Aplicación web para el análisis estadístico del mercado del consumo eléctrico

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Samuel Guerra Marrero

TUTORIZADO POR:
Javier Sánchez Pérez y Nelson Monzón López

Septiembre 2020

Agradecimientos

En primer lugar, quiero agradecer a mis tutores Nelson Monzón López y Javier Sánchez Pérez por la guía y la ayuda que me han ofrecido a lo largo de la elaboración del proyecto. Además, me gustaría dar las gracias a mis amigos y familia, en especial a mi madre y mi abuela por apoyarme y animarme cada día a continuar, haciendo posible que esta etapa haya transcurrido de una forma más amena y agradable, así como permitiéndome alcanzar mis objetivos.

Muchas gracias a todos.

Resumen

Comprender la factura de la luz normalmente resulta una labor compleja debido a la gran cantidad de datos que contienen, requiriendo conocimiento de ciertos términos que dificultan a los clientes su necesidad de entender qué están pagando realmente y qué han consumido. Este proyecto propone un sistema web que ofrece de forma automática un análisis de la información contenida en las facturas de los usuarios obteniendo estadísticas valiosas. Además, plantea un sistema gráfico que estudia la evolución del mercado eléctrico dependiendo de la zona y de la compañía eléctrica.

Abstract

Understanding electricity bills is usually a complex task due to the large amount of data they contain, requiring knowledge of certain terms that make it difficult for users to understand what they are really paying for and what they have consumed. This project proposes a web system that offers an automatic analysis of users bills information getting valuable statistics. In addition, it shows a graphical system that studies electricity market evolution depending on the area and the electricity company.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Debido a la alta competitividad en continuo aumento dentro del sector eléctrico en España, uno de los mayores retos a superar consiste en ofrecer precios lo más competitivos y adaptados al mercado posibles. Por esta razón, este proyecto presenta la idea de crear un sistema web que permita concebir un mercado formado por clientes y empresas que puedan relacionarse entre sí.

En este sector se diferencian principalmente dos tipos de empresa. Por un lado, las distribuidoras son empresas que se dedican a transportar energía a negocios y hogares. Son las propietarias de las infraestructuras que canalizan la energía hasta los distintos puntos de consumo. Por otro lado, las comercializadoras eléctricas son aquellas empresas, encargadas de vender la electricidad a los clientes finales, pagando una tasa a la empresa distribuidora por usar su red eléctrica.

Las compañías comercializadoras envían periódicamente a sus clientes una carta de pago que suele compartir una estructura común. Las facturas de la luz contienen una gran cantidad de datos, entre los cuales destacan, periodo de facturación, importe de potencia, datos del cliente y datos del contrato eléctrico en los que se incluye el número de contrato, el Código Universal del Punto de Suministro (CUPS), la potencia contratada, la tarifa de acceso, importe del peaje, entre muchos otros.

Todo estos parámetros dificultan enormemente la labor de entender el documento por parte de los clientes, ya que les exige conocer la terminología de este sector. Por lo tanto, en la mayoría de los casos, los clientes no saben qué han consumido y qué han pagado realmente.

Para intentar solventar este problema, este proyecto propone un sistema que de forma automática analiza la información que contienen las facturas de los usuarios obteniendo estadísticas útiles. También se plantea desarrollar un sistema gráfico que estudie la evolución del mercado eléctrico a partir de datos, tales como precio, cantidad de potencia contratada y consumida, impuestos, etc; en función de la compañía eléctrica y de la zona. Para llevar a cabo este análisis gráfico se emplearán los datos de las facturas que los usuarios suban a la plataforma.

Adicionalmente, se propone crear un modelo que genere nuevas facturas a partir de "*data augmentation*" y que ayude a integrar una pequeña red neuronal que reaccione ante la información recibida.

Por último, gracias a la unión de clientes y empresas que permite esta aplicación, las empresas comercializadoras podrán ayudarse de esta herramienta para captar nuevos clientes potenciales enviándoles sus ofertas.

1.1 Objetivos

Este proyecto se plantea conforme a los siguientes objetivos:

- Desarrollo de una plataforma web para subir facturas (PDF o imágenes) y visualizar estadísticas a partir del estudio de su información.
- Aprender nuevas tecnologías, extender conocimientos de ingeniería del software y aplicar nuevas técnicas de desarrollo.
- Diseñar y desarrollar una interfaz gráfica de acceso y manejo de la información, que contemplará parámetros de usabilidad y experiencia de usuario.
- Poner en práctica un flujo de trabajo que agrupe técnicas de integración y entrega continua así como familiarización con metodologías ágiles cercanas a su uso empresarial.
- Elaboración de la documentación mínima necesaria en cada una de las fases para su posterior uso incluyendo las fases de pruebas y puesta en producción.

1.2 Competencias

Seguidamente se mencionan las competencias del Grado en Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria que han sido cubiertas en este trabajo:

IS01: Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.

Esta competencia se justifica con la implementación del patrón de diseño MVC (Modelo Vista Controlador) y un desarrollo, en algunos puntos, guiado por pruebas TDD (Desarrollo dirigido por pruebas) que facilitan a su vez un mínimo de calidad, la escalabilidad y el mantenimiento de la aplicación.

IS02: Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones.

La competencia mencionada ha sido cubierta con la realización de varias reuniones con los tutores, en las cuales se ha creado una lista con los requisitos del cliente y los requisitos del sistema (pila de producto), apoyándonos en la metodología de desarrollo ágil *Scrum*, para resolver las distintas necesidades de los usuarios de la aplicación.

IS03: Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.

Esta competencia queda cubierta con el uso de las distintas tecnologías que se listan en el capítulo 3

IS04: Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.

Esta competencia se ve reflejada con el uso del modelo de desarrollo ágil *Scrum* adoptando una estrategia de desarrollo incremental y entrega continua.

TFG01: Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas

La competencia TFG01 queda justificada al acabar el Trabajo de Fin de Grado y presentar el proyecto ante el tribunal universitario.

1.3 Estado del arte

En este apartado se mencionan algunas de las aplicaciones existentes en el mercado, relacionadas con el análisis del mercado del consumo eléctrico en España y Europa.

Por lo general, cada comercializadora suele contar con una web o una aplicación propia en la cual mantienen contacto con sus clientes, como por ejemplo las dos que se mencionan a continuación:

- **Endesa** [End]: es una de las empresas más importantes dentro del sector energético en España. Actualmente, su principal actividad consiste en la generación, comercialización y distribución de energía en España, Marruecos, Francia, Portugal y otros países europeos. La aplicación de Endesa ofrece a sus clientes varias funcionalidades, entre ellas, ver y descargar la factura electrónica, efectuar el pago de la factura online y acceder al histórico de sus facturas. En la figura **Figura** 1.1 [Facb] observamos el apartado "Mis facturas" de esta aplicación.

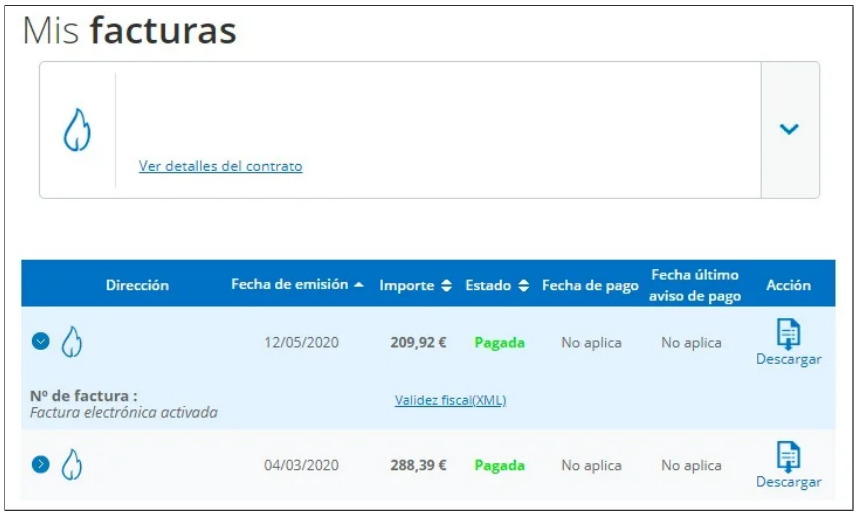

Figura 1.1: Ejemplo sección "Mis facturas" de Endesa

Además, ofrece a sus clientes la ventaja de analizar sus consumos como se observa en la **Figura** 1.2 [Con].

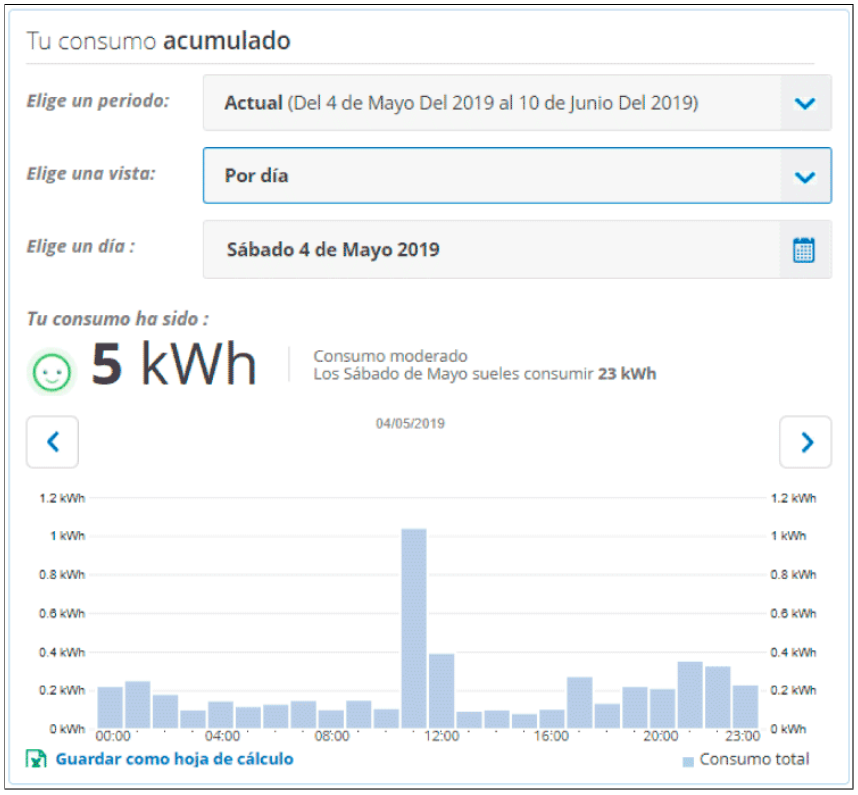

Figura 1.2: Ejemplo gráficos de consumo de cliente de Endesa

- **Naturgy** [Nat]: al igual que Endesa, se trata de una de las mayores comercializadoras y distribuidoras de energía en el territorio español. Esta compañía también permite a sus clientes la visualización de sus facturas online, poner en marcha reclamaciones, modificar parámetros de su contrato (domiciliación bancaria, titular de contrato, tipo de tarifa, etc.). En la **Figura** 1.3 [Ima] se contempla un ejemplo de estadísticas de consumo.

Figura 1.3: Ejemplo gráficos de consumo de cliente de Naturgy

Por otro lado, existe la empresa *GrupoASE* [Lóp01] que realiza informes del mercado energético español y europeo proporcionando los datos más relevantes y actualizados. Además, contextualizan los factores que explican el precio de la luz. En la **Figura** 1.4 se observa un ejemplo de un gráfico del precio del mercado diario eléctrico. No se trata de ninguna comercializadora ni está dirigida a consumidores particulares, sino a asesorar a otras empresas.

Figura 1.4: Ejemplo gráfico Grupo ASE

Por último, mencionar la web *Tarify* [Tar] que permite al usuario comparar precios de distintas comercializadoras de Luz, así como de Gas, móvil e internet. En la **Figura** 1.5 se aprecia un ejemplo de comparativa de precios de la luz de diferentes comercializadoras.

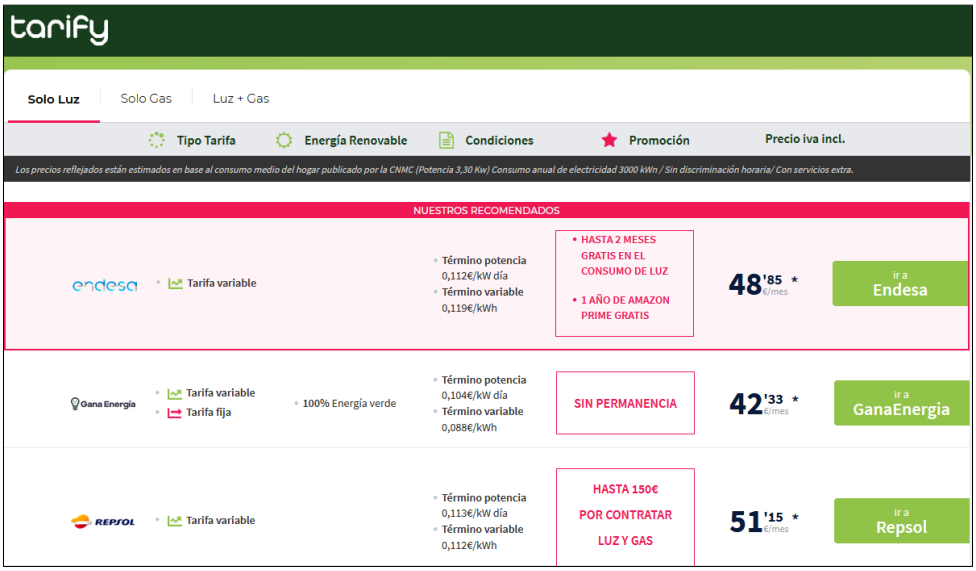

Figura 1.5: Comparativa precios Tarify

1.4 Aportaciones

Como se ha visto en el apartado anterior, los clientes tienen a su disposición varias herramientas para realizar un análisis del mercado del consumo en el sector eléctrico, aunque muchas veces dependen del sistema de su comercializadora cuando se trata de examinar sus consumos y gastos, que no suelen ser los más claros y directos en la información relevante para el usuario. Sin embargo, este proyecto pretende mejorar las alternativas existentes aportando las características que se nombran a continuación, además de ciertas funcionalidades que poseen actualmente las distintas aplicaciones o webs existentes en el mercado, como las que se han explicado anteriormente.

En primer lugar, este proyecto trata de formar un mercado que reúne tanto a las empresas (comercializadoras y distribuidoras) como a los clientes (particulares u otras empresas). Este sistema sirve para que sus usuarios puedan relacionarse entre ellos, de tal forma que las empresas pueden recibir notificaciones de potenciales clientes a los que poder enviar sus ofertas, y los clientes podrán recibir estas ofertas para posteriormente valorarlas y comunicarse con la empresa en cuestión. En cierta medida, esta solución elimina la tarea del cliente de tener que buscar las ofertas ofrecidas por cada comercializadora en su propia web.

Tanto las empresas como los clientes podrán consultar y comparar los precios de las distintas tarifas que han publicado las comercializadoras. En el caso concreto de las comercializadoras podrán comparar sus propias ofertas con las de su competencia.

Por otro lado, los clientes pueden subir las facturas que hayan recibido, bien en formato PDF o en imagen, para que la aplicación les ofrezca de manera automática un análisis de la información que contiene el documento, desechando los datos menos relevantes y mostrando aquellos más importantes para el cliente.

Por último, cualquier usuario que utilice la aplicación podrá buscar y consultar la información de todas las comercializadoras y distribuidoras que formen parte de alguna región de España. Asimismo, podrán visualizar la evolución que han tomado los precios de las comercializadoras en función de la región que elijan, a través de un sistema gráfico que estudia la transformación del mercado eléctrico. Además, será posible comparar dicha variación entre varias regiones.

Capítulo 2

Metodología

En este apartado se explicará qué estrategias de gestión del trabajo se han llevado a cabo durante la ejecución del proyecto.

En este trabajo se ha puesto en práctica la metodología de desarrollo ágil *Scrum*, complementada a su vez, por técnicas de representación visual del flujo de trabajo denominadas *Kanban*. También se ha seguido un desarrollo determinado por el flujo de trabajo *GitFlow*. A pesar de ser un proyecto individual en cuanto a implementación del código, se ha decidido seguir este modelo de ramificación para ponerlo en práctica y seguir un orden de versionado de la aplicación. A continuación, se describirán estos conceptos más detalladamente y cómo se han utilizado a lo largo del trabajo.

2.1 *Scrum*

Se puede considerar la metodología *Scrum* desde dos puntos de vista. Por un lado, el marco estándar o *Scrum* técnico, que se basa en la aplicación de una serie de reglas definidas haciendo uso de roles, artefactos y eventos. Por otro lado, se encuentra el llamado *Scrum* avanzado que promueve la aplicación de valores ágiles, permitiendo a su vez, una gestión más flexible.

El enfoque que se ha efectuado en este proyecto ha sido el *Scrum* avanzado, sin embargo, se utilizan ciertos conceptos del *Scrum* técnico que son necesarios explicar.

2.1.1 Definición

Scrum es un marco de desarrollo ágil que define un conjunto de prácticas y roles que se caracteriza, entre otros aspectos, por: [Ale19, Pag. 16]

- En lugar de emplear una gestión predictiva del producto, se opta por una gestión evolutiva, en la cual no se busca predecir lo que va a ocurrir, sino adaptarse a los cambios que van sucediendo en el tiempo para desarrollar el producto.
- En lugar de ejecutar las fases de desarrollo de forma secuencial o en cascada, se utiliza un procedimiento de fases superpuestas.
- Emplear una táctica de desarrollo incremental ayudándose de iteraciones o *sprints*.
- Dar más importancia la calidad del resultado basado en el conocimiento implícito de las personas, en lugar del conocimiento definido por los procesos y la tecnología utilizada.

Además, *Scrum* es una metodología simple que reduce la carga de gestión. Es un sistema basado en entregas continuas que presupone desde el primer momento requisitos indefinidos y que cambiarán en el tiempo. En la **Figura** 2.1 [Fra] se presenta el flujo de trabajo de *Scrum*.

