UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Monitor en tiempo real de un sistema de fabricación aditiva para OctoPrint

Presentado por David Zotes González
en Universidad de Burgos — 13 de febrero de 2019
Tutor: César Represa Pérez

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado vamos a incluir la planificación temporal que hemos llevado a cabo y la viabilidad tanto económica como legal de nuestro proyecto.

A.2. Planificación temporal

Este aspecto es uno de los que hubiera gustado mejorar, ya que, la planificación que hemos llevado a cabo no ha sido la mejor. Me hubiera gustado hacer la planificación mediante *sprints* pero sabíamos que el desarrollo del proyecto iba a ser más largo que la duración del Trabajo de Fin de Grado y tampoco nos queríamos fijar objetivos que fueran imposibles de cumplir. Además este proyecto comenzó durante unas prácticas curriculares que realicé en la empresa **Abadía Tecnológica** durante los meses de marzo a mayo de 2018; es por ello que no hemos utilizado la metodología basada en *sprints* sino que hemos intentado avanzar con el proyecto lo máximo posible. En el repositorio de GitHub (https://github.com/AbadiaTecnologica/Monitor-en-tiempo-real-de-un-sistema-de-fabricacion-aditiva-para-Octoprint) se puede ver los distintos *commits* que hemos hecho para llevar a cabo el desarrollo del proyecto.

Inicialmente este proyecto sólo se iba a realizar para las prácticas curriculares pero vimos que era una idea interesante y poco explotada, por lo que decidimos proponerlo como Trabajo de Fin de Grado.

Otro aspecto que debemos tener en cuenta es que hemos tenido que realizar la totalidad del proyecto en las instalaciones de la empresa, ya que

es donde se encontraba el hardware que hemos necesitado para el desarrollo de la aplicación. Esto es una gran limitación, ya que teníamos que estar sujetos al horario de la empresa.

A continuación vamos a explicar las fases por las que hemos pasado para el desarrollo de este proyecto.

Fases

FASE 1: Instalación y configuración

Durante la primera quincena del mes de abril de 2018, los primeros pasos que hicimos fue instalar una distribución *Debian* 9 en el ordenador que usaremos como servidor. Una vez tuvimos eso listo, lo que hicimos fue instalar todas las instancias de OctoPrint que íbamos a usar, una por cada máquina; en nuestro caso fueron 7 instancias. Nosotros tenemos un usuario por cada instancia de OctoPrint para llevar un mejor control de las máquinas.

El siguiente paso fue configurar cada OctoPrint con cada máquina, es decir, tuvimos que introducir las características de las máquinas, el tamaño de las camas, etc.

Cuando tuvimos todas las máquinas bien configuradas y funcionando, nos pusimos a desarrollar la aplicación.

FASE 2: Comunicación con la API de OctoPrint

En la segunda quincena del mes de abril de 2018, llevamos a cabo la comunicación con la máquina mediante la API REST de OctoPrint. Durante el periodo de prácticas sólo realizamos peticiones GET ya que era más sencillo y no teníamos experiencia previa haciendo este tipo de operaciones. Es decir, en nuestro monitor no podíamos realizar ninguna operación, tan solo mostrábamos los datos que necesitábamos.

El procedimiento era parecido al que tenemos ahora, pedíamos los datos en una petición GET, los almacenábamos en diccionarios de Python y usando *Flask* y *Bootstrap* y los lanzábamos a una página web. Ahora el proceso el mucho más refinado, ya que, en las primeras versiones teníamos una tarjeta para cada máquina, es decir, no era iterativo. Además la aplicación no era tan estable y si la máquina nos devolvía un error no identificado, la aplicación se caía.

FASE 3: Boceto de la aplicación

Poco a poco fui mejorando la aplicación y para el final del periodo de prácticas, durante la primera quincena del mes de mayo de 2018, teníamos la funcionalidad de desconectar un máquina tal y como se ve en la siguiente imagen.

Figura A1: Vista antigua de la aplicación.

FASE 4: Mejoras en la interfaz de usuario

A comienzos del mes de octubre de 2018 se continuó el desarrollo del proyecto como Trabajo de Fin de Grado.

Lo primero que hicimos fue dar un cambio a la interfaz para ello usamos *Bootstrap* y *MDBootstrap*. En el desarrollo del producto todo el material que hemos utilizado son licencias de uso libre.

Una vez le dimos un diseño más profesional lo siguiente que hicimos fue introducir confirmación en los botones de desconectar las máquinas; ya que si pulsábamos sobre el botón por accidente la máquina de desconectaba y se cancelaba la impresión. Más tarde introducimos un *navbar* donde tenemos todos los nombres de la máquinas con los que cuenta nuestra aplicación y un enlace a cada instancia de OctoPrint desde la que podemos manejar los parámetros más relacionados con la impresión en curso.

FASE 5: Mejoras en la instalación de máquinas nuevas

Hacia mediados de octubre de 2018 añadimos todos los datos de las impresoras a un archivo CSV; de esta manera cuando tenemos que modificar los datos de una impresora lo podemos hacer cómodamente editando el archivo CSV y el propio *back-end* de la aplicación se encargará de leer el CSV y cargar los datos en nuestra aplicación web.

Durante el desarrollo de la aplicación la API de OctoPrint se actualizó y nuestra aplicación no estaba operativa. El problema fue que modificaron el archivo JSON que obtenemos cuando hacemos la petición GET a la API de OctoPrint e intentábamos acceder a un dato que no existía. Para solucionarlo ahora cargamos todos los datos que necesitamos con un campo por defecto y luego actualizamos con los datos reales procedentes del JSON obtenido, de esta manera nos aseguramos que todos los campos críticos tengan un valor (o el valor real o el valor por defecto), de esta manera ganamos en estabilidad.

FASE 6: Adición de nuevas funcionalidades

Durante las primeras semanas de noviembre de 2018 añadimos las funcionalidades de imprimir, pausar y cancelar impresión. Todas las funcionalidades tienen un sistema de confirmación para evitar que pulsemos sobre ellas por error. Además también añadimos un nuevo estado disponible para mostrar el estado de pausa.

Hasta este momento toda la vista de la aplicación estaba en un archivo llamado *index.html* que se recargaba completamente cada cinco segundos para actualizar los datos de las impresoras 3D. Como no tenía mucho sentido recargar la página entera lo que hicimos fue separar los archivos que eran necesarios recargar con los que no lo eran; de esta manera teníamos dos archivos:

- Base: contiene el *navbar* y el *footer* que son estáticos y no es necesario que se actualicen cada cinco segundos.
- Índice: contiene el cuerpo de la aplicación con las tarjetas (vistas) de todas las máquinas, éste es necesario que se actualice para que los datos sean lo más fiables posible.

Hacía finales de noviembre hicimos una función en JavaScript para que cada vez que recargue la aplicación sólo actualice el *body* y el resto de la página se mantenga estática.

FASE 7: Creación de un inicio de sesión

Durante la primera quincena de diciembre de 2018 creamos un *Login* para garantizar la seguridad de nuestra aplicación web. El *Login* contiene tres grupos de usuarios:

- **Admin**: es el administrador de la aplicación. Tiene acceso a todos los elementos de la aplicación y en un futuro será el encargado de añadir usuarios a la base de datos.
- **Operador**: tendrá acceso a todas las funcionalidades del sistema salvo la creación de nuevos usuarios.
- **Visor**: este usuario será el que más restricciones tenga ya que no podrá utilizar la botonera de funcionalidades *(comenzar, pausar, cancelar)* y tampoco deberá tener acceso a la conexión y desconexión de las impresoras 3D.

