



Universidad Politécnica
de Madrid



**Escuela Técnica Superior de
Ingenieros Informáticos**

Grado en Ingeniería Informática

Trabajo Fin de Grado

**Desarrollo de Aplicación Web para
Administración de Sensores
Inalámbricos Mediante el Estándar
BACnet**

Autor: Ciprian Ilut Pop

Tutor: Miguel García Remesal

Madrid, Septiembre de 2020

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Título: Desarrollo de Aplicación Web para Administración de Sensores
Inalámbricos Mediante el Estándar BACnet

Septiembre 2020

Autor: Ciprian Ilut Pop

Tutor:

Miguel García Remesal
Departamento Inteligencia Artificial (DIA)
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

El presente trabajo se centra en el desarrollo de una aplicación web para la administración de sensores administrados mediante el estándar BACnet.

Para este proyecto en particular se ha considerado únicamente la presentación de los datos relacionados a cada objeto BACnet y la administración de los mismo, así como la administración de los usuarios a los que pertenece cada objeto BACnet.

El objetivo principal de esta propuesta es la simplificar la administración de dichos objetos al ver que las principales librerías de libre colaboración disponibles actualmente, como es el caso de *BACnet Stack*, ofrecen una interfaz poco intuitiva y difícil de utilizar.

Este proyecto se ha llevado a cabo mediante trabajo autónomo. Para el diseño de la aplicación, se han empleado principalmente conocimientos adquiridos a lo largo del grado a través de las diferentes asignaturas. Mientras que, para el desarrollo, al tratarse de un conocimiento más especializado que utiliza un framework, se han utilizado tutoriales.

El framework elegido para el desarrollo de la aplicación ha sido Flask al ser lo suficientemente potente y sencillo de adaptar a las necesidades de requeridas para este proyecto. Para el diseño y la implementación del Front-End se ha utilizado principalmente HTML, CSS y Jinja2 y en menor medida JavaScript.

Abstract

The current project focuses on the development of a web application for the administration of sensors controlled with BACnet standard, this project only considers the data presentation related to the BACnet objects and the administration of them and the administration of the users that own each object.

The main objective of this proposal is to simplify the administration job for this kind of objects considering that the current libraries that work with a free licence, such as BACnet Stack, do not concern about the presentation of the data.

To develop this project autonomous was required. For the design part skills, resources and knowledge acquired during the degree were used. Whereas for the coding part, multiples tutorials were consulted as the framework used for this project was not part of the degree curricular program.

The chosen framework was Flask, which was powerful enough and easily adaptable to the needs of this work. For the Front-End coding, a combination of HTML, CSS, Jinja2 and JavaScript was used.

Tabla de contenidos

1	Introducción	1
1.1	¿Qué es BACnet?	1
1.2	Objetivos.....	1
1.3	Descripción tareas	3
2	Estado del arte.....	4
2.1	Aplicación alternativa	4
2.2	Metodología ágil	4
2.3	Framework Flask	5
2.4	Base de datos	6
2.5	Control de versiones GIT.....	6
2.6	Docker	6
2.7	Nginx y Gunicorn.....	7
3	Desarrollo	8
3.1	Especificación de requisitos	8
3.1.1	Requisitos administración de usuarios	8
3.1.2	Requisitos administración de objetos BACnet	9
3.1.3	Restricciones del sistema	10
3.2	Estructura web.....	11
3.2.1	Nivel acceso usuarios sin registrado o Nivel cero.....	11
3.2.2	Nivel acceso usuario estándar o Nivel uno	12
3.2.3	Nivel acceso usuario administrador o Nivel dos.....	13
3.3	Implementación	14
3.3.1	Implementación Back-End	14
3.3.1.1	Factorizado de código en paquetes.	15
3.3.1.2	Ficheros de confirmación de la aplicación.	15
3.3.1.3	Ficheros de confirmación e instrucciones del proyecto ..	16
3.3.1.4	Base de datos.....	16
3.3.2	Implementación Front-End	18
3.3.2.1	Generación código HTML.....	18
3.3.2.2	Análisis paginas HTML.....	18
3.4	Despliegue	32
4	Resultados y conclusiones	34
4.1	Líneas futuras	34
5	Bibliografía	35

Tabla de figuras

Figura 1. Esquema general de aplicación.	2
Figura 2. Diagrama de Gantt de las tareas.	3
Figura 3. Diagrama de flujo de Extreme Programming.	4
Figura 4. Funcionamiento del framework Flask.	5
Figura 5. Esquema interacción software Docker con kernel Linux. (Docker_(software), s.f.)	6
Figura 6. Esquema funcionamiento Nginx combinado con Gunicorn. (Configurar NGIX para FLASK, s.f.)	7
Figura 7. Estructura del nivel de acceso sin usuario registrado.	11
Figura 8. Estructura del nivel de acceso usuario estándar.	12
Figura 9. Estructura del nivel de acceso usuario administrador.	13
Figura 10. Captura de la estructura interna de la aplicación.	14
Figura 11. Diagrama entidad relación base de datos.	17
Figura 12. Pagina home.	19
Figura 13. Página about.	19
Figura 14. Página de inicio de sesión.	19
Figura 15. Página registro.	20
Figura 16. Página de recuperación de contraseña.	20
Figura 17. Recuperación de contraseña via e-mail.	21
Figura 18. Página de perfil del usuario.	21
Figura 19. Página de registro de objeto Schedule del usuario.	22
Figura 20. Página de registro de objeto Alarm del usuario.	22
Figura 21. Página de los objetos Schedules del usuario.	23
Figura 22. Página de los datos de un objeto Schedule.	23
Figura 23. Página de actualización de objeto Schedule.	24
Figura 24. Página de registro de evento semanal.	24
Figura 25. Página de actualización de evento semanal.	25
Figura 26. Página de registro de eventos especiales.	25
Figura 27. Página de actualización de eventos especiales.	26
Figura 28. Página de los objetos Alarm del usuario.	26
Figura 29. Página de actualización de objeto Alarm.	27
Figura 30. Página de los usuarios registrado.	27
Figura 31. Página para actualizar datos de usuario.	28
Figura 32. Página de los objetos Schedules del usuario seleccionado.	28
Figura 33. Página de los objetos Alarm del usuario seleccionado.	29
Figura 34. Página de los objetos Schedules del sistema.	29
Figura 35. Página de los objetos Alarm del sistema.	30
Figura 36. Página de registro de objeto Schedules.	30
Figura 37. Página de registro de objeto Alarm.	31
Figura 38. Página de registro de nuevo usuario.	31
Figura 39. Contenido fichero de configuración de Nginx.	32
Figura 40. Contenido fichero de configuración demonio bacnetweb.service	33

1 Introducción

1.1 ¿Qué es BACnet?

Dentro de la administración de edificios se utilizan diferentes sensores que necesitan ser administrados con diferentes protocolos. Actualmente existen múltiples protocolos para este cometido, uno de los más prometedores y extendido siendo el estándar BACnet. Este estándar permite unificar números tipos de sensores bajo un mismo estándar, siendo estos sensores conocidos como objetos.

Este estándar se planteó por primera en 1987 para finalmente salir a la luz 8 años después como el estándar ANSI/ASHRAE (BACnet Stack, s.f.). Estando en continuo desarrollo desde entonces, fue en 2003 cuando paso a ser un estándar internacional.

La finalidad de BACnet es unificar el control de estos sensores bajo un mismo protocolo, permitiendo así reducir la especialización de los operarios.

Actualmente existen numerosas librerías, así como programas para la administración de estos objetos.

Algunas de estas librerías dan la base para el desarrollo de aplicaciones muy potentes, entre estas tenemos *BACnet Stack* (BACnet Stack, s.f.) que es una librería desarrolla en el lenguaje C y nos permite administrar todo tipo de objetos. Esta está desarrollada y mantenida por la comunidad que implementó este protocolo, es por ello por lo que tiene una licencia libre. Además, es una de las más actualizadas y versátiles a la hora de desarrollar software propio, diseñada para ser ejecutada sobre una gran variedad de arquitecturas.

En cuanto a programas con interfaz, podemos encontrar algunas alternativas interesantes como *Yabe Another BACnet Explorer* (Yet Another Bacnet Explore, s.f.) está desarrollado en el lenguaje C# proporcionando una gran variedad de funcionalidades. En este caso, también desarrollada y mantenida de forma colaborativa, además el código es editable al tener licencia libre.

1.2 Objetivos

El principal objetivo de este proyecto es el de desarrollar una alternativa viable a las actuales aplicaciones de administración de objetos BACnet.

Para ello se desarrollará una aplicación web que nos permita eliminar la dependencia de instalar un software en el equipo que utilicemos para la administración.

Esta aplicación web realizará únicamente las funciones básicas para el manejo de los objetos, que son los de registrar los objetos y posteriormente poder editar sus valores asociados. Inicialmente se han desarrollado las funcionalidades de administración para objetos de tipo Schedule y de tipo Alarm.

Los objetos de tipo Schedule son utilizados para planificar el estado de un sensor en un momento dado. Existen dos tipos de planificaciones, el primer tipo será semanal y el segundo tipo puntual.

Los objetos de tipo Alarm se ocupan de registrar anomalías de los diferentes objetos administrados en la red.

Al tratarse de una aplicación compartida entre varios usuarios también estos se deberán administrar ya que cada objeto deberá pertenecer a un usuario. De manera que la aplicación se estructurará en tres niveles de acceso: usuario no registrado, usuario registrado y usuario administrador.

Con el fin de facilitar el despliegue, la aplicación se va a instalar y a configurar en el interior de un contenedor Docker, que por sus características permite eliminar las complicaciones relacionadas a la instalación de todos los componentes del sistema.

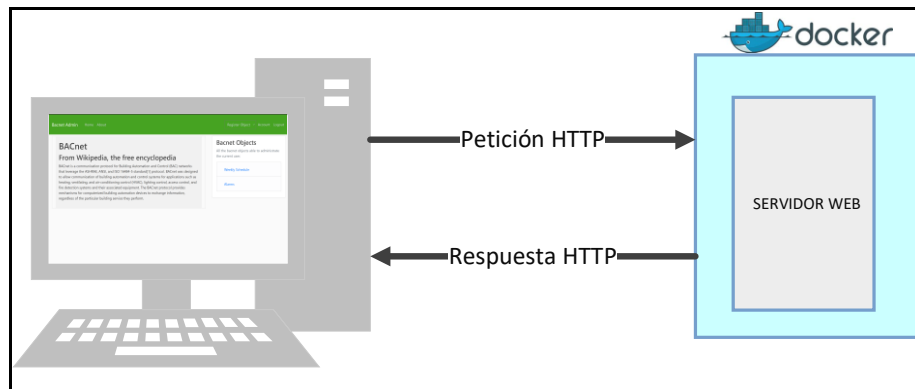


Figura 1. Esquema general de aplicación.

