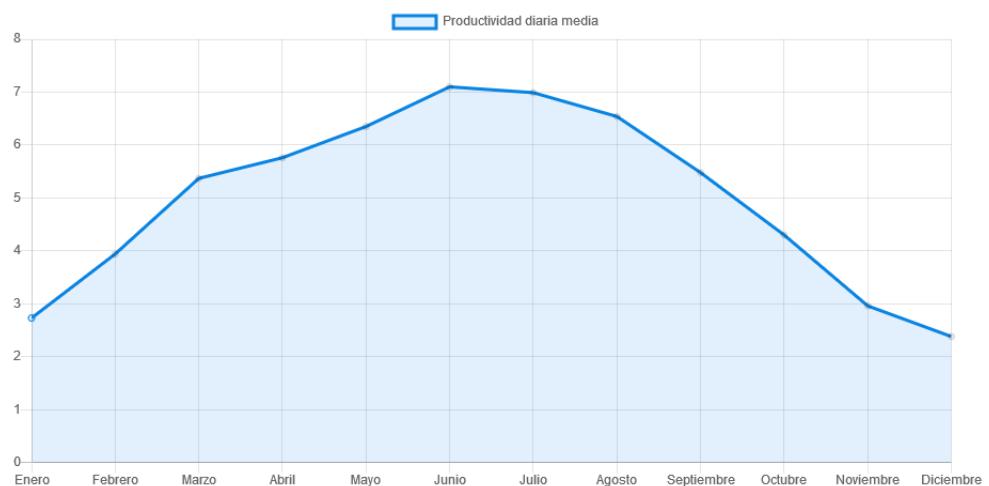


Figura 4.31: Productividad media mensual

Capítulo 5

Detalles de la programación

En este capítulo se va a llevar a cabo una explicación mas extensa de algunos de los apartados de la programación que hubieran resultado muy densos o complejos en la explicación del código. Además se hará una descripción más profunda de las tecnologías usadas en la aplicación. En este anexo solo se utilizarán los extractos de código que acompañen a una explicación o aclaración. Tanto el código completo de la aplicación como éste documento están almacenados en un repositorio online de la página de Github para un acceso rápido y cómodo a través del siguiente enlace: <https://github.com/IonutMorariu/PV-Calculator>

5.1. Descripción de tecnologías

En el desarrollo de cualquier aplicación web intervienen dos partes muy diferencias, cada una con unas funciones específicas.

Por un lado es necesario desarrollar un servidor, que en la industria se conoce como Backend, encargado de gestionar la base de datos donde se almacenan los datos, y proveer la información necesaria en todo momento a través de peticiones HTTP¹.

Por otro lado, para que el usuario pueda interactuar con la página, es necesario desarrollar una interfaz para la web, esto se conoce como Frontend o cliente y es lo que se encarga de recoger la información del usuario a través de un formulario que es posteriormente enviado al servidor. Éste realiza los cálculos necesarios, en este caso todo lo relacionado con la radiación solar, y los devuelve al cliente para mostrar los resultados al usuario.

5.1.1. Servidor

En el caso del servidor, existe una gran variedad de lenguajes de programación que se pueden utilizar, cada uno con sus ventajas y desventajas.

Los lenguajes más utilizados para la programación de servidores web son:

- **Java:** Es quizás el lenguaje de programación de servidores más extendido y usado en la industria

¹HTTP: Protocolo de comunicación para transmitir información a través de la web

- **Python:** Un lenguaje de programación que destaca por su semejanza con el lenguaje natural. Ha adquirido bastante fama en los últimos años en temas relacionados con el aprendizaje máquina
- **PHP:** Uno de los lenguajes más extendidos, en parte por ser el que se ha utilizado para programar Wordpress, el gestor de contenido más utilizado.
- **Javascript:** Un lenguaje creado para programar páginas web, pero que gracias a un entorno de ejecución desarrollado por Google, se puede utilizar para crear servidores. La popularidad de este lenguaje viene de la flexibilidad que ofrece, dado que un solo programador puede realizar las dos tareas, servidor y cliente, si conoce este lenguaje. Comúnmente es confundido con Java, pero son totalmente diferentes.

El lenguaje escogido en este caso es Javascript, por experiencias anteriores y por la flexibilidad de utilizar un solo lenguaje en ambos casos. Es un lenguaje de programación interpretado, es decir, no es necesario compilarlo con cada cambio, sino que se ejecuta directamente desde el archivo de código, agilizando el desarrollo a riesgos de cometer más errores en vivo.

El entorno de Javascript que nos permite crear el servidor es NodeJS. Según la descripción de la página web [11], NodeJS fue ideado como un entorno de ejecución de Javascript orientado a eventos asíncronos, diseñado para crear aplicaciones de red escalables.

Para la base de datos también existen multitud de opciones con diferentes ventajas y desventajas. Algunas de las opciones más utilizadas son:

- **MySQL:** La base de datos relacional más extendida, aunque por su edad y arquitectura de diseño, carece de muchas funcionalidades necesarias en el estado actual de la web.
- **PostgreSQL:** Una base de datos relacional más moderna, y con mas funcionalidad que MySQL, que está en proceso de sustituir.
- **MongoDB:** A diferencia de las SQL, MongoDB es una base de datos no relacional, que guarda la información en documentos individuales en lugar de tablas.

La similitud entre la sintaxis de las peticiones y los resultados de MongoDB con la de Javascript ha hecho que sea la opción elegida para este proyecto.

```

1   db.inventory.find( {
2     status: "A",
3     $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
4   )

```

Extracto de código 5.1: Ejemplo de una petición en MongoDB

Se puede observar que tiene muchas similitudes con el lenguaje de programación utilizado, a diferencia de SQL que tiene una sintaxis mas particular:

```

1   SELECT * FROM inventory WHERE status = "A" AND ( qty < 30 OR item LIKE "p%")

```

Extracto de código 5.2: Ejemplo de una petición en SQL

5.1.2. Cliente

Al el contrario que en el servidor, en el cliente no existen opciones a la hora de programar la interfaz. Solamente se pueden usar 3 lenguajes y cada uno de ellos tiene una función específica.

HTML

HTML es el lenguaje que se utiliza para establecer la estructura de la página y dotarla de contenido. Es decir, es lo que se utiliza para definir que campos tendrá el formulario, el título de la página y cualquier otro tipo de contenido.

La sintaxis de este lenguaje se conoce como lenguaje de marcado, de ahí sus siglas (Hyper Text Markup Language) que se basa en una etiquetas concretas que define sus estructura.

```

1  <!DOCTYPE html>
2
3  <html>
4      <head>
5          <title>Page Title</title>
6      </head>
7      <body>
8          <h1>Esto es un titulo</h1>
9          <p>This is a parágrafo.</p>
10         </body>
11     </html>
```

Extracto de código 5.3: Ejemplo de una petición en SQL

CSS

CSS (Cascading Style Sheet) es el lenguaje que se utiliza en una web para proveer de estilo al contenido definido con HTML. Utiliza un sistema de propiedades y valores que heredan de más general a más específico.

Al tratarse de simplemente listas de propiedades que se aplican a las diferentes etiquetas de HTML, es bastante común utilizar códigos ya creados para ahorrar tiempo. Estas listas son también conocidas como librerías de componentes. Las más conocidas son Bootstrap², Bulma³ y Zurb Foundation⁴

```

1  body {
2      background-color: lightblue;
3  }
```

²Bootstrap: <https://getbootstrap.com/>

³Bulma: <https://bulma.io/>

⁴Zurb Foundation: <https://get.foundation/>

```
5   h1 {  
6       color: white;  
7       text-align: center;  
8   }  
9  
10  p {  
11      font-family: verdana;  
12      font-size: 20px;  
13  }
```

Extracto de código 5.4: Ejemplo de código CSS

La forma en la que se escribe dicho código es aplicando ciertas reglas u normas de estilo predefinidas a las diferentes etiquetas de HTML. Así, por ejemplo, según el extracto 14 todo el documento tendrá de color de fondo el azul, o el título de la página será verde y alineado al centro.

Javascript

Como ya se ha mencionado anteriormente, Javascript es el lenguaje de programación utilizado tanto para el servidor como para el cliente.

En el caso del cliente no existen otras opciones dado que Javascript es el único lenguaje que los navegadores web utilizados en la actualidad son capaces de interpretar.

A diferencia de HTML y CSS, Javascript es el encargado de otorgarle la lógica y la funcionalidad a la página web, pues es el que realiza la recogida de datos y su posterior envío al servidor para que se lleven a cabo parte de los cálculos.

En el cliente se realizan los cálculos relacionados con la potencia y la energía entregadas, pues no requiere de acceso a la base de datos y por tanto es posible delegar parte de la carga de trabajo que soporta el servidor.

5.2. Explicaciones de código

5.2.1. Emplazamiento del usuario

Uno de los datos más importantes para poder estimar la energía que finalmente podrá producir el generador es el de la radiación solar en el lugar de instalación. Para ello, es necesario conocer las coordenadas geográficas del emplazamiento. Sin embargo, es poco intuitivo pedirle a un usuario que introduzca sus coordenadas, dado que la mayoría desconocen dichos datos.

Por lo tanto, la ruta que se ha tomado es la de pedirle al usuario su dirección, o una dirección cercana a su localización, y utilizar la API de Google Maps⁵ para convertir dicha dirección en las coordenadas de latitud y longitud que se necesitan para poder obtener la radiación en dicho lugar.

⁵API Google Maps: Enlace interactivo al que se le pueden enviar los datos de una dirección y devuelve las coordenadas de latitud y longitud de un emplazamiento.

Este proceso comienza por recoger los datos de la dirección, municipio y código postal a través del formulario que aparece en la página web.

Estos datos son recogidos en el código a través de un nombre único que han recibido:

```

1 const addressInput = document.querySelector('#address');
2 const cityInput = document.querySelector('#city');
3 const postalInput = document.querySelector('#postal');
```

Extracto de código 5.5: Variables correspondientes a los tres campos

Una vez que tenemos estos datos recogidos en las variables, podemos pedir a la API de Google Maps las coordenadas de latitud y longitud de dicho emplazamiento encadenando las tres variables y una clave única de identificación, para obtener un enlace único que se corresponde a dicha localización.

```

1 const getCoordinates = async () => {
2   const address = addressInput.value.split(' ').join('+');
3   const city = cityInput.value;
4   const postal = postalInput.value;
5   const requestURL = `${googleEndpoint}address=${address},${city},${postal}
6                           ,spain&key=${googleApiKey}`;
7   const response = await fetch(requestURL);
8   const data = await response.json();
9   const info = {
10     formattedAddress: data.results[0].formatted_address,
11     lat: data.results[0].geometry.location.lat,
12     long: data.results[0].geometry.location.lng
13   };
14
15   return info;
16};
```

Extracto de código 5.6: Función encargada de solicitar los datos a la API

La función de **getCoordinates** (5.6) recoge el valor de la dirección y reemplaza los espacios con el signo + (formato requerido por la API) y lo concatena con el valor del campo de la ciudad y el código postal. Al final le añade una clave única que identifica la aplicación a la hora de establecer límites de uso y evitar abuso de la API.

Una vez creado este enlace único, el código lanza la petición al servicio y retorna con la información que es recogida y se guarda en dos variables **lat** y **long** para ser utilizadas posteriormente, a la hora de obtener los datos de irradiación global.

5.2.2. Área, inclinación, orientación y nivel de suciedad de la superficie de instalación

Además de las información de latitud y longitud del emplazamiento, el cálculo de la instalación también requiere de información relacionada con el área, la inclinación, la orientación y el nivel de suciedad de la superficie donde se va a realizar la instalación, para poder realizar una estimación lo mas exacta posible.

Estos valores son recogidos directamente de los campos de la pagina web, al igual que los campos anteriores, sin necesitar ningún trato especial:

```

1 const slope = document.querySelector('#slope');
2 const area = document.querySelector('#area');
3 const orientation = document.querySelector('#orientation');
4 const dirtLevel = document.querySelector('#dirt-level');
```

Extracto de código 5.7: Variables correspondientes a los campos indicados

5.2.3. Obtención de la información de radiación global en el plano horizontal

En la sección teórica se menciona brevemente como se obtuvo la radiación global en el plano horizontal de manera dinámica en función de la latitud y longitud del emplazamiento.

En ésta sección se va a llevar a cabo un desarrollo más extenso del proceso que se llevó a cabo para poder tener acceso a la información.