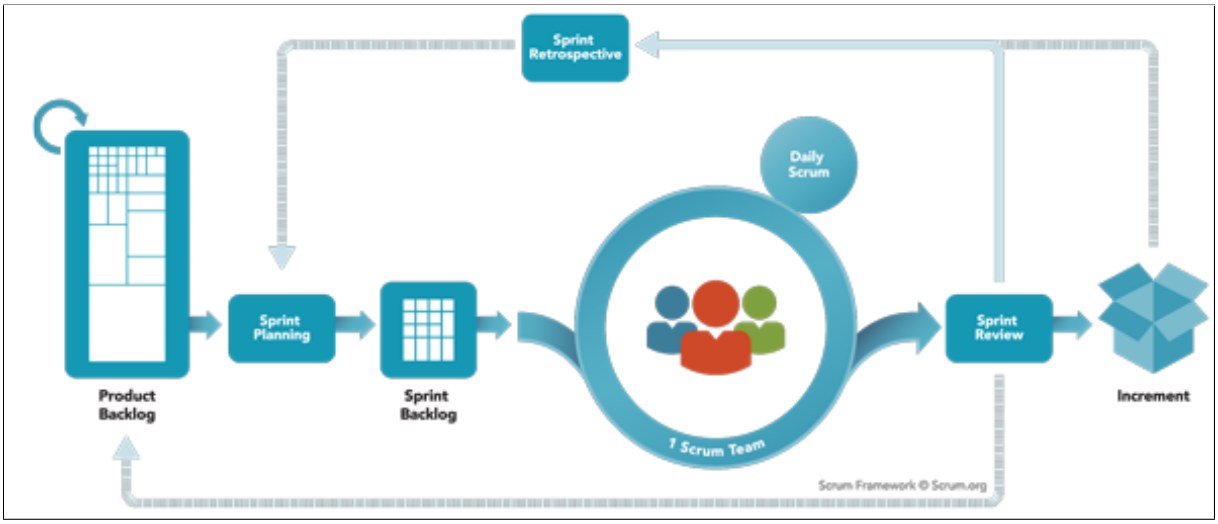

Figura 2.1: Flujo de trabajo *Scrum*

2.1.2 Artefactos

[Ale19, Pag. 21]

- Pila del producto: se trata de un listado de las condiciones requeridas por el usuario, que va cambiando y evolucionando a medida que el producto avanza en su desarrollo.
- Pila del sprint: es el conjunto de historias de usuario que el equipo debe llevar a cabo a lo largo de una iteración, para finalmente originar el próximo incremento.
- Incremento: se llama así a la parte del producto que se genera en cada *sprint*.

Un ejemplo de la configuración de la pila de producto utilizada se observa en la **Tabla** 2.1

Id	Nombre	Descripción	Criterios de validación	Horas estimadas
7	Ver facturas	Como cliente quiero poder ver mis facturas	El cliente puede acceder a un registro de todas las facturas que ha ido subiendo a la plataforma	6

Tabla 2.1: Formato de la pila de producto

2.1.3 Eventos

[Ale19, Pag. 26]

- Sprint: ciclo de tiempo en el que se desarrolla cada incremento iterativo del producto.
- Reunión de Planificación del sprint: reunión en la cual se decide el plan que se seguirá para completar el siguiente sprint. Para ello se seleccionan las historias de usuario necesarias para alcanzar el objetivo.
- Revisión del sprint: reunión en la que se analiza el resultado obtenido del último incremento añadido al producto. Si se precisa, en este punto se puede actualizar la pila de producto.
- *Scrum* diario: reunión que se realiza todos los días en la cual el equipo actualiza el trabajo que ya ha completado y el esfuerzo de lo que queda por realizar.
- Retrospectiva del sprint: en esta reunión se revisan los puntos postivos y negativos del último sprint para tratar de mejorar en próximas iteraciones.

El proyecto se ha dividido inicialmente en los siguientes *Sprints* que serán descritos con más detalle en el **Capítulo** 4. Esta planificación estuvo sujeta a cambios durante el desarrollo del proyecto:

- **Sprint 0**: Diseño de un prototipo simple de la aplicación.
- **Sprint 1**: Elaboración de un sencillo sistema para reconocer e interpretar facturas en formato PDF o imagen, además de una interfaz simple de visualización de resultados.
- **Sprint 2**: Diseño de un esquema de base de datos relacional, para la persistencia de los datos de los distintos modelos/roles que participan en la aplicación, así como, la elaboración de la interfaz que permita a los usuarios interacionar y ejecutar las distintas características de la aplicación. También, añadir la creación de un programa de generación de facturas, clientes y empresas.

2.1.4 Roles

[Ale19, Pag. 31]

- Propietario del producto: es la persona que trata de conseguir que el producto obtenga el máximo valor posible para los usuarios. Se encarga de tomar las decisiones en nombre del cliente con el cual mantiene contacto.
- Equipo de desarrollo: grupo de personas que aportan en cada incremento valor añadido al producto.
- *Scrum Master*: es el líder del equipo de desarrollo. Se ocupa de resolver impedimentos que sucedan durante el desarrollo y desempeña labores de intermediario entre el propietario del producto y el equipo de desarrollo.

Para el caso de este proyecto en particular, no se pueden diferenciar los roles al ser un trabajo individual.

2.2 *Kanban*

Kanban es un sistema de gestión visual que persigue manejar el flujo de trabajo en estados diferenciados. Con esto se consigue identificar las distintas fases por las que va pasando una historia de usuario. Esta herramienta también trata de evitar los cuellos de botella y los tiempos muertos.

En la **Figura** 2.2 se observa un ejemplo del tablero, en el cual se pueden diferenciar varios estados:

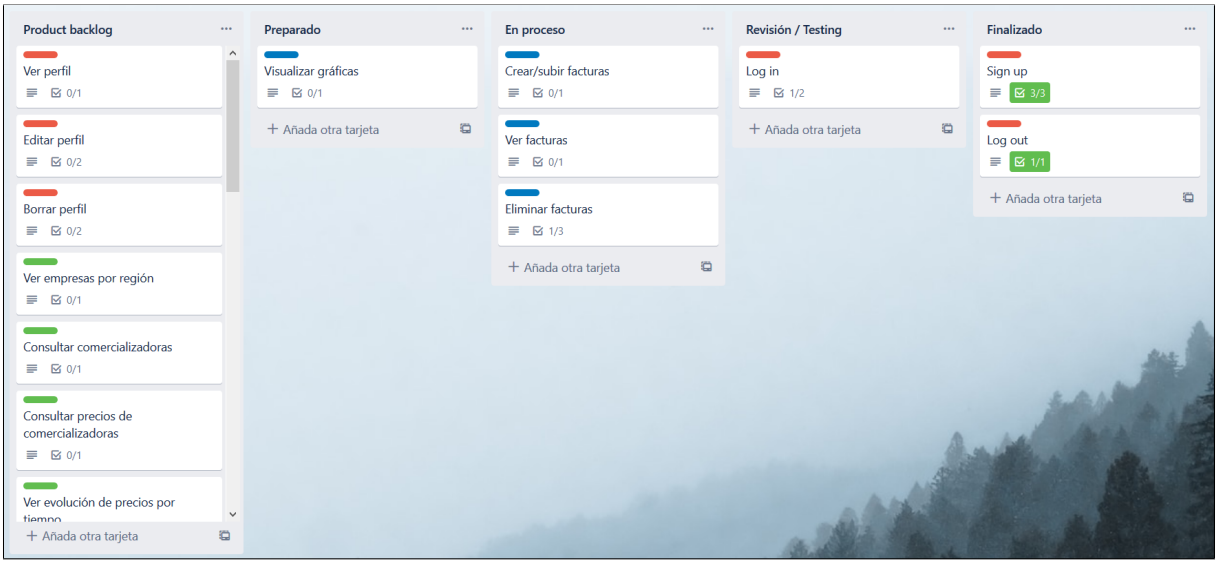

Figura 2.2: Tablero *Kanban*

- **Pila de producto**: En este estado se encuentran las historias de usuario en evaluación. Todavía se debe decidir si se incorporarán al producto o no.
- **Preparado**: A este estado pasan las historias de usuario aprobadas que se añadirán al producto. Estas historias ya están valoradas y priorizadas.
- **En proceso**: Estado en el que permanecen las historias de usuario que se encuentran en desarrollo.
- **Revisión/Pruebas**: Las historias llegan a este estado para pasar por un ciclo de pruebas y evaluación antes de darse por finalizadas.
- **Finalizado**: Estado final al que llegan las historias elaboradas y testeadas.

2.3 *Gitflow*

Gitflow consiste en un conjunto de extensiones del sistema de control de versiones *Git*, que facilitan en gran medida el trabajo y proporciona un marco sólido para la gestión de proyectos. Se detalla a continuación el funcionamiento de los diferentes tipos de ramas que emplea:

2.3.1 Ramas maestra y desarrollo

Estas ramas son las más importantes e imprescindibles en este flujo de trabajo. En la **Figura** 2.3 [Atl] se observa un ejemplo de dichas ramas.

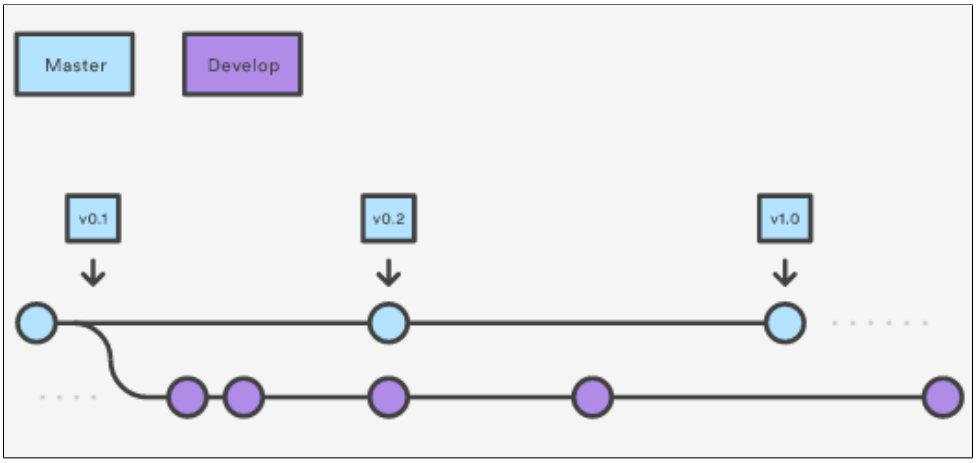

Figura 2.3: *Gitflow* ramas maestra y desarrollo

- **Maestra**: esta rama almacena el historial de versiones en producción.
- **Desarrollo**: esta rama sirve como apoyo para la integración de nuevas funciones y llevará el historial completo de todo el proyecto.

2.3.2 Rama característica

Cada nueva función que se añada al proyecto debe desarrollarse en su propia rama. Para esto se crea una rama característica que toma como rama principal la rama desarrollo. Cuando una rama característica concluye, se fusiona con la rama desarrollo. La rama característica nunca debe entrar en contacto directamente con la rama maestra como se observa en la **Figura** 2.4 [Atl].

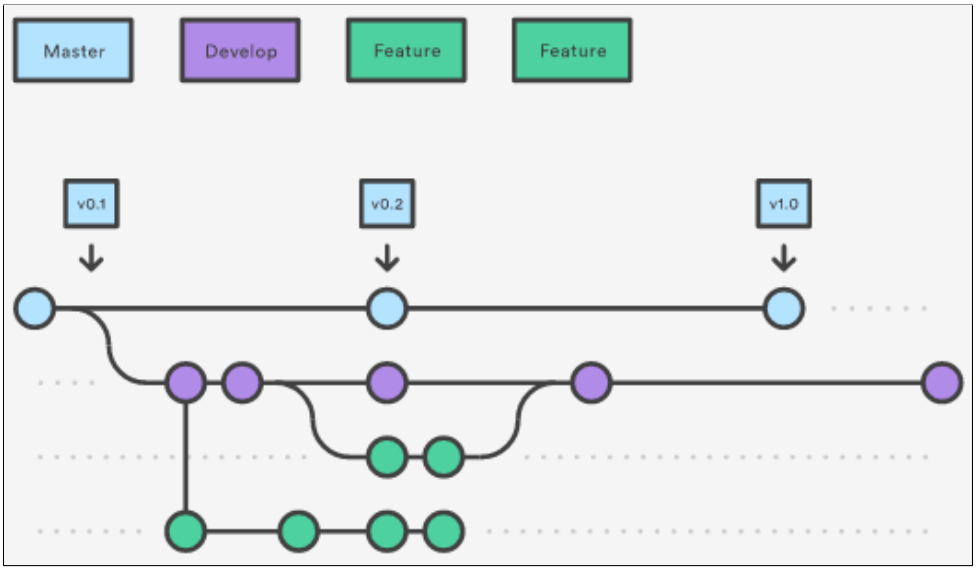

Figura 2.4: *Gitflow* rama característica

En este proyecto se han ido creando diversas ramas característica a lo largo del desarrollo como por ejemplo:

- ***blueprint-structure***: rama empleada para establecer una estructura de *blueprints* en el *back end*.
- ***add-react-frontend***: rama utilizada para añadir *ReactJS* a la aplicación.
- ***user-profile***: rama creada para añadir RUD (leer, actualizar y eliminar) del perfil de usuario.
- ***offers***: rama utilizada para añadir todas las historias de usuario relacionadas con las ofertas.
- ***user-not-logged-in***: rama en la que se añaden las historias de usuario relacionadas con los usuarios que no han iniciado sesión.

2.3.3 Rama versión

Una vez que se determine que la rama desarrollo ha adquirido suficientes nuevas funcionalidades, o bien se esté acercando una fecha de publicación, se creará una nueva rama versión. Mientras esta rama esté activa, no pueden añadirse nuevas ramas característica.

En esta rama versión los desarrolladores pueden solucionar errores, generar documentación y otras tareas relacionadas con la puesta en producción de la nueva versión.

Una vez esté todo listo, la rama versión se fusiona en la rama maestra quedando etiquetada con un número de versión. Igualmente, se fusionará con la rama desarrollo.

En la **Figura** 2.5 [Atl] se observa un ejemplo de funcionamiento de esta rama.

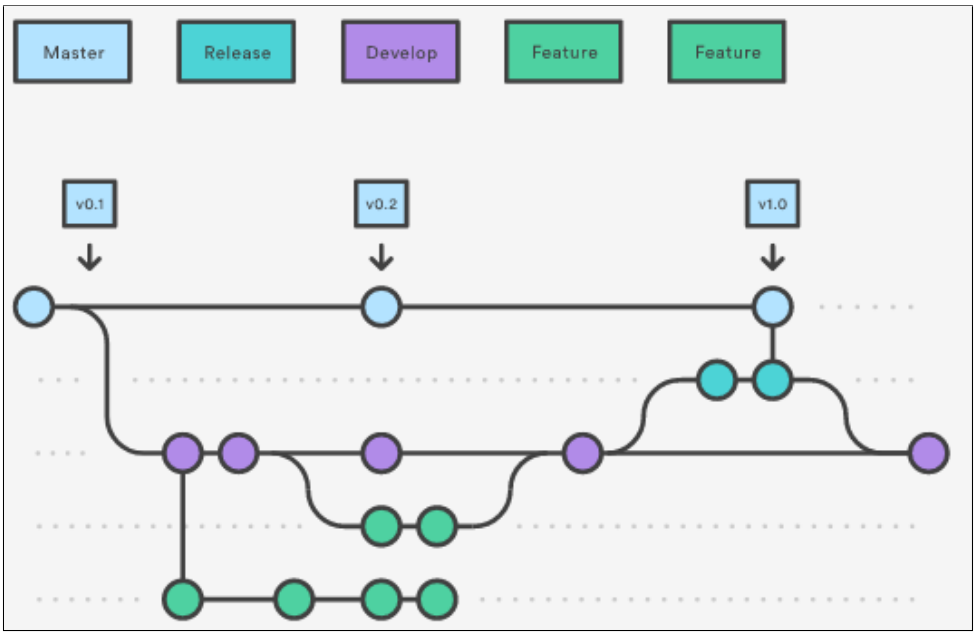

Figura 2.5: Gitflow rama versión

En este proyecto se han generado cuatro ramas versión, correspondiendo con las cuatro versiones que ha experimentado la aplicación como se verá en el capítulo 4.

2.3.4 Rama correción

Estas ramas se utilizan cuando se detectan errores en producción (en la rama maestra) para solucionarlos rápidamente. Es la única rama que se bifurca directamente desde la rama maestra. Una vez solucionado el bug, la rama corrección se fusionará con la rama maestra y la rama desarrollo. Un ejemplo de uso de la rama corrección se observa en la **Figura** 2.6 [Atl].

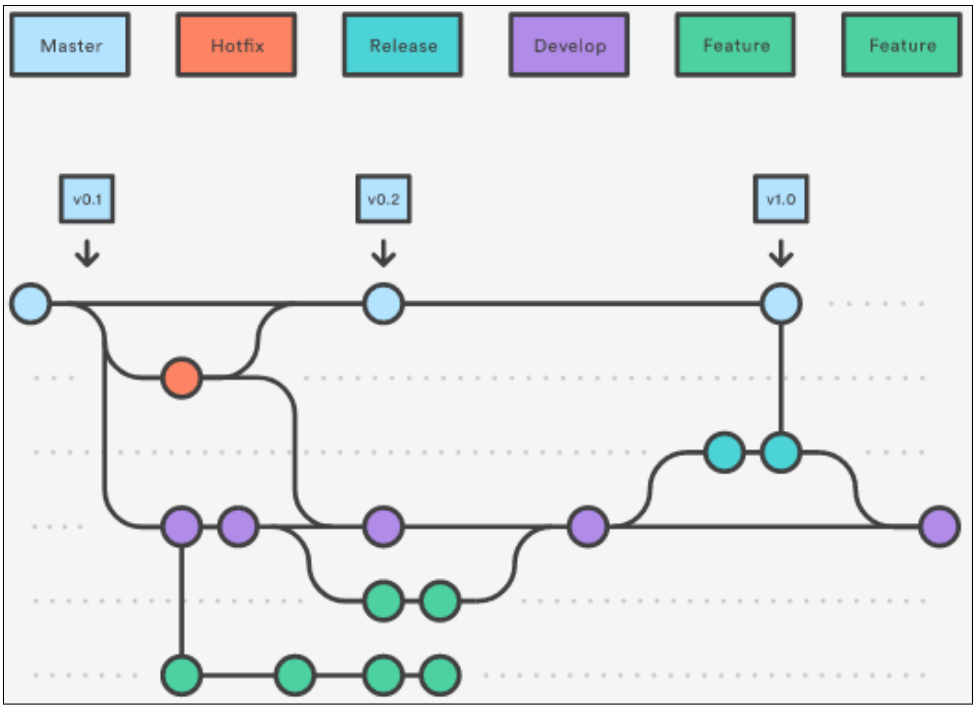

Figura 2.6: Gitflow rama corrección

Obviamente, para este proyecto no se ha requerido el uso de esta rama ya que la aplicación no se ha desplegado en producción.

Capítulo 3

Tecnologías

En este capítulo se nombrarán las diferentes tecnologías y herramientas utilizadas para la realización del proyecto.

