ULPGC
Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería Informática

# Sistema Web para la Evaluación de Métodos de Extracción de Información a partir de Facturas

Trabajo Fin de Grado

Grado en Ingeniería Informática

Alejandro Miranda López

Tutores

Javier Sánchez Pérez

Agustín Javier Salgado de la Nuez

Las Palmas de Gran Canaria
Junio de 2022

# Agradecimientos

*Agradezco el apoyo recibido durante estos años*

*A mis padres,*

*A mi hermano,*

*A mi pareja.*

*Y a mis tutores por darme la oportunidad de realizar este proyecto.*

# Índice general

# Índice de figuras

# Índice de tablas

# Resumen

El objetivo de este proyecto es el desarrollo de una infraestructura web que permita evaluar métodos de inteligencia artificial en el ámbito de la extracción de información a partir de datos semiestructurados. Se dispone de un conjunto de datos de 75.000 facturas de consumo eléctrico que servirá para entrenar los métodos que los usuarios desarrollen.

Una vez obtengan sus resultados, se podrán subir a la plataforma con el fin de evaluarlos y comparar el rendimiento que han obtenido con el resto de los métodos. Este sistema permitirá conocer el estado actual de las técnicas de inteligencia artificial que mejores resultados obtengan para este tipo de problemas.

Las tecnologías que se usarán para desarrollar el proyecto serán, principalmente, React y Python.

# Abstract

The aim of this project is to develop a web infrastructure for assessing the performance of artificial intelligence methods in the field of information extraction from semi-structured data. The web contains a dataset of 75.000 electricity invoices that will serve for training the methods that users develop.

Once they obtain their results, these can be uploaded to the platform in order to evaluate them and compare their performance with the rest of methods. This system will allow stablishing the state-of-the-art in artificial intelligence techniques that obtain the best results for this type of problem.

The technologies that will be used to develop the project will be, mainly, React and Python

# 1.Introducción

Dentro del ámbito de la investigación existen numerosas plataformas para la comparación de métodos de extracción de datos. Actualmente, la necesidad de dichas plataformas está en auge, pues el número de demanda para los algoritmos de extracción de datos es cada vez mayor.

Siendo esta la causa que motivó la elaboración de este proyecto se ha decido desarrollar una plataforma que ofrezca una base de datos de facturas de consumo eléctrico para el entrenamiento de métodos de procesamiento del lenguaje natural. El objetivo que se persigue es el de poder entrenar métodos que sean capaces de seleccionar la información requerida de las facturas, de manera automática y fiable. De esta forma, se permite a los grupos de investigación el acceso a este conjunto de datos con el propósito de que sus métodos sean entrenados. Al mismo tiempo, podrán disponer de una comparativa entre los usuarios de la plataforma, y también, conocer cómo se compara el método de la investigación con otras.

Con respecto al conjunto de datos que ofrece la plataforma, consiste en una recopilación de facturas eléctricas generadas mediante un proceso de simulación con datos reales obtenidos de fuentes oficiales. Este conjunto está dividido en dos partes, entrenamiento y evaluación. En el caso de los archivos de entrenamiento, se encuentran 6 clases distintas de facturas, *templates*, en la que cada carpeta posee 5.000 archivos, lo que conforma un total de 30.000. Además, a estos ficheros se les debe sumar que se encuentra un *JSON* por cada factura con los resultados esperados, para aportar más información al desarrollador. En cuanto a los archivos de evaluación, se encuentran 5.000 ficheros en cada una de las 9 carpetas, sumando un total de 45.000 archivos. En este caso, no existen archivos *JSON*, solo los *PDF* con las facturas.

La importancia que adquiere este proyecto es la de que, diversos usuarios sean capaces de acceder al conjunto de datos creado y puedan utilizarlo para sus investigaciones, además de, poder evaluar la efectividad de sus algoritmos de extracción. En referencia a lo anterior, se consigue que mejoren de manera global los métodos, lo que permitirá la formación de diversos sistemas de gran calidad y utilidad para el usuario final.

Para concebir esta idea, se debe tener en cuenta que la plataforma ha de ser sencilla al uso, accesible y apta para cualquier dispositivo con conectividad a internet. Es por ello, por lo que se ha desarrollado la página siguiendo los principios de diseño Responsive, donde se tienen en cuenta estos aspectos. Además, se ha empleado una semántica en HTML adecuada para que el sistema funcione de manera correcta. Las tecnologías utilizadas en el lado cliente, son React y TailwindCSS.

Uno de los factores más importantes de la plataforma es la evaluación de los métodos. Esta evaluación debe ser precisa y que tenga un buen rendimiento para que los usuarios puedan obtener un resultado rápido. Así, las tecnologías elegidas en el lado servidor, el encargado de ofrecer el soporte a la autenticación, evaluación, etc., son *Python* junto a *FastAPI*. La base de datos elegida fue *MongoDB*, la cual posee un gran rendimiento y grandes capacidades de escalabilidad.

Otro aspecto importante en los proyectos de ingeniería del software es la metodología, pues es la que establece las bases para la organización del trabajo y de su desarrollo. En este caso, se usa *SCRUM* para alcanzar una estrategia en la que sea posible tener una adaptabilidad máxima. En proyectos de desarrollo web, los sistemas son propensos a ser actualizados con el tiempo. Siguiendo esta metodología, se determinan las historias de usuario que definen, de manera sencilla, las características que debe poseer el sitio web de cara al usuario.

Por ende, se apostó y se logró el objetivo de desarrollar una plataforma significativa al usuario, en el que este podrá descargar el dataset ofrecido y entrenar su método de extracción de datos. Una vez consiga los resultados esperados, ejecutará el conjunto de evaluación para subirlos a la plataforma con el fin de obtener las puntuaciones de dicha evaluación e información del rendimiento del método, comparándolo al modelo base dado.

Otras plataformas cuentan con un proceso análogo, sin embargo, pocas se dedican a las facturas eléctricas, como es la particularidad de este proyecto. Si bien es cierto, existen plataformas similares, tal y como *Robust Reading Competiton* [1] que, en una de sus competiciones, ofrece un dataset de facturas. No obstante, esta competición ya no se encuentra activa, aunque los usuarios pueden seguir participando. Adicionalmente, se pueden encontrar plataformas como *Kaggle* [2] donde se halla una diversidad de competiciones, conjunto de datos, etc.

## 1.1 Objetivos del proyecto

Los objetivos que se han fijado para este proyecto son:

1. El desarrollo de una plataforma web que permita a los investigadores descargar el dataset de prueba y encontrar información de cómo está estructurado. Este se trata de un objetivo principal, pues sin esto, la plataforma no sería usada de manera correcta y perdería una gran oportunidad de servir como herramienta para las distintas investigaciones.
2. Los investigadores podrán subir sus resultados a la web y así compararlos con otros investigadores. Estos resultados pueden ser actualizados o cambiados. Al igual que una aplicación normal, las investigaciones también evolucionan con el tiempo, es por ello por lo que es importante que estos resultados que se obtienen puedan ser actualizados o incluso

eliminados en cualquier momento por el usuario, para así ofrecer una información de mayor calidad dentro de la plataforma.

3. Obtener una lista de los distintos métodos que hayan subido los investigadores a la plataforma. De esta manera se podrá visualizar y analizar los resultados obtenidos y usarlos con el fin de mejorar las líneas de investigación

Y como se verá en el apartado de desarrollo, estos fueron completados de manera exitosa e incluso el objetivo dos se puede considerar ampliado, pues, actualmente, se permite visualizar los resultados de cada método en detalle y de ciertas maneras.

## 1.2 Aportaciones

En el ámbito de las investigaciones, esta plataforma va a resultar de gran ayuda, pues permitirá a los investigadores conocer el estado y resultado de otros métodos y cómo los desarrollados por ellos se comparan a estos.

Esta tarea se podía realizar antes de la creación de la plataforma, pero requería tener los resultados de ante mano de los métodos a comparar o el código para obtenerlos. Ahora, al tener una plataforma única el trabajo se simplifica, por lo que aumentará la productividad de los investigadores y, por tanto, la calidad de sus investigaciones.

Además, estos resultados pueden ser modificados por el usuario en el momento que lo desee, por lo que facilita que se muestren lo más actualizados posibles.

Con respecto a los usuarios de la plataforma, estos se encontrarán con un diseño intuitivo y que puede ser visualizado sin perder ninguna funcionalidad desde cualquier dispositivo con un navegador relativamente moderno y con conexión a internet. Gracias a esto, se abre una posibilidad de que usuarios menos avanzados en el uso de las tecnologías, por ejemplo, investigadores procedentes de otras ramas, como estadistas que requieren de la extracción de datos, en concreto de las facturas eléctricas, que necesitan un método que posea una gran precisión para ellos poder realizar el estudio pertinente la plataforma podrá aportarle la búsqueda de manera sencilla.

Dicho lo cual, podemos concluir que la plataforma ayudará a diversos investigadores dentro del mismo campo y, de manera indirecta, a otras personas que necesiten de los servicios de esta.

## 1.3 Organización del documento

Tras esto, los apartados que se verán en el siguiente documento se encuentran organizados de la siguiente manera:

- En el capítulo 2 se verá el estado del arte en el que se encuentran otras plataformas similares a la creada en este trabajo y se realizará una breve explicación de estas.
- En el capítulo 3 se expondrá y se justificará la elección de los distintos recursos software utilizados, es decir, los lenguajes de programación, base de datos, etc.
- En el capítulo 4 se describen la metodología y la planificación empleadas en el proyecto. Es aquí donde se menciona la planificación inicial y la final.
- En el capítulo 5 se hablará sobre el conjunto de datos que la plataforma ofrece y los distintos campos dentro de cada fichero analizado además de la estructura de ficheros.
- En el capítulo 6 se explicará el desarrollo de la aplicación, es decir, el análisis, diseño, implementación y despliegue.
- Por último, se encuentra el capítulo 7 en el que se exponen las conclusiones obtenidas tras el desarrollo de este proyecto y los trabajos futuros planteados

# 2.Estado del arte

La extracción de datos a partir de documentos semiestructurados es una tarea tradicional en el campo del procesamiento del lenguaje natural. Sin embargo, no existe ninguna solución para este problema aún. Si bien es cierto, como se verá a continuación, que algunas competiciones internacionales han usado sistemas para la extracción de datos a partir de recibos, la cobertura de estas resulta limitado a un número pequeño de documentos con campos limitados.

Además, existen diversas plataformas para la comparación de métodos que, sin embargo, no se centran en la extracción de datos desde factura. Estos sitios web que se encuentran en la actualidad se centran en otros campos, como la visión para coches autónomos, o directamente no se centran en ningún campo en concreto, como es el caso de *IEEE Dataport*.

## 2.1 Plataformas similares

Es en este apartado donde se nombrarán algunas de las plataformas que existen en el mercado, que como se mencionó anteriormente, estas no se centran en la extracción de datos sino a otros campos, pero tienen, como plataforma, una finalidad similar que es la comparación de métodos.

### 2.1.1 Robust Reading Competition

*Robust Reading Competition* [1] es una plataforma encargada de actuar como anfitriona para diversas competiciones de visualización por computador. Como se ha ido viendo, aquí se pueden compartir y visualizar los resultados. De la misma manera, se incluye la posibilidad comparar distintos métodos para visualizar las diferencias y similitudes.

Dentro de esta plataforma se debería fijar la atención en la competición centrada en la extracción de datos de facturas, *SROIE* [3], pues tiene una finalidad similar a la propuesta por el sitio web desarrollado, aunque en este caso, el número de ejemplares dentro del dataset es menor.

Esta competición fue dividida en 3 tareas, la localización del texto, el escaneo del recibo para la lectura del texto y la extracción de información clave. Para conocer el rendimiento en estos tres escenarios, se utilizó las puntuaciones obtenidas mediante *Recall*, *Precision* y *Harmonic Mean*.

Una de las grandes ventajas de esta plataforma es que permite visualizar según el recibo que elijas, lo que permite conocer cómo rinde el método desarrollado en cada caso de una manera más precisa, además de conocer cómo rinden otros, pues esta información también aparece en dicha página.

### 2.1.2 MPI Sintel Flow Dataset

*MPI Sintel Flow Dataset* [4], de acuerdo con la descripción encontrada en su sitio web, ofrece un conjunto de datos que provee secuencias de vídeos que ofrecen un desafío a los métodos actuales (*Versión original en* [4]).

Esta plataforma se centra en el entrenamiento y evaluación de algoritmos para el cálculo de flujo óptico. Además, muestra un *ranking* con los resultados compartidos por los usuarios y permite visualizarlos en detalle.

### 2.1.3 KITTI Vision Benchmark Suite

*KITTI Vision Benchmark Suite* [5] es una plataforma desarrollada por diversas instituciones que expone varios *datasets* centrados en el entrenamiento y evaluación de métodos para la visión por computador de vehículos autónomos. Este conjunto de datos lo obtienen mediante las cámaras equipadas en diversos vehículos que obtienen imágenes de situaciones reales en la carretera.

Al igual que en la anterior, los usuarios tienen la posibilidad de compartir los resultados para que otros puedan verlos. También, estos se pueden visualizar de manera detallada.

## 2.2 Sistemas de evaluación genéricos

Adicionalmente a estas plataformas, existen otras que tienen un objetivo más general, como *IEEE Dataport* [6] que se trata de un sitio web que permite a investigadores y analistas el acceso a numerosos conjuntos de datos que los propios usuarios de la plataforma comparten.

De igual manera, esta plataforma sirve de anfitriona para competiciones para los miembros de esta o para los usuarios seleccionados, dependerá del organizador. Cabe mencionar que, *IEEE Dataport* es una de las principales plataformas a nivel científico para la realización de competiciones.

Este sitio web, posee distintas secciones para la visualización de los *datasets* y las competiciones donde se permite, en el caso de los *datasets* buscar por términos, filtrar por categoría o por el tipo de la base de datos. En el caso de las competiciones, estas opciones no están disponible. Al pulsar en uno de los enlaces de cualquiera de las dos listas, se accede a una sección donde se muestran todos los detalles de la competición o del método, sus autores, última actualización, licencia, etc.

También, se encuentra a *Kaggle* [2], otra plataforma enfocada a la inteligencia artificial y *machine learning*. Se puede considerar a esta como una de las grandes plataformas ya que posee competiciones que los usuarios crean, conjuntos de datos e incluso, fragmentos de código de distintas índoles. Estas funciones las consigue gracias a no centrarse en ningún campo específico como sí lo hacen las ya

mencionadas anteriormente y a la gran comunidad de usuarios que posee, que, según su propia página, posee aproximadamente "*9.7 millones de usuarios*".

Como se comenta en el párrafo anterior, *Kaggle* posee un gran número de funcionalidades, por ejemplo, si se accede a la sección de competiciones se podrá encontrar una lista de las competiciones que están en curso actualmente. También, se podrá buscar por el nombre, filtrar por el estado, el premio, la categoría, etc.

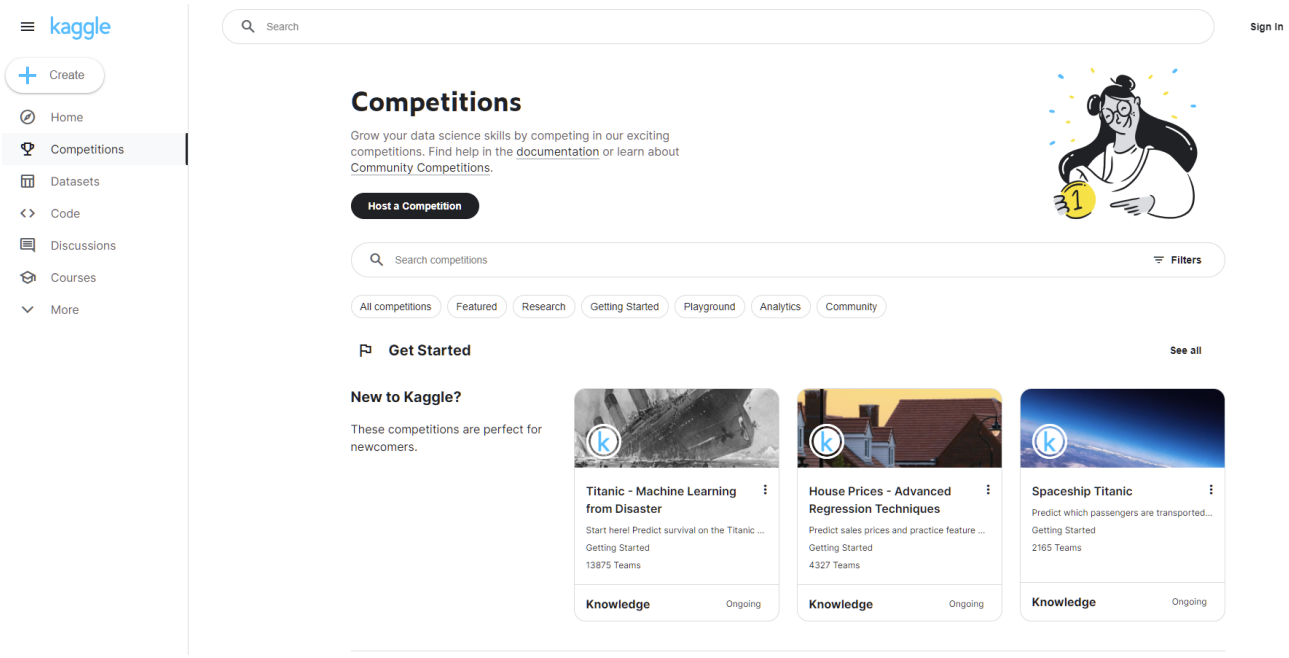


**Figura 1: Página de competiciones de Kaggle**

Asimismo, encontramos una página con un diseño similar para el listado de las bases de datos que distintos usuarios y organizaciones han compartido en la plataforma. Al igual que con las competiciones, estos conjuntos de datos también pueden ser filtrados, se puede buscar por el nombre y adicionalmente, se muestra una lista por más populares o los que más se usan actualmente.

Al igual que en el caso del *IEEE Dataport*, si se pulsa en los enlaces de la lista, se mostrarán los detalles de la competición o *dataset* de una completa y sencilla para el usuario. Adicionalmente, *Kaggle* ofrece visualizar algunos detalles adicionales en el caso de los conjuntos de datos, en concreto, permite ver la estructura y su contenido sin la necesidad de descargarlo lo cual puede suponer un gran ahorro de tiempo.

# 3.Recursos utilizados

A continuación, se expondrán todos los recursos que se han usado a lo largo del desarrollo de este trabajo. En concreto, se explicarán los lenguajes de programación, los *frameworks*, los entornos de desarrollo, la base de datos y otras tecnologías empleadas.

## 3.1 Lenguajes de programación

Para este proyecto se han utilizado únicamente dos lenguajes de programación: JavaScript y Python. Sin embargo, no se han usado estos leguajes de manera directa, sino mediante *frameworks* que están detallados en el siguiente apartado.

El primero, JavaScript es un lenguaje de programación basado en prototipos con soporte para programación orientada a objetos y de tipado débil [7]. Sin embargo, no se ha usado JavaScript puro para este proyecto, sino que se ha empleado un superconjunto de este lenguaje llamado TypeScript [8] que permite añadir tipado estático y objetos basados en clases, lo que proporciona una mayor facilidad y mejor seguridad a la hora del desarrollo ya que, evita problemas que se puedan encontrar en JavaScript como objetos con propiedades que no existen por el tipado. El código creado en TypeScript se compila a JavaScript, por lo que la compatibilidad es la misma. A continuación, se muestra una ilustración [9] que realiza una comparación entre estos:

- **Tipado estático, genérico, estructural y enumerados:** en JS no existen como tal los tipados, aunque los enumerados se pueden simular con clases sencillas. Mientras que TS es un lenguaje fuertemente tipado, donde se pueden crear tipos genéricos o interfaces.
- **Modularización:** TS ofrece un soporte directo para módulos, mientras que JS lo hace a través de ECMAScript 6.
- **Tuplas:** JS no las soporta, pero si TS.
- **Orientación a objetos:** la sintaxis de TS para la programación orientada a objetos es muy similar a la de otros lenguajes como Java o C#. Y además añade clases abstractas y modificadores de acceso, entre otras características. En JS también se puede programar orientado a objetos, pero es algo más complejo.
- **Decoradores:** JS no tiene soporte para decoradores, mientras que TS si.
- **Interfaces:** Como hemos mencionado, en TS las interfaces son imprescindibles, brindándote la posibilidad de crear escenarios más avanzados. En JS no existe soporte para interfaces.

**Figura 2: Comparativa TypeScript y JavaScript (Fuente: Profile [9])**

Asimismo, usamos Python, que es un lenguaje de programación interpretado, orientado a objetos y de tipado débil [10], para el lado cliente por la compatibilidad con diversas librerías de utilidades matemáticas utilizadas en el proyecto, como *Numpy* o *Scikit-learn*. Además, este lenguaje también es sumamente compatible con los sistemas ofreciendo un rendimiento aceptable y sin ningún tipo de ajuste adicional.

## 3.2 Frameworks y librerías

Para añadir una mayor funcionalidad a este proyecto, no se han usado los lenguajes de manera directa, sino que se han utilizado distintos recursos con la finalidad de obtener un código de mayor calidad.

### 3.2.1 Lado cliente

**React** [11] no es un *framework* como podría ser *Angular*, es una librería para el desarrollo de interfaces de usuario que fue desarrollada por *Meta Platforms* que fue lanzada en 2013 y es de código abierto.

Se ha elegido esta librería pues es una de las más utilizadas del mundo y posee una gran comunidad, además su implementación es bastante ligera en comparación con *frameworks* como Angular y permite el uso de *TypeScript* o la creación de *Single Page Application,* en otras palabras, una aplicación web que muestra todo su contenido en el navegador sin la necesidad de recargar la pestaña.

**JEST** es un *framework* para la realización de test unitarios en JavaScript [12] de código abierto y mantenido por *Facebook Inc.*

La elección de este *framework* se debe a que es el que viene incluido con la creación de un proyecto en React y es de los más usados, además de poseer todas las características que se necesitan en un proyecto del ámbito del realizado.

**TailwindCSS** es un *framework* de CSS que permite añadir estilos directamente mediante clases de HTML. [13]

La ventaja de este *framework* es que permite añadir estilos de manera sencilla y rápida, con medidas relativas (*rem*) y compatible con todos los navegadores principales y modernos. También, permite la reutilización de estilos utilizando el comando *@apply* incorporado por defecto en CSS.

### 3.2.2 Lado servidor

*FastAPI* [14] es un marco de trabajo de código abierto para la construcción de APIs haciendo uso de *Python.*

Este *framework* de Python permite la realización de APIs de alto rendimiento a la vez de una fácil implementación en comparación con otros como *Django* o *Flask*. Al usar Python, la compatibilidad con un gran número de librerías está asegurada.

Otra ventaja viene dada por no requerir de un gran número de archivos por lo que, en proyectos grandes, el mantenimiento resulta más sencillo y es menos propenso a los errores.

