Trabajo de Fin de Máster

Título: Desarrollo de una aplicación web para la configuración de instalaciones de energía solar fotovoltaica.

Autor: Aurélien Camilleri

Tutores:

Ángel Marqués Mateu: ETSIGCT, Universitat Politècnica de València

José Ángel Garrido Sarasol: Departamento Técnico I+D+i, TECHNO SUN

Curso: 2020/2021

TECHNO SUN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA GEODÉSICA CARTOGRÁFICA Y TOPOGRÁFICA

“El presente documento ha sido realizado completamente por el firmante; no ha sido entregado como otro trabajo académico previo y todo el material tomado de otras fuentes ha sido convenientemente entrecomillado y citado su origen en el texto, así como referenciado en la bibliografía.”

Aurélien Camilleri

Resumen

La idea general de este proyecto es crear una plataforma web que ayude a configurar y diseñar instalaciones solares fotovoltaicas. Esta plataforma será utilizada, en un principio, por los técnicos e ingenieros de TECHNOSUN, y luego, si todo funciona perfectamente, la plataforma podrá ser abierta a todo el mundo.

Objetivos

Los objetivos del proyecto son los siguientes:

- Crear un visor cartográfico con ortofotos en el que pueda hacer clic para seleccionar la longitud y la latitud de los equipos a instalar,
- Implementar los cálculos electrotécnicos para los equipos fotovoltaicos en Python,
- Utilizar datos astronómicos (orientación del sol, …) y meteorológicos (radiación solar, …) precisos,
- Crear una base de datos con el máximo número de paneles solares e inversores posibles (no necesariamente sólo los propuestos por TECHNOSUN),
- Que el proyecto sea exclusivamente gratis.

Metodología

Para iniciar el proyecto, primero se codificaron los cálculos electrónicos a partir de la documentación que aporto la empresa. Con estos PDFs, se creó una librería de funciones de cálculos electrónicos fotovoltaicos, en Python orientado objectos. Se ha elegido Python orientado objectos porque es más fácil llamar desde otros códigos y otros lenguajes de programación. Las variables y parámetros de entrada son: Longitud, Latitud, Características del panel solar y de los equipos fotovoltaicos (inversores, reguladores, variadores de frecuencia). Con consultas, se calculará la temperatura. La lista de materiales se podrá ampliar durante el desarrollo del software que continuará después de este TFM, en colaboración con la empresa.

Para trabajar con datos precisos, para la temperatura se utilizan las APIs de PVGIS. En el caso de la altitud, se utiliza la API para hacer la consulta a OpenTopoData. En ambos casos, resultan datos muy precisos y la respuesta es muy rápida. La fiabilidad de los datos consultados es buena y permite, a posteriori, cálculos muy precisos.

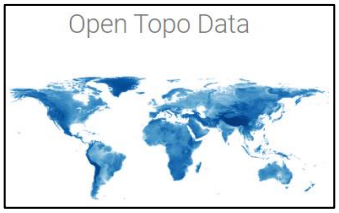


Figura 1 y 2: Logo de PVGIS (izquierda) y de OpenTopoData (derecha)

Inicialmente, no se disponía de coordenadas reales para probar las funciones y lo primero que se hizo fue desarrollar el visor.

La implementación del visor cartográfico se ha realizado utilizando los lenguajes de programación HTML y JavaScript. Posteriormente, se han utilizado los mapas Leaflet. Son

mapa en acceso y código libres. Al mismo tiempo, se van agregando y creando mecanismos para poder obtener las coordenadas del punto que se ha clicado.

Una vez obtenidas las coordenadas del punto, el siguiente paso, ha sido enlazar con la librería Python que contienen los métodos que procesan los cálculos para dichas coordenadas. Es el lenguaje de programación JavaScript, el que envía los datos de las coordenadas tipo de modulo fotovoltaico, tipo de inversor (etc…) a una capa intermedia de software llamada Django.

Django es un framework de lenguaje Python. Se ha elegido porque ya se tenía experiencia y porque ha demostrado ser eficiente, además que continua con la filosofía de software gratis. Es Django, el que lleva el peso de poder actuar directamente con los métodos implementados en Python.

A partir de los parámetros pasados por Django, los métodos de Python calculan y elaboran las respuestas.

Una vez que está bien implementado, se realizó la siguiente prueba:

- Recoger coordenadas desde el mapa,
- Pasarlas a las librerías Python mediante la intervención de Django,
- Inicialmente, se ha utilizado la función para determinar la altitud a partir de las coordenadas,
- Devolver la altitud al código JavaScript pasando por la capa Django,
- Con el fin de visualizar los datos en la pantalla.

Figura 3: Primera prueba de viaje de ida y vuelta entre el código Python y JavaScript

Este paso ha sido la clave para el posterior desarrollo de todas las consultas que se realizan entre las tres piezas software (JavaScript, Django y Python).

Ahora se tiene que crear la base de datos para permitir la elección del material fotovoltaico (módulos e inversores). Se ha decidido crear dos códigos Python, uno para objetos tipo modulo fotovoltaico y otro para los objetos tipo inversores. Al ejecutar, estos códigos rellenan la base de datos de la empresa, con los objetos equipos y módulos, y sus respectivos atributos técnicos. El formato utilizado es JSON, porque se ha convertido en el formato base de intercambio de datos tanto entre máquinas como entre máquinas y usuarios.

Siguiendo con el desarrollo, se pasa al diseño de la interfaz gráfica con más profundidad para mejorar la vista de la página y sus funcionalidades. Se añade más campos y controles para elegir el módulo y el inversor. Luego, se repite el proceso para mostrar en la página web, todos los campos de interés para los usuarios de la empresa.

Un toque final para embellecer la página con los colores del logo de TECHNOSUN y la paleta que de colores que lo acompañan.

Resultados

El resultado final es una página complemente funcional que contiene todos los datos que define los componentes del sistema fotovoltaico. Y los más importante del software es como relaciona las coordenadas cartográficas, determina las temperaturas extremas y medias para dichas coordenadas y finalmente determina el comportamiento exacto del material fotovoltaico para garantizar la fiabilidad y durabilidad del sistema o de la instalación.

La última versión del proyecto resulta así:

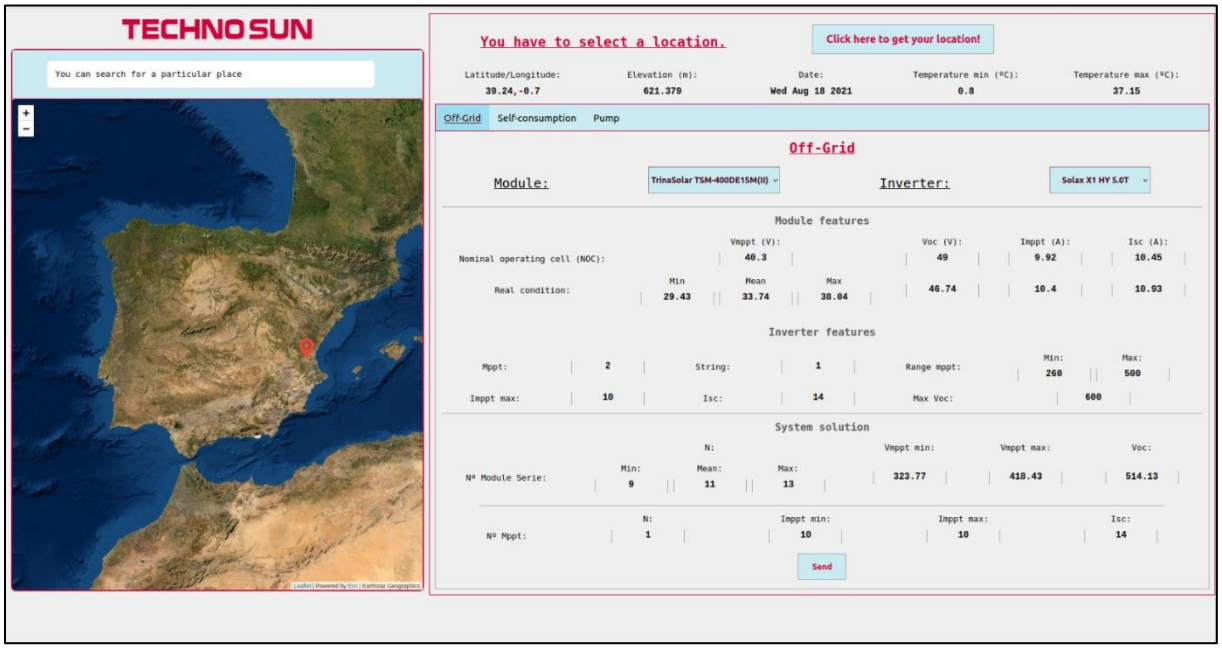


Figura 4: Pagina final del proyecto

Los campos los más interesantes para los usuarios de TECHNOSUN son los campos de la línea N° Module Serie porque dan la estimación final. Pasar por la página web que se ha hecho hace ganar tiempo: no hay que utilizar la calculadora, ni un tablero Excel complejo. Solo enseñar la posición, las marcas de los equipos y el resultado ya está aquí abajo. Además de todo, esta página permite evitar los errores de cálculo y descuidos.

Para que la página sea interesante y fácil de usar, he creado diferentes gadgets como:

- La barra de búsqueda con un geocoding para permitir al usuario de buscar un lugar en particular,
- El botón de localización que permite seleccionar automáticamente tu posición misma (también, te indica la precisión),
- El logo TECHNOSUN contiene un enlace que te dirige directamente en la página oficial de TECHNOSUN,
- Los botones + y –, arriba a la izquierda del mapa, funcionan para ampliarlo y reducirlo,
- Si se reduce el tamaño de la página, el mapa cambia de sitio.

Figura 5 y 6: Pagina web con tamaño de pantalla normal (izquierda) y con un tamaño de pantalla reducida (derecha)

Conclusiones

Para concluir,
El proyecto era complejo e interesante. Gracias a las habilidades desarrolladas durante el curso y a la ayuda de los tutores, el proyecto fue un éxito. Aunque hemos encontrado problemas, como la API de PVGIS que no se puede llamar desde JavaScript sino desde Python, ahora resulta de gran ayuda para los técnicos de TECHNOSUN. Todavía se tiene que verificar si cumplimos todos los objetivos:

- Crear un visor cartográfico con ortofotos en el que pueda hacer clic para seleccionar la longitud y la latitud de los equipos a instalar,

Hecho. Tenemos un visor operacional que permite hacer clic para recoger las coordenadas.

- Crear una base de datos con el máximo número de paneles solares e inversores posibles (no necesariamente sólo los propuestos por TECHNOSUN),

No totalmente hecho. Tenemos una base de datos completas y operacional. Sin embargo, solo contiene los datos de los equipos usados por la empresa. Se tiene, en el futuro, ampliar la base de datos.

- Implementar los cálculos electrotécnicos para los equipos fotovoltaicos en Python,

Hecho. Tenemos una grande librería de funciones usando los diferentes cálculos electrotécnicos. Puede ser necesario añadir otras funciones más adelante para realizar otros cálculos.

- Utilizar datos astronómicos (orientación del sol, …) y meteorológicos (radiación solar, …) precisos,

Hecho. Los datos que no eran seguros fueron sustituidos por datos de las APIs.

- Que sea exclusivamente gratis (al menos en primer lugar).

Hecho. Tenemos un resultado bien potente que puede competir con aplicaciones con costosas económicamente. Los datos que no eran seguros fueron sustituidos por datos de las APIs.

Los objetivos se cumplen. La página web cumple las expectativas.

Puntos negativos,

Todavía se quedan algunos puntos negativos:

- Si una de las APIs decida de cambiar de formato de datos, se necesitará adaptarnos.
- El algoritmo tarda un poco en dar un resultado (hasta 5 a 8 segundas).
- Los cálculos electrotécnicos son basados en solo una fuente: la documentación de la empresa. No tener fuentes cruzadas puede ser peligroso.
- La elección del equipamiento (los módulos como los inversores) es limitada.
- El geocoding no es muy competente. Sólo conoce las grandes ciudades.

Se ha preguntado qué se debería hacer para mejorar la página aún más.

Para ir adelante,

Además de solucionar estos dichos puntos negativos, se tendrá mejorar varias cosas:

- Rellenar las otras pestanas 'Self-consumption' y 'Pump' con los cálculos que corresponden.
- Pasar la página en internet con un nombre de dominio (.com o .es) y/o enlazarla con la página de TECHNOSUN que ya existe.
- Ampliar las bases de datos. Ahora tenemos solamente una selección de los módulos e inversores que TECHNOSUN utiliza. En el futuro, seria bien tener una base de datos mucho más completo. Principalmente, si queremos abrir la página a mucho usuario, tener una amplia base de datos será importante.
- Rehacer la presentación de la página. Se ha hecho lo posible para hacerla clara y sencilla de utilización, pero todavía se puede mejorar el funcionamiento y cambiar la paleta de colores para hacerla más juvenil y atractiva.
- Cambiar el geocoding de Leaflet por un geocoding más potente como Nominatim que es un geocoding basado en el lenguaje Python. Nominatim se basa en los datos de OpenStreetMap. También gratis y en código abierto. Se tendría que agregarlo en la parte Python del proyecto y llamarlo desde el JavaScript.

Aplicaciones

Esta página web ya sirve a los técnicos de la empresa TECHNOSUN. Por ejemplo, en el servicio al cliente, el técnico, mientras habla por teléfono, puede introducir los datos del cliente en la página para tener un resultado inmediato sobre las capacidades del dispositivo del cliente.