- **Oracle VM VirtualBox**: es un software de virtualización por medio del cual es posible instalar sistemas operativos adicionales, conocidos como sistemas invitados, dentro de otro sistema operativo anfitrión, cada uno con su propio entorno virtual. Ha sido necesaria la utilización de esta herramienta para desarrollar el trabajo bajo GNU Linux, más concretamente usando la distribución Ubuntu, ya que el prototipo reconocedor de facturas inicial se elaboró en este entorno. [Gmb07]

- **Visual Studio Code**: es un editor de código fuente desarrollado por Microsoft para los sistemas operativos Windows, Linux y macOS. Incluye soporte para la depuración, control de versiones con Git, resaltado de sintaxis, autocompletado inteligente y refactorización de código. [Mic15b]

- **Git**: es un sistema de control de versiones distribuido y gratuito de código abierto diseñado para manejar con velocidad y eficiencia cualquier recurso, desde un proyecto pequeño hasta proyectos muy grandes. Su objetivo es llevar el registro de los cambios en ficheros y directorios y coordinar el trabajo que varias personas realizan sobre archivos compartidos. [Tor05]

- **GitKraken**: es una potente interfaz gráfica multiplataforma para gestionar proyectos que usen Git como sistema de control de versiones. [Axo14]

- **GitHub**: es una plataforma de desarrollo colaborativo para ubicar proyectos utilizando el sistema de control de versiones Git. [Tom08]

- **MySQL**: es un sistema de gestión de bases de datos relacional. [MyS95]

- **dbdiagram**: es una herramienta que permite diseñar la estructura de bases de datos relacionales. Se ha utilizado para la representación de las tablas y relaciones de la base de datos de la aplicación. [Hol]

- **Python**: es un lenguaje de programación interpretado de alto nivel y de propósito general. Igualmente, se trata de un lenguaje multiparadigma, ya que soporta programación funcional, orientación a objetos y programación imperativa. Además, es multiplataforma, lo que permite ejecutarlo en varios sistemas operativos. [Ros91]

- **Flask**: es un *framework* minimalista escrito en Python. Está diseñado para comenzar a desarrollar de forma rápida y sencilla, y capacitado para escalar a aplicaciones complejas. [Ron04]

- **Flask-SQL-Alchemy**: es una extensión de Flask que permite usar SQLAlchemy en la aplicación Flask. SQLAlchemy es un ORM que ayuda a manejar la información empleando un mapeo de las estructuras de la base de datos sobre una estructura lógica de objetos con el objetivo de simplificar y acelerar el desarrollo de la aplicación. [Alc10] [Bay06]

- **Flask-JWT-Extended**: es una extensión de Flask que permite la gestión de JSON Web Token (JWT) en una aplicación Flask. Se utiliza para aportar una mayor seguridad al sistema de login de la aplicación, empleando un token que validará al usuario logueado que esté usando la aplicación. [Ext]

- **JavaScript**: es un lenguaje de programación interpretado que se utiliza mayormente en el lado del cliente en el desarrollo de aplicaciones web. Es un lenguaje orientado a objetos, imperativo, de tipado débil y dinámico. [Eic95]

- **React**: es una biblioteca de JavaScript de código abierto focalizada en el desarrollo de interfaces de usuario, especialmente de una única página. [Wal13]

- **React-Redux**: es un patrón de arquitectura de datos que permite manejar el estado de de aplicaciones que usan React de una forma predecible. [Dan15]

- **React-Router**: es una colección de componentes que permiten gestionar rutas en aplicaciones que utilicen ReactJS. [Tra]

- **Highcharts**: es una biblioteca moderna de gráficos multiplataforma escrito en JavaScript que ofrece una manera sencilla de agregar gráficos interactivos en aplicaciones móviles o proyectos web. Cuenta con una extensa documentación, un sólido conjunto de funciones y destaca por su facilidad de uso. [Tor09]

- **Highcharts-React**: es una capa envoltorio de Highchart para poder utilizar todas las funcionalidades de esta biblioteca usando la biblioteca de construcción de interfaces React. [Rea]

- **Axios**: es un cliente HTTP para JavaScript basado en promesas. Se utiliza para conectar el front end con la API del back-end realizando llamadas asíncronas. [Zab14]

- **Postman**: es una herramienta que nos permite realizar peticiones a una API. Con este software se consigue hacer más sencillo cada uno de los pasos de creación de las APIs. En este proyecto se ha utilizado para realizar peticiones y probar la API. [Ast]

- **Bootstrap**: es un conjunto de herramientas y componentes que ayudan a diseñar interfaces de usuario de aplicaciones. [Mar11]

- **MaterialUI**: es un *framework* para las interfaces de usuario que usan React. Se compone de un conjunto de componentes de React que facilitan el desarrollo de interfaces web. [Goo14]

- **Tesseract**: es un reconocedor óptico de caracteres disponible para varios sistemas operativos. [Kah03]
- **LaTeX**: es un sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Incluye características diseñadas para la producción de documentación técnica y científica. [Lam84]
- **Overleaf**: es un servicio *online* que posibilita la edición de documentos utilizando LaTeX. Asimismo, permite compartir la edición de los documentos con otros colaboradores. [Joh12]

Capítulo 4

Desarrollo

En este capítulo se observará en primer lugar la planificación establecida. Hay que destacar que ha estado sujeta a cambios durante el desarrollo. Después, se recorrerá cada una de las fases explicando la elaboración de sus respectivas tareas.

4.1 Planificación

Originalmente se ha establecido una organización del trabajo a realizar dividida en las fases y sus correspondientes tareas que se enumeran a continuación en la **Tabla**: 4.1

Fases	**Horas estimadas**	**Tareas**
1. Creación de la pila de producto inicial	20	Análisis de la situación actual. (10 h) Primera toma de historias de usuario y priorización de necesidades. (10 h)
2. Sprint *Zero*	40	Diseño de la base de datos inicial. (15 h) Diseño y desarrollo de prototipo inicial de reconocimiento de facturas. (25 h)
3. Versión 1 (Sprint 1)	55	Sistema reconocedor de facturas mejorado. (5 h) Sistema de simulación de facturas. (10 h) Sistema de autenticación de usuarios. (25 h) Sistema básico de estadísticas de consumo. (15h)
4. Versión 2 (Sprint 2)	50	Implementación *front end* con *ReactJS*. (30 h) Sistema de visualización de gráficos mejorado. (20 h)
5. Versión 3 (Sprint 3)	70	RUD perfil de usuario (10 h). CRUD ofertas (12 h). Clientes potenciales (8 h). Enviar/Recibir ofertas (20 h). Consultar/Comparar ofertas (20 h).
6. Versión 4 (Sprint 4)	25	Ver empresas por región (15 h). Comparativa de evolución de precios por región (10 h).

Documentación / Presentación	40	Redacción de la memoria (30 h) Preparación de la presentación (10 h)

Tabla 4.1: Planificación

4.2 Creación de la pila de producto

4.2.1 Análisis situación actual

Previamente al desarrollo de la aplicación se ha realizado un estudio de soluciones existentes relacionadas con el mercado del consumo eléctrico en España y Europa, redactado en el apartado 1.3. Este análisis ha servido para reconocer las principales virtudes y debilidades de estas aplicaciones y por consiguiente, determinar qué funcionalidades y requisitos debería cumplir esta aplicación. Como resultado se ha obtenido la pila de producto que se describirá próximamente.

4.2.2 Pila de producto

Se puede examinar la pila de producto en el **Anexo** A. Cabe aclarar que las historias de usuario que aparecen marcadas con un fondo gris son aquellas que no se han implementado en la aplicación y quedan pendientes para un posible trabajo futuro.

4.3 Sprint Zero

4.3.1 Tipos de usuario

En esta sección se detallan los dos tipos de usuario presentes en la aplicación, así como sus diferentes funcionalidades.

- Cliente
  - CRUD facturas.
  - Visualización de estadísticas de consumo a través de gráficos.
  - Analizar/Recibir ofertas.
  - Consultar/Comparar comercializadoras y precios.
- Empresa
  - CRUD ofertas.
  - Visualización de clientes propios y potenciales.
  - Enviar ofertas.
  - Consultar/Comparar competencia y precios.

Además, hay que distinguir que un usuario de la aplicación puede haber iniciado sesión o estar navegando como usuario invitado (usuario no logueado). En el caso de haber iniciado sesión, el usuario debe ser de alguno de los tipos anteriormente mencionados. En la siguiente lista, se detallan algunas funcionalidades más que poseen tanto los usuarios logueados como los que no.

- Usuario logueado
  - CRUD perfil.
- Usuario no logueado (invitado)
  - Consultar comercializadoras y precios.
  - Ver empresas por región.
  - Ver evolución de precios por región.
  - Comparar evolución de precios.

Recalcar que CRUD es el acrónimo que representa las cuatro funciones básicas "Crear, Leer, Actualizar y Borrar" (del original en inglés: *Create*, *Read*, *Update* and *Delete*).

4.3.2 Estructura de la base de datos

A continuación se expondrán las tablas correspondientes a los modelos de datos más relevantes utilizados en la aplicación. Se han omitido en esta descripción aquellas tablas que cumplen funciones de relaciones entre tablas. Para ver el diseño completo de la estructura de la base de datos se puede consultar el **Anexo** B

La tabla ***users*** 4.2 contiene la información de los usuarios de la aplicación.

Nombre	**Descripción**
username	Nombre de usuario que sirve para iniciar sesión.
password	Contraseña del usuario necesaria para iniciar sesión.
user-type	Tipo de usuario (cliente o entrenador)

Tabla 4.2: Tabla users

La tabla ***contracts*** 4.3 guarda los datos de los contratos relacionados con las facturas de los clientes.

Nombre	Descripción
contract-number	Número de identificación del contrato.
contracted-power	Potencia contratada por el cliente.
toll-access	Peaje de acceso.
init-date	Fecha de inicio del contrato.
end-date	Fecha de finalización del contrato.
CNAE	Código de Clasificación Nacional de Actividades Económicas.
tariff-access	Tarifa de acceso.
description	Descripción del contrato.
conditions	Condiciones establecidas en el contrato.
cif	Código de identificación fiscal de la empresa que gestiona el contrato.
file	Archivo pdf o imagen del contrato.

Tabla 4.3: Tabla *contracts*

La tabla ***invoices*** 4.4 almacena las facturas que los clientes van subiendo a la aplicación.

Nombre	Descripción
invoice-number	Número de identificación de la factura.
contracted-power-amount	Importe por potencia contratada (euros).
consumed-energy-amount	Importe por potencia consumida (euros).
consumed-energy	Energía consumida en el periodo de facturación (kWh).
issue-date	Fecha de emisión de la factura.
charge-date	Fecha de cargo de la factura.
init-date	Fecha de inicio de la factura.
end-date	Fecha de fin de la factura.
total-amount	Importe total de la factura (euros).
tax	Porcentaje de impuestos ligado a la factura.
tax-amount	Importe correspondiente a los impuestos (euros).
contract-reference	Número de referencia que indica a qué contrato pertenece la factura.
file	Archivo pdf o imagen de la factura.

Tabla 4.4: Tabla *invoices*

La tabla ***customers*** 4.5 conserva los datos de los clientes que se hayan registrado en la aplicación.

Nombre	Descripción
nif	Número de identificación fiscal del cliente.
name	Nombre del cliente.
surname	Apellidos del cliente.
email	Email del cliente.
user-id	Id del usuario relacionado con el cliente.

Tabla 4.5: Tabla *customers*

La tabla ***companies*** 4.6 reúne todos los datos de las empresas que tengan una cuenta en la aplicación.

Nombre	Descripción
cif	Código de identificación fiscal de la empresa.
name	Nombre de la empresa.
address	Dirección de la empresa.
url	Dirección de la página web de la empresa.
email	Dirección de correo electrónico de la empresa.
company-type	Tipo de empresa (comercializadora o distribuidora).
phone	Teléfono de contacto de la empresa.
user-id	Id del usuario relacionado con la empresa.

Tabla 4.6: Tabla *companies*

La tabla ***offers*** 4.7 contiene los datos de las ofertas creadas por las diferentes comercializadoras registradas en la aplicación.

Nombre	Descripción
offer-type	Tipo de oferta (2.0 - Tarifa general, 2.0DHS - Tarifa supervalle, etc.).
fixed-term	Precio de la energía término fijo (euros).
variable-term	Precio de la energía término variable (euros).
tip	Precio de la energía en horario punta (euros).
valley	Precio de la energía en horario valle (euros).
super-valley	Precio de la energía en horario super valle (euros).
cif	Código de identificación fiscal de la empresa que ha creado la oferta.

Tabla 4.7: Tabla *offers*

4.3.3 Prototipo inicial de reconocimiento de facturas

El proyecto partió de un prototipo muy básico para reconocer las facturas. Para utilizarlo, el usuario que hace uso de la aplicación puede elegir una factura que tenga guardada en su dispositivo. Una vez elegida, el sistema automáticamente extrae toda la información del documento subido y la muestra en la página como se observa en la **figura** 4.1.

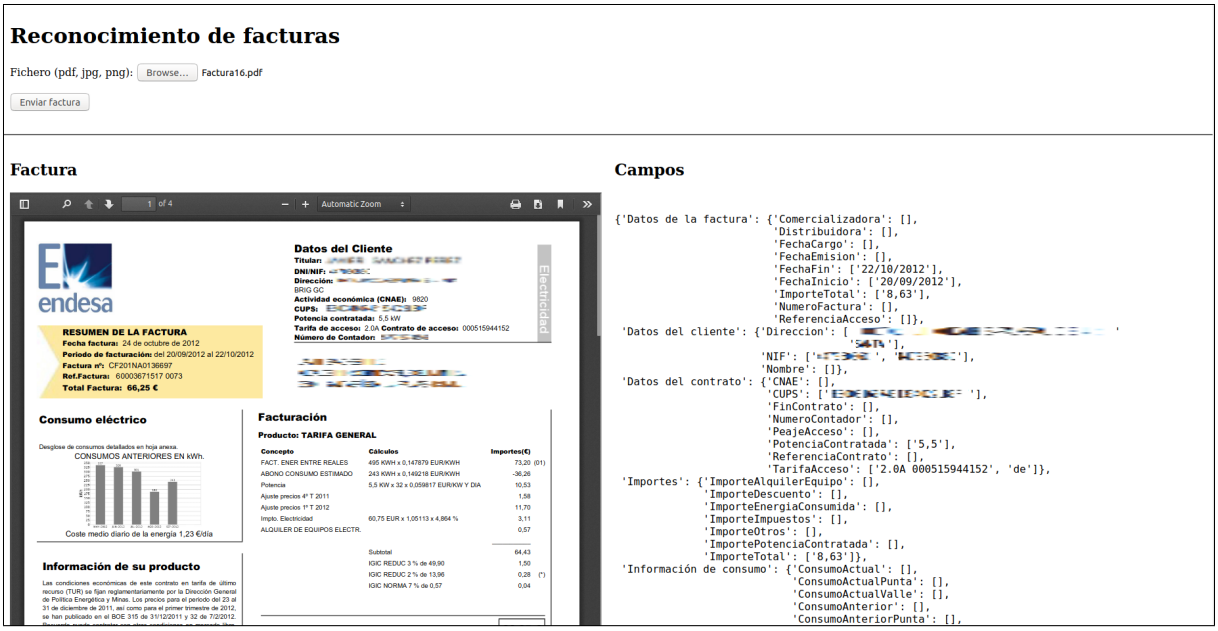

Figura 4.1: Reconocedor de facturas básico

Para realizar la tarea de reconocimiento, el sistema hace uso de un *OCR* (Sistema de reconocimiento óptico de caracteres), en concreto *Tesseract-OCR* [Smi07]. Básicamente, el proceso de reconocimiento se ha dividido en dos módulos. El primero se encarga de reconocer el tipo de formato que tiene la factura elegida por el usuario y de encargar la tarea de transformación a texto al programa correspondiente. El segundo módulo tiene como objetivo la extracción de los datos de la factura. A continuación se describen más en profundidad cada uno de los módulos.

En la **figura** 4.2 se observa qué tratamiento se le da a la factura elegida teniendo en cuenta su formato. Si el usuario ha seleccionado una factura en formato pdf, el sistema la analizará con el paquete "*pdftotext*". Por otro lado, si el usuario selecciona un archivo de texto se analizará directamente este archivo. Por último, para cualquier otro formato permitido de facturas, el sistema analizará el archivo utilizando el *OCR Tesseract*.

```
def process_bill(file_bill, type):

    file_txt = '/tmp/' + str(uuid.uuid4())
    results  = {}

    if type == 'pdf':
        subprocess.call(['pdftotext', file_bill, file_txt + '.txt', '-raw'])
    else:
        if type == 'txt':
            copyfile(file_bill, file_txt + '.txt')
        else:
            subprocess.call(['tesseract', file_bill, file_txt,
                             '--oem', '1', '-l', 'spa'])
```

Figura 4.2: Procesamiento de factura por tipo de archivo

Finalmente, en la **figura** 4.3 se aprecia el código que extrae la información del archivo de texto generado por la etapa anterior. En primer lugar, se lee el fichero generado en la fase anterior donde se encuentran todos los datos de la factura. Seguidamente se leen del archivo "*file_fields*" los nombres de los campos que se van a extraer. Para acabar, se buscan los campos en el fichero de texto de los datos de la factura y se retornan los valores encontrados.

```
def extract_data(file_txt, file_fields):

    with open(file_txt, 'r', encoding="utf-8") as file:
        contents=file.read()

    labels=eval(str(open(file_fields, 'r', encoding="utf-8").read()))
    results={}

    for field in labels.keys():
        results[field]=list()
        for label in labels[field]:
            search = re.search(label, contents, flags=re.IGNORECASE)
            if search:
                results[field].append(search.group(1))

    return results
```

Figura 4.3: Extracción de datos de factura

4.4 Versión 1 (Sprint 1)

4.4.1 Mejora del sistema reconocedor de facturas

En algunas situaciones, el sistema reconocedor de facturas no es capaz de encontrar ciertos datos de la factura analizada. Esto se debe, por un lado, a la gran variedad de formatos y diseños de facturas que existen. Por otro lado, a las limitaciones que presentan las expresiones regulares utilizadas para extraer los datos. En la **figura** 4.4 se observan varias expresiones regulares para la extracción de datos relacionados con la factura.

```
{
    'NumeroFactura': ('N. factura: (\S+)','N. de factura: (\S+)','Factura n.: (\S+)',),
    'ReferenciaAcceso': ('Referencia del contrato de acceso: (\S+)','Nº contrato de acceso\s*.*: (\w*)',),
    'FechaEmision': (
        'Fecha de emisión:? (.+)',
        'Fecha emisión factura:? (.+)',
        'Fecha emisión:? (.+)',
        'Fecha factura: (.+)',
    ),
    'FechaCargo': ('Fecha de cargo: (.+)',),
    'FechaInicio': (
        'Periodo de facturación:? (\S+)',
        'Del (\S+) a[l]? \S+ ',
        'del (\S+) AL \S+',
        'Periodo de consumo: (\S+)',
    ),
    'FechaFin': (
        'Periodo de facturación:? \S+ - (\S+)',
        'Del \S+ a[l]? (\S+)',
        'Periodo de consumo:? \S+ a (\S+)',
    ),
    'Comercializadora': ('comercializadora: (.+)', '(Gas.* S.A.)', '(Naturgy.* S.A.)',),
    'Distribuidora': ('distribuidora: (.+)', '(Endesa.*S.A.)', ),
    'CIF': (
        'CIF:(\d+).?',
        'CIF:\s+(\d+).?',
        'CIF:\s+(\d+).',
        'CIF:\s+(.+)',
        'CIF\s+(.+)',
    ),
}
```

Figura 4.4: Expresiones regulares de extracción de datos de facturas

Para intentar solventar este problema, primero se ha recurrido a incluir algunas expresiones regulares añadidas a las existentes en el prototipo inicial, para poder alcanzar más casos de diversas facturas. Como esta solución no ha sido suficiente para abarcar todo tipo de facturas, se recurrió a la simulación de datos como se explica en el siguiente apartado 4.4.2.