Los objetivos desde el punto de vista académicos son numerosos al tratarse de un proyecto de desarrollo completo. A lo largo del trabajo se van a tratar todos los puntos del ciclo de vida de este, utilizando una metodología ágil que facilite la organización y el desarrollo del proyecto.

El objetivo técnico principal es el de aprender a utilizar el framework Flask (Flask_(web_framework), s.f.) que, aunque se trate de un framework minimalista, es muy potente y fácilmente adaptable a las necesidades requeridas para este proyecto. También se buscará aprender sobre las tecnologías utilizadas actualmente para la puesta en producción de un proyecto de estas características.

1.3 Descripción tareas

Para alcanzar los objetivos propuestos, el trabajo se ha estructurado en las tareas generales presentadas a continuación.

1. Reunión para adquirir requisitos necesarios para la aplicación.
2. Estudio de las tecnologías interesantes a utilizar mediante tutoriales y guías de uso.
3. Diseño de estructura de aplicación web.
4. Desarrollo código de la página web.
5. Creación Docker para su fácil despliegue y pruebas en este entorno.
6. Redacción de memoria y conclusiones.

Durante el desarrollo del proyecto han habido variaciones pero finalmente se ha seguido el siguiente diagrama de Gantt (Descripción de diagrama de Gantt., s.f.).

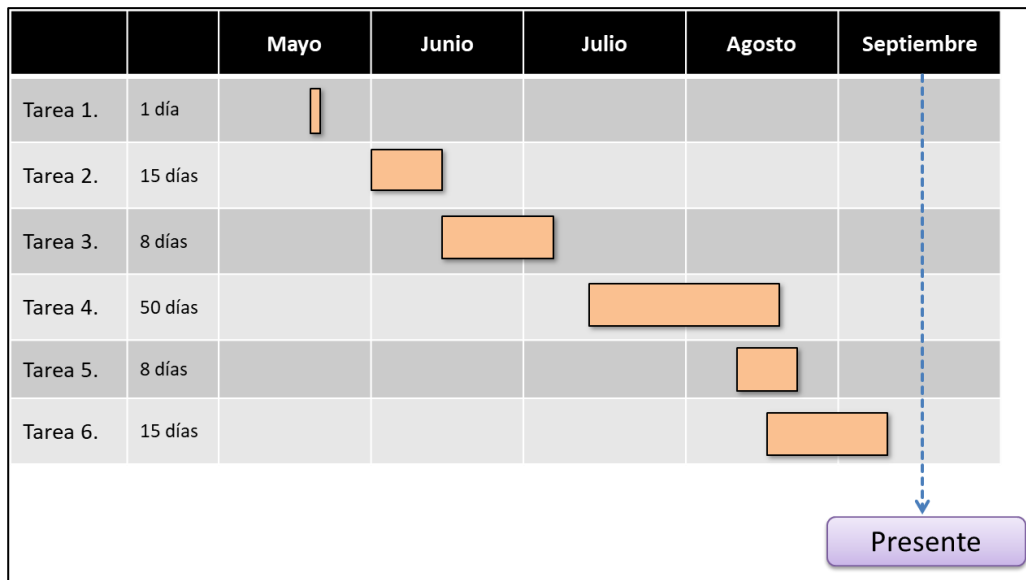


Figura 2. Diagrama de Gantt de las tareas.

2 Estado del arte

2.1 Aplicación alternativa

- **BACnet Explorer**

Este software está desarrollado por la empresa francesa Inneasoft (BACnet Explorer., s.f.), permite administrar únicamente los objetos Schedule, Calendar y Alarm sin posibilidad de añadir más objetos BACnet, siendo una importante limitación.

Dispone de dos versiones gratuita y versión profesional.

- Versión gratuita: esta versión permite descubrir los objetos de tipo Schedule y Calendar que existen en la red, así como mostrar su estado actual en lo que parámetros se refiere.
- Versión profesional: esta versión además de las características de la versión gratuita también permite modificar los parámetros de estos objetos. Esta versión tiene un coste de 390 euros.

2.2 Metodología ágil

En 2001 con la intención de mejorar el desarrollo software, dieciséis desarrolladores escribieron un manifiesto formado por cuatro puntos, lo que dio lugar a un cambio de mentalidad en el desarrollo de software. Gracias a esta metodología se puede desarrollar software de manera más rápida y de mayor calidad. (Metodología Agile, s.f.)

Es por lo que, considerando las características de este proyecto, esta metodología se barajó como la más adecuada.

En particular el modelo utilizado ha sido Extreme Programming (XP) (Joskowicz, s.f.). Este modelo sigue las reglas del manifiesto especificando cuatro fases (Figura 4) que serán repetidas hasta concluir el proyecto.

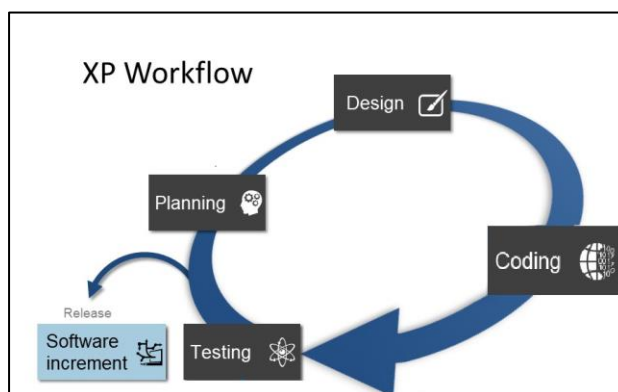


Figura 3. Diagrama de flujo de Extreme Programming.

2.3 Framework Flask

Flask es un microframework desarrollado en Python que permite desarrollar aplicaciones web muy ligeras y a la par potentes. Desarrollado por Armin Ronacher a partir de 2010 (Flask_(web_framework), s.f.), actualmente es uno de los frameworks más populares entre desarrolladores Python. Se puede desarrollar tanto con Python 2.7 como con las últimas versiones, en mi caso he utilizado Python 3.8.

La posibilidad de desarrollar aplicaciones ligeras que Flask ofrece resulta interesante para encapsularlas en un contenedor Docker (Docker_(software), s.f.), ofreciendo la posibilidad de utilizarlo en cualquier equipo sin ninguna característica especial de memoria o procesador. Otra razón por la cual se ha optado por este framework es que, al estar desarrollado en Python, resultaba una buena oportunidad para aprender este lenguaje que no se ha trabajado a lo largo del grado.

Este framework dispone de numerosas bibliotecas para el desarrollo de algunos de los requisitos pedidos. Sin embargo, al necesitar únicamente algunas características, incluir bibliotecas enteras ha resultado innecesario. Es por esto por lo que se ha optado por desarrollar la mayoría de las funcionalidades sin estas bibliotecas.

Para el desarrollo del Font principalmente se ha utilizado HTML5 (Lenguaje_de_marcado, s.f.) ya que está ampliamente integrado con este framework, además de ser el más adecuado para esta labor. En cuanto al diseño, se ha utilizado tanto hojas de estilo CSS (Hoja_de_estilos_en_cascada, s.f.) como el framework Bootstrap para tener un diseño responsive.

El funcionamiento de este framework sigue la arquitectura cliente-servidor, desde el navegador se hace una petición a una dirección URL que será interpretada y solicitada al Web Server Gateway Interface (WSGI) (Que es WSGI, s.f.) levantado por el framework. Posteriormente en caso de coincidir la URL con alguna vista éste solicitará la plantilla HTML y los datos al modelo que está constantemente conectado a la base de datos, generando posteriormente la página que será enviada al navegador.

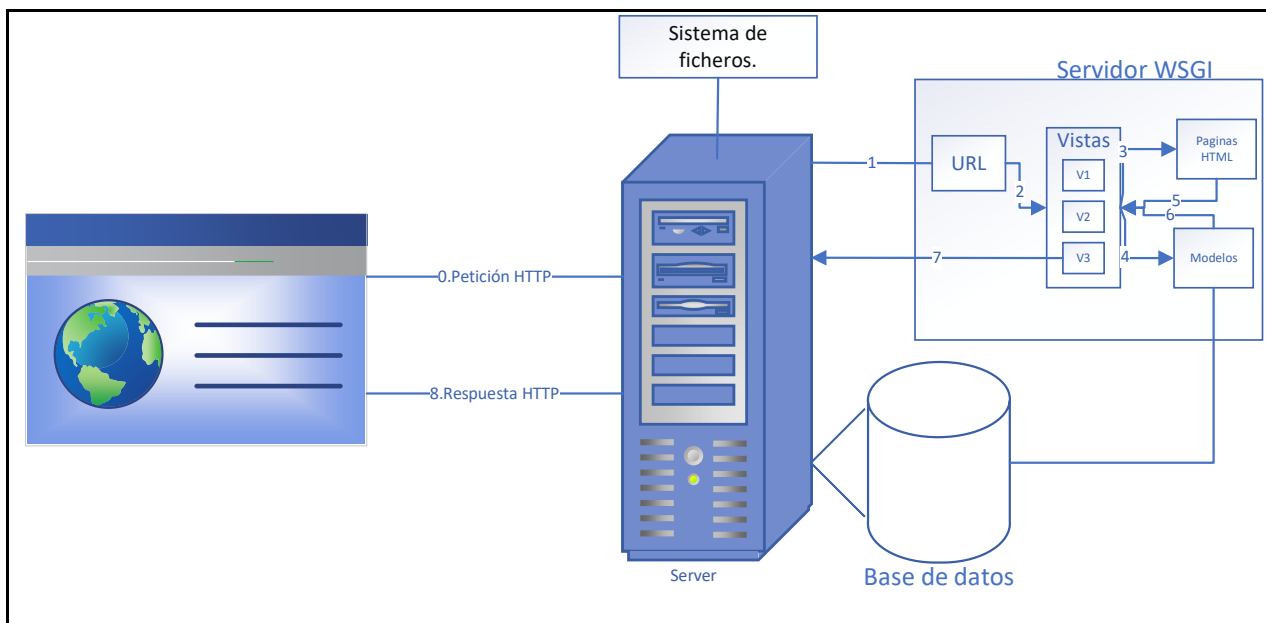


Figura 4. Funcionamiento del framework Flask.