El proceso comienza con la búsqueda de una fuente de la radiación global. Investigando las diferentes posibilidades, surge la página de ADRASE, un proyecto fundado por el Gruo de Radiación Solar del CIEMAT. En la página web⁶ se puede encontrar una descripción detallada acerca del proyecto.

En la página de ADRASE podemos encontrar un mapa interactivo de la península, en la que haciendo clic en cualquier punto, nos aparece un enlace con los datos mensuales de radiación, como se puede observar en la figura 5.1.

Sin embargo, éste mapa no permite tener un acceso dinámico a la información, sin la interacción de un usuario. Además el servicio de ADRASE no ofrece un punto de acceso a los datos, por tanto, la descarga de datos se tiene que hacer previamente y almacenar los datos en la base de datos del servidor.

Cuando se hace clic en el enlace de datos mensuales de radiación global, aparece una ventana emergente con los datos del emplazamiento, como el de la figura 5.2.

La ventana emergente tiene un enlace personalizado de tal manera que pueda generar la información de ese localización concreta. En enlace contiene dos variables que hacen referencia a la latitud y la longitud:

```
1 www.adrase.com/adrasemaps/php/monthly_popup.php?lat=39.12&lon=-3.92&var_tipe=0
```

Extracto de código 5.8: Enlace tipo de la radiación en la peninsula

Por tanto, si generamos un enlace con la latitud y la longitud del emplazamiento, podemos obtener una ventana emergente como la que se muestra en la figura 5.2. Sin embargo, obtener los datos

⁶ADRASE: <http://www.adrase.com/el-proyecto.html>

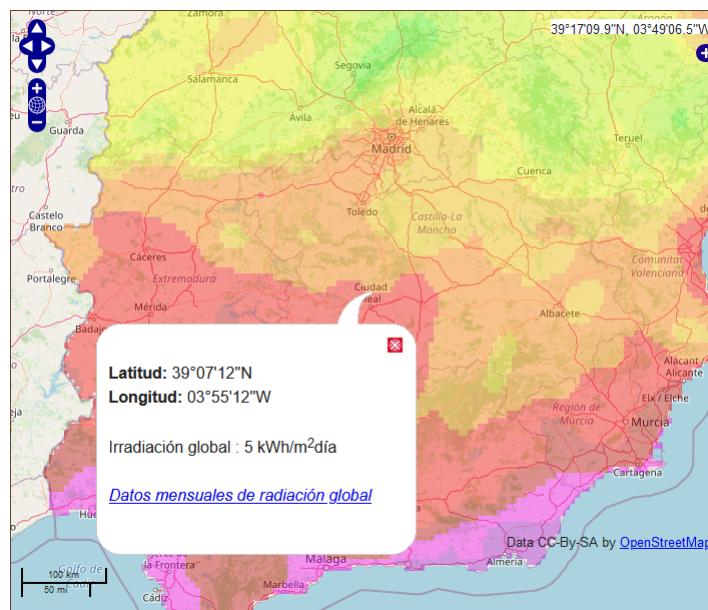


Figura 5.1: Mapa interactivo de radiación

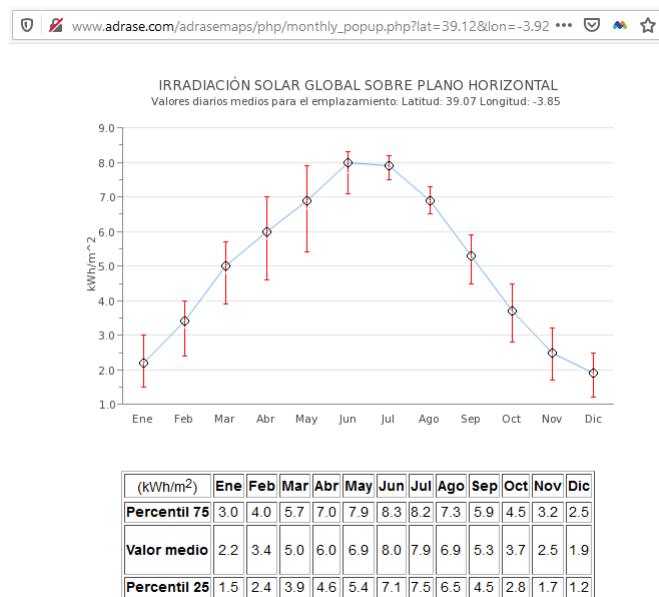


Figura 5.2: Ventana emergente

de radiación no es una tarea inmediata. Es necesario procesar el documento y extraer los valores deseados.

El contenido del documento contiene una tabla de html que contiene los valores medios, percentil 25 y percentil 75 de radiación global en el plano horizontal. La parte que necesitamos extraer y procesar es la siguiente:

```

1 <tr class="customPeriods">
2     <td align="center"><b>Valor medio</b></td>
3     <td align="center"><p>2.2</p></td>
4     <td align="center"><p>3.3</p></td>
5     <td align="center"><p>4.8</p></td>
6     <td align="center"><p>6.0</p></td>
7     <td align="center"><p>7.0</p></td>
8     <td align="center"><p>8.0</p></td>
9     <td align="center"><p>7.8</p></td>
10    <td align="center"><p>6.9</p></td>
11    <td align="center"><p>5.3</p></td>
12    <td align="center"><p>3.7</p></td>
13    <td align="center"><p>2.4</p></td>
14    <td align="center"><p>1.9</p></td>
15 </tr>
```

Extracto de código 5.9: Tabla de valores medios según documento de ADRASE

Por tanto, para procesar el documento, lo que haremos es buscar la frase *Valor medio* y eliminar todo lo anterior. A continuación, buscaremos en el documento el primer *</tr>* que se encuentra, dado que indicará que se ha terminado la fila, ya que la etiqueta *<tr>* de HTML significa table row.

Una vez aisladas las 12 líneas de código que contienen los 12 valores que nos interesa, usaremos una expresión regular (RegEx⁷) para aislar solamente la parte numérica de cada fila.

En código resulta:

```

1 let texts = [];
2 if (e.includes('NO HAY DATOS PARA EL PUNTO SELECCIONADO')) {
3     texts.push(null);
4 } else if (e.includes('Valor medio')) {
5     const regex = /[+-]?\d+(\.\d+)?/g;
6     const str = e.split('Valor medio')[1].split('tr')[0];
7     const floats = str.match(regex).map(function(v) {
8         return parseFloat(v);
```

⁷RegEx: secuencia de caracteres que conforma un patrón de búsqueda

```

9   });
10  texts.push(floats);
11 }

```

Extracto de código 5.10: Proceso de extracción de los valores

En el extracto de código 12 se observa, en la primera linea de código, una condición tal que si el documento contiene la frase *NO HAY DATOS PARA EL PUNTO SELECCIONADO*, se evite el proceso de cálculo, ya que ocasiones, para una coordenada en concreto, sobretodo cercanas a la costa, ADRASE no es capaz de entregarnos los datos de radiación.

Sin embargo, si esa frase no existe, significa que sí existen dichos datos y por tanto podemos extraerlos según el proceso detallado.

La función *split()* es la que se ocupa de cortar el documento por el valor indicado, por tanto, como se observa, la utilizamos para cortar por *Valor medio* y por la etiqueta *tr*, quedando solamente las doce filas de valores medios.

Habiendo creado un script capaz de extraer los doce valores necesarios del documento, podemos replicar este proceso todas las veces que sea necesario.

Por tanto, el siguiente paso es llevar a cabo el proceso para una matriz de latitudes y longitudes, y guardar la información en la base de datos. Para ello, se generará un listado de enlaces al que aplicaremos el algoritmo anterior.

El código que utilizaremos para generar los enlaces es:

```

1  const minLat = 36.1;
2  const maxLat = 43.63;
3  const minLon = -9.15;
4  const maxLon = 3.06;
5  const links = [];
6  const latDiff = 0.25;
7  const lonDiff = 0.25;
8  for (let i = minLat; i <= maxLat; i += 0.25) {
9    for (let j = minLon; j <= maxLon; j += 0.25) {
10      const lon = j.toFixed(2);
11      const lat = i.toFixed(2);
12      links.push(
13        `http://www.adrase.com/adrasmaps/php/monthly_popup.php?
14        lat=${lat}&lon=${lon}&var_tipe=0`
15      );
16    }
17  }

```

Extracto de código 5.11: Proceso de generación de enlaces

Las variables que influyen en el bucle de generación de enlaces son:

- **minLat:** Latitud a partir de la cual se comienza a generar los enlaces.
- **maxLat:** Valor máximo de latitud para parar el bucle.
- **minLat:** Longitud a partir de la cual se comienza a generar los enlaces.
- **maxLat:** Valor máximo de longitud para parar el bucle.
- **links:** Un vector o lista para almacenar los enlaces.
- **latDiff:** Diferencia de latitud en grados entre un punto y el siguiente.
- **lonDiff:** Diferencia de longitud en grados entre un punto y el siguiente.

Con estos valores, lo que conseguimos es configurar una cuadricula con un tamaño concreto, de la que generamos los enlaces. En el caso de la configuración expuesta en el ejemplo, el numero de enlaces que se genera es de 860. A cada uno de los elementos de la lista le aplicaremos el algoritmo de extracción de datos, y obtendremos un documento JSON⁸ con el siguiente aspecto:

```

1  {
2      "lat":"36.35",
3      "lon": "-6.15",
4      "midValues": [
5          2.7,
6          3.8,
7          5.2,
8          6.2,
9          7.3,
10         7.7,
11         7.8,
12         7,
13         5.8,
14         4.4,
15         3.2,
16         2.5
17     ]
18 },

```

Extracto de código 5.12: Ejemplo de un documento JSON con la información de un punto concreto

⁸JSON: Javascript Object Notation es una forma de codificar información, similar a un CSV, pero diseñado específicamente para trabajar de manera nativa con Javascript

Cada uno de los documentos generados es mandado al servidor y guardado en la base de datos. Posteriormente, utilizando una función nativa de MongoDB podremos encontrar el punto más cercano a las coordenadas que envía el usuario, y por tanto obtener una aproximación de la radiación en su emplazamiento.

5.3. Instrucciones

5.3.1. Obtener la inclinación y orientación de la superficie

Medición de la inclinación

En la aplicación se presentan dos campos dentro del formulario que debe llenar el usuario, que requieren introducir la inclinación y la orientación de la superficie donde se va a instalar el generador fotovoltaico.

Para ello, existen multitud de aplicaciones gratuitas en la tienda de aplicaciones tanto de Android como de iOS. En este caso, se va a mostrar un ejemplo con un dispositivo Android, pero esto se puede trasladar igualmente a un dispositivo iOS.

La aplicación que se va a utilizar se llama *Angle Meter*, y la empresa que lo desarrolla se llama *Smart Tool Factory*. Se encuentra en la Play Store de Android de manera gratuita.

