



TRABALLO FIN DE GRAO  
GRAO EN ENXEÑARÍA INFORMÁTICA  
MENCIÓN EN COMPUTACIÓN



# **Perfilado automático de usuarios en corpus sociales sobre el movimiento Black Lives Matter**

**Estudiante:** David Rodríguez Bacelar

**Dirección:** Patricia Martín Rodilla, David Otero Freijeiro

A Coruña, agosto de 2023.

*Dedicatoria*

### **Agradecimientos**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

## **Resumen**

## **Abstract**

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ``Huardest gefburn"? Kjift -- not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### **Palabras clave:**

- First itemtext
- Second itemtext
- Last itemtext
- First itemtext
- Second itemtext
- Last itemtext
- First itemtext

### **Keywords:**

- First itemtext
- Second itemtext
- Last itemtext
- First itemtext
- Second itemtext
- Last itemtext
- First itemtext

# Índice general

---

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Importancia de las redes sociales . . . . .	2
1.2	Perfilado de autor . . . . .	2
<b>2</b>	<b>Estado del arte del perfilado de autor</b>	<b>3</b>
2.1	Algoritmos supervisados . . . . .	4
<b>3</b>	<b>Herramientas, técnicas y lenguajes</b>	<b>5</b>
3.1	<i>Backend</i> . . . . .	5
3.2	<i>Frontend</i> . . . . .	6
3.3	Algoritmos de perfilado . . . . .	6
3.4	Soporte . . . . .	7
<b>4</b>	<b>Metodología</b>	<b>9</b>
4.1	Roles . . . . .	9
4.2	Eventos . . . . .	10
<b>5</b>	<b>Análisis</b>	<b>11</b>
5.1	Requisitos funcionales . . . . .	11
5.2	Requisitos no funcionales . . . . .	13
<b>6</b>	<b>Diseño</b>	<b>14</b>
6.1	Arquitectura . . . . .	14
6.2	Prototipado . . . . .	16
<b>7</b>	<b>Implementación</b>	<b>22</b>
7.1	<i>Backend</i> . . . . .	22
7.1.1	<i>Endpoints</i> . . . . .	22
7.1.2	Clases . . . . .	25

<b>8 Caso de uso: #BLM</b>	<b>28</b>
<b>9 Planificación y costes</b>	<b>29</b>
9.1 Planificación temporal . . . . .	29
9.2 Recursos y costes . . . . .	30
9.3 Viabilidad del proyecto . . . . .	31
<b>10 Conclusiones</b>	<b>32</b>
10.1 Lecciones aprendidas . . . . .	32
10.2 Trabajo futuro . . . . .	32
<b>A Material adicional</b>	<b>35</b>
<b>Bibliografía</b>	<b>37</b>

# Índice de figuras

---

1.1	Evolución del número de usuarios en redes sociales. . . . .	1
5.1	Diagrama de casos de uso . . . . .	12
6.1	Diagrama de capas del patrón de diseño DDD. Adaptado de L, Ryś [1] . . . . .	15
6.2	Diagrama C4 de Contenedor de la arquitectura del sistema. . . . .	15
6.3	Prototipo de la pantalla de inicio . . . . .	16
6.4	Prototipo de la selección del algoritmo de perfilado . . . . .	17
6.5	Prototipo del resumen del perfilado . . . . .	18
6.6	Prototipo de la pantalla de ejemplos de dataset . . . . .	19
6.7	Prototipo del dashboard utilizando el algoritmo Martinc . . . . .	20
6.8	Prototipo del dashboard utilizando el algoritmo Grivas . . . . .	21
7.1	Estructura del JSON devuelto por el <i>endpoint</i> /profilings/{id} . . . . .	24
7.2	Diagrama de <i>endpoints</i> del <i>backend</i> . . . . .	25
7.3	Diagrama de clases del <i>backend</i> . . . . .	27

# Índice de tablas

---

5.1	Historias de usuario . . . . .	12
5.2	Requisitos no funcionales . . . . .	13
9.1	Especificaciones del portátil empleado para el desarrollo del proyecto . . . . .	30
9.2	Coste del proyecto . . . . .	31



# Introducción

---

A PARTIR de la segunda década de los 2000, las redes sociales han experimentado un crecimiento continuado de su uso, tanto en número de usuarios como en cantidad de información generada. Como se muestra en el reporte *"Digital 2023 Global Overview Report"* (We Are Social et al., 2023) [2], a principios del año 2013 existían alrededor de 1.7 millones de usuarios en las redes sociales, mientras que a principios del año 2023, esta cifra aumentó hasta los 4.7 millones, con una variación anual media del 10.8% (se puede ver más información en la Figura 1.1). Así, plataformas como Facebook, Instagram, Twitter o YouTube, cuentan con millones de usuarios activos diariamente, donde se comparte información o se crea contenido de entretenimiento. Además, las redes sociales se han convertido en un lugar donde se debate sobre temas políticos, sociales o económicos, y donde se comparten diversas opiniones y noticias. Este hecho se puede ver, de nuevo, en el reporte mencionado anteriormente, donde se muestra que el 34.2% de los usuarios de redes sociales las utilizan para informarse sobre noticias, el 28.8% para saber cuáles son los temas de actualidad y el 23.4% para compartir opiniones y debatir con otros usuarios.

