Facultad
de
Ciencias

PLATAFORMA WEB DE GESTIÓN DE MEDIDAS CLÍNICAS

(Web-platform for the management of clinical measures)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERIA INFORMATICA

Autor: Jorge García Díaz

Director: Olga Maria Conde Portilla

Co-Director: Jose Alberto Gutiérrez Gutiérrez

Julio – 2021

Resumen

Este Trabajo de Fin de Grado tiene como objetivo la maquetación y el desarrollo de una aplicación web, tanto en su capa *front-end* como en la de *back-end* para la gestión de medidas clínicas. Dicha aplicación está destinada a la obtención de medidas de diferentes instrumentos de imagen óptica que adquieren imágenes, entre otras, de tejido biológico y de material agroalimentario.

La aplicación también permitirá almacenar las diferentes medidas realizadas en una base de datos desarrollada para ello. Posteriormente, se visualizarán las diferentes imágenes y medidas accediendo de forma remota a la instrumentación, con la finalidad de realizar el proceso de gestión más sencillo y cómodo. Dado que las imágenes a obtener tienen un tamaño elevado, que puede llegar a los 20GB, su gestión es complicada. Por tanto, se busca que la plataforma sea sencilla de usar y que tenga una rápida velocidad de respuesta, ya que el principal objetivo es, facilitar la gestión de las diferentes medidas.

Para ello, se utilizará como lenguaje principal Python y como *framework* Django, lo que otorgará una gran variedad de opciones para desarrollar la plataforma, además de proporcionar unas funcionalidades básicas que hacen que el proceso de desarrollo sea más sencillo.

Palabras clave

Aplicación Web, medidas clínicas, Python, Django, base de datos

AGRADECIMIENTOS: Este trabajo ha sido realizado gracias a los medios proporcionados por el proyecto INTRACARDIO (DTS17/00055) financiado por el ISCIII, la Fundación Instituto de Investigación Valdecilla (IDIVAL) y cofinanciado con fondos FEDER.

Abstract

The goal of this TFG is to layout and to develop a web application for the management of clinical measures, developing both the front-end and the back-end layers. This application is intended to obtain measurements from different optical imaging instruments that acquire images, among others, of biological tissue and agri-food material.

The application will also store the different measurements in a database developed for it. Afterwards, the different images and measurements will be remotely visualized by accessing the instrumentation, to make the management process easier and usable. Since the images to be obtained have a large size, which can reach 20GB, their management is complicated. Therefore, the aim for the platform is to be simple to use and to have a fast response speed, favoring and facilitating the management of the different campaign measurements.

To this end, Python will be used as the main language and Django as the framework. This election will provide a wide variety of options and basic functionalities to develop the platform that will make the development process easier.

Key words

Web Application, Clinical measures, Python, Django, Data bases

Índice

1- Introducción al trabajo

1.1 - Introducción

El desarrollo de este trabajo va a estar basado en las plataformas llamadas Paas (*Platform as a Service*), esto significa plataforma como servicio, y hace referencia a un entorno de desarrollo situado en la red, con una serie de aplicaciones ubicadas en él que permiten al usuario acceder a estas y desarrollar las actividades necesarias. En este caso, vamos a desarrollar una plataforma adaptada a la gestión de medidas e instrumentos del Laboratorio de Ingeniería Fotónica que realiza medidas de tipo biomédico.

Una gran parte de las empresas dedicadas a la medicina o salud utilizan plataformas web. Esto son espacios ubicados en Internet, que permiten ejecutar diversos programas para satisfacer las diferentes necesidades de medida que puedan aparecer. Esta plataforma nos va a permitir realizar conexiones remotas a los diferentes instrumentos, así como observar y descargar las diferentes medidas realizadas por estos.

Estas plataformas son esenciales debido a que permiten realizar actividades, que tendrían un coste de tiempo elevado, de una forma más sencilla y rápida. Este tipo de infraestructuras son necesarias debido a que hoy en día se ha aumentado el número de datos que se deben procesar, como menciona la empresa DataIQ [1], “El 90% de los datos mundiales ha sido creado en los últimos dos años”, y por ello la gestión manual de ellos conlleva una gran pérdida de tiempo. Por otra parte, la gran mayoría de los datos tienen relación entre ellos, y mediante este tipo de plataformas se consigue establecer relaciones entre ellos de forma más sencilla y automática.

Hoy en día, este tipo de plataformas están en franco desarrollo, ya que mejoran de forma efectiva la gestión de todos los datos que pueden tener las diferentes clínicas, y en la sociedad actual la velocidad y la eficacia es esencial.

Algunos ejemplos de este tipo de plataformas pueden ser: BioMax [2], Meplis [3], Meddsocial [4]. Todas tienen una finalidad diferente, pero todas ellas se basan en los

mismo, una plataforma web que les permita gestionar todos sus datos de una manera rápida y eficiente.

En definitiva, para adaptarse a los tiempos actuales y poder gestionar el gran número de datos que se obtienen hoy en día, nos vemos obligados a utilizar este tipo de herramientas de gestión distribuida de medidas.

Para finalizar, en la figura 1 se puede apreciar un esquema de la estructura física sobre la que se desarrollará la plataforma con sus diferentes equipos y BBDD. Más adelante en el capítulo 3 se mostrará en aspecto lógico.

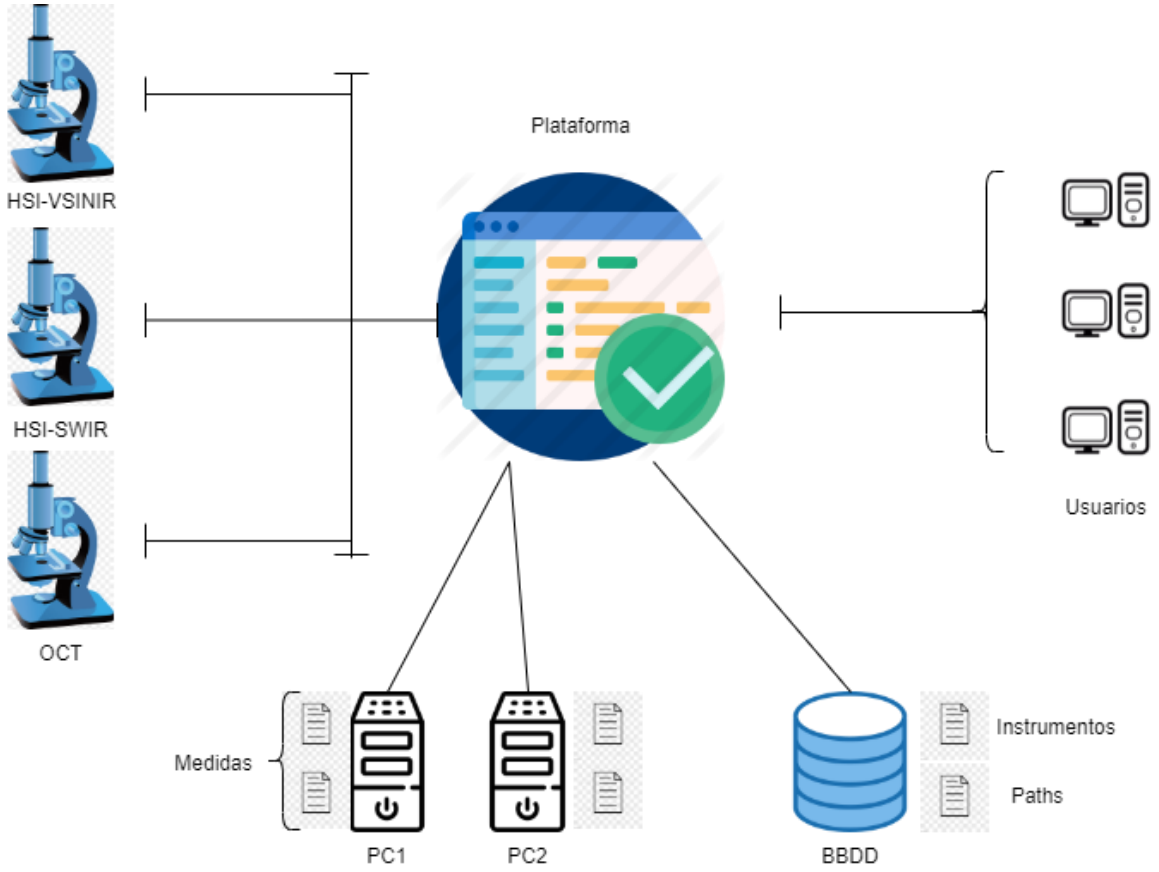


Figura 1. Representación física de la plataforma

1.2 - Objetivos

El objetivo general de este trabajo es desarrollar una plataforma para la gestión remota del equipamiento de imagen existente en el laboratorio de investigación del Grupo de Ingeniería Fotónica que pudiera trasladarse al ámbito clínico. Los requisitos de la plataforma deben incluir el ser fácil de usar y tener unos tiempos de gestión aceptables para poder realizar las acciones de una manera rápida y eficaz.

Centrándonos en las características específicas de la plataforma, aparecen los siguientes objetivos específicos:

- Obtener medidas de los diferentes instrumentos de laboratorio.
- Almacenar las medidas realizadas en una base de datos creada para ello.
- Conocer el estado y realizar una conexión a distancia sobre los diferentes instrumentos mediante un sistema de VNC (*Virtual Network Computing*).
- Tener la posibilidad de visualizar, descargar y transferir las medidas obtenidas, del instrumento a la base de datos.
- Navegar entre las diferentes rutas del sistema para buscar los archivos de medidas.
- Desarrollar un sistema de autentificación con registro e inicio de sesión, para poder acceder a la plataforma y, dependiendo del usuario, diferenciar las operaciones que puede realizar en esta.

La funcionalidad de la plataforma será: iniciar sesión en la web y ver el estado de todos los instrumentos; posteriormente, se accederá de manera remota al instrumento deseado y se realizará una medida; finalmente, se accedería al apartado de medidas generadas por el instrumento y se descargaría la misma para su posterior visualización.

1.3 - Metodología y herramientas de trabajo

En esta sección se describe la metodología de desarrollo utilizada en la implementación de la plataforma, además de las diferentes herramientas que se han empleado para el correcto funcionamiento de la misma.

1.3.1 - Metodología de desarrollo

Una metodología de desarrollo es aquella que ofrece una serie de procedimientos con la finalidad de alcanzar los diferentes objetivos que se han establecido antes de comenzar.

Para la realización de este trabajo, se ha utilizado una metodología de desarrollo incremental que consiste en la división del proyecto en diferentes etapas, que se van

realizando una tras otra. Es decir, el proyecto se divide en pequeñas fases funcionales, que se van desarrollando de forma consecutiva y de forma separada.

Ventajas

- No se debe esperar hasta el final del proyecto para comprobar el funcionamiento del sistema.
- Se minimiza el riesgo de fallo, ya que se van realizando comprobaciones del comportamiento en cada etapa de desarrollo.
- Al dividir el proyecto en diferentes partes de menor tamaño, se facilita el desarrollo de estas.

En este caso el paso a paso que hemos seguido era, poníamos en común el siguiente paso a desarrollar de la plataforma, se comenzaba su desarrollo y en caso de que hubiese alguna duda de consultaba. Una vez que se daba por finalizado se ponía en común en el repositorio, y en caso de encontrar algún fallo, se comunicaba y posteriormente se intentaba solucionar. Una vez acabada la parte elegida, se volvía a elegir el siguiente paso de desarrollo. Y así hasta la finalización de la plataforma [5].