*Scikit-learn* [15] es una biblioteca para *Python* de código abierto para el aprendizaje automático. Esta biblioteca soporta un gran número de algoritmos y operaciones matemáticas. Está construido con base en *Numpy*.

Esta librería es la encargada de los diversos cálculos matemáticos realizados en la evaluación. Fue elegida mediante recomendación del tutor del presente proyecto.

Pytest es un marco de trabajo que permite la creación de pequeños test unitarios y ofrece la posibilidad de que sean escalables para soportar funciones de prueba en aplicaciones y librería con mayor complejidad [16]

Esta fue la librería usada para las pruebas del lado servidor, ya que es la librería por defecto de *FastAPI* y proporciona todas las funcionalidades necesarias.

## 3.3 Entornos de desarrollo

Se han usado dos entornos de desarrollo principalmente, *Visual Studio Code*, un editor de código desarrollado por *Microsoft* con un gran soporte para extensiones y numerosos lenguajes y *frameworks* [17], y *Pycharm*, un entorno de desarrollo integrado utilizado para el lenguaje de programación *Python* y sus diversas librerías [18]. El primero fue elegido por las utilidades y extensiones que ofrece para ayudar en el desarrollo de código para el *front-end* y para el *back-end*. Además, es gratuito por lo que se convierte en uno de los mejores editores de la actualidad.

A pesar de que *VSCode*, es compatible con Python, se prefirió optar por *Pycharm* como IDE para el desarrollo con este lenguaje pues ofrece algunas funcionalidades que el primero no, por ejemplo, un depurador más completo que permite ver los datos almacenados en variables de manera más completa y visual.

## 3.4 Herramientas

A continuación, se nombrarán las herramientas que se han utilizado a lo largo del proyecto.

Postman [19] es una herramienta que sirve para la realización de pruebas mediante solicitudes *HTTP* a una *API* que hayamos desarrollado. esta se ha empleado para las pruebas pertinentes de la API desarrollada para la plataforma, ya que permite probar los *endpoints* del servidor sin la necesidad de programar, lo que facilita la tarea de las pruebas.

*Compass* es un programa desarrollado por MongoDB [20] permite acceder y administrar los datos almacenados en la base de datos mediante una interfaz gráfica. Con esto, se posibilitó poder consultar los datos y comprobar que estos se guardaban de manera correcta en todo momento. Además, también permite crear documentos y colecciones.

## 3.5 Base de datos

La base de datos utilizada es MongoDB, una base de datos no SQL basada en documentos que guarda los datos en estructuras de datos BSON con un esquema dinámico [21]. Esta arquitectura ha hecho que MongoDB sea la base de datos utilizada en el proyecto ya que ofrece un gran rendimiento en búsquedas y permite ampliar los datos guardados en cada documento sin ningún problema, lo que, sin duda, aporta una gran flexibilidad de cara a actualizaciones futuras.

## 3.6 Sistema operativo

El sistema operativo usado es MacOS en su versión 12 Monterey debido a que es el ordenador personal del que se disponía para el desarrollo. Sin embargo, también se han usado sistemas con Windows 10 y Windows 11, ya que es el otro dispositivo en propiedad, aunque, por preferencias personales, el principal es el dispositivo con el sistema operativo de Mac.

Cabe mencionar que, aunque el desarrollo se haya llevado a cabo en estos sistemas operativos, gracias al uso de lenguajes multiplataforma, el código desarrollado se puede ejecutar en cualquier sistema operativo que posea compatibilidad con Python y JavaScript, así como las librerías que se usan a lo largo del proyecto. En el entorno de producción, la infraestructura se ejecuta en el navegador del usuario y se asegura la compatibilidad con las últimas versiones de Firefox, Safari y los navegadores basados en *Chromium* (*Google Chrome*, *Microsoft Edge*, etc.).

# 4.Planificación del proyecto

A continuación, se procederá a explicar cómo el desarrollo de este proyecto ha sido planificado y cuál es el coste de este.

## 4.1 Metodología

Siendo común en los proyectos de ingeniería informática modernos, la metodología usada es *SCRUM* [22]. Esta metodología aplica un conjunto de buenas prácticas con el fin de obtener el mejor resultado posible en un proyecto mediante el trabajo colaborativo. Además, se realizan entregas parciales y de manera regular, en períodos de no más de 1 mes (*Sprint*) con el fin de aportar valor de manera incremental al cliente. Esta manera de trabajar facilita la incorporación de cambios y de corrección de errores debido a que se trabaja mediante pequeños incrementos. De esta manera, *SCRUM* se trata de una metodología diseñada especialmente para entornos donde los requisitos están poco definidos y/o son cambiantes.

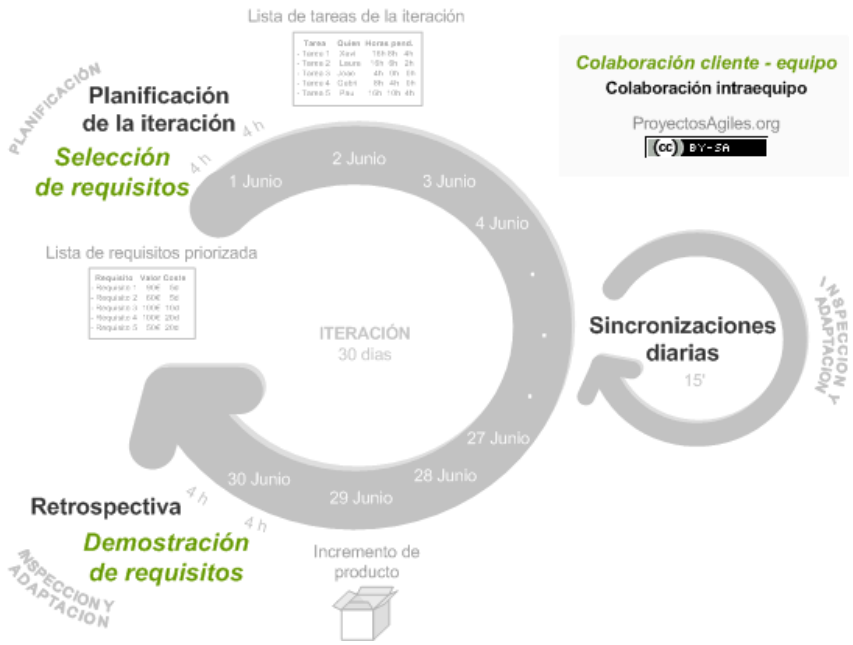


**Figura 3: Proceso de SCRUM (Fuente: ProyectosAgiles.org [22])**

Se definirán una serie de conceptos relacionados con *SCRUM*:

1. **Sprint:** *Sprint* es el nombre que reciben cada una de las iteraciones dentro de un proyecto *SCRUM*. De acuerdo con la definición oficial de la metodología, la duración de cada una de estas iteraciones es de como máximo de 1 mes en proyectos muy grandes.
2. **Pila de producto:** Es una lista ordenada donde se presentan los elementos que el *Product Owner* cree necesarios para la realización del proyecto [23].

3. **Product Owner:** Es el profesional encargado de definir las historias de usuario y establecer la prioridad de la pila de producto para agilizar la ejecución de las historias de usuario [24].
4. **Historias de usuario:** Es una explicación general e informal de una función de software escrita desde la perspectiva de un rol de usuario. La finalidad es añadir valor para el cliente [25].

Esta metodología es elegida por la flexibilidad a los cambios y su orientación a la entrega de producto funcional al final de cada iteración.

A pesar de que SCRUM está pensada para equipos de entre 3 y 9 miembros, puede utilizarse para casos de menor personal, como es el caso de este proyecto debido a su naturaleza. Si bien es cierto que, se pierden ciertas características como las reuniones diarias de corta duración, esto no supone ningún contratiempo y la utilización de *SCRUM* permite una mejor organización de las tareas a realizar y de la temporización del proyecto.

Las iteraciones de esta metodología se han definido de una longitud de 2 semanas, estas iteraciones son un artefacto de la metodología llamado *Sprint*. Al final de cada uno de estos, se realizaba una reunión con el tutor, que realizaba el papel de *Product Owner*, y se le mostraba el resultado de la iteración, se planteaban las historias de usuario a realizar para el siguiente *Sprint* y se revisaba la pila de producto por si era necesario añadir más historias a esta.

## 4.2 Planificación

El plan de trabajo inicialmente planeado se detalla en la siguiente tabla:

**Tabla 1: Planificación inicial**

| ***Fases*** | ***Duración Estimada*** ***(horas)*** | ***Tareas*** ***(nombre y descripción, obligatorio al menos una por fase)*** |
|---|---|---|
| Estudio previo / Análisis | 30 | Análisis de requisitos |
| | | Creación del Product Backlog |
| Diseño / Desarrollo / Implementación | 200 | Diseño de las funcionalidades del Backend (Endpoints) |
| | | Implementación de las operaciones con la base de datos |
| | | Creación de la funcionalidad de descarga del dataset |
| | | Creación del sistema de autenticación de usuarios |
| | | CRUD de los resultados por parte del usuario |

| | | |
|---|---|---|
| | | Implementación de la visualización de los resultados de otros usuarios |
| Evaluación / Validación / Prueba | 40 | Creación de tests para el Backend |
| | | Validación de las funciones en el lado cliente |
| Documentación / Presentación | 30 | Redacción de la memoria |
| | | Preparación de la presentación |

En la realidad, tras realizar el proyecto, se pudo comprobar de que la planificación no se ajustó por un margen mínimo, pero que es importante mencionar. El tiempo total dedicado a cada sección sería el siguiente:

**Tabla 2: Planificación final**

| ***Fases*** | ***Duración Estimada*** ***(horas)*** | ***Tareas*** ***(nombre y descripción, obligatorio al menos una por fase)*** |
|---|---|---|
| Estudio previo / Análisis | 30 | Análisis de requisitos |
| | | Creación del Product Backlog |
| Diseño / Desarrollo / Implementación | ~~200~~ 210 | Diseño de las funcionalidades del Backend (Endpoints) |
| | | Implementación de las operaciones con la base de datos |
| | | Creación de la funcionalidad de descarga del dataset |
| | | Creación del sistema de autenticación de usuarios |
| | | CRUD de los resultados por parte del usuario |
| | | Implementación de la visualización de los resultados de otros usuarios |
| Evaluación / Validación / Prueba | ~~40~~ 30 | Creación de tests para el Backend |
| | | Validación de las funciones en el lado cliente |
| Documentación / Presentación | 30 | Redacción de la memoria |
| | | Preparación de la presentación |

Cabe mencionar que el ajuste de 10 horas realizado se debe a que se implementaron ciertos aspectos que en un primer momento no fueron planeados. Además, la realización de los tests, gracias a las

librerías usadas y herramientas, fue más corta de lo prevista, lo que permitió este ajuste para la mejora de la calidad de la infraestructura.

## 4.3 Presupuesto

**Tabla 3: Presupuesto**

| | |
|---|---|
| **Hardware** | |
| MacBook Pro 13” 2019 | 1449€ |
| **Software** | |
| Pycharm IDE | 89€ |
| Visual Studio Code | 0€ |
| Base de datos MongoDB | 0€ |
| **Despliegue** | |
| Amazon EC2 [26] | 22,56€ |
| MongoDB Atlas [27] | 3,36€ |
| Dominio web | 14,95€ |
| **Personal** | |
| Desarrollador *full stack* | 32000€ |
| **Total Anual sin personal:** | **129,87€** |
| **Total Anual con personal:** | **32.129,87€** |

Establecer el presupuesto de una plataforma web como la desarrollada puede resultar complejo ya que muchos de los precios dependen de muchas variables.

Por un lado, existen una serie de costes fijos. Estos son las licencias para los programas de desarrollo, la base de datos, el dominio web y los servidores. En el caso actual, se supondrá un caso teórico en el que la aplicación se vuelve popular por lo que se necesitaría de un servidor más potente para la ejecución del *back end* y tener la base de datos en un *clúster* de alta disponibilidad y alto rendimiento como el que ofrece *MongoDB Atlas*. El servidor que sería elegido es *Amazon Web Service*. El precio de estas plataformas sería para el *EC2* de *Amazon* 22,56€ (0,099USD/GB, se suponen unos 20GB al mes) y para el servicio de *MongoDB*, 3,36€ (1 millón de lecturas al mes a razón de 0,3USD/1M).

Por otro lado, se debe tener en cuenta el personal. Para este proyecto, uno o dos desarrolladores con algo de experiencia sin llegar a nivel *senior* sería lo correcto. Además, se debe contar con un diseñador gráfico encargado del diseño de las interfaces y recursos digitales que el servicio pueda necesitar. Es aquí, en el apartado del personal, donde los precios podrán variar, aunque para tener una base común, los salarios del personal se obtendrán de la media nacional en España durante el último año [28] [29].

Dentro del lado del diseñador, solo se contrataría una vez para el diseño de la interfaz de usuario y los gráficos necesarios, el costo de este servicio sería de 840€.

Además, sería necesario el uso de un dispositivo para el desarrollador, en este caso se usó un *Macbook Pro 13"* de 2019 cuyo precio es de 1449€.

Esto nos dejaría un total de 129,87€ al año sin contar el personal, a lo que habría que sumar el primer año el coste del portátil y el diseñador gráfico. Si añadimos a la ecuación el costo del desarrollador, la cantidad asciende hasta los 32.129,87€.

# 5.Conjunto de datos de la plataforma

El conjunto de datos que ofrece la plataforma está diseñado para la extracción de información de archivos semiestructurados. Este contiene un gran número de archivos con diferentes estructuras e información que son generados de manera automática mediante un proceso de simulación.

En este proceso de generación, la lista de comercializadoras ha sido obtenida directamente de la *Comisión Nacional de los Mercados y la Competencia* (CNMC) y junto a esta las cantidades que se han introducido dentro de las facturas, para que exista un mayor realismo, fueron extraídas a través de las estadísticas creadas por el mismo organismo y la REE, *Red Eléctrica Española*.

El *dataset* se encuentra estructurado con dos carpetas, una de entrenamiento que contiene 30000 facturas en formato PDF con los correspondientes campos en JSON para que los usuarios puedan ver los datos que se deben en obtener y su formato. La otra carpeta es la de evaluación que contiene 45000 facturas en el mismo formato, PDF, y en la que se incluyen archivos que no se encuentran en el conjunto de pruebas. Esta distribución la podemos apreciar en la siguiente figura.

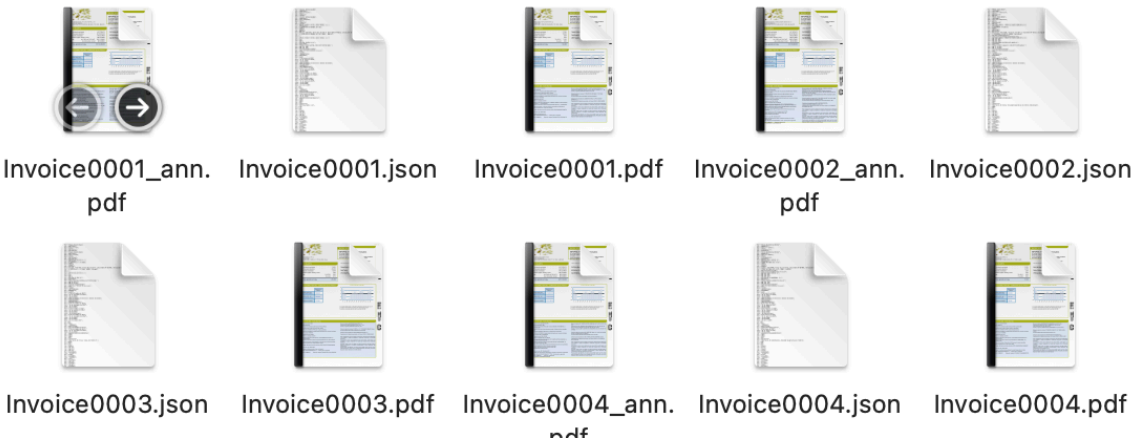


**Figura 4: Contenido de la carpeta de entrenamiento**

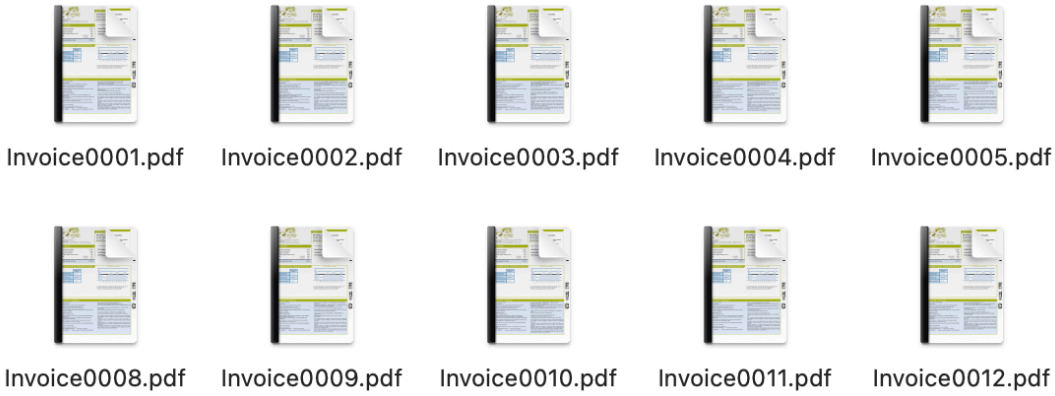


**Figura 5: Contenido de la carpeta de evaluación**

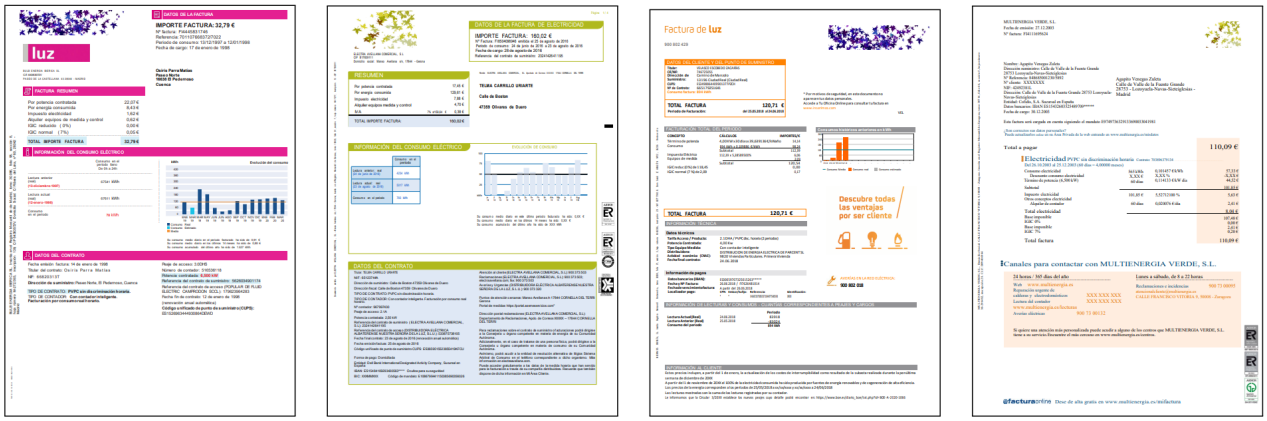

**Figura 6: Ejemplo de la distribución según la comercializadora**

Estas facturas del dataset tienen una estructura diferente entre todos los *templates*, haciendo uso del diseño que se usa distintas compañías. Cada factura encontraría sus apartados divididos en 12 categorías distintas, con varias divisiones en su interior, lo que resulta en que, los métodos que usen el dataset ofrecido deberán analizar un total de 86 campos por factura. A pesar de que este la distribución sea diferente, el contenido de estas es similar, en la primera página se resumirán los detalles esenciales y en las siguientes se muestran todos los detalles de la facturación.

**Tabla 4: Campos de las facturas del dataset**

| ID | Descripción | Rango |
|---|---|---|
| A | Cliente que recibe la factura | 1-6 |
| B | Datos del cliente reflejados en el contrato | 1-6 |
| C | Datos de la comercializadora | 1-E |
| D | Datos del distribuidor | 1-D |
| E | Información del contrato | 1-9 |
| F | Información general de la factura | 1-8 |
| G | Datos bancarios del cliente | 1-A |
| I | Información del consumo de energía | 1-3 |
| J | Resumen del desglose de la factura | 1-5 |
| K | Desglose detallado de la factura | 2-D |
| M | Otros elementos facturados | 3-4 |
| N | Impuestos | 1-8 |

Dentro de un ejemplo de esta base de datos, estos campos se ven reflejados de la siguiente manera:

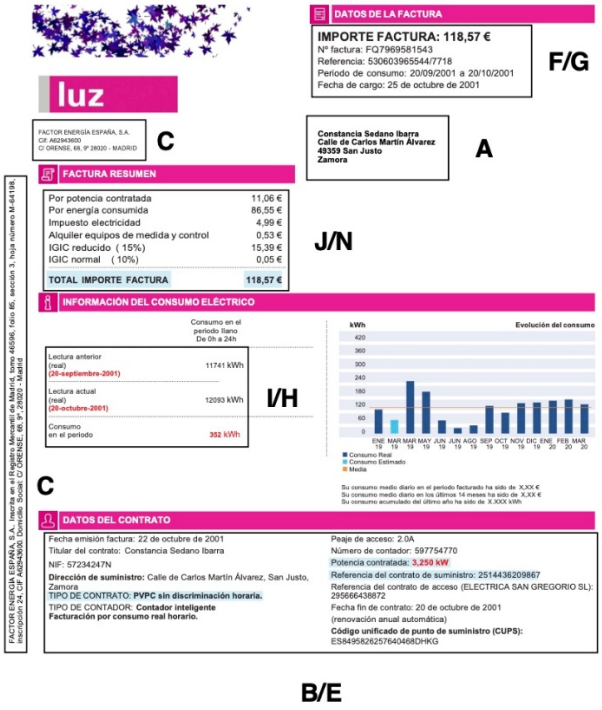


**Figura 7: Campos dentro de una factura (I)**

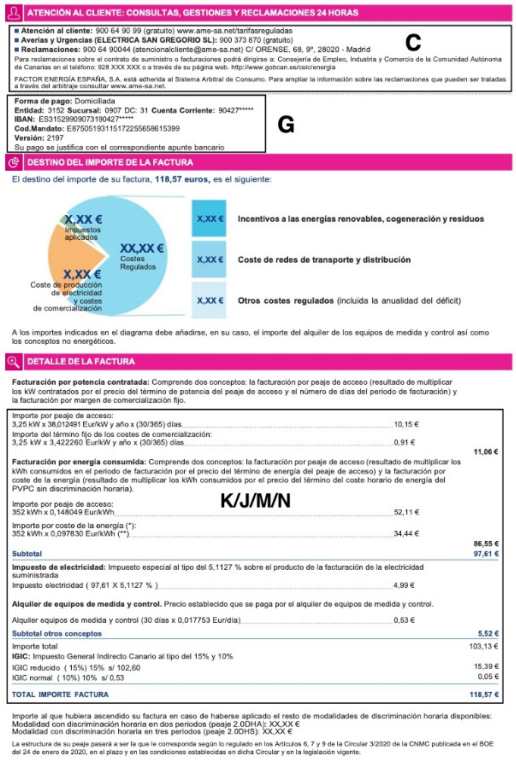


**Figura 8: Campos dentro de una factura (II)**

En el caso del documento mostrado en las figuras anteriores, se pueden observar diversos grupos, la dirección de la comercializadora y su código de identificación debajo del logo de la compañía; en el

lado derecho, se encuentra información general de la factura, como el número de esta, la fecha de emisión o el periodo de facturación; los datos del cliente se colocan debajo de este grupo; el resumen de la cantidad facturada se sitúa en el principio del cuerpo del documento junto al consumo de energía en el periodo de facturación que se encuentra debajo.