Cuando se abre la aplicación por primera vez, encontramos lo siguiente:

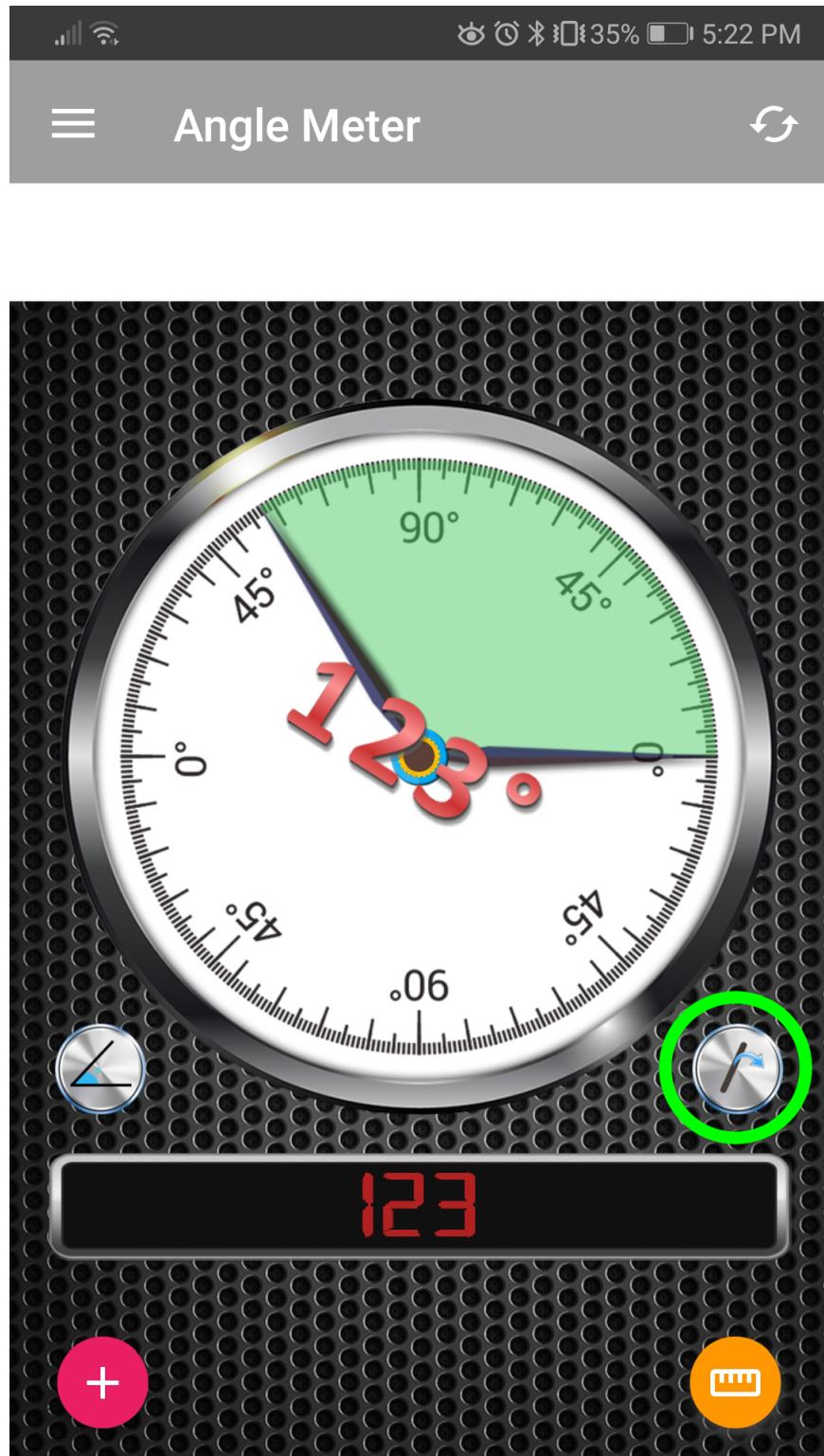


Figura 5.3: Pantalla de inicio de la aplicación

A continuación, pulsamos el botón que sale remarcado con un círculo verde, lo que hará que cambie el modo de medición de ángulos a inclinación, que es el necesario para obtener el dato de

la inclinación. Cuando la aplicación cambie a ese modo, se verá de la siguiente manera:

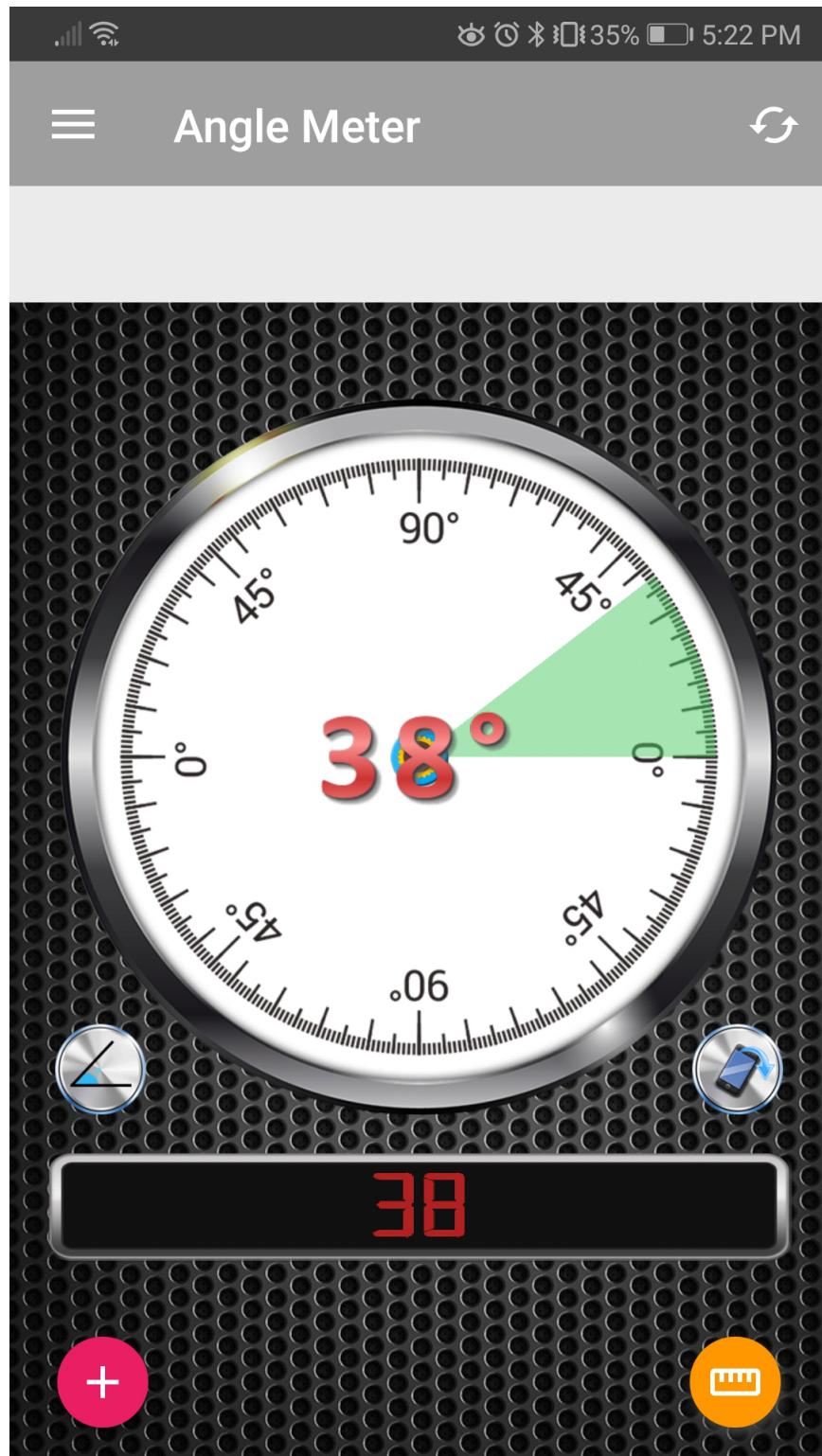


Figura 5.4: Medición de ángulo

A continuación, el dispositivo se debe colocar según la figura que se muestra abajo, teniendo en cuenta que la imagen se muestra con una perspectiva perpendicular a la superficie.



Figura 5.5: Posición del dispositivo

De esta manera, obtenemos la inclinación de la superficie en grados.

Medición de la orientación

Para obtener la orientación de la superficie, debemos cambiar una vez más el modo de funcionamiento de la aplicación, dejando el dispositivo en la misma posición que en la medición de inclinación. Para ello, abrimos el menú desde la izquierda y pulsamos en la opción llamada *Compass*, como se muestra en la siguiente figura.

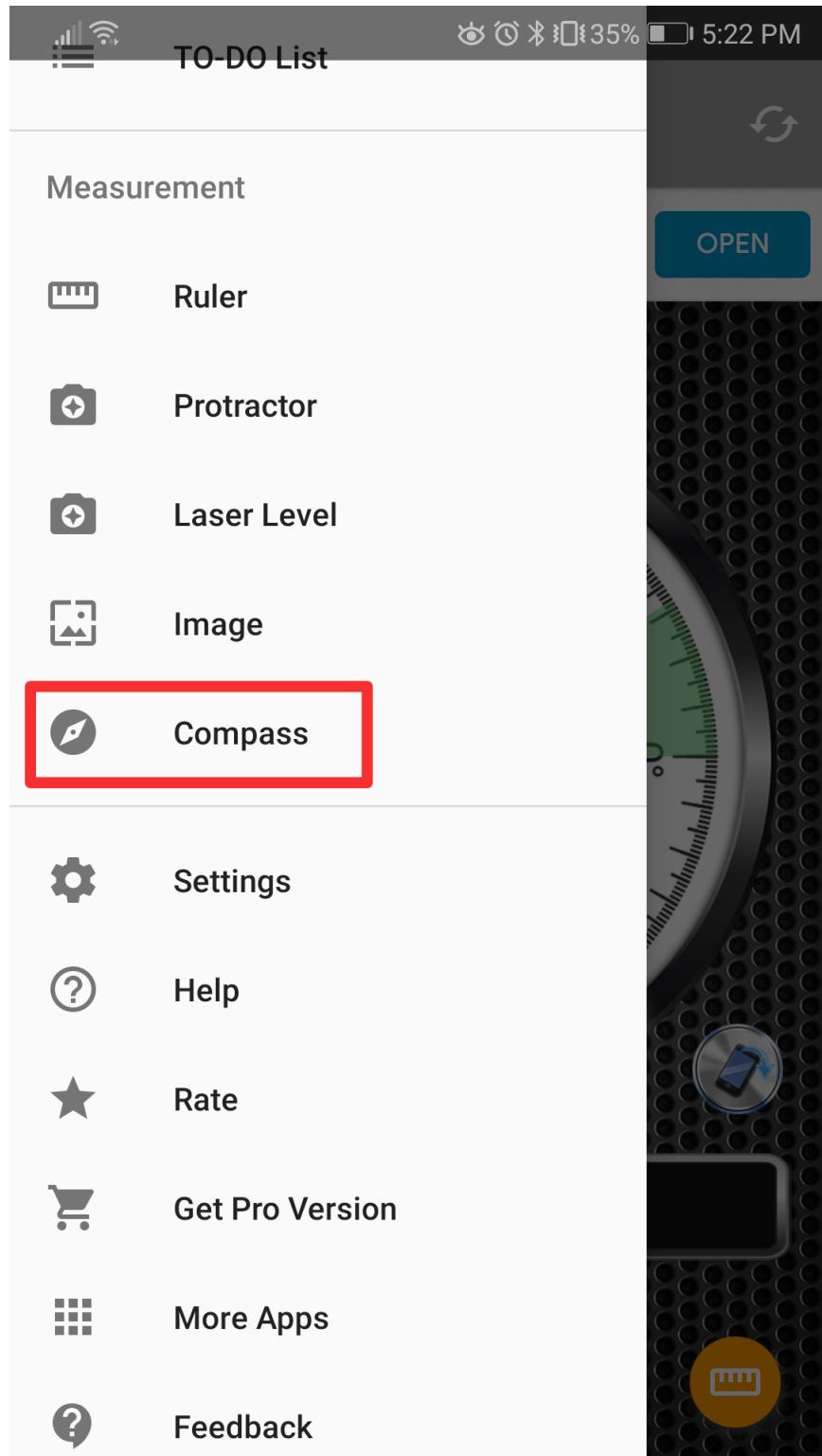


Figura 5.6: Opción brújula del menú

Una vez abierta la brújula, observamos la medición que nos muestra en la parte inferior central de la pantalla. A esa medición debemos restarle 180° para obtener el valor que necesita la aplicación

de cálculo. Es decir, si observamos una orientación de 193° , en el formulario introduciremos 13° . En la figura de abajo se muestra como se ve la pantalla de la brújula.