1.3.2 - Tecnologías

En este desarrollo de este TFG se ha trabajado con diferentes tecnologías del ámbito de la ingeniería informática que se enumeran y describen brevemente en esta sección.

Python

Como lenguaje de programación principal se ha usado Python, ya que nos permite realizar una gran variedad de acciones de manera sencilla y rápida. En el siguiente capítulo se describirá de forma más extensa, explicando sus características y justificando su elección [6].

Django

Como *framework* de desarrollo se ha elegido Django, debido a su relación con Python y las diferentes características que vienen implementadas en él y que facilitan el desarrollo de la plataforma. También se describirá más extensamente en el capítulo 2 [7].

Javascript

Se trata de un lenguaje de programación interpretado, que sirve para realizar acciones sobre el HTML generado en la plataforma. Se ha utilizado principalmente para realizar cambios visuales en la plataforma al realizar diferentes acciones [8].

FileTree

Consiste en una librería de Javascript que se ha utilizado para la gestión de medidas. Esta librería permite que, una vez hayamos seleccionado el *path*, se despliegue un árbol con los diferentes directorios [9].

SQLIte

Es el sistema de gestión de bases de datos utilizada en este caso para crear la base de datos (BBDD) sobre la que actúa la plataforma [10].

PyCharm

Como entorno de desarrollo se ha utilizado PyCharm. Ya que es un entorno que está bien optimizado para su uso con Python, y además permite subir los cambios directamente a GITHUB [11]. GITHUB es un sistema de control de versiones, que sirve para alojar proyectos, se utiliza para trabajar conjuntamente en creación de código fuente principalmente.

DB Browser for SQLite

Se trata de una herramienta de acceso abierto, *open source*, que permite crear, diseñar y editar archivos de bases de datos, compatibles con SQLite [12].

Git

Es un sistema de control de versiones, mediante el cual se puede ir poniendo en común el trabajo realizado. Se ha utilizado para poner en común el desarrollo de la plataforma [13].

VNC

Se trata de un programa de software libre, de estructura cliente-servidor, empleado para realizar una conexión remota entre el instrumento al que se desea acceder, y el PC desde que accedemos a la web [14].

NoVnc

Forma parte del sistema VNC creado para realizar la conexión, y actúa en forma de cliente en el momento de la conexión. Para que funcione debe haber un servidor de VNC ejecutándose en la máquina [15].

Gunicorn

Servidor web HTTP, que sirve para poder desplegar la plataforma creada, ya que permite realizar despliegues desarrollados en el *framework* de Django [16].

Apache

Se trata de un servidor HTTP, que en este caso se va a usar para crear un servidor proxy para que cuando se realice la conexión desde fuera, ésta funcione correctamente. Para esto se deben activar diferentes módulos que vienen integrados en la aplicación [17].

JSON

Es un formato de texto (JSON, *JavaScript Object Notation*) que se utiliza en el intercambio de datos, que tiene las ventajas de ser sencillo de leer y escribir, además de ser fácil de interpretar y generar. En esta plataforma trabajaremos con este formato para conocer las diferentes propiedades de las medidas.

Poniendo en conjunto todas las tecnologías, se utiliza Pycharm como interfaz gráfica para crear la plataforma usando el *framework* de Django que está desarrollado en Python. Esta plataforma hace uso de una BBDD creada en SQLite donde se almacenan los datos relacionados con la web, como pueden ser los instrumentos. Estos últimos son accedidos a través de un proxy desarrollado en Apache, mediante la tecnología VNC que nos permite realizar una conexión remota. Estos instrumentos realizaran medidas a las que accederemos desde la web, y cuya información estará contenida en un archivo JSON. Todo esto estará desplegado mediante Gunicorn.

1.4 - Estructura del documento

La presente memoria está distribuida en 5 capítulos. En el presente capítulo 1 se ha descrito la motivación, contexto, objetivos y tecnología empleada en el desarrollo del TFG.

En el capítulo 2 se ha abordado todo lo relacionado a los materiales y métodos utilizados a la hora de desarrollar la plataforma, como es el lenguaje de programación, framework, los diferentes instrumentos del laboratorio y cómo se realizan las comunicaciones.

En el capítulo 3 se realiza una explicación de como se ha desarrollado la plataforma. Se hace una descripción de cómo es la estructura de la aplicación, así como las diferentes gestiones de esta, como puede ser la gestión de usuarios, de instrumentos o de medidas.

Posteriormente, en el capítulo 4 se ha desarrollado lo relacionado a evaluación y pruebas. Realizaremos un análisis del rendimiento y la seguridad de la web, además de analizar la escalabilidad de la plataforma.

Por último, en el capítulo 5 se mostrarán las diferentes conclusiones que se han obtenido en el desarrollo del TFG, además de analizar los posibles trabajos futuros que se pueden realizar sobre el trabajo.

2 - Materiales y métodos

En este capítulo, se realizará una descripción más profusa de las diferentes herramientas empleadas en el desarrollo de la plataforma y se justificará su elección destacando las ventajas proporcionadas.

2.1 - Python

2.1.1 - Definición

Python es un lenguaje de programación interpretado cuya base es la legibilidad. Es de tipo multiparadigma ya que soporta diferentes tipos de programación, como puede ser la orientada a objetos o la programación funcional. Actualmente se usa en programas de todo tipo, especialmente en Inteligencia Artificial. Durante estos últimos años su uso se ha ido volviendo más popular debido al auge del big data, ya que es el lenguaje más popular en este ámbito.

2.1.2 - Ventajas e inconvenientes

En general Python es un lenguaje que ofrece más ventajas que desventajas. En este apartado vamos a diferenciar entre ambas, para dar una vista general de por qué hemos elegido Python y no cualquier otro lenguaje.

A continuación, se enumeran las ventajas:

- Permite crear programas de cualquier tipo.
- Código abierto, *open source*.
- Se puede usar en la mayoría de los sistemas operativos (SO).
- Es de comprensión y aprendizaje sencillos.
- Multiparadigma.
- Ofrece un gran número de librerías.

Desventajas de Python:

- Ejecución más lenta respecto a otros lenguajes como pueden ser C, Perl o Ruby, muy similar a PHP y más rápido que Java.

- El desarrollo web es complicado, por lo que necesita integrarse con *frameworks* como puede ser Django.

Como se puede observar, el número de ventajas supera por mucho el de desventajas.

Además de todo esto, las librerías de procesado que se utilizan en la plataforma están programadas en Python por lo que resulta bastante más sencillo si se usa el mismo lenguaje. Por todas estas cosas hemos elegido Python como lenguaje.

2.2 - Framework

Un *framework* es un marco de trabajo que se utiliza por los programadores para realizar el desarrollo software. Estos *frameworks* permiten agilizar los diferentes procesos ya que contienen una serie de herramientas y módulos que se pueden reutilizar para distintos proyectos.

Tienen diferentes ventajas como pueden ser:

- Ahorra tiempo ya que ofrece un esqueleto para la aplicación.
- Están ampliamente extendidos.
- Proporcionan seguridad, ya que las vulnerabilidades más comunes ya vienen resueltas.

Otros *frameworks* interesantes para el desarrollo web en Python pueden ser:

- Pyramid: minimalista, rápido y fiable.
- Bottle: simple y proporciona un mínimo de herramientas.
- Flask: ágil, no tiene estructura de la que partir.

Se ha decidido utilizar como *framework* Django, ya que está desarrollado en Python y tiene una serie de características y librerías que facilitará el desarrollo. Además, en comparación con los demás *frameworks*, es bastante más potente, tiene una comunidad muy grande y nos permite realizar una administración más sencilla debido a la interfaz que implementa.

2.2.2 - Django

Descripción

Se trata de un *framework* destinado al desarrollo web y de código abierto. Su lenguaje de programación es Python y sigue un patrón de diseño MVC también llamado Modelo-Vista-Controlador. Este tipo de arquitectura se basa en separar los diferentes componentes del software, para que estos sean independientes entre sí. Para poder realizar esto, el patrón divide la arquitectura en tres componentes:

1. Modelo: que se encarga de la lógica de y los datos del sistema.
2. Vista: que está destinado a la interfaz de usuario y los mecanismos de interacción con este.
3. Controlador: que hace de intermediario entre las otras dos partes.

La meta de este *framework* es, ayudar a la creación de sitios web con una alta complejidad. Dispone de una gran cantidad de características que se explican posteriormente, que han facilitado el desarrollo en gran medida.

Funcionamiento

Como toda aplicación web, el servidor espera peticiones HTTP. Una vez recibida la petición, la plataforma elabora una respuesta basándose en la URL y en la información contenida en los datos POST o GET. Dependiendo de la necesidad que se tenga, la plataforma puede leer o escribir información desde una base de datos u otras tareas que se requieran para la petición.

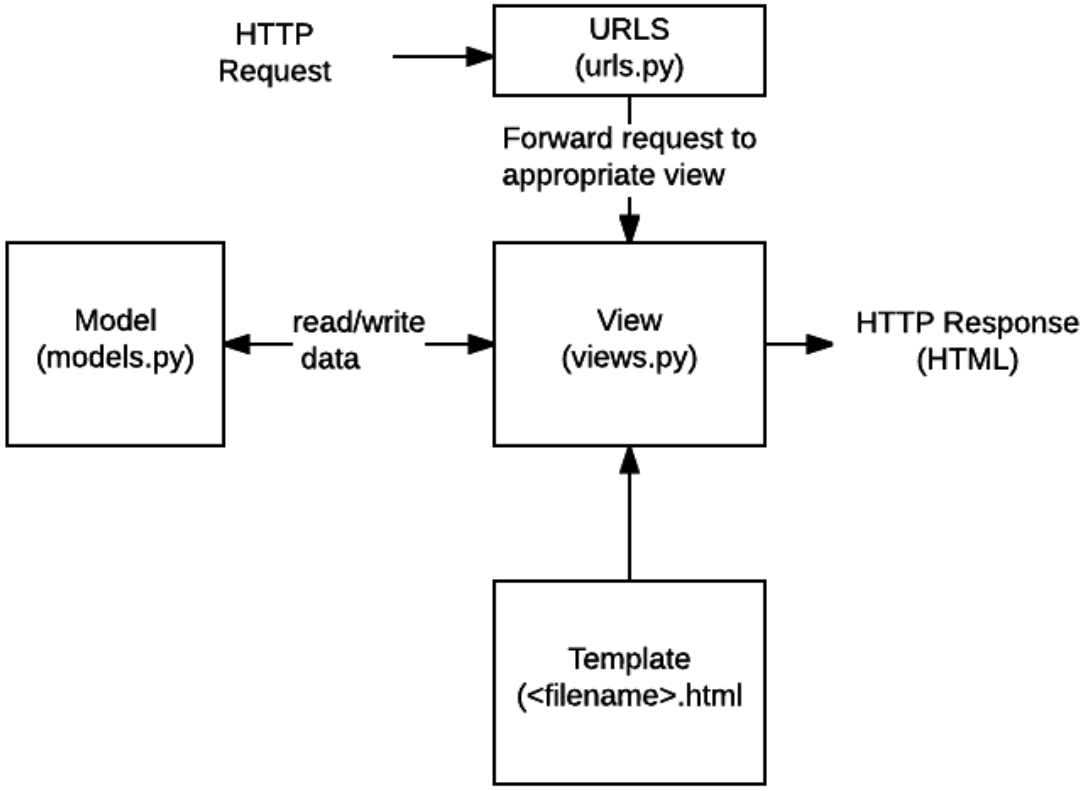


Figura 2. Esquema de funcionamiento de una plataforma Django.