El resto del mes de diciembre se usó para crear una pequeña base de datos con *SQLite* que contiene los usuarios con los que cuenta la aplicación y realizar mejoras en la vista de la aplicación web, como por ejemplo incluir el usuario con el que nos hemos iniciado sesión previamente.

A.3. Estudio de viabilidad

A continuación vamos a llevar a cabo un estudio tanto de la viabilidad económica como legal de nuestra aplicación web.

Viabilidad económica

Para el desarrollo de este proyecto se ha necesitado un total de 4 meses en los que un sólo programador ha llevado a cabo todo el trabajo.

A continuación vamos a desglosar los costos de llevar a cabo este proyecto:

Coste personal

En el coste personal debemos tener en cuenta el salario mínimo de ingeniero [3] y la cuota de la seguridad social [3]. El costo total de personal en los 4 meses es el siguiente.

Concepto	Valor
Salario mensual	1215 €
Seguridad Social (23,60 %)	286 €
Total	1501 €
Total 4 meses	6004 €

Tabla A1: Coste Personal

Coste Hardware

Para llevar a cabo este proyecto necesitamos varios elementos hardware que son indispensables:

- BQ Witbox 2 [2]: necesitamos al menos dos impresoras 3D para que nuestro proyecto tenga sentido como monitor de una granja de impresoras 3D. Hemos elegido este modelo porque es la máquina con la que más familiarizamos estamos y es la que hemos usado para el desarrollo del proyecto.

- Zotac ZBox HD-ID11 [5]: un mini ordenador muy parecido al modelo que hemos usado nosotros. Es el ordenador que hará de servidor de la aplicación.

Concepto	Valor
BQ Witbox 2 (2 unidades)	2738 €
Zotac ZBox HD-ID11	186 €
Total	2924 €

Tabla A2: Costes Hardware.

Coste Software

Todo los programas que hemos usado para el desarrollo de este proyecto son completamente gratuitos por lo que este apartado no encarece el desarrollo final del producto.

Viabilidad legal

En el apartado legal llevaremos a cabo un estudio de las librerías utilizadas con sus correspondientes licencias y el uso que se le puede dar por parte de terceros.

Librería	Versión	Descripción	Licencia
Python	3.7.2	Lenguaje de programación utilizado.	PSFL
Flask	1.0.2	Frameowrk utilizado.	BSD
Requests	2.21.0	Librería para las peticiones a la API.	Apache 2.0
Collections	2.4.0	Tratamiento de estructuras de datos.	PSFL
os	2.4.0	Manejo de rutas del sistema.	PSFL
SQLAlchemy ORM	1.3.0	Librería para la creación de la base de datos.	MIT

Tabla A3: Licencias de las librerías utilizadas.

Las licencias que hemos usado en nuestro proyecto son:

- **PSFL** [12]: licencia de software libre permisiva como BSD y además es compatible con GPL.
- **BSD** [10]: licencia de software libre permisiva parecida a la licencia MIT. BSD tiene menos restricciones en comparación con otras licencias como GPL estando muy cercana al dominio público.
- **Apache 2.0** [9]: es un licencia de software libre pero requiere la conservación del aviso del derecho de autor y descargo de responsabilidad.
- **MIT** [11]: es una licencia de software libre permisiva procedente del Instituto Tecnológico de Massachusetts. Es compatible con otras licencias como *Copyleft* y GNU.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado vamos a llevar a cabo un estudio de los objetivos y requisitos establecidos previamente teniendo en cuenta las necesidades de la empresa **Abadía Tecnológica** antes del desarrollo del producto.

B.2. Objetivos generales

El objetivo principal de nuestra aplicación es conseguir controlar el estado de una granja de impresoras desde una misma aplicación y de una manera simple e intuitiva; de esta manera podemos saber si una impresora está operativa, está imprimiendo o ha tenido algún error de una manera rápida.

B.3. Catálogo de requisitos

A continuación vamos a definir los requisitos tanto funcionales como no funcionales con los que cuenta nuestra aplicación:

Requisitos funcionales

- **RF1-Monitorizar el sistema**: monitorizar el estado de una granja de impresoras 3D conectadas entre sí.
- **RF2-Sistema de usuarios**: crear un sistema de gestión de usuarios y permisos que garanticen la seguridad de la aplicación.

- **RF3-Comenzar impresión**: creación de un sistema que permita comenzar la impresión de una pieza desde nuestra aplicación.
- **RF4-Pausar impresión**: creación de un sistema que permita pausar la impresión en curso desde la propia aplicación.
- **RF5-Cancelar impresión**: creación de un sistema que permita cancelar una impresión en curso desde la propia aplicación.
- **RF6-Conectar impresora**: funcionalidad que permita conectar una impresora 3D a nuestra aplicación.
- **RF7-Desconectar impresora**: funcionalidad que permita desconectar una impresora 3D de nuestra aplicación.

Requisitos no funcionales

- **RNF1-Diseño adaptable**: la aplicación se adaptará dinámicamente a cualquier resolución de pantalla y a cualquier tipo de dispositivo.
- **RNF2-Usabilidad**: la aplicación será lo suficientemente intuitiva para que cualquier usuario sea capaz de utilizarla sin necesidad de ningún tipo de cualificación técnica.
- **RNF3-Autonomía**: la aplicación deberá recargarse ella misma cada pocos segundos con el fin de que la información sea lo más fiable posible.
- **RNF4-Escalabilidad**: debe ser posible y sencillo aumentar el número de máquinas con las que cuenta nuestra aplicación.

B.4. Diagrama de casos de uso

En la figura B1 podemos ver todos los casos de uso correspondientes a nuestra aplicación web en la que podemos diferenciar tres actores principales.

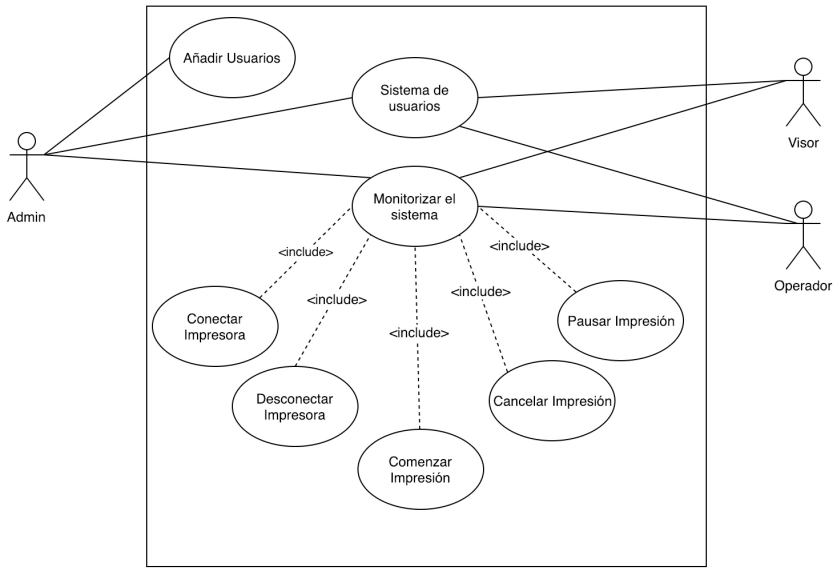

Figura B1: Diagrama de casos de uso.