4.4.2 Simulación de facturas

Además de la problemática expuesta en el apartado anterior, se necesitaba remediar la escasez de facturas que había disponibles para hacer pruebas y continuar el desarrollo de la aplicación. Como solución se ha desarrollado un sistema de simulación de facturas. Este sistema se ha ido incrementando y mejorando a medida que iba creciendo la aplicación e iban apareciendo nuevos modelos de datos como se mostrará en las siguientes fases.

Como primer acercamiento, el sistema de simulación consistía en un pequeño programa que simulaba datos de clientes (**figura** 4.5) y facturas (**figura** 4.6).

```
Nclientes = 1000
Nfacturas = 50

nombres = open("nombres.txt").readlines()
apellidos = open("apellidos.txt").readlines()
comercializadoras = [line.replace('\n', '') for line in open("comercializadora.txt").readlines()]
distribuidoras = open("distribuidora.txt").readlines()
poblaciones = open("poblaciones.txt").readlines()
calles = open("calles.txt").readlines()

result={}

for c in range(Nclientes):
   result['Nombre'] = (random.choice(nombres)).replace('\n','')
   result['Apellido1'] = (random.choice(apellidos)).replace('\n','')
   result['Apellido2'] = (random.choice(apellidos)).replace('\n','')
   result['NIF'] = str(random.randint(10**7, 10**8-1)) + \
                   random.choice(string.ascii_uppercase)
   result['Direccion'] = (random.choice(calles)).replace('\n','')
   result['Comercializadora'] = random.choice(comercializadoras).split(";")[1]
   result['Distribuidora'] = (random.choice(distribuidoras).split(";")[2]).replace('\n','')
   poblacion = random.choice(poblaciones).split(";")
   result['CP'] = poblacion[2].replace('"','').replace('\n','')
   result['Poblacion'] = poblacion[1].replace('"','')
   result['Provincia'] = poblacion[0].replace('"','')
   result['CUPS'] = 'ES' + str(random.randint(10**15, 10**16-1))
```

Figura 4.5: Simulación de clientes

Cabe destacar que se han utilizado varios ficheros de texto que contienen los datos simulados como los nombres, apellidos, comercializadoras, distribuidoras y calles.

```
for f in range(Nfacturas):

   result['FechaInicio'] = \
      time.strftime(format, time.localtime(fechainicio))
   result['FechaFin'] = \
      time.strftime(format, time.localtime(fechainicio+2*month_time))
   result['FinContrato'] = \
      time.strftime(format, time.localtime(fechainicio+2*month_time))
   result['FechaEmision'] = \
      time.strftime(format, time.localtime(fechainicio+2*month_time+2*day_time))
   result['FechaCargo'] = \
      time.strftime(format, time.localtime(fechainicio+2*month_time+5*day_time))
   result['NumeroFactura'] = \
      'FI' + str(random.randint(10**9, 10**10-1))
```

Figura 4.6: Simulación de facturas

El bucle que crea las facturas está anidado dentro del bucle que crea a los clientes para que cada cliente tenga sus propias facturas asociadas.

4.4.3 Autenticación de usuarios

En este punto del desarrollo se necesitaba la funcionalidad de autenticación de los usuarios que accedían a la aplicación para gestionar sus facturas. Para llevar a cabo esta tarea, se han aplicado varias herramientas que se explican a continuación.

Primeramente, se ha utilizado la extensión *Flask-Login* [Cou12] que proporciona gestión de sesiones de usuario para aplicaciones *Flask*. Asimismo, maneja las tareas comunes como iniciar sesión, cerrar sesión y recordar las sesiones de usuarios durante períodos de tiempo prolongados.

En la **figura** 4.7 se muestra cómo se ha implementado la funcionalidad de iniciar sesión. Antes de nada, se comprueba si ya existe un usuario autenticado. Si es así, se redirecciona al usuario a la página de inicio. Si no hay ningún usuario logueado, se procede a validar los datos del formulario. En el caso de ocurrir algún error en las credenciales se le notificará al usuario con mensajes en rojo.

```
@auth_bp.route("/login", methods=["GET", "POST"])
def log_in():
  if current_user.is_authenticated:
    return redirect(url_for('public.index'))
  form = LoginForm()
  if form.validate_on_submit():
    user = User.get_by_username(form.username.data)
    if user is not None and user.check_password(form.password.data):
      login_user(user, remember=form.remember_me.data)
      next_page = request.args.get('next')
      if not next_page or url_parse(next_page).netloc != '':
        next_page = url_for('public.index')
      return redirect(next_page)
    else:
      flash("Las credenciales introducidas no son correctas.")
      return redirect(url_for("auth.log_in"))
  return render_template('login.html', form=form)
```

Figura 4.7: Ruta login

Para validar los datos introducidos por el usuario que pretende iniciar sesión, el sistema se apoya en otra extensión de *Flask* llamada *Flask-WTF* [Jac13]. Esta herramienta permite definir clases que contienen las reglas de validación de los formularios. Por ejemplo, en la **figura** 4.8 se observan los campos pertenecientes al formulario de login y sus respectivas reglas de validación.

Los formularios de registro de los clientes y las empresas se han implementado siguiendo los mismos pasos.

Por último, la extensión *Flask-SQLAlchemy* [Alc10] ha permitido manejar los registros de la base de datos a través de objetos. En esta etapa ha sido de utilidad para definir el modelo **User** que representa al usuario que quiere iniciar sesión. En la **figura** 4.9 se muestra parte de la definición del modelo comentado.

De la misma manera, se han definido en esta fase de desarrollo, los modelos de clientes y empresas, ya que los usuarios deben ser de uno de estos dos tipos. Lógicamente, también se necesitaban datos de empresas simuladas para seguir con el desarrollo. En este punto, se ha incrementado el programa de simulación para la creación de datos de empresas simuladas.

```
class LoginForm(FlaskForm):
    username = StringField(
        "Nombre de usuario",
        validators=[
            DataRequired(message="Debes introducir un nombre de usuario")
        ],
        render_kw={"placeholder": "Nombre de usuario"}
    )
    password = PasswordField(
        "Contraseña",
        validators=[DataRequired(
            message="Debes introducir una contraseña")],
        render_kw={"placeholder": "Contraseña"}
    )
    remember_me = BooleanField('Recuérdame')
    submit = SubmitField('Login')
```

Figura 4.8: Validación del formulario de login

```
class User(db.Model, UserMixin):

  __tablename__ = "users"

  id = db.Column(db.Integer, primary_key=True)
  username = db.Column(db.String(255), nullable=False, unique=True)
  password = db.Column(db.String(255), nullable=False)
  user_type = db.Column(TINYINT(), nullable=False)

  def set_password(self, password):
    self.password = generate_password_hash(password)

  def check_password(self, password):
    return check_password_hash(self.password, password)
```

Figura 4.9: Definición del modelo *User*

4.4.4 Sistema básico de estadísticas de consumo

En este punto, se ha implementado un sistema simple de visualización de gastos en el consumo de cada cliente por año.

En la **figura** 4.10 se observa el código de la ruta que obtiene las facturas de la base de datos. Primero se comprueba si el usuario que ha iniciado sesión es un cliente. Seguidamente, se obtienen todos sus contratos y las correspondientes facturas. Finalmente, se retornan los importes que ha pagado el cliente ordenados por año.

```python
@customer_bp.route("/get_stats")
@login_required
def get_stats():
  customer = None
  contracts = None
  contract_invoices = None
  if current_user.user_type == 1:
    customer = Customer.get_by_user_id(current_user.id)
    customers_dwellings_contracts = Customer_Dwelling_Contract.get_by_nif(customer.nif)
    contracts = []
    for customer_dwelling_contract in customers_dwellings_contracts:
      contracts.append(Contract.get_by_contract_number(customer_dwelling_contract.contract_number))
    contract_invoices = {}
    for contract in contracts:
      year = int(contract.init_date.strftime("%Y"))
      year_invoices = Invoice.get_by_contract_number(contract.contract_number)
      total_amounts = []
      for invoice in year_invoices:
        total_amounts.append(invoice.total_amount)
      contract_invoices[year] = total_amounts
  return contract_invoices
```

Figura 4.10: Ruta obtención de importes por consumo

Para la representación de los datos en gráficos se ha utilizado la biblioteca *Highcharts* [Tor09] como se observa en la **figura** 4.11. En la **figura** 4.12 se muestra un ejemplo de un gráfico de consumos de un cliente con facturas simuladas.

Hasta este punto, toda la interfaz ha sido diseñada simplemente con los recursos que proporciona *Flask* para el renderizado de plantillas con la biblioteca *Jinja* [Ron08]. Próximamente se mejorará utilizando la biblioteca *React*.

```javascript
$.ajax({
    url: '/get_stats',
    type: 'GET',
    success: function(response) {
        consumption_data = response;
        var year = Object.keys(response)[0];
        for (var y in response)
            $("#year").append('<option value="' + y + '">' + y + '</option>');
        options = {···
        myChart = Highcharts.chart(options);
    },
});
```

Figura 4.11: Petición de datos y configuración del gráfico de consumos

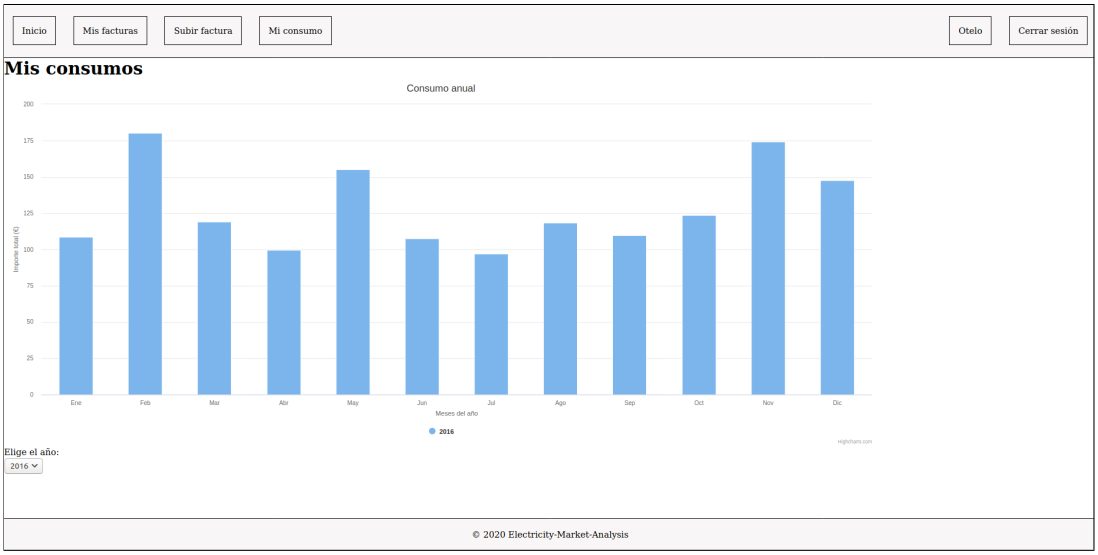

Figura 4.12: Apartado "Mis consumos" de un cliente

A parte de los consumos, los clientes también pueden consultar sus facturas a través de la aplicación como se muestra en la **figura** 4.13. Asimismo, pueden ver los detalles de cada factura pulsando el botón "Consultar" asociado a cada factura, como también pueden borrar facturas pulsando el botón "Eliminar". Con estas funciones se completa el CRUD de facturas.

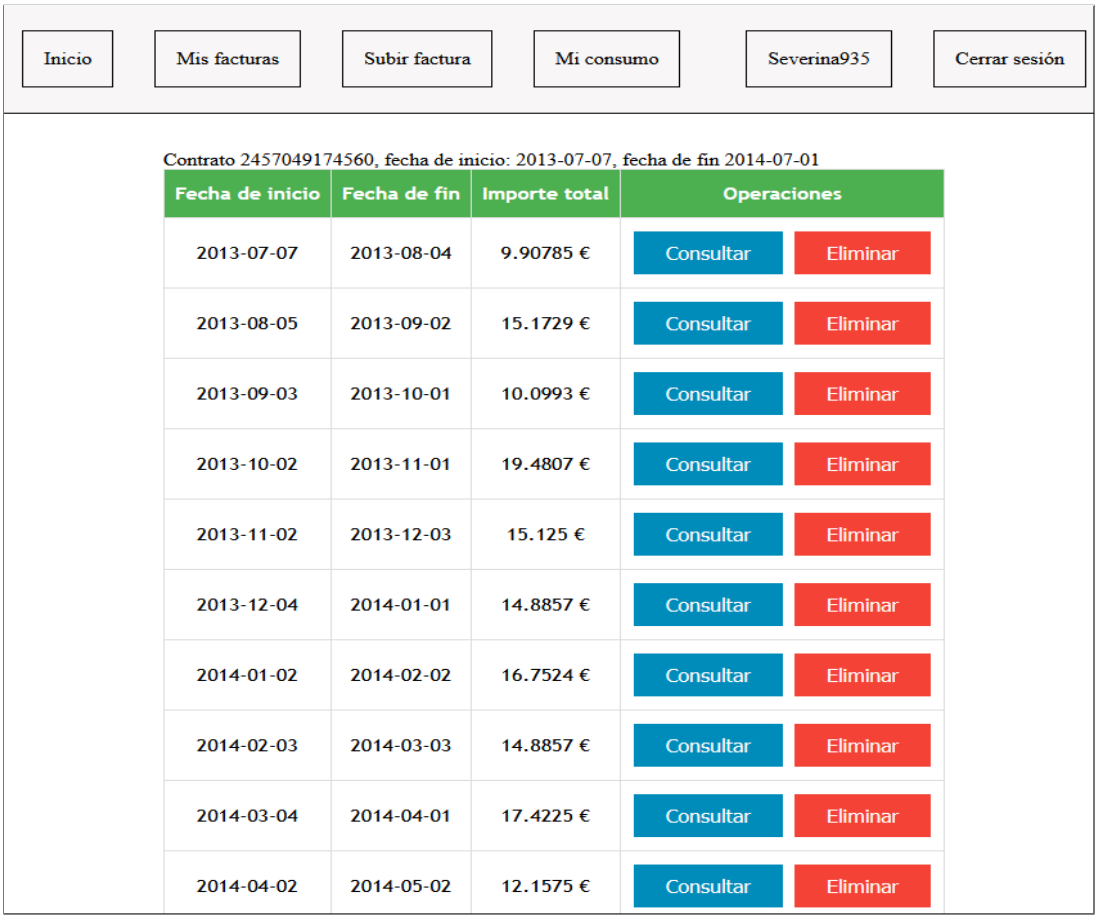

Figura 4.13: Apartado "Mis facturas" de un cliente

4.5 Versión 2 (Sprint 2)

4.5.1 Integración de *ReactJS*

A partir de esta fase, el *front end* de la aplicación pasa a ser gestionado por la biblioteca de *JavaScript* para creación de interfaces *React*. A su vez, el *back end* pasa a comportarse como una API y servir las peticiones que lleguen desde el *fron end*.

Antes de continuar con nuevas funcionalidades, se transforma el código de fases anteriores a código de *React*. Adicionalmente, se han aprovechado los componentes que ofrecen las bibliotecas *Bootstrap* y *MaterialUI* para aportar una interfaz gráfica más atractiva e intuitiva y con una mejor experiencia de usuario. Un ejemplo de ello podría hallarse en la página de inicio, algo más elaborada, donde se usan componentes *Carousel* y *Card*, como se observa en la **figura** 4.14.

Hay que destacar que la aplicación también es *responsive* consiguiendo adaptarse a muchos dispositivos y resoluciones.

Figura 4.14: Página de inicio

La vista de los contratos y las facturas de los clientes queda mejorada con *React* como se muestra en la **figura** 4.15. Ahora, tanto las facturas como los contratos están paginadas, lo que evita tener que desplazar la barra de desplazamiento en la pantalla para ver las últimas facturas. En añadido, se observa que actualmente los clientes pueden buscar cualquiera de los documentos filtrando tanto por mes como por año, lo cual hace mucho más sencilla su gestión.