En la figura 2 se puede apreciar la estructura básica que sigue una plataforma desarrollada por Django. A continuación, se describirán cada una de las partes:

- ***URLS***: Este archivo permite indicarle a la plataforma qué vista pertenece a las diferentes `urls` establecidas. Digamos que, mediante este archivo, la plataforma realiza un mapeo de la `url` establecida, a la vista a la que está asociada. Este mapeador, también permite introducir diferentes parámetros en la `url` para trabajar con ellos en la vista.

- ***View***: La vista, permite gestionar las diferentes peticiones `http` que se reciben, creando una respuesta HTTP. Las vistas pueden acceder a los diferentes datos almacenados en las BBDD y pasárselos a las plantillas que le darán forma a la respuesta.

- ***Modelos***: Estos son objetos de Python que permiten definir la estructura de la base de datos, y permiten realizar operaciones con ella, ya sea añadiendo, borrando o modificando, o simplemente realizando consultas.

- ***Plantillas*** (*Template*): Una plantilla es un fichero de texto que permite definir la estructura de la página HTML. Con él se da forma a la respuesta HTML que se quiera proporcionar.

Este tipo de estructura está definida como MVT "*Model View Template*". Se trata de una arquitectura muy similar a la MVC.

A continuación, se muestran ejemplos del funcionamiento de los diferentes ficheros.

URLS

```
urlpatterns = [
    url(r'^$', views.index),
    url(r'^([0-9]+)/$', views.best),
]
```

Figura 3. Ejemplo de fichero **`url`**.

`Urlpatterns` hace referencia a un listado de funciones `url()`. Esta función tiene como finalidad comparar la dirección a la que se ha conectado el usuario y, dependiendo de cuál sea, ejecutar una vista u otra. Una vez entrada a esta, se llamará a la función indicada en segundo lugar.

Views

```
## fichero: views.py (funciones de visualizacion de Django)
from django.http import HttpResponse

def index(request):
    # Obtener un HttpRequest - el parametro peticion
    # Realizar operaciones usando la infomracion de la peticion.
    # Devolver una HttpResponse
    return HttpResponse('!Hola desde Django!')
```

Figura 4. Ejemplo de fichero **`view`**.

En la figura 4 se puede apreciar cómo se define una vista. Se puede observar cómo se recibe por parametro una `request`, que es un objeto `HttpRequest`. Y la funcion devuelve un objeto `HttpResponse`, en este caso la respuesta sólo es un texto, pero de forma general se suele retornar con un método `render` y como parámetro un

`template`, para así renderizar la plantilla mostrando los datos que se precisen. A continuación un ejemplo:

```
## filename: views.py

from django.shortcuts import render
from .models import Team

def index(request):
    list_teams = Team.objects.filter(team_level__exact="U09")
    context = {'youngest_teams': list_teams}
    return render(request, '/best/index.html', context)
```

Figura 5. Ejemplo de fichero **`view`** con respuesta **`render`**.

Aquí se puede apreciar cómo se pasa un contexto al `template` “index.html”.

Template HTML

```
## filename: best/templates/best/index.html

<!DOCTYPE html>
<html lang="en">
<body>

 {% if youngest_teams %}
    <ul>
    {% for team in youngest_teams %}
        <li>{{ team.team_name }}</li>
    {% endfor %}
    </ul>
{% else %}
    <p>No teams are available.</p>
{% endif %}

</body>
</html>
```

Figura 6. Ejemplo de **`template`**.

En la figura 6 se puede apreciar cómo se podrían mostrar los datos que hemos pasado como parámetro en la anterior `view`. Se puede ver cómo se puede hacer uso de

condiciones y bucles, para ello se hace uso de `{% %}`. También para poder mostrar los datos que se han pasado, se ha de hacer uso de `{{ }}`.

Otras funcionalidades

Admin Site

Django tiene incluido por defecto una página para la administración de la plataforma, siempre y cuando se utilice el esquema básico que otorga este.

Formularios

Los formularios sirven para obtener datos de los usuarios. Mediante Django, se pueden crear de forma sencilla, además de tener un procesamiento sencillo para poder incluir los datos donde se precise.

Autentificación y permisos de los usuarios

Django tiene incluido en su estructura base un sistema de permisos y autentificación, con un sistema de seguridad robusto, que permite gestionar los usuarios de una manera sencilla.

Ventajas

A continuación, se van a listar las diferentes ventajas que tiene este framework y porqué se elige este frente a cualquier otro, enumerando sólo las principales [18]:

- Rapidez: Está diseñado para facilitar el desarrollo, por lo que, a la hora de crear una aplicación, facilita la rapidez de los programadores a la hora del desarrollo.
- Django incluye una gran cantidad de extras, que ayudan a la hora del desarrollo. Ayuda con la autentificación, administración del contenido y mucho más, aunque estos aspectos no se han utilizado en esta plataforma.
- Seguridad: una gran ventaja de este *framework* es la seguridad. Django otorga gran variedad de sistemas de seguridad que ya vienen integrados para facilitar el desarrollo, como pueden ser inyecciones SQL, falsificación de consultas, etc.

- Versatilidad: puede usarse como base para cualquier tipo de plataforma, ofreciendo así una gran diversidad. Se pueden crear desde plataformas científicas, hasta plataformas para grandes organizaciones.

2.3 – Sistemas de imagen en la plataforma

La plataforma pretende ofrecer un servicio consistente en obtener medidas remotas empleando diferentes instrumentos y sistemas de imagen ubicados en el laboratorio de investigación del Grupo de Ingeniería Fotónica. Actualmente se cuenta con tres sistemas completos. A continuación, se muestran sus características y una fotografía de cada equipo.

Sistema HSI, hiperespectral en rango VisNIR

Este equipo de imagen hiperespectral genera un conjunto de datos tridimensional correspondiente a las dos dimensiones espaciales de un determinado objeto, tejido o material, siendo la tercera dimensión la componentes espectral en el rango visible e infrarrojo cercano [19]. Permite obtener una caracterización espectral precisa de todos los puntos sobre un tejido y así identificar su composición química.

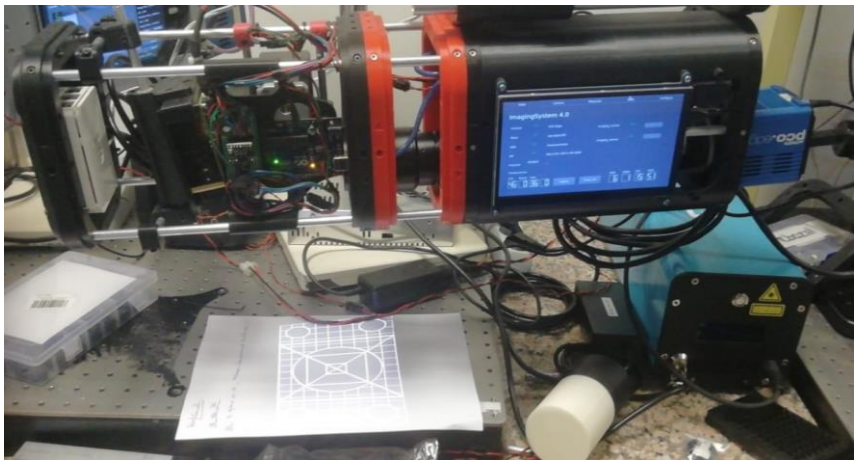


Figura 7. Sistema VisNIR.

Sistema	HSI-VISNIR + Proyector para SFDI
Resolución	Hasta 2048x2048x4012
Rango espectral	400nm-1000nm
Cámara	PCO Edge

Sistema HSI, hiperespectral en rango SWIR

Equipo equivalente al anterior en rango SWIR (*Short Wave Infrared*).

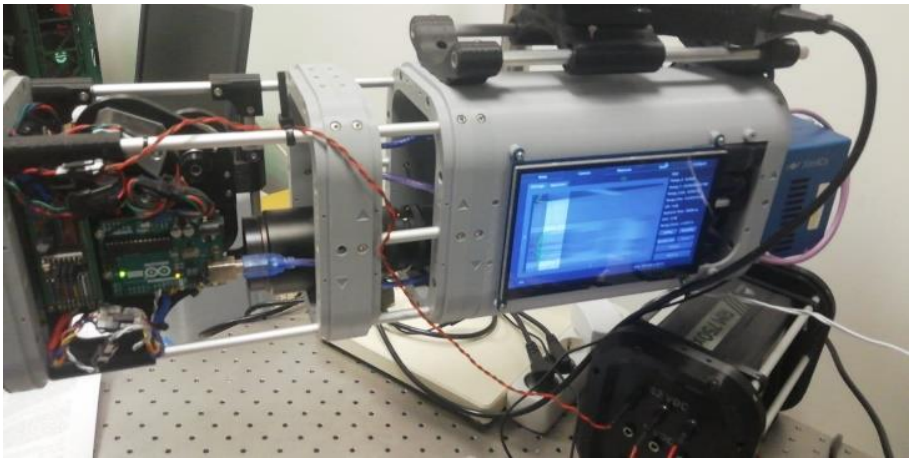


Figura 8. Sistema SWIR.

Sistema	HSI-SWIR
Resolución	Hasta 320x200x240
Rango espectral	1000nm-1600nm
Cámara	Xenics

Sistema OCT

Este equipo de imagen de tomografía de coherencia óptica genera también conjuntos de datos tridimensionales. En este caso, dos son las dimensiones espaciales, y la tercera es la dimensión en profundidad sub-superficial del tejido o material. Permite obtener una caracterización morfológica precisa de la organización del material y7o tejido con una resolución de 5 micrómetros en profundidad y 10 micrómetros en superficie.

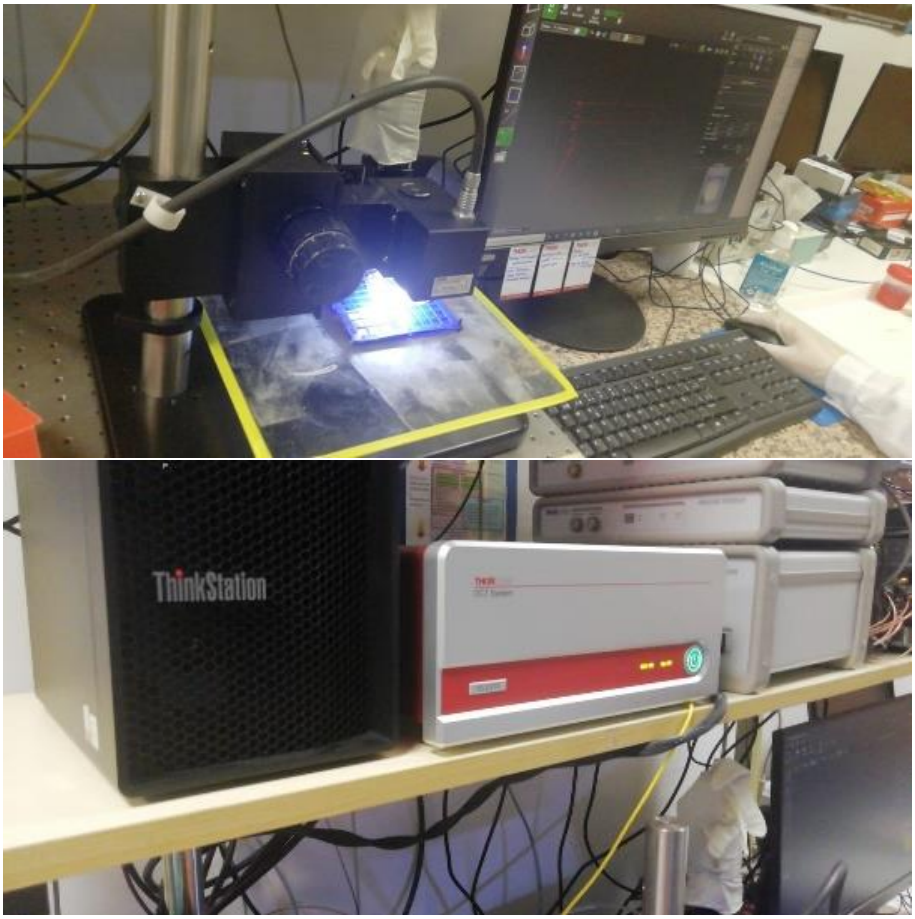


Figura 9. Sistema OCT.