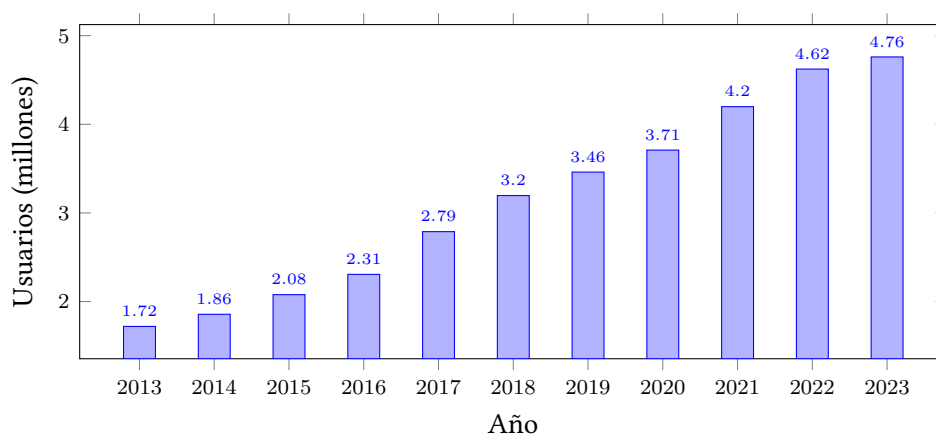


Figura 1.1: Evolución del número de usuarios en redes sociales.

## 1.1 Importancia de las redes sociales

Toda esta información generada tiene una gran relevancia a distintos niveles. En primer lugar, cabe destacar el impacto que tienen las redes sociales a nivel político. Y es que en la actualidad, vivimos en una campaña permanente (Blumenthal, 1980) [3], donde el acceso a las redes sociales ofrece la posibilidad a los ciudadanos de estar informados sobre política, mientras que a las instituciones de poder les permite conocer el estado de la opinión pública (Strömbäck, 2008) [4], pudiendo llegar a influenciar "mucho" o "bastante" la intención de voto (Gallardo-Paúls, 2016)[5]. En segundo lugar, la información generada en las redes sociales también tiene un gran impacto a nivel social. ¿Quiénes son los usuarios más activos? ¿Qué género predomina en las redes sociales? ¿Qué edad tienen los usuarios?. Cabe resaltar también el impacto que tienen las redes sociales a nivel económico, ya que estas plataformas se han convertido en un lugar donde las empresas publicitan sus productos y servicios y a las que destinan gran parte de sus presupuestos en publicidad (Saxena et al., 2013)[6]. Finalmente, destacar el impacto que tienen las redes sociales a nivel de seguridad, ya que estas plataformas se han convertido en un lugar donde se comparte información personal, y donde se pueden cometer delitos como el *cyberbullying* o el *grooming* (Machimbarrena et al., 2018)[7].

## 1.2 Perfilado de autor

Surge de esta forma la necesidad de analizar toda la información que se genera y proporcionar así herramientas útiles en todos los niveles vistos en la Sección 1.1. En este contexto, el perfilado de autor en redes sociales se ha convertido en un área de investigación de gran interés en los últimos años, posicionándose como una herramienta de creciente importancia en áreas como la seguridad, el *marketing* o la investigación forense (Rangel et al., 2013)[8].

El perfilado de autor, también conocido como *author profiling* en inglés, consiste en determinar, a partir de un texto, las características de su autor, como su género, edad, rasgos personales, etc. Para ello se hace uso de diversas técnicas de aprendizaje automático basadas en NLP (del inglés *Natural Language Processing*), que permiten extraer características lingüísticas del texto y utilizarlas para una posterior clasificación. Así, el perfilado de autor sería una herramienta de gran ayuda a la hora de realizar análisis sociales sobre temas específicos; sería muy práctico para empresas que quieran conocer el perfil de los clientes que opinan de forma positiva y negativa de sus productos; tendría un gran valor para los partidos políticos, dado que podrían conocer cuál es el perfil de sus votantes; y sería también de gran utilidad para temas como la evaluación sospechosos teniendo en cuenta su perfil lingüístico.

# Estado del arte del perfilado de autor

---

Durante los inicios del perfilado automático de autor, los algoritmos se centraban en la tarea de la clasificación por género. En esta línea, trabajos como (Koppel et al., 2002)[9] se desmarcaban de la tendencia de la época, la cual se basaba en la clasificación de textos en base a su contenido, para centrarse en la clasificación de textos **en base a su estilo**. En este caso, se centraban en la obtención del género del autor mediante el análisis de 920 documentos de carácter formal escritos en inglés con una media de alrededor de 34.300 palabras cada uno, obteniendo una precisión en la clasificación de aproximadamente el 77%.

