Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

PLATAFORMA WEB PARA LA EVALUACIÓN Y DIFUSIÓN DE SISTEMAS FOTOVOLTAICOS DEL PROYECTO EMERGIENDO CON EL SOL

Alumno: Jesús Alberto Salazar Ortega

Tutor: Prof. D. Antonio Jesús Rivera Rivas
Prof. D. Juan de la Casa Higueras
Dpto: Informática

Febrero, 2016

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Antonio Jesús Rivera Rivas, tutor del Trabajo Fin de Grado titulado: Plataforma Web para la evalución y difusión de sistemas fotovoltaicos del proyecto Emergiendo con el Sol, que presenta Jesús Alberto Salazar Ortega, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Febrero de 2015

El alumno: Los tutores:

Jesús Alberto Salazar Ortega

Antonio Jesús Rivera Rivas
Juan de la Casa Higueras

Agradecimientos

Índice

Capítulo 1

Introducción

1.1 Introducción al proyecto

En este documento se va a mostrar el proceso seguido para realizar el Trabajo de Fin de Grado relacionado con el Proyecto Emergiendo con el Sol que consiste en la monitorización y visualización de datos referentes a las centrales de fotovoltaicas de Lima, Arequipa y Tacna en Perú, donde se contempla la ampliación al proyecto más centrales que quieran añadirse a este programa.

El objetivo principal es el desarrollo de una plataforma donde se puedan mostrar los datos a través del tiempo de forma intuitiva, ver estadísticas de producción, últimos datos obtenidos por central, etc.

Para conseguir esto el diseño principal es la centralización de los datos de las distintas centrales en un servidor localizado en la Universidad de Jaén y mediante una plataforma web mostrar estos datos. Para todo esto se proporcionó una CPU donde se instalaría toda la centralización de la plataforma y se instalará en la Universidad de Jaén con su propio IP pública.

1.2 Motivación

El inicio de este proyecto se basa en la energía. Perú ha conseguido un crecimiento económico muy importante entre los paises latinoamericanos. Desde un programa de reformas iniciado por Alberto Fujimori en los noventa, Perú experimenta un fuerte crecimiento económico y las privisiones son muy favorables en los proximos años. Pero todo esto se tiene que basar en una base sólida, la energía, y ahí empieza el ámbito de este proyecto.

Un buen uso de las energías renovables ampliaría enormemente las condiciones socio-económicas de los ciudadanos, hace que la región sea más autónoma, genera muchos puestos de trabajo, no hay que tener una plan de procesamiento de residuos porque no hay y sobre todo son inacabables y no contaminan.

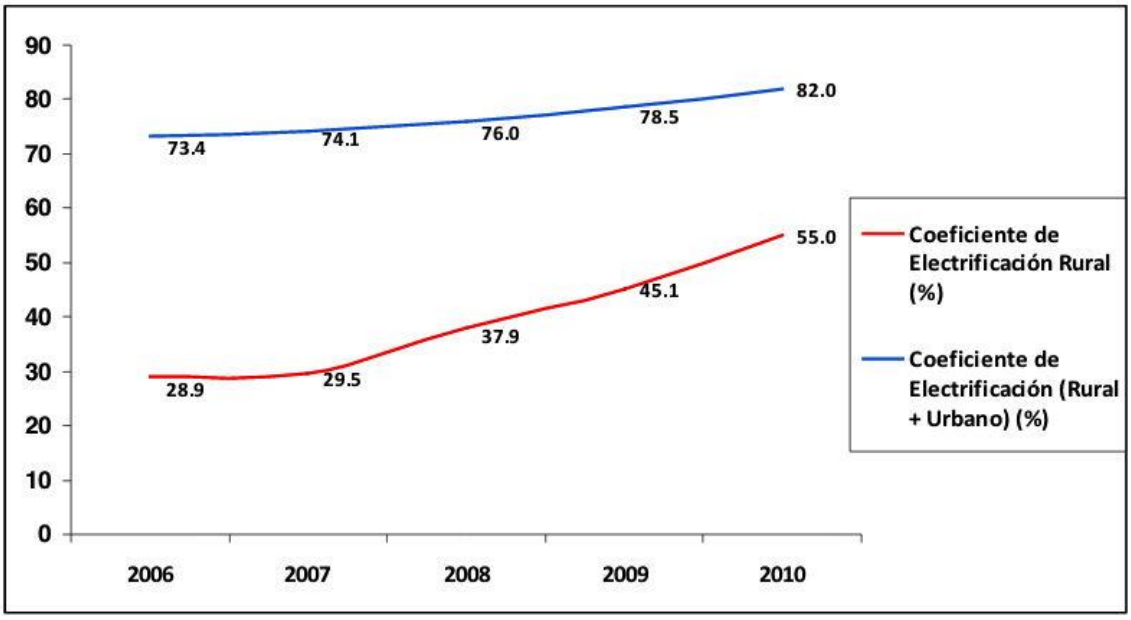


Figura 1. Estadísticas de uso energético en Perú

Encabezando el uso de energías renovables en Perú está la energía hidroeléctrica que es casi la mitad de energía que es consumida por el país, donde la otra mitad es consumida a partir de hidrocarburos, es decir, energía térmica no renovable y contaminadora. Esto se debe al descubrimiento de gas en zonas del bajo Urubamba Camisea que consigió ser el principal acceso de energía antes de empezar a usar la energía hidráulica. La energía solar y la eólica solo suponen un 1.97% de la energía consumida por el país a octubre de 2015. Lo comentado anteriormente se puede ver en la siguiente imagen.

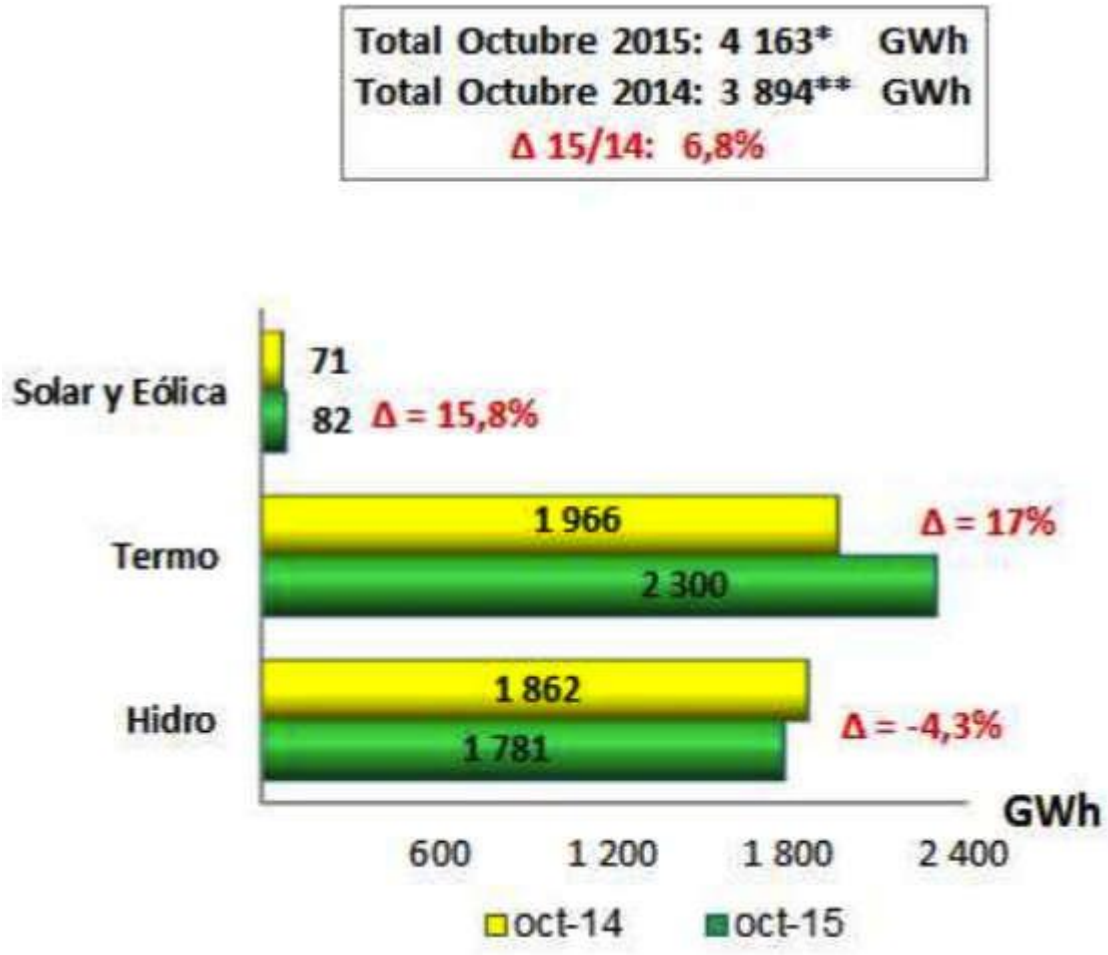


Figura 2. Gráfico sobre las diferentes energías usadas en Perú en 2015

Se pretende que Perú entre al barco de la utilización de la tecnología fotovoltaica para la generación de energía eléctrica haciendose así como el mayor beneficiario de este proyecto, así como la comunidad universitaria con capacidades en ese campo.