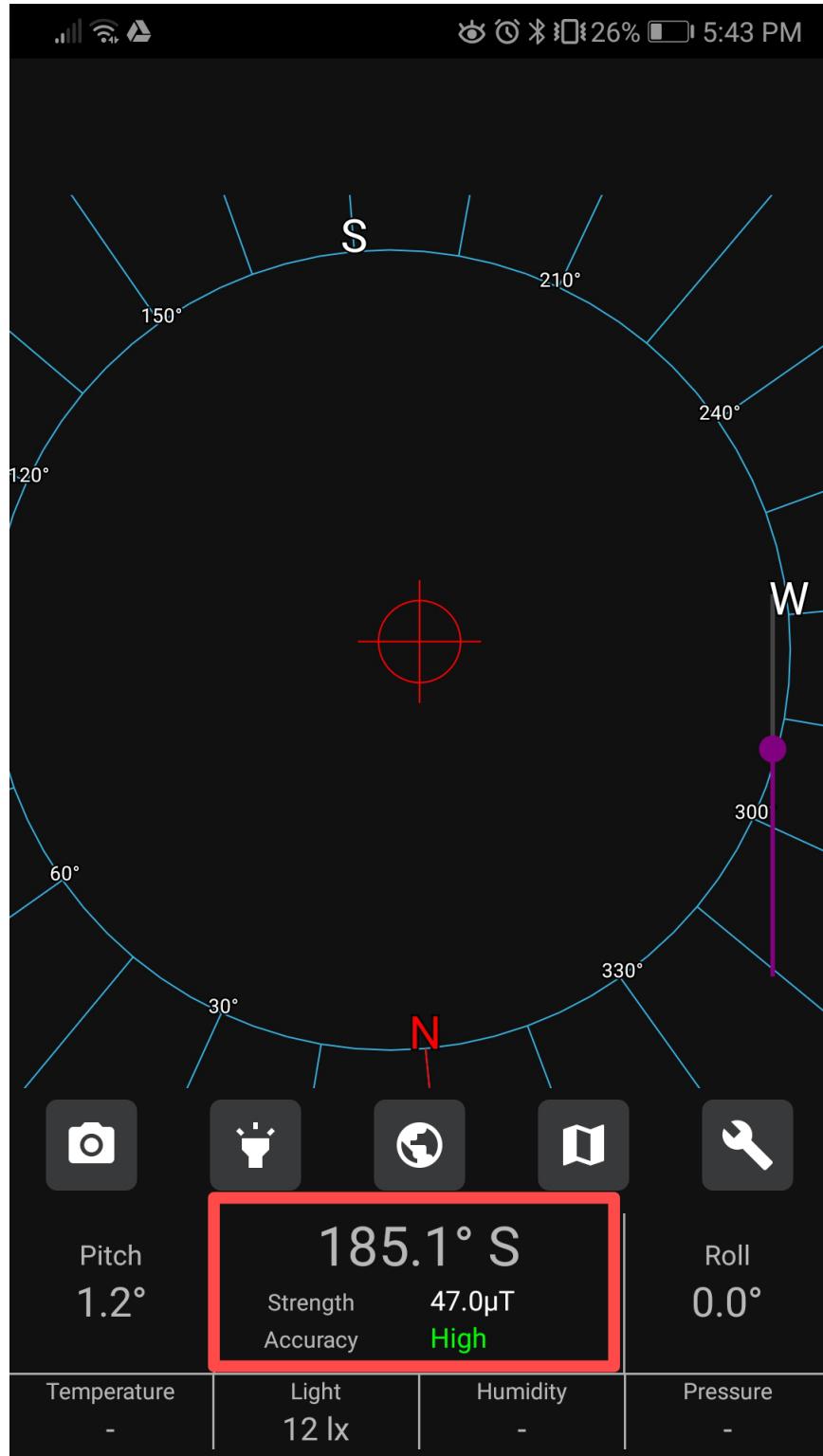


Figura 5.7: Opción brújula del menú

Apéndice A

Código completo

A.1. Código del servidor

A.1.1. DataController.js

```
1 const path = require('path');
2 const mongoose = require('mongoose');
3 const SolarData = mongoose.model('SolarData');
4 const StationData = mongoose.model('StationData');
5 const helpers = require('../helpers/helpers');
6 mongoose.Promise = global.Promise;
7 const axios = require('axios');

8
9 deg2rad = (degs) => {
10   return (degs * Math.PI) / 180;
11 };
12 rad2deg = (rads) => {
13   return (rads * 180) / Math.PI;
14 };

15
16 exports.getData = async (req, res, next) => {
17   const latitude = parseFloat(req.query.latitude);
18   const longitude = parseFloat(req.query.longitude);
19   const location = await SolarData.find({
20     location: {
```

```
21   $near: {
22     $maxDistance: 10000,
23     $geometry: {
24       type: 'Point',
25       coordinates: [longitude, latitude]
26     }
27   }
28 }
29 });
30 const locationData = location[0];
31 res.json({ data: locationData });
32 };
33 exports.saveData = async (req, res, next) => {
34   const data = require('./data.json');
35   data.forEach(async (elem) => {
36     const data = await new SolarData({
37       location: {
38         type: 'Point',
39         coordinates: [parseFloat(elem.lon), parseFloat(elem.lat)]
40       },
41       meanValues: elem.midValues
42     }).save();
43   });
44   res.json({ Data: 'Arrived' });
45 };
46
47 exports.doCalculations = async (req, res, next) => {
48   if (req.query.longitude == undefined || req.query.latitude == undefined) {
49     res.status(400).send('Longitude or latitude is undefined');
50   }
51   const latitude = parseFloat(req.query.latitude);
52   const longitude = parseFloat(req.query.longitude);
53   const angle = parseFloat(req.query.angle);
54   const applyDirtLevel = req.query.applyDirtLevel;
55   const dirtLevel = req.query.dirtLevel;
56   const orientation = parseFloat(req.query.orientation);
```

```
57 const location = await SolarData.find({
58   location: {
59     $near: {
60       $maxDistance: 25000,
61       $geometry: {
62         type: 'Point',
63         coordinates: [longitude, latitude]
64       }
65     }
66   }
67 });
68 if (location == []) {
69   res.status(204);
70   res.send('No data found for this location.
71 Please make sure you have entered a valid pair of coordinates');
72   return;
73 }
74 const locationData = location[0];
75
76 const calcData = {
77   longitude: locationData.location.coordinates[0],
78   latitude: locationData.location.coordinates[1],
79   angle,
80   orientation,
81   applyDirtLevel,
82   dirtLevel,
83   meanValues: [
84     {
85       month: 'Jan',
86       normalDay: 17,
87       meanGR: locationData.meanValues[0]
88     },
89     {
90       month: 'Feb',
91       normalDay: 45,
92       meanGR: locationData.meanValues[1]
```

```
93     },
94     {
95         month: 'Mar',
96         normalDay: 74,
97         meanGR: locationData.meanValues[2]
98     },
99     {
100        month: 'Apr',
101        normalDay: 105,
102        meanGR: locationData.meanValues[3]
103    },
104    {
105        month: 'May',
106        normalDay: 135,
107        meanGR: locationData.meanValues[4]
108    },
109    {
110        month: 'Jun',
111        normalDay: 161,
112        meanGR: locationData.meanValues[5]
113    },
114    {
115        month: 'Jul',
116        normalDay: 199,
117        meanGR: locationData.meanValues[6]
118    },
119    {
120        month: 'Aug',
121        normalDay: 230,
122        meanGR: locationData.meanValues[7]
123    },
124    {
125        month: 'Sep',
126        normalDay: 261,
127        meanGR: locationData.meanValues[8]
128    },
```

```
129  {
130    month: 'Oct',
131    normalDay: 292,
132    meanGR: locationData.meanValues[9]
133  },
134  {
135    month: 'Nov',
136    normalDay: 322,
137    meanGR: locationData.meanValues[10]
138  },
139  {
140    month: 'Dic',
141    normalDay: 347,
142    meanGR: locationData.meanValues[11]
143  }
144 ]
145 };
146
147 const resultData = helpers.calculateValues(calcData);
148
149 res.json({ data: resultData });
150 };
151
152 exports.saveStations = async (req, res, next) => {
153   const response = await axios(
154     'https://opendata.aemet.es/opendata/api/valores
155     /climatologicos/inventarioestaciones/todasestaciones/?api_key='
156     + process.env.AEMET_API
157   );
158   const data = response.data;
159   const datosResponse = await axios(data.datos);
160   const estaciones = datosResponse.data;
161   const convertedStations = estaciones.map((station) => {
162     const latLetter = station.latitud[station.latitud.length - 1];
163     const lonLetter = station.longitud[station.longitud.length - 1];
164     const latSign = latLetter == 'N' ? 1 : -1;
```

```
165     const lonSign = lonLetter == 'W' ? -1 : 1;
166     const lat = latSign * (parseFloat(station.latitud.substr(0,
167         station.latitud.length - 1)) / 10000);
168     const lon = lonSign * (parseFloat(station.longitud.substr(0,
169         station.longitud.length - 1))/ 10000);
170     const stationObj = {
171         emaid: station.indicativo,
172         lat,
173         lon
174     };
175     return stationObj;
176 });
177 convertedStations.forEach(async (elem) => {
178     const data = await new StationData({
179         location: {
180             type: 'Point',
181             coordinates: [elem.lon, elem.lat]
182         },
183         emaid: elem.emaid
184     }).save();
185 });
186 res.json({ Data: 'Arrived' });
187 };
188
189 exports.getTemperatureProfile = async (req, res, next) => {
190     const latitude = parseFloat(req.query.latitude);
191     const longitude = parseFloat(req.query.longitude);
192     let response;
193     const stations = await StationData.find({
194         location: {
195             $near: {
196                 $maxDistance: 50000,
197                 $geometry: {
198                     type: 'Point',
199                     coordinates: [longitude, latitude]
200                 }
201             }
202         }
203     });
204     if (stations.length > 0) {
205         response = {
206             stations: stations.map(station => {
207                 const distance = calculateDistance(
208                     {lat: latitude, lon: longitude},
209                     {lat: station.latitud, lon: station.longitud}
210                 );
211                 return {
212                     ...station._id,
213                     distance
214                 };
215             })
216         };
217     } else {
218         response = {
219             message: 'No stations found'
220         };
221     }
222     res.json(response);
223 }
224
225 function calculateDistance({lat: lat1, lon: lon1}, {lat: lat2, lon: lon2}) {
226     const R = 6371000; // Earth radius in meters
227     const dLat = degToRad(lat2 - lat1);
228     const dLon = degToRad(lon2 - lon1);
229     const a =
230         Math.sin(dLat / 2) * Math.sin(dLat / 2) +
231         Math.cos(degToRad(lat1)) * Math.cos(degToRad(lat2)) *
232         Math.sin(dLon / 2) * Math.sin(dLon / 2);
233     const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
234     const distance = R * c;
235     return distance;
236 }
237
238 function degToRad(degrees) {
239     return degrees * Math.PI / 180;
240 }
```

```
201     }
202   }
203 });
204 const monthlyValues = [];
205 stations.forEach(async (station, index) => {
206   const { emaId } = station;
207   const aemetLink = `https://opendata.aemet.es/
208   opendata/api/valores/climatologicos/mensualesanuales/datos
209   /anioini/2015/aniofin/2015/estacion/${emaId}/?api_key=${
210     process.env.AEMET_API
211   }`;
212   const firstRes = await axios(aemetLink);
213   const firstData = firstRes.data;
214   if (firstData.estado == 404) {
215     return;
216   } else if (firstData.estado == 200) {
217     const secondRes = await axios(firstData.datos);
218     const secondData = secondRes.data;
219     monthlyValues.push(secondData);
220   }
221   if (index == stations.length - 1) {
222     const tempArray = monthlyValues[0].map((month) => {
223       const monthNumber = month.fecha.split('-')[1];
224       const Tmin = parseFloat(month.tm_min) || 5;
225       const Tmax = parseFloat(month.tm_max) || 20;
226       return { Tmin, Tmax, monthNumber };
227     });
228     const normalDays = [17, 45, 74, 105, 135, 161, 199, 230, 261, 292, 322, 347];
229
230     const wp = Math.PI / 4;
231
232     tempArray.pop();
233
234     const tempProfiles = tempArray.map((elem, index) => {
235       const { Tmax, Tmin, monthNumber } = elem;
236       const Tm = (Tmax + Tmin) / 2;
```

```

237     const Tr = (Tmax - Tmin) / 2;
238
239     const Ta = [];
240
241     const decl = 23.45 * Math.sin((2 * Math.PI * (normalDays[index] + 284)) / 365);
242     const cosWs = -Math.tan(deg2rad(decl)) * Math.tan(deg2rad(latitude));
243     const ws = -Math.acos(cosWs);
244
245     for (let h = -12; h < 12; h++) {
246
247         const w = Math.cos(deg2rad(h * 15));
248         const a1 = (Math.PI * 12 * (ws - w)) / (21 * Math.PI + 12 * ws);
249         const a2 = (Math.PI * (3 * Math.PI - 12 * w)) / (3 * Math.PI - 12 * ws);
250         const a3 = (Math.PI * (24 * Math.PI + 12 * (ws - w))) /
251             (21 * Math.PI + 12 * ws);
252
253         const T1 = Tm - Tr * Math.cos(a1);
254         const T2 = Tm + Tr * Math.cos(a2);
255         const T3 = Tm - Tr * Math.cos(a3);
256
257         if (w <= ws) {
258             Ta.push(T1);
259         } else if (w > ws && w <= wp) {
260             Ta.push(T2);
261         } else if (w > wp) {
262             Ta.push(T3);
263         }
264     }
265
266     return { hourlyTa: Ta, monthNumber, Tmax, Tmin };
267 });
268
269 });
270