Esta página permite salvar mucho tiempo de cálculo y evitar los errores de cálculo y descuidos. Entonces permite ganar una gran seguridad y ganar en eficiencia en general. La página debería permitir ser más competitivo y potente que las empresas que no tienen este tipo de herramienta.

índice de figuras

Índice de tabla

índice

Introducción

La empresa TECHNOSUN

Figura 7: Logo de TECHNOSUN

“En una época en la que apenas había empresas europeas en el mercado de la energía solar fotovoltaica, Techno Sun se introdujo en el sector y desarrolló la solidez y la calidad de servicio que la han mantenido como líder en la distribución de material solar fotovoltaico especializado.

Techno Sun fue fundada en 1976 por Antonio Ramos Beneyto, director general de la empresa desde sus orígenes hasta la actualidad. Centrándose en sus inicios en las aplicaciones solares para proporcionar energía autónoma allí donde la red no llegaba, Techno Sun adquirió la experiencia y el conocimiento para ayudar a los profesionales emergentes a llevar la electricidad allí donde se necesitaba, y desarrollar una larga trayectoria basada en la seriedad y profesionalidad características de la empresa, que le permitieron acometer con garantía para sus clientes el floreciente mercado de la conexión a red al más alto nivel.

Como distribuidor de Techno Sun, ha sido seleccionado por los más importantes fabricantes del sector, poniendo a disposición de sus clientes una gama de módulos fotovoltaicos de primer nivel de fabricantes como Kyocera, Panasonic o Kaneka, buscando siempre la máxima calidad y los mejores fabricantes japoneses, líderes en la producción de las tecnologías solares más eficientes. Gracias a ello, Techno Sun ha podido facilitar a sus clientes el acceso a las tres tecnologías más relevantes del mercado (silicio policristalino, monocristalino y amorfo).

Ante el crecimiento exponencial del mercado y la necesidad de sus clientes de disponer de un mayor volumen de producción y rango de costes, Techno Sun amplió su catálogo con fabricantes europeos como REC y asiáticos como Suntech Power, siempre con los niveles de calidad que Techno Sun ha mantenido en su trayectoria. Asimismo, como distribuidor de inversores fotovoltaicos también distribuye los mayores fabricantes del mercado como Xantrex (ahora parte del gran grupo Schneider Electric, y distribuido por Techno Sun desde sus orígenes como Trace Engineering y Heart Interface), SMA, Danfoss o Sunways, además de una gama de pequeños y medianos bombeos tanto eólicos como solares.

Con un enfoque europeo desde sus inicios, Techno Sun destaca como líder en España y una creciente presencia en Italia, Alemania, Francia e Inglaterra, además de actuar puntualmente en África, Arabia Saudí y Latinoamérica. Como distribuidores mayoristas, el mercado de Techno Sun ha sido siempre la distribución mayorista, la venta a instaladores, arquitectos, grandes superficies y grandes empresas de producción eléctrica, con más de 30mW instalados por nuestros clientes gracias a nuestra red comercial.

Hoy en día, Techno Sun destaca como líder en calidad y servicio en la distribución especializada, y una fuerte expansión y refuerzo en los mercados históricos y emergentes, para seguir ofreciendo el más alto nivel en el sector solar fotovoltaico europeo.” About Us – TECHNOSUN

Objeto del proyecto

Así que, TECHNOSUN es una empresa que vende equipos fotovoltaicos de manera internacional. Vende principalmente paneles solares e inversores fotovoltaicos. Pero no solamente se dedica a la venta sino a la instalación y al mantenimiento de equipos solares también.

Durante la instalación, los técnicos de TECHNOSUN tienen que comprobar muchas condiciones para que las instalaciones fotovoltaicas sean lo más rentables posible. Estas condiciones se basan en muchos parámetros que son tanto internos como externos a los sistemas. Aquí es donde se puede intervenir. Esta larga, difícil y tediosa tarea, que son los numerosos cálculos energéticos de tensión y/o corriente, se realizará de forma automática y en un tiempo considerablemente reducido.

La idea general de este proyecto es crear una plataforma web que ayude a configurar la instalación de equipos solares. Esta plataforma servirá, en un primer tiempo, a los técnicos de TECHNOSUN, y luego, si todo funciona perfectamente, la plataforma podrá ser abierta a todo el mundo.

Para ayudar a la empresa correctamente, tengo que cumplir ciertos criterios que la empresa me da. Estos criterios me permitirán definir los objetivos de mi proyecto. Serán a la vez los límites de la obra y el pliego de condiciones que hay que respetar.

Objetivos

Construcción de los objetivos

Visor cartográfico

El primer paso para instalar el equipo es localizar el lugar donde se instalará. Es muy importante conocer las condiciones meteorológicas y climáticas. Un panel solar instalado en zonas lluviosas o nubladas no producirá la misma cantidad de energía que el mismo equipo en una zona soleada. Del mismo modo, la latitud tiene un gran impacto en la eficiencia del equipo. Por lo tanto, para satisfacer esta necesidad, vamos a crear un mapa interactivo donde se puede seleccionar fácilmente la ubicación del panel solar deseado.

Además de la información anterior, en un futuro próximo querremos elegir en qué fachada o pendiente del tejado queremos instalar el sistema. El objetivo final sería alcanzar una precisión tal que se pueda pulsar sobre la pendiente de un tejado para seleccionar los parámetros correspondientes con la misma precisión. Por ello, también será necesario implementar las ortofotos correspondientes.

Por lo tanto, el primer objetivo de este proyecto será:

- Crear un visor cartográfico con ortofotos en el que pueda hacer clic para seleccionar la longitud y la latitud de los equipos a instalar.

Cálculos energéticos

A continuación, las condiciones ambientales deben verificarse mediante numerosos cálculos energéticos. Estos son creados por investigadores expertos en la materia. El objetivo no consistirá en crear o inventar nuevos cálculos, sino en trasladarlos al lenguaje de programación Python. Python es un lenguaje bastante internacional en el ámbito de los lenguajes informáticos. Traducirlos nos permitirá utilizarlos en todo el proyecto sin depender de elementos externos.

El segundo punto será:

- Implementar los cálculos electrotécnicos para los equipos fotovoltaicos en Python.

Obtener datos ambientales precisos

Para seguir con la filosofía de alta precisión, se necesitarán datos ambientales de buena calidad. Es esencial que mantengamos este trabajo riguroso porque queremos que el proyecto sea operativo y cercano a la realidad durante el mayor tiempo posible. Tener datos de baja precisión ya no puede ser útil para los operadores y, por tanto, hará que el proyecto sea inútil.

Una idea sería realizar las mediciones directamente sobre el terreno, pero esto impone una serie de limitaciones muy fuertes. En primer lugar, hay que ir al sitio. O incluso disponer de herramientas precisas y, por tanto, caras, para realizar las mediciones. Por eso, se ha rechazado esta idea. Así que se va a buscar datos en Internet. Pero se va a mantener el mismo nivel de precisión.

Así que tenemos el tercer punto:

- Utilizar datos astronómicos (orientación del sol, …) y meteorológicos (radiación solar, …) precisos.

Listar de todos los equipos

Se empieza a ver el proyecto con claridad. Sin embargo, hay un nuevo problema. La empresa TECHNOSUN tiene muchos diferentes modelos de paneles solares e inversores. Para poder dejar la elección del modelo al operador, se tiene que enumerar todos los equipos con sus propias características. Hacer una base de datos accesible desde Python sería muy ventajoso por varias razones. Unas de ellas serían que es más eficiente y que no perdería datos durante el procesamiento de los cálculos energéticos.

Así que se tiene un punto más:

- Crear una base de datos con el máximo número de paneles solares e inversores posibles (no necesariamente sólo los propuestos por TECHNOSUN).

Un presupuesto reducido

Un punto adicional sería hacer todo el proyecto con un presupuesto reducido. Por el momento, en todos los objetivos definidos, se continua con la filosofía de software gratis. Si se puede alcanzar los objetivos siguiendo esta idea, el proyecto será aún más interesante (tanto para la empresa que para el proyecto mismo). Ser independiente de cualquier elemento externo en proyectos como el mío es un punto fuerte para la empresa.

Por lo tanto, el último punto de los objetivos será:

- Que el proyecto sea exclusivamente gratis (si se puede hacer).

Resumen de los objetivos del proyecto:

- Crear un visor cartográfico con ortofotos en el que pueda hacer clic para seleccionar la longitud y la latitud de los equipos a instalar,
- Implementar los cálculos electrotécnicos para los equipos fotovoltaicos en Python,
- Utilizar datos astronómicos (orientación del sol, …) y meteorológicos (radiación solar, …) precisos,
- Crear una base de datos con el máximo número de paneles solares e inversores posibles (no necesariamente sólo los propuestos por TECHNOSUN),
- Que el proyecto sea exclusivamente gratis.

Datos

- Mapa de código abierto de Leaflet.

“Leaflet es la principal biblioteca JavaScript de código abierto para mapas interactivos adaptados a dispositivos móviles. Con un peso de sólo 39 KB de JavaScript, tiene todas las características de mapeo que la mayoría de los desarrolladores necesitan.

Leaflet se ha diseñado pensando en la sencillez, el rendimiento y la facilidad de uso. Funciona eficazmente en todas las plataformas principales de escritorio y móviles, puede ampliarse con muchos plugins, tiene una API bonita, fácil de usar y bien documentada y un código fuente sencillo y legible al que es un placer contribuir.” - https://leafletjs.com/

Figura 8: Ejemplo de mapa Leaflet

- Base de datos de los paneles fotovoltaicos y de datos de los inversores.

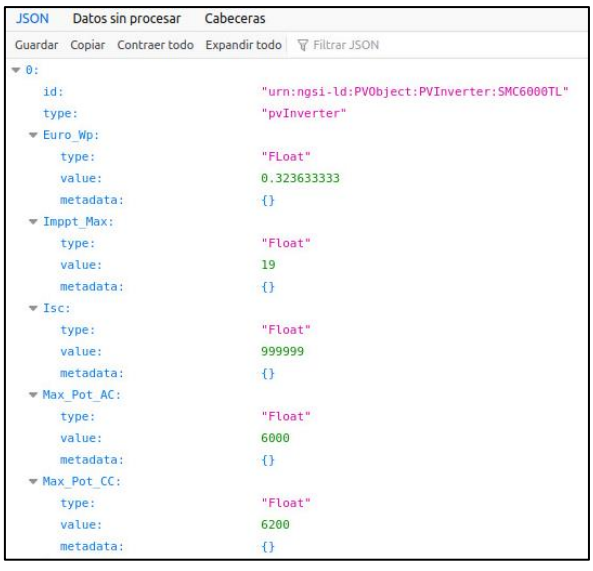
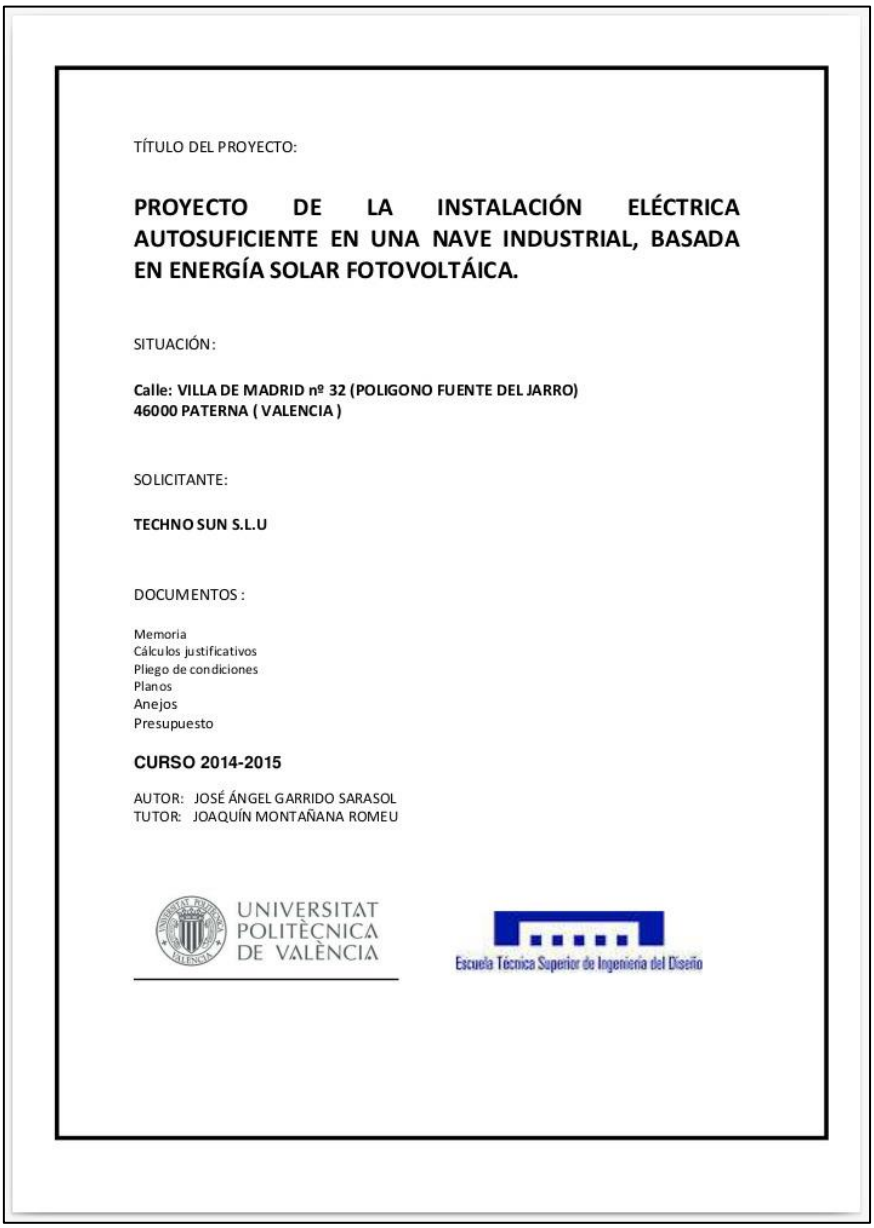


Figura 9: Base de datos

- “Proyecto de la instalación eléctrica autosuficiente en una nave industrial, basada en energía solar fotovoltaica.” – Autor: José Angel Garrido Sarasol – Fecha: 2015.