2.4 Base de datos

Para el cometido del almacenamiento de datos se ha elegido el sistema de base de datos relacional PostgreSQL (PostgreSQL, s.f.), siendo este un proyecto maduro que lleva más de treinta años en desarrollo formando parte del proyecto Postgres desarrollado por la universidad de californiana de Berkeley.

Se ha elegido este sistema por su gran variedad de características aplicables para este proyecto, además de ser un software de acceso libre.

2.5 Control de versiones GIT

En la actualidad existen numerosos softwares de control de versiones, pero por su extensión entre desarrolladores se ha optado GIT (Git, s.f.).

Este software fue desarrollado por Linus Torvalds en 2005 y permite mantener versiones de aplicaciones, lo que en este proyecto ha permitido recuperar versiones estables en caso de desastres. Para ello se basa en un esquema cliente – servidor teniendo una copia de todo el código en el servidor, de manera que los clientes son los que suben su contenido a éste para así poder compartirlo entre desarrolladores.

Para esta parte el trabajo autónomo y el uso de la plataforma GitHub (GitHub, s.f.) que permite tener alojar repositorios privados basado en este software, han sido esenciales.

2.6 Docker

Docker es una plataforma que permite el despliegue de aplicaciones sobre un entorno virtual eliminando así las dependencias de librerías en la ejecución de éstas. Esta encapsulación es conocida como contenedores, que al igual que una máquina virtual, tienen un sistema operativo. Sin embargo, resulta menos pesado para la maquina anfitriona ya que estos contenedores interactúan directamente con la característica de virtualización del kernel reduciendo la carga del sistema. Por tanto, en líneas generales, permite un mejor despliegue y reduce dependencias irresueltas en el equipo de despliegue.

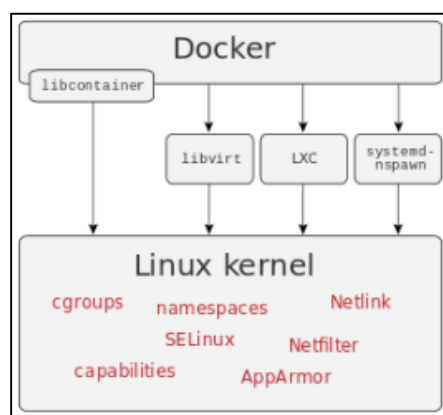


Figura 5. Esquema interacción software Docker con kernel Linux.
(Docker_(software), s.f.)

Este software se puede desplegar tanto en Linux como en Windows, aunque para desplegarlo en Windows se necesita una licencia especial siendo insuficiente la estandar.

2.7 Nginx y Gunicorn

Nginx (Que es NGINX, s.f.) es un servidor web/proxy inverso, un proyecto de libre acceso inicialmente desarrollado por Igor Sysoev, lanzado en su primera versión en 2004.

Este servidor resulta tan interesante porque tiene una gran variedad de funcionalidades, entre las que destacan el balance de carga o la tolerancia a fallos. Cabe destacar también que, a pesar de su reducido gasto de recursos, tiene un gran rendimiento, siendo este uno de los puntos fuertes de Nginx.

Gunicorn (Website oficial de Guicorn, s.f.) es un segundo servidor WSGI que permite ejecutar el código desarrollado con el framework Flask. La diferencia entre éste y Flask es que Gunicorn ejecutará el código con varios procesos, permitiendo tolerancia a fallos.

El funcionamiento de estos dos servidores queda representado en la *Figura 7*. Nginx responde a peticiones estáticas mientras que Gunicorn será el que responda y el que ejecute las peticiones dinámicas.

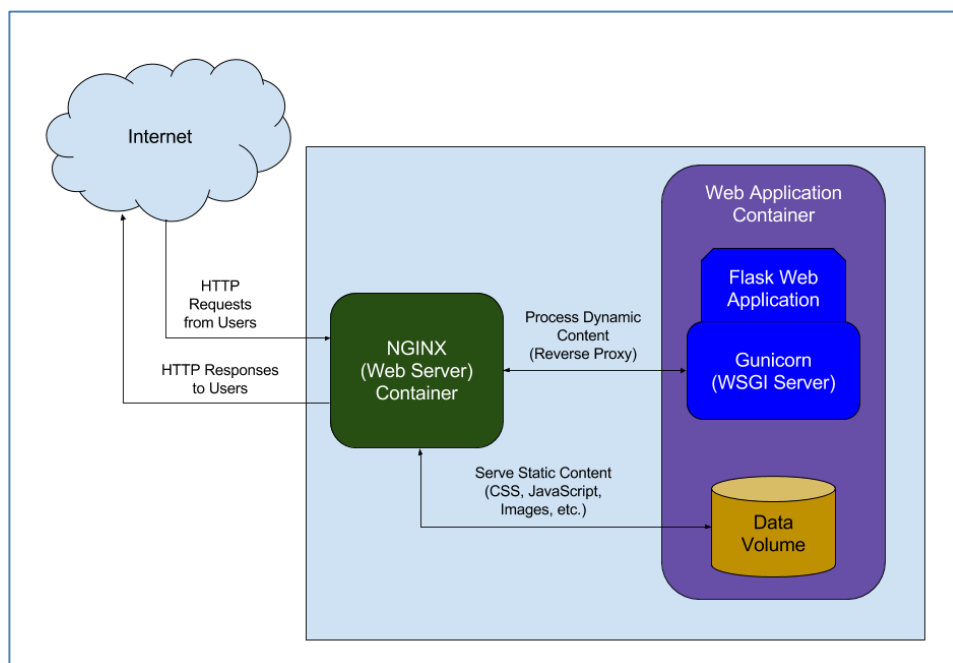


Figura 6. Esquema funcionamiento Nginx combinado con Gunicorn. (Configurar NGIX para FLASK, s.f.)

3 Desarrollo

3.1 Especificación de requisitos

El primer punto del desarrollo ha sido la especificación de requisitos que tiene como meta saber el trabajo que se va a tener que desarrollar para así organizarlo. Para esta labor fue necesario reunirse con el tutor del proyecto, con el fin de especificar los requisitos y así tener un plan de ruta. Con estos requisitos se plantearía cada ciclo de vida, ya que la realización de un proyecto de estas envergaduras requería un guion de trabajo en todo momento.

Los requisitos obtenidos se han clasificado en dos grupos. El primer grupo recoge los requisitos relacionados con la administración de los usuarios, mientras que el segundo recoge aquellos requisitos para la administración de los objetos BACnet. Un tercer punto trata las restricciones del sistema.

3.1.1 Requisitos administración de usuarios

- **Existencia de usuario estándar y usuario administrador**

La aplicación debe permitir distinguir entre dos tipos de usuarios:

- Usuario estándar: es aquel que únicamente es capaz de modificar sus datos y los datos de los objetos que le pertenece.
- Usuario administrador: es aquel que podrá administrar tanto otros usuarios como objetos del sistema.

- **Creación de usuario estándar sin permisos de administrador**

Desde la página de inicio se debe poder crear un usuario estándar sin la necesidad de un acceso administrador.

- **Cambio contraseña y recuperación de contraseña de un usuario**

En caso de haber perdido la contraseña se debe poder recuperar la contraseña vía correo electrónico de una cuenta, conociendo el usuario y el correo electrónico asociados. Teniendo acceso al usuario se podrá cambiar la contraseña antigua relacionado a la cuenta, para esto se debe conocer la contraseña antigua.

- **Cambio de dirección de correo electrónico usuario estándar**

Cada usuario podrá cambiar su dirección de correo electrónico, para ello deberá conocer su contraseña. Siendo esta una medida preventiva.

- **Creación de usuario estándar o administrador con acceso de administrador**

Desde una cuenta administradora se debe poder crear tanto un usuario estándar como un usuario administrador.

- **Asignación o revocación de permisos de administrador**

Será cualquier usuario administrador el que asigne o revoque los permisos de administrador a otros usuarios.

- **Usuario administrador podrá modificar correo electrónico de otros usuarios**

Cualquier usuario administrador podrá cambiar la dirección de correo electrónico asociado a cualquier usuario sin necesidad de conocer la contraseña del usuario en cuestión.

- **Usuario administrador podrá modificar contraseña de otros usuarios**

Cualquier usuario administrado podrá cambiar la contraseña asociada a cualquier usuario.

- **Eliminación de usuario por administrador**

Será únicamente el administrador que podrá eliminar cualquier usuario, ya que en la eliminación se eliminará también el registro de los objetos relacionados al mismo.

- **Operaciones con la tabla de contenidos**

Una de las características necesarias para la administración de los usuarios es la de permitir realizar una búsqueda entre los mismo. Otra característica interesante es que se podrá ordenar las tablas en orden creciente o decreciente en función del contenido de las columnas.

3.1.2 Requisitos administración de objetos BACnet

- **Registro de objetos Schedule y Alarm**

Al tratarse de un desarrollo inicial únicamente se han implementado dos objetos:

- Schedule: Se trata de un objeto BACnet con el cual se puede administrar el estado del sensor en un momento específico. Para este objeto podremos especificar valores de dos maneras:
 - *registro semanal*, un registro que representará un valor que se repetirá de manera semanal.
 - *evento especial*, un registro que representará un valor que solamente representa un valor en un instante especificado.
- Alarm: Este objeto registra una anomalía en cualquier objeto BACnet

- **Modificación datos y valores relacionados a cada objeto**

Tanto en el caso de los objetos Schedule como en el caso de los objetos Alarm cada usuario podrá modificar los valores de aquellos objetos que se le hayan sido asignado.

Cada usuario podrá administrar los valores relacionados a cada objeto, a excepción del identificador del objeto. Los usuarios administradores podrán modificar los datos de cualquier objeto.

- **Usuarios pueden registrar en objetos Schedule eventos especiales y eventos semanales.**

Serán los usuarios los responsables de registrar los eventos especiales y los eventos semanales relacionados a cada objeto de tipo Schedule.

- **Usuarios administradores asignan usuario a objeto**

Serán únicamente los usuarios administradores los que puedan asignar un objeto a un usuario o cambiarlo de ser necesario. De igual manera en el momento de registrar un objeto se podrá asignar el usuario deseado.

- **Marcar objetos Alarm activos**

Los objetos Alarm podrán estar marcados con dos colores en función de su estado, es por ello por lo que en caso de que este activa su color será rojo mientras que si esta desactivada su color será verde.

- **Eliminación de objetos BACnet**

Cada usuario podrá eliminar del registro los objetos que están relacionados a su usuario. Y los usuarios administradores podrán eliminar cualquier objeto del sistema.

- **Operaciones en tabla**

De igual manera que sucede en los requisitos de los usuarios también se podrá realizar una búsqueda en las tablas de los objetos BACnet, así como ordenar las tablas en función del contenido de sus columnas.