Así, la demostración de la existencia de rasgos diferenciadores en la escritura que permitían perfilar ciertos aspectos del individuo, especialmente del género, supuso un gran avance en el campo del perfilado de autor y dió pie a la realización de trabajos como (Argamon et al., 2003)[10], (Corney et.al, 2002)[11] o (Otterbacher et al., 2010)[12], así como también permitió el inicio de una clasificación más compleja en base a otras características como la edad, la orientación sexual o la personalidad.

Más tarde, en el año 2011 se celebraría el primer evento organizado por el PAN (*Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection*) [13], un foro de investigación que organiza eventos científicos y tareas anuales relacionadas con el análisis forense de textos digitales y rasgos estilométricos. La primera de estas tareas centrada en el perfilado de autor se celebraría en el año 2013 (Rangel et al., 2013)[8], en la que se pedía a los participantes que obtuvieran, a partir de una serie de *tweets*, la edad y el género de su autor. El ganador de este concurso obtuvo una precisión del 60% en la clasificación de género y del 67% en la clasificación de edad, haciendo uso, la mayor parte de los participantes, de técnicas de aprendizaje supervisado como los Árboles de Decisión (en inglés *Decission Trees*) o las Máquinas de Soporte Vectorial (en inglés *Support Vector Machines*) e incluyendo en sus modelos características basadas en el TF-IDF, n-gramas, etiquetas POS o características como el número de emoticonos o la frecuencia de signos de puntuación. En los siguientes años se celebrarían

nuevas ediciones de esta tarea (Rangel et al., 2014[14], Rangel et al., 2015[15], Rangel et al., 2016[16]...), añadiendo nuevas sub-tareas como el reconocimiento de rasgos personales, la ocupación o los dialectos del lenguaje, así como también alcanzando mejores resultados en la clasificación.

## 2.1 Algoritmos supervisados

TODO

# Herramientas, técnicas y lenguajes

---

A lo largo de este capítulo se describirán las herramientas utilizadas para el desarrollo del proyecto y se expondrán las razones de su uso. A su vez, para una mejor estructuración, dividiremos dichas herramientas en cuatro grupos diferentes: *frontend*, *backend*, algoritmos de perfilado y herramientas de soporte.

### 3.1 *Backend*

Ya que para el *backend* no se necesitaba nada excesivamente complejo, se decidió utilizar el lenguaje de programación Python [17] junto con el *framework* FastAPI [18], el cual permite crear APIs REST de forma sencilla. La decisión de utilizar Python, viene condicionada por el hecho de que los algoritmos de perfilado de autor utilizados, así como también la mayor parte de los algoritmos de aprendizaje automático y procesamiento de lenguaje natural, están ya programados en Python, evitando así crear nuevos *endpoints*, *sockets* o *bindings* para la ejecución de dichos algoritmos. Destacar también que el desarrollo ágil en Python favorecía mucho al trabajo debido a su tipado dinámico, a su ejecución interpretada y a su sintaxis sencilla.

En cuanto a la persistencia de los datos, se optó por MongoDB [19], una base de datos NoSQL (del inglés *Not Only SQL*) que permite almacenar datos en formato JSON (del inglés *JavaScript Object Notation*). La decisión de utilizar una base de datos NoSQL sobre otras opciones como puede ser PostgreSQL [20] o MySQL [21], se debe a dos factores principales. Uno de ellos es la fácil implementación de MongoDB en Python haciendo uso de la librería PyMongo [22], dado que permite realizar operaciones sobre las colecciones de forma muy intuitiva. El otro factor tiene que ver con la extensibilidad de la aplicación a largo plazo, es decir, frente a la creación de nuevas funcionalidades, campos o relaciones, MongoDB permitiría integrarlos gracias a la flexibilidad que ofrece sobre los esquemas de datos, algo imposible si se emplease una base de datos relacional

## 3.2 Frontend

En cuanto al *frontend*, se decidió utilizar NextJS [23] como herramienta para el desarrollo de la interfaz de usuario, dado que ya se contaba con bastante experiencia previa en su uso. NextJS es un *framework* basado en React [24], es decir, en la construcción de interfaces dinámicas e interactivas mediante la composición de elementos que pueden tener estado. Además, este *framework* implementa varias mejoras sobre React como por ejemplo el *server-side rendering*, algo que ayuda en gran medida al SEO (del inglés *Search Engine Optimizations*), esto es, a que los motores de búsqueda como Google puedan indexar mejor la página y, por tanto, que esta aparezca en una posición superior en los resultados de búsqueda. Además, también cuenta con optimizaciones para la carga y el renderizado de imágenes o fuentes entre otras.

Todo ello se desarrolló utilizando TypeScript [25], un lenguaje de programación que añade tipado estático a JavaScript y que está ganando mucha popularidad con respecto a la mantenibilidad, compresión y escalabilidad que proporciona a los proyectos en los que se usa (Stack Overflow, 2023) [26]. A mayores, para garantizar la consistencia de todas las entidades, y más en específico de los DTOs (del inglés *Data Transfer Object*), se utilizó Zod [27], una librería que permite definir esquemas de validación de datos, incluyendo el tipo específico de cada campo.

