UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Herramienta para la detección de anomalías en el suministro eléctrico en el ámbito asistencial

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: HÉCTOR GARCÍA COSTA

Tutor: VICENTE JAVIER JULIÁN INGLADA
JAVIER PALANCA CÁMARA

Curso 2017-2018

Resum

En aquest projecte s'ha desenvolupat una plataforma d'ajuda a l'assistència de persones majors basant-se en el consum elèctric de la llar. Per a açò s'ha dut a terme la creació d'una plataforma web que ens permet mostrar la informació del consum elèctric de la llar en forma de gràfiques i dades personals de la persona en qüestió. Al seu torn, tindrà un sistema d'alertes generades sobre la base d'una cerca sistemàtica d'anomalies fent ús de xarxes neuronals per a classificar les dades futures com a anomalia o dada normal.

Paraules clau: Web, Django, AgularJS, Docker, Keras, Xarxa neuronal, LSTM, Detecció d'anomalies, Clasificadors, Web Scrapping

Resumen

En este proyecto se ha desarrollado una plataforma de ayuda a la asistencia de personas mayores basándose en el consumo eléctrico del hogar. Para ello se ha llevado a cabo la creación de una plataforma web que nos permite mostrar la información del consumo eléctrico del hogar en forma de gráficas y datos personales de la persona en cuestión. A su vez, tendrá un sistema de alertas generadas en base a una búsqueda sistemática de anomalías haciendo uso de redes neuronales para clasificar los datos futuros como anomalía o dato normal.

Palabras clave: Web, Django, AgularJS, Docker, Keras, Red neuronal, LSTM, Detección de anomalías, Clasificadores, Web Scrapping

Abstract

In this project we have developed a support platform for the assistance of the elderly based on the household's electricity consumption. For this, we have created a web platform that allows us to show the information of the household electrical consumption in the form of graphs and personal data of the person in question. At the same time, it will have a system of alerts generated based on a systematic search for anomalies using neural networks to classify future data as anomaly or normal data.

Key words: Web, Django, AgularJS, Docker, Keras, Neural network, LSTM, anomaly detection, Classifier, Web Scrapping

Índice general

Índice de figuras

Índice de tablas

Definiciones, acrónimos y abreviaturas

- **Paciente:** Persona a la cual se le monitoriza el consumo eléctrico del hogar.
- **CUPS:** Identificación del contrato energético de un paciente con la compañía suministradora.
- **LSTM:** Sistema de aprendizaje profundo basado en redes neuronales recurrentes que soluciona el problema de gradiente de fuga. Es decir, permite aprender tareas en base a situaciones que ocurrieron muchos pasos de tiempo atrás.
- **Scrapping:** Técnica que permite extraer información de una web de forma automática simulando la navegación de un ser humano.
- **ARMA:** Modelo autorregresivo de media móvil. Herramienta que permite entender y predecir valores futuros de una serie de datos.
- **ORM:** Mapeo Objeto-Relacional. Convierte los datos de los objetos en un formato adecuado para guardar información en una base de datos. Así puede crear una base de datos virtual donde los datos que se encuentran en nuestra aplicación queden vinculados a la base de datos dando así persistencia.
- **Contenedor:** Espacios aislados que permiten ejecutar aplicaciones de forma independiente y aisladas. Aportan ligereza, portabilidad, autosuficiencia y flexibilidad.

CAPÍTULO 1
Introducción

1.1 Motivación

En los últimos años el número de mayores de 65 años ha incrementado de forma notoria. Actualmente hay 8,6 millones de personas mayores de 65 años en España, siendo esta cifra mayor que la de menores de 16 años. Según el instituto nacional de estadística se estima que, para el 2030, el 30 % de la población será mayor de 65 años. Conforme avanza la ciencia y la tecnología el ser humano ha logrado incrementar su longevidad, pero sigue habiendo problemas a esas edades que limitan la movilidad u otras capacidades generando así una dependencia. Gran parte de esta población vive sola en sus hogares y no dispone de solvencia económica que le permita contratar otros servicios de asistencia en el hogar. Este conjunto de personas podrían sufrir accidentes que les impidieran pedir auxilio, dejarse el gas abierto sin darse cuenta, el fuego encendido e innumerables sucesos que podrían perjudicar a dichas personas. Por ello vemos la necesidad de aumentar los servicios que se ofrecen a estos colectivos.

Viendo esta problemática planteamos un sistema que nos permita asistir a las personas con riesgo de exclusión y a las más vulnerables (ancianos, personas enfermas,...). Actualmente existen aparatos que permiten gestionar el consumo de los hogares y mantener un control de este incluso diferenciar entre distintos electrodomésticos, aunque plantea dos problemáticas. La primera, su diseño está pensado para controlar el consumo, permitiendo un ahorro en los hogares, de forma que necesitaría otro planteamiento interno para darnos los resultados deseados. La segunda, el coste de estos aparatos no permitiría que estuvieran a disposición de todo el mundo. Por otro lado hay otras formas de asistencia como pueden ser las residencias para ancianos o personas especializadas que dan soporte en el hogar. En estos casos la problemática fundamental es el coste de dichos servicios que no están al alcance de todos.

Actualmente las empresas prestadoras de suministro eléctrico están digitalizando sus contadores. En nuestro país existe un plan de sustitución de contadores eléctricos que planea sustituir todos los contadores por un modelo digital antes de finalizar 2018, los cual nos permitirá tener acceso a los datos de consumo eléctrico de forma sencilla.

Esto nos lleva a proponer una solución para dar asistencia a estas personas directamente en los hogares a través del consumo eléctrico y es por ello que plantearemos este proyecto.

1.2 Objetivos

El principal objetivo de este trabajo es dar asistencia a las personas más vulnerables (ancianos, personas enfermas,...). Para eso desarrollaremos una aplicación web con la intención de encontrar anomalías y notificarlas en el contexto del consumo eléctrico en hogares con necesidades en el ámbito asistencial.

Para llevar a cabo este objetivo necesitaremos hacer los siguientes subobjetivos:

1. Desarrollo de la web que contendrá las notificaciones e información de usuario. Habrá que definir como se estructurará la web y los datos que esta maneja, en nuestro caso haciendo uso de Django.

2. Obtención de los datos de consumo eléctrico de cada usuario de forma automática. A través de la web de Iberdrola, prepararemos un sistema de acceso automático que nos permita acceder a los datos de consumo una vez al día para conseguir el set de datos de cada usuario.

3. Búsqueda de anomalías. Desarrollar el buscador de anomalías. Para ello probaremos diferentes modelos y sistemas para la búsqueda de anomalías. Comenzaremos probando la predicción de datos para compararlos con los datos reales en cada lectura y evaluar si hay o no una anomalía, a su vez, probaremos el uso de herramientas ya desarrolladas de búsqueda de anomalías, y haremos pruebas con clasificadores basados en redes neuronales.

4. Evaluarlo. Para este paso necesitaremos evaluar los resultados obtenidos en cada modelo de forma independiente. En algunos casos bastará con gráficas de los resultados para poder descartar ese modelo o pasar a pruebas más exhaustivas. En otros casos como en los clasificadores haremos pruebas más detalladas de los resultados.

5. Implantar la búsqueda sistemática de anomalías en la plataforma. En este último paso será implantar el modelo elegido para la detección de anomalías en la plataforma para que funcione de forma sistemática para todos los usuarios.

1.3 Estructura de la memoria

Este trabajo esta estructurado en 8 capítulos. En el Capítulo 2 se explicarán conceptos relacionados con la detección de anomalías y se mostraran algunos estudios existentes y como estos han podido o no afectar a nuestro proyecto. En el Capítulo 3 haremos un análisis en profundidad del problema, estableciendo los requisitos de este e identificando como se puede abarcar este problema desde diferentes perspectivas. En el Capítulo 4 mostraremos la arquitectura de la plataforma web y tras eso daremos el diseño tanto a nivel de front-end como de back-end. Este capítulo finalizará explicando las tecnologías utilizadas para la totalidad del proyecto. El Capítulo 5 veremos como ha quedado el diseño final de la página web mediante capturas de esta. Por otro lado explicaremos el desarrollo elaborado en el ámbito de la detección de anomalías, explicando los tipos de anomalías que evaluaremos y los modelos finales generados. En el Capítulo 6 comentaremos la fase de despliegue de la aplicación web. En el Capítulo 7 evaluaremos los diferentes sistemas que hemos planteado durante este proyecto para la detección de anomalías y mostraremos los resultados para quedarnos con uno definitivo. Por último, en el Capítulo 8 se exponen las conclusiones finales del proyecto y se explica un posible trabajo futuro relacionado con este.

CAPÍTULO 2
Estado del arte

En este capítulo vamos a ver diferentes sistemas o modelos que se pueden utilizar para la detección de anomalías. Para poder comprender en su totalidad de lo que hablaremos vamos a explicar primero algunos conceptos básicos sobre la materia en cuestión. Finalizaremos con una breve conclusión que nos permitirá hacer una breve crítica al estado del arte.

2.1 Conceptos básicos

La detección de anomalías es un problema fundamental que ha generado muchas investigaciones desde diversas áreas. Podemos tener detectores de anomalías que abarcan dominios más genéricos y otros que lo hacen de forma más específica. Para comprender esto debemos primero comprender qué es una anomalía.

Las anomalías son patrones en los datos a evaluar que no pueden ser definidos como normales.

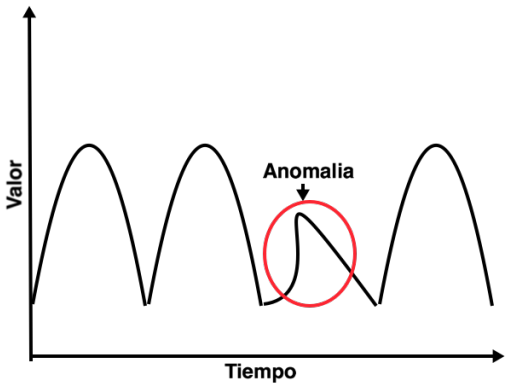

Figura 2.1: Ejemplo de anomalía

Un ejemplo sencillo se puede observar en la figura 2.1. Las anomalías pueden aparecer por diversos motivos. Por ejemplo, si analizamos datos de consumo en una tarjeta, podrían aparecer anomalías cuando exista algún tipo de fraude. Gracias a la investigación de este campo, podemos detectar variaciones en un sistema y con ello saber que

cambios están generándose, como apagones dentro de un hogar, actividad terrorista, o intrusión en las redes.

Existen numerosos libros y artículos que toman como tema principal la detección de anomalías. Algunos han dado un extenso estudio desarrollando técnicas mediante uso de machine learning [1]. Otros más específicos, que presenta un estudio de técnicas para la detección de anomalías en el ámbito de la ciberintrusión [2].

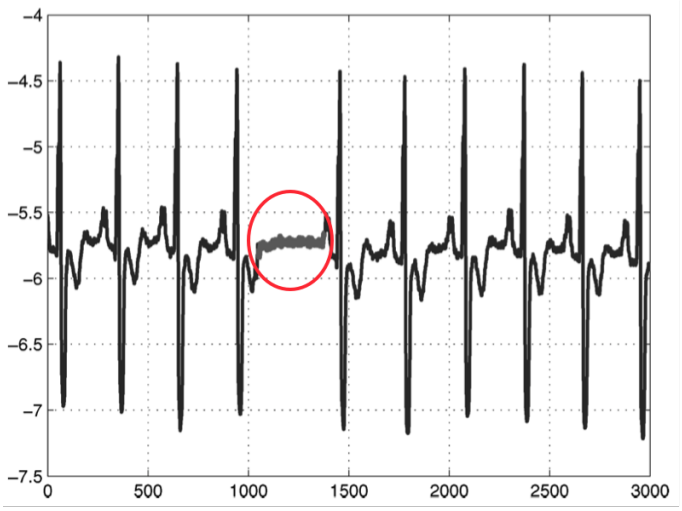

Figura 2.2: Ejemplo de anomalía colectiva

Un aspecto importante a tener en cuenta es la existencia de diferentes tipos de anomalías.

1. Anomalías puntuales. Si aparecen datos que pueden ser considerados anómalos con respecto al resto, como se puede ver en la figura 2.1. Estas son las anomalías más simples.

2. Anomalías contextuales. Son aquellas que tan sólo pueden clasificarse como anómalas en un contexto específico, también son conocidas como anomalías condicionales.

3. Anomalías colectivas. Estas se dan cuando aparece un conjunto de datos relacionados entre sí que forman una anomalía. Aunque por separado estos datos no tienen por qué ser anómalos, su ocurrencia colectiva los convierte en una anomalía, como en el ejemplo de la figura 2.2 en un electrocardiograma.

A su vez, tenemos tres técnicas distintas para la detección de anomalías.

1. Detección de anomalías supervisada. Esta técnica asume la existencia de un conjunto de datos etiquetados para el entrenamiento que diferencia clases normales de clases anómalas. De este modo, cualquier conjunto no evaluado se pasará por el modelo para determinar así a qué clase pertenece. Haciendo esto aparecen dos problemas fundamentales. Por un lado, la cantidad de datos normales es mucho mayor que la cantidad de datos anómalos. Por otro, la dificultad de obtener un etiquetado de clase preciso y representativo es un problema. Para esto se han propuesto técnicas que generan anomalías artificiales en un conjunto de datos normales para obtener así un conjunto etiquetado [3].

2. Detección de anomalías semisupervisada. Dicha técnica requiere unos datos de entrenamiento que contengan etiquetada la clase normal. Al no requerir etiquetas para las anomalías son más sencillas de aplicar que si usáramos detección supervisada. Para esto, se construye un modelo para la clase que corresponde al comportamiento normal, y se usa este para identificar las anomalías en los datos de prueba.

3. Detección de anomalías no supervisada. Haciendo uso de la técnica no supervisada, no requeriremos datos de entrenamiento, por tanto, será más fácilmente aplicable. Se establece la hipótesis de que la clasificación más común de los datos es la normal, frente a un menor conjunto de anomalías. Si esta suposición no es cierta encontraremos un sistema con una gran cantidad de falsas alarmas. Es posible adaptar técnicas semisupervisadas para ser no supervisadas mediante el uso de una muestra del conjunto de datos sin etiqueta como datos de entrenamiento.

Por otro lado, la forma en la que nos es dada una respuesta de la clase a la que pertenece un dato puede variar dependiendo de la técnica utilizada. Las técnicas de puntuación nos darán un valor que se expresa en función de lo que el modelo considera que ese dato es anómalo. A su vez, existe la técnica del etiquetado, la cual devuelve la clase a la que pertenece un dato. La ventaja de la primera es que permitirá a los investigadores analizar los resultados de forma más precisa y evaluar así si se considerará o no una anomalía.

2.2 Panorama actual

Actualmente el uso de redes neuronales para llevar a cabo tareas de clasificación o de predicción ha incrementado notablemente. Se han llevado a cabo investigaciones para la predicción de datos relacionados con el consumo eléctrico con anterioridad, aportando buenos resultados. Alguna de estas investigaciones trabajan con redes neuronales y máquinas de vector soporte [4]. Por otro lado podemos encontrar otras investigaciones que tratan de predecir el consumo eléctrico haciendo uso de diferentes modelos de red neuronal recurrente de tipo LSTM [5]. Las redes neuronales se han utilizado tanto para clasificación multiclase como para una sola clase. La técnica básica para detectar anomalías haciendo uso de clasificadores multiclase mediante redes neuronales se fundamenta en dos fases. Primero se entrena con los datos destinados a ello para aprender las diferentes clases. Tras eso, cada instancia de prueba se pasa como una entrada a la red neuronal. Si la red acepta la entrada de prueba, es normal y si la red la rechaza, es una anomalía. Se construye una red neuronal multicapa, que tendrá el mismo número de neuronas de entrada y de salida. En la fase de entrenamiento se comprimirán los datos en las capas ocultas y durante la fase de prueba, se reconstruirá cada instancia de los datos haciendo uso de la red entrenada para obtener la salida.