En la siguiente página, se puede encontrar más información sobre el contrato, un desglose de los conceptos de la factura, un gráfico con la distribución del destino de las cantidades facturadas, datos sobre el consumo de energía por periodos y la información de contacto.

Uno de los principales conceptos de la factura es la potencia contratada, expresada en kilovatios (kW). Esta información, junto al precio de la energía consumida en €/kW y el número de días, generan el precio de la energía. Dependiendo de la potencia contratada, el peaje de acceso aumenta la cantidad a pagar.

Otro aspecto importante es la energía consumida, que es el número de kWh consumidos por el cliente. La diferencia entre la energía consumida con respecto al periodo anterior en kWh, y el precio unitario en €/kWh, generan el precio de la energía consumida. El peaje de acceso puede incluir discriminación horaria, esto significa que, la energía consumida tiene otra tarifa según la hora.

**Tabla 5: Campos del cliente**

| Campo | Descripción |
|---|---|
| **A1/B1** | Nombre del cliente |
| **A2/B2** | DNI del cliente |
| **A3/B3** | Dirección del cliente |
| **A4/B4** | Código postal |
| **A5/B5** | Ciudad del cliente |
| **A6/B6** | Provincia del cliente |

En la tabla anterior se describen los campos del cliente en los que se diferencia entre el cliente que recibe la factura y el titular de la cuenta. Hay un total de 6 campos en los que se incluyen el nombre, el DNI y la dirección completa.

Los campos relacionados con el distribuidor y la comercializadora se encuentran en la siguiente tabla. Estos campos incluyen datos como la dirección postal, la información de contacto o la página web. En el caso del distribuidor, solo se tienen en cuenta el nombre y el sitio web. Los campos relacionados con el contrato se encuentran en la tabla 7.

**Tabla 6: Campos de la comercializadora y distribuidora**

| Campo | Descripción |
|---|---|
| **C1** | Nombre de la comercializadora |
| **C2** | CIF de la comercializadora |
| **C3** | Dirección de la comercializadora |
| **C5** | Ciudad de la comercializadora |
| **C6** | Provincia de la comercializadora |
| **C7** | Información de la compañía en el registro comercial |
| **C8** | Dirección de la administración de la compañía |
| **C9** | Página web de la comercializadora |
| **CA** | Correo electrónico de la comercializadora |
| **CB** | Nombre corto de la compañía |
| **CC** | Número de contacto |
| **CD** | Número de soporte al cliente |
| **CE** | Número de teléfono para reclamaciones |
| **D1** | Nombre de la distribuidora |
| **D2** | CIF de la distribuidora |
| **D9** | Sitio web de la distribuidora |
| **DC** | Número de teléfono de servicio |
| **DD** | Número de teléfono para interrupciones |

**Tabla 7: Campos del contrato**

| Campo | Descripción |
|---|---|
| **E1** | Código universal del punto de suministro (CUPS) |
| **E2** | Tipo de contrato |
| **E3** | Potencia contratada. Este valor se expresa de dos maneras: en formato corto, 4’5kW(E3), o formato largo, 4’500kW (E31) |
| **E4** | Número de contrato |
| **E5** | Número del contador |
| **E6** | Peaje de acceso |
| **E7** | Fecha final del contrato |
| **E8** | Código para la clasificación nacional de actividades económicas (CNAE) |
| **E9** | Referencia del contrato de suministro |

En la tabla 8 muestra los campos relacionados con la información de facturación. Estos campos incluyen el número de factura y la fecha de facturación. Los campos relacionados con el método de pago se encuentran detallados en la tabla 9.

**Tabla 8: Campos de información de la factura**

| Campo | Descripción |
|---|---|
| **F1** | Número de factura |
| **F2** | Referencia de la factura. Algunas compañías usan dos valores para F1 y F2 |
| **F3** | Fecha de emisión de la factura |
| **F4** | Principio del período de facturación |
| **F5** | Final del período de facturación |
| **F6** | Total de días facturados |
| **F7** | Días por año |
| **F8** | Número del mes |

**Tabla 9: Campos de información del pago**

| Campo | Descripción |
|---|---|
| **G1** | Método de pago |
| **G2** | IBAN (*International Bank Account Number*) |
| **G3** | Fecha del pago |
| **G4** | Secuencia de números que identifican el pago |
| **G5** | Secuencia de números que identifican el pago junto a G4 |
| **G6** | Código del banco |
| **G7** | Código de oficina |
| **G8** | Dígitos de control |
| **G9** | Número de cuenta bancaria |
| **GA** | Nombre del banco |

En cuanto a los campos de consumo energético, estos se encuentran destallados en la tabla 10. Algunas compañías solo incluyen el total de energía consumido en el período de facturación actual. Otras, incluyen adicionalmente la energía consumida en períodos anteriores y la cantidad final es calculada mediante la diferencia entre el período actual y el anterior. En ocasiones, el consumo está detallado en horas de luz punta, llana y valle.

La tabla 11 muestra los campos con el detalle de los precios. Aunque la organización de estos campos es diferente en cada comercializadora, normalmente se mantiene una estructura común. La potencia eléctrica contratada, la energía consumida y los impuestos se encuentran en todas las facturas, sin embargo, aspectos como el alquiler del equipamiento, descuentos o subtotales no siempre son incluidos.

**Tabla 10: Campos del consumo de energía**

| Campo | Descripción |
|---|---|
| **I1** | Consumo energético en kWh en el período anterior |
| **I2** | Consumo energético en kWh en el período actual |
| **I3** | Número de kWh consumidos en el período. Diferencia entre I1 e I2 |

**Tabla 11: Campos de los precios**

| Campo | Descripción |
|---|---|
| **J1** | Precio de la potencia contratada |
| **J2** | Precio de la energía consumida |
| **J3** | Subtotal 1. Suma de J1 y J2 |
| **J4** | Subtotal 2. Precio sin impuestos. Suma de J3 y N3 |
| **J5** | Precio total de la factura |

La tabla 12 enumera los campos de la energía contratada y los precios de esta. La tabla 13, los campos relacionados con los diferentes impuestos incluidos en la factura

**Tabla 12: Campos del desglose de la factura**

| | Campo | Descripción |
|---|---|---|
| **Potencia** | **K2** | Peaje de acceso (€/kW). Las facturas pueden incluir el peaje como €/kW por año o por día |
| | **K3** | Precio del peaje de acceso (€) |
| | **K4** | Tasa de la comercializadora (€/kW) |
| | **K5** | Costo de la comercializadora (€) |
| | **K6** | Costo de la energía contratada calculada a partir de la suma de K2 y K4 |
| **Energía** | **K9** | Tasa del peaje de acceso (€/kW) |
| | **KA** | Precio del peaje de acceso (€) |
| | **KB** | Tasa del costo de energía (€/kWh) |

| | | |
|---|---|---|
| | **KC** | Costo de la energía (€) |
| | **KD** | Tasa de la energía obtenida de la suma de K9 y KB |

**Tabla 13: Campos de los impuestos de la factura**

| Campo | Descripción |
|---|---|
| **M3** | Alquiler del equipo diario |
| **M4** | Alquiler del equipo |
| **N1** | Tasa de los impuestos de la electricidad |
| **N2** | Total de los impuestos |
| **N3** | Subtotal de los impuestos y el alquiler del equipo. Suma de M4 y N2 |
| **N4** | Impuestos |
| **N5** | Impuestos reducidos |
| **N6** | Tasa de los impuestos reducidos |
| **N7** | Sima de N2 y KF |
| **N8** | Total de los impuestos |

# 6.Desarrollo

En este apartado se procederá a explicar todo lo relacionado con el desarrollo del proyecto, las etapas, la planificación, el diseño de la estructura de los datos y la implementación.

Los fragmentos de código que no han sido incluidos en esta parte se encuentran en el Anexo III.

## 6.1 Análisis

En esta parte del proyecto donde se debe analizar la aplicación a realizar teniendo en cuenta qué usuarios la van a usar, qué requisitos por parte del cliente existen, qué funcionalidades tienen mayor prioridad, etc. Es por ello, que se han creado los siguientes casos de uso e historias de usuario.

### 6.1.1 Historias de usuario

Además, en esta etapa, se desarrollaron las siguientes historias de usuario las cuales definen las funcionalidades de la plataforma.

Como se nombró anteriormente, las historias de usuario son descripciones cortas que resumen la necesidad de un usuario mientras utiliza el servicio.

El formato que siguen las historias de usuario es el siguiente:

1. **Identificador**
2. **Título de la historia**
3. **Rol del usuario:** Define si el usuario es, por ejemplo, un usuario registrado, administrador, etc.
4. **Quiero:** Acción que desea realizar el usuario
5. **Para:** Objetivo de la historia de usuario
6. **Criterios de aceptación:** Criterios que se definen para confirmar que una historia está terminada

A continuación, se mostrarán algunas de las tablas con las historias de usuario más destacadas de la planificación, en el Anexo II se podrán ver las otras historias que no están aquí expuestas con el fin de facilitar la lectura de este documento.

**Tabla 14: HU-2 Descarga del dataset**

| | |
|---|---|
| **ID** | HU-2 |
| **HISTORIA** | Descarga del dataset |
| **COMO** | Usuario |

| | |
|---|---|
| **QUIERO** | Poder descargar el dataset al pulsar el botón de descarga en la página de presentación (HU-1) |
| **PARA** | Descargar el dataset |
| **CRITERIOS DE ACEPTACIÓN** | - Debe estar en la pantalla de inicio<br>- Al pulsarlo debe:<br>1. Descargar el archivo directamente<br>2. Llevar al usuario al sitio donde está contenido |

**Tabla 15: HU-3 Página de resultados**

| | |
|---|---|
| **ID** | HU-3 |
| **HISTORIA** | Página de resultados |
| **COMO** | Usuario |
| **QUIERO** | Poder visualizar los resultados obtenidos por otros usuarios |
| **PARA** | Obtener información y/o comparar los métodos |
| **CRITERIOS DE ACEPTACIÓN** | - Mostrar todos los resultados actualizados en una tabla que contenga el nombre del método y los distintos parámetros de evaluación.<br>- Solo se verán los métodos públicos<br>- Si el usuario está registrado, también verá sus propios métodos privados<br>- Poder pulsar en cada método y ver los resultados de este en detalle (HU-4) |

**Tabla 16: HU-4 Página de método en detalle**

| | |
|---|---|
| **ID** | HU-4 |
| **HISTORIA** | Página de método en detalle |
| **COMO** | Usuario |
| **QUIERO** | Visualizar los resultados de un método específico |
| **PARA** | Obtener una mejor información de los resultados |
| **CRITERIOS DE ACEPTACIÓN** | - Mostrar en detalle los distintos parámetros de evaluación y sus resultados. |

**Tabla 17: HU-5 Subir método**

| | |
|---|---|
| **ID** | HU-5 |
| **HISTORIA** | Subir método |

| | |
|---|---|
| **COMO** | Usuario |
| **QUIERO** | Subir mis métodos para que sean evaluados |
| **PARA** | Obtener una evaluación del rendimiento del método |
| **CRITERIOS DE ACEPTACIÓN** | - Si el usuario no está registrado, deberá iniciar sesión<br>- Debe existir un formulario que permita introducir los siguientes campos:<br>1. Nombre del método<br>2. Información adicional<br>3. Enlace a la publicación de este<br>4. Fichero del resultado de la ejecución en formato *.zip*<br>- El formulario debe ser validado antes de ser subido<br>- Mostrar un mensaje:<br>a. De confirmación si el método ha sido subido<br>b. De error en caso de que exista algún error en el proceso<br>- En caso de éxito, mostrar una pantalla de carga mientras se evalúa el método. Una vez finalizado, se redirige al usuario a la página de resultados. |

**Tabla 18: HU-6 Administración de métodos**

| | |
|---|---|
| **ID** | HU-6 |
| **HISTORIA** | Administración de métodos |
| **COMO** | Usuario |
| **QUIERO** | Realizar modificaciones de los métodos subidos en la página |
| **PARA** | Ofrecer funcionalidad CRUD |
| **CRITERIOS DE ACEPTACIÓN** | - Debe existir un apartado en la plataforma llamado: “Mis métodos”, en esta página, se mostrarán todas las subidas existentes del usuario<br>- Cada método subido será mostrado en la tabla con los resultados y existirá un apartado para leer, actualizar o borrar el método. |

**Tabla 19: HU-7 Editar información de un método**

| | |
|---|---|
| **ID** | HU-7 |
| **HISTORIA** | Editar información de un método |
| **COMO** | Usuario |

| | |
|---|---|
| **QUIERO** | Editar la información de cualquiera de los métodos que haya subido al sistema |
| **PARA** | Ofrecer información actualizada sobre este, corregir errores o subir actualizaciones |
| **CRITERIOS DE ACEPTACIÓN** | - Al pulsar la opción de editar en la página de los métodos del usuario (HU-6) se abrirá un formulario con los datos de este método<br>- El formulario tendrá los siguientes campos con la información actual:<br>1. Nombre del método<br>2. Información adicional<br>3. Enlace a la publicación de este<br>4. Fichero del resultado de la ejecución en formato *.zip*<br>- Al terminar las modificaciones el formulario será verificado.<br>- Se subirá el formulario una vez verificado al sistema y será reevaluado el resultado.<br>- Una vez terminado este proceso, el usuario será redirigido a la página inicial. |

**Tabla 20: HU-8 Registro de usuario**

| | |
|---|---|
| **ID** | HU-8 |
| **HISTORIA** | Registro de usuario |
| **COMO** | Usuario |
| **QUIERO** | Crear una cuenta en la plataforma |
| **PARA** | Poder usar las funciones de subir y evaluar métodos |
| **CRITERIOS DE ACEPTACIÓN** | - Debe existir una página dedicada al registro de usuarios.<br>- El formulario de registro incluirá: email, nombre de usuario y contraseña.<br>- Se debe comprobar que el nombre de usuario y el correo electrónico no existen en la base de datos<br>- Una vez registrado, se redirigirá al usuario a su página de perfil. |

**Tabla 21: HU-9 Iniciar sesión**

| | |
|---|---|
| **ID** | HU-9 |
| **HISTORIA** | Iniciar sesión |
| **COMO** | Usuario |

| | |
|---|---|
| **QUIERO** | Autentificarme |
| **PARA** | Acceder a las funciones que lo requieran y entrar en mi espacio personal |
| **CRITERIOS DE ACEPTACIÓN** | - Debe existir una página dedicada a iniciar sesión<br>- Se podrá iniciar sesión con el nombre de usuario o email y la contraseña<br>- Una vez iniciada la sesión, se redirigirá al usuario a su página de perfil |

**Tabla 22: HU-15 Página de perfil de usuario**

| | |
|---|---|
| **ID** | HU-15 |
| **HISTORIA** | Página de perfil de usuario |
| **COMO** | Usuario |
| **QUIERO** | Acceder a mi perfil |
| **PARA** | Visualizar los datos de este perfil |
| **CRITERIOS DE ACEPTACIÓN** | - Página de visualización del perfil<br>- Poder ver los métodos creados por el usuario<br>- Enlace para poder editar los datos<br>- Botón para eliminar el perfil |

**Tabla 23: HU-16 Actualizar perfil de usuario**

| | |
|---|---|
| **ID** | HU-16 |
| **HISTORIA** | Actualizar perfil de usuario |
| **COMO** | Usuario |
| **QUIERO** | Actualizar mi perfil |
| **PARA** | Tener los datos actualizados |
| **CRITERIOS DE ACEPTACIÓN** | - Página de edición del perfil con formulario para la modificación<br>- Poder modificar todos los datos |

**Tabla 24: HU-18 Ver resultados en detalle por template**

| | |
|---|---|
| **ID** | HU-18 |
| **HISTORIA** | Ver resultados en detalle por template |
| **COMO** | Usuario |
| **QUIERO** | Ver los resultados de cada método |
| **PARA** | Poder obtener una mejor visión de estos |

| | |
|---|---|
| **CRITERIOS DE ACEPTACIÓN** | - Tiene que haber un enlace dentro de los detalles de cada método.<br>- Los resultados deben de estar divididos en templates |

**Tabla 25: HU-19 Ver resultados en detalle por campo**

| | |
|---|---|
| **ID** | HU-19 |
| **HISTORIA** | Ver resultados en detalle por campo |
| **COMO** | Usuario |
| **QUIERO** | Ver los resultados de cada método por cada campo de los templates |
| **PARA** | Poder obtener una mejor visión de estos |
| **CRITERIOS DE ACEPTACIÓN** | - Tiene que haber un enlace dentro de los detalles de cada método.<br>- Los resultados deben de estar divididos en categorías |

**Tabla 26: HU-21 Ver resultados en detalle por campo en cada template**

| | |
|---|---|
| **ID** | HU-21 |
| **HISTORIA** | Ver resultados en detalle por campo en cada template |
| **COMO** | Usuario |
| **QUIERO** | Ver los resultados de cada método por cada campo de cada template |
| **PARA** | Poder analizar los resultados en profundidad |
| **CRITERIOS DE ACEPTACIÓN** | - Tiene que haber un enlace dentro de los detalles de cada método.<br>- Los resultados deben de estar divididos en categorías y en ficheros.<br>- Se podrá ver un template por página |

### 6.1.2 Casos de uso

El esquema de casos de uso trata de mostrar de manera gráfica y esquemática las actividades que un actor deberá seguir para llevar a cabo algún proceso.

Para realizar este esquema se deben tener en cuenta diversos aspectos. En primer lugar, quiénes son los actores de la aplicación desarrollada, en este caso, los actores o roles de usuarios que existirán son: usuarios no registrados, son aquellos usuarios que no se han registrado o no han iniciado en la plataforma y los que tendrán la funcionalidad más limitada por este motivo; los usuarios registrados

son aquellos que hayan facilitado sus credenciales y se autentifiquen en la aplicación, tendrán un mayor rango de acciones disponibles y los administradores serán aquellos usuarios identificados que además tienen permisos especiales para administrar el contenido de la página.

En segundo lugar, se debe saber qué acciones van a realizar estos actores y cuáles son sus intereses, el primer tipo, no registrado, necesitará poder iniciar sesión o registrarse, podrá ver los métodos y sus detalles, acceder a los enlaces de descarga del conjunto de datos, descargar los resultados de los métodos públicos, contactar con la plataforma y ver los detalles de esta. Los usuarios registrados, adicionalmente, podrán realizar un *CRUD* con los métodos personales que sean subidos y ver su página de perfil con sus datos. Por último, los administradores necesitarán de un proceso para el CRUD del contenido y *changelog* de la plataforma.

El esquema de casos de uso resultante teniendo en cuenta estas condiciones para los usuarios no registrados sería el siguiente:

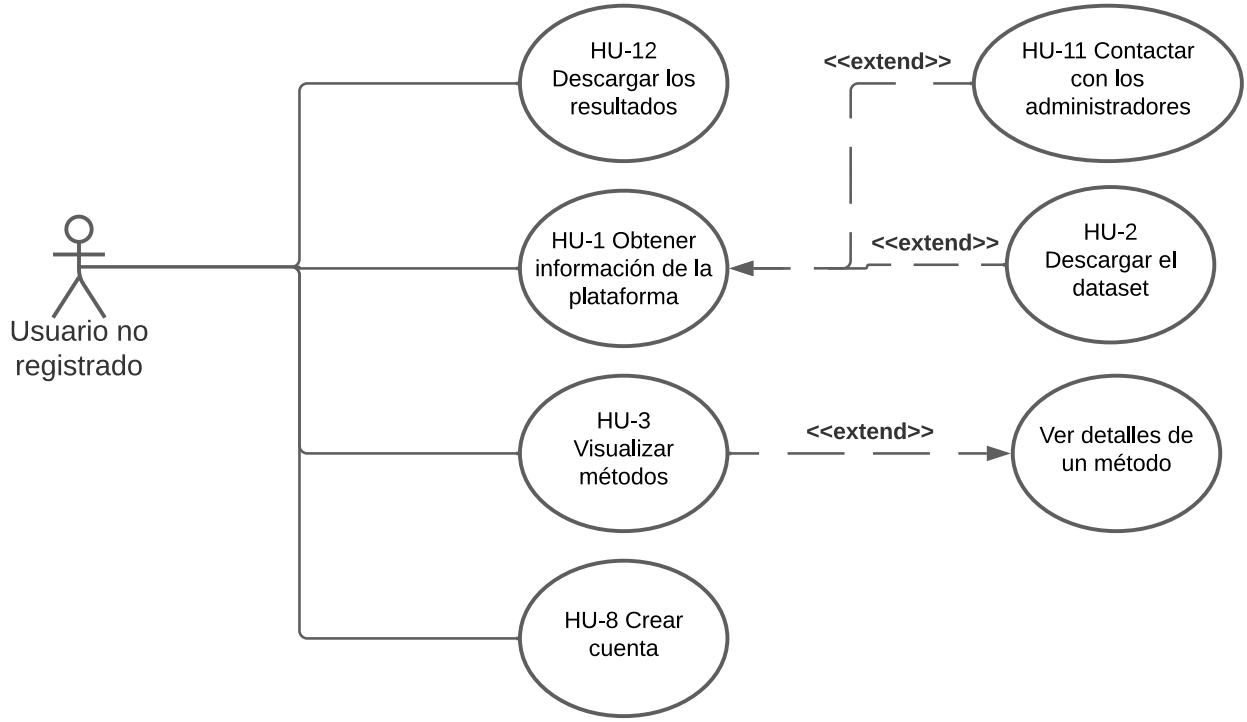


**Figura 9: Casos de uso - Usuario no registrado**

Adicionalmente, el usuario registrado puede realizar todo lo anterior más lo que vemos en la siguiente figura:

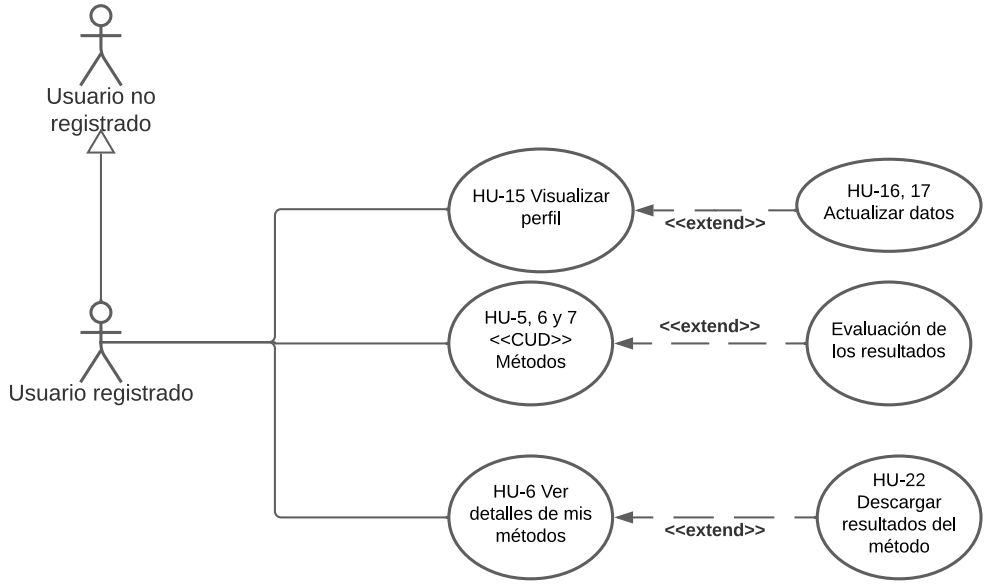


**Figura 10: Casos de uso - Usuario registrado**

Por último, los administradores heredan todas las funcionalidades más las propias de administración.