TÍTULO DEL PROYECTO:

PROYECTO DE LA INSTALACIÓN ELÉCTRICA AUTOSUFICIENTE EN UNA NAVE INDUSTRIAL, BASADA EN ENERGÍA SOLAR FOTOVOLTÁICA.

SITUACIÓN:

Calle: VILLA DE MADRID nº 32 (POLIGONO FUENTE DEL JARRO)
46000 PATERNA (VALENCIA)

SOLICITANTE:

TECHNO SUN S.L.U

DOCUMENTOS :

Memoria
Cálculos justificativos
Pliego de condiciones
Planos
Anejos
Presupuesto

CURSO 2014-2015

AUTOR: JOSÉ ÁNGEL GARRIDO SARASOL
TUTOR: JOAQUÍN MONTAÑANA ROMEU

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Figura 10: Página de portada del proyecto

Título del proyecto: Proyecto de la instalación eléctrica autosuficiente en una nave industrial, basada en energía solar fotovoltaica.

Situación: Calle Villa de Madrid n°32 46000 PATERNA

Solicitante: TECHNOSUN S.L.U

Documentos: Memoria, cálculos justificativos, Pliego de condiciones, Planos, Anejo, Presupuesto

CURSO 2014-2015

Autor: José Ángel GARRIDO SARASOL

Tutor: Joaquín MONTAÑANA ROMEU

- Plugin ESRI Leaflet.

"Un conjunto de herramientas para utilizar los servicios de ArcGIS con Leaflet. Soporte para servicios de mapas, capas de características, mosaicos de ArcGIS Online y más." - http://esri.github.io/esri-leaflet/

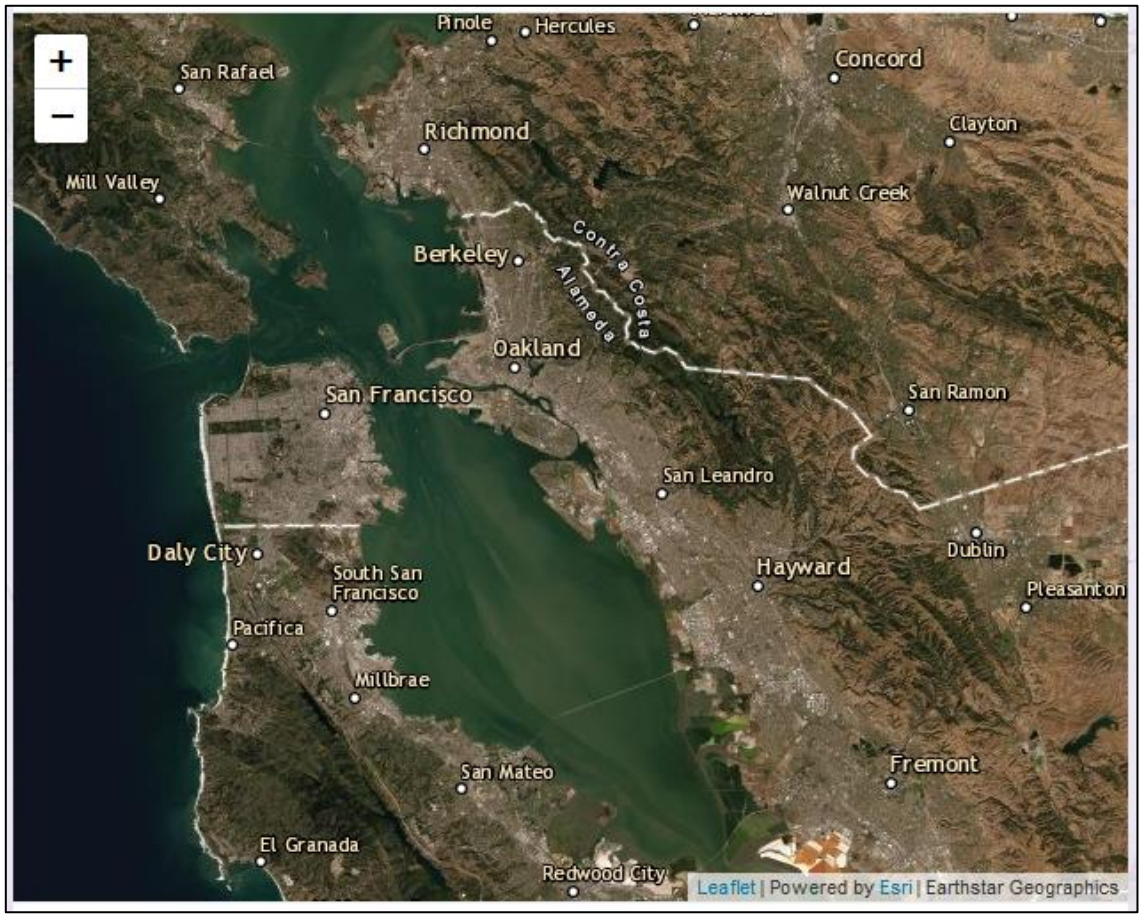


Figura 11: Ejemplo de mapa Leaflet con el plugin ERSI

- Web de la empresa TECHNOSUN

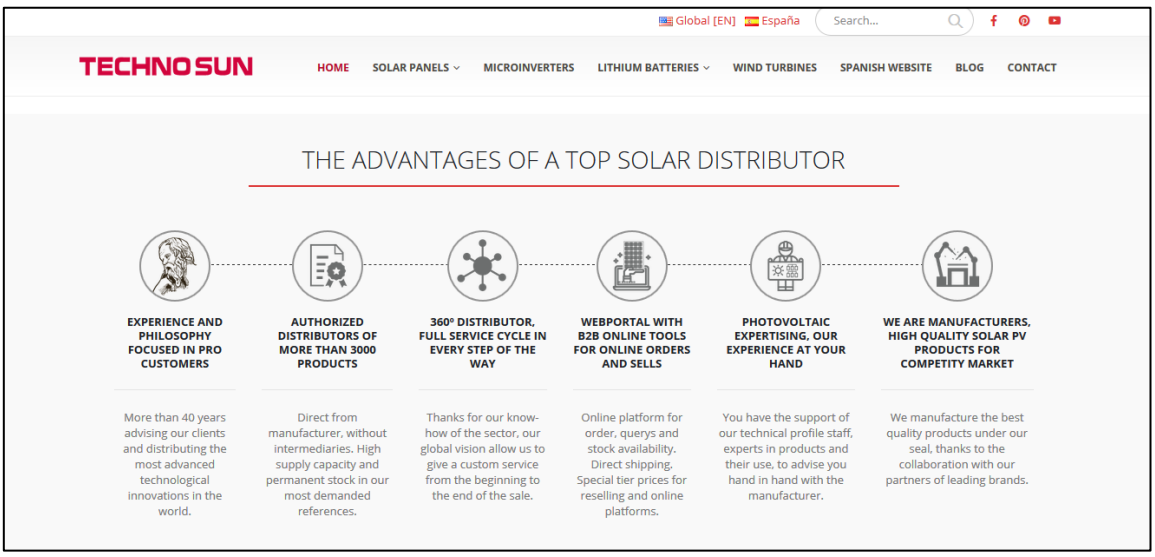


Figura 12: Pagina web de la empresa TECHNOSUN

Metodología

Desarrollo web

Para desarrollar este proyecto, se ha seguido los objetivos establecidos en la sección Objetivos:

- Crear un visor cartográfico con ortofotos en el que pueda hacer clic para seleccionar la longitud y la latitud de los equipos a instalar,
- Implementar los cálculos electrotécnicos para los equipos fotovoltaicos en Python,
- Utilizar datos astronómicos (orientación del sol, …) y meteorológicos (radiación solar, …) precisos,
- Crear una base de datos con el máximo número de paneles solares e inversores posibles (no necesariamente sólo los propuestos por TECHNOSUN),
- Que el proyecto sea exclusivamente gratis (si se puede hacer).

Estos objetivos se han creado a partir de las demandas de la empresa.

En un proyecto de construcción de página web, se puede separar en dos partes: el front-end y el back-end. Así que, para desarrollarlo con claridad, también se dividirá el proyecto en dos.

¿Qué es el desarrollo del front-end?

El front-end se construye utilizando una combinación de tecnologías como Hypertext Markup Language (HTML), JavaScript (JS) y Cascading Style Sheets (CSS). Los desarrolladores de front-end diseñan y construyen los elementos de la experiencia del usuario en la página web o la aplicación, incluidos los botones, los menús, las páginas, los enlaces y los gráficos, entre otros.

HTML: el Hypertext Markup Language es el núcleo de un sitio web, proporcionando el diseño y la funcionalidad general. La versión más reciente se publicó a finales de 2017 y se conoce como HTML5.2. La versión actualizada incluye más herramientas dirigidas a los desarrolladores de aplicaciones web, así como ajustes realizados para mejorar la interoperabilidad.

CSS: las Cascading Style Sheets en ofrecen a los desarrolladores una forma flexible y precisa de crear diseños de sitios web atractivos e interactivos.

JavaScript: este lenguaje basado en eventos es útil para crear elementos dinámicos en páginas web HTML estáticas. Permite a los desarrolladores acceder a elementos separados de la página HTML principal, así como responder a eventos del lado del servidor.

¿Qué es el desarrollo del back-end?

El back-end, también llamado lado del servidor está formado por el servidor que proporciona datos a petición, la aplicación que los canaliza y la base de datos que organiza la información. Por ejemplo, cuando un cliente busca zapatos en un sitio web, está interactuando con el front-end.

Después de seleccionar el artículo que quiere, ponerlo en la cesta de la compra y autorizarla, la información se guarda en la base de datos que reside en el servidor.

Unos días después, cuando el cliente comprueba el estado de su entrega, el servidor extrae la información pertinente, la actualiza con los datos de seguimiento y la presenta a través del front-end.

Herramientas de back-end

La principal preocupación de los desarrolladores de back-end es crear aplicaciones que puedan encontrar y entregar datos al front-end. Muchos de ellos utilizan bases de datos fiables de nivel empresarial como Oracle, Teradata, Microsoft SQL Server, IBM DB2, EnterpriseDB y SAP Sybase ASE. También hay otras bases de datos populares como MySQL, NoSQL y PostgreSQL. Hay una gran variedad de marcos y lenguajes utilizados para codificar la aplicación, como Ruby on Rails, Java, C++/C/C#, Python y PHP.

Esqueleto del proyecto

Tras esta descomposición, se obtiene un proyecto de la siguiente forma:

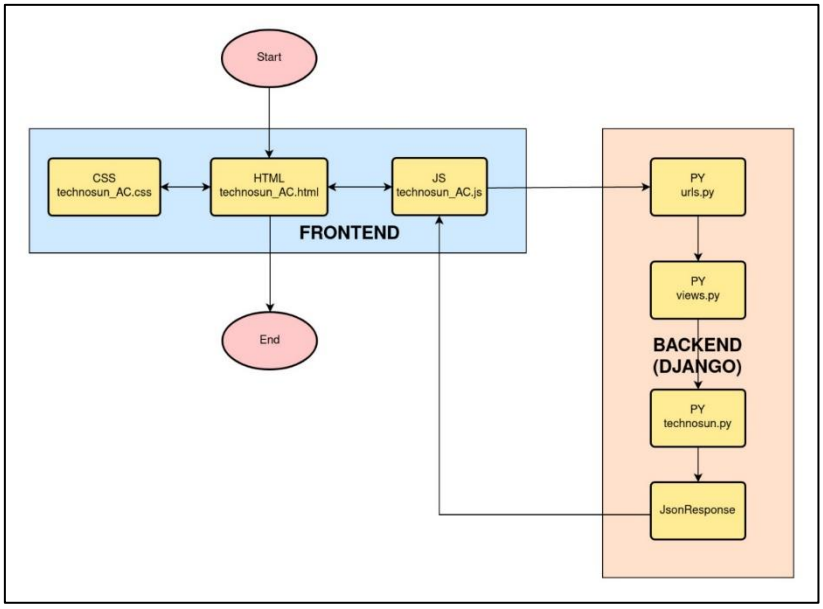


Figura 13: Esqueleto del proyecto

Conforme a la definición del front-end, está compuesto de los tres ficheros HTML, JavaScript y CSS. También según su definición, el back-end está compuesto del servidor Django y de la librería hecho en Python. Es en la próxima parte, se creará la librería Python.

Se podrá ver en detalle las funciones para entender el flujo de datos de entrada a través del proyecto.

Desarrollo del proyecto

Datos de entrada que se tiene: Longitud, Latitud, Nombre del módulo y nombre del inversor. Con estos cuatros datos, se tendrá que lograr al número máximo y mínimo de módulos por ramal.