En cuanto al estilado de la página, se optó por emplear SASS (del inglés *Syntactically Awesome Style Sheets*) [28], un preprocesador de CSS (del inglés *Cascading Style Sheets*) que añade funcionalidades extra como son el uso de variables, bucles o anidamiento de clases. Además, dado que se estaba desarrollando una aplicación novedosa, se buscó crear un estilo propio haciendo uso de CSS "nativo", desmarcándose así de librerías que proporcionasen estilos predefinidos como Bootstrap [29] o componentes ya implementados como Material UI [30] o Chakra UI [31]. Por otra parte, para la creación de gráficos se utilizó la librería ChartJS [32], una de las más conocidas y con más soporte en la actualidad. Finalmente, para implementar las animaciones en la interfaz, se hizo uso, en conjunto con las transiciones nativas de CSS, de Framer Motion [33], una librería que permite crear animaciones de una mayor complejidad desde JavaScript/TypeScript.

## 3.3 Algoritmos de perfilado

A pesar de que los algoritmos de perfilado se ejecutan en el *backend*, se ha decidido incluirlos en esta sección debido a que se trata de una parte importante y característica del proyecto. Decir también que en este caso, las herramientas utilizadas estaban condicionadas,

lógicamente, por aquellas utilizadas en las implementaciones de ambos algoritmos seleccionados.

En cuanto a la parte nuclear del aprendizaje automático, se utilizó Scikit-Learn [34], una librería de Python que proporciona una gran cantidad de algoritmos pre-implementados, así como también herramientas para la extracción de características o la validación de modelos. Además, se hizo uso de otras librerías como Tqdm [35], la cual permite mostrar el progreso de entramiento o predicción de forma visual, Pickle [36], que permite serializar objetos Python (en nuestro caso los modelos ya entrenados), o NumPy [37], una librería que proporciona estructuras de datos y herramientas para el cálculo científico.

### 3.4 Soporte

En lo que respecta a la gestión de tareas, debido a que nuestro ciclo de desarrollo era ágil, se utilizó Trello [38], una herramienta que permite crear tableros con listas de tareas, las cuales pueden ser movidas entre ellas según su estado: pendiente, en progreso o completada.

Además, debido a la importancia de mantener un historial sobre los cambios realizados en el código, se optó por utilizar Git [39] como herramienta de control de versiones junto con GitHub [40], una plataforma que permite almacenar repositorios de Git de forma remota, similar a otras como GitLab [41] o BitBucket [42].

En cuanto al entorno de desarrollo o IDE (del inglés *Integrated Development Environment*), se utilizó Visual Studio Code [43], un editor gratuito y de código semi-abierto desarrollado por Microsoft, el cual brinda una gran flexibilidad para trabajar con cualquier lenguaje de programación gracias a sus extensiones. Esto hecho, posibilitó el desarrollo del *backend*, del *frontend* e incluso de esta memoria en un mismo programa, simplificando la tarea de aprender las peculiaridades de otros IDEs más específicos. Más en concreto, se utilizaron extensiones como Prettier [44] o Black [45], las cuales permiten formatear el código de forma automática para JavaScript o Python, respectivamente; ESLint [46], una herramienta que permite detectar errores en el código JavaScript/TypeScript; o GitLens [47], una extensión que añade funcionalidades extra con respecto al control de cambios.

En relación a la elaboración de los diagramas y los *wireframes* que aparecen en este trabajo, se utilizó la herramienta Draw.io [48], la cual permite crear todo tipo de diagramas *online* de forma gratuita, sin la obligación de registrarse y sin la necesidad de instalar ningún programa, pudiendo exportarlos en varios formatos, entre ellos SVG. Junto a esta herramienta, se utilizó también la librería pgfplots [49] para la creación de las gráficas en el propio  $\text{\LaTeX}$ .

Finalmente, para facilitar la puesta en marcha de todas las partes que conforman el sistema y evitar instalar todas las dependencias de forma manual, se utilizó Docker [50], una herramienta que permite crear contenedores , es decir, entornos de ejecución aislados que contienen todo lo necesario para que una aplicación funcione correctamente.



## Capítulo 4

# Metodología

---

Para el desarrollo de este proyecto se ha utilizado una metodología ágil, concretamente Scrum [51]. Esta metodología se basa en la realización de iteraciones cortas, llamadas *sprints*, en las que se desarrolla una parte del proyecto denominada incremento, es decir, una versión entregable del producto que contiene nuevas funcionalidades o mejoras de las ya existentes. La decisión de optar por una metodología de tipo ágil frente a una tradicional está condicionada por el tiempo de desarrollo corto, de apenas cuatro meses, y por los cambios que se podían producir en los requisitos. Además, todo el equipo se sentía más cómodo con esta metodología dado que ya se tenía experiencia anterior en su uso.