B.5. Especificación de requisitos

RF 1	**Monitorizar el sistema**
Descripción	Se podrá visualizar la vista general de la aplicación con todas las impresoras 3D conectadas entre sí.
Precondiciones	Previamente, el usuario se ha identificado de manera correcta.
Acciones	El usuario visualiza el estado de todas la impresoras conectadas a la aplicación.
Postcondiciones	La aplicación muestra el estado de la impresoras.
Importancia	Alta.

Tabla B1: RF1 - Monitorizar el sistema

RF 2	**Sistema de usuarios**
Descripción	Nos podremos identificar en la aplicación dependiendo del tipo de usuario que seamos.
Precondiciones	Ninguna.
Acciones	El usuario deberá introducir un usuario y contraseña correcto con el fin de identificarse como un grupo de usuarios determinado.
Postcondiciones	Si el usuario y contraseña es correcto se mostrará una vista de la aplicación completa que dependerá del usuario con el que hemos iniciado sesión.
Importancia	Alta.

Tabla B2: RF2 - Sistema de usuarios

RF 3	**Comenzar impresión**
Descripción	Podremos comenzar la impresión de una pieza desde el propio monitor de la aplicación
Precondiciones	▪ Deberemos haber iniciado sesión previamente en la aplicación. ▪ Deberemos haber cargado previamente el G-code en la instancia de OctoPrint.
Acciones	El usuario deberá pulsar sobre el botón destinado para comenzar la impresión.
Postcondiciones	Comenzará la impresión de la pieza que haya sido cargada previamente.
Importancia	Alta.

Tabla B3: RF3 - Comenzar impresión

RF 4	**Pausar impresión**
Descripción	Podremos pausar la impresión en curso de una pieza desde el propio monitor de la aplicación
Precondiciones	▪ Deberemos haber iniciado sesión previamente en la aplicación. ▪ La impresora deberá estar imprimiendo.
Acciones	El usuario deberá pulsar sobre el botón destinado para pausar la impresión.
Postcondiciones	La impresión en curso se pausará indefinidamente.
Importancia	Alta.

Tabla B4: RF4 - Pausar impresión

RF 5	**Cancelar impresión**
Descripción	Podremos cancelar la impresión en curso de una pieza desde el propio monitor de la aplicación
Precondiciones	▪ Deberemos haber iniciado sesión previamente en la aplicación. ▪ La impresora deberá estar imprimiendo.
Acciones	El usuario deberá pulsar sobre el botón destinado para cancelar la impresión.
Postcondiciones	La impresión en curso se cancelará.
Importancia	Alta.

Tabla B5: RF5 - Cancelar impresión

RF 6	**Conectar impresora**
Descripción	Podremos conectar una impresora a nuestra aplicación.
Precondiciones	▪ La impresora deberá estar conectada mediante el puerto serie. ▪ La instancia de OctoPrint deberá estar lanzada.
Acciones	El usuario deberá pulsar sobre el botón destinado para conectar la impresora 3D.
Postcondiciones	La impresora se conectará a la aplicación.
Importancia	Alta.

Tabla B6: RF6 - Conectar impresora

RF 7	**Desconectar impresora**
Descripción	Podremos desconectar una impresora de nuestra aplicación.
Precondiciones	▪ La impresora deberá estar conectada mediante el puerto serie. ▪ La instancia de OctoPrint deberá estar lanzada. ▪ La impresora deberá estar conectada a nuestra aplicación.
Acciones	El usuario deberá pulsar sobre el botón destinado para desconectar la impresora 3D.
Postcondiciones	La impresora se desconectará de la aplicación.
Importancia	Alta.

Tabla B7: RF7 - Desconectar impresora

Apéndice C

Especificación de diseño

C.1. Introducción

A continuación vamos a especificar el procedimiento que hemos seguido para organizar nuestro proyecto y explicaremos cuáles han sido las razones para llevar a cabo el desarrollo de esta manera.

C.2. Diseño de datos

Es importante realizar un buen diseño de datos antes de ponernos a codificar nuestro proyecto, de esta manera podemos valorar la complejidad y la eficiencia de usar unas estructuras de datos u otras.

Todos los datos que recopilamos en nuestra aplicación están en formato *JSON* pero los datos se almacenan en un diccionario de diccionarios en *Python*. Esta estructura es la más óptima para nuestro proyecto ya que tenemos un diccionario y dentro de él tantos diccionarios como máquinas tenga nuestra aplicación con los datos correspondiente de cada máquina. Siguiendo esta estructura de datos es muy fácil para nosotros acceder a los datos desde el índice de la aplicación ya que solo tendremos que poner el nombre del diccionario principal y pasarle como clave la máquina de la que queremos obtener los datos; de esta manera obtenemos una lista con todos los datos que hemos guardado sobre la máquina seleccionada.

A continuación, en la figura C1, podemos ver la estructura en la que guardamos los datos de las máquinas.

```
valoresPrinter["Pausa"] = "-"
valoresPrinter["Imprimiendo"] = "-"
valoresPrinter["Lista"] = "-"
valoresPrinter["Estado"] = "-"
valoresPrinter["TempCamaActual"] = "-"
valoresPrinter["TempCamaFijada"] = "-"
valoresPrinter["TempHottendActual"] = "-"
valoresPrinter["TempHottendFijada"] = "-"
valoresPrinter["CamaDisponible"] = "-"

valoresJob["TiempoEstimadoImpresion"] = "-"
valoresJob["Nombre"] = "-"
valoresJob["TiempoUltimaImpresion"] = "-"
valoresJob["Porcentaje"] = "-"
valoresJob["TiempoImpresion"] = "-"
valoresJob["TiempoRestante"] = "-"
valoresJob["Estado"] = "-"
```

Figura C1: Ejemplo de cómo se organizan los datos.

En la figura C2 podemos ver un ejemplo de cómo accedemos a la información de los diccionarios desde el propio índice de la aplicación.

```
{% if datosPrinter[key]["CamaDisponible"] == "No"%}
<pre> Extrusor: {{datosPrinter[key]["TempHottendActual"]}} / {{datosPrinter[key]["TempHottendFijada"]}} º C</pre>
{% else %}
<pre> Extrusor: {{datosPrinter[key]["TempHottendActual"]}} / {{datosPrinter[key]["TempHottendFijada"]}} º C</pre>
{% if datosPrinter[key]["TempCamaActual"] > "0.0" %}
<pre> Cama: {{datosPrinter[key]["TempCamaActual"]}} / {{datosPrinter[key]["TempCamaFijada"]}} º C</pre>
{% endif %}
{% endif %}
```

Figura C2: Ejemplo de cómo accedemos a la información desde el índice de la aplicación.

C.3. Diseño procedimental

En el momento en el que la aplicación está en línea se abrirá una ventana para que iniciemos sesión; deberemos introducir un usuario y una contraseña y se comprobará si ese usuario está incluido en la base de datos. Si el usuario no está presente en la base de datos la aplicación nos devolverá un error y volverá cargar la página de inicio de sesión, mientras que si el usuario con el

que hemos iniciado sesión es correcto la aplicación cargará el monitor con los permisos que cuente el usuario que haya iniciado sesión.