Sistema	OCT-PSOCT 3D
Resolución	Hasta 1024x1024x1024
Volumen	10mmx10mmx3.5mm

2.4 – Comunicaciones

En este apartado se describirá las comunicaciones establecidas en la plataforma, cómo se generan, así como su tipología y funcionamiento. Como aspecto general, se puede definir una comunicación online como el intercambio de mensajes a través de las diferentes plataformas que ofrece Internet.

2.4.1 – RESTful

Se trata de un sistema que sirve para conectar varios sistemas, utilizando el protocolo HTTP que permite realizar peticiones o generar datos. Una de las características principales de RESTful, es el uso de métodos HTTP, con un sistema de conexión sin

estado, lo que significa que se deben enviar todos los datos necesarios para poder permitir que el servidor realice la operación, ya que, al no tener estado, no se guardan los datos de una interacción a otra.

Uno de ellos es el método `GET`, que sirve para producir datos. Mediante este método, el cliente realiza una petición al servidor y este le responderá con una serie de datos. Por otra parte, el método `POST` sirve para crear recursos en el servidor y de esta manera subir elementos a la plataforma. Por ejemplo, los formularios usan este tipo de métodos, una vez se han introducido los datos, se pulsa sobre un botón que ejecuta un método `POST`, posteriormente la plataforma obtiene estos datos y los guarda en la BBDD.

Existen otra serie de métodos como pueden ser `PUT`, que sirve para cambiar el estado de un recurso o actualizarlo, o por otra parte `DELETE`, que sirve para eliminar o borrar recursos [20].

2.4.2 - Software de soporte de los equipos de imagen

A los instrumentos a interconectar en la plataforma se les ha dotado de un software de distribución de Linux customizada desarrollada previamente por personal del Grupo de Ingeniería Fotónica. Esta distribución es muy sencilla de utilizar, ya que esta creada con la finalidad de ser fácil y rápida. Este software está compuesto de 5 pantallas diferentes descritas a continuación.

State

Figura 10. Pantalla **`State`** del Instrumento.

Esta imagen muestra el apartado de estado, en ella se pueden observar los diferentes datos de la cámara, como puede ser la dirección de la API (*Application Programming Interface*) o qué cámara se está usando.

Camera

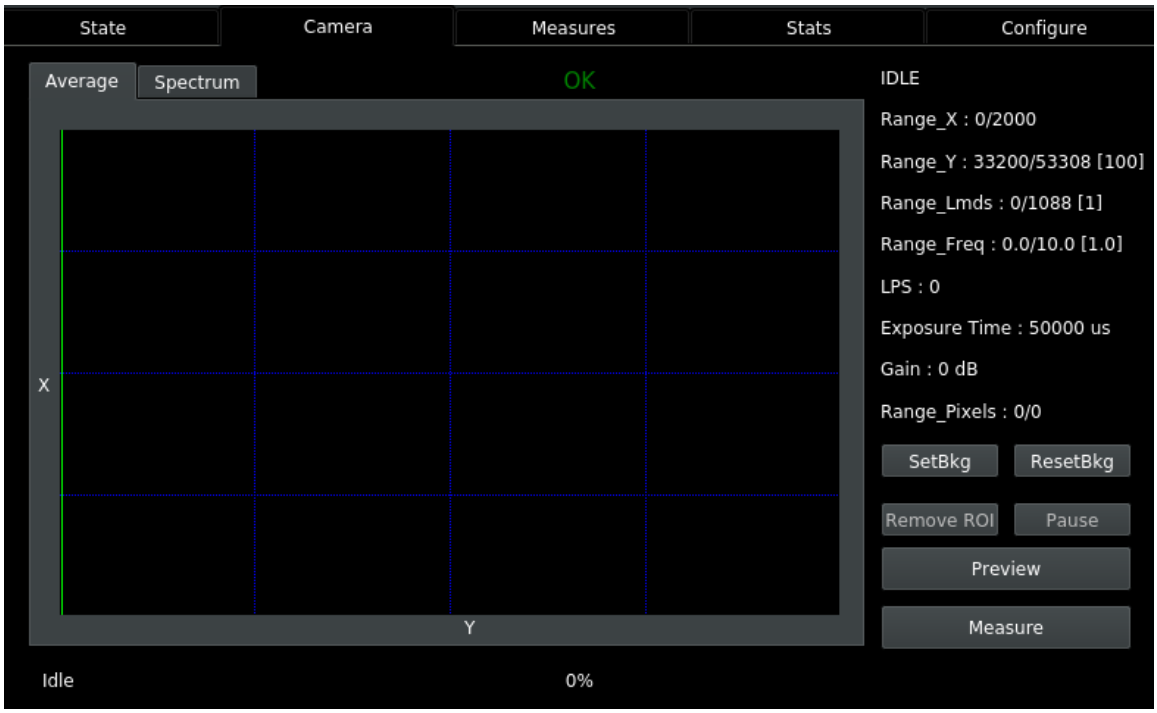


Figura 11. Pantalla **`Camera`** del instrumento.

Esta sección permite realizar una medida y, una vez realizada, la captura realizada se mostrará en la pantalla “Measures” que estará disponible a través de la API.

Measures

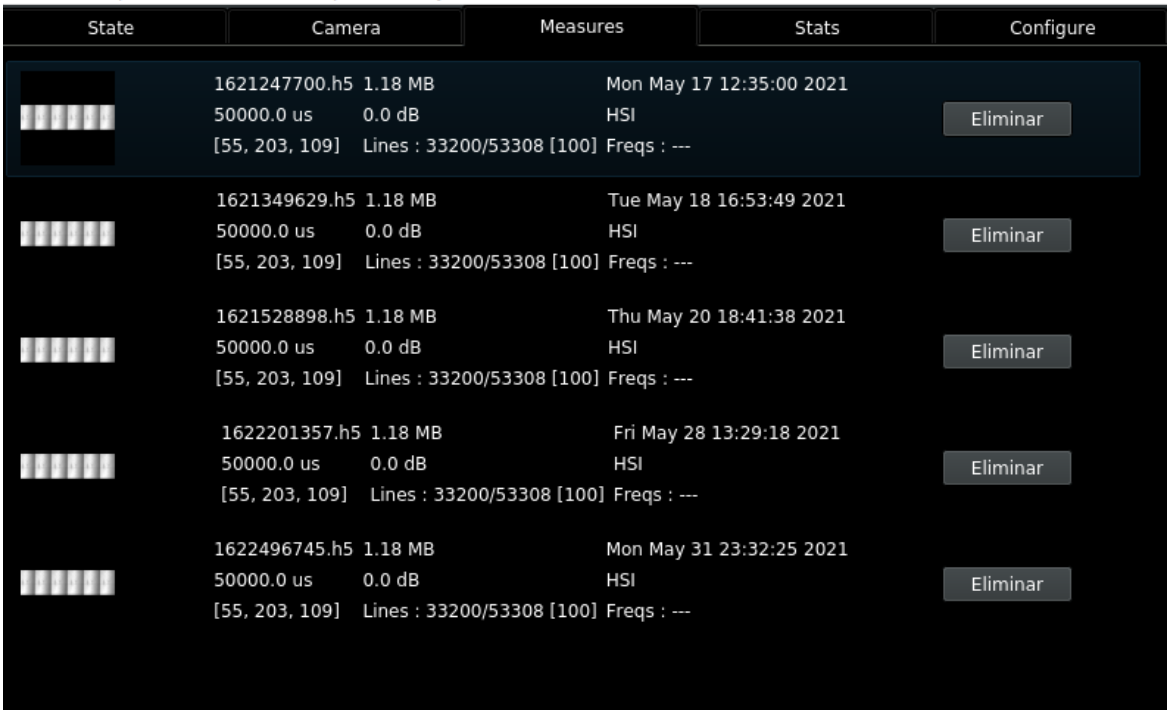


Figura 12. Pantalla **`Measures`** del instrumento.

En esta pantalla se puede ver cómo se muestran las diferentes medidas que se han ido tomando a lo largo del tiempo con los diferentes datos asociados a las mismas. También se ofrece la posibilidad de eliminarlas, y a la izquierda aparece una pequeña previsualización (*preview*) de la medida.

Stats

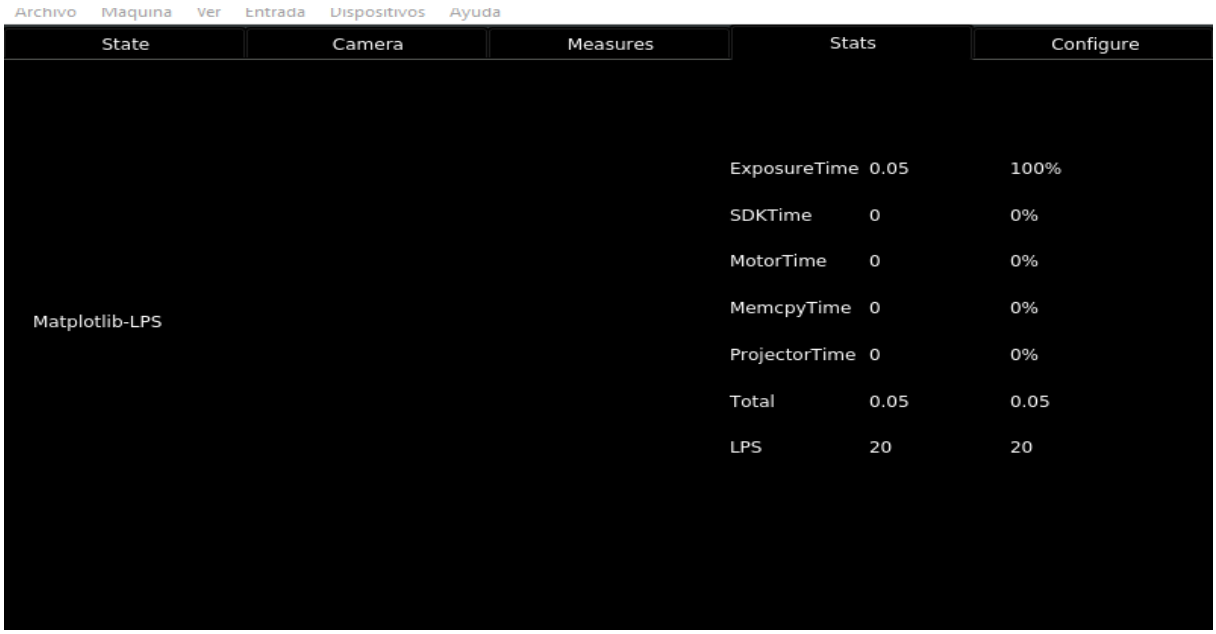


Figura 13. Pantalla **`Stats`** del instrumento.