3.1.3 Restricciones del sistema

Para el correcto funcionamiento de la aplicación se han establecido una serie de restricciones, que son las siguientes:

- **Identificadores únicos**

Tanto los identificadores de los usuarios como los de cada objeto deben ser únicos en cada tabla.

- **Para recuperar contraseña se debe conocer usuario y correo relacionado a cada usuario**

Como plus de seguridad se debe conocer tanto usuario como correo relacionado a la cuenta, además de tener acceso a la cuenta de correo de destino.

- **Trabajar con datos persistentes**

Como en cualquier aplicación web se deberá trabajar con datos persistentes, es por ello por lo que creamos una base de datos.

- **Nombre de usuario permanente**

El identificador de usuario no se podrá modificar, solamente se podrá modificar el correo electrónico y la contraseña.

- **Accesos de usuarios no registrados**

Se deberá proteger la aplicación de tal manera que no se obtenga una vista si no se ha iniciado sesión y se tenga permiso para acceder esa vista.

- **Diseño *responsive***

No se trata de una restricción muy importante, pero cabe destacar que poder acceder desde un smartphone resulta interesante hoy en día.

- **Comprobar formatos antes de enviar formularios**

Todos los formularios deberán comprobar que cumplen los formatos solicitado, como es el caso de correo electrónico, fechas u horas.

3.2 Estructura web

En el siguiente punto se va a explicar la estructura de la aplicación web. Para facilitar la comprensión, se van a utilizar una serie de esquema que muestra la conexión entre las diferentes vistas de la aplicación.

La aplicación se ha diseñado y estructurado principalmente contemplando tres niveles de acceso: usuario sin registrar, usuario estándar y usuario administrador. Estos apartados se desarrollaran a continuación.

3.2.1 Nivel acceso usuarios sin registrado o Nivel cero

Lo primero que vemos si accedemos a la aplicación web, con URL ciprianilut.ddns.net, será la vista de HOME que nos mostrará un pequeño resumen sobre el protocolo BACnet. Posteriormente tendremos varias opciones para acceder a la vista ABOUT, que nos dará una pequeña explicación sobre el proyecto. Estas dos vistas son únicamente para dar una mejor experiencia de usuario.

En este nivel tenemos las funciones de inicio de sesión, registrar usuario y recuperar contraseña. Según se ve en el esquema la mayoría de las vistas están interconectadas, en el esquema también tenemos la terminación de la URI de cada vista.

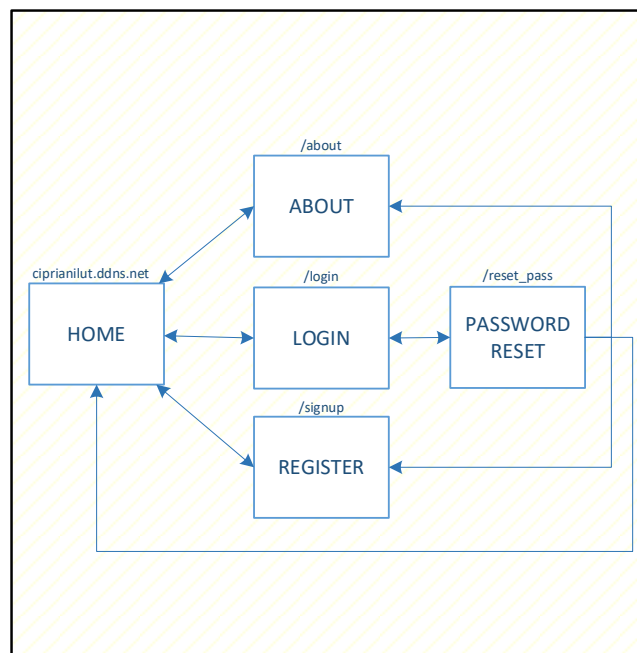


Figura 7. Estructura del nivel de acceso sin usuario registrado.

3.2.2 Nivel acceso usuario estándar o Nivel uno

Tras realizar el inicio de sesión (Tutorial login flask, s.f.) pasamos al segundo nivel. En este nivel podemos realizar todas las tareas de administración de los objetos BACnet que tenemos registrados o asignados al usuario que tiene iniciada sesión.

La primera vista que se nos muestra nos permitirá modificar tanto la contraseña como el correo electrónico del usuario, para estas operaciones se necesitan la contraseña actual del usuario. Desde esta vista podemos realizar varias acciones, algunas de ellas son: ver todos nuestros objetos Schedules o Alarms, registrar un nuevo objeto Schedule o Alarm, etc. También nos lista nuestros objetos, que podemos editar uno por uno o eliminar.

Para administrar un objeto Schedule debemos abrirlo para así poder editar los eventos semanales o los eventos especiales, así como registrar nuevos eventos. Desde esta vista también tenemos la posibilidad de editar los datos relacionados al objeto Schedule. Finalmente, para editar los datos de un objeto Alarm simplemente tenemos que abrirlo desde nuestra lista de alarmas.

A continuación, tenemos un esquema que explica las vistas de la aplicación. En el esquema también tenemos la terminación de la URI de cada vista.

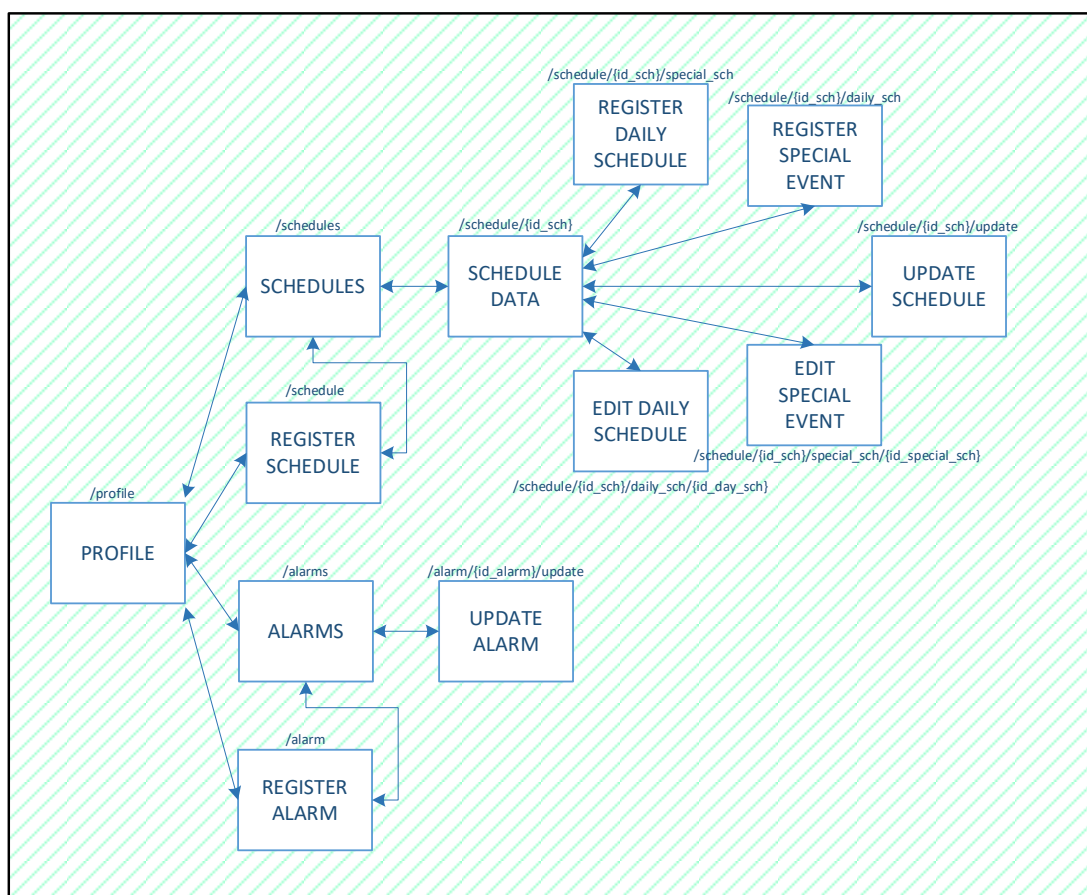


Figura 8. Estructura del nivel de acceso usuario estándar.

3.2.3 Nivel acceso usuario administrador o Nivel dos

En caso de que el inicio de sesión se realice con un usuario administrador además de las características de un usuario estándar tendremos nuevas funcionalidades.

Las nuevas funcionalidades de este nivel traducen en acceder a los datos del usuario de la aplicación, se podrá modificar contraseña y correo electrónico a cualquier usuario, así como modificar los permisos que tienen los mismo, pudiendo asignar permisos de administrador a cualquiera de estos. Finalmente, en cuanto a usuarios se refiere también se podrá eliminar cualquier usuario del sistema y consecuentemente los objetos BACnet relacionados al mismo.

En lo que a objetos BACnet se refiere vamos a tener la posibilidad de registrar objetos de tanto de tipo Schedule como Alarm asignándolos a usuarios específicos, con este nivel de acceso se podrá modificar el usuario al que se ha asignado previamente, así como modificar los datos relacionados a estos.

En el siguiente esquema podemos ver que las funcionalidades y las vistas del nivel de acceso del usuario administrador son una extensión de las desarrolladas para el usuario estándar.