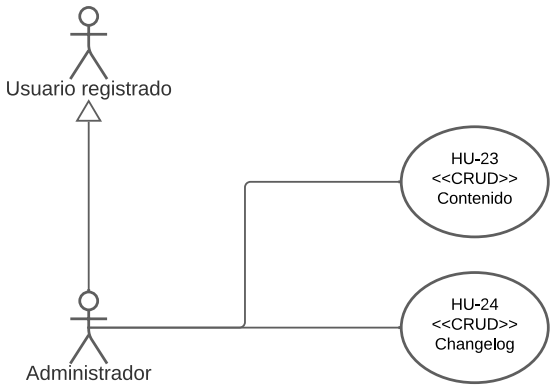


**Figura 11: Casos de uso - Administrador**

## 6.2 Diseño

En este apartado se mencionará cómo se almacenan los datos de la aplicación, ya que se usan dos recursos fundamentales para esto y qué ficheros se admiten en la plataforma. El primero, es la base de datos de MongoDB anteriormente señalado. Aquí es donde se almacenan los datos de los métodos y sus resultados, los datos de los usuarios registrados de la plataforma y el contenido personalizado que los administradores de esta hayan decidido añadir.

Los archivos que los usuarios suben con los resultados de la ejecución se almacenan en el disco duro en una carpeta con el nombre del método, que será único para cada método. Además, estos ficheros una vez se suben se convierten a *.zip* para que puedan ser descargados cuando el usuario lo solicite.

### 6.2.1 Tipos de ficheros admitidos

Los ficheros que se suben a la plataforma deben seguir un formato establecido para que la evaluación sea correcta y el sistema no compare archivos que no provienen del mismo origen.

En primer lugar, estos archivos se deberán subir a la plataforma en formato *.zip* para garantizar la compatibilidad en el servidor, en el formulario de subida y actualización de métodos se avisa al usuario para que tenga en cuenta esta limitación.

Dentro de ese fichero comprimido, se encuentra la siguiente distribución de carpetas:

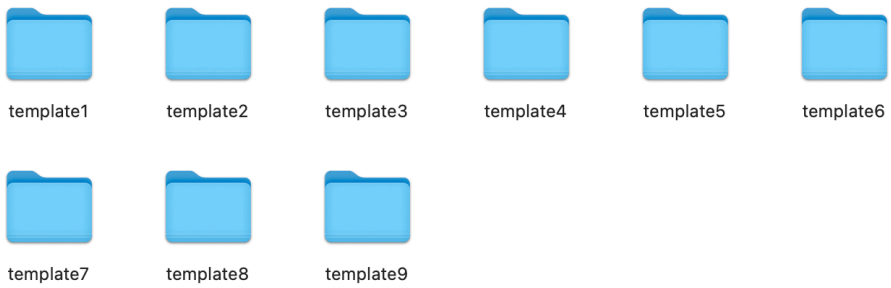


**Figura 12: Distribución de carpetas de los resultados**

En cada una de esas carpetas, se deberán encontrar los archivos *.json* con los resultados obtenidos, en total, se deberían encontrar 5000 ficheros con este formato pues es el tamaño del dataset por cada template. Sin embargo, esta cantidad puede ser modificada en el futuro para ampliar o actualizar el conjunto de datos si así lo decide el grupo de investigación. Si se accede dentro del directorio *./template1/* se verá lo siguiente:

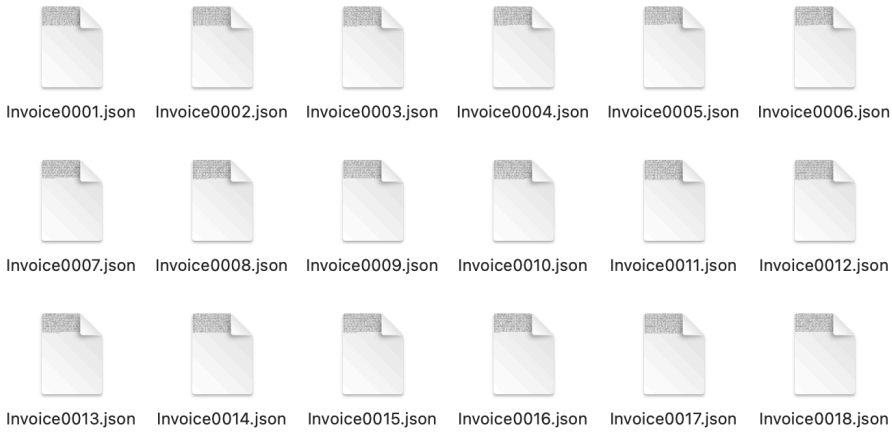


**Figura 13: Ejemplo de ficheros con los resultados**

Y dentro de cada uno de esos ficheros, se verán los resultados de esa factura con los campos que se han mencionado anteriormente y el dato obtenido, por ejemplo:

```
{
      "B1": "Nombre del cliente",
      "B2": "12345678A",
      …,
      "N8": "10,13"
}
```

Luego, dentro del disco duro cuando los distintos métodos sean subidos y evaluados, se almacenarán internamente en un directorio que contendrás las carpetas y un *.zip* de los resultados como se mencionó anteriormente. Es decir, los datos almacenados en el servidor se verían de la siguiente manera:

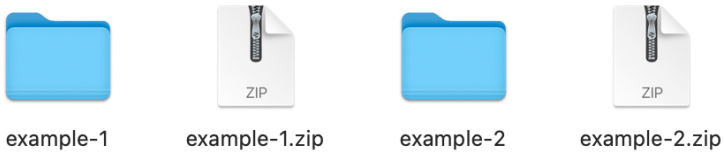


**Figura 14: Ejemplo de ficheros almacenados en el servidor**

### 6.2.2 Diseño de la base de datos

La base de datos del sistema es un elemento clave para el correcto funcionamiento de este, siendo esta la razón por la cual un buen diseño mejora y facilita la posibilidad de futuras actualizaciones y mantenimientos.

Con esta idea en mente, MongoDB fue la mejor opción de cara al desarrollo de la plataforma pues se trata de una BBDD que ofrece un rendimiento excelente y, además, facilita la expansión de los documentos si así se desea. Cabe mencionar en este tipo de bases de datos no existen las relaciones entre tablas, aunque, MongoDB permite simular este comportamiento si así se requiere. Sin embargo, para este proyecto no se ha considerado necesaria la creación de relaciones entre documentos.

En el presente trabajo se han usado 4 esquemas para la creación de datos en la base de datos. El primero y más importante es el esquema de Métodos, este es el encargado de estructurar los datos de los métodos subidos a la plataforma. Se estructura de la siguiente manera:

```python
class NewMethodModel(BaseModel):
    name: str = Field(max_length=25, min_length=3)
    user_id: str = Field(...)
    info: str = Field(max_length=200, min_length=5)
```

```
    link: str = Field(max_length=500)
    source_code: Optional[str] = Field(max_length=500)
    private: bool = Field(...)
    anonymous: bool = Field(...)

    @validator('link', allow_reuse=True)
    @validator('source_code', allow_reuse=True)
    def complete_link(cls, v):
        if 'https' not in v or 'http' not in v:
            raise ValueError('URL should be complete')
        return v

class MethodSchema(NewMethodModel):
    id: str = Field(...)
    file_dir: Optional[str] = Field(...)
    results: dict = Field(...)
    results_by_category: dict = Field(...)
    results_by_category_field: dict = Field(...)
    results_by_field: list = Field(...)
```

Como se puede apreciar, este esquema no se crea desde la base de datos o su gestor, si no que se crea directamente desde el código. En este caso, el esquema se establece haciendo uso de la librería *Pydantic* que viene incluida con *FastAPI*. Además, de esta manera como se observa en la figura, los esquemas son clases, lo que permite utilizar características como la herencia que permite que se la creación de diversos esquemas derivados de una misma base, aunque solo sea uno el almacenado. En este caso, el primer esquema, `NewMethodModel`, es usado para métodos que se reciben en el servidor y aún no tienen un id ni resultados. El segundo esquema se precisa para métodos ya creados y posee los mismos campos que el primero más los resultados y el id.

Se debe destacar que los resultados se encuentran estructurados en formato de diccionario o array de diccionarios para según que tipos. Los tipos son:

1. **Resultados totales (*results*)**

   Se trata de la media de todos los resultados. Son almacenados de la siguiente manera:

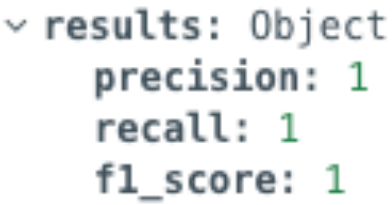


**Figura 15: Ejemplo de resultados totales**

2. **Resultados por template (*results_by_category)***

Es la media de los resultados de cada template. Contiene un diccionario el que el par Clave – Valor se corresponde a *Template* – diccionario de resultados.

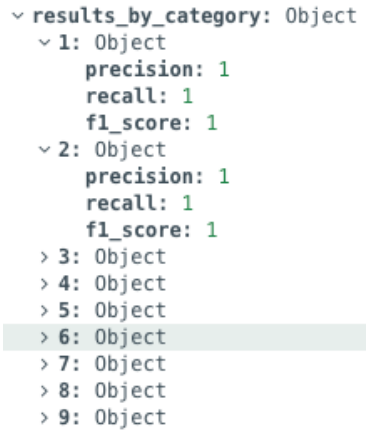


**Figura 16: Ejemplo de resultados por template**

3. **Resultados por template y campo (*results_by_category_field)***

   Media de resultados por campo de cada template. Diccionario con la clave representando el template y cuyo valor se corresponde a otro diccionario con el nombre del campo como clave y un array con los resultados como valor.

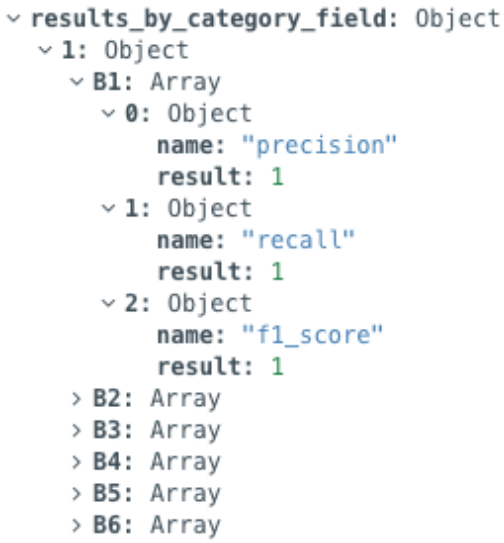


**Figura 17: Ejemplo de resultados por campo y template**

4. **Resultados por campo (*results_by_field)***

   Media de los resultados por campo en todos los templates. Contiene un array de objetos con el nombre del campo y sus resultados.

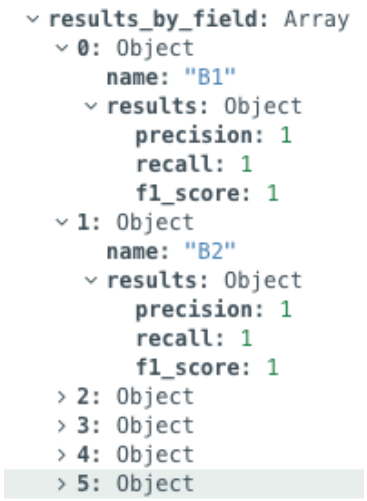


**Figura 18: Ejemplo de resultados por campo**

En el siguiente apartado se explicará con mayor detalle cómo se obtienen estos resultados.

El segundo esquema que se encuentra es el de Usuario, es este el encargado de la serialización de los datos del usuario y de que estos sean validados correctamente para evitar problemas de formato.

```
class BaseUserSchema(BaseModel):
    id: str = Field(...)
    role: str = Field(...)
    email: EmailStr = Field(...)
    username: str = Field(max_length=20, min_length=3)
    password: str = Field(min_length=6)

    @validator('role')
    def valid_role(cls, role):
        if role not in ['admin', 'user']:
            raise ValueError("Not valid role")
        return role
```

Los otros dos esquemas poseen una menor importancia pues no son utilizados directamente para la funcionalidad de la plataforma, sino que se usan para el propio contenido de la plataforma. Estos esquemas son Contenido y Changelog.

```
class ContentSchema(BaseModel):
    id: str = Field(...)
    title: str = Field(...)
    text: str = Field(...)
    page: str = Field(...)


class NewContentSchema(BaseModel):
    title: str = Field(...)
    text: str = Field(...)
```

```
    page: str = Field(...)


class BaseChangelogSchema(BaseModel):
    description: str = Field(...)
    date: str = Field(...)

    @validator('date')
    def date_regex(cls, date):
        pattern = re.compile(r"^(?:(?:31(\/|-
|\.)(?:0?[13578]|1[02]))\1|(?:(?:29|30)(\/|-|\.)(?:0?[13-9]|1[0-2])\2))(?:(?:1[6-
9]|[2-9]\d)?\d{2})$|^(?:29(\/|-|\.)0?2\3(?:(?:(?:1[6-9]|[2-
9]\d)?(?:0[48]|[2468][048]|[13579][26])|(?:(?:16|[2468][048]|[3579][26])00))))$|^
(?:0?[1-9]|1\d|2[0-8])(\/|-|\.)(?:(?:0?[1-9])|(?:1[0-2]))\4(?:(?:1[6-9]|[2-
9]\d)?\d{2})$")
        if not pattern.match(date):
            raise ValueError("Not valid date")
        return date


class ChangelogSchema(BaseChangelogSchema):
    id: str = Field(...)
```

En resumen, la estructura de los modelos de la base de datos quedaría de la siguiente manera:

**Tabla 27: Esquema de Método**

| Campo | Tipo | Descripción |
|---|---|---|
| **Id** | String | Id u ObjectID de 24 caracteres que sirve de identificador del documento |
| **User_id** | String | Id u ObjectID de 24 caracteres que identifica al dueño del método |
| **Info** | String | Descripción textual del método.<br>Longitud entre 5 y 200 caracteres |
| **Link** | String | Enlace a la publicación del método |
| **Source_code** | String<br>(Opcional) | Enlace al repositorio del código fuente |
| **Private** | Boolean | Indica si el método es privado o público |
| **Anonymous** | Boolean | Indicia si el método es anónimo |
| **Results** | Object | Diccionario con los resultados totales |
| **Results_by_category** | Object | Diccionario con los resultados por *template* |

| | | |
|---|---|---|
| **Results_by_field** | Array | *Array* con los resultados por campo |
| **Results_by_category_field** | Object | Diccionario con los resultados por *template* y campo |

**Tabla 28: Esquema de Usuario**

| Campo | Tipo | Descripción |
|---|---|---|
| **Id** | String | Id u ObjectID de 24 caracteres que sirve de identificador del documento |
| **Role** | String | Indica el rol del usuario |
| **Email** | String | String con el email que el usuario ha usado para registrarse |
| **Username** | String | Nombre de usuario |
| **Password** | String | Contraseña del usuario resumida mediante MD5 |

**Tabla 29: Esquema de Contenido**

| Campo | Tipo | Descripción |
|---|---|---|
| **Id** | String | Id u ObjectID de 24 caracteres que sirve de identificador del documento |
| **Title** | String | Título de la sección |
| **Text** | String | Contenido de la sección |

**Tabla 30: Esquema de Changelog**

| Campo | Tipo | Descripción |
|---|---|---|
| **Id** | String | Id u ObjectID de 24 caracteres que sirve de identificador del documento |
| **Date** | Date | Fecha del cambio realizado |
| **Description** | String | Descripción de los cambios realizados |

El diagrama de entidad/relación resultante de esta base sería el siguiente, aunque, se debe tener en cuenta que *MongoDB* no es una base de datos relacional por lo que no existen esas relaciones a nivel del programa como sí existiría en una base de datos que use, por ejemplo, *PostgreSQL* o *MySQL*.

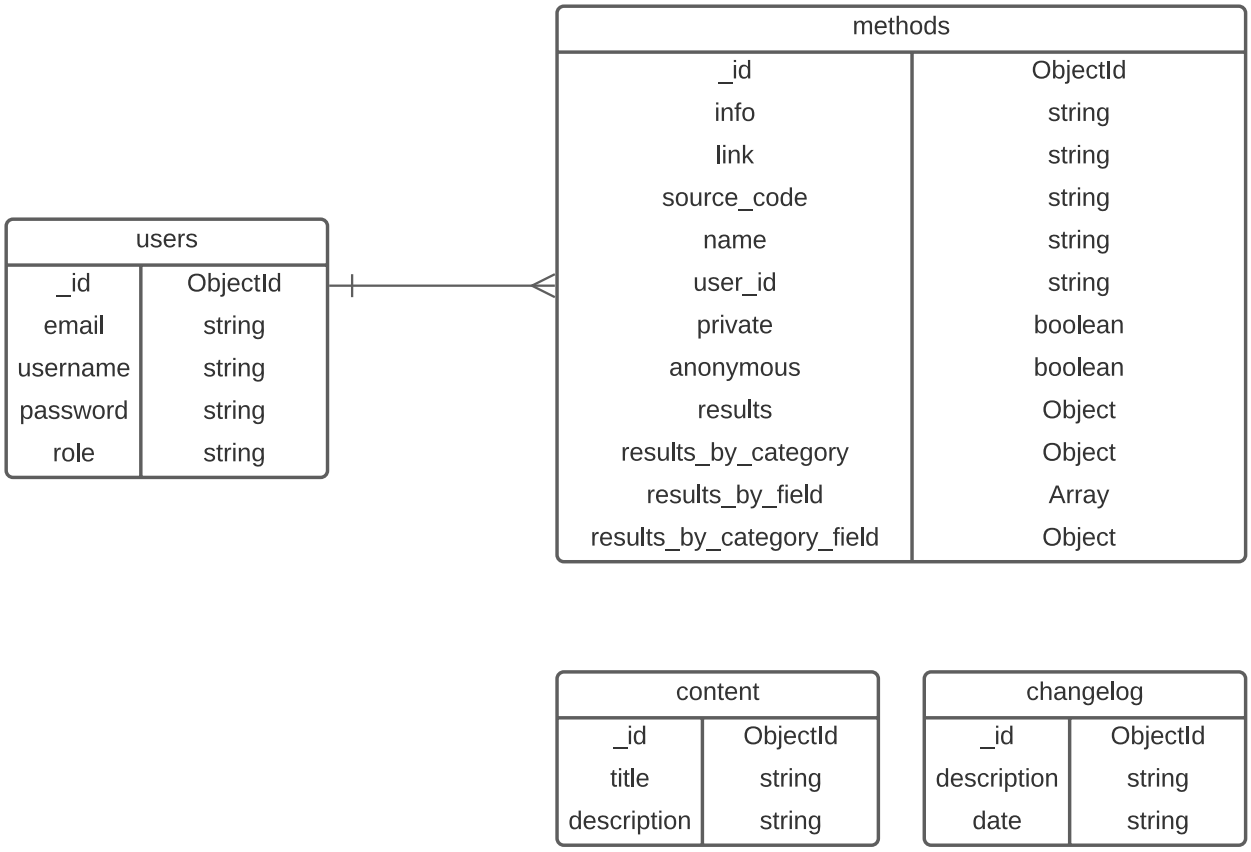


**Figura 19: Diagrama entidad-relación**

## 6.3 Implementación: Lado servidor

En este apartado se tratarán los aspectos más importantes de la implementación del *back end*.

### 6.3.1 JSON Web Token

En primer lugar, se debe hablar de la seguridad en el servidor, para que los usuarios no accedan a datos privados de otros usuarios o puedan realizar acciones que pertenecerían a usuarios con el rol de administrador, la aplicación hace uso de un *JSON Web Token*. Este *token* es un estándar definido en el documento *RFC 7519* [30]. Dentro de este documento, se define a un *JWT* como un mecanismo para poder propagar entre dos partes, de manera segura, la identidad de un usuario, además con una serie de privilegios. Estos datos se encuentran codificados como un objeto *JSON* que se inserta dentro de la cabecera o cuerpo de un mensaje firmado digitalmente.

Este JWT se genera mediante un secreto de la aplicación que se almacena en un archivo *.env* para que sea privado y nadie pueda acceder a este. Los datos que contiene en el caso de la plataforma desarrollada son los siguientes:

```
def sign_jwt(user_id: str, user_role: str) -> Dict[str, str]:
    payload = {
```

```
        "user_id": user_id,
        "role": user_role,
        "expires": time.time() + 604800
    }
    token = jwt.encode(payload, SECRET, algorithm=ALG)
    return token_response(token)
```

Como se ve en el fragmento anterior, estos datos son el id del usuario, el rol y el tiempo de expiración el cual obliga, una vez pasado ese tiempo, que el usuario tenga que iniciar sesión de nuevo.