Aquí se pueden observar diferentes estadísticas relacionadas con el instrumento.

Configure

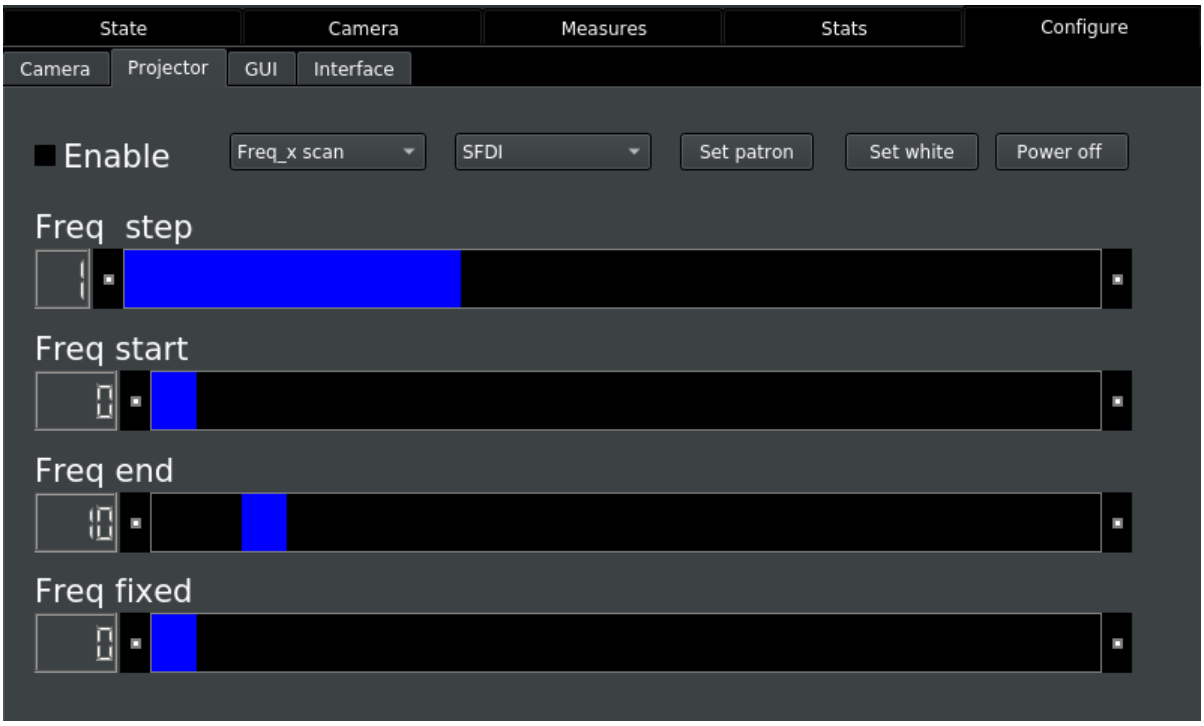


Figura 14. Pantalla **`Configure`** del documento.

Por último, la sección “Configure” permitirá establecer la configuración a la hora de realizar la medida. Se puede observar cómo permite modificar los ajustes de diferentes partes del instrumento.

API

Mediante la API, se pueden observar todas las medidas que se han tomado por ese instrumento. Lo único que se debe hacer es poner en el navegador http://ip:port/measure. De esta manera aparecerá un JSON con las diferentes medidas. Por otro lado, se puede descargar el archivo de la medida poniendo a continuación el nombre de la misma, tal que “`ip:port/measure/nombre`”.

3 - Desarrollo

En este apartado, se mostrará la estructura de la plataforma. Explicando sus diferentes partes y cómo están conectadas entre sí, además de los diferentes modelos y una explicación de la base de datos (BBDD) desarrollada.

3.1 - Estructura de la plataforma

A continuación, se muestra un esquema general de cómo queda la plataforma web al completo:

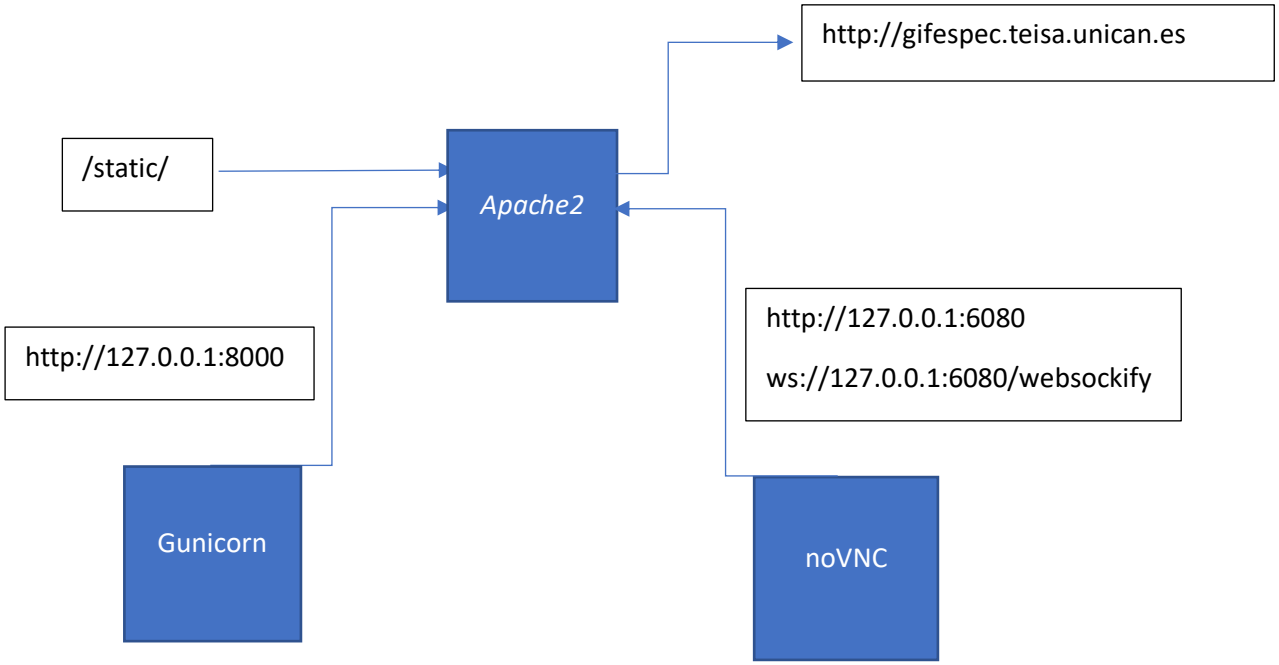


Figura 15. Estructura lógica de la plataforma.

En la figura 15 se puede observar la estructura de la plataforma desde una vista externa. Se puede ver como la plataforma está dividida en tres partes.

Gunicorn

Mediante Gunicorn se despliega la plataforma web creada con Django, que permite usar las diferentes funcionalidades que se han implementado. Como se puede observar en la imagen superior, se despliega en el puerto 8000.

Django proporciona una herramienta para poder desplegar la plataforma, pero esta sólo debe usarse en la etapa de desarrollo debido a que no está orientada a producción. Otra plataforma que podía haberse usado es Apache dado que también tiene un plugin *WSGI* (*Web Server Gateway Interface*) pero da diversos problemas en entornos gráficos. Y puesto que Gunicorn es un servicio enfocado a producción, es lo que ha motivado su uso.

NoVNC

En la plataforma, se despliega el cliente VNC sobre el puerto 6080. De esta manera, a través de este puerto, se es capaz de establecer la conexión con el instrumento deseado.

Apache2

Esta parte es de las más importantes de la plataforma, ya que, gracias a esta herramienta, se puede unificar la plataforma web y el cliente VNC. Para poder realizar esta parte, se ha tenido que implementar un servidor proxy que permita incluir ambas partes en un mismo despliegue. Además, se debe incluir un apartado con la ruta de donde están situados los archivos estáticos para que la ejecución de la web sea la óptima. Para esto se deben activar tres módulos esenciales para apache2:

- `Proxy`
- `Proxy_http`
- `Proxy_wstunnel`

Una vez activados estos módulos, se debe editar el archivo del servidor virtual. Ejemplo del aspecto del archivo:

```
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com
```

```
    ProxyPass /vnc/  http://127.0.0.1:6080/
    ProxyPassReverse /vnc/  http://127.0.0.1:6080/

    ProxyPass /websockify  ws://127.0.0.1:6080/websockify
    ProxyPassReverse /websockify  ws://127.0.0.1:6080/websockify

    Alias /static /home/jorge/TFG_UNICAN/WEB_MEDIDAS/static
    ProxyPassMatch ^/static !

    ProxyPass /  http://127.0.0.1:8000/
    ProxyPassReverse /  http://127.0.0.1:8000/

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

- `ProxyPass`: Hace un mapeo de las peticiones enviadas al primer parámetro, hacia el segundo.
- `ProxyPass Reverse`: Hace la misma función que el `ProxyPass` pero en sentido inverso.

Por otra parte, también se debe añadir el directorio sobre el que se va a trabajar, para dar de esta manera acceso a los archivos estáticos y que la web se muestre correctamente.

```
<Directory /home/jorge/TFG_UNICAN/WEB_MEDIDAS/static>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
```