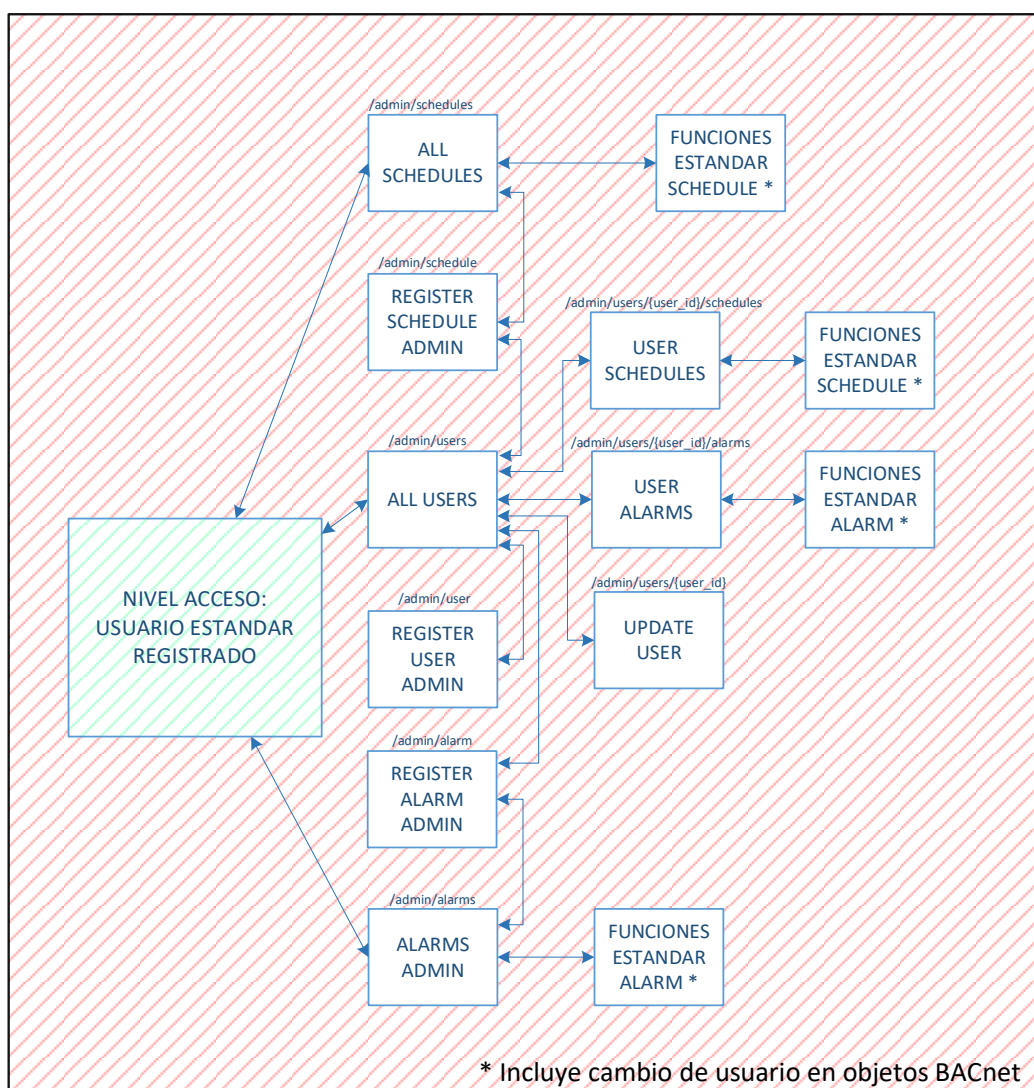


Figura 9. Estructura del nivel de acceso usuario administrador.

3.3 Implementación

3.3.1 Implementación Back-End

Para la implementación de la aplicación se ha utilizado el framework Flask como se ha comentado anteriormente, y la versión de Python ha sido 3.8.

Como ya se ha comentado anteriormente, el servidor de la base de datos esta implementado con PostgreSQL en la versión PSQL (PostgreSQL) 10.14

La conexión entre el servidor PostgreSQL y el framework se realiza mediante la librería SQLAlchemy, que se encarga de serializar las tablas de la base de datos en objetos Python. Es por ello por lo que permite transparencia a la hora de trabajar con la base de datos.

Para facilitar el desarrollo del proyecto, el código Python se ha estructurado en paquetes en función de las funcionalidades implementadas. A continuación, podemos ver la estructura de la aplicación.

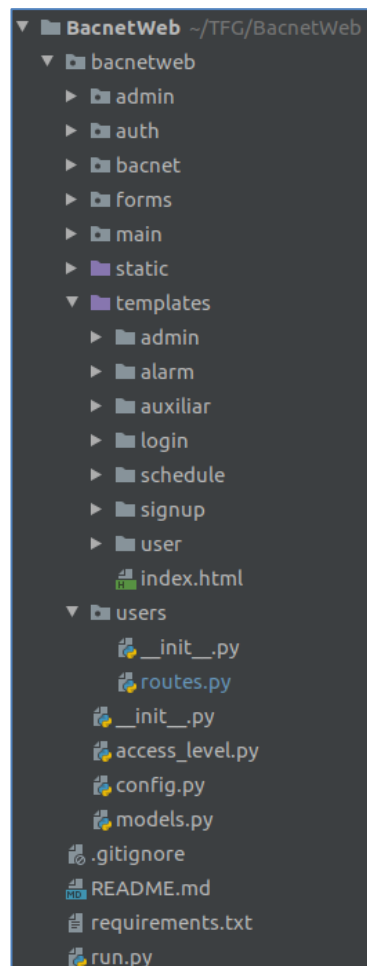





Figura 10. Captura de la estructura interna de la aplicación.

En esta estructura podemos distinguir principalmente 3 tipos de iconos ya que PyCharm, el IDE utilizado para el desarrollo del proyecto, marca el tipo de carpeta en función de su contenido. El icono  que indica que esa carpeta contiene un paquete, el icono  indica un fichero que contiene código Python y

el icono  indica las carpetas que contendrán todas las plantillas HTML, CSS y el contenido estático de la aplicación.

3.3.1.1 Factorizado de código en paquetes.

Como se ha comentado anteriormente, el desarrollo del código se ha decidido factorizar en paquetes, ya que así se podrá tener un código limpio y ordenado, esencial para un proyecto de esta extensión. (Schafer, Factorizar aplicación en paquetes., s.f.)

Para crear un paquete en Python se debe crear una carpeta con dos ficheros. En primer lugar, un fichero denominado `__init__.py` sin contenido (de esta manera el intérprete de Python sabrá que se trata de un paquete) y un segundo fichero denominado `routes.py` que contendrá todas las funcionalidades del paquete.

Para poder utilizar los paquetes combinados y por tanto todas las funcionalidades implementadas, se va a utilizar Flask Blueprint. Blueprint encapsula las funcionalidades y todos los recursos relacionados a los paquetes como si de una aplicación se tratase. Sin embargo, este blueprint debe ser añadido a una aplicación Flask para poder funcionar, en puntos posteriores se abordará este tema.

A continuación, se va a hacer un pequeño resumen del contenido de cada paquete y se indicará que carpetas de *templates* se van a utilizar.

- *Admin*: Este paquete desarrollará las funciones implementadas para un usuario administrador, se ayuda de las plantillas HTML contenidas en la carpeta *templates admin* únicamente.
- *Auth*: Este paquete desarrolla las funciones implementadas para el inicio de sesión y registro de nuevos usuarios, utilizará plantillas contenidas en las carpetas *login* y *signup*.
- *Bacnet*: Este paquete desarrolla las funciones implementadas para la administración de los objetos BACnet, utilizará plantillas de contenidas en las carpetas *Alarm* y *Schedule*.
- *Form*: Este paquete desarrolla las funciones implementadas para la creación de formularios, para ello se ha utilizado la librería WTForms. Esta librería genera principalmente objetos de tipo Flask Form, que utilizados en las plantillas HTML, permiten generar formularios completos con pocas líneas de código, así como validar los datos introducidos.
- *Main*: Este paquete desarrolla la ruta inicial de nuestra aplicación generando la página inicial, utiliza el fichero *index.html*.
- *Users*: Este paquete desarrolla las funciones desarrolladas para la administración del usuario, así como la administración de los datos de la cuenta como la recuperación de la contraseña en caso de ser necesario.

3.3.1.2 Ficheros de confirmación de la aplicación.

En el siguiente punto vamos a tratar los ficheros que se utilizan para la configuración e inicialización de la aplicación implementada. A continuación, se comentará las funciones de estos ficheros.

- `__init__.py`: En este fichero crearemos las instancias de aplicaciones necesarias como una aplicación de correo implementada con la librería *Flask-Mail*. También la aplicación de la base de datos implementada por

la librería *SQLAlchemy* y la aplicación del gestor de sesiones de Flask, que se ocupa de proteger las vistas que requieren inicio de sesión. Esta aplicación se implementa con la librería Flask-Login.

Tras inicializar las instancias de estas aplicaciones se configurarán añadiéndolas a nuestra aplicación. Después debemos añadir las blueprints de todos los paquetes anteriormente generados.

Todo esto será el constructor de nuestra aplicación que se ejecutará desde el fichero de ejecución *run.py*.

- *Config.py*: Este fichero contiene información básica de configuración, así como credenciales almacenadas en las variables del sistema.
- *Model.py*: Este fichero contiene la declaración de los objetos que van a serializar las tablas de la base de datos utilizados por SQLAlchemy.
- *Access_level.py*: Con este fichero generamos los *decorators* necesarios para comprobar que el usuario que está intentado acceder a vistas de administrador tiene esos permisos, se ha hecho de esta manera ya que el gestor de sesiones no tiene esa funcionalidad. (Salvatierra, s.f.)

3.3.1.3 Ficheros de confirmación e instrucciones del proyecto

Siguiendo la idea de hacer software común y accesible en los siguientes ficheros, tenemos una serie de ficheros que nos servirá para tener información relevante del proyecto o incluso la manera de ejecutar el mismo. A continuación, se revisarán estos ficheros para tener una mejor idea de su uso y finalidad.

- *.gitignore*: este fichero se utiliza para indicar al sistema de control de versiones que ficheros no queremos almacenar en el servidor remoto, por tanto, no tener historial de ese fichero. (Crear - Gitignore file , s.f.)
- *README.md*: este fichero se utiliza para hacer un resumen sobre el funcionamiento de la aplicación, así como indicar como se pone en funcionamiento.
- *Requirements.txt*: este fichero se utiliza por el sistema de gestión de paquetes de Python, *PIP*, para instalar las librerías necesarias para el despliegue del proyecto.
- *Run.py*: este fichero se utiliza para inicializar la aplicación y ponerla en ejecución. Será en este fichero donde se podrá indicar el puerto donde la aplicación web atiende peticiones.

3.3.1.4 Base de datos.

En lo que a la base de datos se refiere, como ya se ha comentado, se ha utilizado PostgreSQL. Para ello se ha instalado un servidor PostgreSQL y creado una base de datos en la que a su vez se han creado cuatro tablas. Las tablas son las siguientes:

- **Users**
 - id
 - usr
 - password
 - email

- **Alarms**
 - id
 - state
 - event
 - source
 - source_description
 - alarm_text
 - triggered_time
 - id_user
- **Schedules.**
 - id
 - object_identifier
 - object_name
 - object_type
 - present_value
 - description
 - effective_period
 - exception_schedule
 - list_of_object_property_references
 - priority_for_writing
 - status_flags
 - reliability
 - out_of_service
 - id_user
- **DailySchedules.**
 - id
 - day
 - initial_time
 - final_time
 - value
 - object_id

Para tener una representación gráfica de la base de datos se ha creado este diagrama entidad-relación:

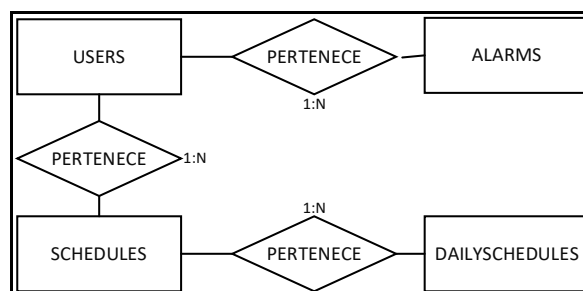


Figura 11. Diagrama entidad relación base de datos.