Como se tiene solamente estos cuatros datos de entrada, será más practico trabajar con Python orientado a objetos. Así que se utilizará el objecto “self” para todos los parámetros de entrada. Entonces, al principio del código, se debe definir el objecto self = (latitud, longitud, datos del módulo, datos del inversor).

Para empezar, se tratará de crear una función independiente, que no tiene impacto en el curso del proyecto. Esta función servirá de ejemplo para todas las demás funciones. Es a partir de las coordenadas que la función calculate_altitude(self) emitirá la altitud que corresponde:

El funcionamiento de la función es simple. Estos tres pasos lo resumen:

- llamar a la API de OpenTopoData, con las coordenadas como argumento, y guarda la salida en un parametro,
- seleccionar el valor que corresponde a la altitud (aquí, se llama ‘elevation’),
- Devolverla.

Para comprobar si la función calculate_altitude funciona, se necesite crear un visor y el enlace entre ellos. La siguiente parte será dedicada a la creación del visor.

Visor cartográfico

Hay algunas condiciones que se debe respectar para la creación de este visor cartográfico. Estas condiciones son:

- Utilizar ortofotos,
- Quedar con la filosofía de software gratis,
- Que sea sencillo a utilizar.

Para cumplir a las dos últimas condiciones, se ha elegido el mapa de Leaflet que esta de código abierto. También, debido a su popularidad, está sencillo a utilizar.

Figura 14: Ejemplo de mapa Leaflet con el servicio de OpenStreetMap

Luego, para obtener ortofotos, se ha añadido el plugin “Ersi Leaflet” que es un conjunto de herramientas para utilizar los servicios de ArcGIS con Leaflet. Y dentro de este conjunto de herramientas, se puede encontrar las ortofotos que necesitamos:

Figura 15: Ejemplo de mapa Leaflet con el plugin “Ersi Leaflet”

Cuando se amplía la imagen, se puede ver los tejados de los edificios:

Figura 16: Zoom en Valencia para comprobar que los tejados se ven

El visor cartográfico está establecido en los tres ficheros que componen el Frontend:

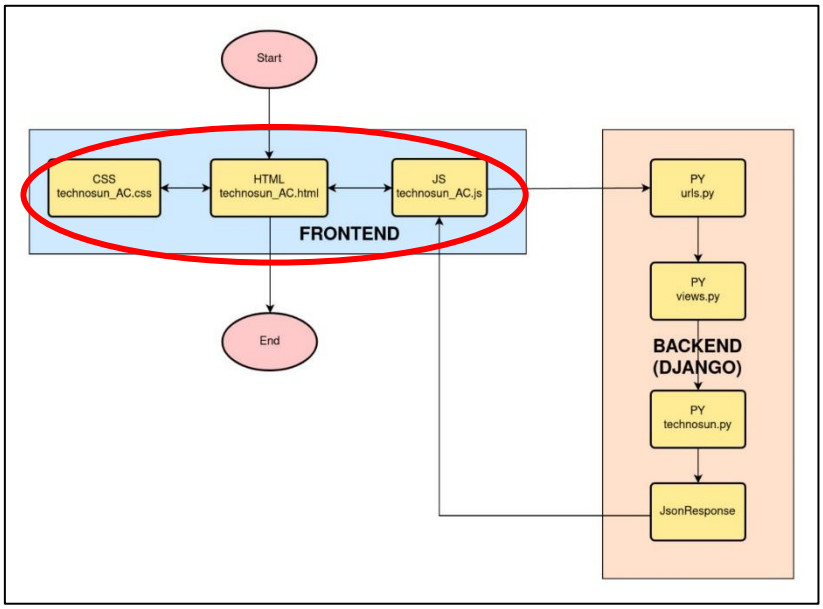


Figura 17: Flujograma general (parte visor)

Una vez que el visor adecuado está operativo, se puede empezar a construir el enlace entre ese visor y la función de Python calculate_altitude que se quiere probar.

Así que, para pasar las coordenadas, voy a utilizar un framework de lenguaje Python. Se ha elegido porque ya se tenía experiencia y porque ha demostrado ser eficiente, además que continua con la filosofía de software gratis. Es Django, el que lleva el peso de poder actuar directamente con los métodos implementados en Python.

Django se define como: "*Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.*"

Incorporación de Django

Para empezar la parte de Django, se cree un nuevo proyecto. Este será llamado technosun.

```
$ django-admin startproject technosun
```

Esta línea de código crea los ficheros siguientes:

```
technosun/
    manage.py
    technosun/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

Estos archivos son:

- El directorio externo technosun/ es el contenedor para el proyecto. Su nombre no le importa a Django.
- manage.py contene unas líneas de comandos que te permite encender el servidor Django servidor con las líneas de comenda siguientes: `$ cd technosun` `$ python3 manage.py runserver`
- El directorio interno technosun/ es el paquete Python real para el proyecto. Su nombre es el nombre del paquete Python que necesitarás para importar cualquier cosa dentro de él (por ejemplo, technosun.urls).
- technosun/__init__.py es un archivo vacío que indica a Python que este directorio debe ser considerado un paquete de Python.
- technosun/settings.py sirve para los ajustes y la configuración del proyecto Django.
- technosun/urls.py será el fichero donde declararemos los URLs del proyecto Django. Corresponde a una "tabla de contenidos".
- technosun/asgi.py es un punto de entrada para que los servidores web compatibles con ASGI.
- technosun/wsgi.py es un punto de entrada para que los servidores web compatibles con WSGI.

Así que ahora, en la carpeta technosun, junto a todos los demás códigos Python, se tiene que agregar una carpeta llamada pyCode. La librería de funciones de Python se sitúa en esta carpeta. Pero de momento, en la librería, sólo hay la función calcular_altitud:

```
technosun/
    manage.py
    technosun/
        __init__.py
                pyCode/
                    technosun.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

En el fichero Views.py, está definido la clase getAltitude(View):

```python
class getAltitude(View):
    def get(self, request, parameters):
        params = parameters.split(",")
        obj = tec.TechnosunIsolated(params[0], params[1], params[2], params[3])
        obj.calculate_altitude()
        r = obj.altitude
        return JsonResponse({"ok": "true", "message": "This is the altitude.", "data": r})
```

Figura 18: Función getAltitude en el fichero View.py

En la primera línea, se recupera y separa los parámetros de entrada. Se tiene que recordar que estos parámetros son Longitud, Latitud, Nombre del módulo, Nombre del inversor. En la segunda línea, está definido el parámetro self (aquí se llama “obj”) que será utilizado en la cuarta línea con la función “altitude”. Este término “altitude” se refiere a la función basica que hemos hecho para recoger la altitud a partir de coordenadas. La última línea permite devolver la respuesta de formato JSON. La “r” corresponde a la altitud, en metros.

En el fichero Urls.py, esta añadido la línea siguiente:

```
urlpatterns = [path('get_altitude/<parameters>/',
                                views.getAltitude.as_view(), name='getAltitude')]
```

En esta línea de código, el get_altitude/<parameters>/ es el nombre que la función debería tener en el fichero JavaScript. Sera a partir de este nombre que se podrá recuperar la altitud. Por lo tanto, se necesita escribir una nueva función en el fichero JavaScript para recuperarla:

```javascript
function getAltitude(technosun) {
  // This function ask the altitude to python files through Django
  parameters =
    technosun[0].toString() + // Latitude
    "," +
    technosun[1].toString() + // Longitude
    "," +
    technosun[2] + // TrinaSolarTSM400DE15MII
    "," +
    technosun[3]; // SUNNYBOY3000TL
  $.ajax({
    url: URL_DJANGO_API + "get_altitude/" + parameters + "/", //url where the data is sent
    success: function (data) {
      //callback function. It will wait for the server answer
      alt = (Math.round(data.data * 1000) / 1000).toString();
      let textBoxAlt = document.getElementById("text-box-alt");
      textBoxAlt.innerHTML = alt;
      if (alt == 0) {
        textBoxAlt.innerHTML = "Out of range...";
      }
    },
    error: function (err) {
      console.log(err); //in case of error logs the error in the console
    },
  });
}
```

Figura 19: Función getAltitude en el fichero technosun.JS

Una vez añadida la función en la librería y modificados los códigos Urls.py, View.py y JavaScript, la ruta de datos completa está definida. Se puede representar de la siguiente manera:

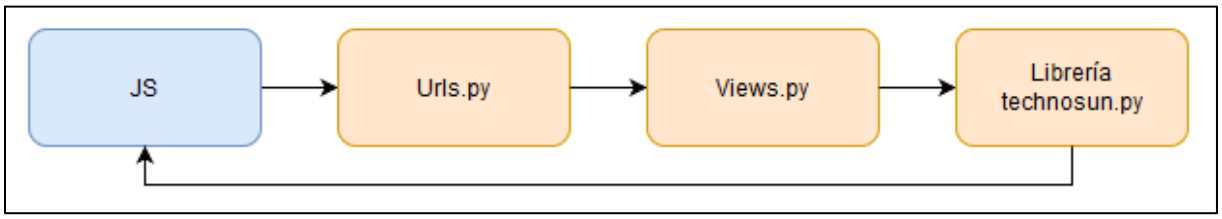


Figura 20: El funcionamiento del back-end que se comunica con el archivo JS

Ahora que está bien implementado, se realizó la siguiente prueba:

- Recoger coordenadas desde el mapa,
- Pasarlas a las librerías Python mediante la intervención de Django,
- Inicialmente, se ha utilizado la función para determinar la altitud a partir de las coordenadas,
- Devolver la altitud al código JavaScript pasando por la capa Django,
- Con el fin de visualizar los datos en la pantalla.

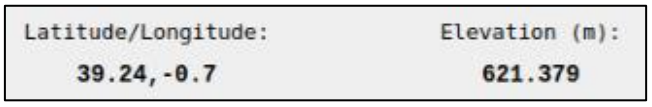


Figura 21: Primera prueba de viaje de ida y vuelta entre el código Python y JavaScript

Este paso ha sido la clave para el posterior desarrollo de todas las consultas que se realizan entre las tres piezas software (JavaScript, Django y Python).

Ahora que se puede copiar el funcionamiento de esta función operativa, se tiene que continuar construyendo las demás funciones del proyecto. Con el fin de utilizar las fórmulas de energía en la página web. Se va implementando funciones más complejas para continuar. Por ejemplo, la siguiente función que será agregada será una función que devolverá la temperatura máxima, mínima y media de un punto con coordenadas. Para lograr a eso, como no se puede ir a medir la temperatura in situ, se tiene que encontrarla en otro sitio, como Internet. Sin embargo, en Internet hay muchos datos que no sean fiable. Por eso, para el proyecto, se recupera los datos fiables de la web de la Comunidad Europea. Por lo tanto, la herramienta PVGIS de la Comunidad Europea será utilizada. ¿Qué es PVGIS?

PVGIS es un sitio web que ofrece información sobre la radiación solar y el rendimiento de los sistemas fotovoltaicos. Puede utilizar PVGIS para calcular la cantidad de energía que puede obtener de diferentes tipos de sistemas fotovoltaicos en casi cualquier lugar del mundo.

La interfaz web de PVGIS tiene el siguiente aspecto:

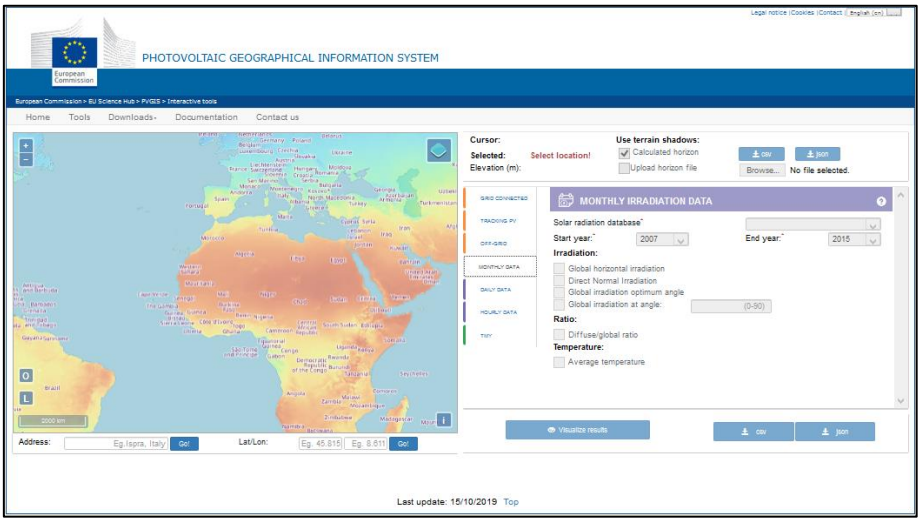


Figura 22: Interfaz web de PVGIS

Se puede elegir un lugar de tres maneras diferentes:

- Haciendo clic en el mapa (puede ampliarlo y desplazarse de la forma habitual).
- Introduciendo el nombre de un lugar (ciudad, calle) en el campo "Dirección" debajo del mapa.
- Escribiendo las coordenadas de latitud y longitud en las casillas situadas debajo del mapa.