A continuación, en la figura C3, vamos a ver un diagrama del diseño procedimental de nuestra aplicación.

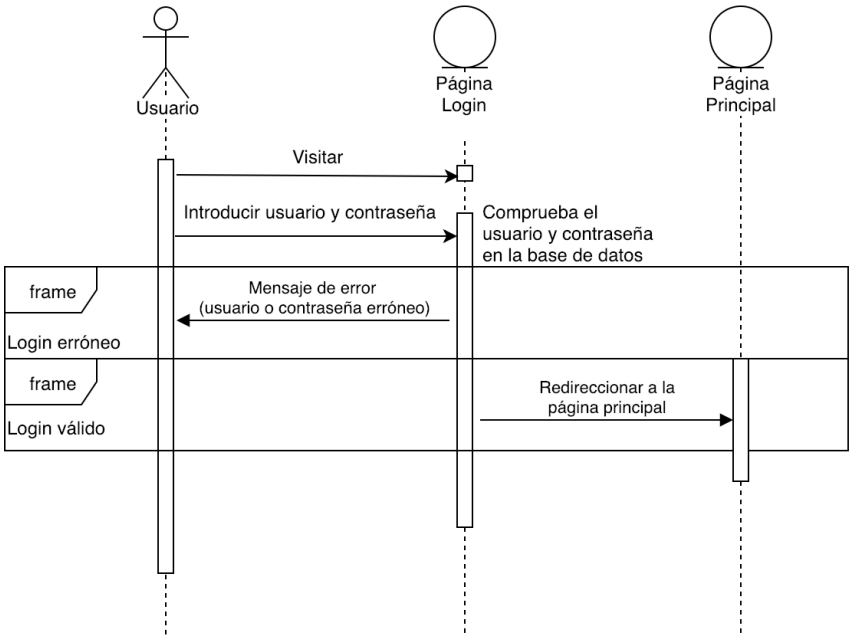

Figura C3: Ejemplo del diseño procedimental de nuestra aplicación.

C.4. Diseño arquitectónico

Para llevar a cabo la codificación de nuestro proyecto hemos utilizado tres paquetes principales:

- **Principal**: en este paquete tenemos toda la codificación principal de la aplicación, es decir, todo el *back-end* de la aplicación y la base de datos con los usuarios.
- **Static**: en este apartado metemos todos los estáticos de la aplicación los archivos con las imágenes, las fuentes, etc.
- **Templates**: aquí tenemos todos archivos *HTML* para mostrar todas las vistas de la aplicación como el *login* y la página principal.

Principal

Tal y como hemos dicho, en este apartado tenemos la parte principal de la aplicación que se llama *monitor.py*, los archivos CSV con todos los datos de las máquinas y todos los archivos de la base de datos que contienen los usuarios con los que está permitido acceder a nuestra aplicación web.

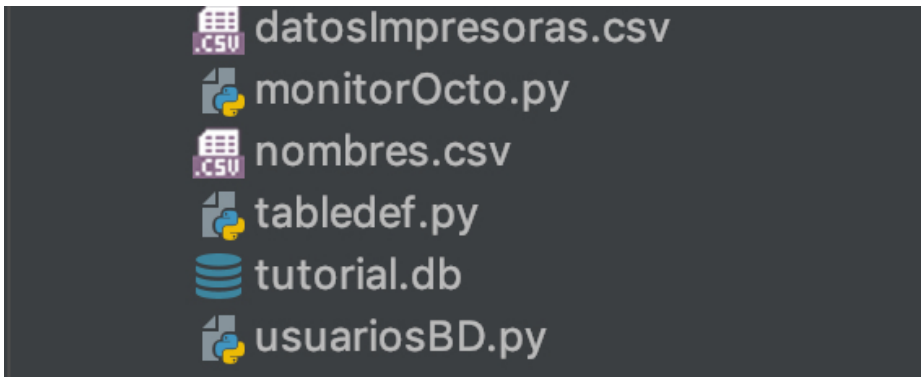

Figura C4: Paquete principal de la aplicación.

Static

En el apartado de los estáticos contamos con todos los archivos de las hojas de estilo, las imágenes, las fuentes, etc. Se organiza todo el carpetas tal y como vemos en la figura C5.

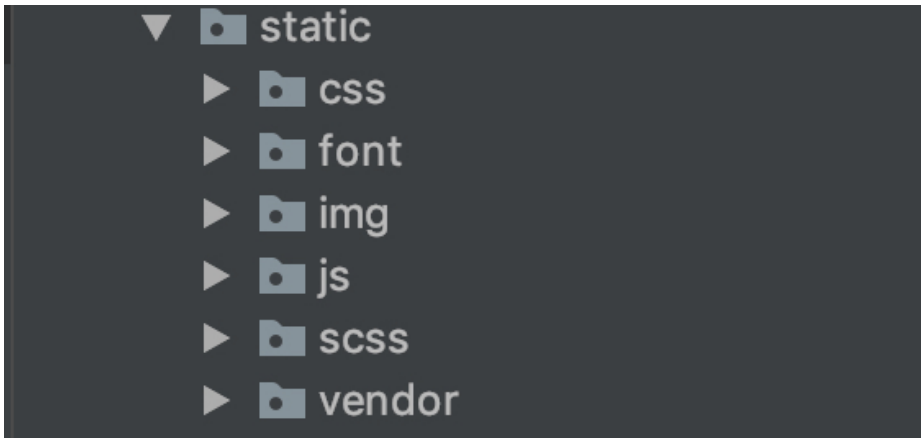

Figura C5: Paquete *static* de nuestra aplicación.

Templates

En el apartado de las plantillas tenemos todos los archivos *HTML* con los que cuenta nuestra aplicación. En nuestro caso tenemos un archivo que se llama *base.html* que cuenta con el *navbar* y el *footer* de la aplicación, por otro lado tenemos el *index.html* que se corresponde con las tarjetas que muestran la información de las máquinas y por último tenemos el *login.html* que es la página de inicio de sesión.

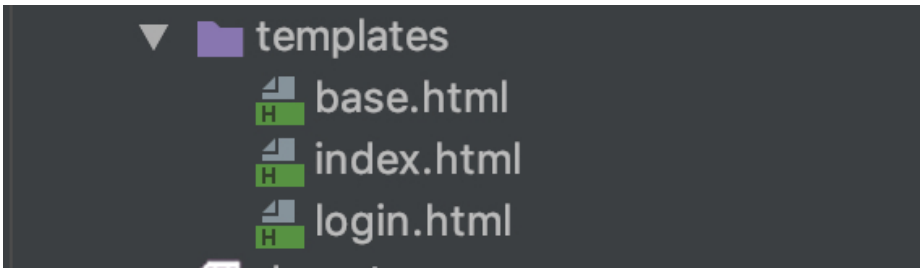

Figura C6: Paquete *Template* de nuestra aplicación.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apartado de los anexos se explican los conceptos relacionados con el manual del programador y con la compilación, instalación y ejecución del proyecto para que alguna persona pueda continuar con el desarrollo de la aplicación web.

D.2. Estructura de directorios

El desarrollo de nuestro proyecto se ha llevado a cabo utilizando los siguientes directorios principales:

- **FlaskApp**: contiene el código fuente del proyecto con todos los archivos necesarios para ejecutar la aplicación.
- **Documentación**: contiene la documentación para poder comprender correctamente el funcionamiento de nuestra aplicación.