```
</Directory>
```

Aquí se puede ver cómo está añadido el directorio estático del proyecto.

3.2 - Estructura interna de la aplicación

Dentro de la plataforma desarrollada en el entorno Django, ésta se divide en diferentes partes llamadas aplicaciones o apps. Esto se hace para realizar una pequeña descomposición en módulos de las diferentes partes de la aplicación, para que puedan ser reutilizadas en otros proyectos, o simplemente para llevar una estructura en el desarrollo que facilite realizar todo tipo de cambios, sin afectar a otras partes.

Por ejemplo, en este proyecto se ha dividido la plataforma en diferentes apps:

- ***Users***: Esta parte gestiona todo lo relacionado a gestión de usuarios y grupos, como pueden ser los registros y los inicios de sesión.
- ***Instruments***: Esta sección gestiona todo lo relacionado con los instrumentos.
- ***Measurements***: API destinada a la muestra de los diferentes paths guardados en la plataforma.
- ***GestorMedidas***: API destinada a la gestión de los paquetes de medidas, tanto descargas, como visualizaciones y procesamiento de las imágenes.
- ***FileTree***: API desarrollada por terceros, que nos permite crear un árbol de directorios atendiendo a una ruta dada.
- ***Api***: permite saber en todo momento el estado de los dispositivos asociados a la plataforma que se han guardado en la BBDD.

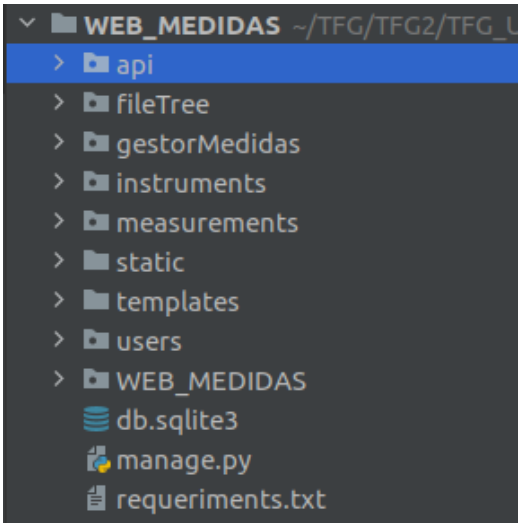


Figura 16. Composición de la plataforma.

Posteriormente existen 3 archivos que no pertenecen a ninguna de las apps:

- `Db.sqlite3`: se trata de la BBDD que se utiliza en la plataforma.
- `Manage.py`: es el archivo Python que permite realizar acciones sobre el proyecto, como puede ser realizar migraciones a la base de datos o iniciar el servidor para realizar pruebas. Los métodos más usados son:
    - `Runserver`: sirve para iniciar el servidor e ir comprobando el funcionamiento de la plataforma.

`Makemigrations`: detecta los cambios que se han hecho en la estructura de la BBDD gracias a los `models` que son clases Python que nos permiten definir los diferentes objetos de la BBDD, y crea una migración.

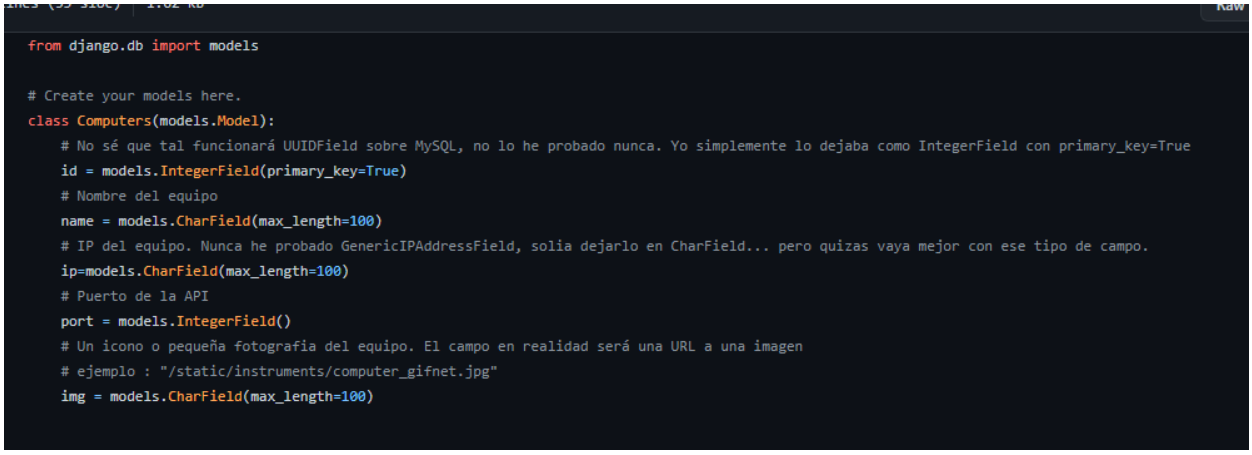
- `Migrate`: aplica la última migración a la base de datos.

- `Requeriments.txt`: sirve para instalar las librerías necesarias en el entorno virtual para poder ejecutar la plataforma.

Después, en el esquema también encontramos una parte llamada WEB_MEDIDAS, que gestiona todos los ajustes relacionados con la plataforma, además de las diferentes `urls`.

Modelos

Un modelo es un tipo de objeto que se guarda en la BD asociada a la plataforma creada. Estos objetos suelen ser creados en el archivo "`models.py`" asociado a cada API desarrollada. Posteriormente, se hace uso de uno de los comandos incluidos en el archivo "`manage.py`" y se hace que la base de datos se adapte a los modelos que se han definido. A continuación, se muestra un ejemplo de cómo se ha creado un modelo, y posteriormente se mostrará la estructura final de la BD.



```python
from django.db import models

# Create your models here.
class Computers(models.Model):
    # No sé que tal funcionará UUIDField sobre MySQL, no lo he probado nunca. Yo simplemente lo dejaba como IntegerField con primary_key=True
    id = models.IntegerField(primary_key=True)
    # Nombre del equipo
    name = models.CharField(max_length=100)
    # IP del equipo. Nunca he probado GenericIPAddressField, solia dejarlo en CharField... pero quizas vaya mejor con ese tipo de campo.
    ip=models.CharField(max_length=100)
    # Puerto de la API
    port = models.IntegerField()
    # Un icono o pequeña fotografia del equipo. El campo en realidad será una URL a una imagen
    # ejemplo : "/static/instruments/computer_gifnet.jpg"
    img = models.CharField(max_length=100)
```

Figura 17. Ejemplo de modelo.

Este es un ejemplo de cómo se crea el objeto `computers`, que corresponde a una computadora del sistema. Podemos observar cómo existen diferentes campos como puede ser el `id` o el `name`, con cada tipo de dato que va contenido. Esta tabla será representada posteriormente en la BD al hacer la migración.

Base de datos

La BD se crea automáticamente en el momento en el que se usa el comando `migrate` del archivo "`manage.py`". Este método toma los diferentes modelos creados en la plataforma y los transporta a la BD con todos los atributos que este establecidos.

Cada vez que se realicen cambios en estos modelos, se deberá crear una nueva migración en la plataforma y ejecutarla, para que de esta manera se efectúen todos los cambios necesarios en la BD. Los modelos creados actualmente en la plataforma son:

- `Instruments`: sirve para almacenar toda la información correspondiente a un instrumento, como puede ser `ip`, `port`, `name`…

- `Computers`: contiene los datos necesarios para almacenar un ordenador que se encuentre en la red.
- `MeasurementsRoute`: guarda los datos necesarios para obtener las rutas guardadas en el sistema, tiene atributos como pueden ser `path`, nivel de acceso, nombre…

Además de estos, se emplean los modelos ya creados por Django como son, `User` y `Group`.

3.3 - Gestión de usuarios y grupos

En este apartado se realizará una pequeña descripción de cómo se ha implementado el apartado de usuarios y grupos. Esta plataforma está basada en un sistema de permisos basado en grupos. Además, siempre que se quiera acceder a la web, se debe haber iniciado sesión, en caso contrario no se podrá acceder y se deberá acceder al apartado de registro.

Para crear este tipo de gestión, se ha hecho uso de los formularios proporcionados por Django. Estos facilitan la oportunidad de iniciar sesión o registrarse de una forma muy sencilla, guardando los datos o consultándolos de la base de datos de una manera fácil y eficaz

Permisos

Como se ha mencionado anteriormente, los permisos dependen del grupo al que pertenece el usuario. Cuando se crea uno, se debe indicar a que grupo pertenece, esto hará que la plataforma muestre algunos datos o no. Por ejemplo, en el apartado donde se navega por los diferentes paths, dependiendo del grupo al que pertenece el usuario, se muestran unos u otros.

Usuarios

Para el desarrollo de este apartado, se ha hecho uso del modelo que ya viene predefinido por Django para los usuarios. Se ha elegido usar este, porque ya tiene los atributos necesarios creados, como pueden ser: `username`, `first_name`, `password` y `groups`.

Además, contiene una serie de métodos, que permiten al desarrollador comprobar si el usuario existe. Esto es muy útil a la hora de iniciar sesión, ya que con un simple método se puede crear el inicio de sesión.

Grupos

Como en el anterior caso, también se ha hecho uso del sistema de grupos creado por Django. Ya que el único uso que se le va a dar es que, dependiendo del grupo al que este unido el usuario, la plataforma mostrará unos datos o no.

3.4 - Gestión de instrumentos

En este apartado, se mostrará cómo se ha realizado la gestión de los diferentes instrumentos del laboratorio. Se centrará en describir cómo se realiza la conexión VNC entre la plataforma y el dispositivo. Y cómo se gestiona la parte relacionada a las medidas tomadas por este.

3.4.1 - VNC

El servicio VNC es primordial para poder realizar una gestión del instrumento a distancia. Para ello son esenciales dos partes, una de ellas es el servidor VNC, que se ejecuta en el instrumento gracias al S.O, en el puerto 5900. Por otro lado, se necesita la parte del cliente, que se ejecutará en la plataforma a través de una aplicación llamada noVNC. Existen una gran variedad, pero hemos elegido esta última, porque es la más sencilla de todas.

En sistemas como Ubuntu, que es en el que se ha desarrollado esta plataforma es muy fácil de instalar. Basta con ejecutar el comando: `$ sudo apt install novnc`. Posteriormente, lo único que se debe hacer es ejecutarlo en algún puerto para que el servicio se ponga en modo escucha, esto se realizará mediante el comando: `$websockify --web=/usr/share/novnc/  6080 192.168.1.251:5900.`

Esto hará que se ponga en escucha en el puerto 6080.

```
jorge@jorge-VirtualBox:~$ sudo websockify --web=/usr/share/novnc/ 6080 192.168.1
.251:5900
[sudo] contraseña para jorge:
WebSocket server settings:
  - Listen on :6080
  - Web server. Web root: /usr/share/novnc
  - No SSL/TLS support (no cert file)
  - proxying from :6080 to 192.168.1.251:5900
```

Figura 18. **noVNC** arrancado en el sistema.

Una vez realizado, se está en condiciones para vincular el cliente al servidor del instrumento. Para ello, se debe pasar tanto la `ip` como el puerto por parámetros en la `url`, para que así sólo haga falta pulsar el botón conectar e introducir la contraseña.

Figura 19. Ejemplo de conexión **noVNC**.

En esta imagen se puede apreciar el paso de parámetros a través de la `url`, para que el cliente lo identifique y realice la conexión a la `ip` seleccionada.

Hay que decir que la ejecución de este cliente está detrás de un proxy creado por Apache2 como ya se ha comentado anteriormente.

Para explicar brevemente cómo se accedería desde la plataforma, es tan sencillo como en la pantalla home donde aparecen los instrumentos, hacer click sobre el botón opciones que aparece en el instrumento, y pulsar sobre conexión VNC.

3.5 - Gestión de medidas

Este apartado se centrará en la parte de gestión de medias. Esta plataforma basa esta parte en archivos llamados paquetes de medidas, que los diferentes archivos de extensión `h5` de las medidas, y un archivo de información que contiene la descripción de cada uno de los archivos `h5`. A continuación se explicará en detalle qué es un paquete de medidas.

3.5.1 - Archivo H5

Se trata de un archivo de datos guardado en el formato de datos jerárquicos (HDF, *Hierarchical Data Format*). Dentro de este archivo se contienen datos de gran tamaño agrupados de forma jerárquica. En este caso, contiene una matriz que compone una imagen para generar. Este tipo de archivos van a ser manejados por una librería de Python llamada `h5py` que permitirá acceder a toda la estructura interna de los archivos, y extraerla cuando se considere necesario [21].

3.5.2 - Paquete de medidas

Se trata de un directorio, que contiene las medidas asociadas a una prueba. En este archivo se pueden encontrar tanto las medidas tomadas en “raw”, como las diferentes medidas procesadas, y todas ellas divididas entre los diferentes instrumentos desde las que se toman. Además de esto, contienen un archivo JSON que describe cada una de las medidas.

En la figura 20 podemos apreciar cómo se divide la estructura de los paquetes de medidas.

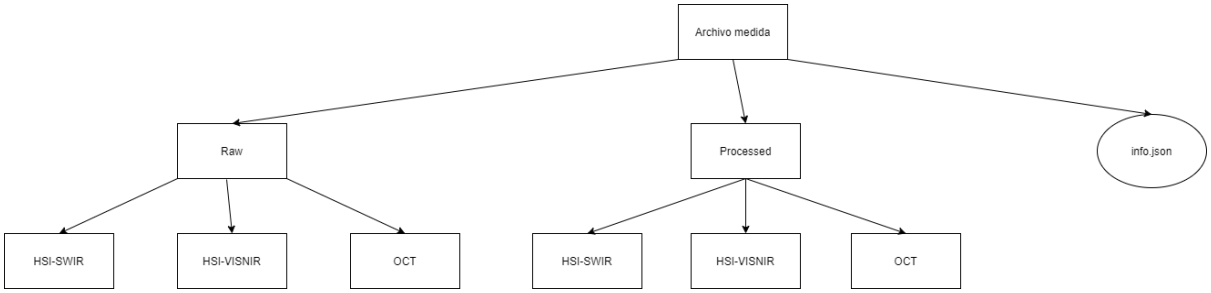


Figura 20. Esquema de archivos de un paquete de medidas.

3.5.3 - Visualización de la medida

Este apartado se desarrolla sobre la parte measurements de la plataforma. Una vez en ella, se tiene una serie de paths para elegir y poder navegar por ellos, gracias a la herramienta FileTree. Cuando se localice un paquete de medidas, tan sólo hay que pulsar sobre él, y una función javascript realizara una llamada a la API destinada a la muestra de medidas. Para que esta api desarrolle la medida correctamente, se tendrá que pasar el path de la medida como parámetro.

Una vez la API obtiene el path, accede al "`info.json`" que contiene la medida, y empieza a realizar una decodificacion medida a medida.

A la hora de realizar esta API, se ha optado por establecer diferentes decodificadores, dependiendo del instrumento y el tipo de medida ("raw","processed"). Porque de esta manera, si se tiene la necesidad de cambiar los datos que se van a mostrar en un instrumento en concreto, se hará de manera mas sencilla.

Por ahora, se ha optado por mostrar los mismos datos en todos. La estructura que se sigue por medida es, en la parte izquierda una pequeña previsualizacion, y en la parte derecha los diferentes datos estructurados.

Para gestionar el JSON se ha hecho uso de la librería `json` que proporciona python, en concreto los metodos:

- `Dump`: convierte objetos Python en objetos JSON.
- `Load`: transforma un archivo JSON en un diccionario de Python.

3.5.4 - Descarga de medidas

Para esta parte se hará uso de la API de cada instrumento. La medida se descargará en uno de los paquetes de medidas que ya dispone el sistema. A la hora de la descarga, se debe entrar en el panel de descarga del instrumento y seleccionar la que se desea. Posteriormente se elegirá el path de descarga y se pulsará sobre el botón indicado. Finalmente se descargará la medida y se añadirán los datos en JSON de información asignado.