Las máquinas de vector soporte también se han utilizado para la búsqueda de anomalías en la clasificación de una sola clase. La máquina de vector soporte es un modelo que representa a los puntos de una muestra en el espacio. Este separa las clases en dos espacios lo más separados posibles mediante un hiperplano definido mediante el vector entre dos puntos de las dos clases más cercanos al cual llamaremos vector soporte. Cuando aparezcan nuevas muestras, se situarán correspondientemente a este modelo, en función del espacio al que pertenezca lo clasificaremos en una u otra clase. Algunas variantes de la técnica básica se han usado para la detección de anomalías en señales de audio [6].

Existen trabajos también sobre la detección de anomalías mediante sistemas basados en reglas. Así, cualquier instancia que no esté cubierta por una regla será considerada una anomalía. Este sistema puede usarse para clasificación multiclase y de una sola clase. A nivel básico, tendremos dos pasos a seguir, primero aprenderemos las reglas para el

conjunto de datos de entrenamiento utilizando un algoritmo de aprendizaje de reglas. Tras eso, tendremos que encontrar para cada instancia, la regla que mejor se ajusta a ella.

El análisis de vecinos más cercanos se ha usado también para la detección de anomalías. En este caso, se supone que los datos clasificados como normales aparecen en vecindarios más densos, mientras que los anómalos ocurren lejos de sus vecindarios. Para esto necesitaremos unas medidas de distancia o similitud entre dos instancias. A su vez, el concepto de clustering o agrupación se utiliza para la detección de anomalías agrupando instancias de datos similares en clústeres. Es una técnica no supervisada. Aunque el clustering no se usa principalmente para ello, se han desarrollado técnicas de detección de anomalías.

Por otro lado, existen investigaciones más enfocadas en la predicción, que tratan de resolver este mismo problema con métodos alternativos, como puede ser haciendo uso de un modelo ARMA mejorado mediante el uso de un algoritmo genético [7]. Para nuestro caso este modelo no nos funcionó correctamente por falta de tiempo en la implementación.

Encontramos también algunas librerías como Prophet de Facebook que permite de forma directa generar predicciones sobre los datos [1]. Otro enfoque para nuestro problema trata la detección de anomalías de forma directa. Algunas librerías facilitan este proceso, como puede ser luminol, desarrollada por Linkedin [8].

[1] https://facebook.github.io/prophet/

2.3 Conclusiones

Para llevar a cabo la detección de anomalías hemos hecho uso de diferentes herramientas durante el transcurso de todo el proyecto. Algunas han sido más útiles o nos han dado mejores resultados y han llegado hasta el final del proyecto con nosotros, mientras que otras han dado resultados que no nos eran de utilidad y hemos tenido que rechazarlas.

Los sistemas basados en reglas dificultaban el problema que queremos plantear puesto que el número de reglas que deberíamos hacer se incrementaba de forma notoria y no daría los resultados buscados. Por otro lado, el clustering o agrupación no están enfocados de forma directa para detección de anomalías, como comentamos anteriormente y aunque existen usos de este, son más complejos.

Otro modelo que descartamos fue el uso de ARMA junto a un algoritmo genético. Hicimos varias pruebas pero no conseguimos un sistema funcional dado que las librerías existentes no permitían hacerlo y debíamos crear todo desde cero. Tras diferentes intentos y viendo los resultados obtenidos decidimos descartarlo por falta de tiempo.

Las librerías como Luminol de LinkedIn no nos dieron buenos resultados puesto que las anomalías que era capaz de detectar no coincidían con lo que nosotros buscábamos puesto que las de nuestro sistema dependen en gran medida de un contexto que no detectaba. Por otro lado, la librería Prophet de Facebook tras probarla vimos que los resultados daba resultados incoherentes debido a que nuestros datos varían su estado con intensidad.

La base de nuestro proyecto a nivel de búsqueda de anomalías es la librería de Python Keras, haciendo uso de diferentes modelos de clasificadores para cada anomalía.

CAPÍTULO 3
Análisis del problema

3.1 Introducción

La plataforma desarrollada en este proyecto tiene como objetivo facilitar la asistencia del hogar a personas mayores. El proyecto se ha podido llevar a cabo haciendo uso del framework de Python Django para la creación de la web y la librería de aprendizaje automático Keras, también de Python, para crear los modelos de clasificador en la detección de anomalías. Gracias a Django podremos plantear el patrón de diseño modelo-vista-controlador, facilitando así el diseño de base de datos que se generará mediante objetos en Python. Keras por su parte nos permitirá generar de forma sencilla los modelos que explicaremos en los siguientes apartados.

Se hará uso del software Selenium para la obtención de los datos de consumo eléctrico scrappeando la web de Iberdrola. Y todo esto irá desplegado sobre contenedores Docker para permitirnos tener aislada cada aplicación y asegurarnos así un mejor funcionamiento del sistema.

Con estas tecnologías hemos desarrollado una plataforma que cumplirá los requisitos estipulados en la Sección 3.2. A su vez explicaremos las tecnologías utilizadas de forma más detallada en la Sección 4.3.

3.2 Especificación de requisitos

La especificación de requisitos tiene como objetivo la descripción completa de la plataforma, mostrando sus funciones y definiendo aquellos conceptos fundamentales para entender este proyecto. Nos basaremos para esto en el estándar IEEE 830.

El propósito de este proyecto es crear una plataforma que permita al administrador visualizar los datos de los hogares subscritos al sistema y gestionar sus notificaciones o alertas. Haciendo uso de esta especificación de requisitos podremos diseñar la arquitectura completa de esta plataforma.

La plataforma en cuestión deberá cumplir las siguientes funcionalidades:

- Generación automática de notificaciones.
- Creación, eliminación y modificación de pacientes.
- Creación, eliminación y modificación de contratos.
- Creación, eliminación y modificación de consumos.
- Gestión de notificaciones.
- Representación gráfica de los datos de consumo.
- Búsqueda de pacientes por nombre o dirección.
- Representación gráfica del objetivo mensual del contrato.

Para el uso de esta plataforma solo será necesario un navegador web.

3.2.1. Requisitos funcionales

Con los requisitos que vamos a plantear a continuación podremos establecer las funcionalidades que nos permitirá esta aplicación.

Identificador	RF01
Nombre	Acción crear paciente
Descripción	Posibilidad de crear un nuevo paciente con nombre, dirección, y usuario de Iberdrola

Tabla 3.1: RF01 - Crear paciente

Identificador	RF02
Nombre	Acción eliminar paciente
Descripción	Posibilidad de eliminar un paciente existente

Tabla 3.2: RF02 - Eliminar paciente

Identificador	RF03
Nombre	Acción modificar paciente
Descripción	Posibilidad de modificar un paciente existente

Tabla 3.3: RF03 - Modificar paciente

Identificador	RF04
Nombre	Acción crear contrato
Descripción	Posibilidad de crear un contrato con nombre y paciente correspondiente.

Tabla 3.4: RF04 - Crear contrato

Identificador	RF05
Nombre	Acción eliminar contrato
Descripción	Posibilidad de eliminar un contrato existente.

Tabla 3.5: RF05 - Eliminar contrato

Identificador	RF06
Nombre	Acción modificar contrato
Descripción	Posibilidad de modificar un contrato existente.

Tabla 3.6: RF06 - Modificar contrato

Identificador	RF07
Nombre	Acción visualizar notificaciones
Descripción	Posibilidad de visualizar todas las notificaciones de un paciente.

Tabla 3.7: RF07 - Visualizar notificaciones

Identificador	RF08
Nombre	Acción visualizar gráfica
Descripción	Posibilidad de seleccionar de que contratos de muestran gráficamente los consumos.

Tabla 3.8: RF08 - Visualizar gráfica

Identificador	RF09
Nombre	Acción seleccionar intervalo gráfica
Descripción	Posibilidad de seleccionar en que intervalo de tiempo se muestran los datos de consumo eléctrico.

Tabla 3.9: RF09 - Seleccionar intervalo gráfica

Identificador	RF10
Nombre	Acción buscar paciente
Descripción	Posibilidad de buscar un paciente por nombre o dirección.

Tabla 3.10: RF10 - Buscar paciente

Identificador	RF11
Nombre	Acción crear consumo
Descripción	Posibilidad de crear un consumo.

Tabla 3.11: RF11 - Crear consumo

Identificador	RF12
Nombre	Acción eliminar consumo
Descripción	Posibilidad de eliminar un consumo existente.

Tabla 3.12: RF12 - Eliminar consumo

Identificador	RF13
Nombre	Acción modificar consumo
Descripción	Posibilidad de modificar un consumo existente.

Tabla 3.13: RF13 - Modificar consumo

Identificador	RF14
Nombre	Acción seleccionar intervalo
Descripción	Posibilidad de seleccionar el intervalo de tiempo a mostrar en la gráfica de consumo.

Tabla 3.14: RF14 - Seleccionar intervalo

Con estos requisitos permitiremos al usuario en cuestión, en nuestro caso administrador, gestionar todo lo relacionado con pacientes, consumos, contratos y notificaciones, visualizar los datos de consumo de cualquier fecha que estén en el sistema, filtrar a los pacientes y elegir que contrato de un paciente se mostrará gráficamente.

3.3 Identificación y análisis de soluciones propuestas

El análisis que llevaremos a cabo a continuación hace referencia únicamente a la parte de detección de anomalías pues es la que nos ha dado un amplio campo de estudio de herramientas y modelos con algunos de los cuales hemos trabajado y elegido o descartado por sus resultados y otros que no llegaron a las pruebas tras ser descartados por otros motivos que explicaremos detenidamente.

En un principio comenzamos planteando un sistema de predicción de datos de consumos del futuro próximo para evaluar si el dato real se correspondía de forma razonable con la predicción para así decidir si es un dato normal o no. Para ello usamos una librería llamada Prophet. Esta librería es fácil de utilizar, lo cual nos permitió hacer pruebas de forma rápida y sencilla. Los resultados obtenidos para nuestro conjunto de datos era altamente ineficiente. Esto se debía a la inestabilidad de los datos. Para un conjunto de datos más estable podríamos haber utilizado esta alternativa como base del proyecto.

Tras esto comenzamos a plantear un sistema que hacía uso de un modelo ARMA para series de datos temporales unido a un algoritmo genético que mejora los resultados del modelo. No llegamos a finalizar este sistema por falta de tiempo. Implementarlo estaba generando muchos problemas y al encontrar otras opciones decidimos descartarlo sin llegar a obtener resultados para evaluar. Con más tiempo sería interesante llevar a cabo este sistema para comprobar su eficacia.

Las redes neuronales estuvieron en mente desde el inicio del proyecto, pero no fue hasta casi el final de este cuando decidimos probarlas para nuestro propósito. Comen-

zamos planteando un modelo en Keras mediante redes neuronales recurrentes, concretamente LSTM. Hubieron varios parámetros que fuimos ajustando para acercarlo más a la solución deseada y conseguimos un sistema que funcionaba con una buena precisión, pero se adecuaba demasiado a la ventana temporal pasada prediciendo así los resultados futuros de forma correcta aun cuando existía una anomalía en estos. De esta forma no se podía automatizar la detección de anomalías en base a la predicción puesto que el resultado obtenido era semejante siempre al real.

Los resultados hasta ahora no eran suficientes con nada de lo probado, por lo que decidimos cambiar el enfoque y usar directamente clasificadores estableciendo cada consumo como normal o anómalo.

La solución que propusimos al final fue el uso de la librería Keras para crear modelos de clasificadores. En principio intentamos hacer un clasificador multiclase haciendo uso de capas LSTM y capas densas que fuera capaz de clasificar en la clase que le correspondiera a cada dato. Este modelo dio buenos resultados en alguna de las clases si se analizaban por separado, pero otras daban resultados demasiado erróneos. Por ello, en lugar de crear un modelo multiclase, hicimos un modelo para cada clase anómala a clasificar. De este modo los resultados de todas las clases fueron adecuados y útiles, por lo que decidimos usarlo en este proyecto.

CAPÍTULO 4
Diseño de la solución

4.1 Arquitectura del sistema

Para poder definir la arquitectura del sistema hemos diseñado un diagrama que simplifica la comprensión de este. El diagrama incluye cada parte del sistema y el contenedor correspondiente en el que está desplegado.

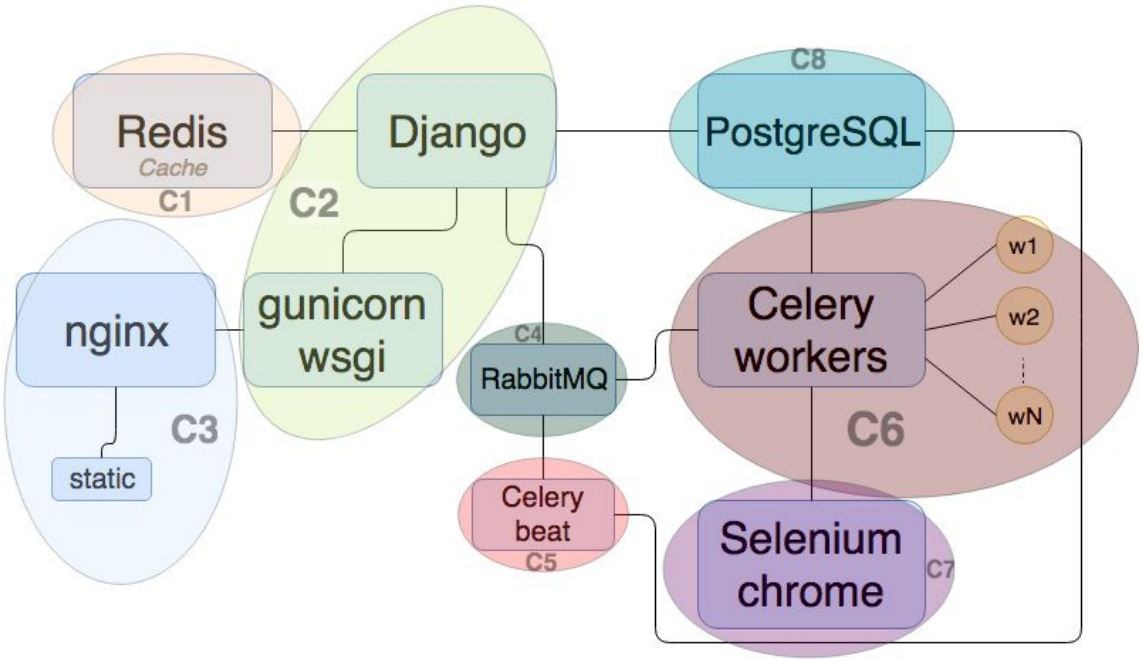

Figura 4.1: Diagrama de la arquitectura.

Como podemos observar en la figura 4.1, tendremos diferentes sistemas para la gestión de los datos de la plataforma.

Django es la base de toda esta arquitectura. Es el framework que facilitará los componentes necesarios para crear una página web de forma sencilla y eficiente. Este deberá poder solicitar datos a través de diferentes servidores y bases de datos como veremos a continuación y es por ello que necesitará de algunos intermediarios para ello. Además llevará toda la gestión de urls. Todo esto lo hará en Python, lo cual nos es útil en este proyecto dado que la base para la detección de anomalías la haremos en Python.