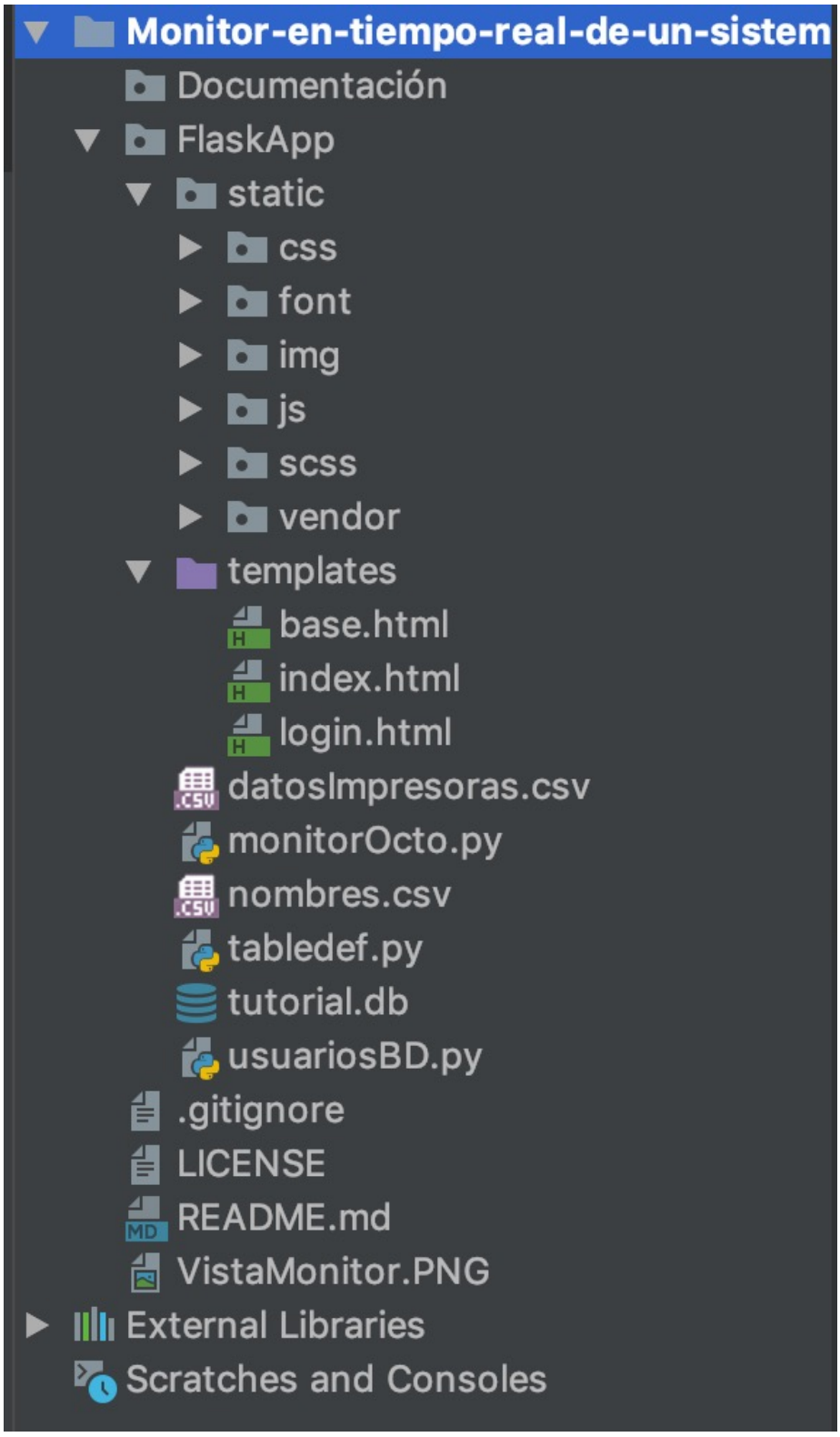

Figura D1: Estructura de los directorios del proyecto.

D.3. Manual del programador

Instalación de Anaconda

La forma más sencilla de instalar un intérprete de *Python* es instalando Anaconda. Para ello nos dirigimos a su web [1] y deberemos pulsar sobre el botón *Download* tal y como se ve en la siguiente figura:

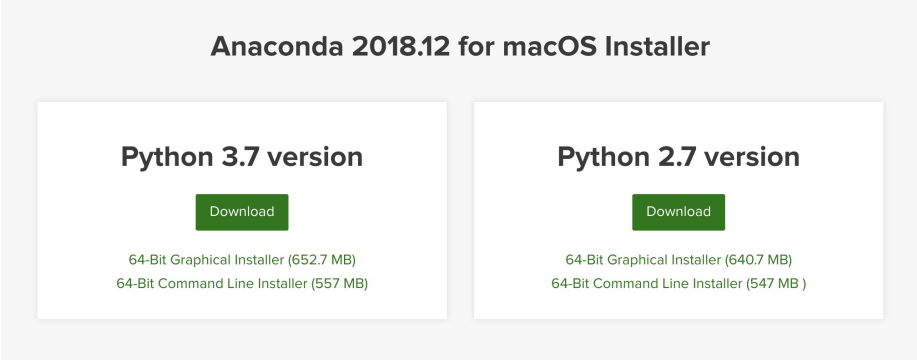

Figura D2: Descarga del intérprete de Python.

Una vez hayamos descargado el archivo deberemos seguir las introducciones del instalador que aparece en pantalla.

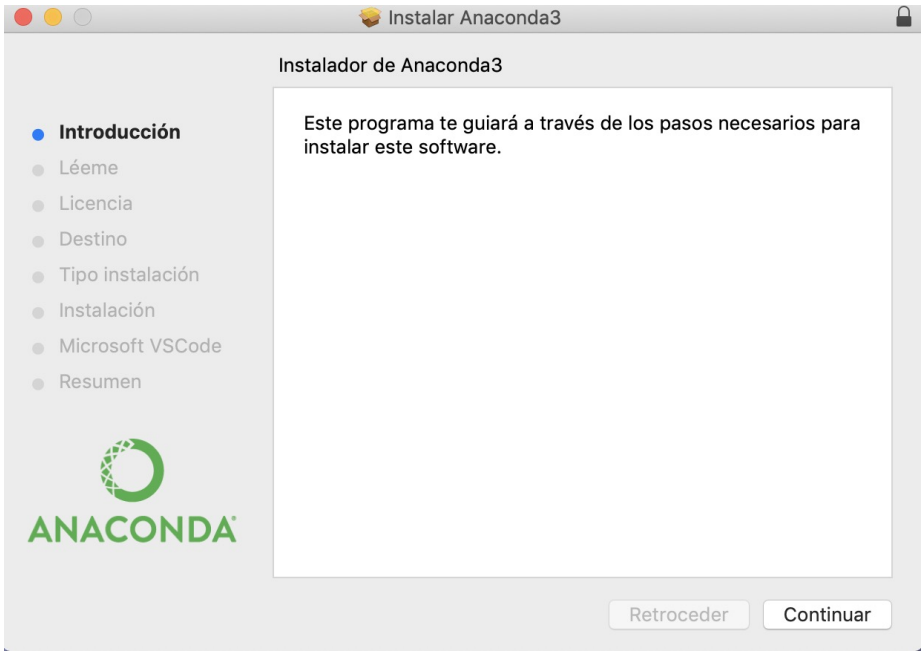

Figura D3: Instalador de Anaconda.