### 4.1 Roles

En Scrum existen tres roles principales:

- **Product Owner:** Como su nombre indica, es el propietario del producto, por lo que es el encargado de identificar las necesidades de los clientes así como de definir y gestionar el *Product Backlog*, esto es, la lista de requisitos del producto ordenados por prioridad. Este rol es desempeñado por el autor de este documento.
- **Scrum Master:** Su rol principal es el de asegurar que el equipo de desarrollo sigue la metodología Scrum y no se producen desajustes en el transcurso del proyecto. Los encargados de asumir este rol fueron Patricia Martín Rodilla y David Otero Freijeiro, los tutores de este trabajo.
- **Equipo de desarrollo:** Formado por un equipo normalmente de entre tres y nueve personas, es el encargado de desarrollar el producto en cada *sprint* cumpliendo los requisitos establecidos en el *Product Backlog*. En este caso, el equipo de desarrollo está formado únicamente por el autor de este documento.

## 4.2 Eventos

Para agilizar el desarrollo y no interferir en el trabajo diario del equipo por las obligaciones externas de cada uno, tanto las reuniones de planificación como las de revisión y retrospectiva se realizan el mismo día que se inicia cada *sprint*, esto es, aproximadamente cada tres semanas, lo que se ajusta a la duración recomendada por la metodología Scrum. Estas reuniones son, por lo tanto, de una mayor extensión que las *dailies* clásicas y son también empleadas para realizar una revisión del incremento desarrollado en el último *sprint*.

## Capítulo 5

# Análisis

---

En este capítulo se expondrán los requisitos obtenidos tras el estudio de las necesidades que debe cubrir la aplicación y se elaborarán las historias de usuario asociadas.

### 5.1 Requisitos funcionales

Para la definición de los requisitos funcionales que debe cumplir la aplicación, se ha hecho uso de las historias de usuario. Esta técnica permite definir los requisitos de una forma más cercana al usuario, ya que se centra en la descripción de las funcionalidades que este desea que tenga el sistema. Con todo, las historias de usuario obtenidas se muestran en la [Tabla 5.1](#).

ID	Historia de usuario
1	Como usuario quiero poder subir mi propio dataset de textos para su perfilado
2	Como usuario quiero conocer ejemplos del formato de los datasets aceptados
3	Como usuario quiero poder seleccionar el algoritmo de perfilado que se va a utilizar
4	Como usuario quiero poder visualizar los datos obtenidos tras el perfilado
5	Como usuario quiero poder ver una lista detallada de cada persona perfilada
6	Como usuario quiero poder ordenar la lista de personas por cada uno de los campos perfilados
7	Como usuario quiero poder saber como funcionan los diferentes algoritmos de perfilado disponibles
8	Como usuario quiero ver el rendimiento de los algoritmos de perfilado disponibles
9	Como usuario quiero poder reentrenar los algoritmos con diferentes datasets

Tabla 5.1: Historias de usuario

A partir de estas historias de usuario se ha obtenido el diagrama de casos de uso que se muestra en la Figura 5.1.

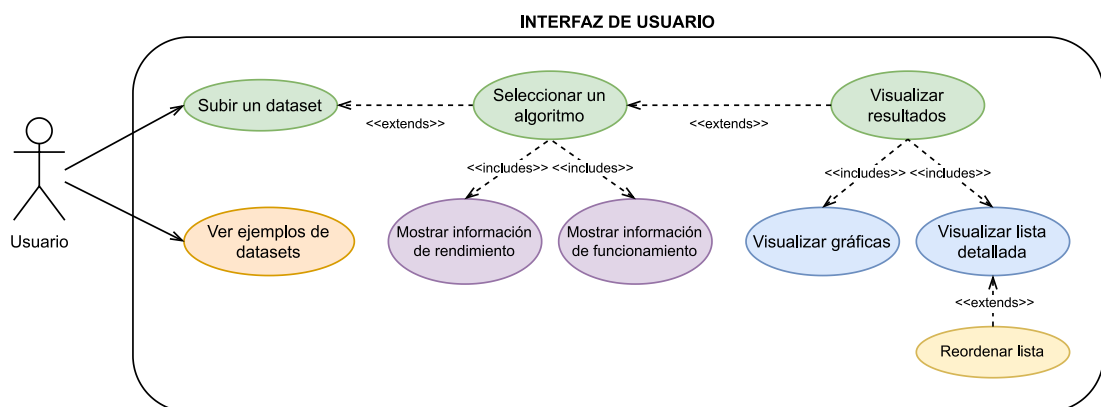


Figura 5.1: Diagrama de casos de uso

## 5.2 Requisitos no funcionales

Una vez definidas las funcionalidades que debe tener la aplicación, es necesario definir cuales van a ser sus requisitos no funcionales, esto es, aquellos que están relacionados en cómo debe funcionar la aplicación. Para ello, se han tenido en cuenta los aspectos recogidos en la Tabla 5.2.