```

Extracto de código A.1: Server/Controllers/dataController.js

A.1.2. Helpers.JS

```

1 deg2rad = (degs) => {
2     return (degs * Math.PI) / 180
3 }
4 rad2deg = (rads) => {
5     return (rads * 180) / Math.PI
6 }
7
8 const cos = (rads) => {
9     return Math.cos(rads)
10 }
11
12 const sin = (rads) => {
13     return Math.sin(rads)
14 }
15 exports.calculateValues = (data) => {
16     const newData = data
17     //1. Declinacion, excentricidad y angulo amanecer
18     newData.meanValues.forEach((elem, index) => {
19         const decl = 23.45 * Math.sin((2 * Math.PI * (elem.normalDay + 284)) / 365)
20         const exct = 1 + 0.033 * Math.cos((2 * Math.PI * elem.normalDay) / 365)
21         elem.decl = decl
22         elem.exct = exct
23         const cosWs = -Math.tan(deg2rad(decl)) * Math.tan(deg2rad(newData.latitude))
24         const ws = -Math.acos(cosWs)
25         elem.ws = ws
26     })
27     //2. Calculo Bod
28     newData.meanValues.forEach((elem, index) => {
29         const Bo = 1.367
30         const zenith = []
31         const tilt = []
32         const B00 = []
33         for (let h = -12; h < 12; h++) {
34             const cosZenit =

```

```

35     Math.cos(deg2rad(elem.decl)) * Math.cos(deg2rad(h * 15)) *
36     Math.cos(deg2rad(newData.latitude)) +
37     Math.sin(deg2rad(elem.decl)) * Math.sin(deg2rad(newData.latitude))
38   const zenithVal = rad2deg(Math.acos(cosZenit))
39   zenith.push(zenithVal)
40 }
41 elem.zenit = zenith
42 for (let h = -12; h < 12; h++) {
43   const betaRad = deg2rad(newData.angle)
44   const alphaRad = deg2rad(newData.orientation)
45   const hRad = deg2rad(h * 15)
46   const latRad = deg2rad(newData.latitude)
47   const declRad = deg2rad(elem.decl)
48   const anglIncidenciaVal =
49     Math.sign(newData.latitude) *
50     (Math.sin(betaRad) * Math.cos(alphaRad) * Math.cos(declRad) *
51      Math.cos(hRad) * Math.sin(latRad) -
52      Math.sin(betaRad) * Math.cos(alphaRad) * Math.cos(latRad) *
53      Math.sin(declRad)) +
54      Math.sin(betaRad) * Math.sin(alphaRad) * Math.cos(declRad) * Math.sin(hRad) +
55      Math.cos(betaRad) * Math.cos(declRad) * Math.cos(hRad) * Math.cos(latRad) +
56      Math.cos(betaRad) * Math.sin(declRad) * Math.sin(latRad)
57   console.log({
58     data: { angle: newData.angle, orientation: newData.orientation,
59             lat: newData.latitude, decl: elem.decl },
60     h,
61     anglIncidenciaVal,
62     mes: index,
63   })
64   const anglIncidencia = rad2deg(Math.acos(anglIncidenciaVal))
65   tilt.push(anglIncidencia)
66   elem.tilt = tilt
67 }
68
69 for (let h = 0; h < 24; h++) {
70   const B00val = Bo * elem.exct * Math.cos(deg2rad(elem.zenit[h]))

```

```
71     B00.push(B00val)
72 }
73 elem.B00 = B00
74
75 const bod =
76   -(24 / Math.PI) *
77     Bo *
78   elem.exct *
79   (elem.ws * Math.sin(deg2rad(newData.latitude)) *
80     Math.sin(deg2rad(elem.decl)) +
81     Math.cos(deg2rad(elem.decl)) * Math.cos(deg2rad(newData.latitude)))
82   * Math.sin(elem.ws))
83 elem.B0d0 = bod
84 })
85 //3. Calculo del indice de claridad y Fd
86 newData.meanValues.forEach((elem, index) => {
87   const Ktd = elem.meanGR / elem.B0d0
88   elem.Ktd = Ktd
89   const Fd = 1 - 1.13 * elem.Ktd
90   elem.Fd = Fd
91 })
92
93 //4. Calculo de Dd y Bd
94 newData.meanValues.forEach((elem, index) => {
95   const Dd0 = elem.Fd * elem.meanGR
96   const Bd0 = elem.meanGR - Dd0
97
98   elem.Dd0 = Dd0
99   elem.Bd0 = Bd0
100 })
101
102 //5. Calculo de la ecuacion de perfil (rd & rg)
103 newData.meanValues.forEach((elem, index) => {
104   const rd = []
105   for (let h = -12; h < 12; h++) {
106     const hRad = Math.cos(deg2rad(h * 15))
```

```
107     const rdval = (Math.PI / 24) * ((hRad - Math.cos(elem.ws))
108     / (elem.ws * Math.cos(elem.ws) - Math.sin(elem.ws)))
109     rd.push(rdval)
110   }
111   const a = 0.409 - 0.5016 * Math.sin(elem.ws + Math.PI / 3)
112   const b = 0.6609 + 0.4767 * Math.sin(elem.ws + Math.PI / 3)
113
114   const rg = []
115   let i = 0
116   for (let h = -12; h < 12; h++) {
117     const hRad = Math.cos(deg2rad(h * 15))
118     const rgval = rd[i] * (a + b * hRad)
119     rg.push(rgval)
120     i++
121   }
122   elem.rg = rg
123   elem.rd = rd
124 })
125
126 //6. Calculo de los valores horarios de G,D y B
127 newData.meanValues.forEach((elem, index) => {
128   const hourlyValues = []
129   let hour = -12
130   const dawn = rad2deg(elem.ws) / 15
131   elem.dawn = dawn
132   for (let i = 0; i < 24; i++) {
133     const cosHour = Math.cos(deg2rad(hour * 15))
134     const cosWs = Math.cos(elem.ws)
135     if (cosHour > cosWs) {
136       const Dh = elem.rd[i] * elem.Dd0
137       const Gh = elem.rg[i] * elem.meanGR
138       const Bh = Gh - Dh
139       hourlyValues.push({
140         Bh,
141         Dh,
142         Gh,
```

```
143         cosHour,
144         cosWs,
145     })
146 } else {
147     hourlyValues.push({
148     Bh: 0,
149     Dh: 0,
150     Gh: 0,
151     cosHour,
152     cosWs,
153   })
154 }
155 hour++
156 }
157 elem.hourlyValues = hourlyValues
158 }

159
160 //8. From horizontal plane to tilted plane
161 newData.meanValues.forEach((elem, index) => {
162   for (let i = 0; i < 24; i++) {
163     const zenithRad = deg2rad(elem.zenit[i])
164     const tiltRad = deg2rad(elem.tilt[i])
165     const betaRad = deg2rad(newData.angle)
166
167     const numerator = Math.max(0, Math.cos(tiltRad))
168     const denominator = Math.cos(zenithRad)
169     const Btilt = elem.hourlyValues[i].Bh * (numerator / denominator)
170     const k1 = elem.hourlyValues[i].Bh / elem.B00[i]
171     const DcTilt = elem.hourlyValues[i].Dh * k1 * (numerator / denominator)
172     const DiTilt = elem.hourlyValues[i].Dh * (1 - k1) * ((1 + Math.cos(betaRad)) / 2)
173     const Dtilt = DcTilt + DiTilt
174     const Gtilt = Btilt + Dtilt
175     elem.hourlyValues[i] = {
176       ...elem.hourlyValues[i],
177       Btilt,
178       Dtilt,
```

```
179     DcTilt,  
180     DiTilt,  
181     Gtilt,  
182     cosZenit: Math.cos(zenitRad),  
183   }  
184 }  
185 })  
186  
187 newData.isDirtApplied = newData.applyDirtLevel == 'true'  
188  
189 if (newData.applyDirtLevel == 'true') {  
190   newData.meanValues.forEach((elem, index) => {  
191     for (let i = 0; i < 24; i++) {  
192       const tiltRad = deg2rad(elem.tilt[i])  
193       const betaRad = deg2rad(newData.angle)  
194       let TDirtTClean = 0  
195       let ar = 0  
196       let c1 = 4 / (3 * Math.PI)  
197       let c2 = 0  
198       switch (newData.dirtLevel) {  
199         case 'CLEAN':  
200           TDirtTClean = 1  
201           ar = 0.17  
202           c2 = -0.069  
203           break  
204         case 'LOW':  
205           TDirtTClean = 0.98  
206           ar = 0.2  
207           c2 = -0.054  
208           break  
209         case 'MID':  
210           TDirtTClean = 0.97  
211           ar = 0.21  
212           c2 = -0.049  
213           break  
214         case 'HIGH':
```

```

215     TDirtTClean = 0.92
216     ar = 0.27
217     c2 = -0.023
218     default:
219         TDirtTClean = 0
220         ar = 0
221         c2 = 0
222     }
223     newData.TDirtTClean = TDirtTClean
224     newData.ar = ar
225     newData.c1 = c1
226     newData.c2 = c2
227     const FTB = (Math.exp(-Math.cos(tiltRad) / ar) -
228         Math.exp(-1 / ar)) / (1 - Math.exp(-1 / ar))
229     const expFTD =
230         -(1 / ar) *
231         (c1 * (Math.sin(betaRad) + (Math.PI - betaRad
232             - Math.sin(betaRad)) / (1 + Math.cos(betaRad))) +
233             c2 * Math.pow(Math.sin(betaRad) + (Math.PI -
234                 betaRad - Math.sin(betaRad)) / (1 + Math.cos(betaRad)), 2))
235     const FTD = Math.exp(expFTD)
236
237     const Btilt = elem.hourlyValues[i].Btilt * TDirtTClean * (1 - FTB)
238     const DiTilt = elem.hourlyValues[i].DiTilt * TDirtTClean * (1 - FTD)
239     const DcTilt = elem.hourlyValues[i].DcTilt * TDirtTClean * (1 - FTB)
240     const Dtilt = DiTilt + DcTilt
241     const Gtilt = Dtilt + Btilt
242
243     //Overwrite data with new values
244     elem.hourlyValues[i].Btilt = Btilt
245     elem.hourlyValues[i].DiTilt = DiTilt
246     elem.hourlyValues[i].DcTilt = DcTilt
247     elem.hourlyValues[i].Dtilt = Dtilt
248     elem.hourlyValues[i].Gtilt = Gtilt
249   }
250 })

```

```
251    }
252    //9. Elección de valores clave
253    newData.meanValues.forEach((elem, index) => {
254      const significantValues = []
255      elem.hourlyValues.forEach((value, index) => {
256        if (Math.abs(value.hour) < Math.abs(elem.dawn)) {
257          significantValues.push(value)
258        }
259      })
260      elem.significantValues = significantValues
261    })
262
263    //9.1 Checking results
264    newData.meanValues.forEach((elem, index) => {
265      let totalGr = 0
266      let totalGtilt = 0
267      elem.significantValues.forEach((value, index) => {
268        totalGr += value.Gh
269        totalGtilt += value.Gtilt
270      })
271    })
272
273    return newData
274  }
275
```

Extracto de código A.2: Server/helpers/helpers.js

A.1.3. SolarData.js

```
1 const mongoose = require('mongoose');
2 const Schema = mongoose.Schema;
3 mongoose.Promise = global.Promise;
4
5 const solarDataSchema = new Schema({
6   location: {
```

```

7   type: {
8     type: String,
9     default: 'Point'
10    },
11   coordinates: [
12     {
13       type: Number,
14       required: 'You must supply coordinates!'
15     }
16   ]
17 },
18 meanValues: [
19   {
20     type: Number
21   }
22 ]
23 });
24
25 solarDataSchema.index({ location: '2dsphere' });
26
27 module.exports = mongoose.model('SolarData', solarDataSchema);

```

Extracto de código A.3: Server/models/SolarData.js

A.1.4. StationsData.js

```

1 const mongoose = require('mongoose');
2 const Schema = mongoose.Schema;
3 mongoose.Promise = global.Promise;
4
5 const stationDataSchema = new Schema({
6   location: {
7     type: {
8       type: String,
9       default: 'Point'
10      },