Instalación de PyCharm

PyCharm el *IDE* que hemos elegido para llevar a cabo la codificación de nuestro proyecto. Para instalar dicho entorno de desarrollo nos tenemos que dirigir a su página web [7] y pulsar sobre la versión que queremos descargar. Tal y como comentamos en la memoria gracias a la cuenta universitaria que nos otorga la Universidad de Burgos tenemos acceso a una licencia "Profesional"de este entorno de desarrollo, en nuestro caso es la versión que hemos utilizado.

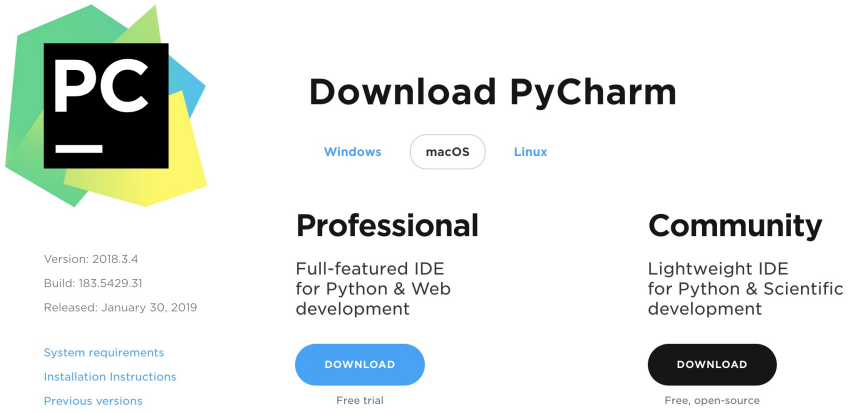

Figura D4: Descarga del entorno de desarrollo *PyCharm*.

Cuando la descarga se haya completado deberemos seguir el instalador como tantas veces hemos hecho.

Instalación de Sublime Text

Sublime Text es un completo editor de texto que hemos utilizado sobre todo para editar los archivos *HTML* que contienen las vistas de la aplicación. Para descargar éste editor de texto deberemos dirigirnos a su página web [8] y pulsar sobre el botón de descargar.

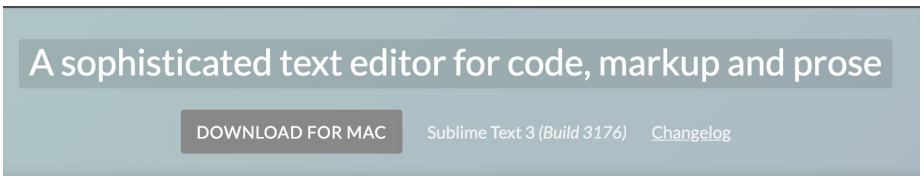

Figura D5: Descarga del editor de texto *Sublime Text*.

Cuando se haya terminado de descargar seguiremos el instalador.

Instalación de PuTTY

PuTTY es el programa que hemos usado para gestionar, mediante el protocolo *SSH*, el servidor donde está corriendo nuestra aplicación web. Para instalar éste programa debemos ir a su página web [6] y pulsar sobre el botón descargar.

Figura D6: Descarga del programa *PuTTY*.

Una vez descargado seguiremos los pasos del instalador.

D.4. Compilación, instalación y ejecución del proyecto

Para instalar y ejecutar el proyecto debemos tener en cuenta que necesitamos tener al menos una instancia de OctoPrint instalada y funcionando sobre un servidor. Cuando hayamos cumplido esta condición deberemos seguir los siguientes pasos para la ejecución del proyecto:

Importar el repositorio

Lo primero que debemos hacer es dirigirnos a la página web [4] del repositorio y descargarnos el proyecto desde el botón verde que pone *Clone or download* y luego en *download zip* tal y como se ve en la siguiente figura:

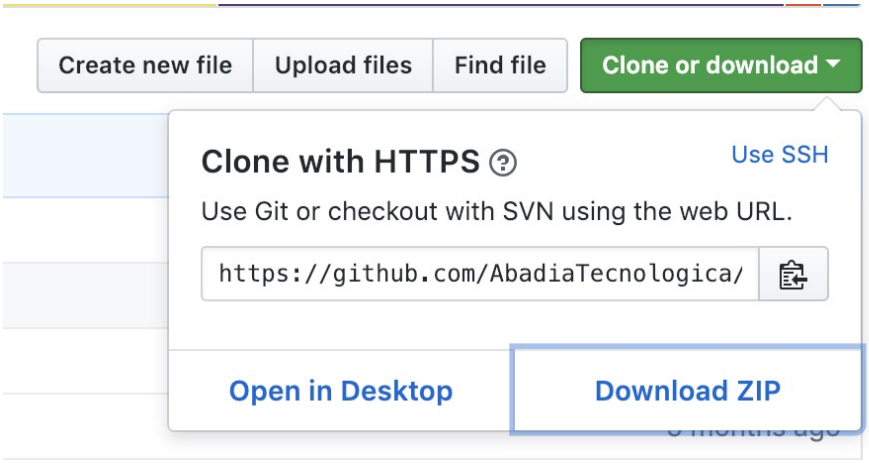

Figura D7: Botón para descargar el repositorio.

Una vez hemos descargado y descomprimido el proyecto nos debemos dirigir al entorno de desarrollo que estemos usando, en nuestro caso *PyCharm* y pulsar sobre el botón *Open*.

Figura D8: Botón para importar el proyecto.

Lo siguiente que debemos hacer es elegir el proyecto que acabamos de descargar y pulsar otra vez sobre el botón *Open* tal y como vemos en la siguiente imagen:

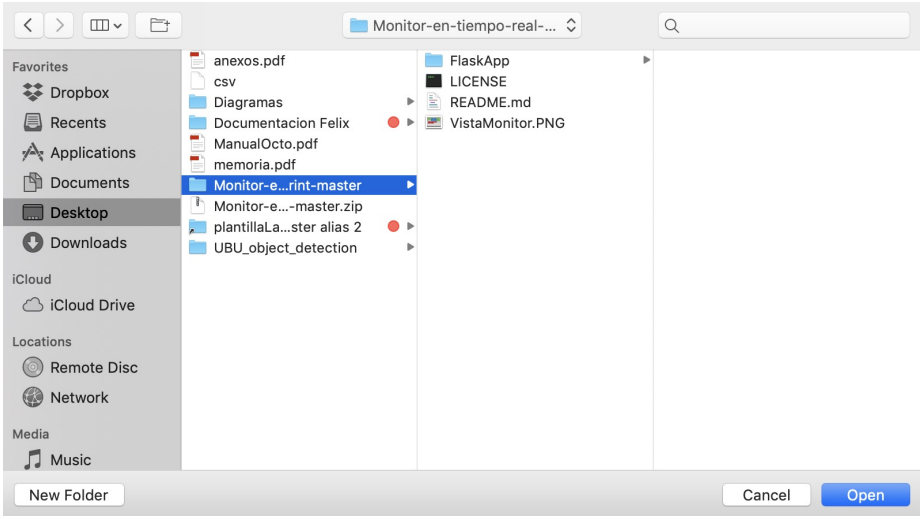

Figura D9: Elegir la ruta donde se encuentra el proyecto.