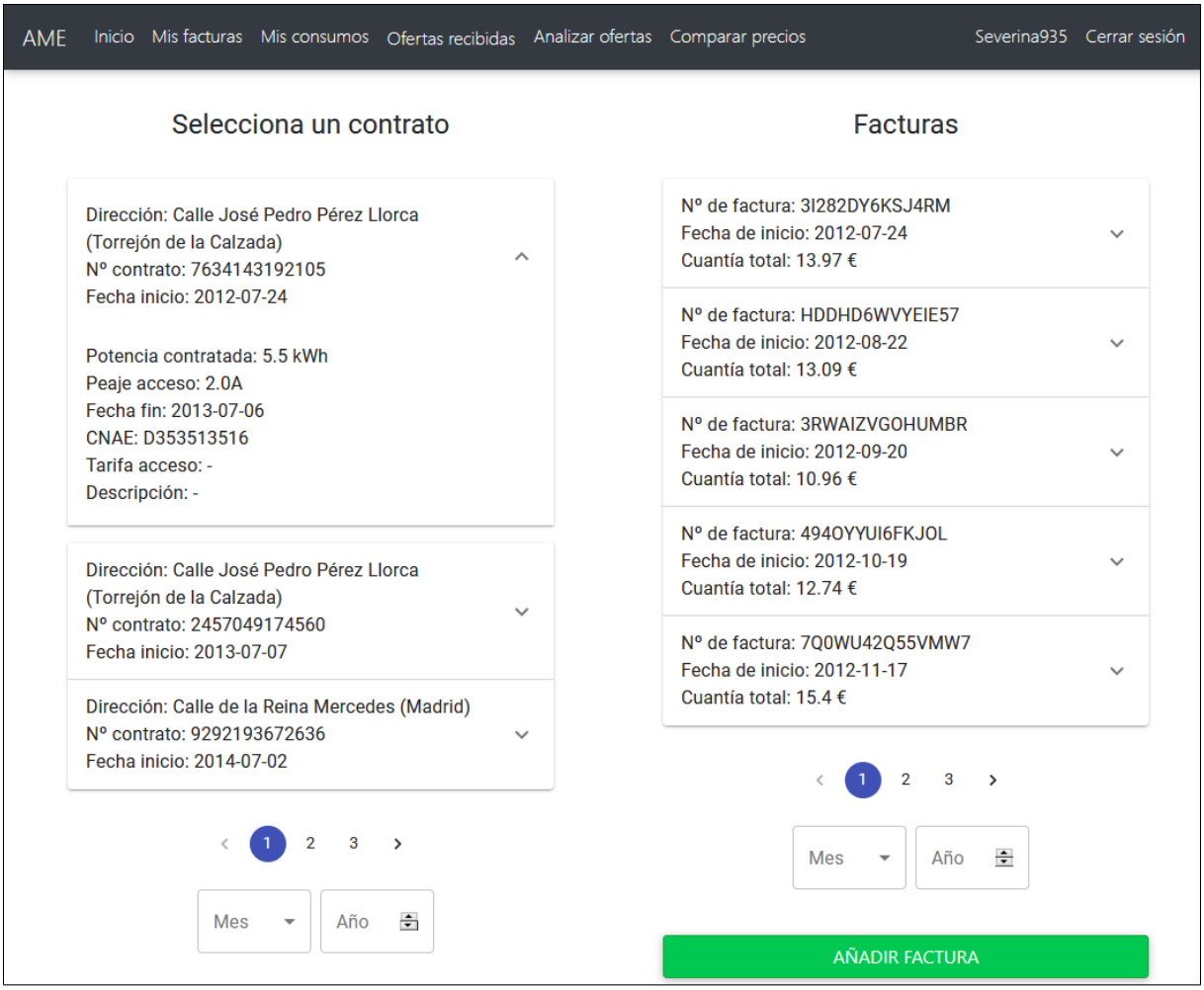

Figura 4.15: Vista de facturas y contratos de un cliente

Una funcionalidad destacable que posee *React* y que se ha empleado en el proyecto han sido los *Hooks*. Los *Hooks* son una nueva característica en *React* que permiten usar el estado y otras características de *React* sin escribir una clase, es decir, creando componentes funcionales. Seguidamente se explican dos de los *Hooks* más utilizados:

- ***useState***: este *Hook* permite a un componente guardar su estado. Por ejemplo, como se muestra en la **figura** 4.16, se declaran varias variables de estado usando *useState*. Las variables pueden ser de cualquier tipo, ya sean *Strings*, listas, objetos, etc. Estas variables permanecen ligadas al componente durante toda su vida. Además de la variable de estado, el *Hook* devuelve una función que sirve para modificar la variable de estado.

```
// CONTRACTS STATE
const [contracts, setContracts] = useState([]);
const [contractExpanded, setContractExpanded] = useState(false);
const [contractsList, setContractsList] = useState("");
```

Figura 4.16: Ejemplos de uso de *Hook useState*

- ***useEffect***: este *Hook* permite llevar a cabo efectos secundarios en componentes funcionales. Por ejemplo, en la **figura** 4.17 se usa el *Hook useEffect* para llamar a la función "*getContracts*" una vez que el componente se ha renderizado.

```
useEffect(() => {
  getContracts();
}, []);
```

Figura 4.17: Ejemplo de uso de *Hook useEffect*

Por otra parte, es necesario resaltar el uso de *React-Router* para la coordinación de las distintas rutas existentes en la aplicación. Se describen a continuación los componentes más importantes utilizados de esta herramienta:

- ***BrowserRouter***: este componente es el núcleo de toda aplicación que funcione con *React-Router*. Es el componente principal del que cuelgan todas las rutas.
- ***Matchers***: existen dos componentes de este tipo:
  - ***Switch***: cuando un componente *Switch* es renderizado, éste busca entre sus hijos (componentes de tipo *Route*) el que coincide con la url actual. Si encuentra alguna coincidencia, renderiza el componente *Route* encontrado. En el caso contrario no renderiza nada.
  - ***Route***: son los componentes hijos del componente Switch.
- ***Redirect***: si un elemento de este tipo es renderizado se navegará a una nueva ubicación.
- ***Link***: son componentes cuya función consiste en crear un link a la dirección que se le asigne. Cuando un componente *Link* es renderizado, en el html generado aparece una etiqueta <a >, es decir un *hyperlink* de html.

En la **figura** 4.18 se observa que el componente Router aparece como componente raíz, englobando a las tres partes principales que componen la estructura de la web, que son el encabezado (*Header*), el *Main* y el pie (*Footer*).

```
return (
  <div className="app">
    <Router>
      <Header
        username={username}
        userType={userType}
        companyType={companyType}
      />
      <div className="main-content">···
      <Footer />
    </Router>
    <Notify />
  </div>
);
```

Figura 4.18: Ubicación del componente *Router*

Dentro de la etiqueta *main* encontramos el componente *Switch* que contiene a su vez varios componentes *Route* con las distintas rutas existentes en la aplicación, como se presenta en la **figura** 4.19

```
<div className="main-content">
  <main>
    <Switch>
      <Route path="/" exact component={Home} />
      <Route path="/invoices" exact>
        {username ? <Invoices /> : <Login />}
      </Route>
      <Route path="/login" exact>
        {username ? <Redirect to="/" /> : <Login />}
      </Route>
```

Figura 4.19: Ejemplos de rutas

También, en esta parte del desarrollo se ha decidido mejorar el sistema de autenticación de usuarios usando JWT (*JSON Web Tokens*) [Mic15a].

JWT es un estándar abierto que define una forma compacta y autónoma de transmitir información de forma segura entre dos entidades a través de un objeto JSON (*JavaScript Object Notation*). Como la información transmitida está firmada digitalmente es confiable y se puede verificar.

Ahora se procede a la explicación de la estructura de los tokens:

- ***Header***: el encabezado generalmente consta de dos partes: el tipo de token, que es *JWT*, y el algoritmo *hash*, como pueden ser *HMAC SHA256* o *RSA*. En la **figura** 4.20 se puede observar un ejemplo.

Figura 4.20: Encabezado JWT

- ***Payload***: esta parte del token consiste en un objeto *JSON* que contiene la carga útil del token, por lo general suelen ser atributos del usuario. En la **figura** 4.21 se observa un ejemplo.

Figura 4.21: *Payload JWT*

- ***Signature***: para crear la firma del token, se deben tomar los otros dos campos codificados, el encabezado y el *payload* y pasarlos por un algoritmo *hash* junto con una clave secreta. Por ejemplo, si se utiliza el algoritmo *HMAC SHA256*, la firma se creará como se muestra en la **figura** 4.22

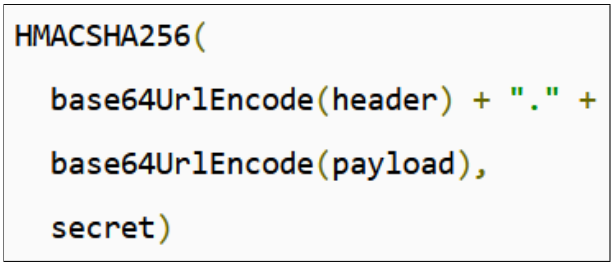

Figura 4.22: *Signature* JWT

Finalmente, al unir las tres partes del token, el resultado es parecido al que se presenta en la **figura** 4.23

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Figura 4.23: Token JWT completo

En la **figura** 4.24 se presenta el flujo de trabajo que sigue JWT. El usuario que pretende iniciar sesión, realiza una petición POST al servidor enviando en ella su nombre de usuario y contraseña. El servidor comprueba que las credenciales adjuntas son correctas y crea un nuevo token JWT que finalmente devuelve al cliente. A partir de este punto, el cliente podrá acceder a los recursos protegidos bajo autenticación enviando en cada petición el token recibido anteriormente. Cuando el servidor recibe una petición de algún recurso protegido, primero verifica el token y si es correcto envía al cliente la respuesta correspondiente con el recurso solicitado.

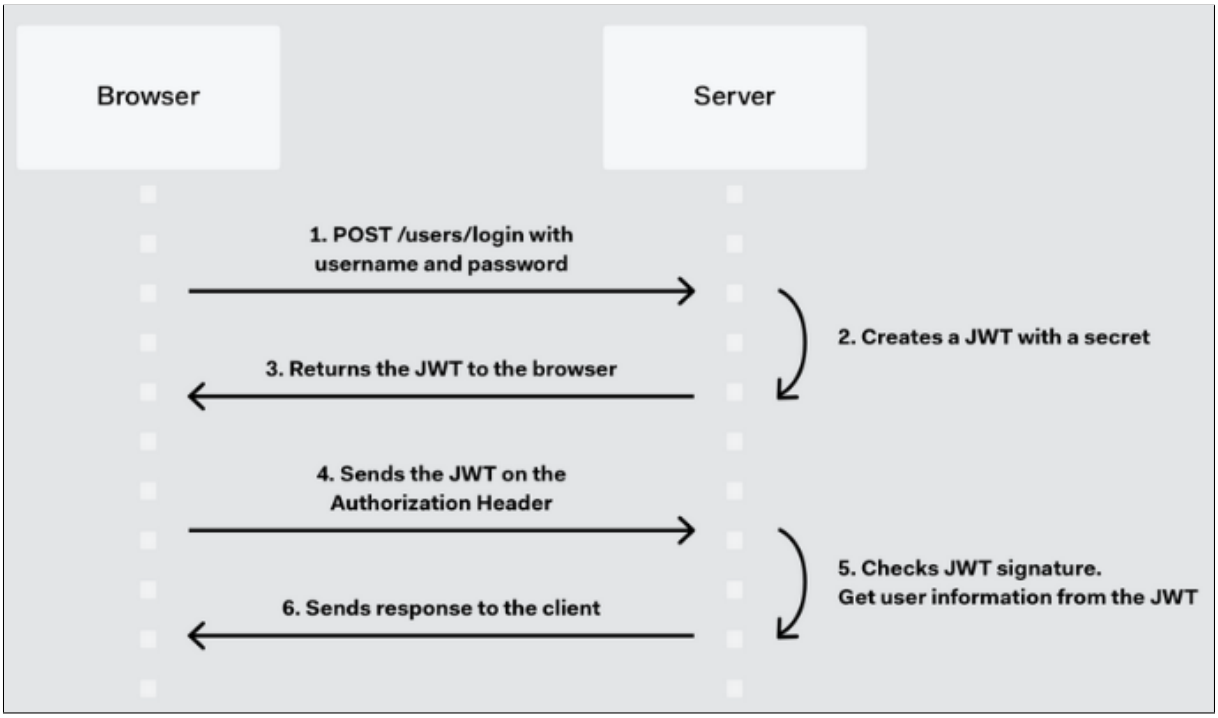

Figura 4.24: Flujo de trabajo JWT

Por último, se explican brevemente los conceptos básicos de *Redux*, el patrón que permite administrar el estado de las aplicaciones. En este proyecto se ha empleado *Redux* para almacenar la información relacionada con la autenticación de la aplicación además de la gestión de las notificaciones como se verá próximamente.

1. **Una única fuente de la verdad**: El estado global de la aplicación es almacenado en un único objeto. En la **figura** 4.25 se muestra el estado inicial de la authenticación de usuarios. Este es el estado que se mantiene cuando no hay ningún usuario logueado.

```
const initialState = {
  loggedUser: {
    username: false,
    userType: false,
    companyType: false
  },
};
```

Figura 4.25: Estado inicial de autenticación en *Redux*

2. **El estado es de sólo lectura**: La única manera de modificar el estado de *Redux* es a través de una acción como se observa en la **figura** 4.26. Una acción es un objeto plano que representa la intención de cambiar el estado. Se compone obligatoriamente de un campo *type* que indica el tipo de acción y otros campos opcionales con datos necesarios para la operación.

```
export const loginCustomer = (user) => {
  return {
    type: LOGIN_CUSTOMER,
    payload: user
  };
}
```

Figura 4.26: Acción de inicio de sesión de cliente

3. **Los cambios se realizan con funciones puras**: esto significa que los cambios en el estado de la aplicación suceden a través de funciones llamadas "*reducers*". Los *reducers* son funciones que reciben dos parámetros, el estado inicial y una acción. Dependiendo de la acción, realizarán una operación u otra modificando el estado. En la **figura** 4.27 se aprecia el *reducer* que controla la autenticación de usuarios en la aplicación.

```
export default function (state = initialState, action) {
  switch (action.type) {
    case LOGIN_CUSTOMER: {
      const { username, userType } = action.payload;
      return {
        loggedUser: {
          username: username,
          userType: userType,
          companyType: false
        }
      };
    }
    case LOGIN_COMPANY: {
      const { username, userType, companyType } = action.payload;
      return {
        loggedUser: {
          username: username,
          userType: userType,
          companyType: companyType
        }
      };
    }
    case LOGOUT_USER: {
      return initialState;
    }
    default:
      return state;
  }
}
```

Figura 4.27: Función *reducer* de autenticación

Para la gestión de las notificaciones se ha utilizado la dependencia ***react-redux-notify*** [Cha16] que funciona utilizando el estado de la aplicación para guardar las distintas notificaciones, exitosas o erróneas, que vayan sucediendo cuando el usuario realice operaciones en la web.

4.5.2 Mejora de sistema de estadísticas de consumo

Para concluir esta versión, se han añadido dos gráficos de estadísticas más al ya existente, además de la posibilidad de alternar entre estadísticas anuales y mensuales.

Por un lado, se ha creado un gráfico apilado de gastos mensuales, en el que se desglosa por partes los importes por tipo de gasto. En la **Figura** 4.28.

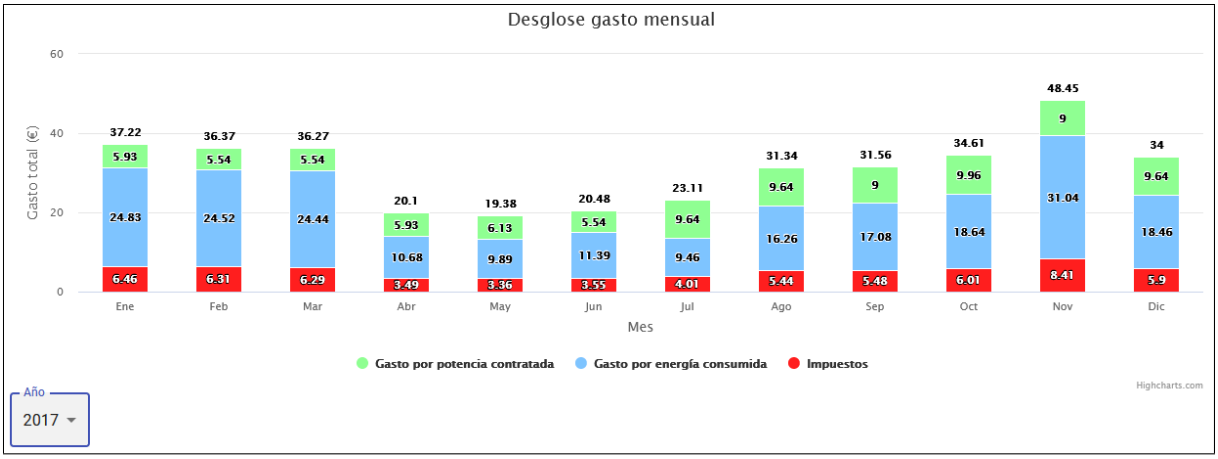

Figura 4.28: Gráfico de desglose de gasto mensual

Por otro lado, como se presenta en la **figura** 4.29 se ha añadido un gráfico de consumos, en esta ocasión con la configuración de gasto medio anual.

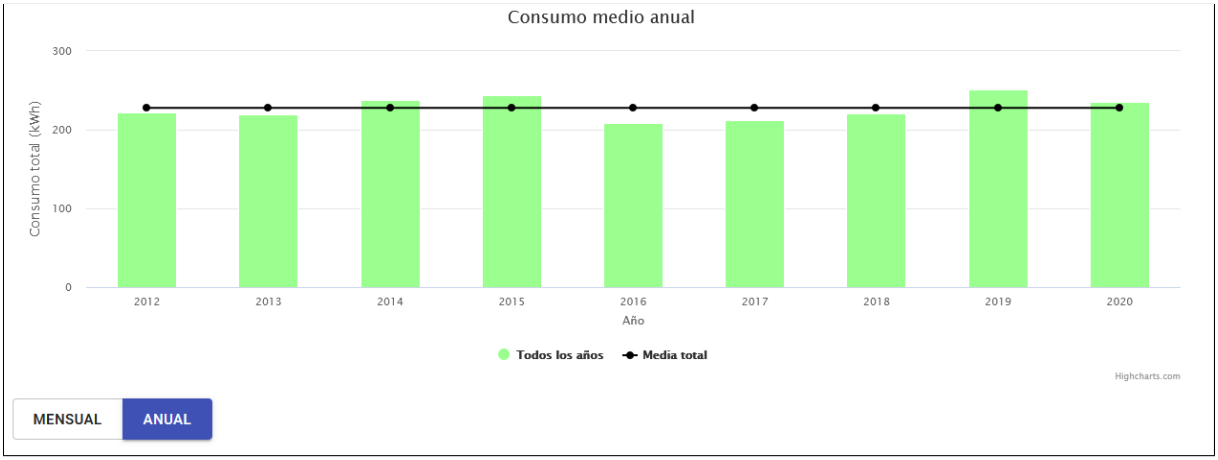

Figura 4.29: Gráfico de consumo medio anual

4.6 Versión 3 (Sprint 3)

4.6.1 RUD perfil de usuario

Como primera tarea de esta fase, se han implementado las funcionalidades de lectura, actualización y borrado del perfil de un usuario de la aplicación. En la **figura** 4.30 se muestra la vista del perfil de una empresa. También se ha desarrollado el homólogo de esta funcionalidad para los clientes.

El usuario puede modificar cada uno de los campos del formulario y pulsar el botón "GUARDAR" para persistir los datos. Si alguno de los campos no pasa la validación, se mostrarán en rojo con un mensaje de error.

Si el usuario desea eliminar su cuenta de la aplicación, puede pulsar el botón "ELIMINAR CUENTA" de color rojo que le abrirá un diálogo de confirmación. Si acepta el diálogo, este usuario y todos sus datos quedarán borrados de la aplicación y no podrá volver a iniciar sesión.