```

```

11   coordinates: [
12     {
13       type: Number,
14       required: 'You must supply coordinates!'
15     }
16   ],
17 },
18 emaId: {
19   type: String
20 }
21 });

22
23 stationDataSchema.index({ location: '2dsphere' });
24
25 module.exports = mongoose.model('StationData', stationDataSchema);

```

Extracto de código A.4: Server/models/StationsData.js

A.1.5. Router.js

```

1 const express = require('express');
2 const router = express.Router();
3
4 const dataController = require('../controllers/dataController');
5
6 // Do work here
7 router.get('/', (req, res) => {
8   res.send('Get your solar data from /solar-data');
9 });
10
11 router.get('/solar-data', dataController.getData);
12
13 //router.get('/save-solar-data', dataController.saveData);
14 //This route is used to save the solar data to the db from the json file (Only do once)
15
16 //router.get('/save-stations', dataController.saveStations);

```

```

17 // This route is used to save the EMA ID of stations in Spain by coordinates from the AEMET OpenData API
18 // (In case API fails & to not abuse their API) (Only do once)
19
20 router.get('/do-calculations', dataController.doCalculations);
21
22 router.get('/temp-profile', dataController.getTemperatureProfile);
23
24 module.exports = router;

```

Extracto de código A.5: Server/routes/index.js

A.1.6. App.js

```

1 const express = require('express');
2 const bodyParser = require('body-parser');
3
4 const routes = require('./routes/index');
5 const app = express();
6
7 app.use(bodyParser.urlencoded({ extended: true }));
8
9 app.use(function(req, res, next) {
10   res.header('Access-Control-Allow-Origin', '*');
11   res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept');
12   next();
13 });
14
15 app.use('/', routes);
16
17 module.exports = app;

```

Extracto de código A.6: Server/App.js

A.1.7. Start.js

```

1 const fs = require('fs');

```

```
2 const http = require('http');
3 const https = require('https');
4 const mongoose = require('mongoose');
5
6 // Certificate
7 const privateKey = fs.readFileSync('/etc/letsencrypt/live/solar-calc.ionut.cc/privkey.pem', 'utf8');
8 const certificate = fs.readFileSync('/etc/letsencrypt/live/solar-calc.ionut.cc/cert.pem', 'utf8');
9 const ca = fs.readFileSync('/etc/letsencrypt/live/solar-calc.ionut.cc/chain.pem', 'utf8');
10
11 const credentials = {
12   key: privateKey,
13   cert: certificate,
14   ca: ca
15 };
16
17 // Make sure we are running node 7.6+
18 const [major, minor] = process.versions.node.split('.').map(parseFloat);
19 if (major < 7 || (major === 7 && minor <= 5)) {
20   console.log(`Wrong version. Update your nodejs version`);
21   process.exit();
22 }
23
24 // import environmental variables from our variables.env file
25 require('dotenv').config({ path: 'variables.env' });
26
27 // Connect to our Database and handle any bad connections
28 mongoose.connect(process.env.DATABASE);
29 mongoose.Promise = global.Promise; // Tell Mongoose to use ES6 promises
30 mongoose.connection.on('error', (err) => {
31   console.error(`Error ${err.message}`);
32 });
33
34 require('./models/SolarData');
35 require('./models/StationsData');
36
37
```

```

38 const app = require('./app');

39

40 // Starting both http & https servers
41 const httpServer = http.createServer(app);
42 const httpsServer = https.createServer(credentials, app);

43

44 httpServer.listen(80, () => {
45   console.log('HTTP Server running on port 80');
46 });

47

48 httpsServer.listen(443, () => {
49   console.log('HTTPS Server running on port 443');
50 });

```

Extracto de código A.7: Server/start.js

A.2. Código de cliente

A.2.1. dataAquisition.js

```

1 const googleEndpoint = 'https://maps.googleapis.com/maps/api/geocode/json?';
2 const serverEndpoint = 'https://solar-calc.ionut.cc';

3

4 const addressInput = document.querySelector('#address');
5 const cityInput = document.querySelector('#city');
6 const postalInput = document.querySelector('#postal');
7 const coordBtn = document.querySelector('#button-get-coords');
8 const coordsContainer = document.querySelector('#coords-container');
9 const slope = document.querySelector('#slope');
10 const area = document.querySelector('#area');
11 const orientation = document.querySelector('#orientation');
12 const dirtLevel = document.querySelector('#dirt-level');
13 const applyDirt = document.querySelector('#apply-dirt');
14 const tableBody = document.querySelector('#radiation-data');

15

16 let calcData = {};

```

```
17 const googleApiKey = API_KEYS.GOOGLE_API_KEY;
18
19 coordBtn.addEventListener('click', getCalcData);
20
21 const moduleData = {
22   Gstar: 1000,
23   VocStar: 46.4,
24   IscStar: 9.05,
25   VmppStar: 37.4,
26   ImppStar: 8.56,
27   Ncs: 12,
28   Ncp: 6,
29   TONC: 45,
30   Tc: 25,
31   Vt: 0.025,
32   m: 1.3,
33   moduleArea: 1.957 * 0.992,
34   nominalPower: 320
35 };
36
37 const generatorData = {
38   seriesModules: 12,
39   parallelModules: 11
40 };
41
42 const inverterData = {
43   k0: 0.01,
44   k1: 0.025,
45   k2: 0.05,
46   power: 40000,
47   vMin: 420,
48   vMax: 750
49 };
50
51 async function getCalcData() {
52   coordBtn.textContent = 'Calculando';
```

```
53 coordBtn.disabled = true;
54 coordBtn.classList.add('is-loading');

55
56 calcData.placement = await getCoordinates();

57
58 calcData.surfaceInfo = {
59     area: area.value,
60     slope: slope.value,
61     orientation: orientation.value,
62     dirtLevel: dirtLevel.value,
63     applyDirt: true
64 };
65 const radiationData = await doCalculations(calcData);
66 const cellTempProfile = await calculateCellTemp(calcData, radiationData);
67 console.log(cellTempProfile);
68 const VocProfile = calculateVoc(cellTempProfile);
69 const IscProfile = calculateIsc(radiationData);
70 const { VmppProfile, ImppProfile, RsStar, rs, koc, DMO, DM, impp, vmpp } = applyVariableFF(VocProfile);
71 const PmppProfile = calculatePmpp(ImppProfile, VmppProfile);
72 const PdcProfile = calculateGeneratorPower(PmppProfile);
73 const PiProfile = calculatePiProfile(PdcProfile);
74 const PoProfile = calculatePoProfile(PiProfile);
75 const PacProfile = calculatePacProfile(PoProfile);
76 const MonthlyEac = calculateMonthlyEac(PacProfile);
77 const MonthlyEdc = calculateMonthlyEdc(PdcProfile);
78 const monthlyYf = MonthlyEac.map(
79     (val) => val / (moduleData.VmppStar * moduleData.ImppStar * generatordata.seriesModules * generatordata.yf)
80 );
81 const MonthlyEnergykWh = MonthlyEac.map((val) => val / 1000);
82 const AnualEnergyW = calculateAnualEnergy(MonthlyEac);
83 const AnualEnergykWh = AnualEnergyW / 1000;
84 const anualYf = AnualEnergyW / (moduleData.VmppStar * moduleData.ImppStar * generatordata.seriesModules * generatordata.yf);

85
86 // createTable(radiationData, MonthlyEac, MonthlyEdc, Yf, tableBody);

87
88 console.log({
```

```
89     radiationData,
90     PdcProfile,
91     PacProfile,
92     MppValues: { VmppProfile, ImppProfile },
93     VocProfile,
94     IscProfile,
95     cellTempProfile,
96     IscProfile,
97     variableFF: { RsStar, rs, koc, DMO, DM, impp, vmpp }
98   );
99
100  saveData(
101    radiationData,
102    MonthlyEnergykWh,
103    AnualEnergykWh,
104    monthlyYf,
105    PacProfile,
106    anualYf,
107    calcData.placement,
108    calcData.surfaceInfo.area,
109    calcData.surfaceInfo.slope,
110    calcData.surfaceInfo.orientation
111  );
112
113  window.location = 'results.html';
114}
115
116const getCoordinates = async () => {
117  const address = addressInput.value.split(' ').join('+');
118  const city = cityInput.value;
119  const postal = postalInput.value;
120  const requestURL = `${googleEndpoint}address=${address},${city},${postal},spain&key=${googleApiKey}`;
121  const response = await fetch(requestURL);
122  const data = await response.json();
123  console.log(data);
124  const info = {
```

```
125     formattedAddress: data.results[0].formatted_address,
126     lat: data.results[0].geometry.location.lat,
127     long: data.results[0].geometry.location.lng
128   };
129
130   return info;
131 };
132
133 const doCalculations = async (calcData) => {
134   const requestURL = `${serverEndpoint}/do-calculations?latitude=${calcData.placement.lat}&longitude=${calcData.placement.lng}&surfaceInfo.slope=${calcData.surfaceInfo.slope}&area=${calcData.surfaceInfo.area}&orientation=${calcData.surfaceInfo.orientation}&applyDirtLevel=${calcData.surfaceInfo.applyDirt}&dirtLevel=${calcData.surfaceInfo.dirtLevel}`;
135
136   const res = await fetch(requestURL);
137   const data = await res.json();
138   return data.data;
139 };
140
141 const calculateCellTemp = async (calcData, radiationData) => {
142   const requestURL = `${serverEndpoint}/temp-profile?latitude=${calcData.placement.lat}&longitude=${calcData.placement.lng}&tiltAngle=${calcData.placement.tilt}&azimuthAngle=${calcData.placement.azimuth}&cellSize=${calcData.cellSize}&radiationType=${calcData.radiationType}&radiationValue=${radiationData.meanValues}&radiationProfileIndex=0`;
143   const res = await fetch(requestURL);
144   const data = await res.json();
145   const tempProfiles = data.profiles;
146   const cellTempProfiles = tempProfiles.map(({ hourlyTa, Tmax, Tmin }, index) => {
147     const meanValue = radiationData.meanValues[index];
148     const TCPprofile = hourlyTa.map((temp, index) => {
149       const Gef = meanValue.hourlyValues[index].Gtilt * 1000;
150       const Tc = 25 + (Gef * (moduleData.TONC - 20)) / 800; //Cambiar 1000 por la G en el plano inclinado
151       return Tc;
152     });
153     return { TCPprofile, month: index + 1, Tmax, Tmin };
154   });
155   return cellTempProfiles;
156 };
157
158
159
160 };
```

```
161
162 const calculateVoc = (cellTempProfile) => {
163   const VocProfile = cellTempProfile.map(({ TCPprofile }, index) => {
164     const VocArray = TCPprofile.map((temp) => {
165       const Voc = moduleData.VocStar + (temp - moduleData.Tc) * (-2.3 / 1000) * moduleData.Ncs;
166       return Voc;
167     });
168     return VocArray;
169   });
170   return VocProfile;
171 };
172
173 const calculateIsc = (radiationData) => {
174   const IscProfile = radiationData.meanValues.map(({ hourlyValues }) => {
175     const IscArray = hourlyValues.map(({ Gtilt }) => {
176       const IscValue = Gtilt * 1000 * (moduleData.IscStar / moduleData.Gstar);
177       return IscValue;
178     });
179     return IscArray;
180   });
181   return IscProfile;
182 };
183
184 const applyVariableFF = (VocProfile, IscProfile, cellTempProfile) => {
185   const Vtn = (moduleData.Vt * (273 + 25)) / 300;
186   const RsStar =
187     (moduleData.VocStar / moduleData.Ncs -
188      moduleData.VmppStar / moduleData.Ncs +
189      moduleData.m * Vtn * Math.log(1 - moduleData.ImppStar / moduleData.IscStar)) /
190     (moduleData.ImppStar / moduleData.Ncp);
191
192   const rs = VocProfile.map((VocArray, arrIndex) => {
193     return VocArray.map((VocValue, valIndex) => {
194       return RsStar * ((moduleData.Ncs / moduleData.Ncp) * (IscProfile[arrIndex][valIndex] / VocValue));
195     });
196   });
}
```

```
197 const koc = VocProfile.map((VocArray, arrIndex) => {
198   return VocArray.map((VocValue, valIndex) => {
199     return VocValue / moduleData.Ncs / (moduleData.m * ((moduleData.Vt * (cellTempProfile[arrIndex].T - moduleData.T0)) / moduleData.R));
200   });
201 });
202 const DMO = koc.map((kocArray, arrIndex) => {
203   return kocArray.map((kocValue, valIndex) => {
204     return (kocValue - 1) / (kocValue - Math.log(kocValue));
205   });
206 });
207 const DM = DMO.map((DMOArray, arrIndex) => {
208   return DMOArray.map((DMOValue, valIndex) => {
209     return DMOValue + 2 * rs[arrIndex][valIndex] * Math.pow(DMOValue, 2);
210   });
211 });
212
213 const impp = DM.map((DMArry, arrIndex) => {
214   return DMArry.map((DMValue, valIndex) => {
215     return 1 - DMValue / koc[arrIndex][valIndex];
216   });
217 });
218
219 const vmp = impp.map((imppArray, arrIndex) => {
220   return imppArray.map((imppValue, valIndex) => {
221     return 1 - Math.log(koc[arrIndex][valIndex] / DM[arrIndex][valIndex]) / koc[arrIndex][valIndex];
222   });
223 });
224
225 const Vmp = vmp.map((vmppArray, arrIndex) => {
226   return vmppArray.map((vmppValue, valIndex) => {
227     return vmppValue * VocProfile[arrIndex][valIndex];
228   });
229 });
230 const Imp = impp.map((imppArray, arrIndex) => {
231   return imppArray.map((imppValue, valIndex) => {
232     return imppValue * IscProfile[arrIndex][valIndex];
```

```
233     });
234   });
235 
236   return { VmppProfile: Vmpp, ImppProfile: Impp, RsStar, rs, koc, DMO, DM, impp, vmpp };
237 };
238 
239 const calculatePmpp = (ImppProfile, VmppProfile) => {
240   const PmppProfile = ImppProfile.map((ImppArray, dayIndex) => {
241     const PmppArray = ImppArray.map((ImppValue, hourIndex) => {
242       return ImppValue * VmppProfile[dayIndex][hourIndex];
243     });
244     return PmppArray;
245   });
246   return PmppProfile;
247 };
248 
249 const calculateGeneratorPower = (PmppProfile) => {
250   const PdcProfile = PmppProfile.map((PmppArray) => {
251     return PmppArray.map((PmppValue) => {
252       return PmppValue * generatordata.seriesModules * generatordata.parallelModules;
253     });
254   });
255   return PdcProfile;
256 };
257 
258 const calculatePiProfile = (PdcProfile) => {
259   return PdcProfile.map((PdcArray) => {
260     return PdcArray.map((PdcValue) => {
261       return PdcValue / inverterData.power;
262     });
263   });
264 };
265 
266 const calculatePoProfile = (PiProfile) => {
267   return PiProfile.map((PiArray) => {
268     return PiArray.map((PiValue) => {
```

```
269     if (PiValue <= 0) {
270
271         return 0;
272
273     const a = inverterData.k2;
274
275     const b = inverterData.k1 + 1;
276
277     const c = inverterData.k0 - PiValue;
278
279     const firstSolution = (-b + Math.sqrt(b * b - 4 * a * c)) / (2 * a);
280
281     const secondSolution = (-b - Math.sqrt(b * b - 4 * a * c)) / (2 * a);
282
283     return firstSolution > 0 ? firstSolution : secondSolution;
284
285 });
286
287 });
288
289
290 const calculatePacProfile = (PoProfile) => {
291
292     return PoProfile.map((PoArray) => {
293
294         return PoArray.map((PoValue) => {
295
296             return PoValue * inverterData.power;
297
298         });
299
300     });
301
302
303 const calculateMonthlyEac = (PacProfile) => {
304
305     return PacProfile.map((PacArray) => {
306
307         return (
308
309             30 *
310
311             PacArray.reduce((total, currentValue) => {
312
313                 return total + (currentValue > 0 ? currentValue : 0);
314
315                 EacCell.textContent = MonthlyEac[index];
316
317             }, 0)
318
319         );
320
321     });
322
323 };
324
325
326 const calculateMonthlyEdc = (PdcProfile) => {
327
328     return PdcProfile.map((PdcArray) => {
329
330         return (
```

```
305     30 *
306     PdcArray.reduce((total, currentValue) => {
307         return total + (currentValue > 0 ? currentValue : 0);
308     }, 0)
309 );
310 });
311 };
312
313 const calculateAnualEnergy = (MonthlyEnergy) => {
314     return MonthlyEnergy.reduce((total, currentValue) => {
315         return total + (currentValue > 0 ? currentValue : 0);
316     }, 0);
317 };
318
319 const createTable = (radiationData, MonthlyEac, MonthlyEdc, Yf, tableBody) => {
320     tableBody.innerHTML = '';
321     const { meanValues } = radiationData;
322
323     meanValues.forEach((value, index) => {
324         const row = document.createElement('tr');
325         let Gefd = 0;
326         let Defd = 0;
327         let Befd = 0;
328
329         value.hourlyValues.forEach(hValue) => {
330             Gefd += hValue.Gtilt;
331             Defd += hValue.Dtilt;
332             Befd += hValue.Btilt;
333         });
334         const G0dCell = document.createElement('td');
335         G0dCell.textContent = value.meanGR;
336         const Bd0Cell = document.createElement('td');
337         Bd0Cell.textContent = value.Dd0;
338         const Dd0dCell = document.createElement('td');
339         Dd0dCell.textContent = value.Bd0;
340         const GefdCell = document.createElement('td');
```

```
341 GefdCell.textContent = Gefd;
342 const DefdCell = document.createElement('td');
343 DefdCell.textContent = Defd;
344 const BefdCell = document.createElement('td');
345 BefdCell.textContent = Befd;
346 const monthCell = document.createElement('td');
347 monthCell.textContent = index + 1;
348 const EacCell = document.createElement('td');
349 EacCell.textContent = MonthlyEac[index] / 1000;
350 const EdcCell = document.createElement('td');
351 EdcCell.textContent = MonthlyEdc[index] / 1000;
352 const YfCell = document.createElement('td');
353 YfCell.textContent = Yf[index];
354 row.appendChild(monthCell);
355 row.appendChild(GOdCell);
356 row.appendChild(Bd0Cell);
357 row.appendChild(Dd0dCell);
358 row.appendChild(GefdCell);
359 row.appendChild(DefdCell);
360 row.appendChild(BefdCell);
361 row.appendChild(EacCell);
362 row.appendChild(EdcCell);
363 row.appendChild(YfCell);
364 tableBody.appendChild(row);
365 });
366 };
367
368 const saveData = (radiationData, MonthlyEnergykWh, AnualEnergykWh, monthlyYf, PacProfile, AnualYf, loca
369 const totalModules = generatordata.seriesModules * generatordata.parallelModules;
370 const totalArea = moduleData.moduleArea * totalModules;
371 const areaRelation = area / totalArea;
372 const installedModules = parseInt(areaRelation * totalModules);
373 const installedPower = installedModules * moduleData.nominalPower;
374 const inverterPower = installedPower * 0.9;
375
376 const results = {
```

```

377     placementData: {
378         area,
379         location,
380         slope,
381         orientation
382     },
383     calculationsData: {
384         installedPower,
385         installedModules,
386         inverterPower,
387         AnualEnergykWh,
388         MonthlyEnergykWh,
389         monthlyYf,
390         AnualYf,
391         PacProfile,
392         areaRelation
393     },
394     radiationData
395 };
396 localStorage.setItem('results', JSON.stringify(results));
397 };
398 
```