Por una lado, Django gestiona los datos de la cache a través de Redis. Este es un motor de base de datos en memoria, basado en el almacenamiento en tablas hash. Es rápido y eficiente y es por ello que tiende a utilizarse para almacenamiento en caché.

PostgreSQL por su parte se encargará de la gestión de los datos del servidor. Este es un sistema de gestión de bases de datos relacional orientado a objetos. Contendrá todo aquello que no esté en caché ni sea estático.

Gunicorn es un servidor HTTP que hace de Interfaz de puerta de enlace del servidor web de Python. En nuestro caso lo utilizaremos como intermediario entre Django y nginx. Es compatible con Django, liviano en los recursos del servidor y rápido.

Nginx es un servidor web/proxy inverso, ligero, de alto rendimiento y un proxy para protocolos de correo electrónico. En nuestro caso recibirá las peticiones directamente de modo que en caso de recibir una petición que se encuentre como archivo estático la devolverá directamente y en cualquier otro caso redirigirá la petición a través de gunicorn hacia Django. Este esquema nos permitirá trabajar de forma más rápida y sencilla.

Para ejecutar tareas de larga duración en segundo plano dentro de la plataforma, haremos uso de Celery worker. Esto permitirá crear diferentes tareas que después serán lanzadas de forma temporizada a través de Celery beat. Esto irá relacionado estrechamente con PostgreSQL puesto que los datos informativos de cuando lanzar dichas tareas estará almacenado ahí y además cuando una tarea lo necesite podrá almacenar la información deseada en dicha base de datos. Para poder gestionar esas tareas a través de Django, haremos uso de RabbitMQ, que es una cola de mensajes que actúa como middleware entre emisores y receptores. Así cuando se quiera crear una tarea se hará desde Django y se enviará a través de RabbitMQ para que Celery worker las cree. A su vez, para crear el lanzamiento temporal también hará de intermediario RabbitMQ entre Celery beat y Celery worker.

El uso de Selenium chrome en nuestro sistema es necesario puesto que llevaremos a cabo una tarea de scrappeo (obtención de información de una pagina web simulando un navegador). En este caso será un navegador Chrome headless, es decir, sin interfaz. Se lanzará a través de Celery una vez al día y adquirirá los datos de consumo eléctrico de los usuarios de la web a través del portal de Iberdrola Distribución [1]. Estos datos se almacenarán en PostgreSQL. Esta tarea se explicará detalladamente en el apartado de Desarrollo, en cual comentaremos el por qué es necesaria y todo su funcionamiento.

Toda la arquitectura que hemos mostrado anteriormente está lanzada en contenedores Docker. Esto permite tener independencia del sistema operativo y automatizar el despliegue de las aplicaciones. A su vez proporcionará una capa de abstracción que permitirá aislar los recursos de forma independiente cuando sea necesario. Como vemos en la figura 4.1, tendremos cada aplicación desplegada de forma independiente, teniendo un total de 8 contenedores distintos desplegados. Django y gunicorn estarán lanzados en un mismo contenedor, mientras que el resto estarán en contenedores separados. Nginx por su parte, contendrá los datos estáticos en su contenedor para facilitar la transmisión de estos de forma rápida.

Con esta arquitectura podremos construir nuestra aplicación de forma que sea eficiente y fácil de llevar a producción cuando llegue el momento y cada parte es necesaria para el funcionamiento total del sistema tal y como explicamos anteriormente.

[1] https://www.iberdroladistribucion.es

4.2 Diseño detallado

En esta parte vamos a centrarnos en el diseño de la página web. Para ello mostraremos el diseño de la parte de back-end, la cual se centra en los datos internos y como interactúan entre si, y la parte front-end, que se centra más en la visualización final de la web.

4.2.1. Back-end

Para comprender el diseño seguido en la base de datos, mostraremos a continuación un diagrama de entidad-relación para representar la estructura de los datos en el sistema y sus relaciones.

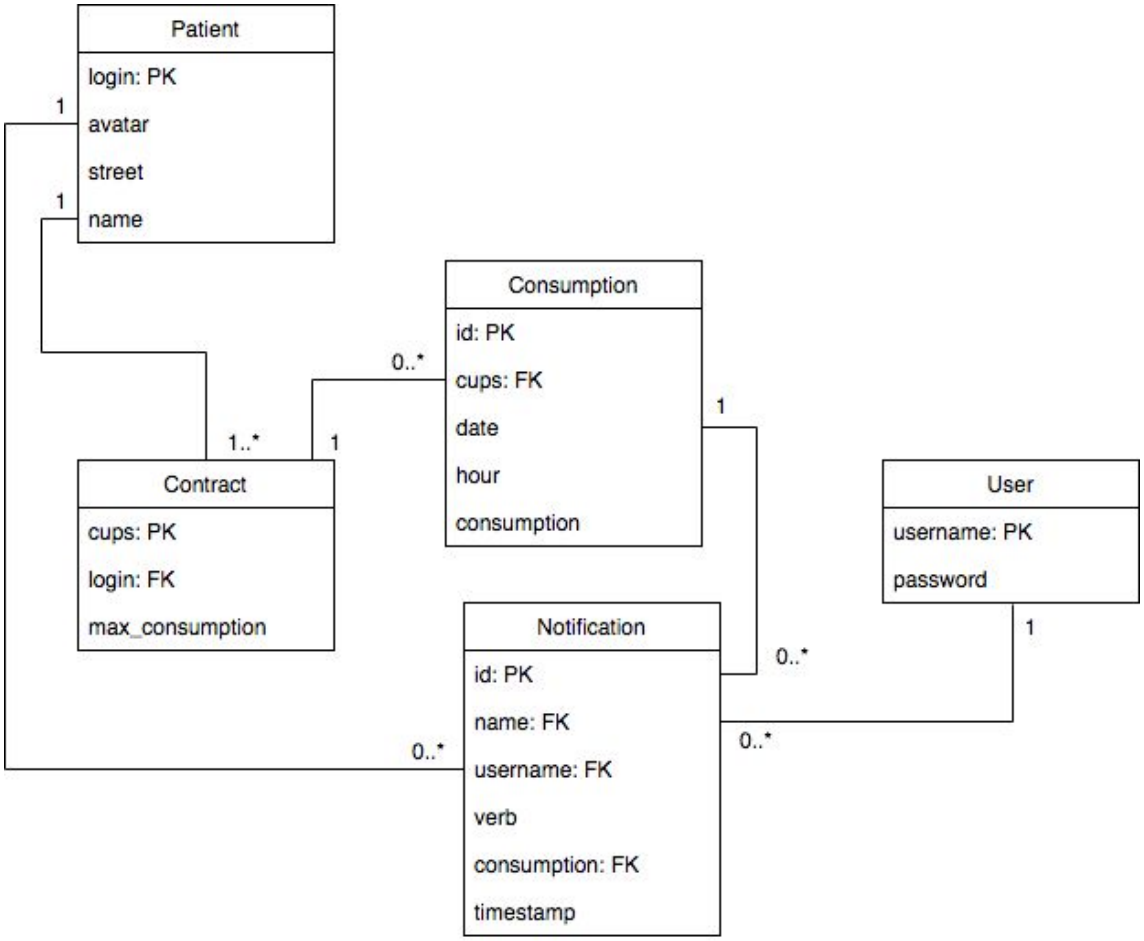

Figura 4.2: Diagrama de la base de datos

Como vemos en el diagrama de la figura 4.2 tenemos en nuestro sistema 5 clases distintas. Por un lado tendremos a los pacientes. Estos estarán identificados por el login, es decir, el usuario de acceso a la web de Iberdrola. Esta clase deberá contener también un avatar, una calle y un nombre. Por otro lado tenemos los contratos, los cuales se identifican por su CUP. Tendrá datos de login a la web de Iberdrola y el consumo máximo establecido. Los consumos por su parte se identifican por un identificador que se establece de forma automática. Deberán estar asociados a un contrato a través de un Código Universal de Punto de Suministro (CUPS). Este es un código único que comienza por ES y que identifica un punto receptor de abastecimiento de toda la red de luz y tendrán

fecha, hora y valor de consumo. Las notificaciones se identifican al igual que los consumos con un identificador automático. Estas deberán tener un nombre, un administrador asociado, un texto descriptivo, el identificador del consumo que genera la notificación y una fecha de lectura de consumo. Por último tenemos la clase usuario que se identifica por un nombre y tiene una contraseña establecida. Esta clase es en la que se encuentran los administradores.

Las interacciones de las clases son sencillas. Un paciente tendrá asociado un contrato o más de uno en caso de tener más de un hogar inscrito. A su vez, el paciente puede tener cero o muchas notificaciones, las cuales serán creadas por un administrador e irán asociadas a un consumo, y dicho consumo tendrá que ir asociado a un contrato.

4.2.2. Front-end

Para el front-end tendremos un conjunto de vistas, controladores y servicios con los que interactuará de forma directa el usuario. Esto será gestionado a través de Django y angularJS.

Mostraremos ahora algunos mockups que nos permitirán definir el diseño de la web que verá directamente el usuario.

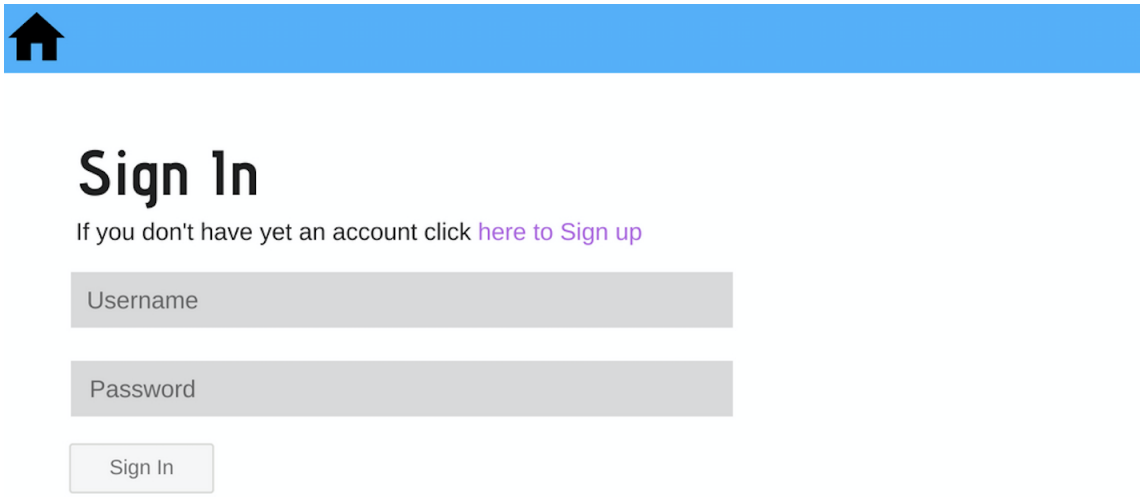

Figura 4.3: Inicio de sesión

En cuanto accedemos a la plataforma lo primero que se solicita es la identificación. Esto es así puesto que es un sistema ideado para el uso de un administrador.

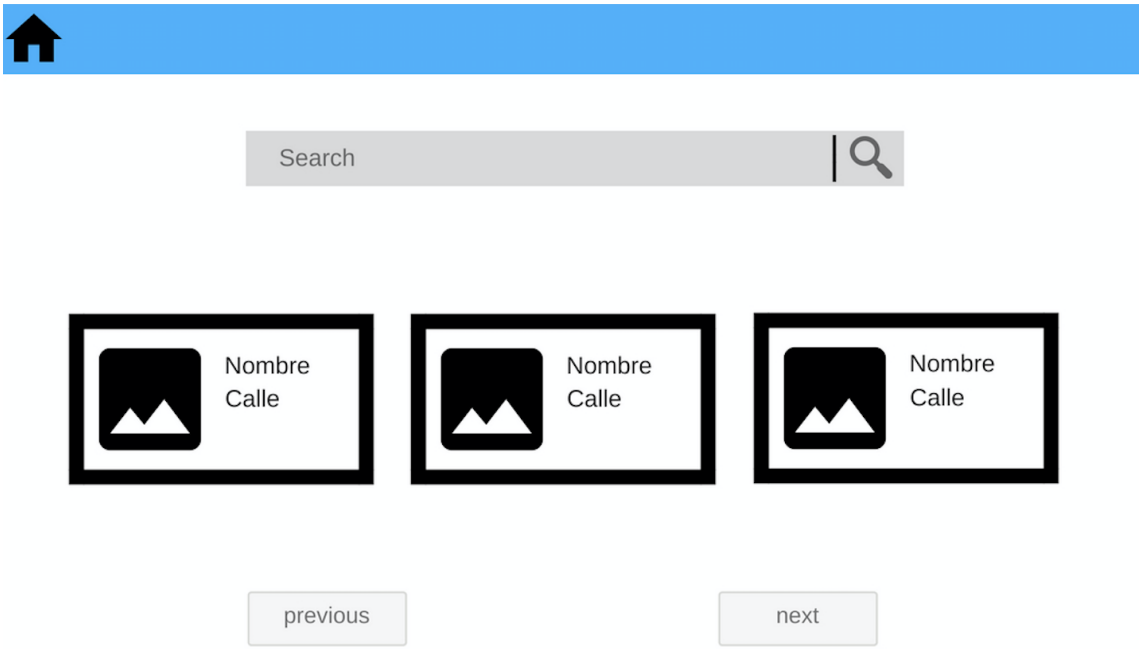

Figura 4.4: Primera vista

La primera pantalla al iniciar sesión que se ve es la figura 4.4. Aquí tendremos acceso a todos los pacientes del sistema y podremos buscarlos mediante el buscador de la parte superior.

Figura 4.5: Pantalla principal de un paciente

Tras la cabecera encontraremos una lista de notificaciones que muestra las últimas 5 notificaciones generadas para ese paciente. Al pulsar el botón de Ver todas, accederemos a otra pantalla en la que tendremos la lista total de notificaciones. Volviendo a la pantalla principal, tras las notificaciones encontraremos la siguiente gráfica.

Como se ve en la figura 4.5, podremos seleccionar el intervalo de tiempo a mostrar, los contratos visibles y el formato de gráfica por horas, días, meses o años.

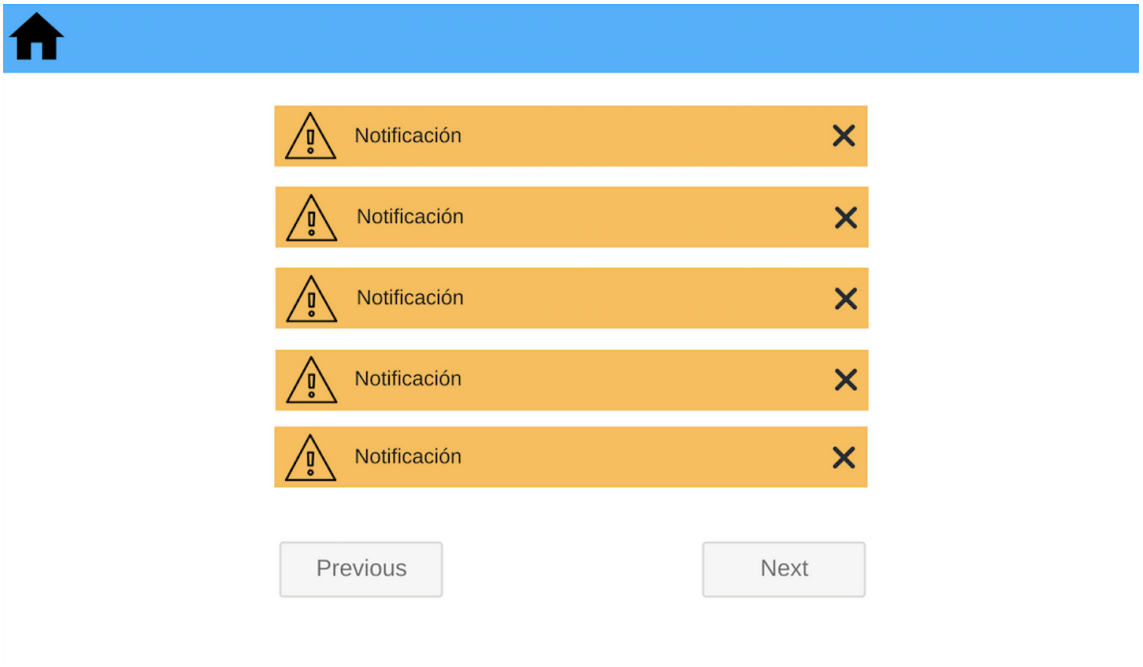

Figura 4.6: Pantalla principal de notificaciones

Esta pantalla nos muestra las notificaciones totales del sistema para un usuario.