Para poder leer este token se usa la siguiente función que lo decodifica y verifica la fecha de caducidad:

```
def decode_jwt(token: str) -> dict:
    try:
        decoded_token = jwt.decode(token, SECRET, algorithms=[ALG])
        if decoded_token['expires'] <= time.time():
            raise Exception('not valid token')
        return decoded_token
    except:
        return {}
```

Gracias a las funcionalidades de FastAPI no es necesario leer ese token en todas las funciones que lo requieran, simplemente, se crea una clase de la siguiente manera:

```
class JWTBearer(HTTPBearer):
    def __init__(self, auto_error: bool = True):
        super(JWTBearer, self).__init__(auto_error=auto_error)

    async def __call__(self, request: Request):
        credentials: HTTPAuthorizationCredentials = await super(JWTBearer,
self).__call__(request)
        if credentials:
            if credentials.scheme != "Bearer":
                raise HTTPException(status_code=403, detail="Invalid auth.
scheme")
            if not self.verify_jwt(credentials.credentials):
                raise HTTPException(status_code=403, detail="Invalid token")
            return credentials.credentials
        else:
            raise HTTPException(status_code=403, detail="Invalid auth. code")

    def verify_jwt(self, token: str) -> bool:
        token_valid: bool = False
        try:
            payload = decode_jwt(token)
        except:
```

```
            payload = None
        if payload:
            token_valid = True
        return token_valid
```

Y cada vez que se haga una solicitud a alguna de las rutas que tengan por dependencia esta clase, se llamará al método *__call__* con esta solicitud de parámetro. Dentro de esta función, se verificará el token, en caso de que sea erróneo se lanza una excepción con el código de estado 403 y se envía como respuesta el mensaje que se establezca con el parámetro *detail*. Un ejemplo de función con esta clase como dependencia la encontramos en la ruta *POST /methods/* que es la encargada de la creación y evaluación de los métodos.

```
@router.post("/",
             status_code=201,
             responses={
                 201: {"model": MethodSchema},
                 500: {"model": ErrorResponse}
             },
             dependencies=[Depends(JWTBearer())])
```

Otra ruta que, también, requiere del token es *GET /methods/all* aunque, en este caso, se requiere simplemente saber si existe para obtener el id del usuario y así poder obtener métodos públicos y los privados del usuario.

```
@router.get("/all",
            responses={
                200: {"model": MethodSchema},
                404: {"model": ErrorResponse}
            })
async def get_all_methods(request: Request):
    user_id = ""
    if 'authorization' in request.headers:
        try:
            user_id = get_id_from_token(request.headers['authorization'].split("
")[1])
        except IndexError:
            user_id = ""
    methods = find_all(user_id)
```

Como se mencionará en el apartado de lado cliente, este token generado se le envía al usuario tras el acceso o registro del usuario y se almacenará como una *cookie* en el navegador.

### 6.3.2 Política CORS

"El Intercambio de Recursos de Origen Cruzado (CORS) es un mecanismo que utiliza cabeceras HTTP adicionales para permitir que un user agent obtenga permiso para acceder a recursos seleccionados desde un servidor, en un origen distinto (dominio) al que pertenece." [31]

Para una mejor seguridad, la API por defecto tiene activada la política *CORS*, gracias a esto se consigue que solo las direcciones que sean agregadas tendrán permiso para usar los recursos. En FastAPI, esto se configura creando un array con las URLs permitidas y estableciéndolo como parámetro en la función correspondiente. En la siguiente figura se mostrará la configuración usada.

```python
origins = [
    "http://localhost",
    "http://localhost:8000",
    "http://localhost:3000",
    "https://idsem.ulpgc.es"
]

app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"]
)
```

**Figura 20: Configuración de la política CORS**

### 6.3.3 Rutas del servidor

Esta API que se ha desarrollado se encuentra divida en 4 rutas principales: */methods/*, */users/*, */content/* y */changelog/*. Cada una de estas rutas posee diversos métodos que pertenecen a una ruta anidada (Ej. GET */methods/all*) creados para cumplir con la funcionalidad establecida en las historias de usuario. A continuación, se muestran los esquemas con estas rutas.

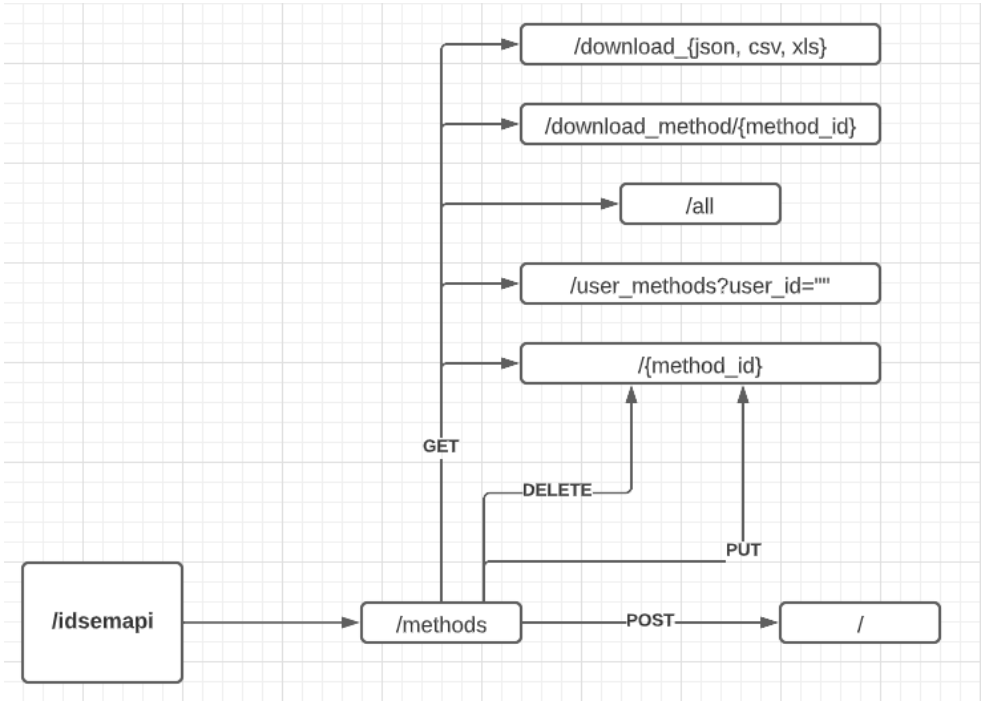


**Figura 21: Rutas para los métodos**

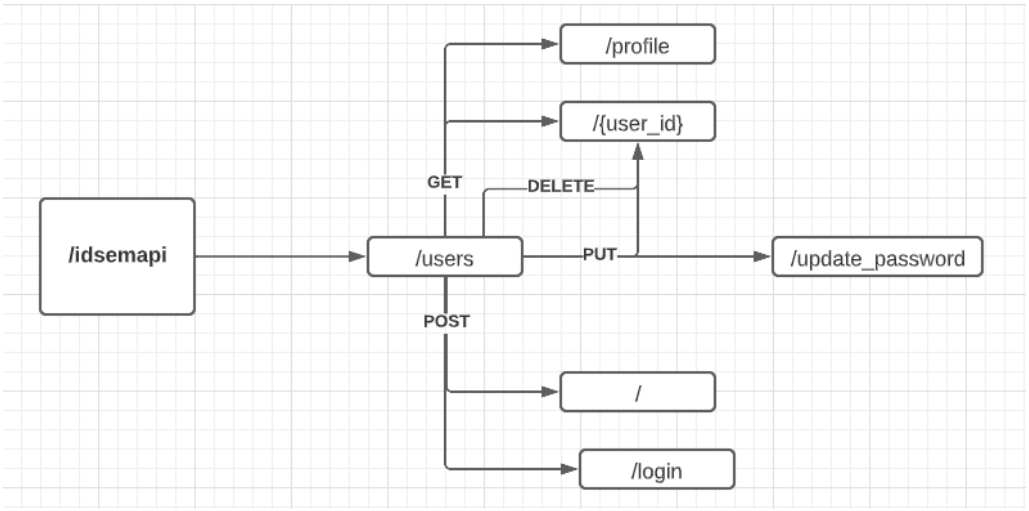


**Figura 22: Rutas para los usuarios**

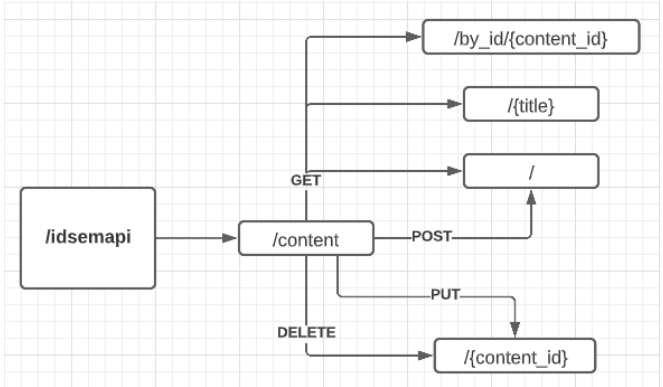


**Figura 23: Rutas para el contenido**

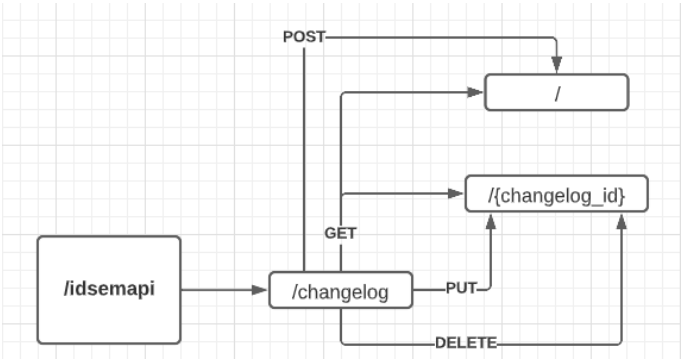


**Figura 24: Rutas para el changelog**

### 6.3.4 Evaluación de los métodos en el servidor

La ruta encargada de subir y evaluar los métodos a la plataforma mediante *POST* a la API es la siguiente:

```
@router.post("/",
             status_code=201,
             responses={
                 201: {"model": MethodSchema},
                 500: {"model": ErrorResponse}
             },
             dependencies=[Depends(JWTBearer())])
async def upload_method(file: bytes = File(...), data: str = Body(...)):
    data_json = json.loads(data)
    data = jsonable_encoder(data_json)
    file = io.BytesIO(file)
    new_method = create_method(data, file)
    if not new_method:
        raise HTTPException(500, "The method could not be created")
    return new_method
```

Dentro de esta, se procesa el cuerpo de la solicitud para convertir los datos a un diccionario que Python pueda leer, además, se procesa el archivo con los resultados. A continuación, dentro de la llamada al método correspondiente del controlador, *create_method*, se verifica que los datos cumplen con la validación del esquema que se vio anteriormente, y se comprueba de que no exista ningún método con el nombre del que se tiene intención de crear ya que este debe ser único.

```
def create_method(method, method_file):
    if not method_validation_helper(method, "", True):
        return False
    try:
        if methods_collection.find_one({"name": method['name']}):
            raise DuplicateKeyError('El valor ya existe')
        method = evaluate_method(method, method_file)
        m = methods_collection.insert_one(method)
```

```
        new_method = methods_collection.find_one({"_id": m.inserted_id})
        return methods_helper(new_method)
    except DuplicateKeyError:
        return False
```

Es en esta función donde se llama a la función encargada de la evaluación que sigue los siguientes pasos:

1. Primeramente, se inicializan el diccionario de los resultados del método y se crea la carpeta para almacenar los resultados obtenidos. Tras esto, se extraen los archivos a dicha carpeta que tiene el nombre del método.
2. Luego, se obtiene la lista de los archivos y se ordenan en orden alfabético para así tener el template número uno y el archivo 1 en primer lugar.
3. Se realiza un bucle *for* para recorrer los ficheros y procesarlos junto al modelo base que se carga desde la carpeta que esté indicada en el archivo *.env*. Dentro de este bucle, cada fichero se convierte en un diccionario para acceder a los valores obtenidos y estos se añaden a un array con los resultados, el modelo base tiene un array y el que se está evaluando, otro. Además, se van añadiendo los campos existentes y se tiene el control de que *template* se está leyendo para luego añadir el resultado obtenido en su correspondiente lugar de una manera más sencilla.

```
    for file, base in zip(file_list, base_model):
            test_data = json_to_dict(file)
            base_data = json_to_dict(base)
            base_values, fields = get_values(base_data)
            base_result.append(base_values)
            test_result.append(get_values(test_data)[0])
            i += 1
            if i == files_by_template:
                method['results_by_category'][str(template)] = {
                    'precision': 0.0,
                    'recall': 0.0,
                    'f1_score': 0.0
                }
                raw_base_result[template] = {}
                raw_test_result[template] = {}
                results_template_field[str(template)] = {}
                for f in fields:
                    raw_base_result[template][f] = []
                    raw_test_result[template][f] = []
                    results_template_field[str(template)][f] = []
                i = 0
                template += 1
```

4. Tras esto, se crean dos diccionarios que serán los que contengan los datos ordenados en formato de diccionario.

```
    template = 1
    i = 0
    for file in base_result:
        for res, field in zip(file, fields):
            raw_base_result[template][field].append(res)
        i += 1
        if i >= files_by_template:
            template += 1
            i = 0
    i = 0
    template = 1
    for file in test_result:
        for res, field in zip(file, fields):
            raw_test_result[template][field].append(res)
        i += 1
        if i >= files_by_template:
            template += 1
            i = 0
```

5. Tras esto pasos previos, se procede a la evaluación. Se empieza un bucle *for* que recorra el diccionario que se creó en el paso anterior para el modelo base.
    a. Se inicializa el diccionario con el array de las puntuaciones por template
    b. Se recorren los campos del fichero actual y se inicializan los arrays para los resultados si es necesario.
    c. Se calculan las puntuaciones y se añaden al array correspondiente.

```
for k, v in raw_base_result.items():
    f1_template[k] = []
    recall_template[k] = []
    precision_template[k] = []

    for f, res in v.items():
        if k <= 1:
            f1_field[f] = []
            recall_field[f] = []
            precision_field[f] = []
        f1s = get_f1_score(res, raw_test_result[k][f])
        recs = get_recall_score(res, raw_test_result[k][f])
        pres = get_precision_score(res, raw_test_result[k][f])

        results_template_field[str(k)][f].append(
            {
                "name": 'precision',
                "result": pres
```

```
                }
            )
            results_template_field[str(k)][f].append(
                {
                    "name": 'recall',
                    "result": recs
                },
            )
            results_template_field[str(k)][f].append(
                {
                    "name": 'f1_score',
                    "result": f1s
                }
            )
            f1_field[f].append(f1s)
            recall_field[f].append(recs)
            precision_field[f].append(pres)

        for f, res in f1_field.items():
            f1_template[k].append(res[k-1])
        for f, res in recall_field.items():
            recall_template[k].append(res[k-1])
        for f, res in precision_field.items():
            precision_template[k].append(res[k-1])
```

6. Una vez finalizado el bucle, se procede al cálculo de las medias de las puntuaciones utilizadas y se añaden al diccionario con los datos del método. Estas puntuaciones son: *F1, recall* y *precision* como explicaremos en un apartado posterior.

```
    for t, res in precision_template.items():
        method['results_by_category'][str(t)]['precision'] =
np.round(np.mean(res), decimals=4)
    for t, res in recall_template.items():
        method['results_by_category'][str(t)]['recall'] = np.round(np.mean(res),
decimals=4)
    for t, res in f1_template.items():
        method['results_by_category'][str(t)]['f1_score'] =
np.round(np.mean(res), decimals=4)

    method['results_by_category_field'] = results_template_field

    method['results_by_field'] = []
    for f in fields:
        method['results_by_field'].append({
            'name': f,
            'results': {
                'precision': np.round(np.mean(precision_field[f]), decimals=4),
                'recall': np.round(np.mean(recall_field[f]), decimals=4),
```

```
                'f1_score': np.round(np.mean(f1_field[f]), decimals=4),
            }
        })

    method['results'] = {
        'precision': np.round(np.mean(list(precision_template.values())),
decimals=4),
        'recall': np.round(np.mean(list(recall_template.values())), decimals=4),
        'f1_score': np.round(np.mean(list(f1_template.values())), decimals=4)
    }
```

7. Para finalizar la evaluación, se comprime el fichero y se devuelve el método con los nuevos datos.

Si la ejecución de esto ha resultado exitosa, se devolverá una respuesta con el código 201 y los datos del método junto a los resultados en formato *.json*. En caso contrario, se devolverá un mensaje de error con el código 500.

## 6.4 Implementación: Lado cliente

Tras la explicación del lado del servidor, se procederá a explicar cómo se ha desarrollado el lado cliente.

### 6.4.1 Diseño del sitio web

Para el diseño del sitio web se ha tratado de seguir un estilo sencillo y que resulte intuitivo, teniendo en cuenta a los usuarios que accedan desde pantallas más grandes o pequeñas, es decir, el diseño es *responsive* [32], con lo que se consigue que todo tipo de usuario pueda disfrutar de todas las características de la página sin importar el dispositivo ni el navegador, siempre y cuando este sea un navegador. Este diseño se ha logrado usando el *framework* de TailwindCSS que se explicó en el apartado de recursos.

Se debe tener en cuenta que, el usuario objetivo de esta plataforma es investigadores y en la mayoría de los casos se usará la página desde un ordenador con una pantalla más grande que un móvil.

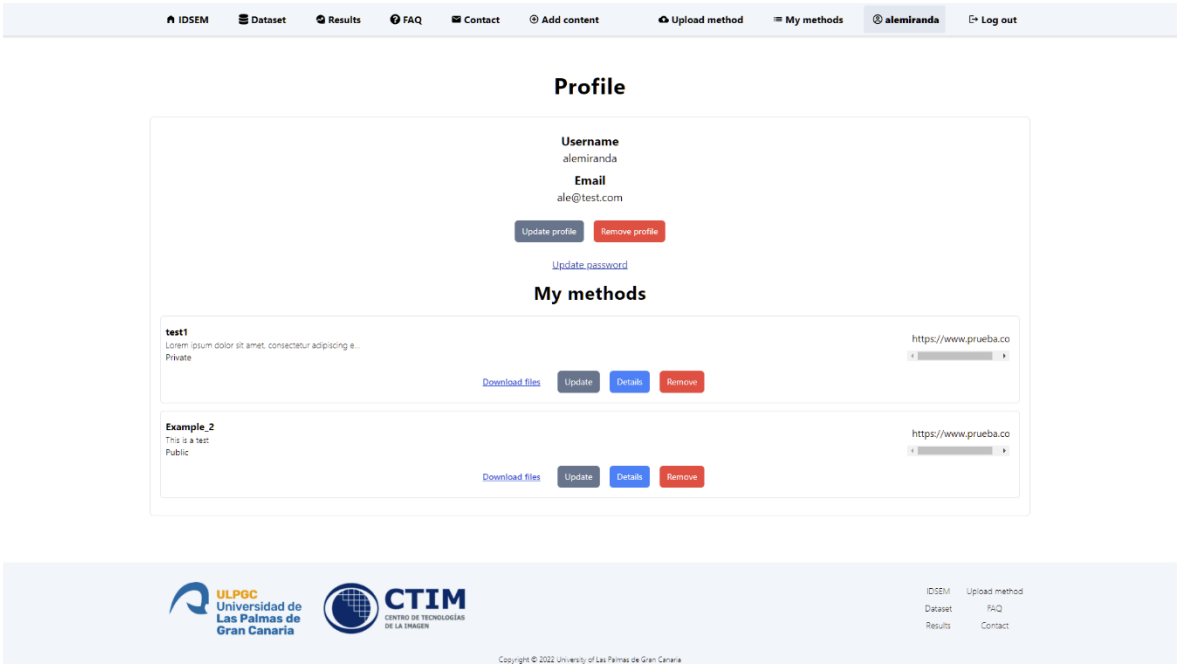


**Figura 25: Vista del perfil desde un ordenador**

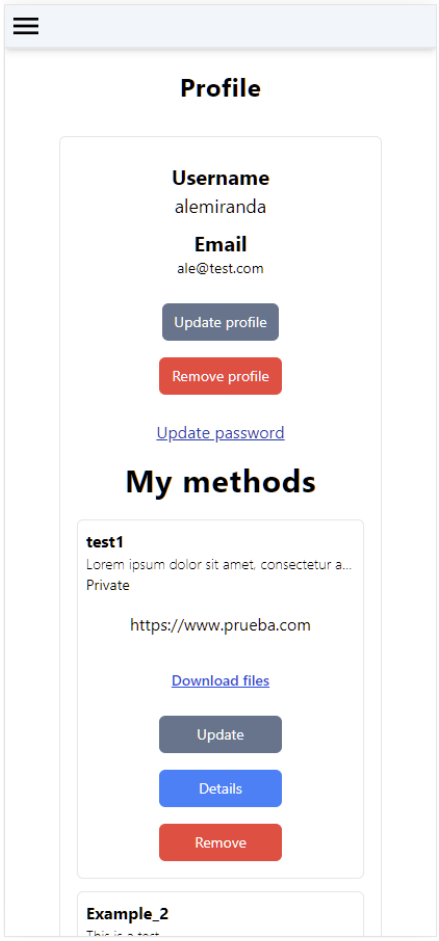


**Figura 26: Vista simulando un dispositivo móvil**

### 6.4.2 Análisis de accesibilidad

En los últimos años, los usuarios con discapacidad han ganado en visibilidad para las aplicaciones y sitios web ya que hasta, no hace mucho, acceder a una página para una persona con alguna discapacidad era imposible. Gracias a los avances que se han realizado, esto ya no es así, pues muchas páginas incorporan opciones de accesibilidad o hacen uso de etiquetas en HTML para que herramientas como los lectores de pantalla puedan leer el contenido del sitio de manera correcta y no se añada dificultad a los usuarios.

Es por esta razón, que se han seguido, en la medida de lo posible, las prácticas recomendadas y siguiendo una buena sintaxis en HTML para conseguir que los usuarios puedan hacer uso de la plataforma.