Extracto de código A.8: public/scripts/DataAquisition.js

A.2.2. GenerateLinks.js

```

1 const minLat = 36.1;
2 const maxLat = 43.63;
3 const minLon = -9.15;
4 const maxLon = 3.06;
5 const latDiff = 0.25;
6 const lonDiff = 0.25
7 const links = [];
8 let data = [];
9 for (let i = minLat; i <= maxLat; i += latDiff) { 
```

```
10  for (let j = minLon; j <= maxLon; j += lonDiff) {
11    const lon = j.toFixed(2);
12    const lat = i.toFixed(2);
13    links.push(
14      `http://www.adrase.com/adrasemaps/php/monthly_popup.php?lat=${lat}&lon=${lon}&var_type=0` );
15  };
16  data.push({
17    lat,
18    lon
19  });
20}
21}
22
23const textPromises = links.map(async (link) => {
24  const res = await fetch(link, {
25    method: 'GET',
26    mode: 'no-cors'
27  });
28  const text = await res.text();
29  return text;
30});
31Promise.all(textPromises).then(function(values) {
32  let texts = [];
33
34  values.forEach(function(e, i) {
35    if (e.includes('NO HAY DATOS PARA EL PUNTO SELECCIONADO')) {
36      texts.push(null);
37    } else if (e.includes('Valor medio')) {
38      const regex = /[+-]?\d+(\.\d+)?/g;
39      const str = e.split('Valor medio')[1].split('tr')[0];
40      const floats = str.match(regex).map(function(v) {
41        return parseFloat(v);
42      });
43      texts.push(floats);
44    }
45  });
46});
```

```

46
47   for (let i = 0; i < texts.length; i++) {
48     data[i].midValues = texts[i];
49   }
50
51   const filteredData = data.filter((dataElement) => {
52     return dataElement.midValues != null;
53   });
54
55   stringData = JSON.stringify(filteredData);
56   console.log(stringData);
57 });
58

```

Extracto de código A.9: public/scripts/generateLinks.js

A.2.3. index.html

```

1  <!DOCTYPE html>
2
3  <html lang="en">
4
5    <head>
6      <title>Solar calculator</title>
7      <link rel="stylesheet" href=".//styles/style.css">
8      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bulma/0.6.2/css/bulma.css">
9      <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css">
10     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
11
12   </head>
13
14   <body>
15
16     <nav class="navbar is-warning" role="navigation" aria-label="main-navigation">
17       <div class="navbar-brand">
18         <a href="index.html" class="navbar-item title is-5">
19           <span class="icon is-large"><i class="fas fa-lg fa-solar-panel"></i></span> Solarcalc</a>
20         </div>