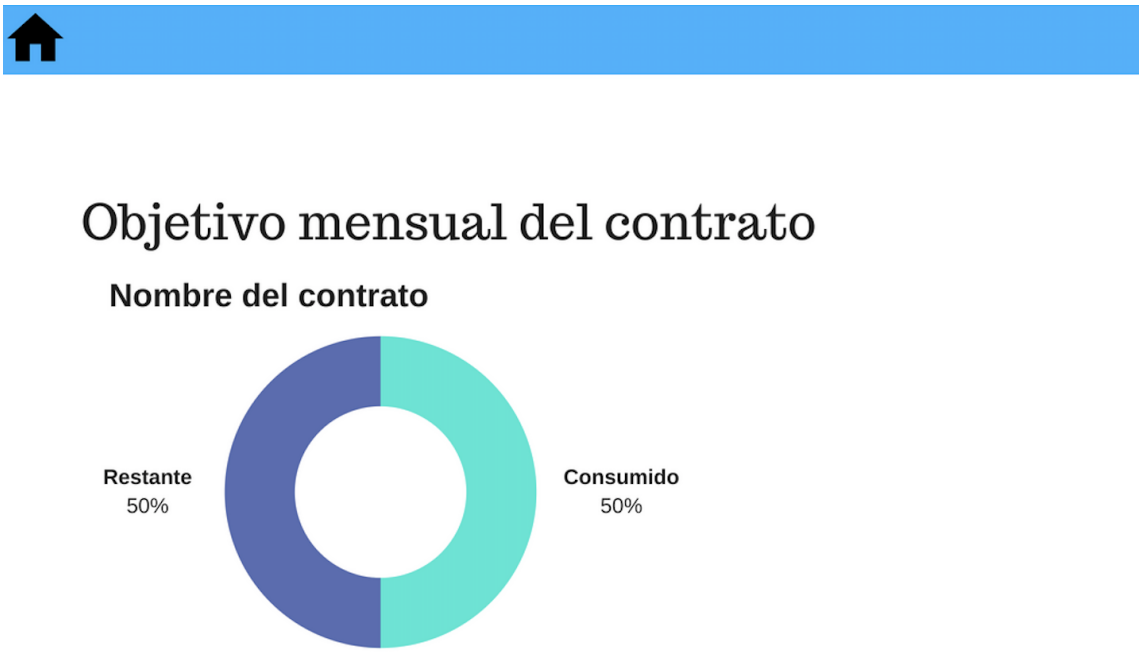

Figura 4.7: Gráficas de objetivo mensual de contrato

Aquí podremos ver lo consumido contra lo restante para saber si superamos el objetivo mensual de consumo establecido previamente.

A su vez, Django nos proporcionará un panel de administración generado automáticamente a partir de los modelos de datos que permitirá controlar los consumos, pacientes y contratos de forma sencilla. Por ello no precisamos un diseño concreto para ello.

4.3 Tecnologías utilizadas

4.3.1. Django

Django es un framework de desarrollo web escrito en Python que mantiene el patrón de diseño Modelo–vista–controlador.

El objetivo de Django es facilitar la creación de sitios web complejos. Django pone énfasis en la reutilización, la conectividad y extensibilidad de componentes y el desarrollo rápido. Se usa Python en todas las partes de este framework, incluso en configuraciones, archivos, y en los modelos de datos.

Un esquema básico de Django sería:

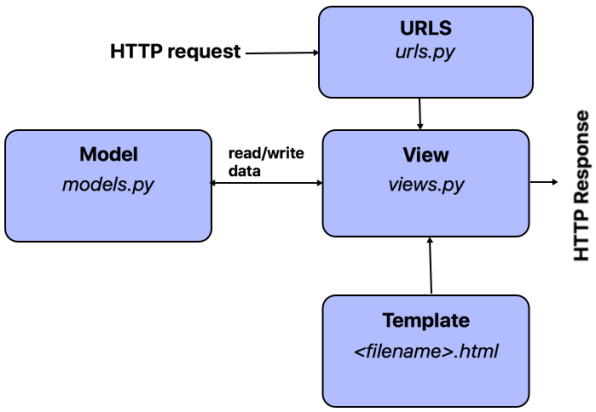

Figura 4.8: Esquema de Django.

Como se puede observar en la figura 4.8, tras recibir una petición HTTP, se pasará por un mapeador de URL que redirigirá la petición a la vista que corresponda. A su vez este mapeador podrá trasladar cadenas y dígitos a la función o clase de la vista.

La vista gestionará las peticiones que le llegan devolviendo respuestas en formato HTTP. Los datos que sean necesarios en estas vistas serán accedidos mediante los modelos. Estos permiten estructurar los datos y tramitar la gestión y consulta a la base de datos, todo ello en python. Por otro lado, las plantillas permitirán definir la estructura de las páginas HTML. Todo esto es posible porque Django utiliza un ORM (Object-Relational mapping) que permite utilizar una base de datos relacional como motor de persistencia desde un lenguaje orientado a objetos.

A su vez, Django también nos facilita el uso de formularios a nivel de creación, validación y procesamiento. Tambíen incluye un sistema de seguridad robusto en cuanto a permisos y autenticación. En cuanto al sistema de caché, nos permite almacenar partes de una página para no renderizar de nuevo más que cuando haya cambios necesarios. Una característica interesante de este framework es que incluye un sitio de administración por defecto para la edición y creación de los modelos. Para ver el funcionamiento de Django, se nos facilita una documentación completa [2].

[2]https://docs.djangoproject.com/en/2.0/

4.3.2. Docker

La función principal de docker es crear contenedores portables para las aplicaciones que puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo. Este sistema permite automatizar el despliegue de aplicaciones sobre contenedores de software. De este modo proporciona una capa de abstracción y automatización a la virtualización a nivel de sistema operativo. Implementa una API de alto nivel que proporciona contenedores que permiten la ejecución de aplicaciones de manera aislada. Haciendo uso de esta herramienta podremos aislar los recursos, restringir los servicios y tener componentes de una aplicación compleja aislados e independientes. Podemos encontrar más información en su documentación [3]. Como podemos ver en la

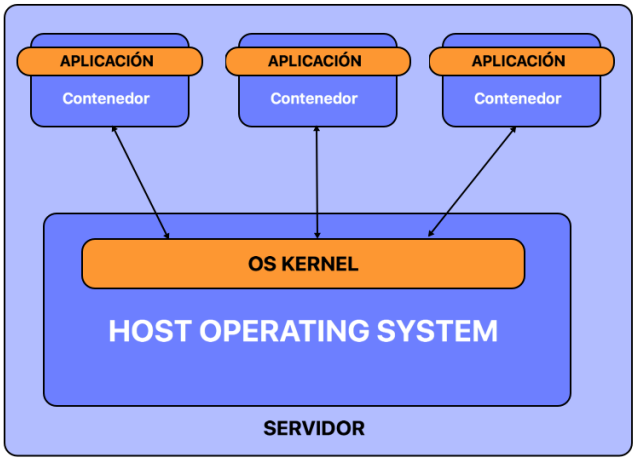

Figura 4.9: Esquema de Docker

figura 4.9, Docker puede mantener diversos contenedores con aplicaciones distintas dentro de un mismo servidor. Docker es una herramienta diseñada así para beneficiar tanto a desarrolladores, testers, como administradores de sistemas.

4.3.3. Selenium

Selenium nos presenta un entorno para realizar pruebas de software para aplicaciones basadas en la web. En concreto hablaremos de Selenium WebDriver, el cual permite enviar comandos directamente al navegador.

Contiene un controlador específico para cada navegador, entablando así las conexiones pertinentes.

La idea principal de esta herramienta es el testeo de aplicaciones de forma automática para diferentes navegadores y simulando diferentes usuarios, aunque en nuestro caso el uso será para llevar a cabo el scrappeo de una web como veremos más adelante. A demás Selenium es capaz de realizar muchas más funciones que escapan del ámbito de este trabajo y que se pueden encontrar en su documentación. [4].

Desde diferentes lenguajes selenium nos permite (usando un protocolo concreto) trabajar con el driver de un navegador y esto permite controlar un navegador de forma remota mediante código, permitiendo así simular a un usuario, como se observa en la

[3] https://docs.docker.com
[4] https://www.seleniumhq.org/docs/

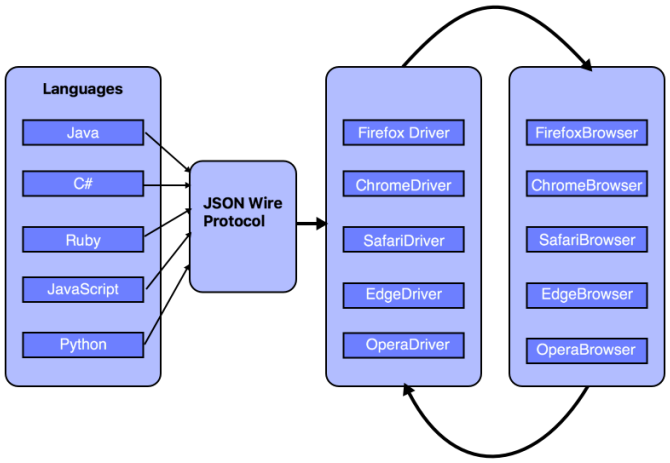

Figura 4.10: Esquema de Selenium

figura 4.10. La ventaja de Selenium frente a otros softwares de petición y procesado de páginas web es que, al utilizar el navegador y su motor de Javascript, podemos acceder a páginas web con mucha carga de Javascript, cosa mucho más complicada en otros softwars como requests, curl, scrappy, etc.

4.3.4. Redis

Es el acrónimo de Remote Dictionary Server. Redis es un almacén de datos en memoria rápida y de código abierto. Incorpora un conjunto de estructuras en memoria que le permiten crear con facilidad diferentes aplicaciones personalizadas. Los usos principales de este almacén suelen ser el almacenamiento en caché, la administración de sesiones, pub/sub y las clasificaciones.

Esta escrito en C y admite diversos lenguajes de desarrollo. Por su gran velocidad y facilidad de uso, es una opción muy usada en aplicaciones web y móviles.

4.3.5. Celery

Celery es una cola de tareas asíncrona basada en el envío de mensajes distribuidos. Se centra en el funcionamiento en tiempo real. Las unidades de ejecución, denominadas tareas, se ejecutan simultáneamente en uno o más servidores de trabajo mediante multiproceso. Las tareas se pueden ejecutar de forma asíncrona o síncrona. De esta forma permite lanzar servicios como tareas.

4.3.6. RabbitMQ

RabbitMQ es un software de negociación de mensajes de código abierto que puede ser considerado un middleware de mensajería. El servidor RabbitMQ está escrito en Erlang y utiliza el framework Open Telecom Platform (OTP) para construir sus capacidades de ejecución distribuida y conmutación ante errores. El servidor de intercambio RabbitMQ genera pasarelas para los protocolos HTTP, XMPP y STOMP.

4.3.7. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional orientado a objetos. Permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente. Esto proporciona una alta concurrencia. Además, da un amplio soporte de tipos nativos, al mismo tiempo que permite a los usuarios crear nuevos tipos de datos.

4.3.8. Keras

Keras es una librería de Python que nos proporciona una manera sencilla de implementar modelos de deep learning sobre las librerías TensorFlow, Theano o CNTK.

Tiene la ventaja de trabajar a más alto nivel que otras librerías, lo cual la hace más intuitiva y sencilla de implementar, obteniendo grandes resultados. A pesar de esto para algunos problemas precisaremos de librerías de más bajo nivel, puesto que no permite la modificación de algunos campos de la red neuronal.

CAPÍTULO 5
Desarrollo

5.1 Introducción

En este capítulo vamos a explicar cada parte de nuestra plataforma final, separando la parte web de la parte de detección de anomalías. De esta forma mostraremos la visualización final del sistema mediante capturas de la web para mostrar el desarrollo de la aplicación. Para la detección de anomalías mostraremos el modelo final obtenido, el cual se ha llevado a producción para la detección de anomalías, explicando sus valores y variables, aunque se complementará en el capítulo de validación.

5.2 Desarrollo web

En el apartado 4.2.2 mostramos como habíamos diseñado inicialmente nuestra página web para tener todas las funcionalidades deseadas. Estos diseños han sido llevados a desarrollo para crear la interfaz necesaria para que el administrador pueda trabajar con nuestra plataforma de la forma que habíamos determinado en primer lugar. Por esto mostraremos a continuación algunas capturas de la web que nos permita ver el resultado obtenido.

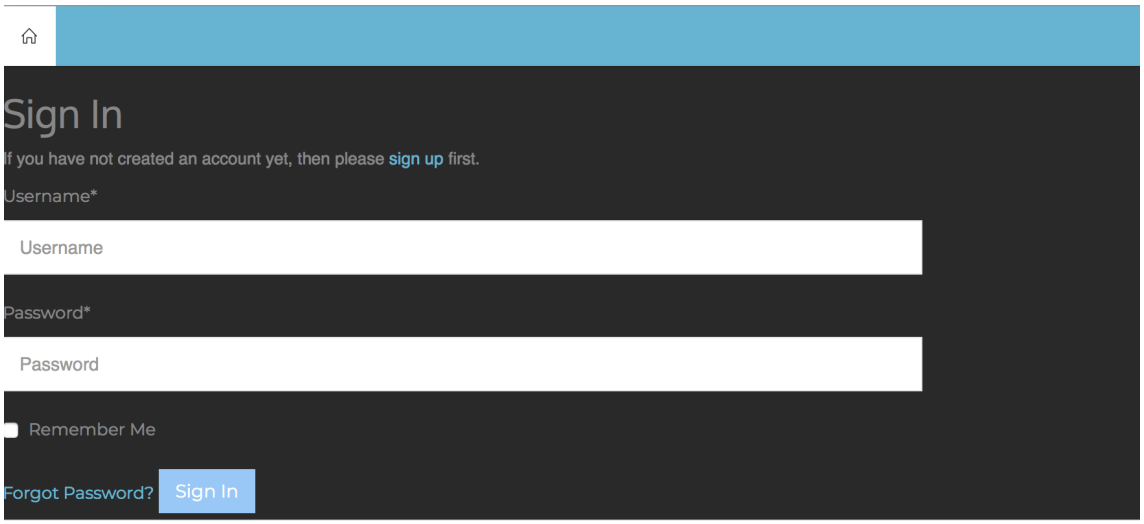

Figura 5.1: Página de inicio de sesión

Como podemos observar, en esta pantalla no hay mucha diferencia con la idea inicial que preparamos en la etapa de diseño.