A causa de todo esto se originó el proyecto Emergiendo con el Sol, Apoyo institucional al Centro de Energías Renovables de la Universidad Nacional de Ingeniería, en el campo de la generación de energía eléctrica, empleando tecnología fotovoltaica. Y es ahí donde entra el ámbito de este proyecto Trabajo fin de Grado, donde nos encargaremos de la monitorización y visualización de la generación de energía de las centrales fotovoltaicas en Perú.

1.3 Objetivos

El objetivo base de este Trabajo fin de Grado es el de centralizar todos los datos monitorizables de las diferentes centrales fotovoltaicas de Perú y visualizar esos datos de forma inteligente para que puedan ser útiles para las entidades que necesiten estos datos.

Como otros objetivos tenemos como objetivos buscar la rigidez del sistema, ya que se preveen difilcutades técnicas (como apagones) en las centrales donde se va a implementar este sistema, así como filtrar posibles errores de medida de los aparatos para que el sistema muestre los datos más fiables posibles.

Todo esto siguiendo el marco de las normas de monitorización de sistemas fotovoltaicos para la medida, intercambio de datos y análisis, UNE-EN 61724 de agosto 1998 que adopta la Norma Internacional CEI 61724:1998 que podemos encontrar en el Anexo.

Capítulo 2

Tecnologías

2.1 Servidor

2.1.1 Características del servidor

Al inicio del proyecto se proporcionó el siguiente ordenador con las siguientes carácterísticas:

- Arquitectura: 64-bits
- Número de CPUs: 4
- Modelo CPU: Intel (R) Core (TM) i3-4130 CPU @ 3.40GHz
- CPU MHz: 962.625
- L1d cache: 32K
- L1i cache: 32K
- L2 cache: 256K
- L3 cache: 3072K
- Memoria RAM: 1.651.624 kB
- Tamaño del disco duro: 600GB

Como podemos ver es un ordenador con características medias, nuestras partes más importantes son el tamaño del disco, que nos sirve para guardar los datos de las distintas centrales, por un lado los ficheros originales y por otro la copia introducida ya a la base de datos, en el analisis del proyecto se hará un estudio de cuan adecuada es esta capacidad de disco para este proyecto.

También tenemos una cantidad de RAM reducida, pero teníendo en cuenta que es un proyecto donde su plataforma web no debería sufrir una carga excesiva, no debería haber problema, además todas las consultas estarán optimizadas para dar un rendimiento óptimo.

2.1.2 Sistemas operativos para servidores

Un sistema operativo para servidores es un sistema operativo especialmente diseñado para funcionar en servidores, que son ordenadores especializados en trabajar con arquitectura cliente/servidor a las peticiones del cliente en la red.

El sistema operativo para servidores es una capa alta de software desde la cual otros programas software o aplicaciones pueden ejecutarse en el hardware del servidor. Este sistema ayuda a habilitar y facilitar roles típicos para un servidor, como un servidor Web, un servidor de correo, uno de base de datos, de aplicación o de impresión.

Pese a haber distribuciones de todo tipo sólo han sido estudiadas las basadas en linux, ya que son conocidas por su mejor estabilidad, seguridad, escalabilidad y utilidad para este proyecto.

Para la categoría de servidores Linux tenemos los siguientes ejemplos de servidores que son los más utilizados:

- Ubuntu
- Red Hat Enterprise Linux
- SUSE Linux Enterprise Server
- CentOS
- Debian

2.1.3 CentOS

CentOS es una distribución de Linux basada en Red Hat Enterprise Linux, pero quitando todas las limitaciones de Red Hat, haciendo el sistema operativo libre para su uso y distribución.

Es un sistema operativo desarrollado por la comunidad y lo seguirá siendo hasta 2020, por lo tanto tendremos soporte y actualizaciones durante mucho tiempo en el caso de que vayan saliendo nuevas versiones más adelante.

2.1.4 ¿Por qué elegir CentOS?

Nuestro proyecto tiene un gran peso en una plataforma web y CentOS es una muy buena opción para este cometido, además de que posee o pueden ser instaladas fácilmente tecnologías extras que vamos a usar para por ejemplo seguridad con el protocolo SSH y para la instalación de una base de datos adecuada.

Otro objetivo es dejar este proyecto abierto para futuras actualizaciones, instalando CentOS como sistema base, gracias a su comunidad nos garantiza que segirá funcionando durante mucho tiempo y será ampliamente actualizable a los estandares futuros y permite la escalabilidad del proyecto si fuese necesario.

2.2 Servidor Web

2.2.1 ¿Qué es un servidor Web?

Un servidor Web es un programa que usa HTTP para servir archivos que forman las páginas Web a los usuarios en respuesta a sus peticiones las cuales llegan directamente desde los clientes HTTP de sus ordenadores.

Este utiliza un conjunto de protocolos y estandares que sirven para intercambiar datos entre aplicaciones. Diostintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utulizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adipción de estándares abientos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se han creado el organismo WS-I, encargado de desrrollar diversos perfiles para definir de manera más exhaustiva estos estándares. Es una máquina que atiende las peticiones de los clientes web y les envía los recursos solicitados.

2.2.2 Estándares empleados

- **Web Services Protocol Stack**: Así se le denomina al conjunto de servicios y protocolos de los servicios Web.
- **XML (Extensible Markup Language)**: Es el formato estándar para los datos que se vayan a intercambiar.
- **JSON (JavaScript Object Notation)**: Es un formato de texto ligero para el intercambio de datos.
- **SOAP (Simple Object Access Protocol) o XML-RPC (XML Remote Procedure Call)**: Protocolos sobre los que se establece el intercambio.

- **Otros protocolos**: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), o SMTP (Simple Mail Transfer Protocol).
- **WSDL (Web Services Description Language)**: Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.
- **UDDI (Universal Description, Discovery and Integration)**: Protocolo para publicar la información de los servicios Web. Permite comprobar qué servicios web están disponibles.
- **WS-Security (Web Service Security)**: Protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados.
- **REST (Representational State Transfer)**: arquitectura que, haciendo uso del protocolo HTTP, proporciona una API que utiliza cada uno de sus métodos (GET, POST, PUT, DELETE, etc) para poder realizar diferentes operaciones entre la aplicación que ofrece el servicio web y el cliente.
- **CSS (Hoja de estilo en cascada)**: Es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML. El W3C es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

2.2.3 Motivación para la creación de un servidor web

El gran interés por hacer un servicio web es que puede ser utilizado internacionalmente sin instalación previa, simplemente con un navegador que acepte JavaScript. De este modo creamos unos datos accesibles para todo el mundo interesado en nuestros datos.

También nos da la posibilidad de crear capas de usuarios, donde los administradores pueden hacer acciones más concretas que los usuarios estandar.

Desde el punto de vista de carga del sistema también nos interesa mucho este tipo de enfoque, la Computación Distribuida es algo que optimiza mucho la carga del sistema, con esto nos referimos a que parte de los calculos se pueden hacer desde el propio navegador del cliente por medio de JavaScript y se usará cuando sea posible para quitarle carga al sistema.