3.3.2 Implementación Front-End

Para el desarrollo del Front-End se han utilizado el lenguaje de marcas HTML y la hoja de estilos CSS. Para darle un aspecto más profesional y permitir un diseño *responsive* se ha utilizado tanto el framework Bootstrap (Bootstrap_(front-end_framework), s.f.) como algunos scripts de JavaScript (Buscador datos en la tabla con Javascript, s.f.).

Otras funciones, como pueden ser las notificaciones de errores o funcionamiento correcto en las operaciones, se han implementado a través de mensajes de Flask con la librería Message Flashing (Message Flashing Library, s.f.) hacia las plantillas HTML, permitiendo su integración en las mismas mediante Jinja2 (Yu, s.f.). De igual manera para la implementación de formularios se ha utilizado la librería WTForms (Librería WTFForm, s.f.), esta librería genera código HTML que nuevamente se integra en las plantillas HTML gracias a Jinja2.

3.3.2.1 Generación código HTML

Para la generación del código de las páginas HTML se ha utilizado Jinja2 que es un lenguaje para desarrollar plantillas para Python que está altamente integrado con HTML, permitiendo un rápido desarrollo de plantillas y ahorrando de esta manera mucho código en caso de tener muchas páginas con un diseño común como es éste. Para la generación del código HTML tenemos los siguientes elementos:

- *Layout.html*: plantilla HTML que implementará todos los elementos comunes entre todas nuestras páginas HTML.
- *Pagina.html*: página HTML de la funcionalidad que queremos implementar.
- *Código Python*: El renderizado de las páginas y por tanto la creación del código de las páginas HTML se realiza con una llamada desde código Python. Desde éste se podrá enviar variables de manera unidireccional, desde el código Python a la plantilla (como puede ser una variable utilizada en el código) o variables de manera bidireccional, como en el caso de los formularios (que envían variables relacionadas a los campos solicitados y también recibe los resultados introducidos).

3.3.2.2 Análisis paginas HTML

A continuación, se va a hacer un análisis de las vistas de la página utilizando como referencia las URIS de las mismas y se va a obviar la parte inicial de la dirección <http://ciprianilut.ddns.net/>. Para realizar el análisis se va a utilizar los tres niveles de accesos comentados en el punto [3.2](#) de este documento, cada imagen contiene un esquema para así situarse mejor en esquema general.

3.3.2.2.1 Nivel acceso cero

- **/:** Esta página utiliza la plantilla *Index.html*, únicamente muestra el contenido inicial de nuestra página como podemos ver.

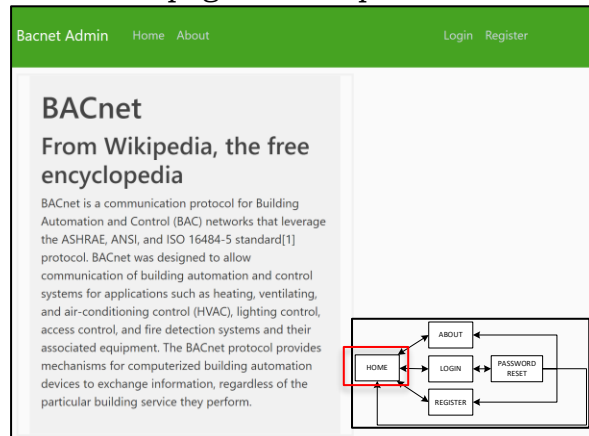


Figura 12. Pagina home.

- **/about:** Esta página utiliza la plantilla *About.html*, contendrá únicamente información sobre la aplicación

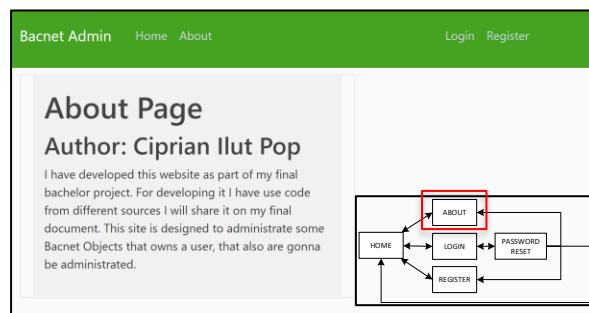


Figura 13. Página about.

- **/login:** Esta página utiliza la plantilla *Login.html*, implementa la función de inicio de sesión.

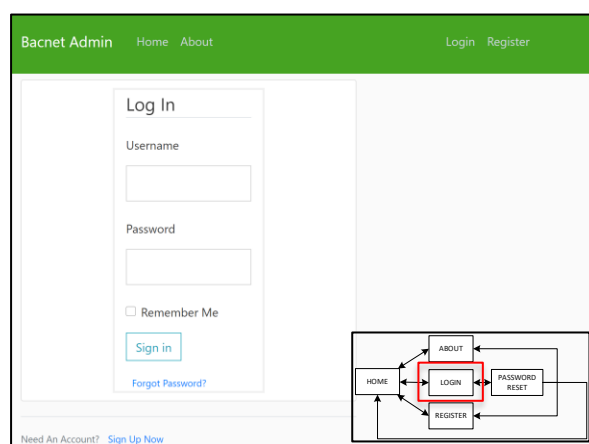


Figura 14. Página de inicio de sesión.

- **/signup:** Esta página utiliza la plantilla *Signup.html*, implementa la función de registro de usuario estándar.

The screenshot shows the 'Join Today' section of the website. It contains four input fields: 'Username', 'Email', 'Password', and 'Password Repeat'. Below these fields is a 'Sign up' button. In the bottom right corner, there is a navigation diagram with boxes for HOME, ABOUT, LOGIN, REGISTER, and PASSWORD RESET. The 'ABOUT' box is highlighted with a red border, and arrows indicate navigation paths between the other pages.

Figura 15. Página registro.

- **/reset_pass:** Esta página se utiliza la plantilla *reset_pass.html*, implementa la función de recuperación de contraseña vía correo electrónico.

The screenshot shows the 'Reset Password' section of the website. It contains two input fields: 'Username' and 'Email'. Below these fields is a 'Reset Password' button. In the bottom right corner, there is a navigation diagram with boxes for HOME, ABOUT, LOGIN, REGISTER, and PASSWORD RESET. The 'PASSWORD RESET' box is highlighted with a red border, and arrows indicate navigation paths between the other pages.

Figura 16. Página de recuperación de contraseña.

- **/new_pass/{token_seguridad}**: Esta página utiliza la plantilla *new_pass.html*, implementa la función de cambio de contraseña. La URL se genera con un token de seguridad relacionado al usuario válido durante treinta minutos y se envía por correo electrónico. (*Implementación de sistema de correo en aplicación., s.f.*)

Figura 17. Recuperación de contraseña vía e-mail.

3.3.2.2.2 Nivel de acceso uno

- **/profile**: Esta página utiliza la plantilla *profile.html*, implementa la función de modificación de datos del usuario que ha iniciado sesión.

Figura 18. Página de perfil del usuario.

- **/schedule:** Esta página utiliza la plantilla *register.html* de la carpeta *Schedule*, implementa la función de registro de un objeto de tipo Schedule.

Figura 19. Página de registro de objeto Schedule del usuario.

- **/alarm:** Esta página utiliza la plantilla *register.html* de la carpeta *Alarm*, implementa la funcionalidad de registro de un objeto de tipo Alarm.

Figura 20. Página de registro de objeto Alarm del usuario.

- **/schedules:** Esta página utiliza la plantilla *all_schedules.html* de la carpeta *Schedules*, implementa la funcionalidad de listar todos nuestros objetos de tipo Schedule.

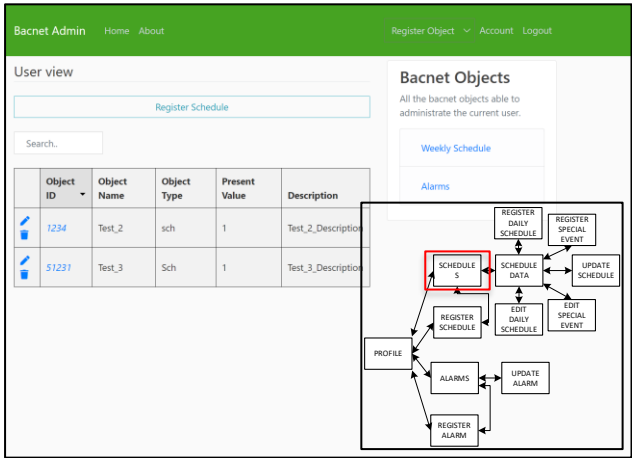


Figura 21. Página de los objetos Schedules del usuario.

- **/schedule/{id_schedule}:** Esta página utiliza la plantilla *schedules.html* de la carpeta *Schedules*, implementa la funcionalidad de administrar los eventos especiales y los eventos semanales relacionados al objeto.

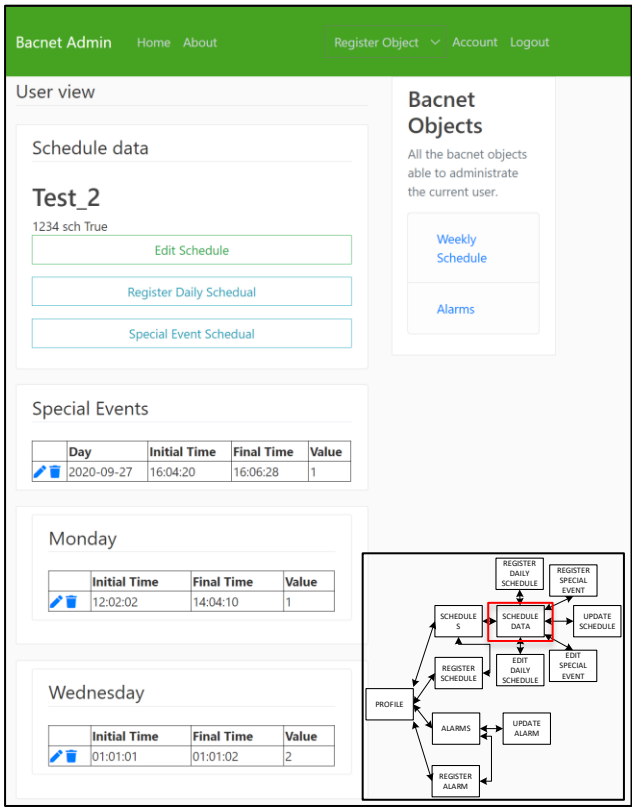


Figura 22. Página de los datos de un objeto Schedule.