Y después, se puede pedir diferentes cálculos a PVGIS, como:

- **Rendimiento de la energía fotovoltaica conectada a la red**. Aquí se puede calcular la producción media de energía a largo plazo de los sistemas fotovoltaicos que están conectados a la red eléctrica, de modo que la energía producida puede utilizarse localmente o enviarse a la red. Esto funciona para los sistemas FV fijos, en los que los módulos FV están montados en una posición fija, ya sea en un bastidor independiente o en un edificio.
- **Rendimiento de los sistemas fotovoltaicos con seguimiento**. Aquí puede calcular la producción energética media a largo plazo de los sistemas fotovoltaicos conectados a la red en los que los módulos se colocan en un montaje con seguimiento solar para que los módulos reciban más luz solar. Puede elegir entre varios tipos de sistemas de seguimiento.
- **Rendimiento de los sistemas fotovoltaicos sin conexión a la red**. Esta herramienta le permite realizar cálculos sobre los sistemas fotovoltaicos que no están conectados a la red eléctrica pero que utilizan baterías como almacenamiento de energía.
- **Radiación mensual**. Con esta herramienta puede obtener los datos mensuales de radiación y temperatura media de cada mes en un rango de años.
- **Radiación diaria**. Aquí puede calcular la irradiación solar y la temperatura media durante el día para un día medio de cada mes.
- **Radiación horaria**. Esta herramienta le permite descargar una serie temporal de valores de radiación solar y/o potencia fotovoltaica por hora.
- **TMY**. Herramienta para generar datos del Año Meteorológico Típico (TMY) de radiación solar, temperatura y otros datos meteorológicos, utilizados en muchos campos, por ejemplo, en el cálculo del rendimiento energético de los edificios.

Así es como funciona la herramienta PVGIS. Una ventaja de PVGIS es que estos mismos cálculos se pueden estar incorporado en Python. Para hacer eso, se tiene que comunicar con las APIs que PVGIS propone. ¿Qué es una API? Una API es el acrónimo de Application Programming Interface (o interfaz de programación de aplicaciones). Las API me permiten comunicar con otros productos y servicios sin tener que saber cómo están implementados. Lo cual es muy útil para el proyecto porque se va a importar las temperaturas de esta manera.

APIs

Ahora, se necesitan datos medioambientales, como temperatura o la radiación solar, para calcular las condiciones reales de un edificio. Igualmente, se conoce el funcionamiento de las APIs propuestas por la web de Photovoltaic Geographical Information System (PVGIS) (https://ec.europa.eu/jrc/en/pvgis). El próximo paso será de integrarlas a la función de búsqueda de temperatura.

En el enlace, solo se tiene que elegir las coordenadas y los años:

https://re.jrc.ec.europa.eu/api/seriescalc?lat=39.4&lon=-0.4&startyear=2014&endyear=2016

Entonces, al utilizar, se obtiene varias variables ambientales:

- G(i): Global irradiance on the inclined plane (plane of the array) = Irradiación global en el plano inclinado (plano del conjunto) (W/m2)
- H_sun: Sun height (degree) = Altura del sol (grado)
- T2m: 2-m air temperature (degree Celsius) = Temperatura del aire a 2 m (grados Celsius)
- WS10m: 10-m total wind speed (m/s) = Velocidad total del viento de 10 m (m/s)
- Int: 1 means solar radiation values are reconstructed PVGIS = 1 significa que los valores de radiación solar se reconstruyen PVGIS

Para resumer, solo se piden los datos ambientales para el punto con coordenadas (39,4; -0,4) y para los años 2014, 2015 y 2016.

La API que esta utilizada en la función get_altitude funciona de la misma manera. La única diferencia es que llama a OpenTopoData en lugar de PVGIS. El enlace esta similar, solo se tiene que elegir las coordenadas:

https://api.opentopodata.org/v1/eudem25m?locations=39.4,-0.4

Finalmente, con la API de PVGIS, se ha conseguido crear la función calcular_temperatura. La salida de esta función está compuesta de tres variables. Para los años seleccionados, sale el valor máximo de la temperatura, el valor mínimo y el valor medio. Ademas, para tener datos relevantes, se tiene que elegir solamente las temperaturas durante el día. Por eso, se tendría que analizar el campo G(i) y recuperar solamente los datos de temperatura cuando este campo es distinto de cero.

Una vez la función calcular_temperatura bien implementada, se puede continuar escribiendo las funciones. Para calcular la eficiencia de los paneles solares, es necesario calcular sus características en condiciones reales. Para ello, en la siguiente función, será necesario tratar los datos internos de los dispositivos utilizados. TECHNOSUN ya dispone de una base de datos sobre las características de los paneles solares y los inversores. Están en formato PDF (un archivo por dispositivo) y se almacenan en una sola carpeta. Para poder trabajar fácilmente con estos datos, será necesario crear una base de datos que reúna todos los archivos PDF. Una búsqueda manual de cada dato no es factible en el contexto de nuestro proyecto. Por lo tanto, es necesario crear un espacio para almacenar y organizar los datos adecuadamente. Este espacio será la base de datos del proyecto.

Base de datos

Lo que se busca para el proyecto, es una base de datos sencilla y, para quedar con la idea de software gratis, sería mejor si sea gratuita. No es necesario tener una base de datos muy competente porque la empresa sólo tiene pocos dispositivos. Sólo se quiere registrar los módulos solares y los inversores. Tal vez en el futuro se podrá tener más datos. En cualquier caso, las bases de datos de libre acceso serán más que suficientes para los usos que necesita el proyecto.

Por eso se ha optado el framework Fiware.

Figura 23: Logo de Fiware

Este framework de código abierto ofrece un servidor de almacenamiento de datos. La definición de Fiware es la siguiente:

"*FIWARE is an open source initiative defining a universal set of standards for context data management which facilitate the development of Smart Solutions for different domains such as Smart Cities, Smart Industry, Smart Agrifood, and Smart Energy.*"

Una vez que el servidor está encendido, se puede buscar https://127.0.0.1:1026/v2/entities en la barra de búsqueda de cualquier navegador internet. Con eso, se puede ver el contenido de la base de datos. Para el proyecto, será en esta base de datos que se almacenaron las características de los dispositivos solares. Sin embargo, para comunicar con esta plataforma, se debe que utilizar el lenguaje CRUD, el lenguaje de Fiware. Igualmente, se pueden traducir las funciones de Python.

El lenguaje CRUD está compuesto de 4 tipos de acciones: Crear, leer, actualizar y eliminar son las cuatro funciones básicas del almacenamiento persistente. Estas operaciones suelen denominarse con el acrónimo CRUD (Create, Read, Update, Delete = CRUD). Para el proyecto, si se quiere modificar datos del servidor, el hecho de tener pocos datos permite cambios rápidos. Sólo se necesitaron los tres códigos siguientes (Create, Read et Delete):

HTTP Verb	POST	GET	DELETE
/v2/entities	CREAR una nueva entidad y añadirla al contexto.	LEER los datos de la entidad desde el contexto. Esto devolverá datos de múltiples entidades. Los datos pueden ser filtrados.	X
/v2/entities/<entity-id>	CREAR o ACTUALIZAR un atributo de una entidad especificada.	LEER los datos de una entidad especificada. Esto devolverá los datos de una sola entidad. Los datos pueden ser filtrados.	BORRAR una entidad del contexto.

Tabla 1: Funcionamiento del lenguaje CRUD aplicado a Fiware

Crear un elemento

```
def post(data, url):
    '''This function is used to send data to the server'''
    headers = {"Content-Type": "application/json"}
    datat = json.dumps(data)
    end = "/v2/entities"
    # Here is the line where we send data to the server
    r = requests.post(url+end, headers=headers, data=datat)
    return r
```

Figura 24: Función post

Leer un elemento

```
def get(url, iden):
    '''This function is used to read data from the server'''
    end = "/v2/entities/"+str(iden)
    pars = {"options": "keyValue"}
    # Here is the line where we acquire the data
    r = requests.get(url+end)
    return r
```

Figura 25: Función get

Borrar todos los elementos del servidor

```
def deleteAll(url):
    '''This function is used to delete everything in the server'''
    end = "/v2/entities"
    r1 = requests.get(url+end).json()
    for i in range(0, len(r1)):
        url_complete = url+end+"/"+r1[i]["id"]
        # Here is the line where we delete a data, we repeat it for every data in the server
        r = requests.delete(url_complete)
    return requests.get(url+end).json()
```

Figura 26: Función deleteAll

Esta última función, deleteAll, no existía antes. Se tenía que crearla, a partir de la función "Delete" ya existente, para satisfacer las necesidades del proyecto.

Ahora que se sabe cómo utilizar la base de datos, se tiene que crear un código Python para leer los PDF y enviarlos directamente a la base de datos en formato JSON. El formato utilizado es JSON, porque se ha convertido en el formato base de intercambio de datos tanto entre máquinas como entre máquinas y usuarios.

Conversión de los ficheros PAN en JSON

Junto con los archivos PDF, la empresa generó un archivo PAN con los mismos datos. archivos, un archivo PAN, que contiene los mismos datos. Este formato de archivo es muy similar al tipo de archivo TXT. Sabiendo que el archivo PAN no sólo es más detallado, sino también más legible que el archivo PDF. Entonces, se tiene que abrir y leer el archivo PAN en lugar del archivo PDF. A continuación, se creará una nueva funcion para convertir los archivos PAN en formato JSON. Esta función se llamará PANtoJSON y servirá a hacer el cambio de formato:

```
PVObject_=pvModule
  Version=6.77
  Flags=$00500043

  PVObject_Commercial=pvCommercial
    Comment=www.trinasolar.com (China)
    Flags=$0041
    Manufacturer=Trina Solar
    Model=TSM-405DE15M(II)
    DataSource=TSL,2018_12
    YearBeg=2017
    Width=1.004
    Height=2.024
    Depth=0.035
    Weight=26.400
    NPieces=0
    PriceDate=12/04/17 16:06
    Currency=EUR
    Remarks, Count=3
      Str_1=Frame: Aluminum
      Str_2=Structure: Glass-Backsheet
      Str_3=Connections: MC4 Compatible
    End of Remarks=Connections: MC4 Compatible
  End of PVObject pvCommercial

  Technol=mtSiMono
  NCelS=72
  NCelP=2
  NDiode=3
  GRef=1000
  TRef=25.0
  PNom=405.0
  PNomTolLow=0.00
  PNomTolUp=1.40
  Isc=10.520
  Voc=49.20
  Imp=10.000
  Vmp=40.50
  muISC=5.26
  muVocSpec=-116.0
  muPmpReq=-0.370
  RShunt=1550
  Rp_0=5000
  Rp_Exp=30.00
  RSerie=0.267
  Gamma=1.067
  muGamma=-0.0006
  VMaxIEC=1500
  VMaxUL=1000
  Absorb=0.90
  ARev=3.200
  BRev=11.040
  RDiode=0.010
  VRevDiode=-0.70
```

Figura 27: Datos en el fichero PAN

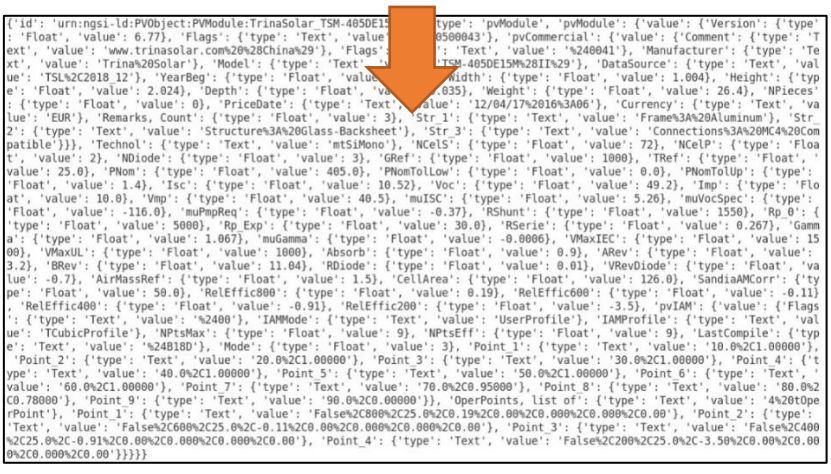


Figura 28: Datos en formato JSON

Aunque los datos aparecen menos ordenados después del procesamiento, en formato JSON, que en el archivo PAN, son mucho más fáciles de leer.

Una vez que se tienen estos datos, se pueden enviarlos a la base de datos con la función “post”. Ahora, los datos se pueden encontrar en la base de datos con este formato:

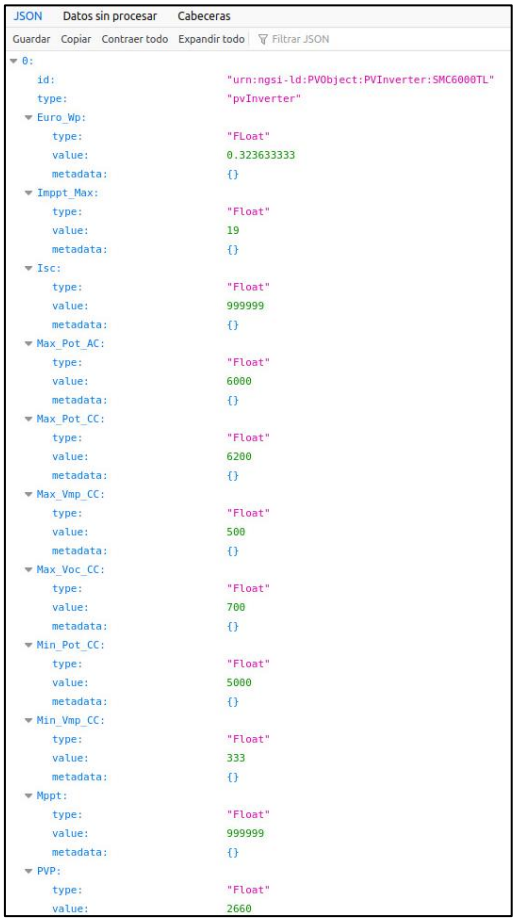


Figura 29: Datos en la base de datos

Ahora que se tiene una base de datos ordenada. Se puede seguir escribiendo las funciones de la librería Python. La próxima función se servirá calcular las características en condiciones reales de los dispositivos. Esta función se llamará: calculate_real_condition_module(self).