En el caso de librerías también es muy útil este enfoque, ya que vamos a necesitar librerías para visualizar gráficas adecuadamente en las cuales tenemos una amplia gama donde elegir.

2.2.4 Infraestructura LAMP

LAMP es un acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

- **L**inux, el sistema operativo; En algunos casos también se refiere a LDAP.
- **A**pache, el servidor web;
- **M**ySQL/MariaDB, el gestor de bases de datos;
- **P**erl, **P**HP, o **P**ython, los lenguajes de programación.

En nuestro caso como sistema Linux hemos elegido CentOS, que en el caso de instalalor para ser un servidor web (para más info ir a implementación de servidor) nos instala todos por defecto todas las aplicaciones actualizadas para utilizar LAMP como un servicio web.

Se estudió la idea de usar un Framework para mantener el servidor web, como Drupal o Jango, pero la idea fue rechazada rápidamente, ya que no necesitamos un gran gestor de usuarios ni de contenido, así que se decidió usar LAMP.

Los componentes que usamos para el LAMP son:

- Como servidor de linux: CentOS
- Como servidor web: Apache
- Como gestor base de datos: MariaDB
- Como lenguaje de programación: PHP

Se eligió usar esta configuración ya que es la más adecuada para el tipo de proyecto que vamos a abordar, ya que es un diseño muy propio se eligió empezar a hacer una plataforma web con PHP, HTML, JAVAScript, AJAX y con librerías para los gráficos, este tipo de esquema simplificaba y ayudaba a lo que queríamos conseguir.

Al ser LAMP una parte de la distribución de CentOS también nos aseguramos que tenemos soporte en un futuro para todo nuestro servicio web y una comunidad enorme para apoyarla.

2.2.5 MariaDB

MariaDB es un sistema de gestión de bases de datos derivado de MySQL con licencia GPL. Está desarrollado por Michael (Monty) Widenius (fundador de MySQL) y la comunidad de desarrolladores de software libre. Introduce dos motores de almacenamiento nuevos, uno llamado Aria -que reemplaza con ventajas a MyISAM- y otro llamado XtraDB -en sustitución de InnoDB. Tiene una alta compatibilidad con MySQL ya que posee las mismas órdenes, interfaces, APIs y bibliotecas, siendo su objetivo poder cambiar un servidor por otro directamente. Este SGBD surge a raíz de la compra de Sun Microsystems -compañía que había comprado previamente MySQL AB2 - por parte de Oracle. MariaDB es un fork directo de MySQL que asegura que permanecerá una versión de este producto con licencia GPL. Monty decidió crear esta variante porque estaba convencido de que el único interés de Oracle en MySQL era reducir la competencia que MySQL daba al mayor vendedor de bases de datos relacionales del mundo que es Oracle.

MariaDB tiene varias diferencias con respecto a MySQL que vamos a listar a continuación:

- Diferentes mecanismo de almacenamiento.
- Más facilidad de uso a los programadores añadiendo más datos, comandos, precisión, etc.
- Mayores prestaciones, funcionando más rápido con cargas complejas.

2.2.6 Lenguajes de programación

En este apartado tenemos múchisimas opciones donde elegir, pero empezar en el desarrollo web hay dos tipos de lenguajes de programación basicos, los que se ejecutan en el servidor y los que son ejecutados en el cliente.

Lenguaje utilizado en el servidor: **PHP**

Como hemos comentado anteriormente, al tener una arquitectura LAMP vamos a usar PHP como lenguaje utilizado en el servidor, pero podríamos haber usado cualquier lenguaje del lado del servidor que sea compatible con MariaDB (o estilo MySQL), como es por ejemplo Perl, ASP o JSP.

Lenguaje utilizado en el cliente: **JavaScript**

Para el lado del cliente también tenemos una amplia variedad de lenguajes, pero en este caso vamos a usar JavaScript acompañado de su librería JQuery, que añade muchísima más funcionalidad al JavaScript convencional. Además gracias a JQuery usaremos la técnica de desarrollo web AJAX (***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML*) para crear una página web interactiva, sobre todo para el ámbito de los gráficos.

2.3 Gestión de datos del servidor

2.3.1 Introducción

Teniendo ya definido nuestro gestor de base de datos en esta sección vamos a explicar las tecnologías que vamos a usar para la obtención de los datos de las distintas centrales.

Cada central genera su propio conjuntos de datos monitorizados en un intervalo determinado, que es guardado en un archivo determinado, este archivo es diario, es decir, hay uno por cada día que el sistema ha sido monitorizado y cada archivo contiene todos los datos obtenidos ese día ordenados mediante la hora que fueron tomados. El formato exacto de estos archivos será explicado en Apartado.

En este apartado vamos a definir que tecnologías vamos a usar para recibir los archivos y como lo vamos a introducir.

2.3.2 SSH como protocolo de intercambio

SSH (**S**ecure **SH**ell, en español: intérprete de órdenes seguro) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos, y también puede redirigir el tráfico de X (Sistema de Ventanas X) para poder ejecutar programas gráficos si tenemos ejecutando un Servidor X (en sistemas Unix y Windows).

Este protocolo nos permite intercambiar datos de forma segura entre dos máquinas, que no tienen porque tener la misma arquitectura, con diferentes modos de seguridad y autentificación.

A parte de ser una buena opción ya que CentOS nos da la opción de funcionar como servidor SSH y poder conectarnos remotamente a él, también nos permite transacción de archivos de clientes que nosotros eligamos mediante la criptografía asimétrica.

La criptografía asimetrica es el método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que ha enviado el mensaje. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. Además, los métodos criptográficos garantizan que esa pareja de claves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves.

De este modo a cada central diferente le damos su clave privada única para que encripte sus ficheros y cuando nos lleguen, a parte de estar protegidos sabremos cual es la central que lo manda sin problema, esto es llamado firma electrónica. Aun así para asegurar más la seguridad, cada central nueva tiene un usuario con bajos privilegios donde le está permitido introducir sus datos, haciendo que la entrada al servidor mediante esa clave privada sea viable. Cada central tiene un pequeño espacio donde introducir sus datos.

2.3.3 Lenguajes de programación

Aquí tenemos también muchos lenguajes de programación donde elegir, pero teniendo en cuenta de que solo queremos un programa que este activo permanentemente a forma de daemon, que controla los archivos que llegan y los mete en la base de datos correctamente, así que la elección acabo por ser **C++** como lenguaje para ser el controlador de toda la introducción en la base de datos, este programa tendrá más interactuación de la aquí comentada, que será descrita en Apartado.

2.4 Cliente

Cada central tendrá que tener un cliente instalado en el mismo ordenador donde se están monitorizando los datos y bien configurado para funcionar correctamente con nuestro proyecto. Estos clientes son únicos para cada central, es decir, tendrán un archivo de configuración y unas claves propias para cada uno de ellos.

Se definió al principio del proyecto que estos clientes solo iban a ser instalados en ordenadores con Windows (cualquier versión) y que sería posible que tuviera problemas de conexión a internet o apagones.

Como el programa de servidor, se ha elegido usar **C++** como lenguaje de programación para tener una consistencia entre los dos programas. Aun este requería de una pequeña interfaz gráfica para su mejor visualización, la cual se ha creado con **Qt**.

Qt es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con interfaz gráfica de usuario, así como también para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos y consolas para servidores.

El protocolo de comunicación aquí es el **SSH** como hemos comentado antes, para ellos hemos hecho uso de la librería para C/C++ libssh que nos ofrece muchísimas opciones para trabajar con el protocolo SSH, entre ellas las que necesitamos, gracias a esta tecnología podemos tener un sistema robusto y datos consistentes.

Para la comodidad de nuestros administrados y clientes/centrales, estos clientes serán dados en forma de instalador mediante **NSIS**, un sistema profesional para general instaladores para Windows.