```
654
655  def descargaEnPath(request):
656      ip = request.GET.get('ip', '')
657      port = request.GET.get('port', '')
658      name = request.GET.get('name', '')
659      path = request.GET.get('path', '')
660      tipo = request.GET.get('tipo', '')
661      url = ip+':'+port+'/measure/'+name
662      base = request.GET.get('base', '')
663      myfile = requests.get('http://'+url, allow_redirects=True)
664      open(path+'/'+name, 'wb').write(myfile.content)
665      response = urllib.request.urlopen('http://' + ip + ':' + port + '/measure')
666      JSON_object = json.loads(response.read())
667      incluye = ''
668      for r in JSON_object:
669          if r == name:
670              incluye = JSON_object[r]
671      dire = base + 'info.json'
672      with open(base + 'info.json', 'r') as file:
673          data = json.load(file)
674          data['measurements'][tipo][name] = incluye
675          data['measurements'][tipo][name]['path'] = path+'/'+name
676          json.dumps(data)
677          with open(dire, 'w') as out:
678              json.dump(data, out, indent=4)
679
680      return HttpResponse('Descarga realizada')
681
```

Figura 21. Código de descarga.

En la figura 21 se puede apreciar cómo funciona el proceso de descarga de la medida en la plataforma. Se puede observar cómo se pasan por parámetros diferentes datos para realizar de forma correcta la descarga. Posteriormente se accede a la url creada mediante los parámetros recibidos y se descargará la medida en el paquete indicada. Más tarde, se deberá acceder al JSON asociado al instrumento y realizar un filtrado para rescatar sólo la medida indicada. Una vez filtrado, se añade el conjunto de datos a la parte correcta del archivo de información del paquete.

Finalmente, la medida descargada ya estará preparada para visualizarla correctamente.

3.5.5 - Generación de previsualizaciones

Una de las partes principales de la plataforma, es generar una previsualización del archivo. Para la realización de esta parte se ha empleado la librería de Python llamada `h5py` e `Image`. La primera de ella sirve para acceder a los archivos `h5` donde se almacenan las medidas, y la segunda, dada la matriz de la imagen que se obtiene del archivo, genera una previsualización.

Para la realización de esta parte, se hace uso de una `api` ya creada, donde se recibe como parámetro en la `url` el `path`, y posteriormente se devuelve la imagen creada.