Requisito	Descripción
<i>Usabilidad</i>	La aplicación debe ser fácil de usar, de forma que cualquier usuario pueda utilizarla sin necesidad de tener conocimientos previos sobre el perfilado de autores
<i>Escalabilidad</i>	La aplicación debe ser capaz de procesar datasets de cualquier tamaño, de forma que no exista un límite en el tamaño de los mismos
<i>Portabilidad</i>	La aplicación debe ser capaz de ejecutarse en cualquier sistema, de forma que los usuarios no tengan que preocuparse por el dispositivo que utilizan

Tabla 5.2: Requisitos no funcionales

## Capítulo 6

# Diseño

---

Para abordar el diseño de la aplicación a lo largo de este capítulo, se detallará la arquitectura software elegida para el sistema y se mostrarán los prototipos de la interfaz de usuario.

### 6.1 Arquitectura

Teniendo en cuenta los requisitos definidos en las Secciones 5.1 y 5.2, así como también las limitaciones temporales del proyecto, se ha optado por una arquitectura cliente-servidor. Esta arquitectura se ha elegido por la facilidad en su implementación y por su capacidad de intercomunicación con otros sistemas, especialmente con clientes web. Como se aprecia en la Figura 6.2, el servidor cuenta con un controlador REST capaz de redirigir las peticiones al servicio correspondiente. En este sentido, a pesar de solo contar con un único servicio (*Servicio de perfilado*) en la versión actual, la arquitectura cliente-servidor permite añadir nuevos servicios, junto a nuevas funcionalidades, de forma sencilla y sin poner en riesgo al resto del sistema. Dicho servicio será el que se comunique con la base de datos para garantizar la persistencia de los datos del perfilado y permitir así una posterior consulta.

Por otro lado, para que el código estuviese bien organizado, se decidió utilizar el patrón de diseño DDD (del inglés *Domain-Driven Design*) [52], el cual permite separar el código en tres capas: la capa de aplicación, la capa de dominio y la capa de infraestructura. La capa de aplicación es la encargada de gestionar la entrada y salida de la aplicación que, en nuestro caso, es el controlador que maneja los *endpoints* REST; la capa de dominio es la que contiene la lógica de negocio y las entidades; y la capa de infraestructura es la responsable de administrar las interacciones internas de la aplicación que, en nuestro caso, se encarga de la comunicación con la base de datos.

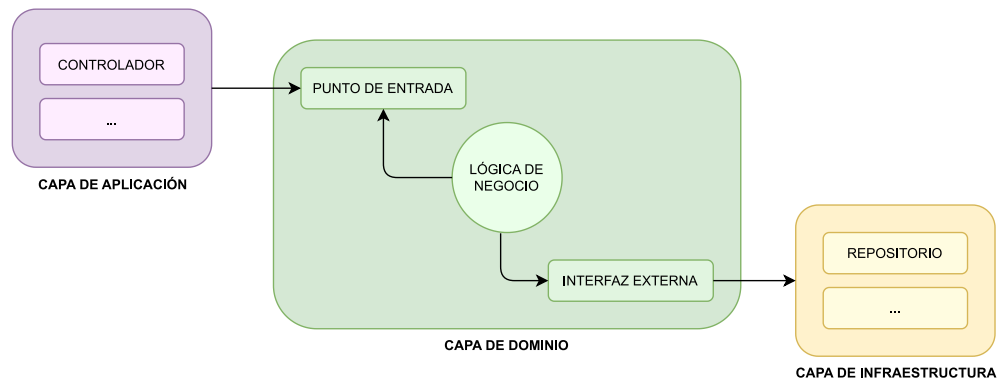


Figura 6.1: Diagrama de capas del patrón de diseño DDD. Adaptado de Ł, Ryś [1]

A mayores, el propio *frontend* web tendrá también una arquitectura basada en cliente-servidor, en la que el servidor será el encargado de ofrecer al cliente los archivos estáticos necesarios para mostrar la interfaz (HTML, CSS, JavaScript, fuentes, imágenes...). El hecho de elegir una interfaz web gira en torno al requisito no funcional de portabilidad definido en la Sección 5.2, puesto que de esta forma, la aplicación podría ser utilizada por cualquier dispositivo con un navegador web, evitando desarrollar aplicaciones nativas para cada sistema operativo.

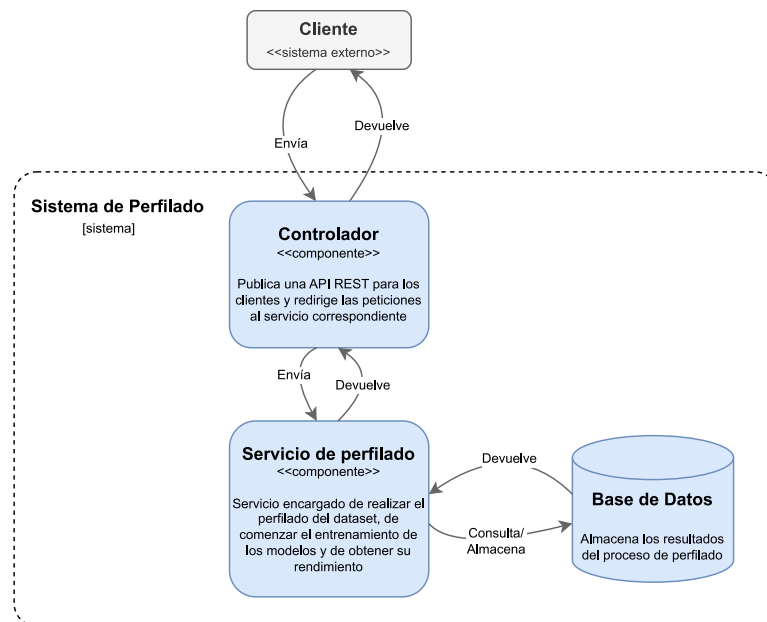


Figura 6.2: Diagrama C4 de Contenedor de la arquitectura del sistema.