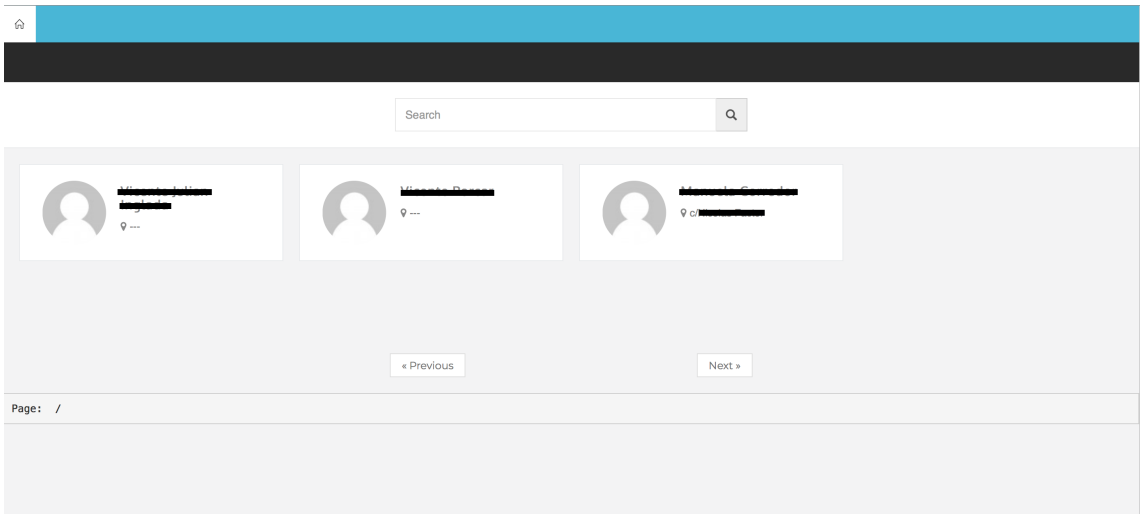

Figura 5.2: Pantalla inicial de pacientes

En este caso, al igual que el anterior, la visualización ha sido bastante similar a lo esperado en la fase de diseño. Hemos tenido que tapar los nombres y direcciones que habían en la web por protección de datos.

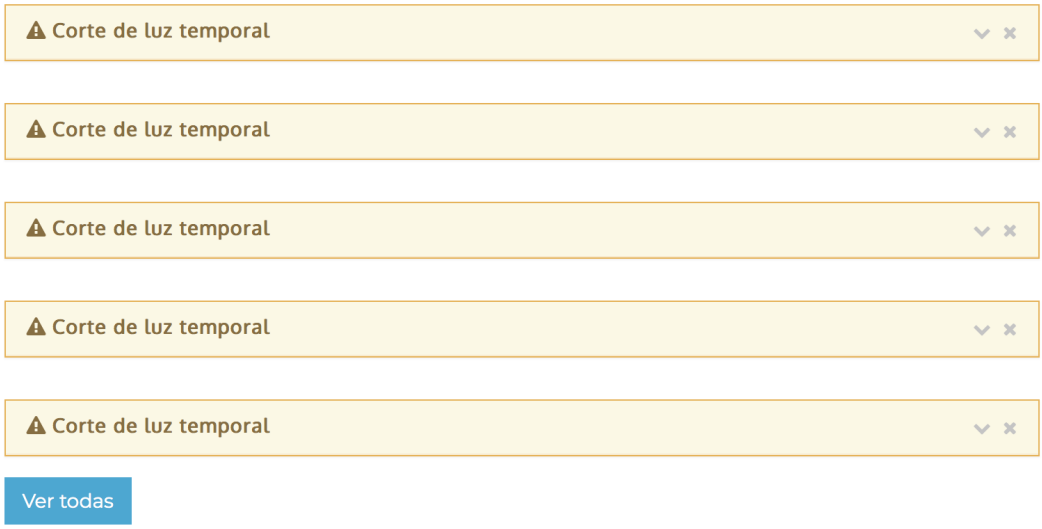

Figura 5.3: Listado de notificaciones inicial

Este listado se muestra al entrar a un paciente seleccionado como se estableció en la fase de diseño. Al pulsar el botón **"Ver todas"** pasaremos a la siguiente pantalla.

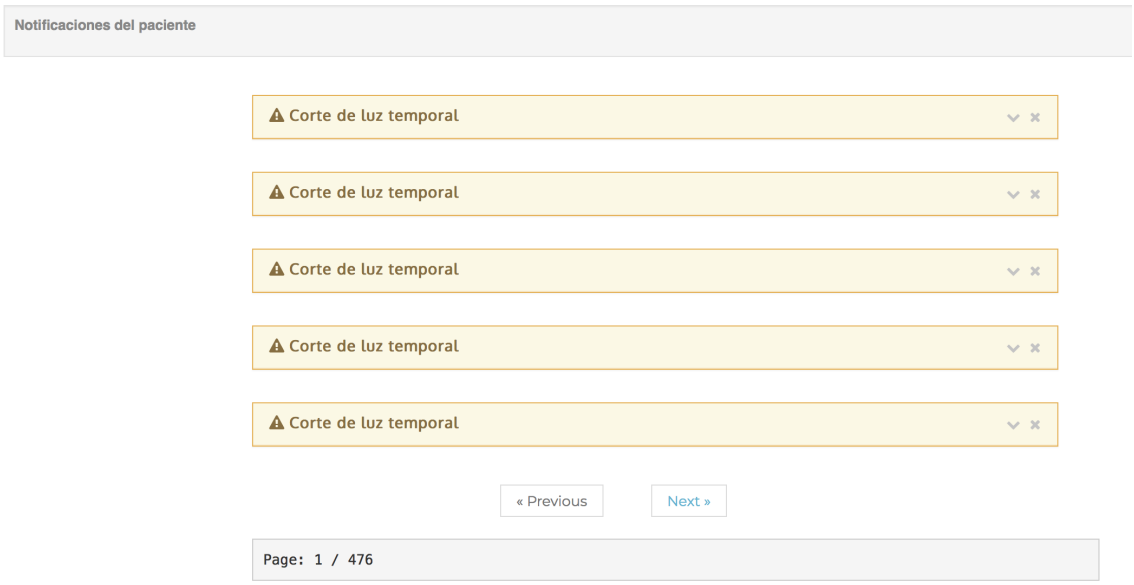

Figura 5.4: Listado de notificaciones total

Aquí podemos recorrer el total de notificaciones que tenga cada paciente, eliminarlas y ver más información sobre ellas.

Volviendo a la pantalla principal de un paciente podremos ver diferentes gráficas.

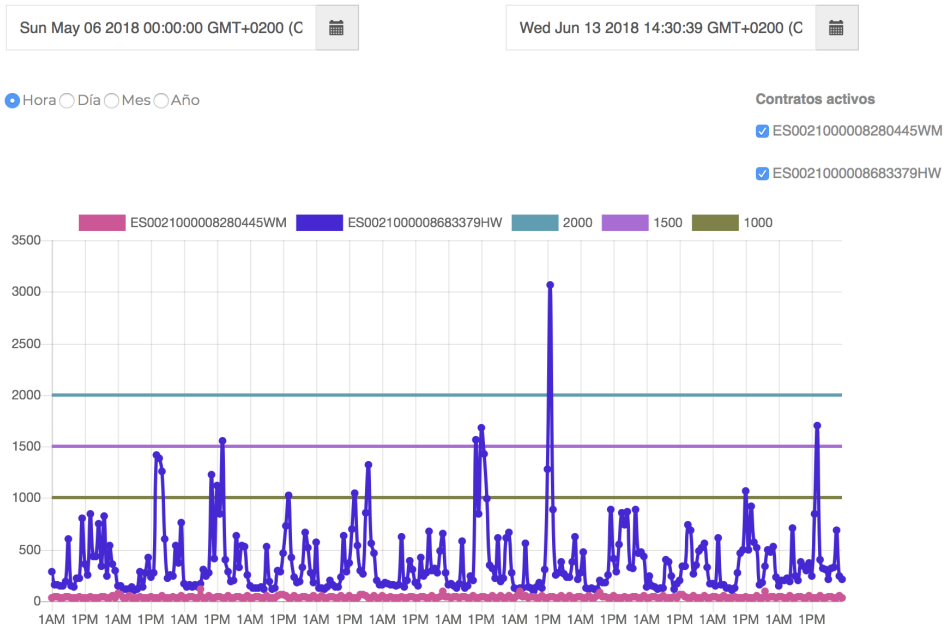

Figura 5.5: Gráfica principal

Hemos añadido diferentes funciones que permitirán visualizar los datos de consumo en horas, días, meses o años. Además podremos seleccionar que contratos de ese paciente queremos visualizar y el intervalo de tiempo que queremos mostrar.

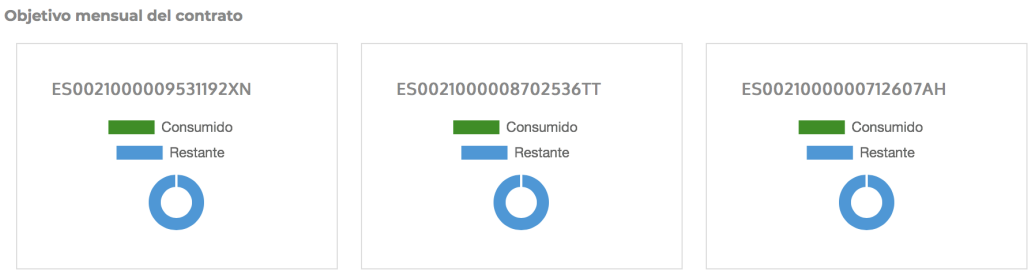

Figura 5.6: Gráficas de objetivo mensual de consumo

En cuando al sistema de administración, como dijimos en la fase de diseño, Django nos proporciona un sistema fácilmente editable que hará de portal de administración para toda la edición, creación y eliminación de pacientes, contratos, consumos, tareas y notificaciones.

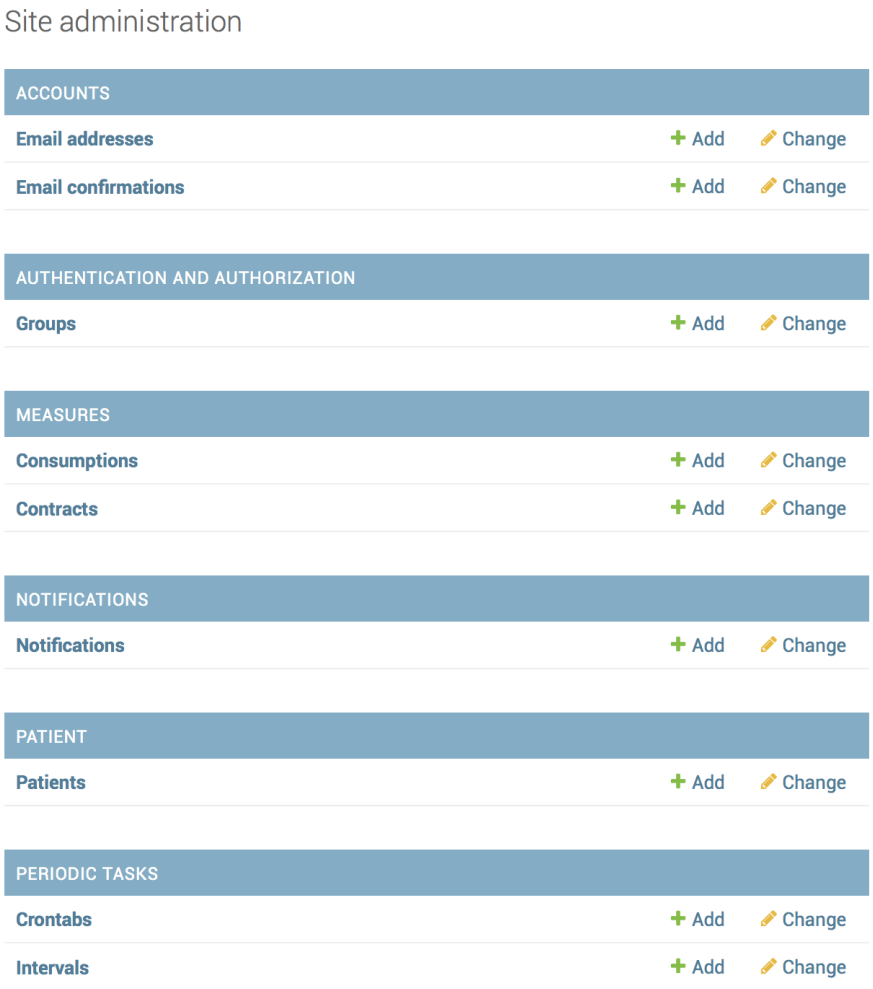

Figura 5.7: Pantalla inicio administración

5.3 Detección de anomalías

Como hemos comentado anteriormente, lo que hemos utilizado finalmente para la detección de anomalías son diferentes modelos de clasificadores en Keras. Tendremos 3 modelos distintos, cada uno de ellos clasificará como normal o anómalo. De este modo lograremos clasificar en 3 tipos distintos de anomalías.

Para comenzar vamos a mostrar las distintas anomalías que buscamos.

5.3.1. Consumos anormalmente altos

En primer lugar, deseamos detectar aquellos consumos cuya unidad tipificada sea mayor o igual a 2 y que ocurran de forma repetida un mínimo de 3 veces. Esto nos servirá para detectar si durante 3 o más horas seguidas ha habido un consumo que exceda lo normal dentro del hogar, de modo que podremos saber si la persona ha podido dejarse encendido algún electrodoméstico más tiempo del deseado o si ha habido algún problema en el consumo de su hogar. Esto tenemos en cuenta que podrá generar algunos falsos positivos debido a que el factor humano es impredecible al cien por cien, pero trataremos de notificar siempre de forma que sea un operario administrador el que decida que hacer finalmente con dicha notificación.

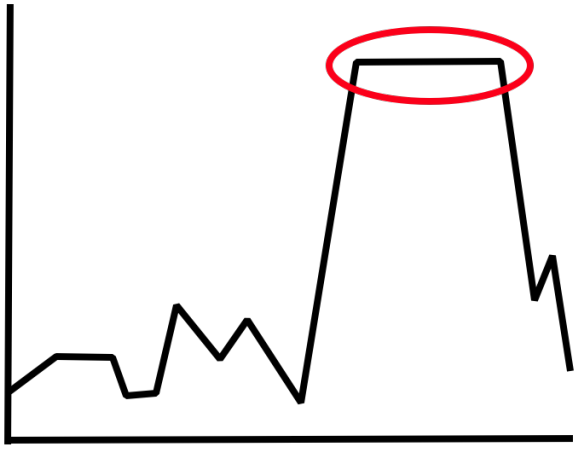

Figura 5.8: Ejemplo consumo anormalmente alto

En la figura 5.8 se puede ver un ejemplo de este tipo de anomalía de forma simplificada. Como se observa, pasamos de un consumo normal a un consumo excesivo durante un periodo largo de tiempo. Si este exceso fuese puntual no sería una anomalía pues podría deberse a diversos factores del hogar normal.

5.3.2. Corte de luz detectado

Por otro lado, tenemos la anomalía más sencilla en principio, cuando el consumo del hogar sea 0. Si durante dos horas el consumo del hogar es 0 procederemos a notificar dado que puede significar una caída de la luz cuando esa persona no estaba en casa, lo cual puede afectar a ciertos electrodomésticos, o están en casa, lo cual significaría que esa persona o no sabe, o no puede activar el sistema eléctrico de nuevo.

Figura 5.9: Ejemplo de corte de luz

En este caso, en la figura 5.9 vemos un ejemplo de un corte de luz. Como en el caso anterior, en este también precisaremos de un tiempo determinado para considerarlo anomalía.