Para evaluar nuestro sitio web existe una herramienta gratuita incluida en los navegadores con *Chromium* llamada *Lighthouse*. Esta herramienta ejecuta una serie de pruebas automatizadas con el fin de obtener una puntuación e informar al desarrollador web que debe mejorar en su sitio web para que este rinda mejor, siga las buenas prácticas recomendadas, etc. Si se ejecuta esta utilidad en la web desarrollada obtendremos la siguiente puntuación:

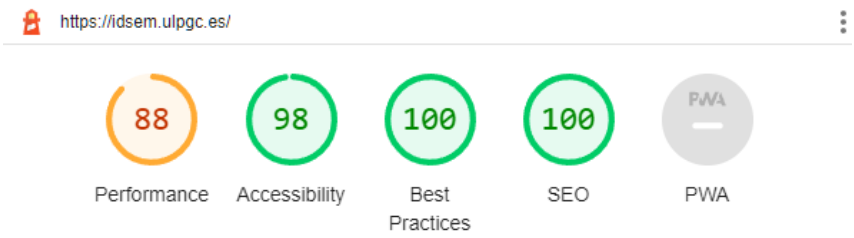


**Figura 27: Puntuación página de inicio**

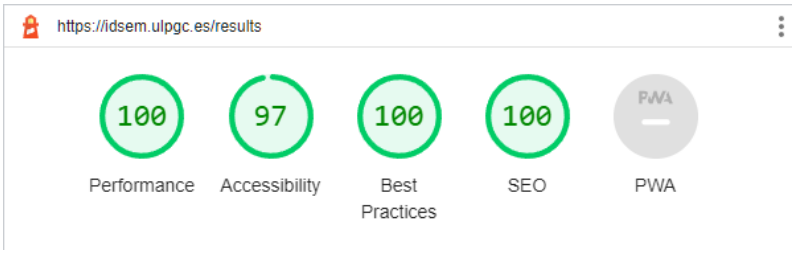


**Figura 28: Puntuación página de resultados**

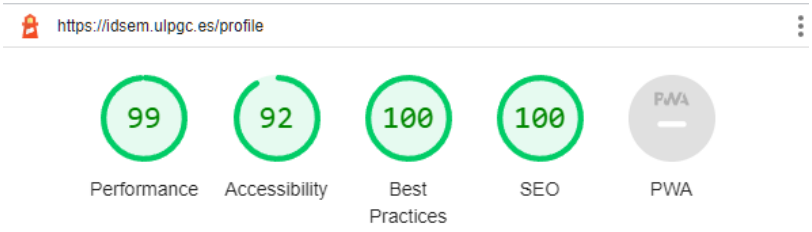


**Figura 29: Puntuación página de perfil**

Como se puede ver, se obtiene una buena puntuación en las páginas principales, en todos los aspectos, salvo en la capacidad de PWA que no aplica en este caso. Actualmente, no se ha configurado ese aspecto ni se tiene planeado configurar.

De igual manera, si se hace uso de la extensión *NoCoffee* para el navegador *Mozilla Firefox* se puede simular el efecto de distintos problemas de visión que pueden afectar a la experiencia de uso en la página.

Gracias al uso de colores neutros, blancos y grises, las personas con problemas de visión de los colores no tendrán una dificultad mayor a la de un usuario sin problemas.

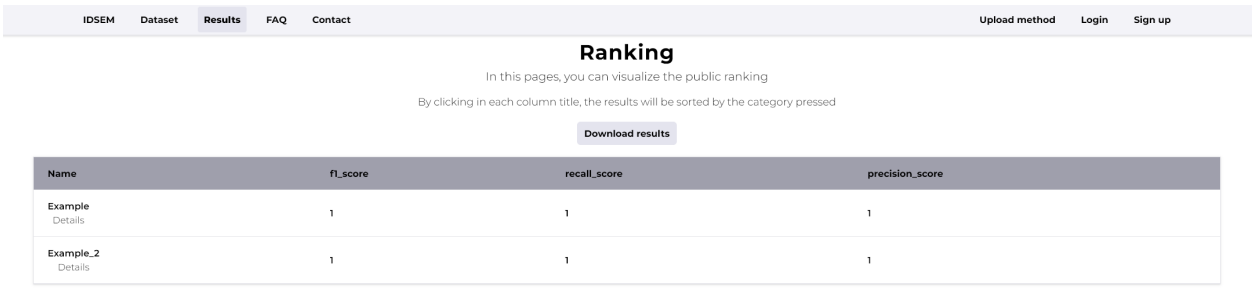


**Figura 30: Simulación de protanopia**

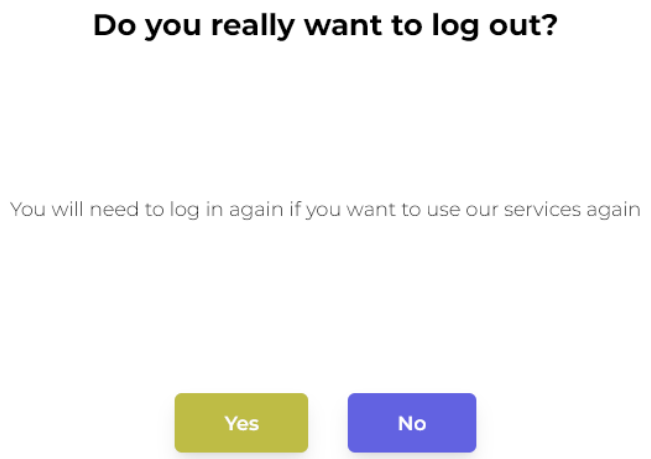


**Figura 31: Simulación protanopia (II)**

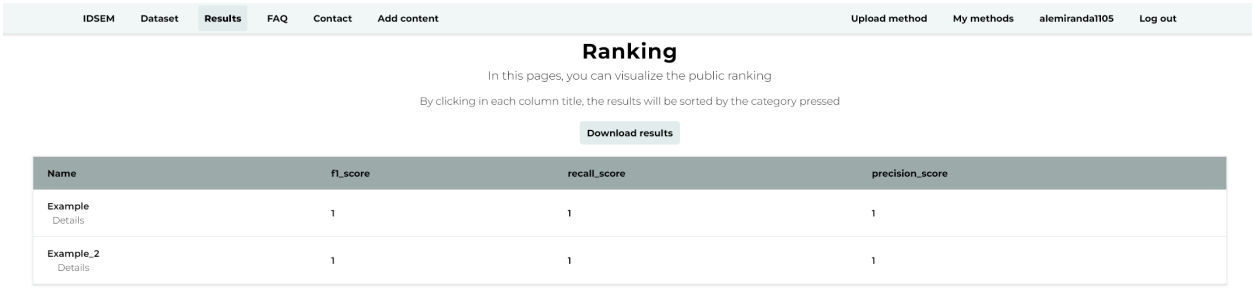


**Figura 32: Simulación de tritanopia**

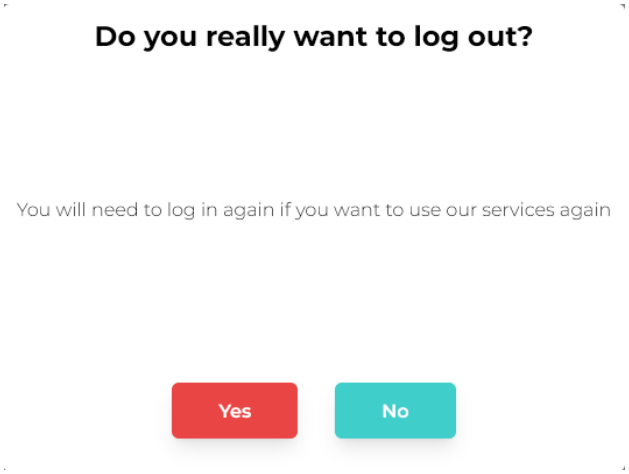


**Figura 33: Simulación de Tritanopia (II)**

### 6.4.3 Llamadas a la API

Para las llamadas a la API desde el *front end* se creó una función especial con tipos genéricos que permite centralizar las llamadas y poder acceder a los datos de la API sin importar el tipo el que sean. Esto provoca que el código desarrollado mantenga un aspecto homogéneo y en caso de que sea necesario o se realice algún cambio en la API, solo se deberá modificar una función y no múltiples. Además, la URL del servidor se almacena en archivos *.env* diferenciados entre el entorno de desarrollo y de producción, lo que facilita enormemente la labor de prueba.

Esta función desarrollada permite realizar llamadas con los cuatro métodos usados en el servidor, GET, POST, DELETE y PUT.

Sin embargo, en el caso de algunas llamadas no se hace uso de esta función pues resultaba más sencillo el uso de una personalizada, es el caso de, por ejemplo, el *login* y registro del usuario.

### 6.4.4 Iniciar sesión y crear cuenta de usuario

A continuación, se explicará la implementación del inicio de sesión y registro de usuario desde el punto de vista del lado cliente. Para conseguir esta funcionalidad se han creado dos páginas para esto

con un formulario que valida los datos antes de enviarlo al servidor. Cada campo del formulario es obligatorio y se le indica al usuario en caso de que intenten subir el formulario vacío.

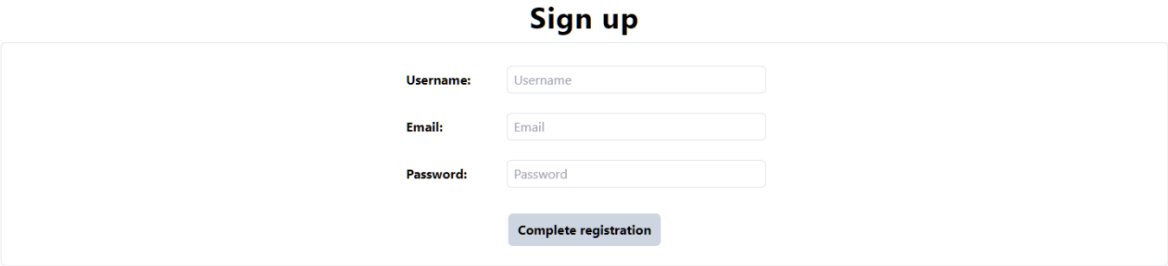


**Figura 34: Formulario de registro**

Además, se le mostrarán los mensajes de errores al momento de cambiar el campo en caso de que este sea erróneo para que se disponga de la información al momento.

En el caso del *login* ocurre lo mismo, se le muestra al usuario los campos de nombre de usuario o correo electrónico, lo podrá elegir con un selector situado justo encima del formulario, y contraseña. Estos campos son igualmente validados.

Para no repetir el código de validación, se desarrolló un *custom hook* utilizando las ventajas que *ReactJS*. Esto permitirá que el código sea más mantenible y fácil de entender.

```
    const signUp = async () => {
        if(validateUsername(userData.username) && validateEmail(userData.email)
&& validatePassword(userData.password)) {
            await axios.post(`${process.env.REACT_APP_API_URL}/users/`, userData)
            .then(res => res.data as UserDataInterface)
            .then(data => {
                if(data.token && data.id) {
                    setLoginError("");
                    setData(data);
                    saveSession(data.id, data.token);
                    setIsLogged(true);
                } else {
                    throw Error("Not valid token");
                }
            })
            .catch(error => {
                setIsLogged(false);
                if(axios.isAxiosError(error)) {
                    setLoginError(error.response?.data['detail'] ?? 'Something
went wrong, please try again');
                } else {
                    setLoginError('Something went wrong, please try again');
                }
```

```
            });
        }
    }
```

Una vez, realizado el inicio de sesión o el registro se redirigirá al usuario a la pantalla de inicio y se le actualizará la barra de navegación para un usuario registrado con los enlaces a las páginas a las que dispone de permiso para visitar.

Tras autenticarse de manera correcta, el sistema guardará el *token* recibido como una *cookie* para poder utilizarlo en las solicitudes que requieran de autenticación, por ejemplo, subir un nuevo método y mantener la sesión del usuario activa.

Cada vez que se reabre la web, se cargan los datos del usuario mediante una llamada a la *API* y se almacenan haciendo uso de la función de *React useContext*, que es una funcionalidad encargada de contener datos que se consideren de manera global en todos los componentes que se elijan. Para que estos componentes puedan acceder al contexto, deben estar entre las etiquetas de *Context Provider* y cada vez que alguna de las dependencias del contexto cambie de valor, se renderizan los componentes de nuevo. Esto puede suponer problemas de rendimientos si las variables cambian a menudo de valor, aunque en el caso de este trabajo, este problema no sucede.

Esta cookie con el *token* será eliminada al cerrar sesión o si el sistema detecta que la validez de este ha expirado y/o no puede recibir los datos del usuario.

### 6.4.5 Perfil del usuario

Esta página se corresponde a la historia de usuario *HU-15* y *HU-16*, en las que se especificaba que el usuario debía disponer de una página para poner visualizar sus datos, actualizarlos o borrar el perfil, además de poder administrar los métodos, es decir, realizar las acciones de un *CRUD*.

Para lograr esto, se ha creado un componente en el que, mediante el ID del usuario, se realiza una llamada al servidor donde se obtienen todos los datos de este, excepto la contraseña. Asimismo, se realiza otra llamada para obtener los métodos de ese usuario, tanto los públicos como los privados. Estos métodos se muestran en un componente a parte que se trata de lo mismo que vería el usuario si accede a la página de *Mis métodos*, esta acción se corresponde con la historia *HU-6*.

En el siguiente fragmento de código se muestra el código para el perfil del usuario.

```
<>
    <div className="flex flex-col flex-wrap items-center m-2 overflow-auto text-
ellipsis">
        <h3 className="text-xl font-bold">Username</h3>
```

```
        <h4 className="text-lg">{userData.username}</h4>
    </div>
    <div className="flex flex-col flex-wrap items-center m-2 overflow-auto text-
ellipsis">
        <h3 className="text-xl font-bold">Email</h3>
        <h4 className="text-sm text-left md:text-center md:text-
lg">{userData.email}</h4>
    </div>
    <div className="flex flex-col">
        <div className="flex flex-col md:flex-row items-center justify-center
text-center m-2">
            <Link to={`/update_user`} className="px-3 py-2 m-2 rounded-md text-sm
bg-slate-500 hover:bg-slate-500/40 text-white">Update profile</Link>
            {
                !showDelete &&
                <button className="px-3 py-2 m-2 rounded-md text-sm bg-red-500
hover:bg-red-500/40 text-white" onClick={() => setShowDelete(!showDelete)}>Remove
profile</button>
            }
        </div>
        <button
        className="m-2 text-blue-800 underline hover:font-bold duration-300"
        onClick={() => setShowUpdatePassword(!showUpdatePassword)}>
            { showUpdatePassword ? "Cancel": "Update password" }
        </button>
    </div>
    {
        showUpdatePassword &&
        <UpdatePasswordComponent />
    }
    {
        showDelete &&
        <DeleteUserComponent handleShow={setShowDelete} />
    }
</>
```

### 6.4.6 Visualización de los resultados

El objetivo principal de esta plataforma es ofrecer a los numerosos investigadores dedicados al desarrollo de métodos para la extracción de datos mediante visualización por computador una plataforma donde poder comparar y evaluar sus métodos. Es por ello, que se ha desarrollado una página exclusiva para la visualización de resultados totales en la que los métodos se agrupan en orden de subida por defecto y se muestra su nombre junto a los resultados totales obtenidos en las distintas puntuaciones.

| Name | f1_score | recall_score | precision_score |
|---|---|---|---|
| Example<br>Details | 1 | 1 | 1 |
| Example_2<br>Details | 1 | 1 | 1 |

**Figura 35: Tabla de resultados mostrada en la página**

En el caso de pulsar en el título de cada columna se colocará por el orden descendente de esa columna y si se vuelve a pulsar, en orden ascendente, esto sirve al usuario para buscar un resultado en específico u obtener más información de como rinde en cierta puntuación.

Pulsando en el enlace con el texto *"Details"* el usuario puede acceder a los detalles de dicho método, estos son: Nombre de usuario del autor, nombre del método, descripción, enlace a la publicación y al código fuente (este campo es opcional), privacidad del método y los resultados totales que se veían en la página anterior. Si el autor que ha subido la evaluación ha decido que es privado, el usuario no podrá acceder a los datos de este ni verlo en la tabla. En el caso de que se decida que sea anónimo no se verá el nombre del autor, pero si el resto de los campos.

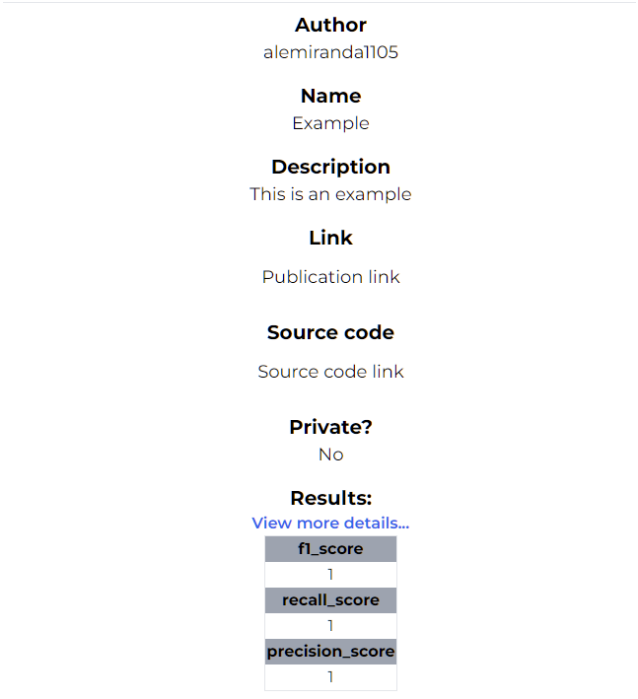


**Figura 36: Componente de los detalles del método**

En el caso de querer obtener aún más información, se podrá acceder a la visualización los resultados de manera completa. Los resultados se podrán ver por cada *template*, por cada campo o por cada campo del *template*.

Esta función sirve de gran utilidad a los investigadores, pues les permite conocer el rendimiento del método en cada punto, dónde deben mejorar, etc.

En el siguiente fragmento, se carga el tipo de visualización que ha elegido el usuario, estos son: Por campo, por *template* o por campo y *template*. Para ello se establece una interfaz que especifica dichos valores y, además, los parámetros que se le deben asignar a la función.

```
interface ResultsDetailsProps {
    methodId: string
    details: "FIELD" | "TEMPLATE" | "FIELD_TEMPLATE"
}

export const ResultDetailsComponent = ({methodId, details}: ResultsDetailsProps)
```

```
<div className="flex flex-col items-center w-full">
                <div className="flex flex-col items-center m-2 w-full">
                    {
                        (details === "TEMPLATE") &&
                        <ResultsByTemplateTable method={method}/>
                    }
                    {
                        (details === "FIELD") &&
                        <ResultsByFieldComponent method={method} />
                    }
                    {
                        (details === "FIELD_TEMPLATE") &&
                        <ResultsByFieldTemplateComponent method={method} />
                    }
                </div>
                <button
                className="p-2.5 m-2 bg-blue-500 rounded text-white font-bold
hover:rounded-none hover:bg-blue-300 duration-300"
                onClick={() => setShowDetails(!showDetails)}>
                    {showDetails? "Hide details": "Show details"}
                </button>
        showDetails &&
        <MethodDetailsComponent methodId={methodId} />
    }
</div>
```

### 6.4.7 Descarga de los resultados

Los usuarios podrán descargar los resultados subidos en la plataforma en tres formatos: JSON, CSV y XLSX. Para ello, simplemente deberán acudir a la página de resultados generales y pulsar el botón con el formato que desean. Tras esto, se mostrará una página de carga mientras se obtienen todos los

métodos y la descarga se comenzará automáticamente. En caso de que la descarga automática falle, existe un botón para forzar la descarga de manera manual.

Para el proceso de descarga, se ha realizado una función que se encarga de la llamada a la API y de convertir esos datos que le llegan a un fichero con el formato indicado en la respuesta.

```
axios.get(`${process.env.REACT_APP_API_URL}/${url}`, config)
.then(res => res.data)
.then(data => {
    if(mounted) {
        const downloadLink = window.URL.createObjectURL(data);
        setFile(downloadLink);
        const link = document.createElement('a');
        link.href = downloadLink;
        if(fileType === 'text/csv') {
            link.setAttribute('download', `file.csv`);
            setFileName('file.csv');
        } else if(fileType === 'application/json') {
            link.setAttribute('download', `file.json`);
            setFileName('file.json');
        } else if(fileType === 'application/vnd.ms-excel') {
            link.setAttribute('download', `file.xlsx`);
            setFileName('file.xlsx');
        } else if(fileType === 'application/x-zip-compressed') {
            link.setAttribute('download', 'file.zip');
            setFileName('file.zip');
        } else {
            throw Error('Operation failed');
        }
        setDownloading(false);
        link.click();
    }
})
```

### 6.4.8 Formulario de subida, actualización de resultados y evaluación

Este formulario es el encargado de subir los resultados o actualizarlos, para ello se ha generado una serie de componentes que permiten la reutilización del código ya que ambos casos requieren del mismo tipo de datos.

Este formulario posee validación, al igual que los anteriores pues resulta esencial que los datos sigan el formato esperado para que estos puedan ser almacenados de manera correcta. Por ejemplo, a continuación, se verá cómo se validan los campos de texto.

```
const handleChange = (e: React.FormEvent<HTMLInputElement|HTMLTextAreaElement>)
=> {
```

```
    const {name, value} = e.currentTarget;
    var validation: string = "";
    if(name === "name") {
        validation = validateText(value, 25, 3);
    } else if(name === "info") {
        validation = validateText(value, 200, 5);
    } else if(name === "link" || name === "source_code") {
        validation = validateText(value, 500, 0, /^(http|https)/);
    }

    setValidationError(prevState => ({
        ...prevState,
        [name]: validation
    }));

    setSubmitData(prevState => ({
        ...prevState,
        [name]: value
    }));

    var newFormData = formData || new FormData();
    newFormData.delete('data');
    newFormData.append('data', JSON.stringify(submitData));
    setFormData(newFormData);
}
```

La función *validateText()* se ha desarrollado para permitir una validación uniforme a lo largo de toda la aplicación. Esta necesita como parámetros: la longitud máxima y mínima, una expresión regular y el texto a validar. En caso de que no se valide, se devuelve un texto de error, en caso contrario, se devuelve una *String* vacía.

```
export const validateText = (text: string, maxLength?: number, minLength?:
number, regex?: string | RegExp) => {
    let textLength = text.length;
    if(regex) {
        let regexp = new RegExp(regex);
        if(!regexp.test(text)) {
            return "Invalid format";
        }
    }
    if(maxLength && (textLength > maxLength)) {
        return `The text is too long, must be shorter than ${maxLength}
characters`;
    }
    if(minLength && (textLength < minLength)) {
        return `The text is to short, must be longer than ${minLength}
characters`;
    }
```

```
    return "";
}
```

Tras esta validación, el usuario deberá adjuntar el archivo comprimido con los resultados en el caso de que se encuentre subiendo un nuevo método o desee actualizar los resultados de uno ya subido a la plataforma.

Tras adjuntar el archivo, en caso de que sea necesario, y pulsar el botón de subir, se comprueba que las validaciones son correctas y se muestra una pantalla de carga mientras se sube el resultado y se evalúa. El usuario puede permanecer o salir de la página mientras se evalúan sus resultados.

El tiempo estimado es aproximadamente 4 minutos por evaluación con el conjunto de datos actual.

En caso de permanecer en la página, se le mostrará una pantalla con el resultado de la subida, si ha sido exitosa aparece el enlace a los detalles del nuevo método subido. En caso de error se muestra el mensaje que se ha obtenido.

## 6.5 Evaluación de los métodos

Como ya se ha visto en los apartados anteriores, el servidor es el encargado de recibir los datos del método (nombre, descripción, autor, …) junto al archivo con los resultados y que para que esto sea posible, se ha desarrollado un formulario en el lado cliente para que el usuario pueda subir los datos de manera correcta y sencilla gracias a la validación del formulario en todos sus campos. Adicionalmente, se le muestra al usuario el estado (pendiente, error y correcta) de la subida en todo momento.