Si hemos seguido estos pasos correctamente se nos abrirá nuestro entorno de desarrollo con el proyecto que hemos descargado de *GitHub*.

Ejecución

Para ejecutar nuestro proyecto deberemos editar los archivos CSV que se encuentran dentro de nuestro proyecto y tendremos que copiar el token de API que nos proporciona OctoPrint y pegarlo en los archivos CSV que hemos comentado. Además también tendremos que cambiar el nombre de la impresora con el que queremos que se muestre en nuestra plataforma.

Otra cosa que debemos modificar es la dirección IP sobre la que está corriendo el servidor: para ello nos dirigimos al archivo *monitorOcto.py* y deberemos modificar la variable llamada *host* tal y como vemos en la siguiente figura:

```
1  #Direccion sobre la que corre el Octoprint.
2  host = "http://192.168.1.200"
```

Figura D10: Dirección sobre la que corre el servidor.

Una vez hayamos completado todos los pasos anteriores debemos posicionarnos sobre el archivo *monitorOcto.py* pulsar con el botón derecho y hacer click sobre la opción *run 'monitorOcto'* tal y como vemos en la siguiente figura:

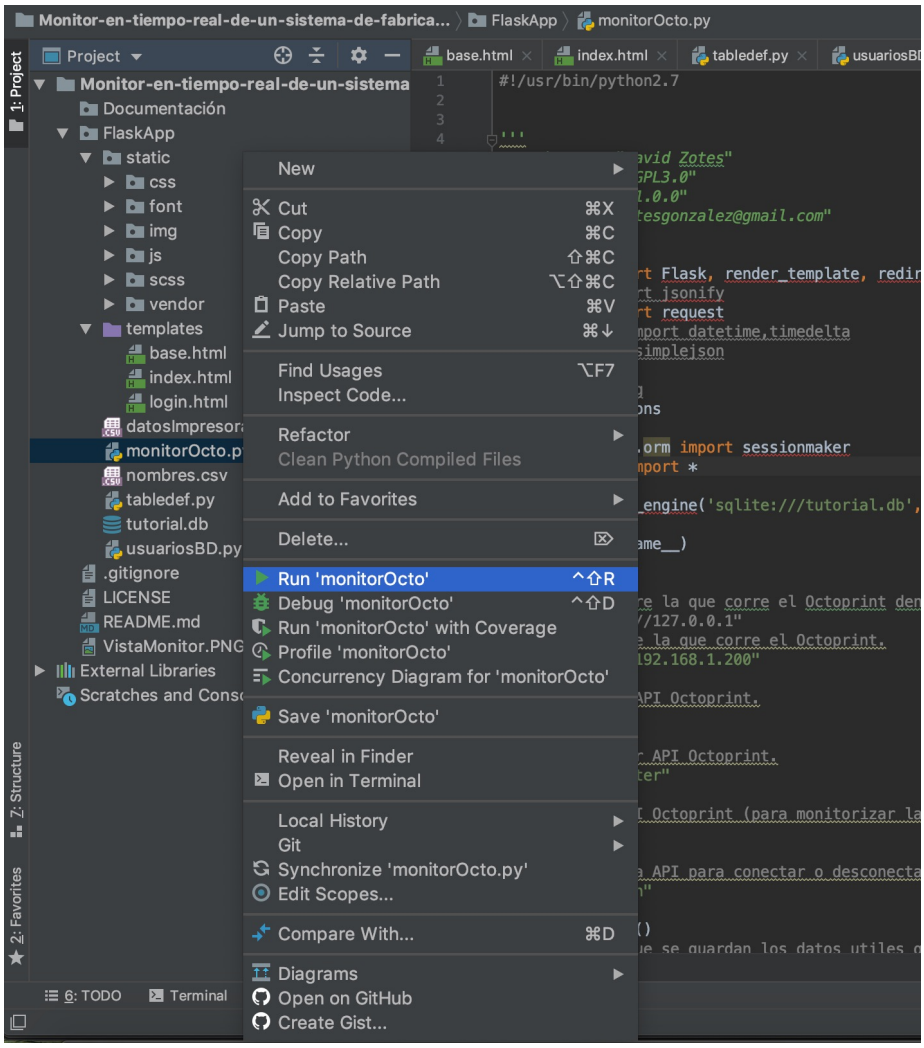

Figura D11: Ejecutar el proyecto.

Una vez esté el proyecto corriendo deberemos irnos a nuestro navegador predeterminado y escribir en la barra de busqueda *localhost* y nos cargará una página con el *login* de la aplicación que hemos desarrollado.

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado se detallan los requisitos que debemos cumplir y los pasos necesarios para llevar a cabo la instalación de la aplicación. Además explicaremos en el manual de usuario cómo se debe utilizar nuestra aplicación.

E.2. Requisitos de usuarios

A continuación vamos a enumerar los diferentes requisitos para poder usar nuestra aplicación:

- **Servidor**: deberemos contar con un servidor con *Debian* instalado. En nuestro caso hemos utilizado la version número 9 de esta de distribución de Linux, en ésta distribución de *Debian* ya viene preinstalado *Python* y los comandos para que funcione el protocolo *SSH* por lo que no será necesario instalarlo por separado.
- **Impresora 3D**: como no podía ser de otra manera necesitamos al menos una impresora 3D para conectar al servidor, aunque es recomendable utilizar la aplicación con más de una impresora.
- **Ordenador**: un ordenador con algún entorno de desarrollo desde el que podremos ejecutar o mandar al servidor ejecutar nuestra aplicación.
- **Hub USB**: necesitamos un hub USB para conectar todas la impresoras 3D al servidor.

E.3. Instalación

Para que la aplicación funcione correctamente debemos seguir cuidadosamente el siguiente apartado:

Instalación de OctoPrint

Tal y como comentamos en la memoria del proyecto debemos tener una instancia de OctoPrint por cada máquina que tengamos en nuestra aplicación. Para llevar un mejor control de cada máquina deberemos crear un usuario en el servidor por cada máquina que vayamos a instalar en la aplicación. Para ello introduciremos los siguientes comandos en una consola de comandos dentro del servidor:

1. *adduser impresor1*
2. *adduser impresor2*
3. *adduser impresor3*
4. *adduser impresor4*
5. *adduser impresor5*
6. *adduser impresor6*
7. *adduser impresor7*

Ahora deberemos instalar una instancia de OctoPrint en cada usuario que hemos creado. Para instalar OctoPrint deberemos introducir los siguientes comandos en una terminal dentro del servidor:

1. *git clone https://github.com/foosel/OctoPrint*
2. *cd OctoPrint*
3. *virtualenv venv*
4. *./venv/bin/pip install pip –upgrade*
5. *./venv/bin/python setup.py install*
6. *mkdir /.octoprint*

Una vez hemos seguido todos los pasos anteriores debemos introducir el siguiente comando para lanzar el servicio de OctoPrint y comprobar que funciona correctamente:

1. */OctoPrint/venv/bin/octoprint serve*

La primera vez que se lanza el servicio de OctoPrint debemos configurar los parámetros de la impresora que vayamos a utilizar tales como las medidas de la cama, etc.

Debemos repetir estos pasos con cada usuario que hayamos creado anteriormente.