Capítulo 4

Ingeniería del software

3.1 Introducción

El análisis de un proyecto informático es la parte principal por la que empezar ya que por ahí se van a sostener todos los demás pilares de cualquier proyecto. Este apartado se trata básicamente de determinar los objetivos a realizar, los limites que este sistema puede llegar atener, caracterizar y definir su estructura, las acciones que puede llevar y su funcionamiento, además de que debe de marcar las directrices que le permiten alcanzar sus objetivos principales que se le proponen y evaluar el factor causa-consecuencia.

En esta fase se analiza la situación actual a fin de desarrollar una visión de lo que se quiere en un futuro y a seleccionar las estrategias que se aplicarán para alcanzar lo deseado.

La planificación en ingeniería del Software es una de las etapas más importantes de un proyecto ya que nos hace una estimación de recursos, costes y planificación temporal, aunque este siendo un proyecto, a mi criterio, complejo y amplio es difícil poner en un marco de medidas cuantitativas variables que solo conocemos como datos cualitativos, pero aun así conseguiremos unas estimaciones razonables.

3.1.1 Desarrollo del proyecto

En este apartado veremos de forma resumida todo lo que veremos en los capísulos sucesivos, para comodidad del lector han sido separadas en diferentes áreas.

- **Análisis de requerimientos**: Se extraen los requisitos del producto de software. En esta etapa la habilidad y experiencia en la ingeniería del software es crítica para reconocer requisitos incompletos, ambiguos o contradictorios. Usualmente el cliente/usuario tiene una visión incompleta/inexacta de lo que necesita y es necesario ayudarle para obtener la visión completa de los requerimientos. El contenido

de comunicación en esta etapa es muy intenso ya que el objetivo es eliminar la ambigüedad en la medida de lo posible.

- **Especificación**: Es la tarea de describir detalladamente el software a ser escrito, de una forma rigurosa. Se describe el comportamiento esperado del software y su interacción con los usuarios y/o otros sistemas.
- **Diseño y arquitectura**: Determinar como funcionará de forma general sin entrar en detalles incorporando consideraciones de la implementación tecnológica, como el hardware, la red, etc. Consiste en el diseño de los componentes del sistema que dan respuesta a las funcionalidades descritas en la segunda etapa también conocidas como las entidades de negocio. Generalmente se realiza en base a diagramas que permitan describir las interacciones entre las entidades y su secuenciado.
- **Programación**: Se traduce el diseño a código. Es la parte más obvia del trabajo de ingeniería de software y la primera en que se obtienen resultados “tangibles”. No necesariamente es la etapa más larga ni la más compleja aunque una especificación o diseño incompletos/ambiguos pueden exigir que, tareas propias de las etapas anteriores se tengan que realizarse en esta.
- **Prueba**: Consiste en comprobar que el software responda/realice correctamente las tareas indicadas en la especificación. Es una buena praxis realizar pruebas a distintos niveles (por ejemplo primero a nivel unitario y después de forma integrada de cada componente) y por equipos diferenciados del de desarrollo (pruebas cruzadas entre los programadores o realizadas por un área de test independiente).
- **Documentación**: Realización del manual de usuario, y posiblemente un manual técnico con el propósito de mantenimiento futuro y ampliaciones al sistema. Las tareas de esta etapa se inician ya en el primera fase pero sólo finalizan una vez terminadas las pruebas.

3.1.2 Análisis de requerimientos

Los requerimientos especifican qué es lo que el sistema debe hacer (sus funciones) y sus propiedades esenciales y deseables. La captura de los requerimientos tiene como objetivo principal la comprensión de lo que los clientes y los usuarios esperan que haga el sistema. Un requerimiento expresa el propósito del sistema sin considerar como se va a implantar. En otras palabras, los requerimientos identifican el qué del sistema, mientras que el diseño establece el cómo del sistema.

El análisis de requerimientos es el conjunto de técnicas y procedimientos que nos permiten conocer los elementos necesarios para definir un proyecto de software. Es una tarea de ingeniería del software que permite especificar las características operacionales del software, indicar la interfaz del software con otros elementos del sistema y establecer las restricciones que debe cumplir el software.

La especificación de requerimientos suministra al técnico y al cliente, los medios para valorar el cumplimiento de resultados, procedimientos y datos, una vez que se haya construido.

La tarea de análisis de los requerimientos es un proceso de descubrimiento y refinamiento, el cliente y el desarrollador tienen un papel activo en la ingeniería de requerimientos de software. El cliente intenta plantear un sistema que en muchas ocasiones es confuso para él, sin embargo, es necesario que describa los datos, que especifique las funciones y el comportamiento del sistema que desea. El objetivo es que el desarrollador actúe como un negociador, un interrogador, un consultor, o sea, como persona que consulta y propone para resolver las necesidades del cliente.

El análisis de requerimientos proporciona una vía para que los clientes y lo desarrolladores lleguen a un acuerdo sobre lo que debe hacer el sistema. La especificación, producto de este análisis proporciona las pautas a seguir a los diseñadores del sistema.

Como tenemos claro el propósito de nuestro proyecto, vamos a definir los requerimientos de este. Estos requerimientos han de ser realistas, correctos, consistentes y completos. Entendiendo esto podemos clasificar los requerimientos en dos apartados, funcionales y no funcionales.

3.1.3 Requerimientos funcionales

Son declaraciones de los servicios quedebe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer.

En este apartado de los requerimientos nos vamos a basar en el estudio de las funcionalidades del cliente que quiera utilizar nuestro sitio web y para que disfrute adecuadamente de una utilización del sistema. En este sistema tenemos varios roles, así que los vamos a diferenciar las funciones por rol.

Cliente sin cuenta

- Iniciar sesión.
- El sistema deberá proporcionar medios adecuados para seleccionar la central que el usuario esté interesado en visualizar.
- El sistema deberá proporcionar los medios adecuados para seleccionar el día/mes/año que el usuario esté interesado en visualizar.
- Mostrar datos relevantes dependiendo de lo que estemos viendo, si por ejemplo estamos viendo el día actual, datos relevantes serían los datos más actuales que tenga el sistema.

- Gráficos interactivos para que la selección de datos sea útil para el usuario que quiera visualizarla.

Como podemos ver, un cliente sin cuenta o como podríamos decir, un cliente anónimo lo único que puede hacer si viene de visitante es visualizar todos los datos que elija de cualquier central. La funcionalidad que se busca es que cualquier persona interesada en este datos los tenga de forma ordenada, fácil y sobre todo útil.

Ahora pasamos a las funciones que un administrador podra usar, lógicamente las listadas anteriormente también pueden ser usadas por un administrador, que no necesita hacer identificarse para ello, pero en el caso de querer acceder a las siguientes funciones deberá hacerlo.

Administrador

- Añadir administrador.
- Añadir una nueva central.
- Disponer de los instaladores para cada central en concreto para que puedan añadirse al programa Emergiendo con el Sol.
- Descargar los datos de la gráfica del día, mes o año seleccionado.
- Mostrar un contador de visitas.

Los administradores podrán crear nuevas centrales y por cada central se generará un instalador que es lo que el administrador deberá mandar a la central de destino para que pueda añadirse al programa.

La descarga de datos mostrados será diferente a los datos reales, ya que los ficheros base tienen demasiada información y muchísima entradas, la versión que ofrecemos es una compactada y resumida para que pueda ser útil en el caso de necesidad de consultarla.

Las centrales fotovoltaicas que entrar en este proyecto son parte de nuestros clientes por el otro lado, la funcionalidad que le debemos ofrecer es totalmente distinta pero muy simple, el transpaso correcto de los datos. Aquí simplemente tenemos que asegurar la llegada correcta y consistente de los datos a nuestro servidor.

3.1.4 Requerimientos no funcionales

Como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema.

Este apartado podemos dividirlo en dos partes en requerimientos de hardware y requerimientos de software y a su vez podemos dividirlo dependiendo del sistema.

Acceso al servicio web

Para acceder a nuestra plataforma web simplemente hace falta un navegador web actual que acepte HTML 5 y JavaScript, que hoy en día es muy básico contar con esta tecnología.

Ordenador donde se instalará el cliente

El cliente está pensado para funcionar en cualquier tipo de hardware, pero como software se especifica que use una distribución de windows. Al estar hecho en C++ y Qt, es posible que funcione también en distribuciones linux, pero no está pensado para que sea compatible. También se requiere una conexión a internet.