## 6.2 Prototipado

Como se mencionó anteriormente, para el diseño de los prototipos de pantallas, también conocidos como *wireframes*, se utilizó la herramienta Draw.io [48] debido a su sencillez y rapidez en la elaboración con respecto a otros programas más avanzados como pueden ser Adobe XD [53] o Figma [54].

La filosofía de diseño de la interfaz se centró en el minimalismo y la usabilidad, como se indica en la Sección 5.2, por lo que todo estará caracterizado por no contener excesivos elementos y por ser fácilmente comprensible para el usuario.

La pantalla de inicio, como se ve en la Figura 6.3, está compuesta simplemente por un título, un subtítulo y un campo que permitirá al usuario subir un *dataset*, tanto mediante un elemento *drag and drop* como mediante un botón que abrirá el explorador de archivos del sistema operativo.

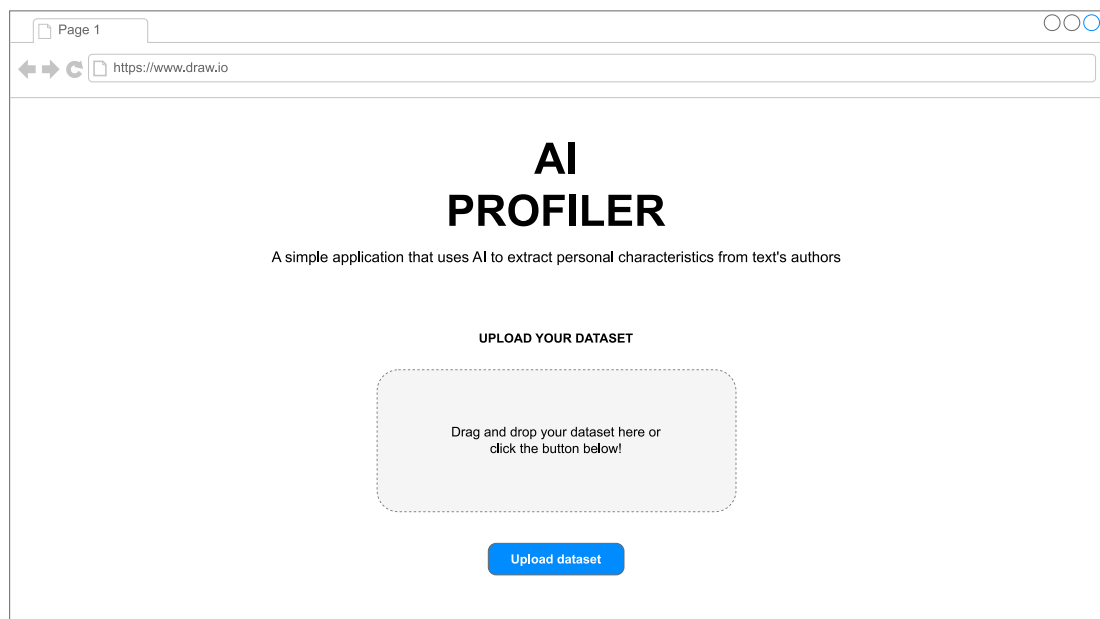


Figura 6.3: Prototipo de la pantalla de inicio

Una vez el usuario haya subido el *dataset*, el campo de subida se sustituirá por la lista de los algoritmos de perfilado disponibles, como se puede ver en la Figura 6.4. De cada algoritmo se mostrará, en una tarjeta, las características que es capaz de extraer del perfilado junto con su título. Además, ya que uno de los requisitos era el de mostrar información del funcionamiento y del rendimiento de los algoritmos, se decidió crear un *tooltip* que se mostrase cada vez que el



usuario pasase el ratón por encima de cada tarjeta. Finalmente, si el usuario deseara cambiar el *dataset* seleccionado, se permitirá retroceder mediante la flecha situada junto al título del paso actual.