5.3.3. Consumos anormalmente bajos

Por último, la anomalía más compleja será detectar un consumo eléctrico bajo, pero no cero, en horas en las que no debería haber dicho consumo. Esto significaría que la persona en cuestión no ha hecho el consumo normal durante al menos 4 horas seguidas, de modo que podría no haberse levantado o haber sufrido algún tipo de accidente.

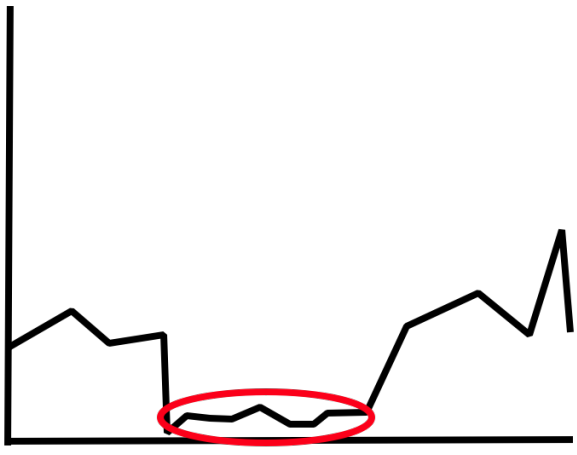

Figura 5.10: Ejemplo consumo anormalmente bajo

Este último caso lo representamos en la figura 5.10. Este es distinto del consumo 0 dado que tendrá un consumo mínimo y será motivo de alerta cuando suceda durante el día, no durante la noche.

Al final, los valores temporales que definimos aquí son orientativos, y deberá ser quien haga uso de esta herramienta y tenga los conocimientos suficientes del ámbito

asistencial el que establezca las horas mínimas de espera para lanzar la notificación y ponerse en contacto con la persona o familiar designado.

5.3.4. Conjunto de datos

Para lograr generar los clasificadores necesitábamos tener un conjunto de datos clasificado. Como no teníamos dicho conjunto, decidimos programar tres generadores de anomalías que tomarían como entrada el conjunto de datos inicial y sobre este añadirían anomalías clasificándolas de forma correcta. Obteniendo finalmente el conjunto de datos con anomalías y sin ellas y una lista con la clasificación de cada dato del conjunto.

El conjunto de datos del cual dispondremos será un set de 9096 elementos. Para representar cada dato lo trataremos junto con los 4 datos anteriores temporalmente. A su vez, irá acompañado de su clasificación, la cual habrá sido definida siguiendo el proceso definido anteriormente.

5.3.5. Modelos

Una vez obtenidos los datos clasificados planteamos tres modelos distintos tras diferentes pruebas que clasificarían de la forma más correcta que nos fuese posible. El uso en nuestros modelos de capas LSTM se debe a la necesidad de aprendizaje a largo plazo. Este tipo de red neuronal fue diseñada para evitar el problema de la dependencia a largo plazo, que consiste en que si se tiene una red más grande a través del tiempo, el gradiente se desvanece rápidamente durante la fase de back-propagation. Las LSTMs lo evitan permitiendo hacer un camino (estados célula) a través del tiempo, estos caminos permiten que el gradiente fluya libremente hacia atrás en el tiempo. En nuestro caso, necesitaremos plantear tres modelos distintos, uno para cada clase de anomalía, debido a que tras diferentes pruebas vimos que para cada una de estas anomalías funcionaba mejor un tipo diferente de red neuronal. En todos los modelos tendremos como datos de entrada un conjunto de los 5 últimos consumos, contando dentro de este conjunto el consumo que vamos a clasificar.

Modelo 1 - Consumos elevados

Estará compuesto por dos capas LSTM y una capa densa. Se utilizará la activación softmax y categorical crossentropy. Este dará como resultado dos valores que serán correspondientes a la clase normal o a la clase anómala. La dimensión del espacio de salida de la primera capa será 256, la segunda capa 132 y la última, la capa densa, tendrá 2, para dar así las dos clases deseadas, normal o anómala. En las capas LSTM aplicaremos un dropout de 0.3 que tiene como objetivo reducir la complejidad del modelo con el objetivo de evitar el sobreajuste. El número de iteraciones sobre toda la información del conjunto de datos en este modelo será de 500 epochs.

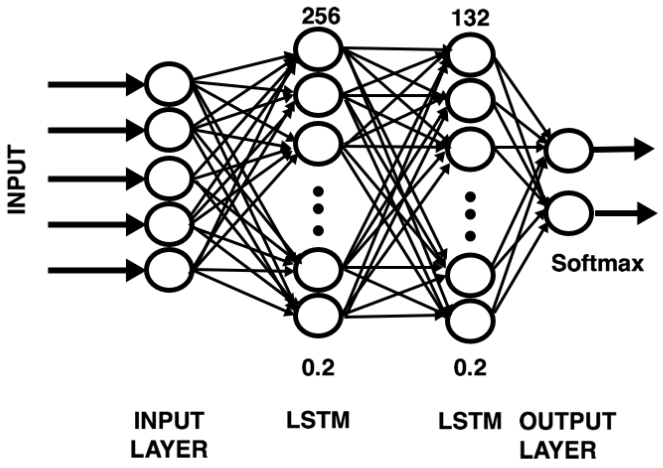

Figura 5.11: Diagrama modelo 1

Modelo 2 - Consumos a cero

Estará compuesto por dos capas densas . Se utilizará la activación softmax y categorical crossentropy. Este dará como resultado dos valores que serán correspondientes a la clase normal o a la clase anómala al igual que el anterior. La dimensión del espacio de salida de la primera capa será 256 y la última tendrá 2. El número de iteraciones sobre toda la información del conjunto de datos en este modelo será de 100 epochs.

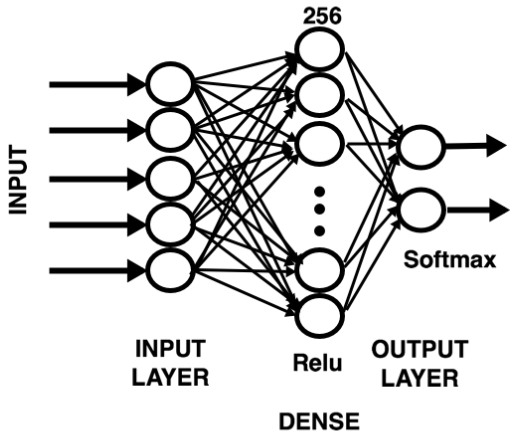

Figura 5.12: Diagrama modelo 2

En este caso el uso de una red neuronal no es estrictamente necesario. Podríamos haber usado otros métodos para lograr un resultado similar, pero con redes neuronales

podíamos utilizar modelos muy similares para todos las anomalías y en este caso precisaba muy poco tiempo de desarrollo ya que nos basábamos en lo ya implementado.

Modelo 3 - Consumos bajos

Estará compuesto por dos capas densas. Se utilizará la activación softmax y categorical crossentropy. Este dará como resultado dos valores que serán correspondientes a la clase normal o a la clase anómala. La dimensión del espacio de salida de la primera capa será 256 y la última tendrá 2. El número de iteraciones sobre toda la información del conjunto de datos en este modelo será de 1000 epochs.

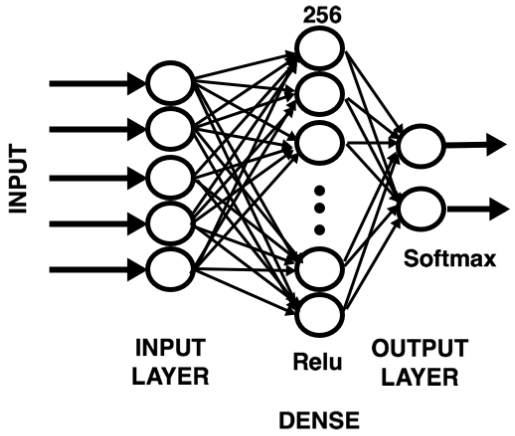

Figura 5.13: Diagrama modelo 3

El número de muestras que se propagarán a través de cada modelo será de 525 para todos los modelos. A su vez, en todos los casos usaremos el 80 % de los datos para entrenar y el 20 % para probar. Todos estos valores han sido seleccionados mediante experimentación, quedándonos con los que se adecuaban mejor a nuestros datos.

CAPÍTULO 6
Implantación

En este capítulo vamos a explicar la fase de despliegue de la aplicación. Esta fase es muy importante puesto que permite mostrarla al usuario definitivo y evaluar si el producto final es el deseado o no.

Este proyecto se ha llevado a cabo junto con el grupo de investigación del DSIC GTI-IA. Fundaciones como Cruz roja o Vodafone han presentado interés en este sistema y es por eso que el despliegue será fundamental para que este interés aumente y puedan ver el resultado final en funcionamiento.

El grupo de investigación dispuso de un servidor sobre el que desplegaremos esta aplicación. Como vimos en el capítulo 4, deberemos instalar diferentes herramientas en el servidor para poder desplegar toda la arquitectura.

1. Tendremos Docker para mantener todos los componentes de la aplicación que se mantendra desplegado en el servidor en cuestión.
2. Hará falta tener Django como base la arquitectura.
3. Utilizaremos Redis para gestionar la cache desde Django.
4. PostgreSQL se encargará de la gestión de los datos del servidor.
5. Gunicorn lo tendremos que implantar para utilizarlo como puerta de enlace del servidor web entre Django y Nginx.
6. Nginx lo usaremos para recibir las peticiones HTTP y gestionar la redirección de peticiones entre estáticos y no estáticos.
7. Celery worker y Celery beat para crear y gestionar distintas tareas en segundo plano, como la tarea de scrappeo.
8. Selenium chrome como navegador sin interfaz para llevar a cabo tarea de scrappeo.

Otra parte importante de la aplicación se encuentra en la implantación en la industria. La institución Cruz Roja y la fundación Vodafone han mostrado interés en este proyecto para ser utilizado en los hogares más desfavorecidos. Esto podría hacer que este trabajo pudiese ser transferido de forma directa en la sociedad, adaptándose a las necesidades de cada grupo para solucionar los diferentes problemas que se les plantean.

CAPÍTULO 7
Validación

7.1 Introducción

En este capítulo vamos a mostrar las pruebas llevadas a cabo sobre los diferentes sistemas de predicción y clasificación de datos destinados a la detección de anomalías que hemos trabajado en este proyecto.

7.2 Luminol

Esta librería esta destinada para la clasificación, aunque lo hace mediante la predicción de datos y evaluación. En nuestro caso quisimos utilizar solo la parte predictiva de esta librería.

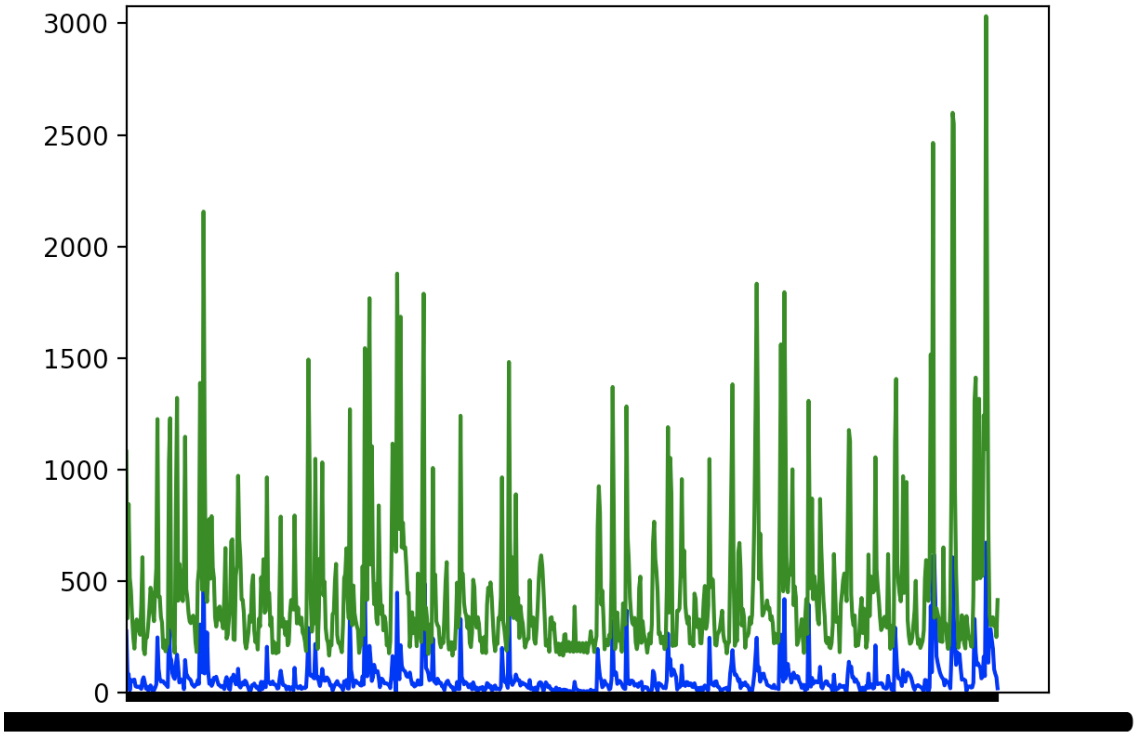

Figura 7.1: Gráfica predicción Luminol

En la gráfica de los resultados vemos que se asemeja la predicción, en azul, al caso real, en verde, pero solo en su forma. Los valores que alcanza son excesivamente bajos y no nos permite trabajar con ellos de forma adecuada y es por ello que descartamos utilizar esta utilidad.

7.3 Prophet

La librería Prophet nos proporciona un sistema de predicción que pensamos que podríamos probar al comenzar este proyecto. Tiene una programación sencilla y muy guiada e incluye también formas de trabajar con series temporales y estacionalidad.

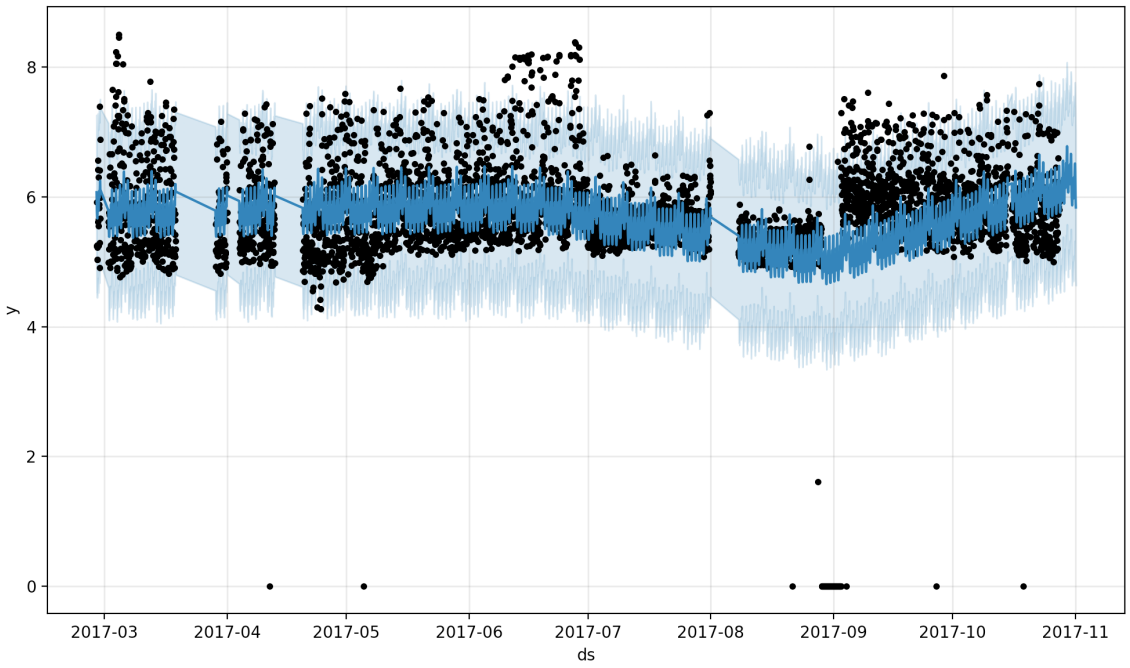

Figura 7.2: Gráfica predicción Prophet

Esta librería proporciona en su predicción incluso un rango de predicción como se ve en la figura 7.2. Los huecos que aparecen en la gráfica se deben a interrupciones en nuestra lectura de datos de forma automática, aunque esto no debería afectar al sistema. Los resultados obtenidos han sido muy inadecuados para nuestros datos. Con datos menos inconsistentes podría dar mejores resultados. En nuestro caso tuvimos que descartarla.