- **/schedule/{id_schedule}/update:** Esta página utiliza la plantilla *register.html* de la carpeta *Schedule*, implementa la funcionalidad de editar los atributos de nuestro objeto Schedule. Utiliza la misma plantilla que la página de registro, se ha utilizado Jinja2 para hacer las modificaciones necesarias.

Figura 23. Página de actualización de objeto Schedule.

- **/schedule/{id_schedule}/daily_sch:** Esta página utiliza la plantilla *new_daily_schedule.html* de la carpeta *Schedule*, implementa la funcionalidad de registro de un evento semanal.

Figura 24. Página de registro de evento semanal.

- **/schedule/{id_schedule}/daily_sch/{id_daily_sch}**: Esta página utiliza la plantilla *update_daily_schedule.html* de la carpeta *schedule*, implementa la funcionalidad de modificar los valores asociados a un evento semanal.

Figura 25. Página de actualización de evento semanal.

- **/schedule/{id_schedule}/special_sch**: Esta página utiliza la plantilla *new_daily_schedule.html* de la carpeta *Schedule*, implementa la funcionalidad de registrar eventos especiales. Utiliza la misma plantilla que se usa para eventos semanales y se utiliza Jinja2 para hacer las modificaciones necesarias.

Figura 26. Página de registro de eventos especiales.

- **/schedule/{id_schedule}/special_sch/{id_special_sch}**: Esta página utiliza la plantilla *update_daily_schedule.html* de la carpeta *schedule*, implementa la funcionalidad de modificar los valores asociados a un evento especial. Utiliza la misma plantilla que los eventos semanales y se utiliza Jinja2 para hacer las modificaciones necesarias.

Figura 27. Página de actualización de eventos especiales.

- **/alarms**: Esta página utiliza la plantilla *all_alarms.html* de la carpeta *Alarm*, implementa la funcionalidad de listar todos los objetos de tipo Alarm asociados al usuario.

Id	State	Event	Source	Source Description
6	Active	Normal	1234	Test
2	Inactive	Fault	SCH25	Las alarmas no estan relacionados a objetos en la BD

Figura 28. Página de los objetos Alarm del usuario.

- **/alarm/{id_alarm}/update:** Esta página utiliza la plantilla *register.html* de la carpeta *Alarm*, implementa la funcionalidad de modificar los datos relacionados a un objeto Alarm. Se utiliza la misma plantilla que en el caso de registro se modificará con Jinja2.

Figura 29. Página de actualización de objeto Alarm.

3.3.2.2.3 Nivel acceso dos

- **/admin/users:** Esta página utiliza la plantilla *all_users.html* de la carpeta *Admin*, implementa la funcionalidad de listar todos los usuarios registrados en la aplicación.

Figura 30. Página de los usuarios registrado.

- **/admin/users/{id_user}**: Esta página utiliza la plantilla *user.html* de la carpeta *Admin*, implementa la funcionalidad de editar los datos de un usuario cualquiera, así como asignar permisos de administrador a dicho usuario.

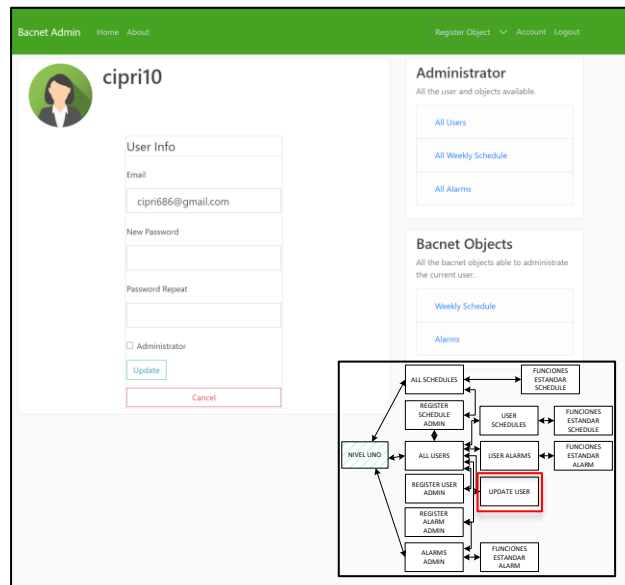


Figura 31. Página para actualizar datos de usuario.

- **/admin/users/{id_user}/schedules**: Esta página utiliza la plantilla *all_schedules.html* de la carpeta *Schedule*, implementa la funcionalidad de listar los objetos de tipo Schedule relacionados a un usuario seleccionado.

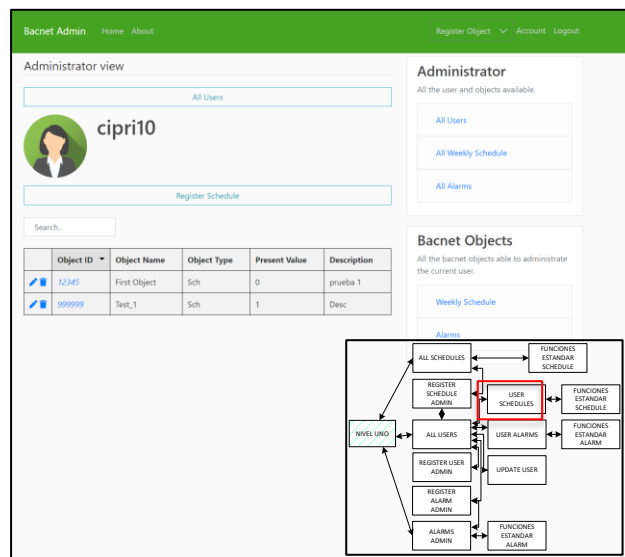


Figura 32. Página de los objetos Schedules del usuario seleccionado.

- **/admin/users/{id_user}/alarms:** Esta página utiliza la plantilla *all_alarm.html* de la carpeta Alarm, implementa la funcionalidad de listar los objetos de tipo Alarm relacionados a un usuario seleccionado.

The screenshot shows the 'Administrator view' for user 'cipri10'. The page has a green header with 'Bacnet Admin', 'Home', 'About', 'Register Object', 'Account', and 'Logout'. Below the header, there's a search bar and a 'Register alarm' button. A table lists alarms with columns: Id, State, Event, Source, and Source Description. The table has two rows: one with 'Inactive' state and 'SCH1' source, and another with 'Active' state and 'SCH1' source. To the right, there's a sidebar with 'Administrator' and 'Bacnet Objects' sections. At the bottom right, a diagram shows the system architecture with 'USER ALARMS' highlighted in a red box.

Figura 33. Página de los objetos Alarm del usuario seleccionado.

- **/admin/schedules:** Esta página utiliza la plantilla *all_schedules.html* de la carpeta Admin, implementa la funcionalidad de listar todos los objetos de tipo Schedule registrados en la aplicación.

The screenshot shows the 'Administrator view' for user 'cipri10'. The page has a green header with 'Bacnet Admin', 'Home', 'About', 'Register Object', 'Account', and 'Logout'. Below the header, there's a search bar and a 'Register Schedule' button. A table lists schedules with columns: Object ID, Object Name, Object Type, Present Value, Description, and User. The table has five rows with various object IDs and descriptions. To the right, there's a sidebar with 'Administrator' and 'Bacnet Objects' sections. At the bottom right, a diagram shows the system architecture with 'ALL SCHEDULES' highlighted in a red box.

Figura 34. Página de los objetos Schedules del sistema.

- **/admin/alarms:** Esta página utiliza la plantilla *all_alarms.html* de la carpeta Admin, implementa la funcionalidad de listar todos los objetos de tipo Alarms registrados en la aplicación.

Figura 35. Página de los objetos Alarm del sistema.

- **/admin/schedules/register:** Esta página utiliza la plantilla *regsiter.html* de la carpeta Schedule, implementa la función de registro de un objeto de tipo Schedule pero en este caso permitiendo la elección del usuario al que se le asigna.

Figura 36. Página de registro de objeto Schedules.

- **/admin/Alarms/register:** Esta página utiliza la plantilla *register.html* de la carpeta *Alarm*, implementa la función de registro de un objeto de tipo Alarm pero en este caso permitiendo la elección del usuario al que se le asigna.

The screenshot shows the 'Register alarm' page in the Bacnet Admin interface. The page has a green header with 'Bacnet Admin', 'Home', 'About', 'Register Object', 'Account', and 'Logout'. The main content area is titled 'Administrator view' and contains a 'Register alarm' form. The form has the following fields: 'User' (dropdown menu with 'cipri10' selected), 'State' (dropdown menu with 'Active' selected), 'Event' (dropdown menu with 'Normal' selected), 'Source' (text input), 'Source Description' (text input), 'Alarm Text' (text input), and 'Triggered Time' (text input). There are 'Register' and 'Cancel' buttons at the bottom of the form. On the right side, there are two panels: 'Administrator' with links for 'All Users', 'All Weekly Schedule', and 'All Alarms'; and 'Bacnet Objects' with links for 'Weekly Schedule' and 'Alarms'. An inset diagram in the bottom right corner shows a flowchart of the system architecture. The diagram includes boxes for 'NEW USER', 'REGISTER SCHEDULE ADMIN', 'ALL SCHEDULES', 'REGISTER SCHEDULE ADMIN', 'ALL USERS', 'REGISTER USER ADMIN', 'REGISTER ALARM ADMIN', 'ALARMS ADMIN', 'FUNCTIONES ESTANDAR SCHEDULE', 'USER SCHEDULES', 'USER ALARMS', 'UPDATE USER', and 'FUNCTIONES ESTANDAR ALARM'. The 'NEW USER' and 'REGISTER SCHEDULE ADMIN' boxes are highlighted with a red border.

Figura 37. Página de registro de objeto Alarm.

- **/admin/users/signup:** Esta página utiliza la plantilla *new_user.html* de la carpeta *Admin*, implementa la funcionalidad de registro de un nuevo usuario permitiendo asignarle permisos de administrador.