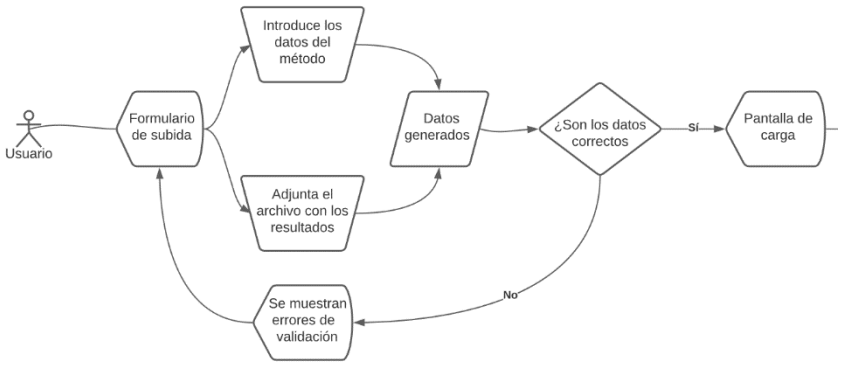


**Figura 37: Diagrama de flujo de la evaluación (I)**

En esta primera parte de la evaluación, el usuario es el encargado de acceder al formulario de subida e introducir los datos y archivo *ZIP* contenedor de los resultados generados tras la ejecución del

dataset con su método desarrollado. Esto genera un objeto de la clase *FormData* que sirve para que el servidor pueda procesar el archivo, además de los datos recibidos.

Antes de la subida del archivo, se deben comprobar que los datos cumplan con los criterios de validación establecidos, si es el caso, se continúa con el proceso, en caso contrario, se muestra un mensaje de error en el campo que no esté validado correctamente.

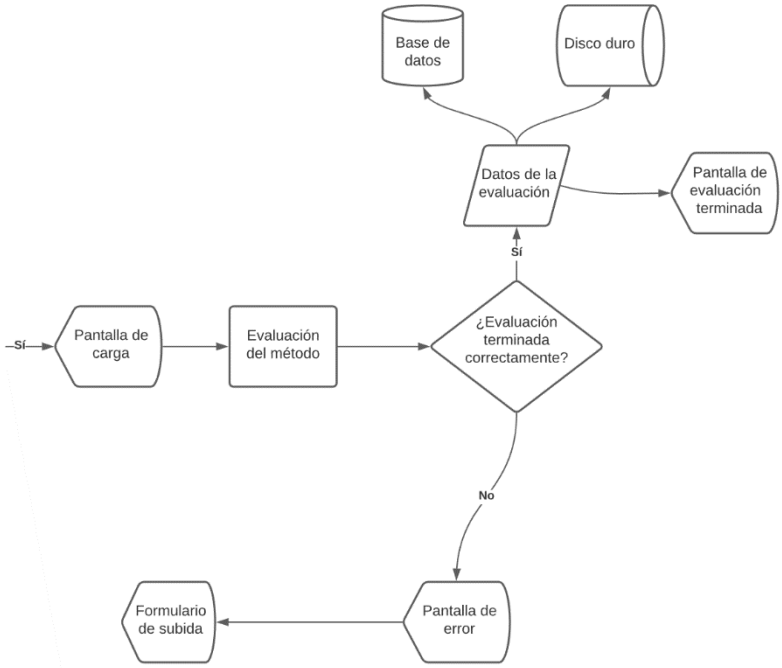


**Figura 38: Diagrama de flujo de la evaluación (II)**

Si la validación fue correcta, se procede al envío de los datos al servidor y a la evaluación, mientras esto sucede, se le muestra al usuario una pantalla de carga. Este proceso continúa con la evaluación del método, que, tras terminar, si no ocurrió ningún error, se almacenarán los datos en la base de datos y el archivo con los resultados subidos en el disco duro. Tras esto, se envían los resultados de vuelta al lado cliente y se le muestra una página al usuario enseñando que la operación fue completa de manera exitosa, en esta página tendrá un enlace a los detalles del método.

Si algo no fue de la manera esperada, se lanza un mensaje de error que para el usuario se convertirá en una pantalla de error que lo redirige al punto inicial.

### 6.5.1 Métricas usadas para la evaluación

Las métricas usadas para evaluar el rendimiento de los métodos son 3: *Precision*, *Recall* y *F1 Score*. [33]

1. **Precision**: Es la relación de positivos correctamente detectados (TP) entre el número de casos totales (TP + TF). Con esta métrica se puede medir la calidad del modelo generado.

$$Precision = \frac{TP}{(TP + TF)}$$

2. **Recall**: Es la relación entre las observaciones positivas predichas correctamente y todas las observaciones reales. Nos informa de la cantidad o porcentaje que el modelo es capaz de identificar.

$$Recall = \frac{TP}{(TP + FN)}$$

3. **F1 Score**: Es la media ponderada de *Precision* y *Recall*.

$$F1\ Score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)}$$

## 6.6 Tests

Durante el desarrollo de la aplicación en ambos lados, servidor y cliente, se siguió la metodología de desarrollo llamada *Test Driven Development*, TDD de aquí en adelante, que se trata de "*una práctica de programación que consiste en escribir primero las pruebas (generalmente unitarias), después escribir el código fuente que pase la prueba satisfactoriamente y, por último, refactorizar el código escrito*" [34]. Esta metodología permite la creación de un código más limpio y que se queden menos partes de la aplicación por probar, es decir, ayuda a que la cobertura de las pruebas unitarias sea mayor.

Para esto, en el lado servidor el *framework* usado fue *Pytest*, este marco de trabajo es el que viene incorporado con *FastAPI* y da la posibilidad probar los *endpoints* de la aplicación como si fuese una solicitud externa por lo que ofrece unos resultados de la ejecución más reales.

En cuanto al lado cliente, *JEST* es el *framework* usado. Este se trata del más popular y es compatible con *React* y *TypeScript*. Ofrece una gran flexibilidad pues permite probar el código según esté divido: en páginas, componentes, funciones, etc. En el caso de este proyecto, además, se hizo uso de una de las funciones que trata de simular las llamadas a la API para no necesitar de esta y poder corromper los datos.

```
def test_create_methods():
    for m in methods_data_test:
        response = client.post(
            "idsemapi/methods/",
            headers={
                'Authorization': 'Bearer {}'.format(mocked_jwt)
            },
            files={
                'file': (file, open(file, 'rb')),
                'data': (None, json.dumps(m)),
```

```
            }
        )
        assert response.status_code == 201
        data = response.json()
        inserted_methods.append(data)
        assert data['name'] == str(m['name'])
        assert len(data['results']) == 3
```

```
test("Form validation without file", async () => {
    const mockedAxios = axios as jest.Mocked<typeof axios>;
    const mockedResponse: AxiosResponse = {
        data: mockedMethodsList[0],
        status: 200,
        headers: {},
        config: {},
        statusText: 'OK'
    };
    mockedAxios.get.mockResolvedValueOnce(mockedResponse);
    render(
        <BrowserRouter>
            <MethodFormComponent methodId={""} withFile={false} action={""}
actionUrl={""} withMethod={true} />
        </BrowserRouter>
    );
    expect(await screen.findByText(/Name/)).toBeInTheDocument();
    expect(await screen.findByText(/Information/)).toBeInTheDocument();
    expect(await screen.findByText(/Link/)).toBeInTheDocument();

    // Fill form
    fireEvent.input(screen.getByRole("textbox", {name: 'Name:'}), {
        target: {
            value: "e"
        }
    });
    fireEvent.input(screen.getByRole("textbox", {name: 'Information:'}), {
        target: {
            value: "rr"
        }
    });
    fireEvent.submit(screen.getByText(/Submit results/));
    expect(await screen.findByText(/The text is to short, must be longer than 3
characters/)).toBeInTheDocument();
    expect(await screen.findByText(/The text is to short, must be longer than 5
characters/)).toBeInTheDocument();
    expect(await screen.findByText(/Check the data and try
again/)).toBeInTheDocument();
});
```

## 6.7 Despliegue

A continuación, se comenta el despliegue de la plataforma para que esta esté disponible a los usuarios lo antes posible y puedan empezar a utilizarla. Los recursos de los que se disponen ofrecen un rendimiento perfecto para el código desarrollado. En concreto, se trata del servicio de Plesk que se usará para el lado cliente y un servidor con Ubuntu 20.04 con Apache instalado para desplegar la API. Para entender mejor esta distribución de los servicios se debe visualizar el siguiente esquema:

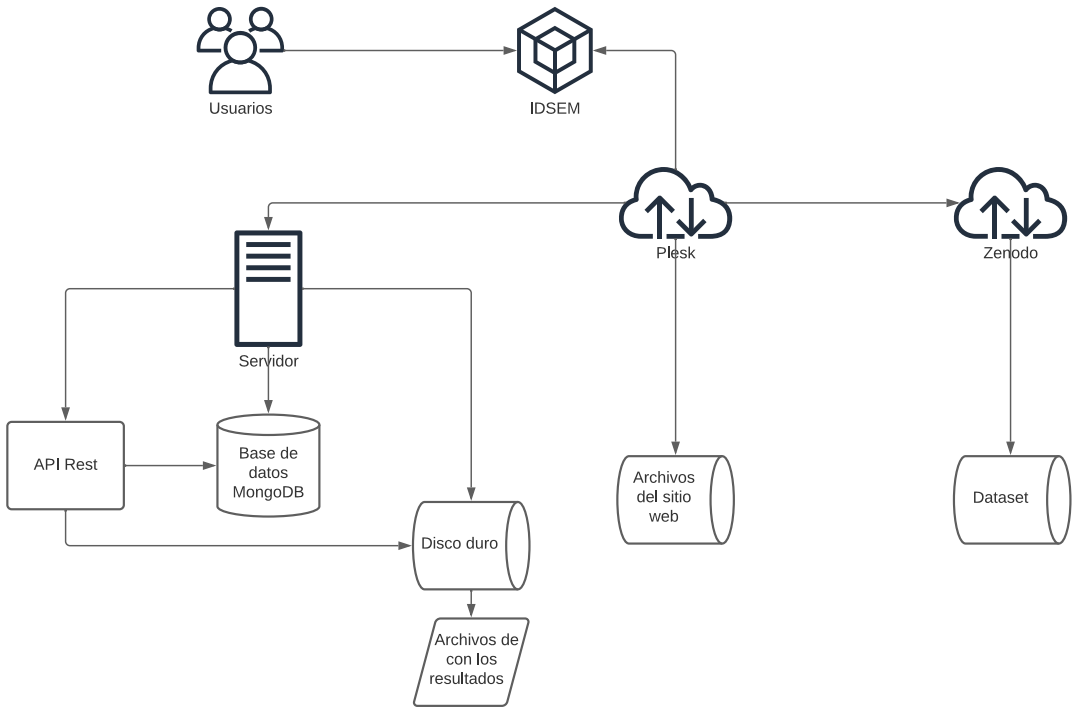


**Figura 39: Esquema del despliegue de la plataforma**

### 6.7.1 Despliegue del lado cliente

En primer lugar, para el despliegue del *front end* es necesario ejecutar el comando *npm run build* y este se encargará de compilar y optimizar todos los archivos de *TypeScript* a *JavaScript* puro para que sea compatible con el navegador. Este comando, además, añade automáticamente las imágenes y archivos CSS generados en el desarrollo del proyecto.

Tras esto, se subieron los archivos generados en la carpeta *build* al directorio raíz de Plesk.

Como se puede observar, los pasos para el despliegue de una aplicación en React son bastante sencillos en comparación con otros *frameworks* que pueden requerir de una mayor configuración y complejidad.

### 6.7.2 Despliegue del lado servidor

A diferencia del despliegue del lado cliente, el *back end* requiere de una mayor configuración para que los usuarios puedan acceder a este servicio.

El primer paso que hubo que realizar es descargar el código e instalar las dependencias que existen en el archivo *requirements.txt*, para esto se ejecuta el comando *pip install -r requirements.txt*. Además, se instaló MongoDB de manera local pues, aunque se permite crear un *clúster* en la nube a través de *Amazon Web Services* (AWS) y MongoDB Atlas, el rendimiento ofrecido de manera local era mayor que el ofrecido en la versión gratuita de este servicio en la nube.

Asimismo, se añadió en la carpeta donde se encuentra el código de la API un archivo *.env* que almacena datos importantes como la localización del modelo base para comparar o el secreto para la generación de los tokens de autenticación. Con este estilo de archivos, se permite actualizar configuración a nivel general de la aplicación sin necesidad de modificar el código.

Una vez instaladas las dependencias y softwares anteriores, se procede a la creación de un fichero para que la API se ejecute como un servicio del sistema en el arranque y se pueda controlar el estado de manera sencilla.


**Figura 40: Archivo del servicio en Ubuntu**

Con este creado, se deben ejecutar los siguientes comandos con el fin de que se active el servicio por defecto en cada arranque y permanezca en segundo plano:

1. *sudo systemctl daemon-reload*
2. *sudo systemctl start api-idsem.service*
3. *sudo systemctl enable api-idsem.service*

Para que los usuarios puedan acceder desde el exterior se hace necesario configurar un servicio de Proxy para Apache y este debe hacer uso del protocolo HTTPS, de lo contrario, los navegadores pueden rechazar la conexión por seguridad. Es por ello, que se utilizó *certbot* [35], una herramienta

gratis y de código abierto que permite la generación de certificados de *Let's Encrypt*. Para ello, se debe instalar el software y ejecutar *certbot –apache*, con esto el servidor ya hace uso de HTTPS.

Dentro de los archivos de configuración de Apache se añadió el contenido que se ve en la siguiente figura:


**Figura 41: Configuración del proxy**

Con esto, tras reiniciar Apache, se dispondrá de la API en línea para que el lado cliente pueda acceder a los datos que hayan almacenados.

# 7.Conclusiones y trabajos futuros

El objetivo del grupo de investigación era la creación de una plataforma web que permitiese el entrenamiento y evaluación de métodos de extracción de datos a partir de facturas eléctricas mediante el uso del *dataset* que se ha creado para dicha finalidad.

La respuesta a este ha sido el desarrollo de esta plataforma que permite a los usuarios realizar lo anterior mencionado de una manera sencilla y accesible mediante una interfaz web.

Este desarrollo ha permitido aumentar los conocimientos sobre las distintas herramientas y lenguajes disponibles como es el caso de *React* y *TypeScript* o *FastAPI*, que fueron las tecnologías usadas de manera principal.

En conclusión, los objetivos propuestos de manera inicial fueron cumplidos de manera exitosa permitiendo que la aplicación en el momento de redacción de este trabajo sea completamente usable de cara al usuario sin que surja ningún contratiempo. Sin embargo, por la naturaleza de las aplicaciones web se deberán seguir haciendo actualizaciones y mejorando ciertos aspectos para que la experiencia de los usuarios mejore con el paso del tiempo.

En cuanto al trabajo futuro, se tienen plateadas una serie de actualizaciones con el fin de mejorar la plataforma para aportar la mayor utilidad y mejor experiencia posible a los usuarios:

1. **Inicio de sesión con cuentas de redes sociales**
   Dar la posibilidad a los usuarios de que puedan acceder a la plataforma mediante su cuenta de, por ejemplo, *Google* o *Github*. Esto dará la oportunidad a una mejora de la seguridad debido al uso del estándar *OAuth* que usan este tipo de compañías. [36]
2. **Cumplimiento de la ley de protección de datos**
   De igual manera que en el punto anterior, asegurarse del cumplimiento de la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales resulta en una mejora de la privacidad y seguridad para el usuario y para la plataforma. Asimismo, esta ley es de obligado cumplimiento en España.
   Para cumplir con lo desarrollado en la ley, se precisa de la creación de una política de privacidad y de tratamiento de los datos del usuario. Además, se debe de establecer un responsable y encargado del tratamiento.
3. **Visualizar resultados en el archivo en comparación con el resultado esperado**
   En un futuro se pretende añadir la posibilidad de que el usuario pueda visualizar los resultados fichero por fichero y compararlos de manera visual con el fin de obtener una mayor profundidad en los análisis de estos.

4. **Comparar métodos de manera individual**

   Permitir que se puedan seleccionar entre 2 y 3 métodos y comparar sus resultados de manera individual, viendo en cada campo o *template* el resultado obtenido. Asimismo, los resultados de esta comparativa podrían ser descargados en los formatos *JSON*, *CSV* y *XLSX.*

# Anexo I: Competencias

Las competencias generales del grado y de la mención de tecnologías de la información que tienen relación con el desarrollo de este trabajo son las siguientes. [37]

***CII08****:" Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente."*

Esta competencia queda cubierta a la hora de la elección de las tecnologías usadas como se verá en el apartado de recursos utilizados, ahí se justificará las elecciones realizadas.

***TI03****:" Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas."*

En los apartados de metodología y desarrollo, se puede ver reflejada esta competencia ya que se muestra lo realizado para que el desarrollo de este proyecto resulte satisfactorio.

***TI05****:" Capacidad para seleccionar, desplegar, integrar y gestionar sistemas de información que satisfagan las necesidades de la organización, con los criterios de coste y calidad identificados"*

Durante el desarrollo se demuestra el despliegue de la aplicación y además en el apartado de Desarrollo, como se ha nombrado anteriormente, se tiene en cuenta los recursos disponibles a la hora de las distintas decisiones.

***TI06****:" Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil."*

Esta competencia cumple con el desarrollo exitoso de la aplicación ya que se trata de una infraestructura web completamente desarrollada desde cero.

# Anexo II: Historias de usuario

A continuación, se mostrarán las historias de usuario que no fueron expuestas en la memoria principal para no dificultar la lectura de esta:

**Tabla 31: HU-1 Página de presentación**

| | |
|---|---|
| **ID** | HU-1 |
| **HISTORIA** | Página de presentación |
| **COMO** | Usuario |
| **QUIERO** | Acceder a la plataforma y ver una página inicial con la descripción del proyecto y ejemplos del dataset propuesto |
| **PARA** | Obtener información sobre el dataset y la plataforma |
| **CRITERIOS DE ACEPTACIÓN** | - Debe ser la página inicial<br>- Debe contener las características de la plataforma<br>- Debe contener las características del dataset o base de datos |

**Tabla 32: HU-10 FAQ**

| | |
|---|---|
| **ID** | HU-10 |
| **HISTORIA** | FAQ |
| **COMO** | Usuario |
| **QUIERO** | Acceder a una página de preguntas frecuentes |
| **PARA** | Obtener una mejor información sobre la plataforma |
| **CRITERIOS DE ACEPTACIÓN** | - Debe mostrarse las preguntas más frecuentes respondidas de manera clara. |

**Tabla 33: HU-11 Página de contacto**

| | |
|---|---|
| **ID** | HU-11 |
| **HISTORIA** | Página de contacto |
| **COMO** | Usuario |
| **QUIERO** | Contactar con los responsables de la plataforma |
| **PARA** | Cualquier incidencia/duda que pueda ocurrir |

| | |
|---|---|
| **CRITERIOS DE ACEPTACIÓN** | - Debe haber un formulario de contacto donde el usuario ponga su mensaje<br>- Debe haber un link para contactar mediante correo electrónico. |

**Tabla 34: HU-12 Descarga tabla de resultados**

| | |
|---|---|
| **ID** | HU-12 |
| **HISTORIA** | Descarga tabla de resultados |
| **COMO** | Usuario |
| **QUIERO** | Descargar la tabla de resultados |
| **PARA** | Integrar la información en artículos y analizar los resultados |
| **CRITERIOS DE ACEPTACIÓN** | - Debe haber un botón de descarga de la tabla<br>- Se podrá descargar la tabla en 3 formatos distintos: JSON, CSV, XLSX |

**Tabla 35: HU-13 Ordenar tabla de resultados**

| | |
|---|---|
| **ID** | HU-13 |
| **HISTORIA** | Ordenar tabla de resultados |
| **COMO** | Usuario |
| **QUIERO** | Ordenar la tabla de resultados |
| **PARA** | Visualizar las mejores puntuaciones en las distintas métricas u ordenar de manera alfabética. |
| **CRITERIOS DE ACEPTACIÓN** | - Podrá ordenarse la tabla, por cada resultado, de mayor a menor o viceversa.<br>- Podrá ordenarse la tabla en orden alfabético, descendente o ascendente. |

**Tabla 36: HU-14 Hacer método público o privado**

| | |
|---|---|
| **ID** | HU-14 |
| **HISTORIA** | Hacer método público o privado |
| **COMO** | Usuario |
| **QUIERO** | Modificar la privacidad de mis métodos |
| **PARA** | Poder subir los métodos y compararlos de manera privada |

| | |
|---|---|
| **CRITERIOS DE ACEPTACIÓN** | - Poder modificar la privacidad antes de subir el método<br>- Poder cambiar la privacidad en cualquier momento una vez el método sea publicado.<br>- Poder hacer un método público pero que no se muestre el autor |

**Tabla 37: HU-17 Borrar perfil de usuario**

| | |
|---|---|
| **ID** | HU-17 |
| **HISTORIA** | Borrar perfil de usuario |
| **COMO** | Usuario |
| **QUIERO** | Borrar mi perfil |
| **PARA** | Eliminar los datos de este perfil |
| **CRITERIOS DE ACEPTACIÓN** | - Botón dentro de los detalles del perfil<br>- Botón dentro del formulario de actualización<br>- Pedir confirmación antes de eliminar |

**Tabla 38: HU-20 Enlace código fuente del método**

| | |
|---|---|
| **ID** | HU-20 |
| **HISTORIA** | Enlace código fuente del método |
| **COMO** | Usuario |
| **QUIERO** | Añadir un enlace al código fuente |
| **PARA** | Que el resto de los usuarios pueda obtener más información |
| **CRITERIOS DE ACEPTACIÓN** | - Tiene que aparecer en el formulario de creación como campo opcional<br>- Tiene que aparecer en el formulario de actualización como campo opcional |

**Tabla 39: HU-22 Descargar los archivos del método**

| | |
|---|---|
| **ID** | HU-22 |
| **HISTORIA** | Descargar los archivos del método |
| **COMO** | Usuario |
| **QUIERO** | Descargar los resultados de mis métodos |
| **PARA** | Poder recuperarlo, compararlos, etc. |

| | |
|---|---|
| **CRITERIOS DE ACEPTACIÓN** | - En la página de perfil, aparecerá un botón de descarga, al lado de cada método personal<br>- En la página de mis métodos, aparecerá un botón de descarga |

**Tabla 40: HU-23 Modificar el contenido**

| | |
|---|---|
| **ID** | HU-23 |
| **HISTORIA** | Modificar el contenido |
| **COMO** | Administrador |
| **QUIERO** | Modificar el contenido de las distintas páginas de presentación |
| **PARA** | Poder actualizar el contenido o corregir errores de esta |
| **CRITERIOS DE ACEPTACIÓN** | - Al iniciar sesión, aparecerá en el menú un botón de edición.<br>- En cada sección, aparecerá un botón de edición.<br>- Las páginas que se podrán modificar serán: Página de presentación y Changelog |