Servidor

Esta es la parte de más carga de funcionalidades tiene, donde abarca desde la plataforma web, el almacenamiento de datos o la base de datos. Esto lo podemos separas en dos grandes grupos:

Hardware servidor

- **CPU:** Esta es la parte que más carga va a soportar ya que el servidor va a ejecutar muchas aplicaciones a la vez, se recomienda como mínimo un procesador multinucleo.
- **Memoria RAM:** Para la cantidad de personas que pueden estar viendo nuestro servicio web a la vez, que se estima no sean muchas, unos 2GB de RAM deberían soportar la carga perfectamente.
- **Gráficos:** No requiere tarjeta gráfica ni pantalla.
- **Capacidad de disco duro:** Esto dependerá del número de centrales que queramos monitorizar, como máximo un día una central se calcula que se requeriran unos 2MB de memoria. Si por ejemplo queremos mantener 10 centrales durante 10 años necesitaremos un disco duro de: 10 centrales x 2MB x 10 años x 365 días = 73000MB que es lo equivalente a unos 70GB.

Software servidor

- **Sistema operativo:** Para este proyecto se ha elegido CentOS, pero debería funcionar todo el sistema en cualquier sistema similar a esta distribución de linux con su propio apache, MariaDB/MySQL y el servidor SSH.
- **Base de datos:** Cualquier gestor de base de datos con notación de MySQL debería funcionar perfectamente si se quiere cambiar de gestor.
- **SSH:** Se puede usar cualquier tipo de aplicación que funcione con las propiedades básicas que se piden para un servidor SSH.

Requerimientos de la Interfaz Web

Como para cualquier tipo de interfaz a cara de usuario hay que enfocarlo en que nuestra interfaz se usable para que nuestro usuarios puedan trabajar con nuestra aplicación de forma fácil y efectiva.

La Organización Internacional para la Estandarización (ISO) ofrece dos definiciones de usabilidad:

ISO/IEC 9126: "La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso"

ISO/IEC 9241: "Usabilidad es la eficacia, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico"

A partir de la conceptualización llevada a cabo por la ISO, se infieren los principios básicos en los que se basa la usabilidad:

- **Facilidad de Aprendizaje**: facilidad con la que nuevos usuarios desarrollan una interacción efectiva con el sistema o producto. Está relacionada con la predicibilidad, sintetización, familiaridad, la generalización de los conocimientos previos y la consistencia.
- **Facilidad de Uso**: facilidad con la que el usuario hace uso de la herramienta, con menos pasos o más naturales a su formación específica. Tiene que ver con la eficacia y eficiencia de la herramienta.
- **Flexibilidad**: relativa a la variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información. También abarca la posibilidad de diálogo, la multiplicidad de vías para realizar la tarea, similitud con tareas anteriores y la optimización entre el usuario y el sistema.

- **Robustez**: es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos. Está relacionada con la capacidad de observación del usuario, de recuperación de información y de ajuste de la tarea al usuario.

3.2 Análisis de sistema

El desarrollo de un sistema de información basado en computación incluye una fase de análisis de sistema. El Análisis de Sistemas trata básicamente de determinar los objetivos y límites del sistema objeto de análisis, caracterizar su estructura y funcionamiento, marcar las directrices que permitan alcanzar los objetivos propuestos y evaluar sus consecuencias.

Una de las formas más habituales y convenientes de analizar un sistema consiste en construir un prototipo (un modelo en definitiva) del mismo.

Para estos modelos vamos a definir que tipos de usuarias van a entrar a nuestras aplicaciones, los casos de usos y escenarios para ejemplificar la utilización de nuestra aplicación.

3.2.1 Perfiles de usuario

Aquí tenemos varios perfiles a identificar de muy diferentes ámbitos, donde destacan los perfiles de usuarios de nuestro servicio web donde se encuentra los usuarios anónimos y los administradores. Por el otro lado tenemos otra gran perfil bien definido que son las centrales fotovoltaicas de nuestro programa.

Siendo esto así haremos un análisis de estos tipos de usuarios:

Usuario anónimo o sin registrar

Un usuario normal de nuestro servicio web no require especial conocimiento/entrenamiento ya que el servicio web está pensado para ser intuitivo en su diseño. Aunque los datos en los gráficos presentes pertenecen al ámbito

fotovoltaico y es recomendable conocerlos para sacarle provecho a nuestro proyecto, pero no es un requerimiento fundamental.

Administrador

A parte de los conocimientos previos ya marcados para un usuario anónimo, un administrador tiene que tener un claro conocimiento relativo al proyecto a parte de estar involucrado activamente en el mismo. Añadir una nueva central es algo más allá que simplemente añadirla a nuestra web, primero nuestros administradores tienen que llegar a algún tipo de comunicación para tener acuerdos para añadir nuevas centrales.

No se requiere amplios conocimientos de informática para ello, pero adjuntado a esta documentación en el Anexo podremos encontrar un manual de como instalar en nuestra plataforma nuevas centrales en el caso de que haya cualquier tipo de duda.

Central

Con central nos referimos al tipo de usuario encargado de que nuestro cliente sea instalador donde corresponde y ver su estado periodicamente para notificar si hay cualquier tipo de incidencia o error.

Lo que se le proporcionará al operario que instale nuestro cliente es un instalador muy simple con unas instrucciones mínimas. Todo esto será proporcionado por un administrador a una central en concreto. Si el operario tiene los suficientes conocimientos informáticos ni siquiera requerirá de instrucciones para una correcta instalacción, aun así se proporcionará un manual en el Anexo.

Si se requiere una buena habilidad social para la buena comunicación con nuestros administradores para que se lleve la instalación correctamente.

3.2.2 Casos de uso

Los casos de uso son ampliamente utilizados en el modelado en el análisis de sistemas para identificar y expresar los requerimientos funcionales de un sistema. Cada caso de uso es un escenario genérico o un evento del cual el sistema debe dar una respuesta previamente definida.

En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo la especialización y la generalización son relaciones.

Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo.

Un caso de uso da una descripción de los componentes, acciones y reacciones a lo que sucede. Así que lo vamos a definir de la siguiente forma:

- Nombre del caso de uso
- Actores
- Condiciones previas que se deben cumplir
- Operaciones simples que se quieren realizar
- Excepciones que puedan ocurrir

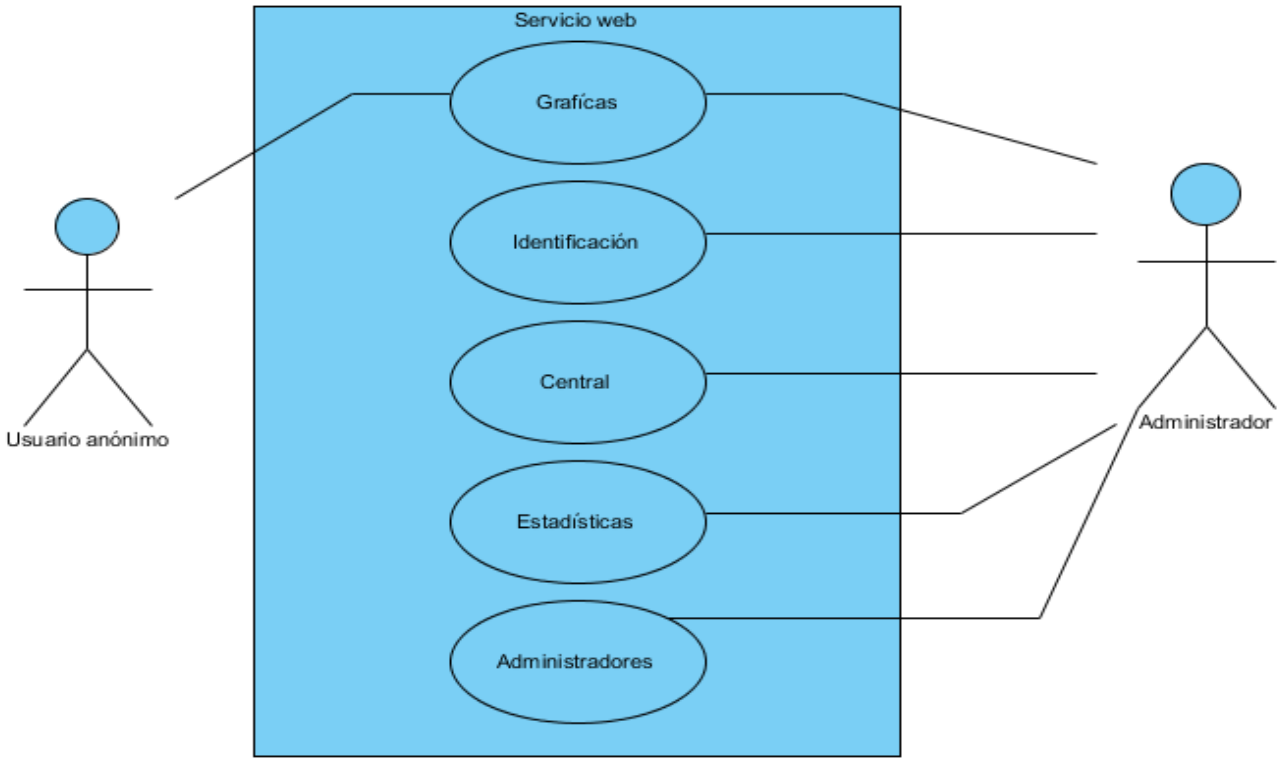
Todos los casos de uso que vamos a comentar aquí están asociados con nuestro servicio web, que es la única parte de nuestro proyecto que tiene amplia interacción con los usuarios y es la interesante de estudiar.