7.4 Redes neuronales

Tras las pruebas anteriores nos decantamos por el uso de redes neuronales para la detección de anomalías, tratando en primer lugar de hacer un predictor que nos permitiría valorar si el dato era o no anómalo y tras eso un clasificador que nos diera directamente los resultados de la clasificación.

7.4.1. Predictor

Una vez comenzamos con las redes neuronales planteamos la idea de hacer un modelo de predictor que nos permitiera, sabiendo el dato futuro, poder decidir si un dato era anómalo o no.

De esta forma hemos diseñado un modelo basado en redes neuronales recursivas, en concreto LSTM. Dado que trabajamos con series temporales este tipo de red era adecuado usarlo pues necesitaremos aprender a largo plazo.

Planteamos un modelo compuesto por tres capas LSTM y una capa densa. Para la capa de salida se utilizó la activación lineal, puesto que es la que más se adecua para un problema de regresión. La función objetivo utilizada, o función de puntuación de la optimización, es la función de error cuadrático medio que funciona de forma adecuada con la activación lineal. Este dará como resultado el valor predicho de hora siguiente a la evaluada. La dimensión del espacio de salida de la primera capa será 100, la segunda capa 50,la tercera tendrá 25 y la última 1. En las capas LSTM aplicaremos un dropout de 0.2 que tiene como objetivo reducir la complejidad del modelo con el objetivo de evitar el sobreajuste. El número de iteraciones sobre toda la información del conjunto de datos en este modelo será de 50 epochs.

Del conjunto total de datos usaremos el 90 % para entrenamiento y el 10 % para test.

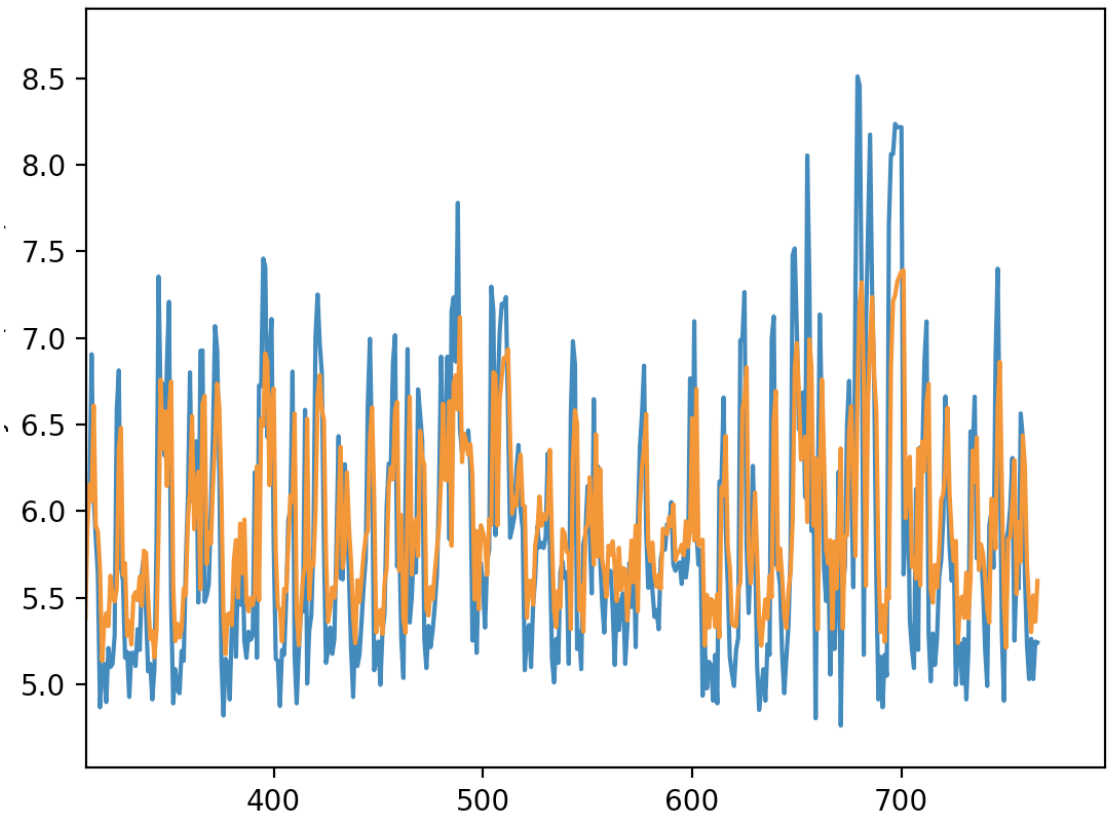

Figura 7.3: Gráfica predictor LSTM

Como vemos en la figura 7.3, el resultado de la predicción en naranja se asemeja al del dato real en azul. A pesar de esto no nos sirve para nuestro problema, puesto que predice de forma correcta las circunstancias que son anómalas, de forma que no podríamos sacar

información significativa. Esto se debe a que el sistema se basa de forma excesiva en el histórico cercano, de modo que va tomando la pendiente en función de este. Es por esto que descartamos esta solución

7.4.2. Clasificadores

Clasificador multiclase

En un primer lugar planteamos un clasificador multiclase que tomase la decisión entre 4 clases. Este modelo estará compuesto por dos capas LSTM y una capa densa. En la capa de salida se utilizó la función de activación softmax que es muy usada para clasificadores y como función objetivo se usó categorical crossentropy por ser clasificación multiclase. Este dará como resultado dos valores que serán correspondientes a la clase normal o a la clase anómala. La dimensión del espacio de salida de la primera capa será 256, la segunda capa 132 y la última, tendrá 4, para dar así las cuatro clases deseadas: normal o 3 tipos diferentes de anomalías. En las capas LSTM aplicaremos un dropout de 0.2 que tiene como objetivo reducir la complejidad del modelo con el objetivo de evitar el sobreajuste. El número de iteraciones sobre toda la información del conjunto de datos en este modelo será de 500 epochs.

Tras diseñarlo y lanzar las pruebas obtendremos los siguientes resultados de la representación gráfica de la sensibilidad frente a la especificidad para un sistema clasificador (Curva ROC), es decir, que presenta el ratio entre verdaderos positivos y falsos positivos. En la representación de esta gráfica tendremos por un lado las lineas que representan cada clase con su ratio de verdaderos positivos frente a falsos positivos, por otro lado habrá una recta que representa el límite del 50 %, se llama recta de no discriminación. Cada curva nos dará un porcentaje obtenido mediante el área por debajo de la curva de cada una de ellas siendo mejor cuanto más grande sea este área. Habrá dos curvas más que representarán micro-average y macro-average. En el primer caso calcula las métricas globalmente contando el número total de veces que cada clase se predijo correctamente e incorrectamente. En el segundo caso calcula las métricas de cada clase de forma independiente y calcula su media no ponderada.

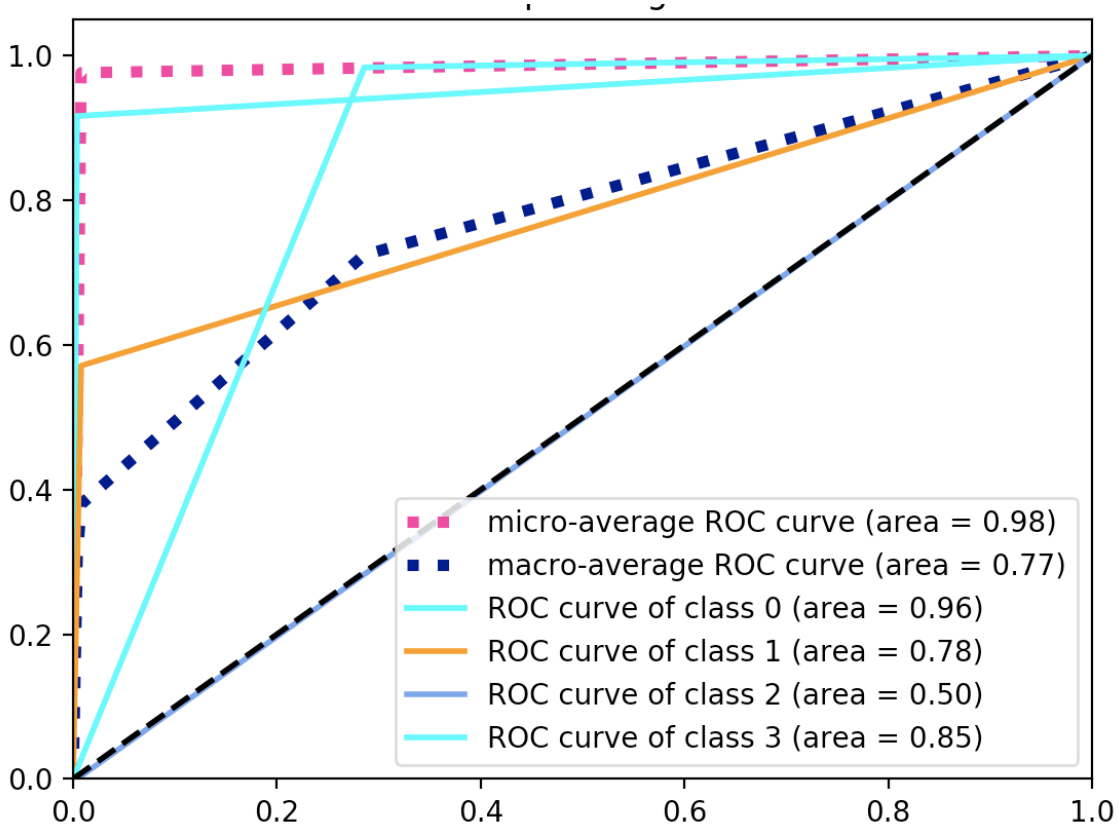

Figura 7.4: Curva ROC clasificador multiclase

Como observamos en la figura 7.4, obtenemos resultados aceptables para 3 de las 4 clases. Esto nos lleva a reajustar algunos parámetros, pero en todos los cambios hechos a dicho modelo se mantiene como hasta ahora o empeora.

Viendo estos resultados decidimos hacer 3 clasificadores distintos que trabajaran con la clase normal y una de las clases anómalas.

Clasificador de consumos anormalmente altos

En este caso, como comentamos en el capítulo de desarrollo, tendremos un modelo que estará compuesto por dos capas LSTM y una capa densa. Se utilizará la activación softmax y categorical crossentropy. La dimensión del espacio de salida de la primera capa será 256, la segunda capa 132 y la última 2. En las capas LSTM aplicaremos un dropout de 0.3. El número de iteraciones sobre toda la información del conjunto de datos en este modelo será de 500 epochs.

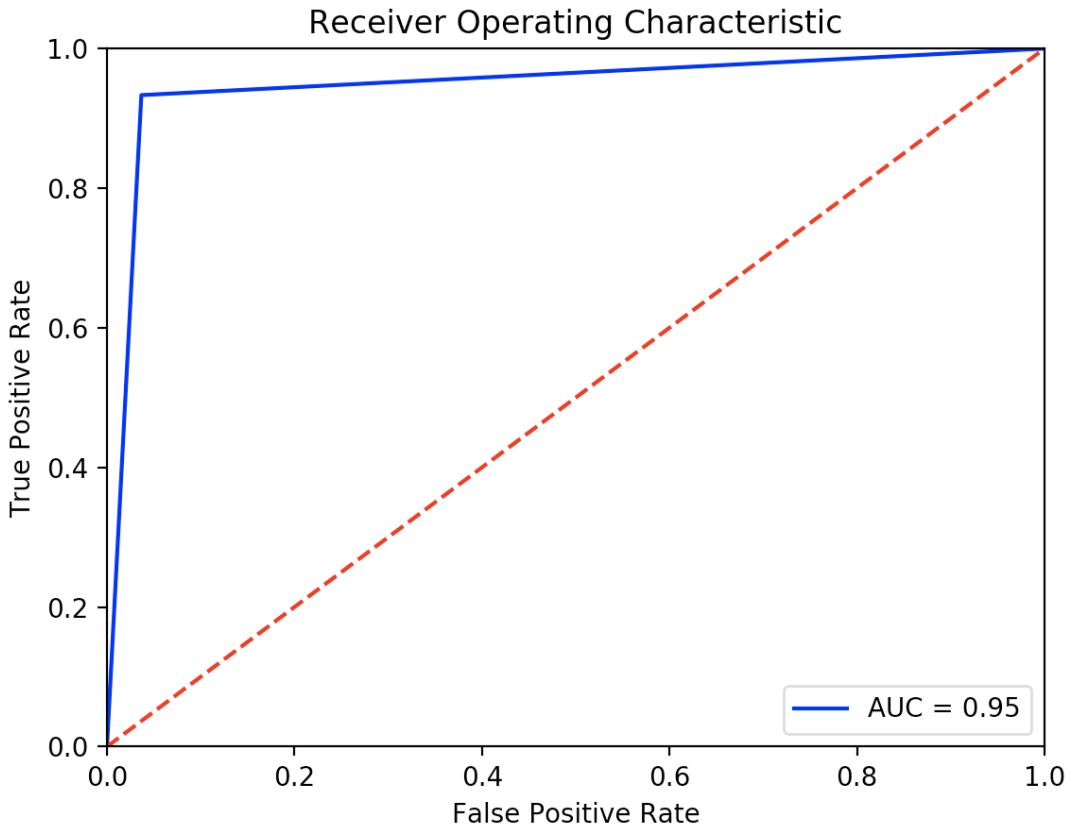

Figura 7.5: Curva ROC consumos anormalmente altos

En este caso, el resultado para esta clase se adecua a lo deseado como se observa gráficamente, dando un 95 % de acierto.

True negative: 1.708	False negative: 65
False positive: 3	True positive: 42

Tabla 7.1: Matriz de confusión consumos anormalmente altos

Se puede observar que lo datos no están balanceados, esto se debe a que dichos datos pertenecen a una serie temporal, por lo que existen muchos más datos normales que anómalos.

La tabla anterior nos dice que de un total de 1773 datos cuya clasificación real es normal, ha habido 1708 que el sistema ha clasificado como normales y 65 que ha clasificado erróneamente como anómalos. Por otro lado, del total de datos anómalos reales que es 45, 3 han sido clasificados erróneamente como normales y 42 han sido clasificado de forma correcta como anómalos.

Clasificador de consumos anormalmente bajos

Este caso es el más complejo de analizar puesto que al evaluar consumos bajos, estos consumos pueden darse de forma normal sin ser realmente una anomalía, como puede ser por la noche, pero debemos encontrar aquellos que realmente si sean problemáticos. El modelo estará compuesto por dos capas densas. Se utilizará la activación softmax y ca-

tegorical crossentropy. Este dará como resultado dos valores que serán correspondientes a la clase normal o a la clase anómala. La dimensión del espacio de salida de la primera capa será 256 y la última 2. El número de iteraciones sobre toda la información del conjunto de datos en este modelo será de 1000 epochs.