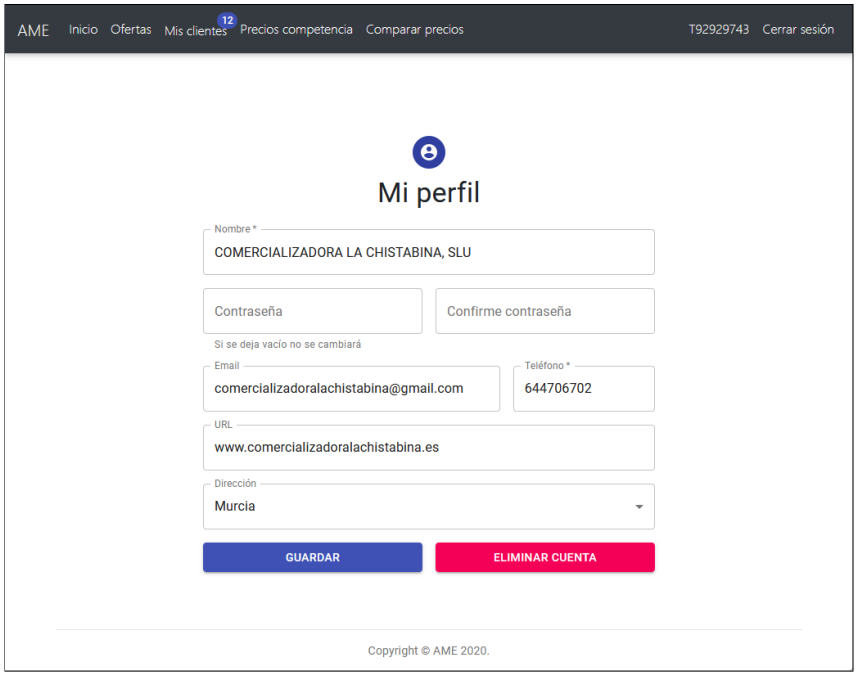

Figura 4.30: Perfil de usuario de empresas

4.6.2 CRUD ofertas

En este apartado se comenta resumidamente las funcionalidades de crear, leer, actualizar y eliminar ofertas por parte de las empresas.

En la **figura** 4.31 se presenta la vista general de las ofertas que tiene publicadas una comercializadora. La empresa puede pulsar el botón "CREAR OFERTA" si quiere publicar nuevas ofertas. Adicionalmente, la empresa tiene la opción de eliminar la oferta que desee pulsando el botón "ELIMINAR" que tiene asociado cada oferta.

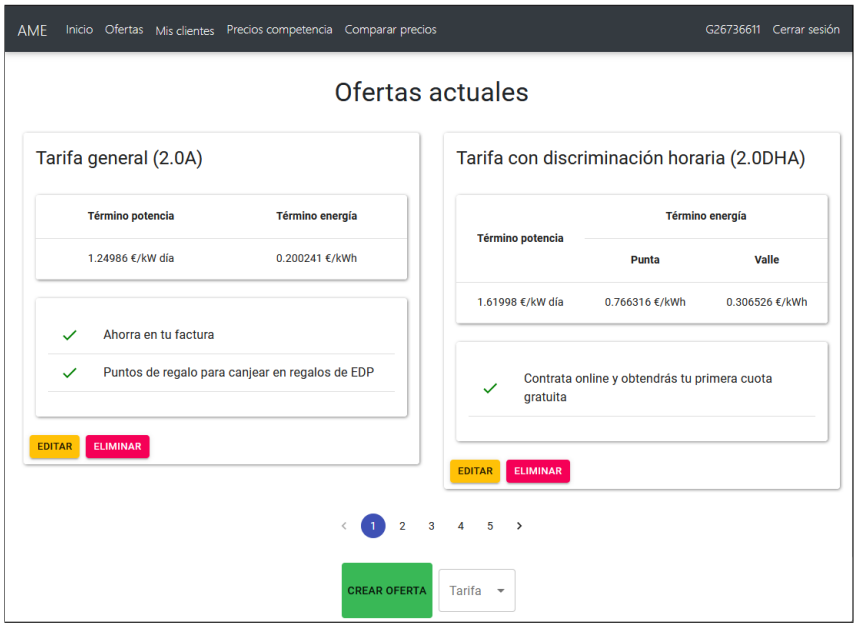

Figura 4.31: Vista de las ofertas de una comercializadora

De la misma manera, la comercializadora puede, pulsando el botón "EDITAR" de

cada oferta, modificar los parámetros de la oferta seleccionada, como se observa en la **figura** 4.32. En cualquier caso, si algún campo de la oferta es erróneo, se avisará con un mensaje en rojo.

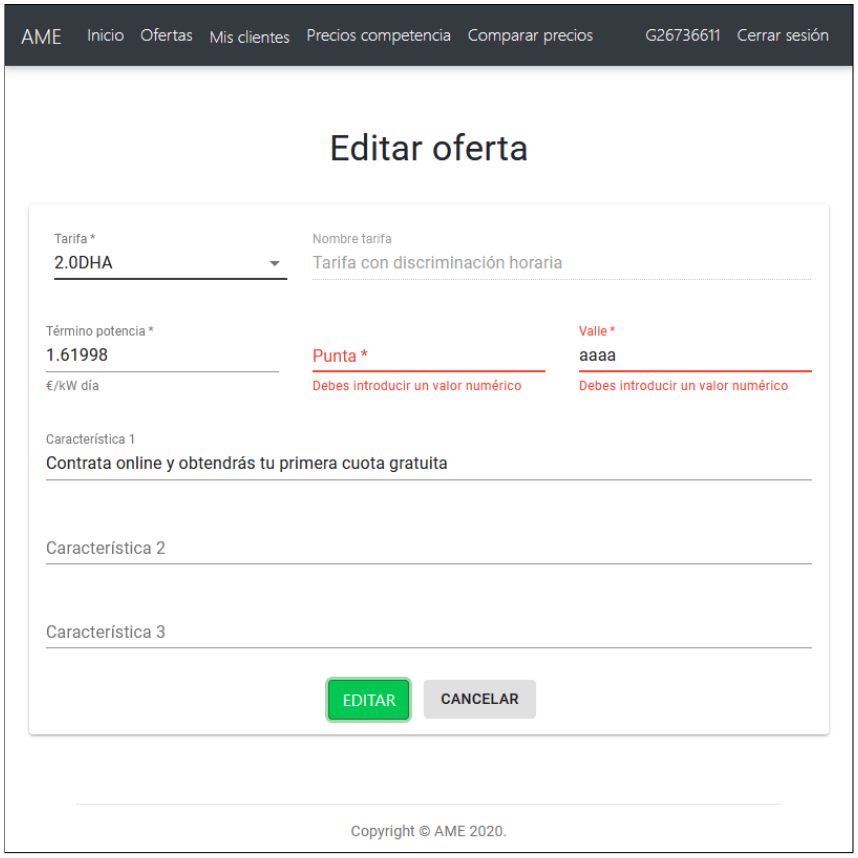

Figura 4.32: Vista de edición de una oferta

En cualquiera de las vistas, la empresa puede filtrar las ofertas por tipo de tarifa.

4.6.3 Clientes potenciales

Una de las características más destacadas de la aplicación es la notificación por parte del sistema a las empresas comercializadoras sobre clientes potenciales a los que poder enviar sus ofertas. En la **figura** 4.33 se muestra con un número en la barra de navegación, la cantidad de clientes potenciales que el sistema ha recomendado a esta comercializadora.

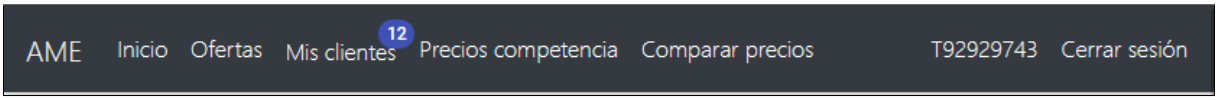

Figura 4.33: Notificaciones de clientes potenciales

Más abajo en la página, aparecen en una lista esos clientes potenciales notificados como se observa en la **figura** 4.34. Estos clientes podrán ser seleccionados para su posterior envío de ofertas, como se explica en el apartado 4.6.4. Las empresas también tienen la posibilidad de eliminar a los clientes que ya hayan sido tratados pulsando el botón "ELIMINAR".

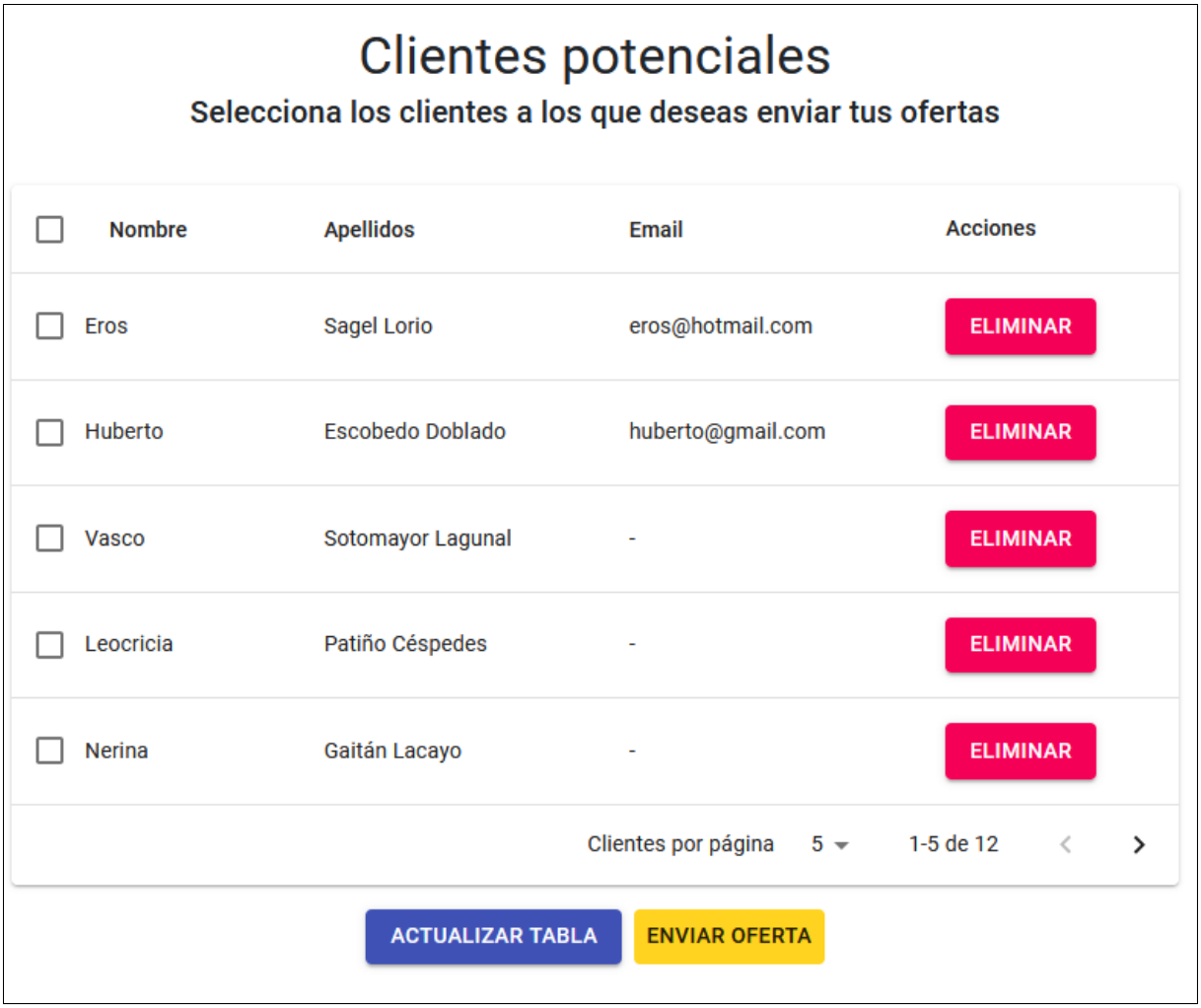

Figura 4.34: Lista de clientes potenciales

Hay que resaltar que cuando un cliente nuevo se registra en la aplicación, el sistema elige aleatoriamente varias empresas a las que enviarles la notificación de posible cliente potencial. Una posible mejora que se podría aplicar, sería la de notificar solamente a aquellas empresas que sean de la región del cliente que se acaba de registrar.

4.6.4 Enviar/Recibir ofertas

Tanto las empresas comercializadoras como los clientes pueden enviar y recibir ofertas respectivamente.

En la **figura** 4.35 se aprecia la vista que tiene una empresa después de haber elegido uno o varios clientes potenciales para enviarles sus ofertas. Pulsando el botón "ENVIAR OFERTA" la oferta elegida será enviada a los clientes seleccionados previamente.

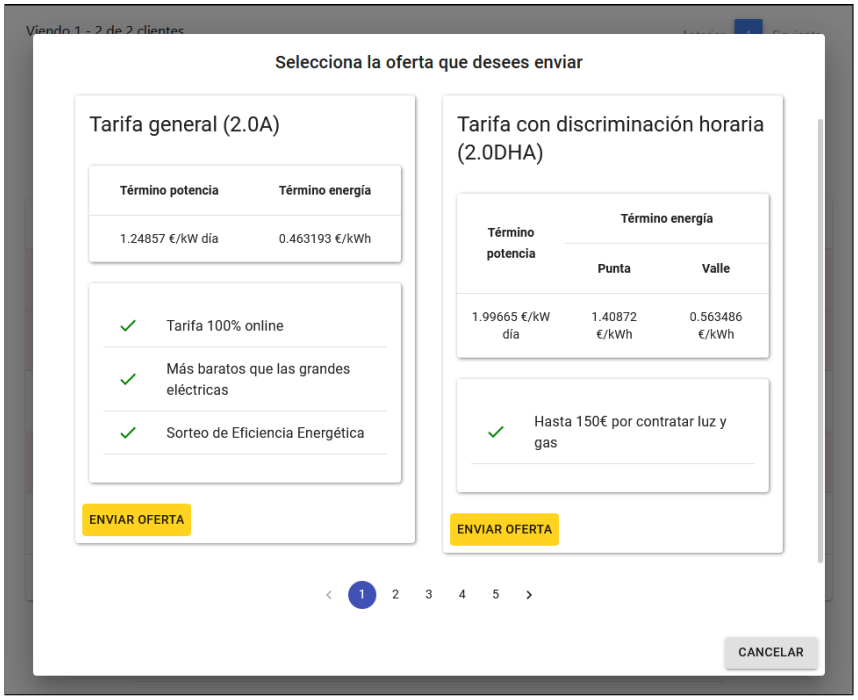

Figura 4.35: Funcionalidad envío de ofertas

En el lado del cliente, aparecerá una notificación en la barra de navegación con el número de ofertas que ha recibido. Seguidamente, debajo de la barra de navegación se encontrará con las ofertas recibidas en detalle, junto con la información de la comercializadora que las ha enviado, como se muestra en la **figura** 4.36

Figura 4.36: Vista de ofertas recibidas por un cliente

4.6.5 Consultar/Comparar ofertas

Tanto los clientes como las empresas pueden comparar ofertas. En el caso de los clientes, pueden buscar ofertas de dos compañías diferentes y comparar sus precios. En el caso de las empresas comercializadoras, compararán sus propias ofertas con las de su competencia. Esta funcionalidad queda representada en la **figura** 4.37.

Figura 4.37: Vista de comparativa de ofertas y precios

Se compararán los precios del mismo tipo de tarifa y se mostrarán en verde los más económicos y en rojo los más caros.

La funcionalidad de consultar precios es compartida tanto por los clientes y las empresas como por los usuarios invitado. Por lo tanto, para este caso se ha creado un solo componente llamado "*AnalyzeOffers*" compartido. La distinción radica en que las comercializadoras no deberían ver en este apartado sus propias ofertas, solamente las de su competencia, como se observa en la **figura** 4.38. A este componente se le pasa como atributo *props* (información que recibe un componente de sus componentes superiores) una variable para conseguir distinguir cuando se trata de una comercializadora y cuando no, en este caso "*companyType*", como se aprecia en la **figura** 4.39.

```
const getAllTradingCompanies = async () => {
  setCompaniesLoading(true);
  try {
    let response = []
    if (companyType === 0)
      response = await axios.get('/api/company/get-compentency-trading-companies');
    else
      response = await axios.get('/api/public/get-all-trading-companies');
```

Figura 4.38: Lógica del componente *AnalyzeOffers*

```
<Route path="/analyze-offers" exact>
  <AnalyzeOffers companyType={companyType} />
</Route>
```

Figura 4.39: Paso de atributo al componente AnalyzeOffers

4.7 Versión 4 (Sprint 4)

A parte de la funcionalidad de consultar ofertas comentada en el apartado anterior 4.6.5, los usuarios invitado pueden realizar varias operaciones que se describen a continuación, además de iniciar sesión y registrarse.

4.7.1 Ver empresas por región

Los usuarios invitado pueden seleccionar en el menú de navegación el botón "Empresas por región" donde encontrarán la vista de la **figura** 4.40. En ella, encontrarán un mapa del territorio español separado por regiones. Cada región está coloreada según existan más o menos empresas registradas en la aplicación pertenecientes a esa región. Si el usuario selecciona alguna de las zonas del mapa, a su derecha o debajo del mapa en versiones móviles, aparecerá una lista paginada de las empresas que pertenecen a esa región.

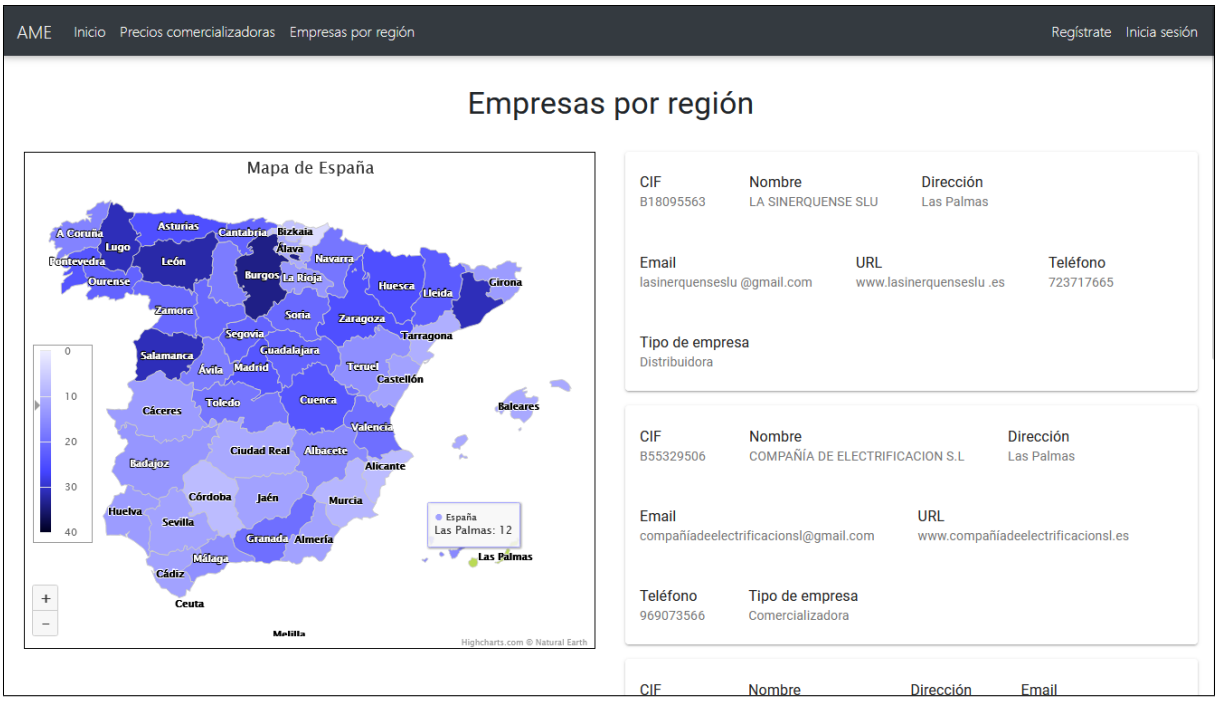

Figura 4.40: Vista de empresas por región

4.7.2 Comparativa de evolución de precios por región

Del mismo modo, cuando se selecciona una región en el mapa, aparecerá un gráfico que reflejará la media de precios anuales que han seguido cronológicamente en dicha región seleccionada. Adicionalmente, si se seleccionan varias regiones se pueden comparar la

evolución de precios durante el tiempo en las regiones seleccionadas como se aprecia en la **figura** 4.41.