The screenshot shows the 'New User' page in the Bacnet Admin interface. The page has a green header with 'Bacnet Admin', 'Home', 'About', 'Register Object', 'Account', and 'Logout'. The main content area is titled 'New User' and contains a form for registering a new user. The form has the following fields: 'Username' (text input), 'Email' (text input), 'Password' (text input), 'Password Repeat' (text input), and a checkbox labeled 'Administrator'. There is a 'Sign up' button at the bottom of the form. On the right side, there are two panels: 'Administrator' with links for 'All Users', 'All Weekly Schedule', and 'All Alarms'; and 'Bacnet Objects' with links for 'Weekly Schedule' and 'Alarms'. An inset diagram in the bottom right corner shows a flowchart of the system architecture. The diagram includes boxes for 'NEW USER', 'REGISTER SCHEDULE ADMIN', 'ALL SCHEDULES', 'REGISTER SCHEDULE ADMIN', 'ALL USERS', 'REGISTER USER ADMIN', 'REGISTER ALARM ADMIN', 'ALARMS ADMIN', 'FUNCTIONES ESTANDAR SCHEDULE', 'USER SCHEDULES', 'USER ALARMS', 'UPDATE USER', and 'FUNCTIONES ESTANDAR ALARM'. The 'NEW USER' and 'REGISTER SCHEDULE ADMIN' boxes are highlighted with a red border.

Figura 38. Página de registro de nuevo usuario.

3.4 Despliegue

Como se ha comentado en apartados anteriores, para facilitar el despliegue de la aplicación web se va a generar un Docker, así como un fichero, *Dockerfile* para poder generar el mismo, sin necesidad hacer uso de Docker Hub.

El contenedor contendrá los paquetes desarrollados, así como instalados el servidor Nginx y Gunicorn además de las configuraciones necesarias para el despliegue. Además, tendrá instalado un servidor SSH para permitir la administración remota.

En su configuración inicial de la aplicación se ha creado un usuario *root* administrador que deberá cambiar su contraseña vía correo electrónico para poder ser utilizado, el correo al que se enviará la URL para este cometido se especificará en el fichero *docker-entrypoint.sh*, en caso de generar el contenedor sin utilizar Docker, en caso contrario se deberá cambiar el correo directamente en la base de datos ya que por defecto se ha generado del contenedor con un correo de ejemplo.

A continuación, vamos a ver las configuraciones necesarias para el funcionamiento de los servidores Nginx y Gunicorn (Schafer, Desarrollo de aplicación Flask., s.f.).

Para configurar el servidor Nginx se ha creado el fichero de configuración *bacnetweb* en la carpeta con ruta */etc/nginx/sites-enabled*. En este fichero se han especificado la dirección y el puerto donde atiende este servidor peticiones, la dirección de los ficheros estáticos y la dirección del servidor Gunicorn que atenderá las peticiones dinámicas. Teniendo este fichero el contenido representado en la *Figura 40*.

```
server {
    listen 80;
    server_name 10.0.0.15;
    access_log /var/log/nginx/access.log;

    location /static {
        alias /home/ubuntu/TFG/BacnetWeb/bacnetweb/static;
    }

    location / {
        proxy_pass http://localhost:8000;
        include /etc/nginx/proxy_params;
        proxy_redirect off;
    }
}
```

Figura 39. Contenido fichero de configuración de Nginx.

Para la ejecución del servidor Gunicorn se ha tenido que crear un demonio, de esta manera se ejecuta en segundo plano estando supervisado por el sistema volviendo a arrancarlo en caso de fallo. Para ello se ha creado un fichero *bacnetweb.service* en la carpeta con ruta */etc/systemd/system*. En este fichero se indican información sobre el servicio, así como la dirección donde debe escribir este demonio los registros o el comando que va a ejecutar. Al no estar ejecutado en segundo plano también se deberán declarar las variables de entorno que va a utilizar. El contenido de este fichero se refleja en la *Figura 41*.

```
# ***bacnetweb.service***
[Unit]
Description=Servicio que se ocupa de tener ejecutando gunicorn
After=multi-user.target

[Service]
Type=simple
ExecStart=/usr/local/bin/gunicorn -w 3 run:app
User=ubuntu
WorkingDirectory=/home/ubuntu/TFG/BacnetWeb
Restart=on-failure
StandardOutput=syslog
StandardError=syslog
Environment="FLASK_APP=__init__.py"
Environment="FLASK_ENV=development"
Environment="DATABASE_URL='postgres://tfg:tfq@localhost:5432/tfg'"
Environment="USER_GMAIL='correo@gmail.com'"
Environment="PASS_GMAIL='correo_pass'"
Environment="LC_ALL=C.UTF-8"
Environment="LANG=C.UTF-8"

[Install]
WantedBy=multi-user.target
```

Figura 40. Contenido fichero de configuración demonio bacnetweb.service .

4 Resultados y conclusiones

En primer lugar, me ha resultado interesante aprender cómo se debe desarrollar una aplicación completa tocando todos los puntos, desde la organización, diseño, implementación y despliegue siendo así uno de los fines de mi preparación académica y posibilitando aplicar esto en un entorno laboral.

En segundo lugar, me gustaría valorar la experiencia como muy positiva ya que, aun teniendo un trabajo de tal extensión sin una guía clara, las metodologías ágiles han sido un recurso muy útil para desarrollar la carga de trabajo sin mayores imprevistos.

También me gustaría poner en valor los conocimientos adquiridos al realizar la documentación de un proyecto de tanta extensión, tanto desde el punto de vista de validar la información como desde el uso de las herramientas de ofimática de Microsoft tan potentes e importantes hoy en día.

En lo relativo a los lenguajes y el software utilizado me gustaría destacar lo interesante que me ha resultado aprender un lenguaje como Python que tenía pendiente desde hace mucho tiempo y que hoy en día es uno de los más extendidos. También me ha resultado de gran interés aprender sobre el software Docker, aunque lo haya utilizado con anterioridad, gracias a este proyecto he conseguido profundizar más en su uso.

Y finalmente, como se ha visto a lo largo de esta memoria se ha conseguido cumplir con los objetivos marcado, desarrollando así una aplicación web completa para la administración de objetos BACnet con una interfaz amigable y fácilmente utilizable.

4.1 Líneas futuras

La amplitud de este proyecto ofrece varias posibilidades que se podrían desarrollar en líneas de trabajo futuras, tales como la implementación de nuevas vistas que permitiesen la administración de más objetos BACnet ya que hay una gran variedad o la utilización de la librería BACnet para escribir y leer de la base de datos para así conectar la interfaz web a los objetos a través de la base de datos, creando un sistema plenamente funcional; por mencionar algunas.

5 Bibliografía

- BACnet Explorer*. (s.f.). Obtenido de <https://www.inneasoft.com/en/bacnet-explorer/>
- BACnet Stack*. (s.f.). Obtenido de <http://bacnet.sourceforge.net/>
- Bootstrap_(front-end_framework)*. (s.f.). Obtenido de [https://en.wikipedia.org/wiki/Bootstrap_\(front-end_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))
- Buscador datos en la tabla con Javascript*. (s.f.). Obtenido de <https://www.tutofox.com/javascript/buscador-datos-en-la-tabla-con-javascript/>
- Configurar NGIX para FLASK*. (s.f.). Obtenido de <https://www.patricksoftwareblog.com/how-to-configure-nginx-for-a-flask-web-application/>
- Crear - Gitignore file*. (s.f.). Obtenido de <https://www.lifewire.com/gitignore-file-4153363>
- Descripción de diagrama de Gantt*. (s.f.). Obtenido de https://es.wikipedia.org/wiki/Diagrama_de_Gantt
- Docker_(software)*. (s.f.). Obtenido de [https://es.wikipedia.org/wiki/Docker_\(software\)](https://es.wikipedia.org/wiki/Docker_(software))
- Flask_(web_framework)*. (s.f.). Obtenido de [https://en.wikipedia.org/wiki/Flask_\(web_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework))
- Git*. (s.f.). Obtenido de <https://es.wikipedia.org/wiki/Git>
- GitHub*. (s.f.). Obtenido de <https://es.wikipedia.org/wiki/GitHub>
- Hoja_de_estilos_en_cascada*. (s.f.). Obtenido de https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada
- Implementación de sistema de correo en aplicación*. (s.f.). Obtenido de https://www.tutorialspoint.com/flask/flask_mail.htm
- Joskowicz, J. (s.f.). *eXtremme Programming*. Obtenido de <https://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf>
- Lenguaje_de_marcado*. (s.f.). Obtenido de https://es.wikipedia.org/wiki/Lenguaje_de_marcado
- Librería WTFForm*. (s.f.). Obtenido de <https://wtforms.readthedocs.io/en/2.3.x/>
- Message Flashing Library*. (s.f.). Obtenido de <https://flask.palletsprojects.com/en/1.1.x/patterns/flashing/>
- Metodología Agile*. (s.f.). Obtenido de <https://blogs.informacion.com/blog/recursos-humanos/recursos-humanos/metodologia-agile-que-es-y-como-aplicarla-a-la-empresa/>
- PostgreSQL*. (s.f.). Obtenido de <https://es.wikipedia.org/wiki/PostgreSQL>
- Que es NGINX*. (s.f.). Obtenido de <https://es.wikipedia.org/wiki/Nginx>
- Que es WSGI*. (s.f.). Obtenido de <https://wsgi.readthedocs.io/en/latest/what.html>
- Ronacher, A. (n.d.). *Flask Website*. Retrieved from <https://palletsprojects.com/p/flask/>

- Salvatierra, J. (s.f.). *Generacion de decorators*. Obtenido de <https://blog.tecladocode.com/learn-python-defining-user-access-roles-in-flask>
- Schafer, C. (s.f.). *Desarrollo de aplicación Flask*. Obtenido de <https://www.youtube.com/watch?v=MwZwr5Tvyxo&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH&index=1>
- Schafer, C. (s.f.). *Factorizar aplicación en paquetes*. Obtenido de <https://www.youtube.com/watch?v=44PvX0Yv368&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH>
- Tutorial BACnet*. (s.f.). Obtenido de <http://www.bacnet.org/Tutorial/HMN-Overview/sld001.htm>
- Tutorial login flask*. (s.f.). Obtenido de <https://www.youtube.com/watch?v=K0vSCCAM2ss>
- Website oficial de Guicorn*. (s.f.). Obtenido de <https://gunicorn.org/>
- Yet Another Bacnet Explore*. (s.f.). Obtenido de <https://sourceforge.net/projects/yetanotherbacnetexplorer/>
- Yu, B. (s.f.). *CS50's Web Programming with Python and*. Obtenido de <https://video.cs50.io/j5wysXqaIV8?screen=zSLVuchPWWI>