```

```
21 <div class="navbar-end">
22   <div class="navbar-item">
23     <p class="control">
24       <a href="#" target="_blank" class="has-text-dark title is-5">
25         <span>About</span>
26       </a>
27     </p>
28   </div>
29   <div class="navbar-item">
30     <p class="control">
31       <a href="https://github.com/IonutMorariu/PV-Calculator"
32           target="_blank" class="button is-dark">
33         <span class="icon">
34           <i class="fab fa-github-alt"></i>
35         </span>
36         <span>Github</span>
37       </a>
38     </p>
39   </div>
40 </div>
41 </nav>
42
43 <div class="container">
44   <div class="columns is-centered">
45     <div class="column is-half">
46       <h3 class="title is-4 has-text-centered">Datos del emplazamiento</h3>
47       <div class="field">
48         <label class="label" for="address">Direccion</label>
49         <div class="control has-icons-left has-icons-right">
50           <input class="input" type="text" placeholder="Calle"
51               id="address" required value="">
52           <span class="icon is-small is-left">
53             <i class="fas fa-map-marker"></i>
54           </span>
55         </div>
56       </div>
```

```
57      <div class="field is-horizontal">
58          <div class="field-body">
59              <div class="field">
60                  <label class="label" for="city">Municipio</label>
61                  <div class="control has-icons-left has-icons-right">
62                      <input class="input" type="text"
63                          placeholder="Municipio" id="city" required>
64                      <span class="icon is-small is-left">
65                          <i class="fas fa-building"></i>
66                      </span>
67                  </div>
68              </div>
69              <div class="field">
70                  <label class="label" for="postal">Código postal</label>
71                  <div class="control has-icons-left has-icons-right">
72                      <input class="input" type="text"
73                          placeholder="Código postal" id="postal" required>
74                      <span class="icon is-small is-left">
75                          <i class="fas fa-hashtag"></i>
76                      </span>
77                  </div>
78              </div>
79          </div>
80      </div>
81      <h3 class="title is-4">Datos de la casa</h3>
82      <div class="field is-horizontal">
83          <div class="field-body">
84              <div class="field">
85                  <label class="label" for="slope">Inclinación
86                  del tejado en grados </label>
87                  <div class="control has-icons-left has-icons-right">
88                      <input class="input" type="text"
89                          placeholder="Inclinación aproximada del tejado"
90                          id="slope" required value="20">
91                      <span class="icon is-small is-left">
92                          <i class="fas fa-home"></i>
```

```
93                     </span>
94                 </div>
95             </div>
96             <div class="field">
97                 <label class="label" for="area">rea del tejado
98                     en metros cuadrados</label>
99                 <div class="control has-icons-left has-icons-right">
100                     <input class="input" type="text"
101                         placeholder="rea del tejado" id="area"
102                         required value="23">
103                     <span class="icon is-small is-left">
104                         <i class="fas fa-calculator"></i>
105                     </span>
106                 </div>
107             </div>
108         </div>
109     </div>
110     <div class="field is-horizontal">
111         <div class="field-body">
112             <div class="field">
113                 <label for="" class="label">Orientacion</label>
114                 <p class="control has-icons-left">
115                     <input class="input" type="number"
116                         placeholder="Orientacion" id="orientation"
117                         required value="0">
118
119                     <!-- <span class="select">
120                         <select id="orientation">
121                             <option disabled>Orientacion</option>
122                             <option value="180">Norte (N)</option>
123                             <option value="225">Noreste(NE)</option>
124                             <option value="270">Este (E)</option>
125                             <option value="315">Sureste (SE)</option>
126                             <option selected value="0">Sur (S)</option>
127                             <option value="45">Suroeste (SW)</option>
128                             <option value="90">Oeste (W)</option>
```

```
129                     <option value="135">Noroeste (NW)</option>
130                 </select>
131             </span> -->
132             <span class="icon is-small is-left">
133                 <i class="fas fa-compass"></i>
134             </span>
135         </p>
136     </div>
137     <div class="field">
138         <label for="" class="label">Nivel de suciedad</label>
139         <p class="control has-icons-left">
140             <span class="select">
141                 <select id="dirt-level">
142                     <option disabled>Suciedad</option>
143                     <option value="CLEAN">Limpio</option>
144                     <option selected value="LOW">Nivel bajo</option>
145                     <option value="MID">Nivel medio</option>
146                     <option value="HIGH">Nivel alto</option>
147                 </select>
148             </span>
149             <span class="icon is-small is-left">
150                 <i class="fas fa-broom"></i>
151             </span>
152         </p>
153     </div>
154 </div>
155
156 </div>
157 <!-- <div class="field">
158     <label class="checkbox">
159         <input id="apply-dirt" type="checkbox" checked aria-checked="true">
160         Aplicar suciedad y angulo de incidencia (solo para pruebas)
161     </label>
162 </div> -->
163 <div class="control has-text-centered">
164     <a class="button is-warning is-centered" id="button-get-coords">
```

```
165             <span class="icon">
166                 <i class="fas fa-paper-plane"></i>
167             </span>
168             <span>Enviar</span>
169         </a>
170     </div>
171 </div>
172 <!--
173     <div>
174         <table class="test-data" border="1">
175             <thead>
176                 <tr>
177                     <th>Month</th>
178                     <th>G0d</th>
179                     <th>Bd0</th>
180                     <th>Dd0</th>
181                     <th>Gefd</th>
182                     <th>Defd</th>
183                     <th>Befd</th>
184                     <th>Eacd</th>
185                     <th>Edc</th>
186                     <th>Yf</th>
187                 </tr>
188             </thead>
189             <tbody id="radiation-data"></tbody>
190         </table>
191     </div> -->
192 </body>
193 <script src=".//scripts/apiKeys.js"></script>
194 <script src=".//scripts/dataAquisition.js"></script>
195
196
197
198 </html>
```

Extracto de código A.10: public/index.html

A.2.4. Results.html

```

1  <!DOCTYPE html>
2
3  <html lang="en">
4
5      <head>
6          <title>Resultados</title>
7          <link rel="stylesheet" href=".//styles/style.css" />
8          <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bulma/0.6.2/css/bulma.css" />
9          <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" />
10         <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0/dist/Chart.min.js"></script>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
12
13     </head>
14
15
16     <body>
17
18         <nav class="navbar is-warning" role="navigation" aria-label="main-navigation">
19             <div class="navbar-brand">
20                 <a href="index.html" class="navbar-item title is-5">
21                     <span class="icon is-large"><i class="fas fa-lg
22                         fa-solar-panel"></i></span> Solarcalc</a>
23
24             >
25
26         </div>
27
28         <div class="navbar-end">
29             <div class="navbar-item"></div>
30
31             <div class="navbar-item">
32                 <p class="control">
33                     <a href="https://github.com/IonutMorariu/PV-Calculator"
34                         target="_blank" class="button is-dark"
35                         <span class="icon">
36                             <i class="fab fa-github-alt"></i>
37                         </span>
38                         <span>Github</span>
39                     </a>
40
41                 </p>
42
43             </div>
44
45         </div>
46
47     </body>
48
49 
```

```
32         </div>
33     </div>
34 </nav>
35 <div class="section">
36     <div class="columns is-centered">
37         <div class="column is-half">
38             <div class="box">
39                 <h1 class="title is-4 has-text-centered">Resultados del cálculo</h1>
40                 <h2 class="title is-5">Datos del emplazamiento:</h2>
41                 <ul class="list">
42                     <div class="data-grid">
43                         <li><span class="has-text-weight-semibold">
44                             Latitud:</span> <span id="latitude"></span></li>
45                         <li><span class="has-text-weight-semibold">
46                             Longitud:</span> <span id="longitude"></span></li>
47                         <li><span class="has-text-weight-semibold">
48                             Radiación global media:</span>
49                             <span id="global-radiation"></span></li>
50                         <li><span class="has-text-weight-semibold">
51                             Superficie:</span> <span id="area"></span></li>
52                         <li><span class="has-text-weight-semibold">
53                             Inclinación:</span>
54                             <span id="slope"></span></li>
55                         <li><span class="has-text-weight-semibold">
56                             Orientación:</span>
57                             <span id="orientation"></span></li>
58                     </div>
59                 </ul>
60                 <hr style="margin: 0 0 3rem;" />
61                 <h2 class="title is-5">Datos de la instalación calculada:</h2>
62                 <ul class="list">
63                     <div class="columns">
64                         <div class="column is-one-half">
65                             <li><span class="has-text-weight-semibold">
66                                 Número de módulos:</span>
67                                 <span id="installed-modules"></span></li>
```

```
68         <li><span class="has-text-weight-semibold">
69             Potencia minima del inversor:</span>
70             <span id="inverter-power"></span></li>
71         </div>
72         <div class="column is-one-half">
73             <li><span class="has-text-weight-semibold">
74                 Potencia instalada:</span>
75                 <span id="installed-power"></span></li>
76             <li><span class="has-text-weight-semibold">
77                 Energia anual producida:</span>
78                 <span id="anual-energy"></span></li>
79             </div>
80         </div>
81     </ul>
82     <hr style="margin: 0 0 1rem;" />
83
84     <p class="content is-small">
85         Los calculos han sido realizados en
86         base al módulo de potencia nominal
87         320W del fabricante JINKO SOLAR, modelo JKM320PP
88         <a target="_blank" href="https://autosolar.es/pdf
89             /Ficha-Tecnica-Jinko-Solar-305-320W.pdf">ENLACE</a>
90         <br />
91         <!-- Esta aplicación es el
92             resultado del Trabajo de Fin de Grado de Ionut Morariu.
93             Puede leer todo el proceso del cálculo que se ha
94             seguido para conseguir los resultados en este documento.
95             <a target="_blank" href="https://github.com/
96                 IonutMorariu/pv-calculator">ENLACE</a> -->
97         </p>
98     </div>
99     <h2 class="title is-5 has-text-centered">
100        Gráficos adicionales:</h2>
101
102        <canvas id="chart-1" width="400" height="200"></canvas>
103        <canvas id="chart-2" width="400" height="200"></canvas>
```

```

104         <canvas id="chart-3" width="400" height="200"></canvas>
105     </div>
106   </div>
107 </div>
108 </body>
109 <script src="bling.js"></script>
110 <script src="renderResults.js"></script>
111 </html>
112

```

Extracto de código A.11: public/results.html

A.2.5. renderResults.js

```

] const results = JSON.parse(localStorage['results']) console.log(results) //Placement data ('area').innerHTML =
results.placementData.area + ' m <sup>2</sup>' ('latitude').textContent = results.placementData.location.lat
('longitude').textContent = results.placementData.location.long.toFixed(2) ('orientation').textContent
= results.placementData.orientation + ' °' ('slope').textContent = results.placementData.slope + '
('global-radiation').innerHTML = ( results.radiationData.meanValues.reduce((prev, next) => return
(prev += next.meanGR) , 0) / 12 ).toFixed(2) + ' kWh/m<sup>2</sup>'
//Calculation data
('installed - modules').textContent = results.calculationsData.installedModules('installed-power').textContent
= results.calculationsData.installedPower + ' W' ('inverter - power').textContent = results.calculationsData.inver
W('anual-energy').textContent = ((results.calculationsData.AnualEnergykWh / 1000) * results.calculationsData.a
+ ' MWh'
const ctx1 = document.getElementById('chart-1').getContext('2d') const ctx2 = document.getElementById('chart-2').getCo
const ctx3 = document.getElementById('chart-3').getContext('2d')
const myChart1 = new Chart(ctx1, type: 'line', data: labels: ['Enero', 'Febrero', 'Marzo', 'Abril',
'Mayo', 'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre'], datasets: [
label: 'Energia mensual en kWh', data: results.calculationsData.MonthlyEnergykWh.map(val =>(val
* results.calculationsData.areaRelation).toFixed(2)), backgroundColor: ['ff76751f'], borderColor: ['ff7675'],
lineTension: 0, , ], , options: responsive: true, scales: yAxes: [ ticks: beginAtZero: true, , , ], , )
const myChart2 = new Chart(ctx2, type: 'line', data: labels: ['Enero', 'Febrero', 'Marzo', 'Abril',
'Mayo', 'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre'], datasets: [
label: 'Productividad diaria media', data: results.calculationsData.monthlyYf.map(val =>(val /
30).toFixed(2)), backgroundColor: ['0984e31f'], borderColor: ['0984e3'], lineTension: 0, , ], , options:
responsive: true, scales: yAxes: [ ticks: beginAtZero: true, , , ], , ) console.log(results.calculationsData.PacProfile.
=>Math.max(...profile))) const myChart3 = new Chart(ctx3, type: 'line', data: labels: ['Enero',
'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre',
'Diciembre'], datasets: [ label: 'Potencia Máxima diaria en W', data: results.calculationsData.PacProfile.map(profil
=>(Math.max(...profile) * results.calculationsData.areaRelation).toFixed(2)), backgroundColor: ['00b8941f'],
borderColor: ['00b894'], lineTension: 0, , ], , options: responsive: true, scales: yAxes: [ ticks:
beginAtZero: true, , , ], , )

```

Bibliografía

- [1] Unión Española Fotovoltaica. 2019. Informe anual 2019: El sector fotovoltaico impulsor de la transición energética. https://unef.es/wp-content/uploads/dlm_uploads/2019/09/memoria_unef_2019-web.pdf.
- [2] PVsyst. PVsyst SA. <https://www.pvsyst.com/download-pvsyst/>.
- [3] CalculationSolar. 2014. <http://www.calculationsolar.com/>.
- [4] SISIFO.Instituto de Energía Solar. <https://www.sisifo.info/es/default>.
- [5] PVGIS. European Commission, Joint Research Centre. <https://re.jrc.ec.europa.eu/pvg-tools/es/tools.html>.
- [6] System Advisor Model. NREL, US Department of Energy <https://sam.nrel.gov/>.
- [7] Oscar Perpiñán (2012). solaR: Solar Radiation and Photovoltaic Systems with R, Journal of Statistical Software, 50(9), 1-32. <http://www.jstatsoft.org/v50/i09/>.
- [8] Perpiñán, O. 2018. Energía Solar Fotovoltaica. <http://oscarperpinan.github.io/esf/>.
- [9] B. Y. H. Liu y R. C. Jordan. "The interrelationship and characteristic distribution of direct,diffuse, and total solar radiation". *Solar Energy* 4 (1960), págs. 1-19
- [10] Estimating average daytime and daily temperature profiles within Europe. Thomas A.Huldm Marcel Šúri, Ewan D.Dunlop, Fabio Micale 2006 <https://www.sciencedirect.com/science/article/abs/pii/S1364815205001593>
- [11] NodeJS ORG. <https://nodejs.org/es/about/>.