Se debe recordar que en ‘self’ se encuentra la latitud, la longitud, el nombre del módulo seleccionado y el nombre del inversor seleccionado. Con estos dos últimos parámetros, se tiene que encontrar los datos que corresponden.

```
# We are browsing the database of panels
line = json.loads(self.__module_data)["pvModule"]['value']
# For the model we have chosen, we are saving the parameters
v_mpp = line['Vmp']['value']
voc = line['Voc']['value']
impp = line['Imp']['value']
isc = line['Isc']['value']
muVocSpec = line['muVocSpec']['value']
muIsc = line['muISC']['value']
```

Figura 30: Primeras líneas de la función calculate_real_condition_module

Con la primera línea, se extrae el JSON almacenado en la variable module_data. En las siguientes líneas, se definen todos los valores que se necesitaron.

```
temp_c_max = Temp_max+1000*(tonc-20)/800
temp_c_min = Temp_min+1000*(tonc-20)/800
self.__v_max = v_mpp + (muVocSpec/voc/10) * (temp_c_min - 25)
self.__impp_max = impp + (muIsc/isc/10) * (temp_c_max - 25)
self.__isc_max = isc + (muIsc/isc/10) * (temp_c_max - 25)
self.__v_mean = v_mpp + (muVocSpec/voc/10) * \
    ((temp_c_max+temp_c_min)/2 - 25)
self.__v_min = v_mpp + (muVocSpec/voc/10) * (temp_c_max - 25)
self.__voc_max = voc + (muVocSpec/voc/10) * (temp_c_min - 25)
```

Figura 31: La última parte del código

Con los términos bien definidos, se puede calcular todos los datos que serán en la página web. La última tarea que va a faltar será de presentar la página web, de manera organizada, para poder mostrar todas las variables que interesan el técnico.

Aspecto de la página web

En primer lugar, se dividirá la página en dos partes: el mapa y los cálculos.

En la primera mitad, se encontrará el logotipo de la empresa TECHNOSUN y, por supuesto, el mapa que se utilizará para recoger las coordenadas.

Figura 32: Parte izquierda de la página web (mapa y logo)

La segunda mitad de la página se utilizará para los cálculos de energía.

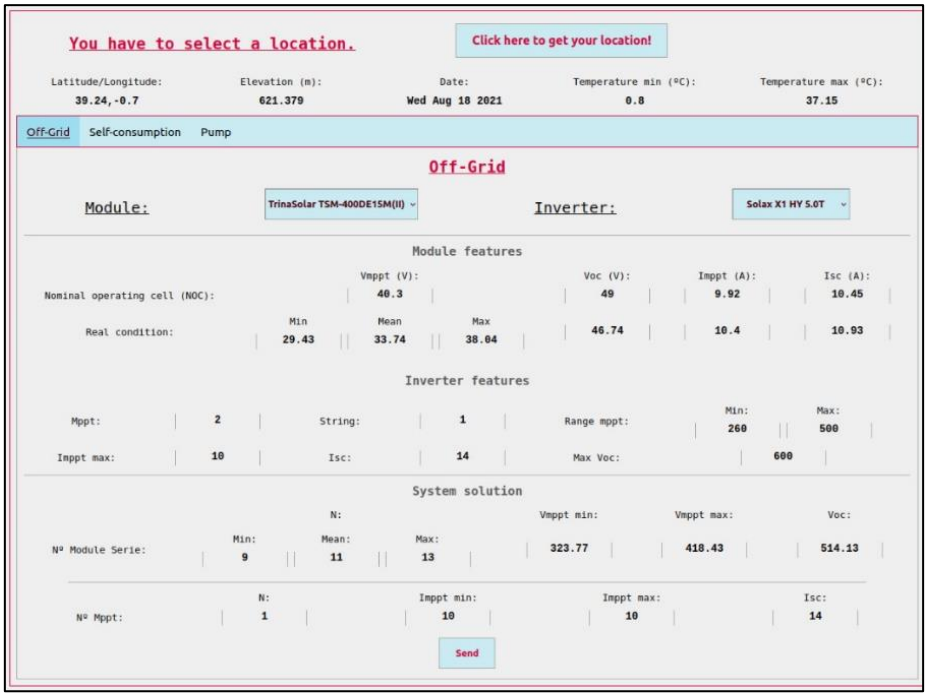


Figura 33: Flujograma general (parte librería Python)

La primera parte corresponderá a los datos externos de los dispositivos. Esta pancarta contendrá las coordenadas, la altitud, la fecha del día, la temperatura máxima y mínima.

Luego se pueden encontrar tres pestañas: Off-grid (que corresponde a los cálculos en Python), Self-consumption y Pump. Las dos últimas pestañas se podrán rellenar durante el desarrollo del software que continuará después de este TFM, en colaboración con la empresa.

Los dos controladores, debajo de las pestañas, permiten elegir un módulo y un inversor dentro de la lista de dispositivos disponibles. Son listas desplegables con la posibilidad de desplazarlas con la rueda del ratón.

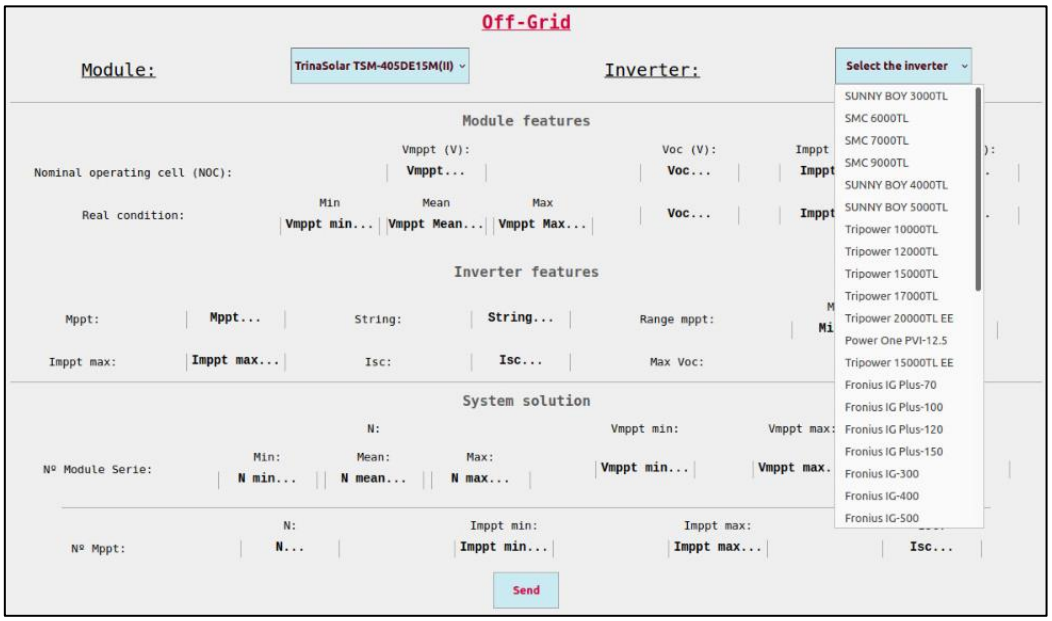


Figura 34: Prueba del geocoding

Una vez seleccionado, el nombre permanece escrito en el controlador. Una vez seleccionada, el nombre queda escrito en el botón. Cuando ambas opciones estén hechas, se debe pulsar el botón "Send" para iniciar los cálculos, y todos los términos aparecerán en sus respectivas casillas.

Ademas de los cálculos, y para facilitar la utilización de la página, se ha añadido varias herramientas a la página.

Agregar herramientas

Un enlace dinámico

El logo TECHNOSUN contiene un enlace que dirige el usuario directamente en la página oficial de TECHNOSUN. Se ha escrito de esta forma:

```
<a href="https://www.technosun.com/"><img src="img/Logo.webp" alt="Logo"/></a>
```

Figura 35: Línea de código para implementar el enlace de la web de TECHNOSUN

El 'href' corresponde al enlace de Internet al que será redirigido el usuario.

El 'src' llama a la imagen 'Logo.webp' que se sitúa en la carpeta 'img'.

Un geocoding con una barra de búsqueda

Para ayudar a los usuarios de la página web, se ha añadido una barra de búsqueda vinculada al geocoding. Esta barra permitirá al usuario buscar un lugar específico y hacer que el mapa lo localice y obtenga las coordenadas. El geocoding es el proceso informático por el que se convierte una dirección física en coordenadas geográficas. Se ha hecho importando un geocódigo existente: el de OpenStreetMap. Esta insertado en mi código JavaScript por comodidad.

Figura 36: Logo de OpenStreetMap

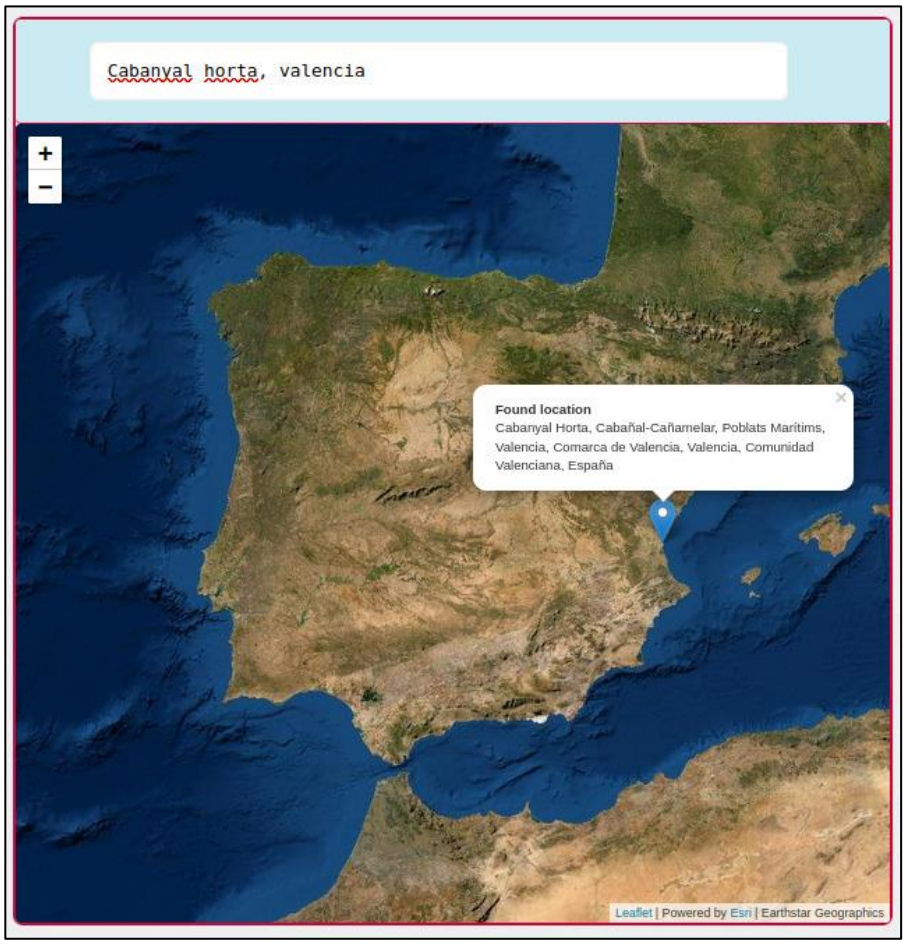


Figura 37: Prueba del geocoding

Una localización automática

El botón de localización que permite seleccionar automáticamente la posición del usuario (también, indica la precisión de la posición). Para utilizar esta función, solo se necesita cliquear en el botón "Click here to get your location" (Haz clic aquí para obtener su ubicación), y aceptar que el navegador le localice. Inmediatamente después, la ubicación se inserta en el mapa y se obtiene los datos correspondientes a las coordenadas.