Tanto el mapa de España como la gráfica de comparativa de precios por región han sido implementadas con *Highcharts*.

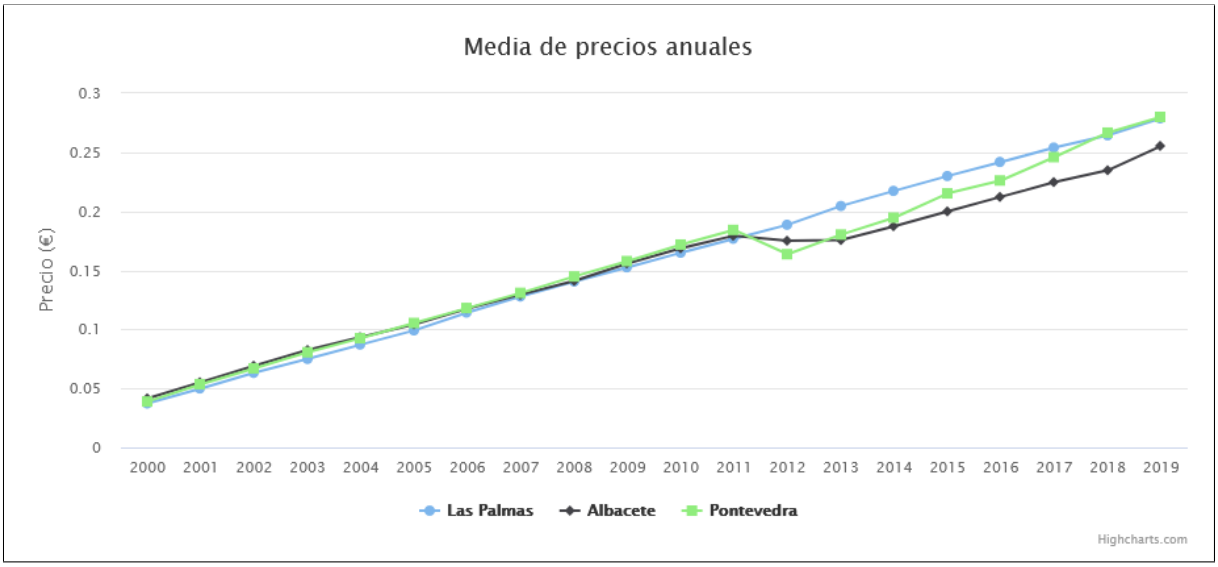

Figura 4.41: Gráfico comparativo de precios medios por regiones

Capítulo 5

Conclusión y trabajos futuros

Para finalizar, se analiza el resultado obtenido tras la elaboración del proyecto, así como el grado de consecución de los objetivos establecidos inicialmente.

5.1 Conclusión

En primer lugar, se ha conseguido construir una aplicación web pensada para intentar solucionar el problema de interpretación de las facturas de la luz, por parte de los clientes. Esto es posible gracias a la función que permite al cliente subir las facturas que posea a la plataforma, para posteriormente obtener un análisis automático de la información más relevante contenida en el documento. En consecuencia, el cliente consigue a través de un sistema de gráficos, una representación clara y precisa de parámetros tan importantes como los gastos y los consumos, que suelen ser los que más se consultan por la gran mayoría de los consumidores.

Por otro lado, esta aplicación supone una mejora considerable de la relación empresa-cliente en el sector de la energía eléctrica en España. Esto se consigue gracias a que el programa permite a las empresas enfocarse y llamar la atención de nuevos clientes potenciales enviándoles las ofertas que estimen oportunas. Por su parte, los clientes pueden recibir estas ofertas y valorarlas antes de una posible contratación. Además, los clientes tienen la posibilidad de buscar fácilmente a las comercializadoras y sus ofertas, sin tener que acceder a cada una de las páginas webs de cada empresa por separado.

Asimismo, tanto clientes como empresas pueden comparar los precios de las comercializadoras en función de la tarifa elegida. Esta herramienta es de gran ayuda para ambos, por un lado para ofrecer un mejor punto de vista a la hora de tomar una decisión de contratación para los clientes, y por otro lado para analizar los precios de la competencia en el caso de las comercializadoras.

Finalmente, los usuarios de esta aplicación tendrán la posibilidad de encontrar todas las empresas y su información, pertenecientes a una región del territorio español. De igual forma, podrán observar un gráfico cronológico de los precios fijados en la región que estén consultando. También les permitirá hacer una comparativa de los precios de diferentes regiones, lo cual puede resultar un conocimiento muy útil para los consumidores que están planteándose cambiar sus condiciones de contrato de la luz.

En definitiva, esta aplicación favorece la creación de un mercado en el que los clientes y empresas pueden relacionarse de forma clara y transparente.

A parte de los objetivos comentados, este proyecto me ha servido para aprender y practicar un nuevo marco de trabajo como *Flask*, para el diseño de aplicaciones web

del lado del servidor. Igualmente, he asimilado los conceptos básicos de *React* para crear interfaces de usuario más avanzadas. Además, ha servido de ensayo para poner en práctica metodologías de desarrollo ágil en un proyecto más cercano a un entorno laboral próximo. Finalmente, también ha sido de gran ayuda para afianzar conocimientos y mejorar en el uso del servicio *online Overleaf* y el sistema de composición de textos LaTeX, con los cuales se ha redactado esta memoria.

5.2 Trabajos futuros

Sin duda alguna, a pesar de haber conseguido en gran medida los objetivos previstos, siempre cabe la posibilidad de mejorar. Por ende, en este apartado se exponen algunas mejoras que podrían aplicarse al proyecto en un futuro.

Principalmente, se podría mejorar sustancialmente la funcionalidad de extracción de datos de las facturas que añaden los clientes, apoyándose en sistemas más eficaces como algún método que emplee redes neuronales.

Por otro lado, se podría enriquecer los datos históricos y evolutivos de precios de las distintas comercializadoras utilizando los datos de las facturas de los clientes, en vez de ser simulados.

Como las comercializadoras en principio dan más juego a este tipo de aplicación, se ha dejado en segundo plano a las empresas distribuidoras. Por lo tanto, una posibilidad de mejora sería implementar funcionalidades relacionadas con este tipo de empresa. Por ejemplo, la capacidad de que las distribuidoras puedan ver su lista de comercializadoras con las que tienen relación contractual y poder gestionar esta relación a través de la web.

También se podría plantear un sistema similar a una red social para comunicación entre clientes y empresas, además de un sistema de valoraciones de empresas por parte de los clientes.

Por último, en cuanto a las empresas comercializadoras, ofrecer la oportunidad de que puedan crear ofertas de varios tipos en función de su visibilidad de cara a los clientes. Por ejemplo, para los clientes nuevos quizás interese algún tipo de promoción u oferta que no corresponde a los clientes actuales. Por esta razón la empresa podría crear ofertas "privadas" destinadas a ciertos tipos de clientes.

Anexo A

Pila de producto

Id	Nombre	Descripción	Criterios de validación	Horas estimadas
1	Sign up	Como usuario quiero poder registrarme para poder usar la aplicación web como usuario identificado.	El usuario crea una cuenta y sus datos quedan guardados en la aplicación. La cuenta de usuario no se creará si falta por cumplimentar algún campo obligatorio del formulario de registro o si se introducen con algún error. Se mostrará un mensaje resaltando el error que se haya cometido.	12
2	Log in	Como usuario quiero poder iniciar sesión en la aplicación web para poder identificarme y acceder a mis datos personales.	El usuario puede entrar en su cuenta identificándose con su usuario y contraseña. Si las credenciales no son correctas se mostrará un mensaje indicandolo.	10
3	Log out	Como usuario quiero poder cerrar mi sesión en la aplicación web.	El usuario cierra su sesión y deja de tener acceso a sus datos. Pasaría a tener el rol de invitado (usuario no logueado) en la aplicación.	4
4	Ver perfil	Como usuario quiero poder visualizar mis datos personales para poder gestionarlos.	El usuario puede ver correctamente sus datos personales.	6

5	Editar perfil	Como usuario quiero poder modificar mis datos personales para poder subsanar errores o actualizarlos.	El usuario puede modificar los datos de su perfil con éxito. Si algún campo presenta algún error se mostrará en rojo y no se actualizarán los datos.	10
6	Borrar perfil	Como usuario quiero poder eliminar mi cuenta de la aplicación web.	El usuario pulsa el botón "ELIMINAR CUENTA" y se abre un diálogo de confirmación. Si el usuario acepta la confirmación no tiene posibilidad de acceder de nuevo a la aplicación con sus credenciales. Si se trata de un cliente, sus facturas y datos personales quedarán eliminados del sistema.	3
7	Sistema de notificaciones	Como usuario quiero poder recibir notificaciones para acciones exitosas y erróneas.	El usuario es informado con una notificación cuando realiza acciones exitosas o erróneas.	8
8	Recuperar contraseña	Como usuario quiero poder recuperar la contraseña de mi cuenta para poder acceder a la aplicación.	El usuario puede pulsar el link "recuperar contraseña" para solicitar un procedimiento de recuperación de su contraseña. El usuario recibe un correo a la dirección guardada en su perfil para iniciar el proceso de recuperación.	10
9	Ver facturas	Como cliente quiero poder ver mis facturas.	El cliente puede acceder a un registro de todas las facturas que ha ido subiendo a la plataforma. Si el cliente no tiene facturas el sistema mostrará un mensaje indicando que no tiene facturas actualmente. El cliente puede consultar los datos de cada factura individualmente.	6

10	Crear/subir facturas	Como cliente quiero poder subir nuevas facturas a la aplicación web para poder gestionarlas.	El cliente pulsa el botón de subir factura y selecciona la factura que desea subir desde el explorador de archivos del sistema. Los datos de la factura quedan guardados en la aplicación. El cliente puede acceder a la factura subida y consultar sus datos.	4
11	Actualizar facturas	Como cliente quiero poder actualizar una factura que existe actualmente en la aplicación por otra diferente.	El cliente selecciona de entre sus facturas aquella que desea cambiar y desde el explorador de archivos del sistema aquella factura nueva que desea subir. La factura actual desaparece y se guarda la nueva.	6
12	Eliminar facturas	Como cliente quiero poder borrar facturas existentes en mi registro de facturas.	El cliente busca la factura que desea borrar y pulsa el botón "Eliminar". Antes de eliminar la factura elegida se muestra un mensaje de confirmación. Si se acepta la confirmación la factura ya no es accesible desde la aplicación.	4
13	Analizar ofertas	Como cliente quiero poder analizar ofertas publicadas por empresas para poder compararlas posteriormente.	El cliente puede seleccionar una de las empresas registradas en la aplicación para analizar sus ofertas. El cliente puede visualizar las diferentes ofertas que la empresa seleccionada haya publicado en la aplicación.	10
14	Comparar precios	Como cliente quiero poder comparar ofertas y precios publicados por empresas para poder contrastar diferentes propuestas.	El cliente puede elegir dos comercializadoras para comparar sus ofertas. Una vez elegidas las comercializadoras el cliente podrá elegir una oferta de cada una y ver sus diferencias. Los precios de la misma categoría serán comparados y se mostrarán en verde los que sean más económicos y en rojo los más caros.	14

15	Visualizar gráficas	Como cliente quiero poder ver gráficas con los datos de mis facturas.	El cliente puede visualizar diversas gráficas que proporcionan información de forma ilustrativa y accesible de sus facturas. El cliente puede observar un gráfico de barras de sus consumos (kWh) tanto mensual como anualmente. El cliente puede observar un gráfico de barras de sus gastos (euros) tanto mensual como anualmente. El cliente puede observar un gráfico de barras apilado con un desglose de los gastos mensuales.	8
16	Recibir ofertas	Como cliente quiero poder recibir ofertas enviadas por empresas.	El cliente puede observar en la barra de navegación el número de ofertas recibidas por las empresas. El cliente puede pulsar en el botón de ofertas recibidas en la barra de navegación y visualizar el contenido de las ofertas además de la información de la empresa.	12
17	Consultar comercializadoras	Como cliente quiero poder acceder a la información empresarial de cada comercializadora.	El cliente puede ver exitosamente la información individual de cada comercializadora registrada en el sistema. El cliente puede buscar filtrando a través de un campo de texto por los campos de información de la comercializadora.	6
18	Mejora de condiciones	Como cliente quiero poder solicitar una mejora de condiciones a mi comercializadora actual.	El cliente, en la sección "MIS FACTURAS" puede pulsar el botón "MEJORAR CONDICIONES" para pedir una mejora de las condiciones de su contrato a su comercializadora. La comercializadora afectada recibe una notificación de su cliente para ponerse en contacto con él.	5

19	Valoraciones clientes	Como cliente quiero poder valorar a las comercializadoras.	El cliente puede buscar la comercializadora que quiere valorar. El cliente puede valorar a la empresa eligiendo entre 1 y 5 estrellas. El cliente puede añadir algún comentario a la valoración.	6.5
20	Descargar documento de factura	Como cliente quiero poder guardar el documento pdf o la imagen de la factura que subo, para poder recuperarlo más adelante.	El cliente puede guardar en el sistema el documento pdf o imagen cuando sube exitosamente una factura a la aplicación. El cliente puede pulsar en el botón "DESCARGAR DOCUMENTO" asociado a cada factura para descargar el documento pdf o imagen de su factura.	7
21	Ver precios competencia	Como empresa quiero poder ver precios publicados por otras compañías.	La empresa puede acceder y visualizar las ofertas y precios de otras compañías. Si se trata de una comercializadora las ofertas propias de la empresa no aparecen en esta lista, solo aparecen las ofertas de la competencia.	10
22	Comparar precios empresas	Como empresa comercializadora quiero poder comparar precios y ofertas de otras compañías para poder realizar un análisis del mercado.	La empresa puede elegir una comercializadora de su competencia para comparar sus ofertas. Una vez elegida la comercializadora la empresa podrá elegir una oferta de dicha empresa y otra suya propia y ver sus diferencias. Los precios de la misma categoría serán comparados y se mostrarán en verde los que sean más económicos y en rojo los más caros.	12
23	Recibir notificaciones potenciales clientes	Como empresa quiero poder obtener avisos de potenciales clientes para poder ofrecerles nuestro servicio.	La empresa recibe notificaciones del sistema en la barra de navegación cuando encuentre posibles nuevos clientes.	10

24	Crear ofertas	Como empresa quiero poder crear ofertas para poder ofrecerlas a mis clientes.	La empresa puede crear con éxito una oferta de su producto/servicio. Si algún campo de la oferta no es correcto, se mostrará un mensaje de error en dicho campo.	10
25	Ver ofertas	Como empresa quiero poder ver mis ofertas creadas.	La empresa puede visualizar satisfactoriamente sus ofertas creadas. La empresa puede filtrar sus ofertas por tipo de oferta.	3
26	Actualizar ofertas	Como empresa quiero poder modificar mis ofertas creadas.	La empresa puede actualizar exitosamente una oferta actualmente publicada en la aplicación. Si algún campo de la oferta no es correcto, se mostrará un mensaje de error en dicho campo.	6
27	Eliminar ofertas	Como empresa quiero poder eliminar mis ofertas creadas.	La empresa puede seleccionar y eliminar una oferta creada. Al pulsar el botón "ELIMINAR" aparecerá un diálogo de confirmación. Si la empresa acepta la confirmación la oferta en cuestión no aparecerá ni para la empresa ni para los clientes.	2
28	Enviar ofertas	Como empresa quiero poder mandar ofertas a clientes potenciales para poder informar a dicho cliente de nuestras promociones.	La empresa puede crear una oferta de su producto/servicio y puede seleccionar a los clientes a los que desea enviársela. Los clientes seleccionados reciben una notificación que les lleva a la oferta en cuestión.	16

29	Ver mis clientes	Como empresa quiero poder visualizar la información de mis clientes.	La empresa puede visualizar correctamente la información de los clientes con los que tiene alguna relación contractual. La empresa puede visualizar correctamente la información de posibles clientes potenciales a los que poder enviar ofertas.	6
30	Eliminar clientes potenciales	Como empresa quiero poder eliminar los clientes que el sistema me ha recomendado como potenciales.	La empresa en la vista de clientes potenciales puede pulsar el botón "ELIMINAR" en cada cliente potencial para eliminarlo. Al eliminarse un cliente potencial, desaparecerán sus datos de la lista de clientes potenciales además de la notificación de la barra de navegación. Al eliminarse un cliente potencial, la empresa no podrá enviarle ofertas.	2
31	Ver contratos de clientes	Como empresa quiero poder observar los contratos de mis clientes.	La comercializadora que tenga clientes puede pulsar el botón "CONTRATOS" asociado a cada cliente y visualizar una lista de sus contratos.	1.5
32	Ver facturas de clientes	Como empresa quiero poder observar las facturas de cada contrato de mis clientes.	La comercializadora que tenga clientes puede pulsar el botón "FACTURAS" asociado a cada cliente y visualizar una lista de sus facturas.	1.5
33	Mejorar condiciones clientes actuales	Como empresa quiero poder mejorar las condiciones de contratación de mis clientes actuales.	La comercializadora puede, en la vista de "MIS CLIENTES", pulsar el botón "MEJORAR CONDICIONES" asociado a cada cliente actual para enviar una solicitud de mejora de la calidad y el estado de su contrato.	3