Una definición previa, es que un Actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema. En nuestro sistema podemos encontrar dos tipos de actores:

- **Usuarios anónimos**: Un usuario anónimo es aquel que navega en nuestro sitio web (o usa cualquier servicio de Internet) sin identificarse como usuario registrado.
- **Administrador**: Cualquier usuario identificado es un administrador, que se encarga de la actualización y supervisión del servicio.

Habiendo definido ya esta parte vamos a empezar a definir los casos de uso, estos mayoritariamente usarán los requerimientos funcionales que definimos en un apartado anterior.

Para empezar vamos a mostrar un diagrama del funcionamiento general del sistema:



Partiendo de este caso de uso genérico pasaremos a explicar cada caso de uso independientemente.

Teniedo este primer caso de uso del funcionamiento general del sistema aprovecharemos para explicar que es cada uno de los componentes que se muestran:

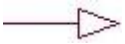
- **Actores**: Son los roles anteriormente comentados, son representados mediantes monigotes.
- **Sistema**: Todos los casos de uso pertenecen a un sistema que es representado por un rectángulo con el nombre del sistema en él.
- **Casos de uso**: Son las tareas que se pueden hacer dentro del sistema al que pertenecen en este caso todas esas tareas se pueden en nuestro servicio web. Se representa mediante una elipse.
- **Asociaciones**: Representa que el actor puede interactúar con el caso de uso para llevarla a cabo.
- **Relaciones**: Los casos de uso pueden tener relaciones entre ellos, se pueden dar las siguientes:

  - **Asociación** 

    Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

  - **Dependencia o Instanciación** 

    Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.

- **Generalización** 

  Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de **Uso**(<<uses>>) o de **Herencia** (<<extends>>).

  Este tipo de relación esta orientado exclusivamente para casos de uso (y no para actores).

  <<extends>>: Se recomienda utilizar cuando un caso de uso es similar a otro (características).

  <<uses>>: Se recomienda utilizar cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica.

  De lo anterior cabe mencionar que tiene el mismo paradigma en diseño y modelamiento de clases, en donde esta la duda clásica de **usar** o **heredar**.

Definido esto pasamos a estudiar los diferentes casos de usos concretos que hay en nuestro sistema.

Caso de uso 1: Identificarse

Actores: Administrador

Condiciones previas: Se ha ingreado a nuestro servicio web.

Operaciones:

1. El usuario va al apartado de login, introduce los datos y acepta. (E-1)

Excepciones:

E-1. Los datos son incorrectos o el usuario no existe

Caso de uso 2: Visualizar gráficas

Actores: Usuario anónimo ó Administrador

Condiciones previas: Se ha ingreado a nuestro servicio web.

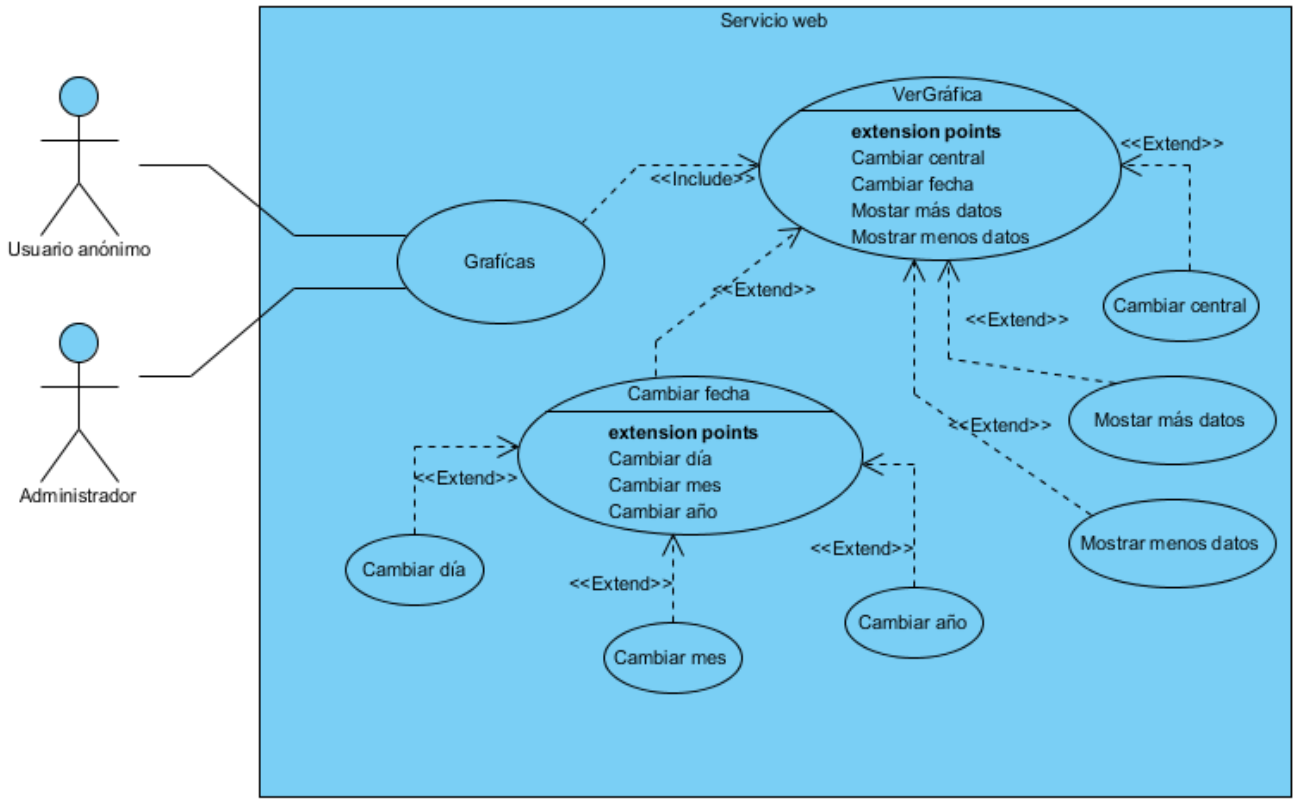
Operaciones:

1. El usuario selecciona la central que quiere ver (E-1, E-2)
2. El usuario selecciona el día/mes/año que quiere ver (E-1)

Excepciones:

E-1. La combinación fecha-central no tiene datos

E-2. No existen centrales en el sistema



Caso de uso 3: Añadir central

Actores: Administrador

Condiciones previas: Se ha ingreado a nuestro servicio web y se ha identificado correctamente.