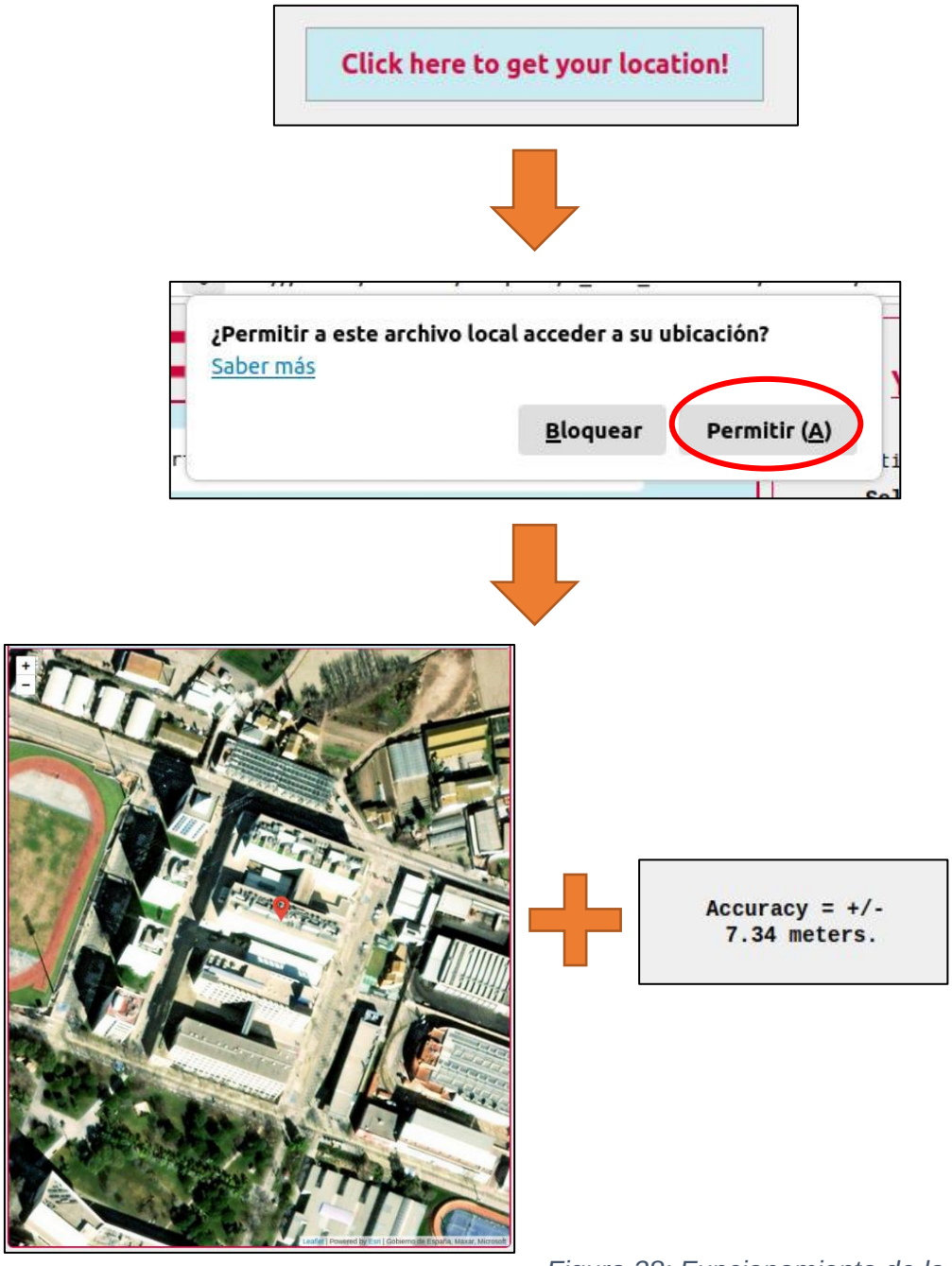


Figura 38: Funcionamiento de la herramienta "Localización automática"

Esta herramienta es muy útil para un técnico que está en el campo y no sabe exactamente dónde se sitúa. Sólo se tendrá que pulsar esta botón para conocer su posición y los datos ambientales.

Botones para ampliar y recudir el mapa

Los botones + y –, arriba a la izquierda del mapa, permiten ampliar y reducir el mapa. Esta acción también se puede hacer con la rueda del ratón.

Figura 39: Zoom y deszoom máximos que se pueden hacer

Una página adaptativa

La página es dinámica. Ella se adapta al tamaño de la ventana. En una pantalla grande o en un teléfono móvil, la página web seguirá siendo utilizable de la misma manera. Igualmente, si se reduce el tamaño del navegador, el mapa cambia de sitio.

Figura 40 y 41: Pagina web con tamaño de pantalla normal (izquierda) y con un tamaño de pantalla reducida (derecha)

Librería de las funciones Python

La librería se sitúa en el fichero con el nombre technosun.py, en la parte Backend:

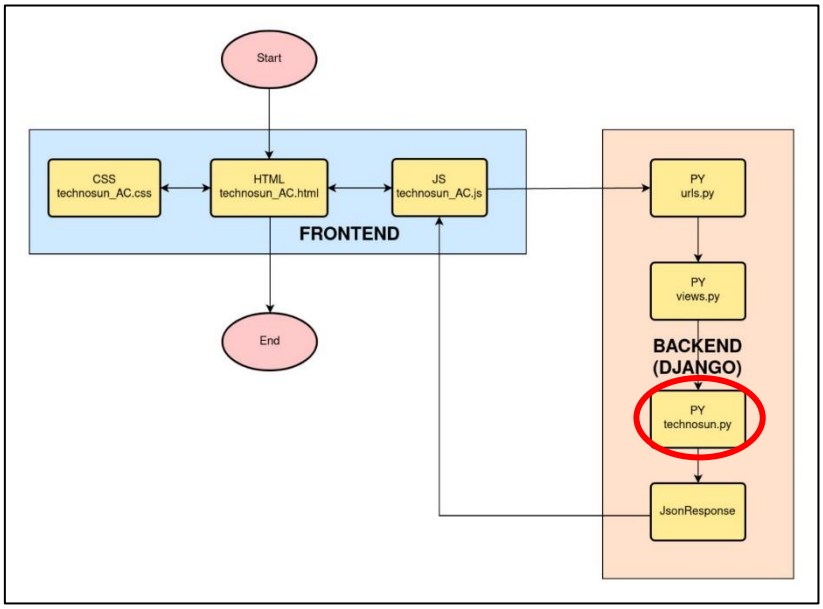


Figura 42: Flujograma general (parte librería Python)

calculate_altitude(self)

Parámetros de entrada: la latitud (float), la longitud (float).

Parámetros de salida: la altitud (float).

Descripción: Utilizando la API de OpenTopoData, la función devuelve la altitud que corresponde a la latitud y la longitud de entrada.

Precisión de desarrollo: Primero, se tiene que llamar a la API con la función “get(URL)” de la librería “requests”. El URL de la API que se utiliza en la función, esta de la forma siguiente: https://api.opentopodata.org/v1/eudem25m?locations=39.4,-0.4. Luego, se debe convertir los datos en formato JSON. Y, para terminar, se tiene que leer el campo ‘elevation’ del apartado ‘results’.

read_module_from_fiware(self)

Parámetros de entrada: el nombre del módulo (string).

Parámetros de salida: los datos del módulo (json).

Descripción: Dentro de la base de datos Fiware, la función busca los datos que corresponden al nombre del módulo. Y les devuelven.

Precisión de desarrollo: De la misma manera que en la función calculate_altitude, read_module_from_fiware utiliza la función “get(URL)” para recuperar los datos de la base de datos. El nombre del módulo será escrito dentro del enlace URL. En el fin de devolver los datos en formato JSON, se tienen que convertirlas.

read_inverter_from_fiware(self)

Parámetros de entrada: el nombre del inversor (string).

Parámetros de salida: los datos del inversor (json).

Descripción: Dentro de la base de datos Fiware, la función busca los datos que corresponden al nombre del inversor. Y les devuelven.

Precisión de desarrollo: Esta función tiene exactamente el mismo funcionamiento que la función read_module_from_fiware pero con los invesores.

calculate_temperature(self)

Parámetros de entrada: la latitud (float), la longitud (float).

Parámetros de salida: las temperaturas medias de cada mes, las temperaturas máximas de cada mes, las temperaturas mínimas de cada mes.

Descripción: A partir de la posición, la función devuelve la temperatura media, máxima y mínima de cada mes, solo con medidas tomadas durante el día. Es decir, durante la noche, la radiación solar esta nula. Entonces, hemos cogido las temperaturas únicamente cuando la radiación solar no está nula.

Precisión de desarrollo: Inicialmente, con la función “get(URL)” de la librería “requests”, se llama a la API de PVGIS con la longitud y la latitud como parámetros. Para una precisión máxima, se debe utilizar los datos lo más reciente. Aquí son los datos el año 2016. Al llamar la API, ella devuelve:

- G: Global irradiance on the inclined plane (plane of the array) (W/m2),
- H_sun: Sun height (degree),
- T2m: 2-m air temperature (degree Celsius),
- WS10m: 10-m total wind speed (m/s),
- Int: 1 means solar radiation values are reconstructed PVGIS (c).

La variable que es interesante para el proyecto es la temperatura. Pero se necesita recoger los datos de temperatura solamente durante el día. Entonces, en el código, se necesita presicar que se tiene que guardar la variable T2m solo si H_sun es diferente de 0. Así, se puede calcular la promedia, el valor máximo y el valor mínimo de cada mes. Para calcular la promedia, se tiene que hacer cuidado y dividir por el buen numero de día (con un contador se puede evitar contar los anos bisiestos).

calculate_real_condition_module(self)

Parámetros de entrada: la latitud (float), la longitud (float), el nombre del módulo (string).

Parámetros de salida: las temperaturas medias de cada mes, las temperaturas máximas de cada mes, las temperaturas mínimas de cada mes.

Descripción: A partir de los datos de entrada, primero, la función recoge las temperaturas máximas y minimas calculadas con la función calculate_temperature. Segundo, carga el json creado con la función read_module_from_fiware. Y por fin, calcula las diferentes variables como la tensión máxima/mínima y el corriente y otras…

Precisión de desarrollo: Se recupera la temperatura máxima y mínima y los datos del módulo. Con estas variables, se calcula las temperaturas max/min de la célula:

$$Temperatura\ máxima\ de\ la\ célula = Temperatura\ máxima + 1000 * \frac{NOCT-20}{800} \quad (1)$$

La NOCT (Nominal Operating Cell Temperature) se define como la temperatura alcanzada por las células en circuito abierto de un módulo en las condiciones que se indican a continuación:

- Irradiación en la superficie de la célula = 800 W/m2,
- Temperatura del aire = 20°C,
- Velocidad del viento = 1 m/s,
- Montaje = parte trasera abierta.

Por fin, para terminar todo, este flujograma general resume proyecto:

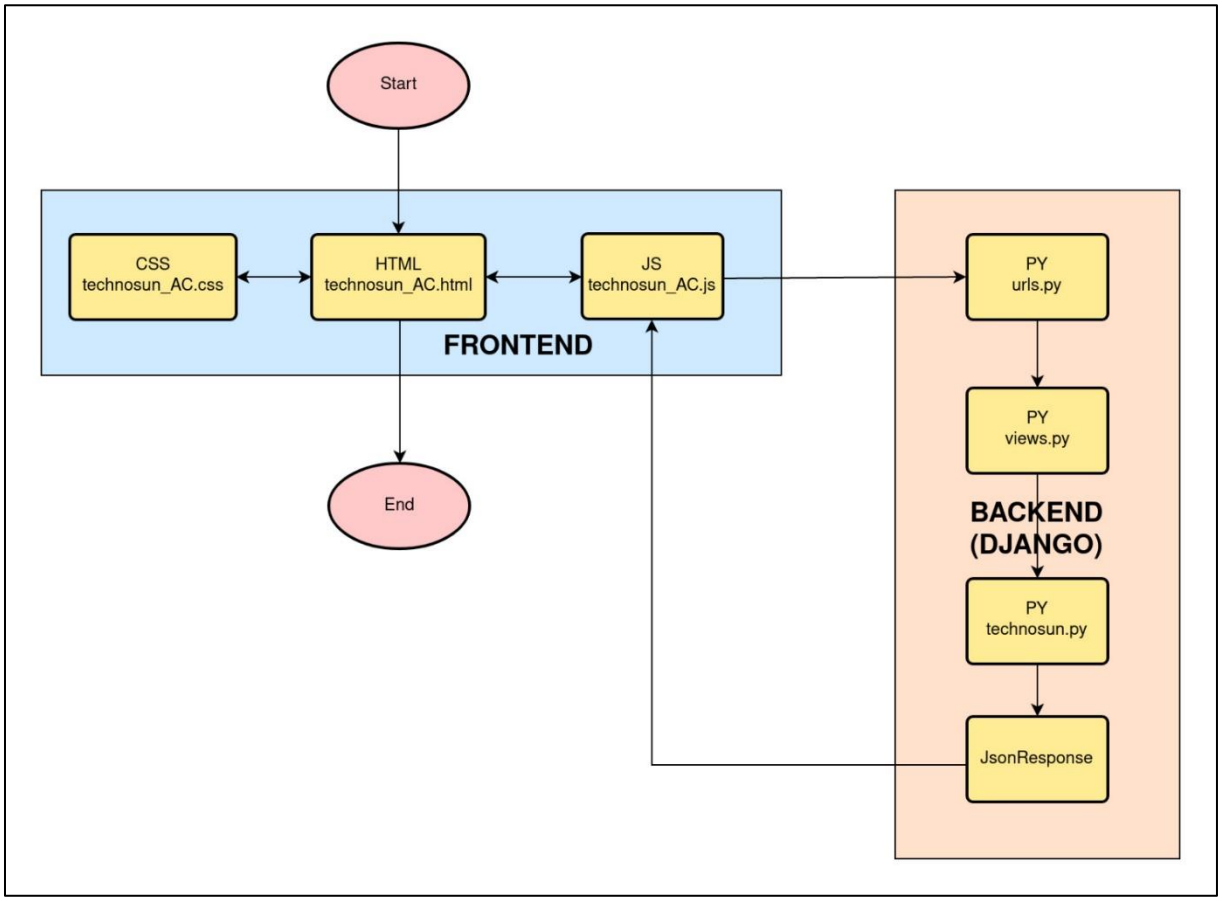


Figura 43: Flujograma general

Resultados

El resultado final es una página complemente funcional que contiene todos los datos que define los componentes del sistema fotovoltaico. Y los más importante del software es como relaciona las coordenadas cartográficas, determina las temperaturas extremas y medias para dichas coordenadas y finalmente determina el comportamiento exacto del material fotovoltaico para garantizar la fiabilidad y durabilidad del sistema o de la instalación.