```
472  ##Metodo que recibe un path de un archivo h5 y un path de guardado, y genera una imagen
473  def muestraImagen (request):
474      path = request.GET.get('id', '')
475      response = HttpResponse(content_type="image/png")
476      with h5py.File(path,"r") as fp:
477          data = np.array(fp['data'])
478          lamdas = np.array(fp['lambdas'])
479          preview = data.mean(axis=2)
480          im = Image.fromarray(preview)
481          im = im.convert('RGB')
482          im.save(response, 'PNG')
483
484          return response
485
```

Figura 22. Código mostrar imagen.

Como se puede observar en la figura 22, éste es el proceso para generar una imagen partiendo de un archivo `h5`. Dado el `path`, se abre el archivo con "`h5py.File`". Una vez abierto, nos quedamos con la matriz que forma la imagen que se encuentra bajo la key "`data`". Una vez formado el array con los datos obtenidos, gracias a la librería "`Image`" se genera la imagen necesaria, se guarda en la respuesta y se retorna.

4 - Evaluación y pruebas

En este capítulo se abordará todo lo referente a la funcionalidad de la plataforma, cómo pueden ser tiempos de descarga o de carga, los mecanismos de seguridad aplicados, la posibilidad de añadir nuevos instrumentos o de instalar contenidos.

4.1 – Rendimiento

En esta parte se va a analizar el rendimiento de la página, esto significa, analizar los diferentes tiempos de respuesta. Para ello se va a utilizar la consola que viene integrada en el navegador de Firefox, ya que proporciona gran información sobre los tiempos de carga. En la memoria se incluye el análisis de 3 partes, el tiempo de carga de la página principal, el tiempo en realizar la *preview* de una medida y el tiempo de descarga de una medida. Para poner en contexto el rendimiento, se ha realizado el análisis desde casa con una conexión de fibra simétrica de 300MB y con un ordenador con las siguientes características: 8GB de RAM, Intel Core i7-10700K 3,8GHz, SSD 140GB, HDD 1TB.

4.1.1 - Carga de pagina

En el apartado de carga de la página, se ha obtenido un tiempo de respuesta de 1014 ms. Este tiempo según Google, se encuentra por encima del rango deseado, pero tampoco hace molesto el navegar por la web. En la imagen inferior se puede apreciar el tiempo.

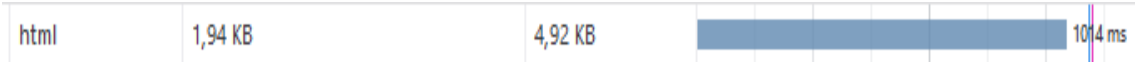


Figura 23. Tiempo de carga de página.

4.1.2 – Preview

Este es uno de los sistemas más importantes en la plataforma, ya que gracias a él se puede ver en detalle la imagen de la medida tomada. Debido a esto se desea tener un sistema de previsualización que funcione rápido. Y se ha conseguido, ya que el tiempo de respuesta en la llamada y el retorno de la imagen, conlleva 74 ms, lo que es un tiempo más que aceptable.

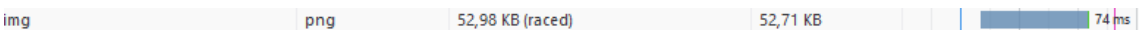


Figura 24. Tiempo en mostrar preview.

4.1.3 – Descarga

En el apartado de descarga, se ha obtenido un tiempo de 210 ms, esta prueba está sujeta a cambios, ya que el archivo de prueba que hemos utilizado es de un tamaño bastante menor, unos 260 MB, que con los que se trabajará, que pueden rondar los 20GB. Pero en este caso el tiempo es aceptable y no dificulta el funcionamiento.

Figura 25. [Tiempo de descarga].

4.2 – Seguridad

La mayoría de los sistemas se encuentran dentro de Django, por lo que no ha sido necesario implementarlos. Por nuestra parte, solo se hemos implementado un sistema de autentificación, cuyos modelos han sido extraídos del *framework*. Mediante este sistema, se ha comprobado que para poder acceder a esta plataforma se tenga que estar registrado. Por otra parte, la plataforma estará desplegada en la intranet de la Universidad o en su VPN, por lo que gracias a esto su acceso estará más limitado.

4.2.1 - Seguridad implementada por Django

Django provee de diferentes tipos de seguridad frente a diferentes vulnerabilidades, lo que permite desarrollar la plataforma, sin estar en todo momento pendiente de posibles brechas de seguridad [22].

Cross site scripting protection

Un ataque XSS o *cross-site scripting* se trata de una vulnerabilidad que afecta principalmente a las aplicaciones Web, este tipo de ataque permite a una persona externa inyector código Javascript en la Web, haciendo que la plataforma funcione de una forma no deseada.

Por esto Django implementa un sistema mediante el cual se filtra el texto introducido y se omiten caracteres específicos que puedan aprovechar esta vulnerabilidad.

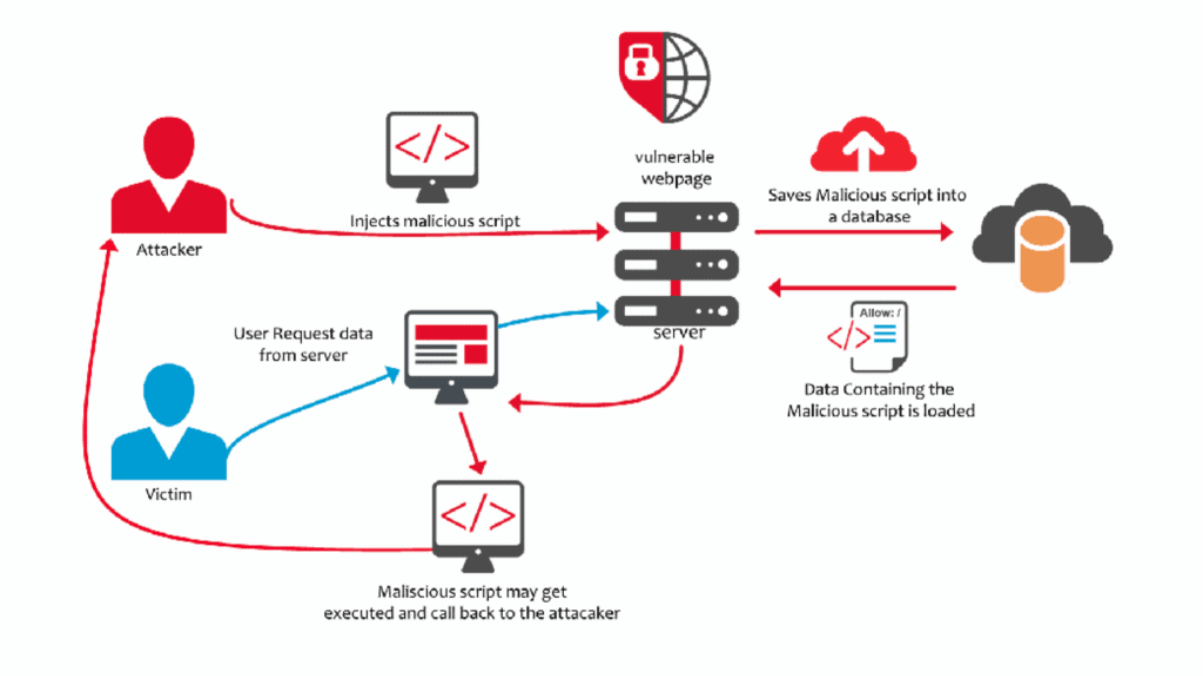


Figura 26. Esquema de ataque.

Cross site request forgery protection

Un ataque CSRF significa una falsificación de petición en sitios cruzados. Este funciona de manera que, si un usuario esta autentificado en una plataforma, se realiza desde el navegador infectado de este una petición a la web en su nombre, de manera que la plataforma piense que ha sido el usuario, pero en realidad no es así.

La protección que emplea Django frente a esto es, analizar las peticiones POST en busca de un “secreto”. Este secreto sólo es conocido solo por el usuario en específico (a través de una cookie), por lo que el usuario malicioso no lo debería saber.

SQL injection protection

La inyección SQL es uno de los ataques más utilizados. Esto se produce cuando un usuario es capaz de alterar la Base de Datos, se puede hacer introduciendo código malicioso a través de formularios o mecanismos similares.

Django se protege de este tipo de ataques desde que empezó a usar “`query parameterization`”. Lo que significa que la “`query`” de código SQL, se encuentra separada de los parámetros “`query`”.

Clickjacking protection

Se trata de un ataque que se basa en engañar al usuario para que pulse sobre un botón que piensa que es de una página, pero en realidad es de otra. Esto significa que el atacante está capturando las pulsaciones que estás haciendo sobre la otra página para llevárselos a la suya. También pueden capturar pulsaciones de teclado.

Django implementa un sistema mediante el cual se verifique que las "`request`" salen del mismo sitio al que se accede.

SSL/HTTPS

El "`Secure Socket Layer`" permite realizar una conexión segura entre los usuarios y el sitio web. Funciona de tal manera que los datos que se transfieren de un punto a otro son imposibles de leer, excepto por los dos extremos. Para realizar esto, se emplean una serie de algoritmos de cifrado, para garantizar el cifrado de los datos. De esta manera cualquier hacker que capture el paquete, no tendrá posibilidad de leerlo.

Figura 27. Esquema SSL.

Otros

Django contiene muchos más mecanismos de seguridad, pero se ha estimado que estos eran los más importantes. Entre otros podemos encontrar, "`Host Header Validation`", "`Referrer policy`" o "`Session security`".

4.2.2 – VPN

El sistema se ejecutará bajo una VPN, esto significa que estará desplegado en una red privada. Esto ofrece un alto grado de seguridad, ya que no se podrá acceder de manera externa, por lo que se controlará en todo momento los usuarios que se conectaran a ella y limitarlo.

4.3 – Escalabilidad

La escalabilidad es la capacidad que tiene un sistema para crecer. En este apartado se analizará la capacidad que tiene plataforma desarrollada para crecer, ya sea añadiendo nuevos instrumentos o añadiendo "`plugins`" para analizar las diferentes medidas.

4.3.1 – Nuevos instrumentos

A la hora de incluir nuevos instrumentos el proceso es sencillo, lo único que se debe hacer es añadir los diferentes datos a la Base de Datos y posteriormente la pantalla home los mostrará de manera correcta y se podrá realizar las operaciones sin ningún problema. Ya que la plataforma está desarrollada para la posibilidad de que se añadan múltiples instrumentos, y además de estos también computadores.

4.3.2 - Plugins

La plataforma debe estar abierta a la incorporación de nuevos "`plugins`" que sirvan para realizar acciones sobre las medidas tomadas. Debido a la estructura que sigue la plataforma y al desarrollo que se ha realizado en Django, la plataforma acepta todo tipo de "`plugins`" ya que es tan sencillo como ir añadiendo las diferentes "`apps`", y definiendo las "`urls`" para cada parte.

Por todo esto, se puede decir que la plataforma desarrollada tiene un índice de escalabilidad bastante alto, ya que está abierto a la introducción de nuevos elementos, da igual de que tipo sea.

5 - Conclusiones y trabajos futuros

En este punto se realizará un análisis global de la plataforma, los diferentes conocimientos que se han adquirido durante el desarrollo de esta y los trabajos futuros que se podrán llevar a cabo partiendo de esta base.

5.1 – Conclusiones

Una vez finalizado el Trabajo de Fin de Grado, se analizarán los objetivos propuestos y si han sido completados con éxito o no. Además, se expondrán las diferentes dificultades encontradas en el desarrollo y una breve valoración de lo que ha supuesto el trabajo en mis conocimientos.

Abordando los objetivos, la mayoría han sido resueltos satisfactoriamente, se ha conseguido que la plataforma obtenga las medidas de los diferentes instrumentos, almacenándolas en el directorio definido por el usuario. También se ha logrado que desde la plataforma se consiga conocer el estado del instrumento o PC que se desee, así como realizar una conexión remota mediante VNC. Se ha implementado un sistema de navegación por rutas para poder buscar los diferentes archivos de medidas situados en la plataforma. Por último, se ha desarrollado un sistema de autentificación para limitar el acceso a la plataforma. Haciendo un análisis general y un poco de autocrítica, hay algunos aspectos que han faltado de implementar, como puede ser añadir una serie de `plugins` a la hora de gestionar una medida, para realizar operaciones sobre ella. Por otra parte, la implementación de la lectura del JSON relacionado a la información podría ser más eficiente.

Analizando las diferentes dificultades que han aparecido en el desarrollo del proyecto, se debe destacar la implementación del sistema VNC detrás de un Proxy desarrollado por Apache. El problema apareció principalmente porque en un primer momento se intentó implementar el sistema VNC a la plataforma, lo cual es verdaderamente complicado y da un innumerable número de fallos. Por lo que, posteriormente, se descargó el cliente noVNC que se ejecutaría de forma externa a la plataforma, uniéndolo entre sí mediante Apache.

Otro problema que surgió fue a la hora de hacer el *decode* del JSON relacionado a la información, ya que tenía un gran número de atributos y se encontraban dentro del mismo, aspectos tales como las medidas de diferentes instrumentos y niveles de procesado. Por lo que se optó por realizar diferentes métodos para tomar los datos por partes.

Por otra parte, gracias a este proyecto he adquirido un gran número de conocimientos que no había obtenido en el grado. He profundizado en el lenguaje Python y el uso de librerías, ya que es un lenguaje que en la mención que he desarrollado no se ve. También he aprendido bastante sobre *back-end* y *front-end*, poniéndolos en el contexto práctico del desarrollo del proyecto.

En general creo que este trabajo me ha servido para ver desde un punto de vista diferente cómo se desarrolla una plataforma web sencilla desde un principio, además de afrontar los diferentes problemas que van apareciendo.

5.2 - Trabajos futuros

A la hora de desarrollar esta plataforma, se tenía en cuenta que debía planificarse para poder extenderse en un futuro. Ya que lo que se ha creado es una base sobre la que trabajar con una serie de funcionalidades básicas, pero no se han tratado `plugins`, ni la gestión más detallada de los paquetes de medidas. Por lo que se podrán añadir un gran número de mejoras.

La primera de las mejoras que se podría llevar a cabo es, añadir a la plataforma una serie de `plugins` que realicen operaciones sobre las medidas tomadas. La web ya está adaptada a ello por lo que se añadiría de forma sencilla. También sería útil implementar el código en los botones de gestión de la imagen, como pueden ser invertir el eje o aplicar filtros, ya que en la actualidad sólo se ha desarrollado el que sirve para refrescar.

Por otra parte, una de las mejoras que habría que aplicar sería reescribir el código que se encarga de obtener los datos en el JSON del archivo de medidas, ya que en este momento no es eficiente y no facilitaría elevar el índice de escalabilidad.

Bibliografía

[1] Paloma Rojo Cresto, “10 estadísticas reveladoras sobre el crecimiento del Big Data”, Big Data & Business Intelligence Slider, 6 marzo 2020 [En línea]. Disponible en: https://dataiq.com.ar/blog/10-estadisticas-reveladoras-sobre-el-crecimiento-del-big-data/. [Accedido: 30-06-2021]

[2] https://www.biomax.com

[3] https://www.meplis.com

[4] https://medssocial.com

[5] Ana Pérez “Características y fases del modelo incremental “, 16 agosto 2016 [En línea]. Disponible en: https://www.obsbusiness.school/blog/caracteristicas-y-fases-del-modelo-incremental. [Accedido: 30-06-2021]

[6] ChrisM , “Beginner's Guide to Python”, 4 julio 2020 [En línea]. Disponible en: https://wiki.python.org/moin/BeginnersGuide. [Accedido: 30-06-2021]

[7] “About the Django Software Foundation”, [En línea]. Disponible en: https://www.djangoproject.com/foundation/. [Accedido: 30-06-2021]

[8] MDN contributors, “JavaScript”, 23 junio 2021 [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript. [Accedido: 30-06-2021]

[9] Jak, “jQuery-File-Tree”, 6 agosto 2012 [En línea]. Disponible en: https://github.com/jak/jQuery-File-Tree. [Accedido: 30-06-2021]

[10] “About SQLite”, [En línea]. Disponible en: https://www.sqlite.org/about.html. [Accedido: 30-06-2021]

[11] “Funcionalidades de PyCharm”. [En línea]. Disponible en: https://www.jetbrains.com/es-es/pycharm/features/. [Accedido: 30-06-2021]

[12] “About”, [En línea]. Disponible en: https://sqlitebrowser.org/about/. [Accedido: 30-06-2021]

[13] https://git-scm.com

[14] Juan Antonio Soto, “¿Qué es VNC y para qué sirve?”, 16 agosto 2020 [En línea]. Disponible en: https://www.geeknetic.es/VNC/que-es-y-para-que-sirve. [Accedido: 30-06-2021]

[15] Joel Martin, “noVNC” [En línea]. Disponible en: https://novnc.com/info.html. [Accedido: 30-06-2021]

[16] “Gunicorn - WSGI server” 12 febrero 2021 [En línea]. Disponible en: https://docs.gunicorn.org/en/latest/index.html. [Accedido: 30-06-2021]

[17] “What is the Apache HTTP Server Project?”, [En línea]. Disponible en: https://httpd.apache.org/ABOUT_APACHE.html. [Accedido: 30-06-2021]

[18] Pablo Camino Bueno, “Qué es Django y por qué usarlo”, 3 agosto 2018 [En línea]. Disponible en: https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/. [Accedido: 30-06-2021]

[19] J.A.Gutiérrez-Gutiérrez, A.Pardo, E.Real, J.M.López-Higuera, O.M.Conde, “Custom Scanning Hyperspectral Imaging System for Biomedical Applications: Modeling, Benchmarking and Specifications,” Sensors 19, 1692 (2019).

[20] Alex Rodriguez, “Servicios Web de RESTful: Los aspectos básicos”, 9 febrero 2015, [En línea]. Disponible en: https://developer.ibm.com/es/technologies/web-development/articles/ws-restful/. [Accedido: 30-06-2021]

[21] Andrew Collette, “HDF5 for Python”, 2014, [En línea]. Disponible en: https://docs.h5py.org/en/stable/. [Accedido: 30-06-2021]

[22] “Security in Django”, [En línea]. Disponible en: https://docs.djangoproject.com/en/1.10/topics/security/. [Accedido: 30-06-2021]