E.4. Manual del usuario

A continuación explicaremos los pasos necesarios para utilizar nuestra aplicación de forma correctamente.

Login

Una vez lancemos nuestra aplicación se nos abrirá la página de inicio de sesión en la que deberemos introducir nuestro usuario y contraseña. En la siguiente figura vemos nuestra página de inicio de sesión.

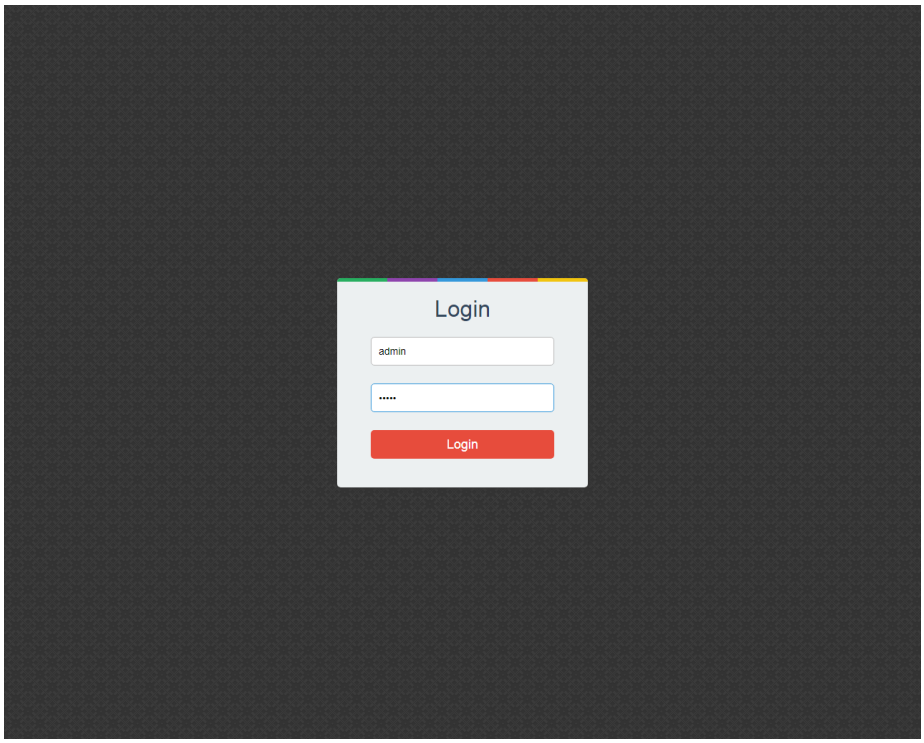

Figura E1: Página de inicio de sesión de nuestra aplicación.

En caso de que el inicio de sesión haya sido incorrecto la página nos indicará que ese usuario o contraseña no es correcto y tendremos que volver a introducir el usuario y la contraseña, mientras que si el inicio de sesión ha sido satisfactorio se abrirá la página principal de la aplicación con los permisos que tengamos dependiendo del usuario con el que hayamos iniciado sesión tal y como vemos a continuación.

Figura E2: Aplicación cuando hemos iniciado sesión como *Admin*.

A continuación podemos ver una vista de la aplicación cuando hemos iniciado sesión como *Visor*; podemos ver que en este caso no tenemos disponible ninguna funcionalidad, tan sólo es un visor para poder visualizar las maquinas pero no podemos ejercer ninguna acción sobre las máquinas.

Figura E3: Aplicación cuando hemos iniciado sesión como *Visor*.

Funcionalidad

Una vez hemos iniciado sesión en la aplicación tendremos acceso a las funcionalidades de la aplicación tales como comenzar, pausar o cancelar una impresión. Por ejemplo, cuando una máquina está *Operativa* quiere decir que está lista para comenzar una impresión y la tarjeta contará con la siguiente botonera:

Figura E4: Botonera cuando la impresora está operativa.

Desde esta botonera solo podemos comenzar la impresión ya que los

demás botones están desactivados hasta que comience la impresión.

En cambio cuando una impresora esta imprimiendo tendremos la siguiente botonera disponible:

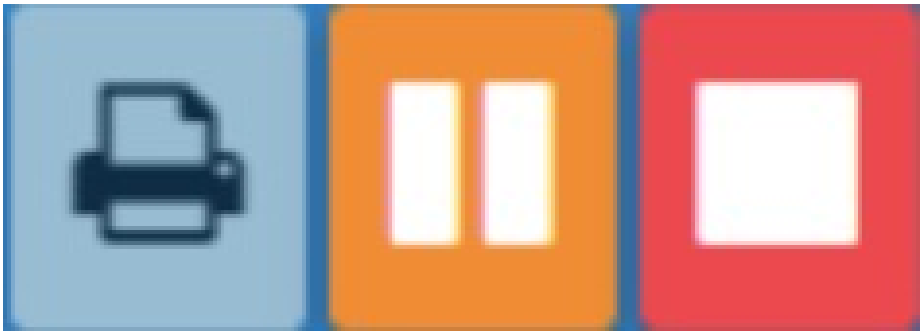

Figura E5: Botonera cuando la impresora esta imprimiendo.

Como la impresora ya está imprimiendo tendremos habilitados los botones de pausar y cancelar impresión, mientras que el botón de comenzar impresión estará deshabilitado.

En la parte de superior de nuestra aplicación (*navbar*) tenemos todos los nombres de las impresoras con las que cuenta nuestra aplicación. Si pulsamos sobre cada uno de esos nombres nos dirigirá a la instancia de OctoPrint de esa máquina seleccionada. En la parte derecha del *navbar* nos indicará el tipo de usuario con el que hemos iniciado sesión y un botón para cerrar la sesión actual que nos llevará otra vez a la página de inicio de sesión.

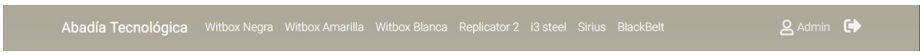

Figura E6: *Navbar* de nuestra aplicación.

Bibliografía

[1] Anaconda. Anaconda 2018.12 for macos installer. https://www.anaconda.com/distribution/#download-section, 2019.

[2] BQ. Bq witbox 2. https://www.bq.com/es/witbox-2, 2019.

[3] Gobierno de España. Seguridad social — bases y tipos de cotización 2019. http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537, 2019.

[4] Github. Monitor en tiempo real de un sistema de fabricación aditiva para octoprint. https://github.com/AbadiaTecnologica/Monitor-en-tiempo-real-de-un-sistema-de-fabricacion-aditiva-para-Octoprint, 2019.

[5] Pccomponentes. Zotac zbox hd-id11 barebone. https://www.pccomponentes.com/zotac-zbox-hd-id11-barebone, 2019.

[6] PuTTY. Download putty. https://www.putty.org/, 2019.

[7] Pycharm. Download pycharm. https://www.jetbrains.com/pycharm/download/#section=mac, 2019.

[8] Sublime Text. A sophisticated text editor for code, markup and prose. https://www.sublimetext.com/, 2019.

[9] Wikipedia. Apache license — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Apache_License, 2019.

[10] Wikipedia. Licencia bsd — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Licencia_BSD, 2019.

[11] Wikipedia. Licencia mit — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Licencia_MIT, 2019.

[12] Wikipedia. Python software foundation license — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Python_Software_Foundation_License, 2019.