34	Crear ofertas privadas	Como empresa quiero poder crear ofertas de tipo "privadas", las cuales no estarán visibles para el público. Estas ofertas pueden estar destinadas a ciertos clientes concretos.	En la vista de creación de ofertas, la comercializadora puede elegir el tipo de visibilidad que tendrá la oferta. La comercializadora puede elegir entre visibilidad "PÚBLICA" o "PRIVADA". Las ofertas públicas son visibles por todos los usuarios de la aplicación mientras que las ofertas privadas solo pueden ser visibles por aquellos clientes que la reciban.	6
35	Nuevos tipos de oferta	Como empresa quiero poder crear nuevos tipos de ofertas.	La comercializadora, en la sección "MIS OFERTAS" puede crear nuevos tipos de ofertas	4.5
36	Ver mis comercializadoras	Como empresa distribuidora quiero poder ver una lista de las comercializadoras con las cuales tengo relación contractual.	La empresa distribuidora puede pulsar el enlace "MIS COMERCIALIZADORAS" que se encuentra en el menú de navegación y acceder a un listado de sus comercializadoras. La empresa puede observar correctamente la información de cada comercializadora de la lista.	5
37	Ver empresas por región	Como usuario no logueado quiero poder ver cuáles son las comercializadoras y distribuidoras de cada región (incluida mi zona).	El usuario puede seleccionar una zona geográfica y visualizar correctamente las distintas empresas que pertenecen a la zona elegida.	8
38	Consultar comercializadoras	Como usuario no logueado quiero poder acceder a la información particular de cada comercializadora.	El usuario puede ver exitosamente la información individual de cada comercializadora. El usuario puede buscar filtrando a través de un campo de texto por los campos de información de la comercializadora.	4

39	Consultar precios de comercializadoras	Como usuario no logueado quiero poder informarme de los precios que ofrecen las comercializadoras.	El usuario puede seleccionar una de las empresas registradas en la aplicación para visualizar sus ofertas. El usuario puede visualizar las diferentes ofertas que la empresa seleccionada haya publicado en la aplicación.	6
40	Ver evolución de precios por tiempo	Como usuario no logueado quiero poder ver la evolución de los precios en el tiempo por zonas.	El usuario puede elegir una zona geográfica y ver en un orden cronológico la variación de precios que han tenido a lo largo del tiempo.	12
41	Funciones red social	Como usuario no logueado quiero poder realizar preguntas y ver respuestas online.	Como usuario puedo acceder a un listado de preguntas y respuestas frecuentes de otros usuarios y propias. Los demás usuarios pueden ver las preguntas formuladas y responderlas.	16
42	Comprender facturas	Como usuario no logueado quiero poder comprender los distintos elementos que componen las facturas de la luz.	Como usuario puedo acceder a una explicación a través de un pequeño tutorial o un documento explicativo de las partes de una factura.	14
43	Conocer la legislación vigente	Como usuario no logueado quiero poder informarme sobre la legislación actual en el entorno del mercado del consumo eléctrico.	Como usuario puedo tener acceso a las normas del marco regulatorio de la red eléctrica.	10
44	Comparar precios de comercializadoras en el tiempo	Como usuario no logueado quiero poder comparar los precios de varias comercializadoras a lo largo del tiempo.	El usuario no logueado puede elegir dos o más comercializadoras y ver la variación de precios que han tenido a lo largo del tiempo.	7

Tabla A.1: Pila de producto

Anexo B

Estructura de la base de datos

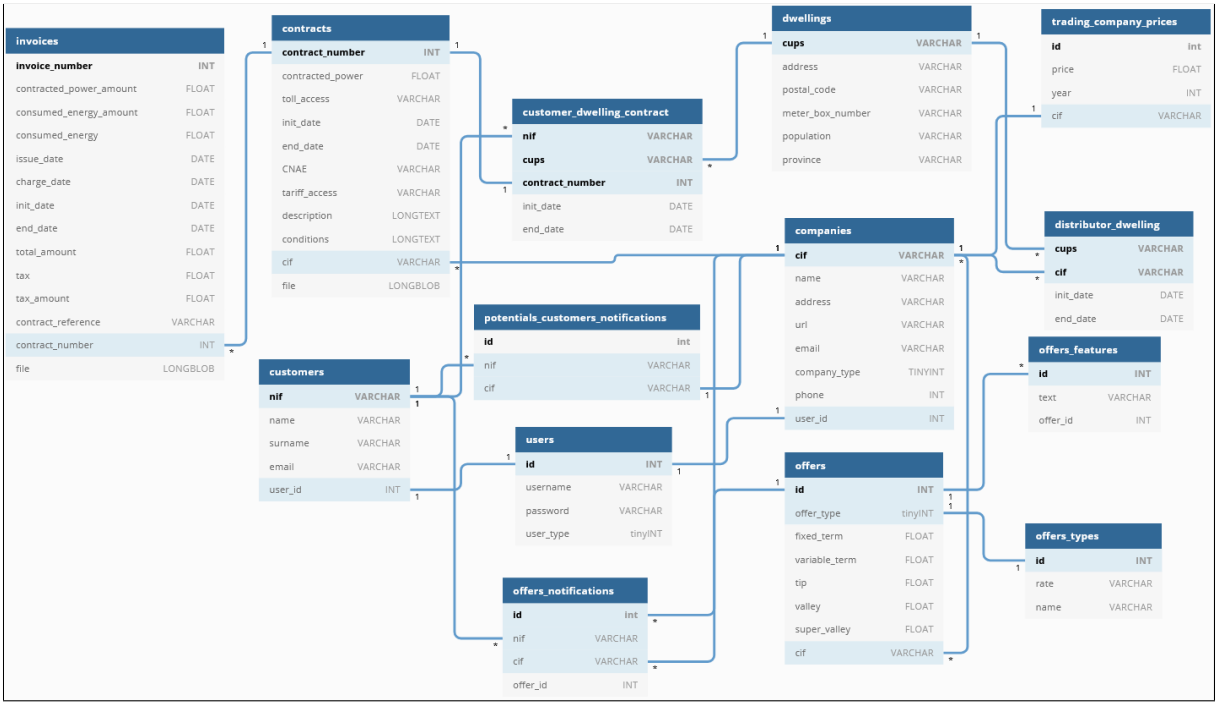

Figura B.1: Estructura de la base de datos

Anexo C

Estructura del proyecto

En este anexo se mostrarán las partes más relevantes de la estructura de carpetas creadas en el *back end* y en el *front end*.

C.1 *back end*

En la **figura** C.1 se observa la estructura de las carpetas del *back end* de la aplicación. Como se puede apreciar, todo el *back end* se encuentra dentro de la carpeta raíz *api*.

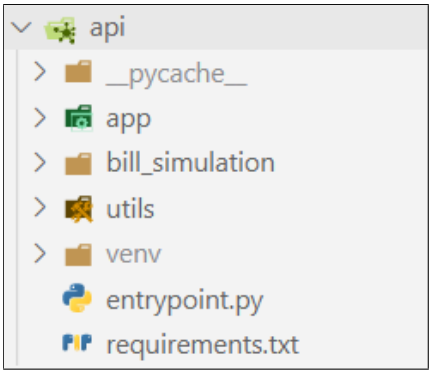

Figura C.1: Estructura del *back end*

Para empezar, en la carpeta *app* se encuentran los distintos *blueprints* de las diferentes funcionalidades de la aplicación como se explica más adelante.

Por otro lado, en la carpeta *bill_simulation* se encuentran los *scripts Python* para la simulación de facturas y otros datos de los modelos existentes como las empresas, clientes, ofertas, etc. El archivo *entrypoint* es el punto de inicio de la aplicación *Flask*. El fichero *requirements* contiene todas las dependencias necesarias para que funcione la aplicación.

En la **figura** C.2 se observa el contenido de la carpeta *app*. Dentro de ella se encuentran los distintos *blueprints*. Los *blueprints* son una forma que ofrece *Flask* para organizar el código en módulos. Estas son las partes en las que se ha dividido la *API*:

- ***auth***: que gestiona el sistema de autenticación de usuarios.
- ***customers***: que gestiona la lógica relacionada con los clientes.
- ***companies***: que maneja la lógica relacionada con los clientes.

- ***public***: que administra la lógica de las funcionalidades que no requieren autenticación.

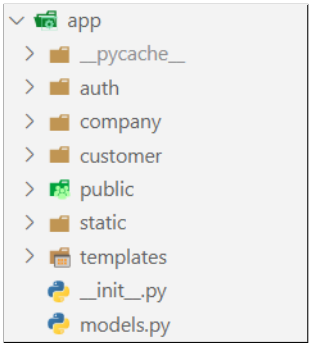

Figura C.2: Estructura de la carpeta app

Dentro de cada *blueprint* se encuentran los ficheros *Python* correspondientes a las rutas, modelos, esquemas, plantillas y formularios si fueran necesarios, además del fichero principal, *__init__.py* de cada módulo, como se observa en la **figura** C.3.

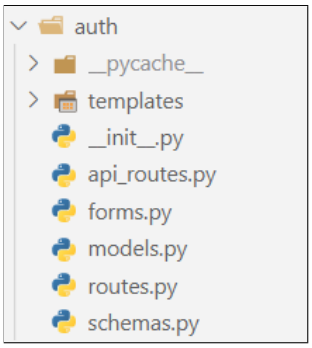

Figura C.3: Estructura de un *blueprint*

C.2 *Front end*

En la **figura** C.4 se aprecia la configuración de carpetas del *front end*, incluyendo la carpeta *api* que se analizó en el apartado anterior. Dentro de la carpeta *public* se observa el fichero principal de la web, *index.html*. En la carpeta *src* se encuentra el código con los componentes de *React* que se analizará en profundidad a continuación.

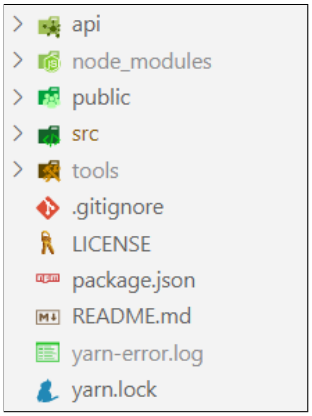

Figura C.4: Estructura del *front end*

La **figura** C.5 refleja la carpeta *src* en la que se distinguen tres directorios importantes.

Primero, la carpeta *components* que contiene componentes generales de toda la aplicación, como el componente *App* que mantiene la jerarquía de rutas de la aplicación. También se encuentran los componentes *Header* y *Footer*, los cuales están presentes en todas las páginas.

Por otro lado, en la carpeta *features* se localizan los componentes que forman las distintas funcionalidades de la aplicación. Por ejemplo, dentro de la carpeta *authentication* se ubican los componentes *Login*, *CustomerSignUp* y *CompanySignUp* que permiten el inicio de sesión y el registro respectivamente de los usuarios de la aplicación.

Finalmente, en la carpeta *redux* se hallan las carpetas que contienen las acciones, constantes y los *reducers* utilizados para su funcionamiento.

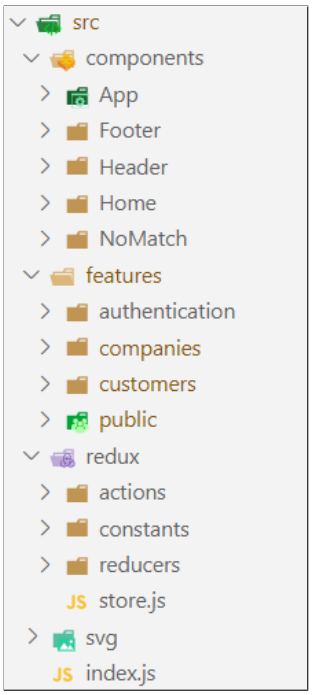

Figura C.5: Estructura de la carpeta *src*

Anexo D

Puesta en marcha del proyecto en local

En este anexo se explicarán los pasos a seguir para poner en funcionamiento la aplicación en local. Aclarar que este proyecto ha sido desarrollado en el sistema operativo *Ubuntu* y es posible que su ejecución en otro sistema operativo pueda causar errores y por consiguiente un mal funcionamiento de la aplicación. Antes de comenzar, hay que asegurarse de tener instalado en el sistema el siguiente software:

- **NodeJS** [Dah09]
- **Python**: versión 3.6.9 o superior [Ros91]
- **Virtualenv**: [Gáb]
- **MySQL** [MyS95]
- **Tesseract-OCR**: asegurarse de descargar e instalar también, el paquete de *Tesseract* correspondiente al idioma español que permitirá la correcta interpretación de la información de las factuas. [Smi07]
- **Yarn**: [Faca] en este proyecto se ha utilizado Yarn como administrador de paquetes pero se pueden emplear otros como por ejemplo npm.

Antes que nada, se debe clonar el código del repositorio de *GitHub* [Mar20] en la máquina donde se va a ejecutar.

D.1 *Back end*

Se comenzará a preparar e iniciar el *back end* siguiendo los pasos descritos a continuación:

- Abrir una consola y situarla en la carpeta *api* del proyecto.
- Crear un entorno virtual ejecutando:

  ```
  $ virtualenv venv
  ```

- Activar el entorno virtual ejecutando:

  ```
  $ source venv/bin/activate
  ```

- Instalar las dependencias del back end ejecutando:

  ```
  $ pip3 install -r requirements.txt
  ```

- Seguidamente se procede a crear la base de datos e inicializarla con datos simulados (debe estar en funcionamiento el servidor de bases de datos *MySQL* previamente instalado). Para crear la base de datos, se utilizarán los programas contenidos en la carpeta *bill_simulation*:

  - Primero, para crear la base de datos *electricity_market_analysis* ejecutar:

    ```
    $ python bill_simulation/reset_database.py
    ```

  - En segundo lugar, para insertar datos de empresas y ofertas simuladas ejecutar:

    ```
    $ python bill_simulation/init_db.py
    ```

  - Finalmente, para crear facturas y clientes simulados, ejecutar:

    ```
    $ python bill_simulation/create_bills.py numero_clientes
    ```

    donde numero_clientes es la cantidad de clientes con facturas simuladas que se desean crear.

- Llegados a este punto se puede arrancar el servidor *back end* ejecutando:

  ```
  $ flask run
  ```

D.2 *Front end*

Para iniciar el *front end* se deben seguir los siguientes pasos:

- Abrir una consola y situarla en el directorio raíz del proyecto.
- Para instalar las dependencias del *front end* recogidas en el fichero *package.json* ejecutar:

  ```
  $ yarn install
  ```

- Para iniciar el servidor *front end* ejecutar:

  ```
  $ yarn start
  ```

- Finalmente navegar a la ruta *localhost:3000* en algún navegador para comenzar a usar la aplicación.

Para iniciar sesión con algún usuario simulado, tanto empresa como cliente, basta con buscar en la tabla *users* de la base de datos *electricity_market_analysis* el nombre de usuario de algún registro. Tener en cuenta que la contraseña para estos usuarios simulados, por motivos de simplicidad, es la misma que el nombre de usuario. Por ejemplo:

Nombre de usuario: Jesús935

Contraseña: Jesús935

Para las comercializadoras y distribuidoras se ha empleado el número cif de la empresa como nombre de usuario, ya que los nombres eran demasiado largos para hacer pruebas manuales.

Bibliografía

[Alc10] Flask SQL Alchemy. 2010. URL: https://flask-sqlalchemy.palletsprojects.com/en/2.x/.

[Ale19] Juan Palacio Alexander Menzinsky Gertrudis López. *Scrum Master*. 2019. URL: https://www.scrummanager.net/files/scrum_manager.pdf.

[Ast] Abhinav Asthana. URL: https://www.postman.com/.

[Atl] Gitflow Atlassian. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow.

[Axo14] Axosoft. 2014. URL: https://www.gitkraken.com/.

[Bay06] Michael Bayer. 2006. URL: https://www.sqlalchemy.org/.

[Cha16] Deeptesh Chagan. 2016. URL: https://github.com/deep-c/react-redux-notify.

[Con] Endesa Imagen Consumos. URL: https://selectra.es/energia/companias/endesa/clientes.

[Cou12] Max Countryman. 2012. URL: https://flask-login.readthedocs.io/en/latest/.

[Dah09] Ryan Lienhart Dahl. 2009. URL: https://nodejs.org/es/.

[Dan15] Andrew Clark Dan Abramov. 2015. URL: https://react-redux.js.org/.

[Eic95] Brendan Eich. 1995. URL: https://es.wikipedia.org/wiki/JavaScript.

[End] Endesa. URL: https://www.endesa.com/.

[Ext] Flask JWT Extended. URL: https://flask-jwt-extended.readthedocs.io/en/stable/.

[Faca] Facebook. URL: https://yarnpkg.com/.

[Facb] Endesa Imagen Facturas. URL: https://tarifaluzhora.es/companias/endesa/area-clientes.

[Fra] Joel Francia. URL: https://www.scrum.org/resources/blog/que-es-scrum.

[Gáb] Bernát Gábor. URL: https://virtualenv.pypa.io/en/latest/.

[Gmb07] Innotek GmbH. 2007. URL: https://www.virtualbox.org.

[Goo14] Google. 2014. URL: https://material-ui.com/.

[Hol] Holistics. URL: https://dbdiagram.io/home.

[Ima] Naturgy Imagen. URL: https://selectra.es/energia/companias/naturgy/clientes.

[Jac13] Dan Jacob. 2013. URL: https://flask-wtf.readthedocs.io/en/stable/.

[Joh12] John Lees-Miller John Hammersley. 2012. URL: https://es.overleaf.com/.

[Kah03] Alec Kahney. 2003. URL: https://es.wikipedia.org/wiki/Tesseract_OCR.

[Lam84] Leslie Lamport. 1984. URL: https://www.latex-project.org/.

[Lóp01] Ramón López. 2001. URL: https://www.grupoase.net/.

[Mar11] Jacob Thornton Mark Otto. 2011. URL: https://getbootstrap.com/.

[Mar20] Samuel Guerra Marrero. 2020. URL: https://github.com/Sguerreroo/electricity-market-analysis.

[Mic15a] Nat Sakimura Michael B. Jones John Bradley. 2015. URL: https://auth0.com/learn/json-web-tokens/.

[Mic15b] Microsoft. 2015. URL: https://code.visualstudio.com/.

[MyS95] Oracle Corporation MySQL AB Sun Microsystems. 1995. URL: https://www.mysql.com/.

[Nat] Naturgy. URL: https://www.naturgy.es/hogar.

[Rea] Highcharts React. URL: https://github.com/highcharts/highcharts-react.

[Ron04] Armin Ronacher. 2004. URL: https://www.palletsprojects.com/p/flask/.

[Ron08] Armin Ronacher. 2008. URL: https://jinja.palletsprojects.com/en/2.11.x/.

[Ros91] Guido van Rossum. 1991. URL: https://www.python.org/.

[Smi07] Ray Smith. 2007. URL: https://github.com/tesseract-ocr/tesseract.

[Tar] Tarify. URL: https://tarify.es/luz.

[Tom08] P. J. Hyett Tom Preston-Werner Chris Wanstrath. 2008. URL: https://github.com/.

[Tor05] Linus Torvalds. 2005. URL: https://git-scm.com/.

[Tor09] Grethe Hjetland Torstein Hønsi. 2009. URL: https://www.highcharts.com/.

[Tra] React Training. URL: https://reactrouter.com/.

[Wal13] Jordan Walke. 2013. URL: https://reactjs.org/.

[Zab14] Matt Zabriskie. 2014. URL: https://github.com/axios/axios.