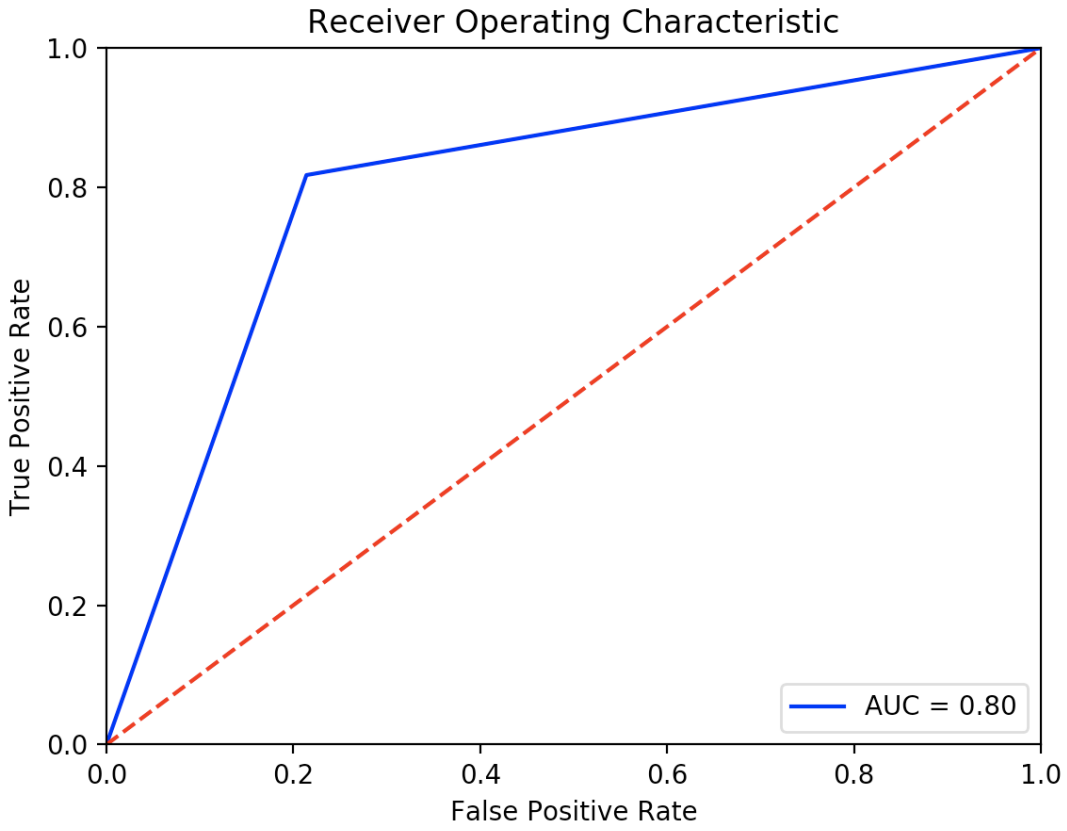

Figura 7.6: Curva ROC consumos anormalmente bajos

True negative: 1425	False negative: 206
False positive: 69	True positive: 118

Tabla 7.2: Matriz de confusión consumos anormalmente bajos

En este caso, a pesar de tener un 80 % de acierto según la curva ROC, como vemos en la matriz de confusión, el resultado es algo menos preciso, aunque admisible. Del total de datos considerados normales, 1425 han sido clasificados de forma correcta como normales y 206 de forma errónea. A su vez, del total de datos anómalos, 69 han sido mal clasificados, mientras que 118 se han clasificado bien.

Clasificador de cortes de luz

Por último, este modelo estará compuesto por dos capas densas. Se utilizará la activación softmax y categorical crossentropy. La dimensión del espacio de salida de la primera capa será 256 y la última será 2. El número de iteraciones sobre toda la información del conjunto de datos en este modelo será de 100 epochs.

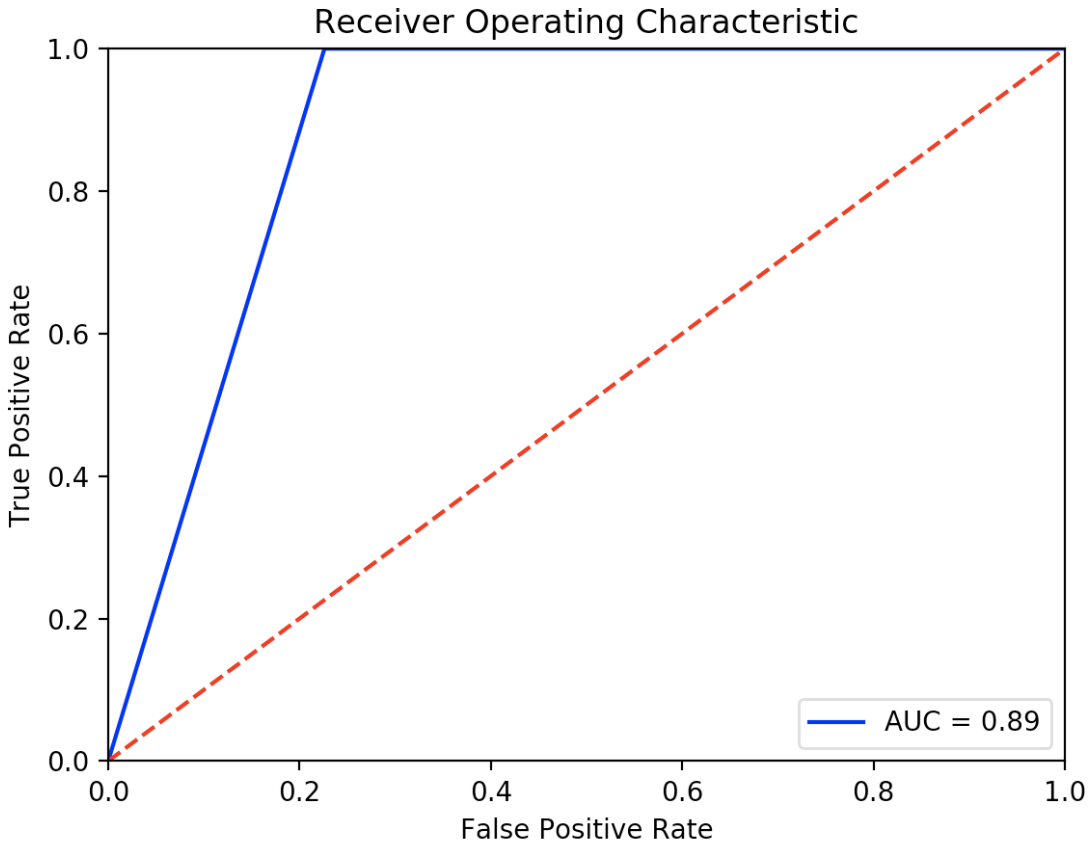

Figura 7.7: Curva ROC cortes de luz

True negative: 1263	False negative: 259
False positive: 14	True positive: 182

Tabla 7.3: Matriz de confusión cortes de luz

Este último modelo da un mejor resultado que el caso anterior, pero como se ve en la matriz de confusión presenta un numero de errores elevado en la clase normal.Esto creemos que se debe a los casos en los que aparecen cortes de luz durante menos de tres horas seguidas, la primera hora no se clasificará como anómala, pero la segunda si debería serlo y el sistema podría llegar a clasificar la segunda hora aún como normal, comenzando a clasificar correctamente a partir de la tercera. Tenemos que del total de datos normales, el sistema ha clasificado 1263 de forma correcta y 259 de forma errónea. En los datos anómalos, el sistema ha clasificado 14 de forma incorrecta y 282 de forma adecuada. A pesar de fallar en la clase normal más de lo debido, presenta muy buenos resultados para la clase anómala.

Para este caso hicimos pruebas haciendo uso de un modelo similar al que usamos para los consumos anormalmente altos con dos capas LSTM y daba unos resultados muy similares a usando únicamente capas densas.

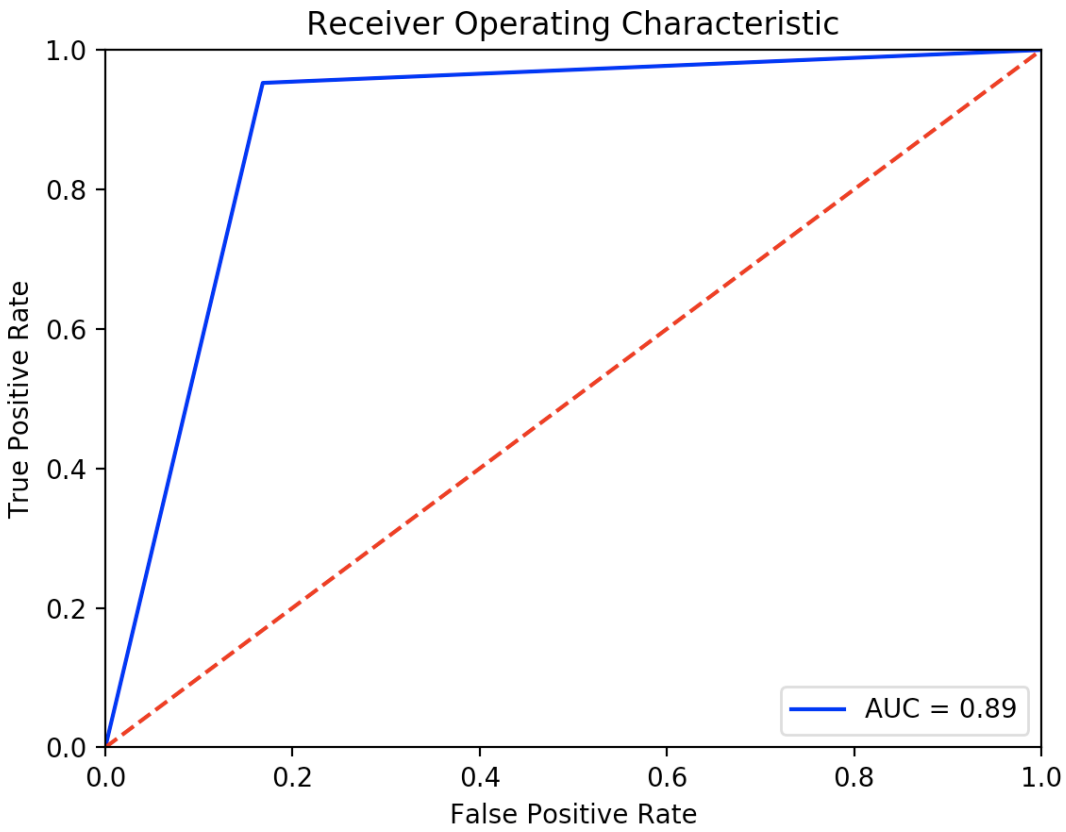

Figura 7.8: Curva ROC cortes de luz LSTM

True negative: 1266	False negative: 256
False positive: 14	True positive: 282

Tabla 7.4: Matriz de confusión cortes de luz LSTM

Viendo resultados tan similares nos decidimos por el de capas densas puesto que era mucho más rápido de entrenar.

CAPÍTULO 8
Conclusión y trabajo futuro

8.1 Conclusiones

En cuanto a los objetivos planteados en este proyecto podemos afirmar que se han alcanzado. Se ha creado una aplicación funcional y accesible que soluciona la problemática planteada inicialmente: dar asistencia a las personas más vulnerables desarrollando una aplicación con la intención de detectar anomalías en el consumo eléctrico del hogar y notificarlas en el ámbito asistencial. La plataforma permite visualizar las notificaciones e información de consumo de las personas inscritas a este sistema de forma que un administrador podrá mantener un control y monitorizar el bienestar de estas personas a través de nuestra plataforma.

De este modo, para conseguir superar el primer objetivo, se ha desarrollado la web que contendrá toda la información de usuario y sus notificaciones, además de un panel de administración completo. Por otro lado, el segundo objetivo se ha logrado obteniendo los datos de consumo eléctrico de cada paciente de forma automática haciendo uso de selenium para scrappear la web de una distribuidora eléctria, en nuestro caso Iberdrola. En cuanto al tercer objetivo, se han planteado diferentes modelos y soluciones para la detección de anomalías. Esto nos lleva al cuarto objetivo, pues se ha llevado a cabo un estudio de herramientas y modelos que se adaptaran mejor a nuestros datos para la detección de anomalías, desarrollando los modelos finales y evaluándolos. Por último, se ha implantado la búsqueda de anomalías en la plataforma de modo que queda funcional y en producción con lo que conseguimos cumplir los objetivos propuestos al principio de este proyecto.

8.2 Relación del trabajo desarrollado con los estudios cursados

La base de este proyecto ha sido la detección de anomalías, lo cual ha enriquecido notablemente este trabajo y lo ha convertido en una gran formación personal. He podido trabajar junto a profesores e investigadores de los cuales he podido aprender mucho tanto a nivel de programación web, como lo relacionado con el aprendizaje automático a nivel de predicción y clasificación de datos. Durante la carrera me he formado en muchas materias distintas, algunas de ellas he podido aplicarlas a este proyecto, como puede ser el aprendizaje de JavaScript y Python, o ciertas nociones de aprendizaje automático, aunque sin llevarlo tanto al terreno práctico. En definitiva, con este proyecto he podido ampliar esos conocimientos y aprender otros muchos.

8.3 Trabajo futuro

Aún habiendo cumplido con los objetivos marcados, este proyecto puede seguir avanzando y mejorándose. Podremos añadir nuevos tipos de anomalías en función de las necesidades que se nos marquen o añadir nuevas funcionalidades. Un ejemplo podría ser detectar fallos en ciertos electrodomésticos o ser capaces de predecir, en función de los datos aprendidos, cuando debería estar despierta una persona y así poder notificar de forma más precisa. Además, este proyecto ha suscitado interés por parte de fundaciones como la Cruz Roja, por lo que podrá seguir obteniendo mejoras y se irá adaptando a las necesidades del cliente que obtenga este servicio.

Bibliografía

[1] Victoria Hodge and Jim Austin. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2):85–126, October 2004.

[2] Animesh Patcha and Jung-Min Park. An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends. *Comput. Netw.*, 51(12):3448–3470, August 2007.

[3] James P. Theiler and D. Michael Cai. Resampling approach for anomaly detection in multispectral images. volume 5093, pages 230–241. International Society for Optics and Photonics, September 2003.

[4] Gamze Oğcu and and Omer F. Demirel. Forecasting Electricity Consumption with Neural Networks and Support Vector Regression - ScienceDirect, 2012.

[5] Daniel L. Marino, Kasun Amarasinghe, and Milos Manic. Building Energy Load Forecasting using Deep Neural Networks. *arXiv:1610.09460 [cs]*, October 2016. arXiv: 1610.09460.

[6] M. Davy and S. Godsill. Detection of abrupt spectral changes using support vector machines an application to audio signal segmentation. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II–1313–II–1316, May 2002.

[7] Beyzanur Cayir Ervural and Omer Faruk Beyca. Model Estimation of ARMA Using Genetic Algorithms: A Case Study of Forecasting Natural Gas Consumption. *Procedia - Social and Behavioral Sciences*, 235:537–545, November 2016.

[8] Ritesh Maheshwari Yang, Yang. Robust anomaly detection for real user monitoring data: Systems engineering conference: O'Reilly Velocity, Santa Clara, CA, June 20 - 23, 2016.