La última versión del proyecto resulta así:

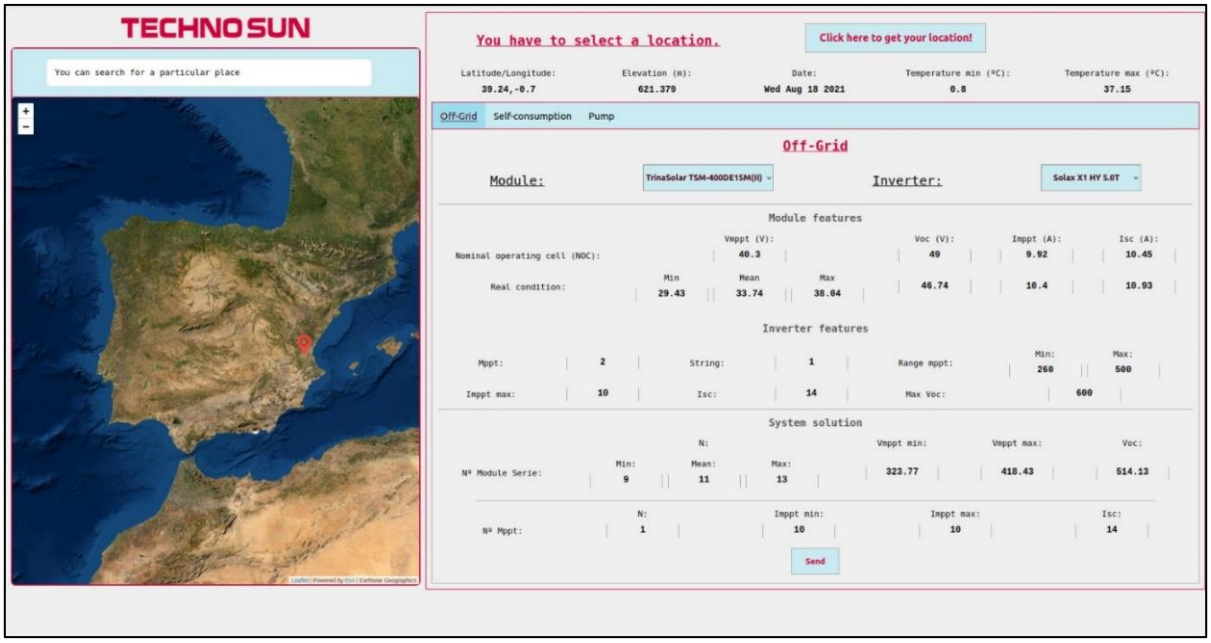


Figura 44: Pagina final del proyecto

Los campos los más interesantes para los usuarios de TECHNOSUN son los campos de la línea N° Module Serie porque dan la estimación final. Pasar por la página web que se ha hecho hace ganar tiempo: no hay que utilizar la calculadora, ni un tablero Excel complejo. Solo enseñar la posición, las marcas de los equipos y el resultado ya está aquí abajo. Además de todo, esta página permite evitar los errores de cálculo y descuidos.

Para que la página sea interesante y fácil de usar, he creado diferentes gadgets como:

- La barra de búsqueda con un geocoding para permitir al usuario de buscar un lugar en particular,
- El botón de localización que permite seleccionar automáticamente tu posición misma (también, te indica la precisión),
- El logo TECHNOSUN contiene un enlace que te dirige directamente en la página oficial de TECHNOSUN,
- Los botones + y –, arriba a la izquierda del mapa, funcionan para ampliarlo y reducirlo,
- Si se reduce el tamaño de la página, el mapa cambia de sitio.

Figura 45 y 46: Pagina web con tamaño de pantalla normal (izquierda) y con un tamaño de pantalla reducida (derecha)

Presupuesto

El presupuesto utilizado para este Trabajo de Fin de Máster es de 0€. De hecho, durante todo el proyecto, se ha utilizado herramientas libres y/o de código abierto. (Por ejemplo, Leaflet, Python, Django y otros…).

En primer lugar, las herramientas utilizadas son lo suficientemente eficaces como para satisfacer las necesidades actuales de mi proyecto y las exigencias de TECHNOSUN. Y no requieren adaptación o cambio en el futuro. Sin embargo, en caso de que la plataforma sea muy utilizada, sería mejor cambiar y utilizar herramientas de pago/profesionales puede ser necesario.

En segundo lugar, es más conveniente para el proyecto y para la empresa. Pedir a una pequeña empresa que pague licencias muy caras puede ser complicado. Por eso, con las escuelas (ETSIGCT o la ESTP), se aprende a trabajar en todas las situaciones. Tanto en grandes empresas que ya poseen las licencias de muchos programas informáticos, tanto en pequeñas empresas, como TECHNOSUN, donde se puede empezar proyectos desde cero.

Conclusiones

Para concluir,

El proyecto era complejo e interesante. Gracias a las habilidades desarrolladas durante el curso y a la ayuda de los tutores, el proyecto fue un éxito. Aunque hemos encontrado problemas, como la API de PVGIS que no se puede llamar desde JavaScript sino desde Python, ahora resulta de gran ayuda para los técnicos de TECHNOSUN. Todavía se tiene que verificar si cumplimos todos los objetivos:

- Crear un visor cartográfico con ortofotos en el que pueda hacer clic para seleccionar la longitud y la latitud de los equipos a instalar,

Hecho. Tenemos un visor operacional que permite hacer clic para recoger las coordenadas.

- Crear una base de datos con el máximo número de paneles solares e inversores posibles (no necesariamente sólo los propuestos por TECHNOSUN),

No totalmente hecho. Tenemos una base de datos completas y operacional. Sin embargo, solo contiene los datos de los equipos usados por la empresa. Se tiene, en el futuro, ampliar la base de datos.

- Implementar los cálculos electrotécnicos para los equipos fotovoltaicos en Python,

Hecho. Tenemos una grande librería de funciones usando los diferentes cálculos electrotécnicos. Puede ser necesario añadir otras funciones más adelante para realizar otros cálculos.

- Utilizar datos astronómicos (orientación del sol, …) y meteorológicos (radiación solar, …) precisos,

Hecho. Los datos que no eran seguros fueron sustituidos por datos de las APIs.

- Que sea exclusivamente gratis (al menos en primer lugar).

Hecho. Tenemos un resultado bien potente que puede competir con aplicaciones con costosas económicamente. Los datos que no eran seguros fueron sustituidos por datos de las APIs.

Los objetivos se cumplen. La página web cumple las expectativas.

Puntos negativos,

Todavía se quedan algunos puntos negativos:

- Si una de las APIs decida de cambiar de formato de datos, se necesitará adaptarnos.
- El algoritmo tarda un poco en dar un resultado (hasta 5 a 8 segundas).
- Los cálculos electrotécnicos son basados en solo una fuente: la documentación de la empresa. No tener fuentes cruzadas puede ser peligroso.
- La elección del equipamiento (los módulos como los inversores) es limitada.
- El geocoding no es muy competente. Sólo conoce las grandes ciudades.

Se ha preguntado qué se debería hacer para mejorar la página aún más.

Para ir adelante,

Además de solucionar estos dichos puntos negativos, se tendrá mejorar varias cosas:

- Rellenar las otras pestanas ‘Self-consumption’ y ‘Pump’ con los cálculos que corresponden.
- Pasar la página en internet con un nombre de dominio (.com o .es) y/o enlazarla con la página de TECHNOSUN que ya existe.
- Ampliar las bases de datos. Ahora tenemos solamente una selección de los módulos e inversores que TECHNOSUN utiliza. En el futuro, seria bien tener una base de datos mucho más completo. Principalmente, si queremos abrir la página a mucho usuario, tener una amplia base de datos será importante.
- Rehacer la presentación de la página. Se ha hecho lo posible para hacerla clara y sencilla de utilización, pero todavía se puede mejorar el funcionamiento y cambiar la paleta de colores para hacerla más juvenil y atractiva.
- Cambiar el geocoding de Leaflet por un geocoding más potente como Nominatim que es un geocoding basado en el lenguaje Python. Nominatim se basa en los datos de OpenStreetMap. También gratis y en código abierto. Se tendría que agregarlo en la parte Python del proyecto y llamarlo desde el JavaScript.

Aplicaciones

Esta página web ya sirve a los técnicos de la empresa TECHNOSUN. Por ejemplo, en el servicio al cliente, el técnico, mientras habla por teléfono, puede introducir los datos del cliente en la página para tener un resultado inmediato sobre las capacidades del dispositivo del cliente.

Esta página permite salvar mucho tiempo de cálculo y evitar los errores de cálculo y descuidos. Entonces permite ganar una gran seguridad y ganar en eficiencia en general. La página debería permitir ser más competitivo y potente que las empresas que no tienen este tipo de herramienta.

Bibliografía

- Mapa de código abierto de Leaflet.

https://leafletjs.com/

- Base de datos de los paneles fotovoltaicos y de datos de los inversores.

Datos de la empresa.

- “Proyecto de la instalación eléctrica autosuficiente en una nave industrial, basada en energía solar fotovoltaica.” – Autor: José Angel Garrido Sarasol – Fecha: 2015.

Datos de la empresa.

- Plugin ESRI Leaflet.

http://esri.github.io/esri-leaflet/

- Web de la empresa TECHNOSUN

https://www.technosun.com/

- Web de w3schools

https://www.w3schools.com/

Se ha utilizado la página web de w3schools para ayudarse con diferentes lenguajes informáticos.

Anexos

En la parte anexos, se pueden encontrar los códigos utilizados en los dos archivos: *technosun* y *technosun_front*.

El árbol de la estructura general del proyecto esta así:

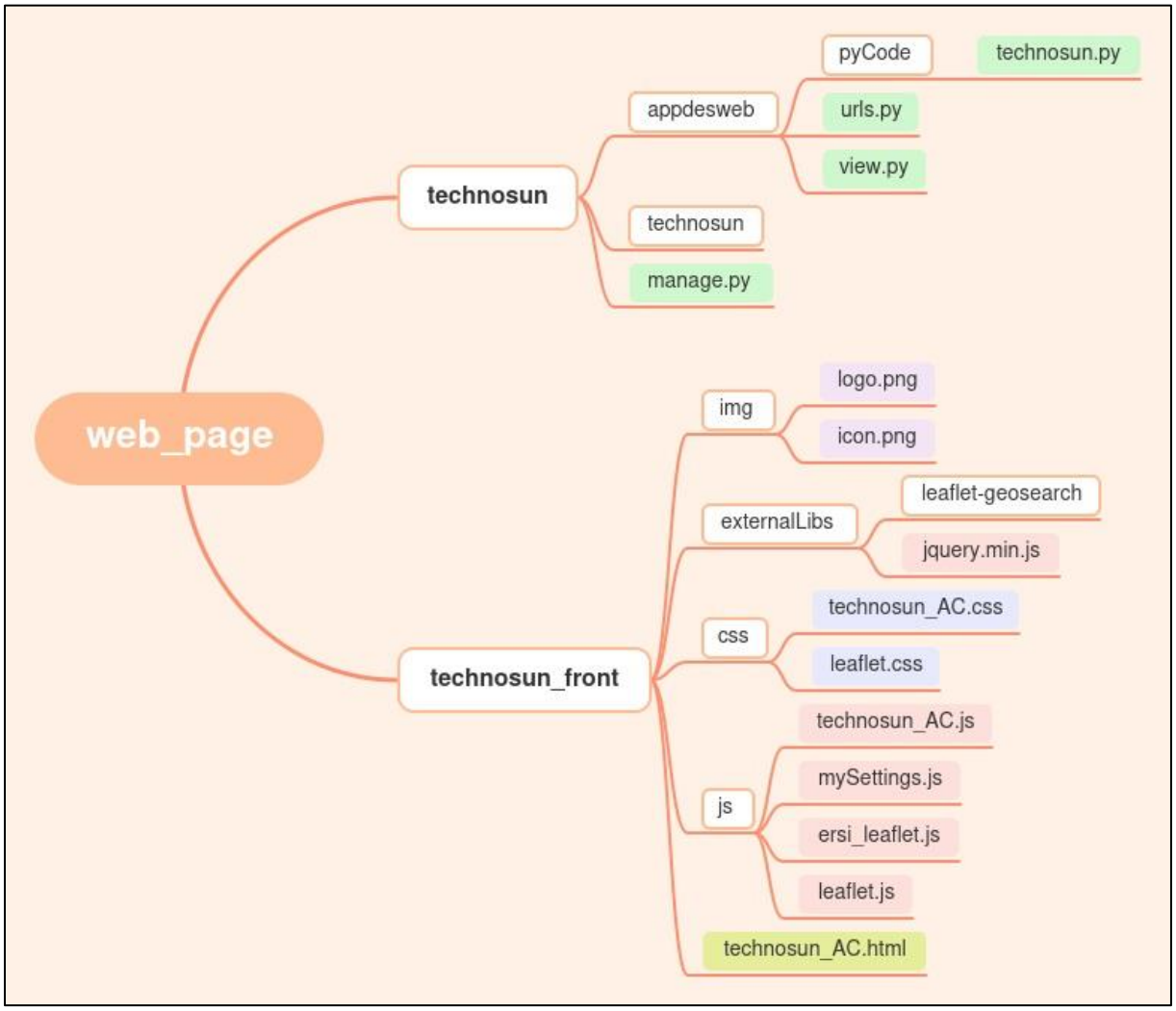


Figura 47: Árbol de la estructura general del proyecto