**Tabla 41: HU-24 Changelog**

| | |
|---|---|
| **ID** | HU-24 |
| **HISTORIA** | Changelog |
| **COMO** | Administrador |
| **QUIERO** | Añadir contenido al changelog de la página |
| **PARA** | Mantener a los usuarios informados de los cambios de la plataforma |
| **CRITERIOS DE ACEPTACIÓN** | - El changelog debe aparecer en la página inicial.<br>- Aparecerá un botón al lado del título para editar |

**Tabla 42: HU-25 Información sobre copyright**

| | |
|---|---|
| **ID** | HU-25 |
| **HISTORIA** | Información sobre copyright |
| **COMO** | Usuario |
| **QUIERO** | Ver la información del copyright de la plataforma |
| **PARA** | Saber los derechos de autor |

| | |
|---|---|
| **CRITERIOS DE ACEPTACIÓN** | - Esta información aparecerá en la página inicial |

**Tabla 43: HU-26 Citar a la plataforma**

| | |
|---|---|
| **ID** | HU-26 |
| **HISTORIA** | Citar a la plataforma |
| **COMO** | Usuario |
| **QUIERO** | Poder citar a la plataforma |
| **PARA** | Poder utilizarla en trabajos de investigación |
| **CRITERIOS DE ACEPTACIÓN** | - Esta información aparecerá en la página inicial<br>- Podrá aparecer en distintos formatos (Ej.: LATEX) |

# Anexo III: Código

## III.1. Código para las llamadas a la API

```
export function useFetch<T, R>(url: string, method: string = "get", body?: R) {
    const [data, setData] = useState<T>();
    const [isPending, setPending] = useState(true);
    const [error, setError] = useState("");
    useEffect(() => {
        let mounted = true;
        setPending(true);
        setError("");
        let config = {
            headers: {
                Authorization: `Bearer ${getCookie('token')}`
            }
        }
        const selectMethod = async () => {
            if(method === "post") {
                return axios.post(`${process.env.REACT_APP_API_URL}/${url}`,
body, config);
            } else if(method === "delete") {
                return axios.delete(`${process.env.REACT_APP_API_URL}/${url}`,
config);
            } else if(method === "put") {
                return axios.put(`${process.env.REACT_APP_API_URL}/${url}`, body,
config);
            } else {
                return axios.get(`${process.env.REACT_APP_API_URL}/${url}`,
config);
            }
        }
        selectMethod()
        .then(res => res.data)
        .then(data => {
            if(mounted) {
                setData(data);
                setError("");
                setPending(false);
            }
        })
        .catch(error => {
            if(axios.isAxiosError(error)) {
                setError(error.response?.data.detail);
                setPending(false);
            } else {
                setError('Something went wrong, please try again');
                setPending(false);
```

```
            }
        });
        return function cleanup() {
            mounted = false;
        }
    }, [url, body, method])
    return { data, isPending, error }
}
```

## III.2. Validaciones

```
useEffect(() => {
    if(userData.username.length > 0 &&!validateUsername(userData.username)) {
        setValidationError(prevState => ({
            ...prevState,
            username: "Write a valid username"
        }));
    } else {
        setValidationError(prevState => ({
            ...prevState,
            username: ""
        }));
    }
    if(userData.email.length > 0 && !validateEmail(userData.email)) {
        setValidationError(prevState => ({
            ...prevState,
            email: "Write a valid email"
        }));
    } else {
        setValidationError(prevState => ({
            ...prevState,
            email: ""
        }));
    }
    if(!fromLogin) {
        if(userData.password.length > 0 && !validatePassword(userData.password))
{
            setValidationError(prevState => ({
                ...prevState,
                password: "The password is too short"
            }));
        } else {
            setValidationError(prevState => ({
                ...prevState,
                password: ""
            }));
        }
    }
}, [userData.username, userData.email, userData.password, fromLogin])
```

## III.3. Registro del usuario

```
const signUp = async () => {
    if(validateUsername(userData.username) && validateEmail(userData.email) &&
validatePassword(userData.password)) {
        await axios.post(`${process.env.REACT_APP_API_URL}/users/`, userData)
        .then(res => res.data as UserDataInterface)
        .then(data => {
            if(data.token && data.id) {
                setLoginError("");
                setData(data);
                saveSession(data.id, data.token);
                setIsLogged(true);
            } else {
                throw Error("Not valid token");
            }
        })
        .catch(error => {
            setIsLogged(false);
            if(axios.isAxiosError(error)) {
                setLoginError(error.response?.data['detail'] ?? 'Something went
wrong, please try again');
            } else {
                setLoginError('Something went wrong, please try again');
            }
        });
    }
}
```

## III.4. Inicio de sesión

```
const login = async () => {
    let loginData = {
        data: userData.email,
        password: userData.password
    };
    if(userData.password.length <= 0) {
        setValidationError(prevState => ({
            ...prevState,
            password: "Write a password, please"
        }));
        return;
    }
    await axios.post(`${process.env.REACT_APP_API_URL}/users/login`, loginData)
    .then(res => res.data as UserDataInterface)
    .then(data => {
        if(data.token && data.id) {
```

```
            setLoginError("");
            setData(data);
            saveSession(data.id, data.token);
            setIsLogged(true);
        } else {
            throw Error("Not valid token");
        }
    })
    .catch(error => {
        setIsLogged(false);
        if(axios.isAxiosError(error)) {
            setLoginError(error.response?.data['detail'] ?? 'Something went
wrong, please try again');
        } else {
            setLoginError('Something went wrong, please try again');
        }
    });
}
```

# Anexo IV: Manual de usuario

## IV.1. Requisitos previos

Los requisitos necesarios para poder acceder a la plataforma son:

- Navegador compatible con *JavaScript ES2018*
- Acceso a internet

## IV.2. Barra de navegación

La barra de navegación a la que se referirá este manual de aquí en adelante es la siguiente:

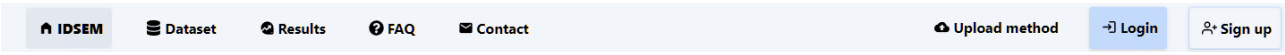


**Figura 42: Barra para un usuario no registrado**


**Figura 43: Barra para un usuario registrado**

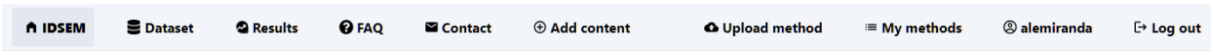


**Figura 44: Barra para el administrador**

Esta barra permite la navegación de manera sencilla a lo largo de todo el sitio web.

## IV.3. Inicio de sesión y registro

Para el registro en la plataforma, se debe acudir la página correspondiente mediante la barra de navegación y pulsar en el enlace con el texto *Sign up*.

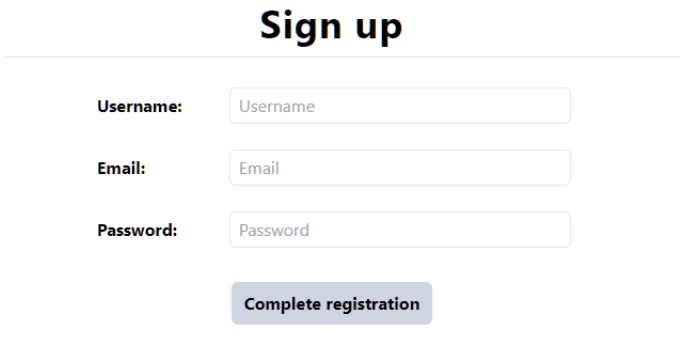


**Figura 45: Formulario de registro**

A continuación, se debe rellenar los datos solicitados en el formulario de manera obligatoria con las siguientes condiciones:

- Nombre de usuario con una longitud mínima de 3 caracteres y una longitud máxima de 20.
- Correo electrónico
- Contraseña de mínimo 6 caracteres
    - Se recomienda el uso de mayúsculas, minúsculas y símbolos especiales para una mayor seguridad. El uso de generadores y gestores de contraseña está completamente recomendado.

Tras completar los datos, se pulsará en el botón de *Complete registration*. Si todo ha ido bien, será redirigido a la página de inicio, en caso contrario, se mostrará un mensaje de error.

Para iniciar sesión, deberá pulsar en el botón de *Login* que se encuentra en la barra de navegación en la parte superior de la plataforma y aparecerá un formulario. Podrá elegir si iniciar sesión con el correo electrónico o el nombre de usuario. Al igual que en el registro, si el proceso se completa exitosamente, se redirigirá a la página inicial o se mostrará una página de error.

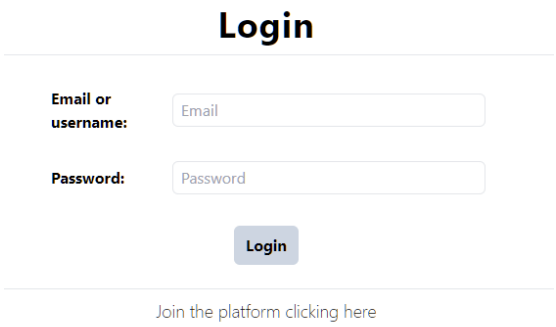


**Figura 46: Formulario de login**

## IV.4. Subida de resultados

Para subir los resultados que la ejecución del dataset ofrecido ha generado, deberá ir a la página de *Upload method* y cumplimentar el formulario mostrado con todos los datos, el único campo opcional es del código fuente.

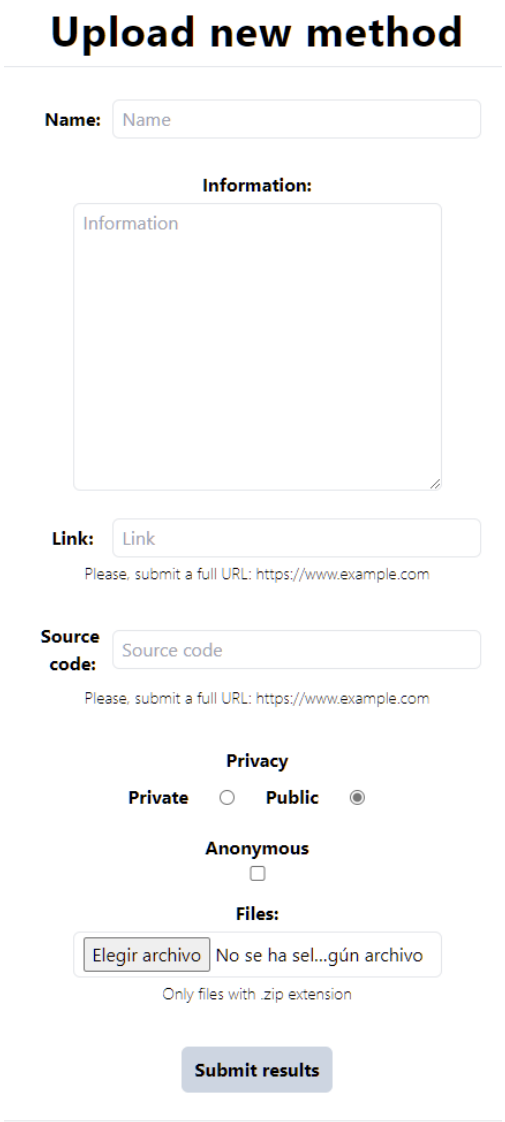


**Figura 47: Formulario de subida de un método**

Tras esto, si los campos están correctamente validados podrá realizar la subida del método. Una vez pulsado el botón se mostrará la página de carga mientras se evalúa el método en el servidor, no es necesario que permanezca en la página, esta puede ser cerrada. La subida requiere de un tiempo de aproximadamente 5 minutos.

## IV.5. Edición de datos de un método

En caso de necesitar editar los datos de un método, se debe acudir a la página de su perfil o *My methods* donde se encontrará la vista de métodos que se han subido a la plataforma con la cuenta de usuario actual. Seleccione el método a editar pulsando en el botón *Update*.

Se abrirá un formulario como el del apartado anterior con la única diferencia que los datos antiguos estarán precargados, edite los campos siguiendo las validaciones obligatorias y pulse el botón de

subida. En caso de una actualización correcta, se mostrará un mensaje informando de la completitud de la solicitud y dispondrá de un enlace a los detalles del método.

En caso de que adicionalmente se necesite actualizar los archivos del método, seleccione esa opción en la parte superior mediante el selector correspondiente.

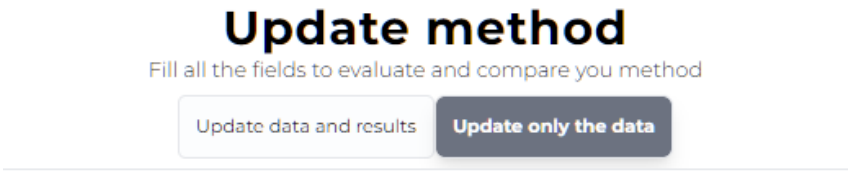


**Figura 48: Selección para actualizar el método**

## IV.6. Resultados de la plataforma

Para visualizar los resultados de los métodos en la plataforma se debe acudir a la página de *Results* mediante el enlace de la barra de navegación. En esa pestaña se mostrará una tabla de la siguiente manera:

| Name | precision | recall | f1_score |
|---|---|---|---|
| test1 Details | 0.5901 | 0.5901 | 0.5901 |
| test2 Details | 1 | 1 | 1 |

**Figura 49: Tabla de resultados**

Pulsando en el título de cada columna, los resultados serán ordenados de manera descendente. Si vuelve a pulsar, de manera ascendente según la columna seleccionada.

## IV.7. Resultados del método

Para visualizar los resultados de cada método, deberá situarse en la página de detalles de este y pulsar en el enlace de *View more details* dentro del apartado de resultados.


**Figura 50: Página de detalles**

Tras pulsar el enlace, se mostrarán, por defecto, los resultados por *template* empezando por el primero. Seleccione en los tres botones superiores qué resultados quiere ver.

**Results by template**

| Template | precision | recall | f1_score |
|---|---|---|---|
| Template 1 | 1 | 1 | 1 |
| Template 2 | 1 | 1 | 1 |
| Template 3 | 1 | 1 | 1 |
| Template 4 | 1 | 1 | 1 |
| Template 5 | 1 | 1 | 1 |
| Template 6 | 1 | 1 | 1 |
| Template 7 | 1 | 1 | 1 |
| Template 8 | 1 | 1 | 1 |
| Template 9 | 1 | 1 | 1 |

**Figura 51: Resultados por template**

**Results by field**

| Field | precision | recall | f1_score |
|---|---|---|---|
| Field B1: Customer's name | 1 | 1 | 1 |
| Field B2: Customer's identity card number | 1 | 1 | 1 |
| Field B3: Customer's address | 1 | 1 | 1 |
| Field B4: Postal code | 1 | 1 | 1 |
| Field B5: Customer's city | 1 | 1 | 1 |
| Field B6: Customer's province | 1 | 1 | 1 |
| Field A1: Customer's name | 1 | 1 | 1 |
| Field A2: Customer's identity card number | 1 | 1 | 1 |

**Figura 52: Resultados por campo**

**Results by field and template**

Template 1

| Field | precision | recall | f1_score |
|---|---|---|---|
| Field B1: Customer's name | 1 | 1 | 1 |
| Field B2: Customer's identity card number | 1 | 1 | 1 |
| Field B3: Customer's address | 1 | 1 | 1 |
| Field B4: Postal code | 1 | 1 | 1 |
| Field B5: Customer's city | 1 | 1 | 1 |
| Field B6: Customer's | 1 | 1 | 1 |

**Figura 53: Resultados por campo y template**

Pulsando en el botón de *next* o *back* podrá avanzar y retroceder, respectivamente entre los *templates,* campos o campos y *template* dependiendo del apartado en el que se encuentre.

# Bibliografía


[1] D. Karatzas, L. Gómez, A. Nicolaou y M. Rusiñol, «The Robust Reading Competition Annotation and Evaluation Platform,» *2018 13th IAPR International Workshop on Document Analysis Systems (DAS),* nº 13, pp. 61-66, 2018.

[2] Alphabet Inc., «Kaggle,» Google LLC, Abril 2010. [En línea]. Available: https://www.kaggle.com. [Último acceso: 5 Mayo 2022].

[3] H. Zheng, C. Kai, H. Jianhua, B. Xiang, K. Dimosthenis, L. Shjian y C. Jawahar, «ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction,» *2019 International Conference on Document Analysis and Recognition (ICDAR),* 2019.

[4] D. J. Butler, J. Wulff, G. B. Stanley y M. J. Black, «A naturalistic open source movie for optical flow evaluation,» *European Conf. on Computer Vision (ECCV),* vol. IV, pp. 611-625, 2012.

[5] A. Geiger, P. Lenz, C. Stiller y R. Urtasun, «Vision meets Robotics: The KITTI Dataset,» *International Journal of Robotics Research (IJRR),* 2013.

[6] IEEE, «IEEE Dataport,» [En línea]. Available: https://ieee-dataport.org. [Último acceso: 5 Mayo 2022].

[7] Mozilla Foudation, «MDN Web Docs,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/JavaScript. [Último acceso: Marzo 2022].

[8] Microsoft Corporation, «TypeScript,» 2012. [En línea]. Available: https://www.typescriptlang.org. [Último acceso: Marzo 2022].

[9] J. L. Chacón, «Profile,» 25 Octubre 2021. [En línea]. Available: https://profile.es/blog/que-es-typescript-vs-javascript/. [Último acceso: Marzo 2022].

[10] Telefónica, S. A., «LUCA,» Febrero 2020. [En línea]. Available: https://web.archive.org/web/20200224120525/https://luca-d3.com/es/data-speaks/diccionario-tecnologico/python-lenguaje. [Último acceso: Marzo 2022].

[11] Meta Platforms, Inc., «React,» Meta Platforms, Inc., [En línea]. Available: https://es.reactjs.org. [Último acceso: Marzo 2022].

[12] Facebook Open Source, «JEST,» Facebook, Inc., [En línea]. Available: https://jestjs.io. [Último acceso: Marzo 2022].

[13] Tailwind Labs, «TailwindCSS,» Tailwind Labs, 1 Noviembre 2017. [En línea]. Available: https://tailwindcss.com. [Último acceso: Marzo 2022].

[14] S. Ramírez, «FastAPI Docs,» 2019. [En línea]. Available: https://fastapi.tiangolo.com/tutorial/cors/. [Último acceso: 24 Abril 2022].

[15] Itop Management Consulting, S.L., «iTop,» [En línea]. Available: https://www.itop.es/soluciones-tecnologicas/business-analytics-business-intelligence/scikit-learn.html. [Último acceso: 5 Mayo 2022].

[16] H. Krekel, «Pytest,» 2004. [En línea]. Available: https://docs.pytest.org/en/7.1.x/. [Último acceso: 5 Mayo 2022].

[17] Microsoft Corporation, «VisualStudio Code,» [En línea]. Available: https://code.visualstudio.com/docs. [Último acceso: Marzo 2022].

[18] Wikipedia, «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/PyCharm. [Último acceso: Marzo 2022].

[19] A. López, «OpenWebinars,» 3 Junio 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-postman/. [Último acceso: 5 Mayo 2022].

[20] MongoDB, Inc., «MongoDB,» [En línea]. Available: https://www.mongodb.com/es/products/compass. [Último acceso: 5 Mayo 2022].

[21] Á. Robledano, «OpenWebinars,» 28 Octubre 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-mongodb/. [Último acceso: Marzo 2022].

[22] L. Arnedo, J. Gama y G. P. Masramon, «ProyectosAgiles.org,» [En línea]. Available: https://proyectosagiles.org/que-es-scrum/. [Último acceso: Marzo 2022].

[23] ComparaSoftware, «ComparaSoftware,» 6 Marzo 2021. [En línea]. Available: https://blog.comparasoftware.com/pila-de-producto-scrum/. [Último acceso: Marzo 2022].

[24] A. Arboleda, «Rock Content,» 13 Marzo 2020. [En línea]. Available: https://rockcontent.com/es/blog/product-owner/. [Último acceso: Marzo 2022].

[25] M. Rehkopf, «Atlassian,» [En línea]. Available: https://www.atlassian.com/es/agile/project-management/user-stories. [Último acceso: Marzo 2022].

[26] Amazon Web Services, Inc., «Precios de las instancias bajo demanda de EC2,» 2021. [En línea]. Available: https://aws.amazon.com/es/ec2/pricing/on-demand/. [Último acceso: 10 Mayo 2022].

[27] MongoDB, Inc., «MongoDB Atlas,» 2020. [En línea]. Available: https://www.mongodb.com/pricing. [Último acceso: 10 Mayo 2022].

[28] Ecoaula.es, «¿Cuánto gana un programador web en España?,» Eleconomista.es, 3 Septiembre 2021. [En línea]. Available: https://www.eleconomista.es/ecoaula/noticias/11378880/09/21/Cuanto-gana-un-programador-web-en-Espana.html. [Último acceso: 10 Mayo 2022].

[29] Levenant, «Tarifas Diseño gráfico,» Levenant, 1 Enero 2022. [En línea]. Available: https://www.levenant.com/tarifas-diseno-grafico-precios-diseno-logotipos/. [Último acceso: 10 Mayo 2022].

[30] M. Jones, J. Bradley y N. Sakimura, «JSON Web Token (JWT),» RFC Editor, Mayo 2015. [En línea]. Available: https://www.rfc-editor.org/info/rfc7519. [Último acceso: 31 Mayo 2022].

[31] Mozilla Foudation, «Control de acceso HTTP (CORS) - HTTP,» 28 Abril 2022. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/CORS. [Último acceso: 8 Mayo 2022].

[32] M. S. Boneu, «freeCodeCamp.org,» freeCodeCamp.org, 2020. [En línea]. Available: https://www.freecodecamp.org/espanol/news/diseno-web-responsive-como-hacer-que-un-sitio-web-se-vea-bien-en-telefonos-y-tabletas/. [Último acceso: 8 Mayo 2022].

[33] Exsilio Solutions, «Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures,» Exsilio Solutions, 9 Septiembre 2016. [En línea]. Available: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/. [Último acceso: 21 Mayo 2022].

[34] J. I. Herranz, «TDD como metodología de diseño de software,» Paradigma Digital, 2011. [En línea]. Available: https://www.paradigmadigital.com/dev/tdd-como-metodologia-de-diseno-de-software/. [Último acceso: 12 Mayo 2022].

[35] Electronic Frontier Foundation, «Cerbot,» 2022. [En línea]. Available: https://certbot.eff.org/. [Último acceso: 9 Mayo 2022].

[36] M. Chu, «Is It Safe To Sign In With Google? (Facebook, Twitter, Apple, Etc),» Data Overhaulers, 2020. [En línea]. Available: https://dataoverhaulers.com/sign-in-google-safe/. [Último acceso: 10 Mayo 2022].

[37] Universidad de Las Palmas de Gran Canaria, «Universidad de Las Palmas de Gran Canaria,» [En línea]. Available: https://www2.ulpgc.es/archivos/plan_estudios/4008_40/ObjetivosyCompetenciasdelGII.pdf. [Último acceso: Marzo 2022].