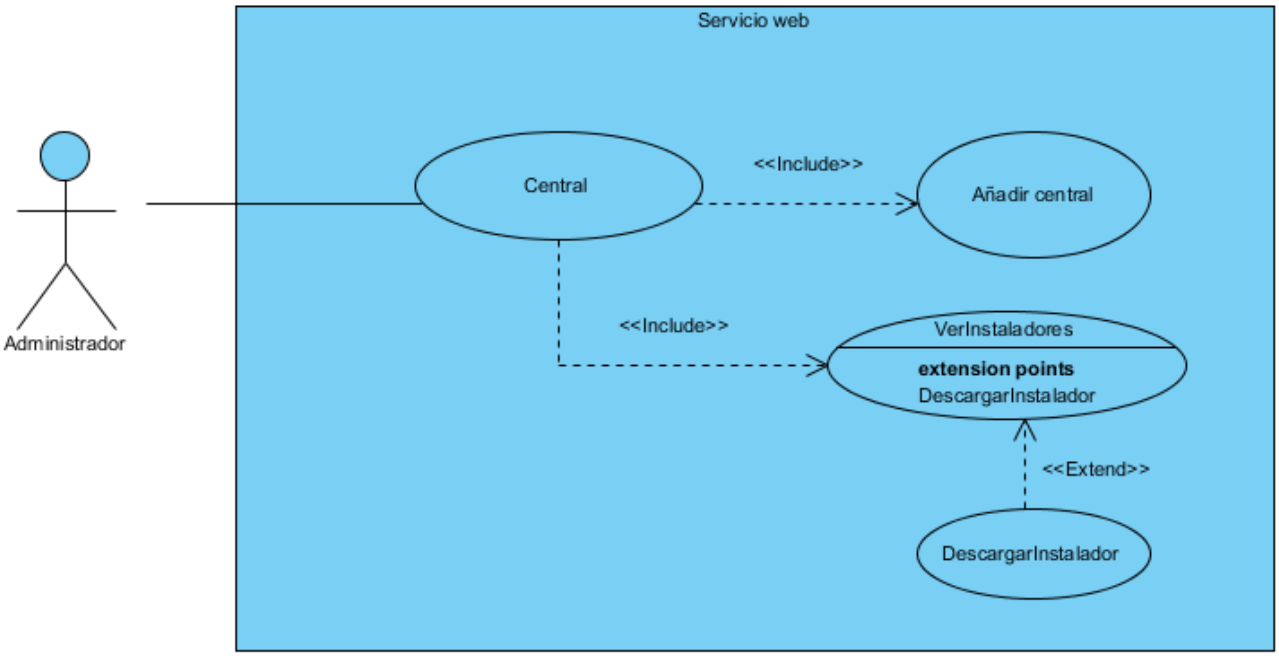
Operaciones:

1. Añadir una nueva central (E-1,E-2)
2. Descargar el instalador de esa central en concreto.

Excepciones:

E-1. La central que se intenta introducir ya existe.

E-2. Los datos introducidos son incorrectos.



Caso de uso 4: Descargar datos de una gráfica

Actores: Administrador

Condiciones previas: Se ha ingreado a nuestro servicio web y se ha identificado correctamente.

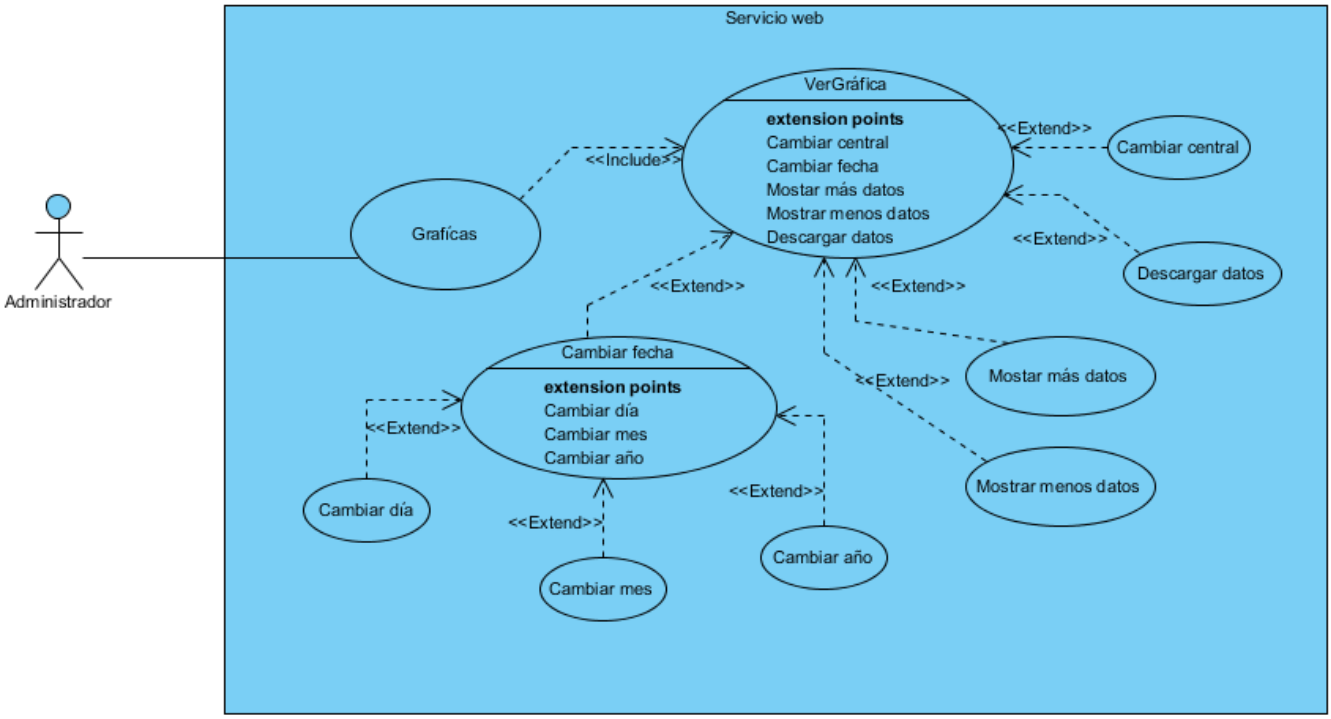
Operaciones:

1. El administrador selecciona la central que quiere ver (E-1, E-2)
2. El administrador selecciona el día/mes/año que quiere ver (E-1)
3. Descargar datos

Excepciones:

E-1. La central que se intenta introducir ya existe.

E-2. Los datos introducidos son incorrectos.



Caso de uso 5: Añadir nuevo administrador

Actores: Administrador

Condiciones previas: Se ha ingreado a nuestro servicio web y se ha identificado correctamente.

Operaciones:

1. Introducir el nuevo usuario y contraseña y darle a Aceptar (E-1)

Excepciones:

E-1. Los datos son erróneos o el usuario ya existe.

Caso de uso 6: Visualizar estadísticas

Actores: Administrador

Condiciones previas: Se ha ingreado a nuestro servicio web y se ha identificado correctamente.

Operaciones:

1. El administrador selecciona ver las estadísticas.

3.2.3 Escenarios

En los casos de uso definimos funcionalidades genéricas y como se enlazaban estas operaciones. Los escenarios ejemplifican estos casos de uso en casos que se podrían dar realmente.

Estos escenarios no son más que ejemplos ficticios pero con un nivel de detalle más profundo para dar a conocer las reacciones de usuarios concretos a situaciones relacionadas con nuestro sistema para descubrir poblemas o incidencias de cualquier tipo.

Estos escenarios por lo general los usuarios lo encuentras más simple de entender que una función de funcionalidad. Así que vamos a hacer un escenario por cada caso de uso que hemos definido, se pueden hacer muchos escenarios por cada uno, pero con uno debería ser suficiente para aclarar las cosas.

Los escenarios los a dividir en los siguientes componentes:

- Nombre del escenario
- Actores involucrados
- Descripción
- Eventos

Escenario 1: IdentificaciónCarlos

Actores: Carlos (Administrador)

Descripción: El administrador Carlos quiere identificarse para acceder a las funcionalidades de administrador.

Eventos:

1. El usuario Carlos entra a nuestro servicio Web
2. Carlos despliega el menú de usuarios y hace selecciona Iniciar sesión
3. Carlos introduce su nombre de usuario, contraseña y presiona Entrar
4. Si ha introducido los datos bien los datos volverá a la página principal con las opciones especiales de administración

Escenario 2: VisualizarGráficasCarlos

Actores: Carlos (Administrador o usuario anónimo, no importa)

Descripción: El usuario Carlos quiere ver los datos de la central fotovoltáica de Lima del mes de abril de 2015.

Eventos:

1. El usuario Carlos entra a nuestro servicio Web
2. Carlos selecciona la central de Lima
3. Carlos selecciona el mes de abril de 2015
4. Si ese mes ya ha pasado y hay datos para ese mes para la central de Lima se le mostrarán, en el caso de que no haya se le avisará.

Escenario 3: AñadirCentral

Actores: Un administrador

Descripción: El administrador quiere añadir una nueva central (llamemos central Omega) a nuestro sistema y mandar el software necesario.

Eventos:

1. El administrador a creado un convenio con la central Omega (esto no forma parte de nuestro proyecto) para unirse a nuestro sistema.
2. El administrador entra a nuestro servicio Web
3. El administrador se identifica en nuestro servicio Web como administrador
4. En los menús el adminsitrador selecciona Añadir central y rellena el formulario correctamente, usando el manual de instalación de nuevas centrales en caso de duda
5. El administrador ahora selecciona el menú de Gestionar centrales y ahí encontrará los instaladores de todas las centrales, se descarga la de la central Omega
6. El administrador envía al operario de la central de Omega el instalador para que sea instalado en su central dándole el manual de instalación de clientes de centrales para su correcta instalación.
7. Si todo ha ido bien la central empezará a enviar datos a nuestro sistema e conforme vayan llegando se podrán visualizar en nuestro sistema

Escenario 4: DescargarDatos

Actores: Un administrador

Descripción: Un administrador quiere descargar los datos de la central fotovoltáica de Arequipa del día 15 de julio de 2015.

Eventos:

1. El administrador entra a nuestro servicio Web
2. El administrador se identifica en nuestro servicio Web como administrador
3. El administrador selecciona la central de Arequipa
4. El administrador selecciona el mes de día 15 de julio de 2015
5. Si ese mes ya ha pasado y hay datos para ese mes para la central de Lima se le mostrarán, en el caso de que no haya se le avisará.
6. En el caso de ese día tenga datos, el administrador tendrá la opción de descargarse una copia simplificada de ese día en formato csv ordenado por hora.

Escenario 5: AñadirAdministrador

Actores: Un administrador

Descripción: Un administrador quiere añadir un nuevo administrador al sistema, Jorge.

Eventos:

1. El administrador entra a nuestro servicio Web
2. El administrador se identifica en nuestro servicio Web como administrador
3. El administrador selecciona el menú de Añadir administrador
4. El administrador introduce un nombre de usuario y contraseña para nuevo administrador Jorge
5. El administrador le da los datos de la nueva cuenta a Jorge para que puede identificarse como administrador

Escenario 6: VisualizarEstadísticas

Actores: Un administrador

Descripción: Un administrador quiere ver estadísticas de tráfico del servicio web

Eventos:

1. El administrador entra a nuestro servicio Web
2. El administrador se identifica en nuestro servicio Web como administrador
3. El administrador en la página principal ya puede ver un resumen del tráfico de la página
4. El administrador selecciona la interfaz resumida de estadísticas y le dirige a unas estadísticas más detallas con varios parámetros seleccionables

3.3 Diseño de Software

El diseño es técnicamente la parte central de la ingeniería del software. Durante el diseño se desarrollan, revisan y se documentan los refinamientos progresivos de las estructuras de datos, de la estructura del programa y de los detalles procedimentales. El diseño da como resultado representaciones cuya calidad puede ser evaluada.

En buen diseño debe tener tres características fundamentales que son:

- El diseño debe implementar todos los requisitos explícitos obtenidos en la etapa de análisis.
- El diseño debe ser una guía que puedan leer y entender los que construyen el código y los que prueban y mantienen el software.
- El diseño debe proporcionar una idea completa de lo que es el software.

La actividad del diseño la vamos a dividir en tres partes: es establecimiento de la estructura del sistema (diseño del sistema), el establecimiento de la estructura de datos (diseño de los datos) y las representaciones de interfaz (diseño de la interfaz).

3.3.1 Diseño del sistema

En esta parte vamos a definir todas las clases que formarán parte de nuestro proyecto en lo que se llama un diagrama de clases.

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones (incluyendo herencia, agregación, asociación, etc.). Los diagramas de clase son el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño). El diagrama de clases de más alto nivel, será lógicamente un dibujo de los paquetes que componen el sistema. Las clases se documentan con una descripción de lo que hacen, sus métodos y sus atributos. Las relaciones entre clases se documentan con una descripción de su propósito, sus objetos que intervienen en la relación y su opcionalidad (cuando un objeto es opcional el que intervenga en una relación).

Estos diagramas se dividen en las siguientes partes:

1. **Clase**: Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio

En UML, una clase es representada por un rectángulo que posee tres divisiones: En donde:

- Superior: Contiene el nombre de la Clase
- Intermedio: Contiene los atributos (o variables de instancia) que caracterizan a la Clase (pueden ser private, protected o public).
- Inferior: Contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: private, protected o public).

2. **Atributos**: son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. Ejemplo: el objeto es una puerta, sus propiedades o atributos serían: la marca, tamaño, color y peso.

Tipos de atributos:

- public (+): Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- private (-): Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos lo pueden utilizar).
- protected (#): Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de las subclases que se deriven (ver herencia).

3. **Operaciones/Métodos**: son aquellas actividades o verbos que se pueden realizar con o para este objeto, como por ejemplo abrir, cerrar, buscar, cancelar, confirmar, cargar. El nombre de una operación se escribe con minúsculas si consta de una sola palabra. Si el nombre contiene más de una palabra, cada palabra será unida a la anterior y comenzará con una letra mayúscula, a excepción de la primera palabra que comenzará en minúscula. Por ejemplo: abrirPuerta, cerrarPuerta, buscarPuerta, etc.

Tipos de métodos:

- public (+): Indica que el método será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- private (-): Indica que el método sólo será accesible desde dentro de la clase (sólo otros métodos de la clase lo pueden utilizar).
- protected (#): Indica que el método no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de métodos de las subclases que se deriven (ver herencia).

4. **Herencia (Especialización/Generalización)**: Indica que una subclase hereda los métodos y atributos especificados poruna Super Clase (también llamada clase padre), por ende la Subclaseademás de poseer sus propios métodos y atributos, poseerá lascaracterísticas y atributos visibles de la Super Clase (public y protected).

Bibliografía

1. Perú como economía emergente. http://www.eaeprogramas.es/internacionalizacion/peru-como-economia-emergente/ [Último acceso 23/11/2015]
2. Energías renovables, ventajas y desventajas. http://erenovable.com/energias-renovables-ventajas-y-desventajas/ [Último acceso 23/11/2015]
3. Energía Hidroeléctrica, Energía Renovable y Tradicional en el Perú. http://deltavolt.pe/energia-renovable/renovable-peru [Último acceso 23/11/2015]
4. Top 10 Linux Server Distributions of 2015 http://www.serverwatch.com/columns/slideshows/top-10-linux-server-distributions-of-2015.html [Último acceso 26/11/2015]
5. Web server http://whatis.techtarget.com/definition/Web-server [Último acceso 26/11/2015]
6. Servicios Web http://www.ecured.cu/Servicios_Web [Último acceso 26/11/2015]
7. MariaDB https://es.wikipedia.org/wiki/MariaDB [Último acceso 27/11/2015]
8. Lenguajes Web http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html [Último acceso 27/11/2015]
9. Uso de PHP http://php.net/manual/es/intro-whatis.php
10. Librerias de gráficos de JavaScript http://www.flotcharts.org/flot/examples/