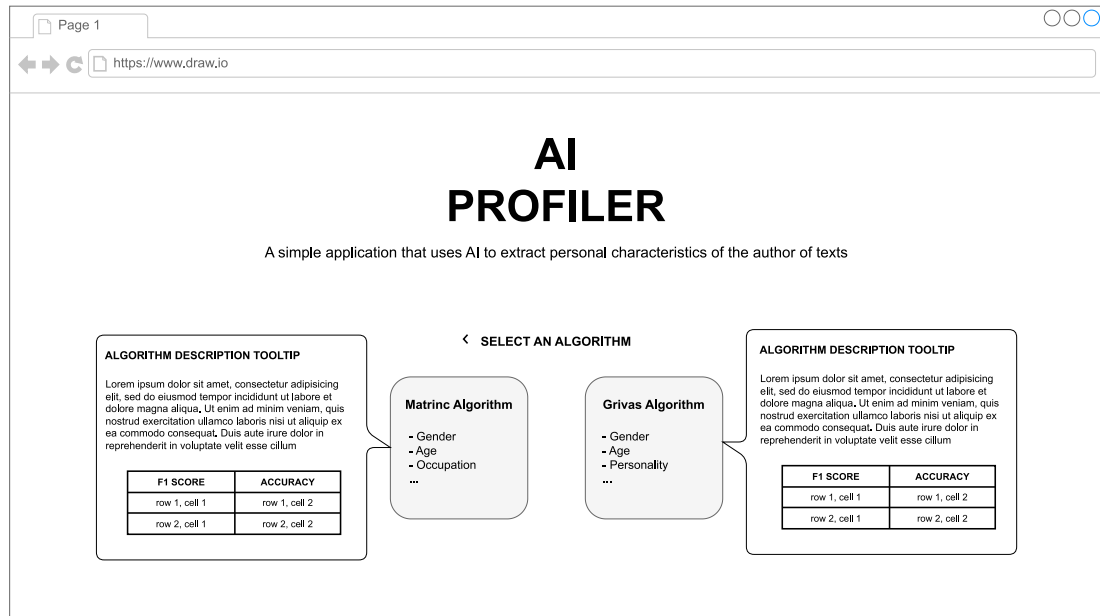


Figura 6.4: Prototipo de la selección del algoritmo de perfilado

Ya con el *dataset* y el algoritmo seleccionados, se presentará al usuario un resumen del perfilado, donde se mostrará el nombre del fichero subido, la tarjeta del algoritmo seleccionado y un botón que permitirá al usuario comenzar con el proceso de perfilado. Además, como se aprecia en la Figura 6.5, una vez comienza el perfilado, se mostrará una barra de progreso o un *spinner* que le indicará al usuario que el proceso está en marcha. De la misma forma que en el paso anterior, si el usuario decide cambiar de algoritmo, puede retroceder haciendo uso de la flecha situada junto al título del paso actual.

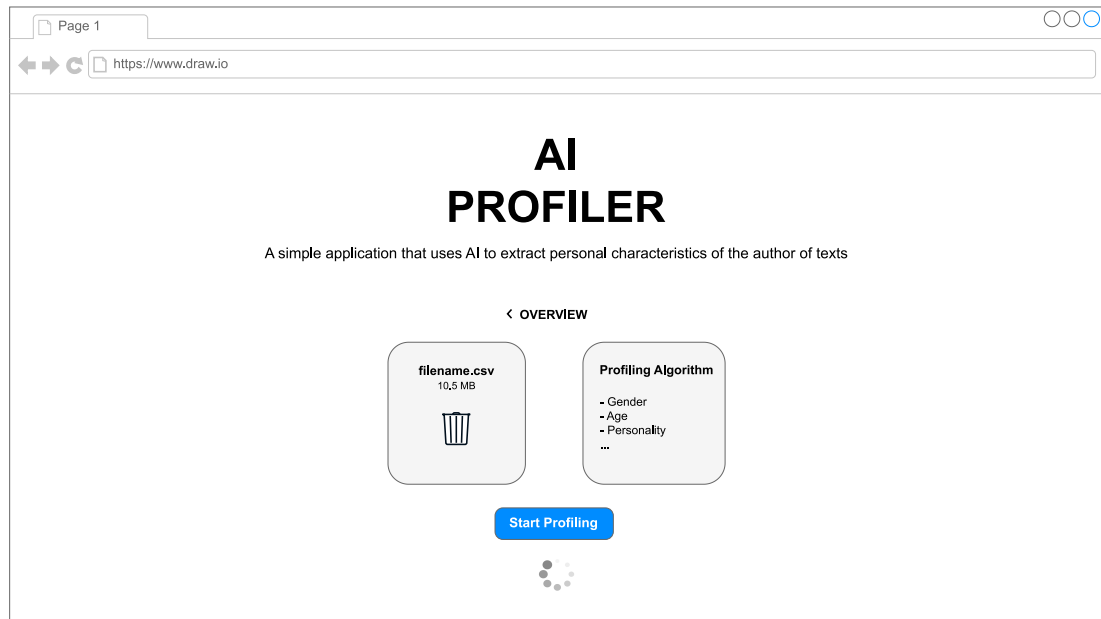


Figura 6.5: Prototipo del resumen del perfilado

Con respecto a la pantalla de ejemplos de *datasets*, como se puede ver en la Figura 6.6, se mostrará un pequeño ejemplo de 10-15 líneas aproximadamente. Asimismo, se le dará la opción al usuario de seleccionar el formato de *dataset*, ya sea NDJSON o CSV, dado que son los que soportará el *backend*.

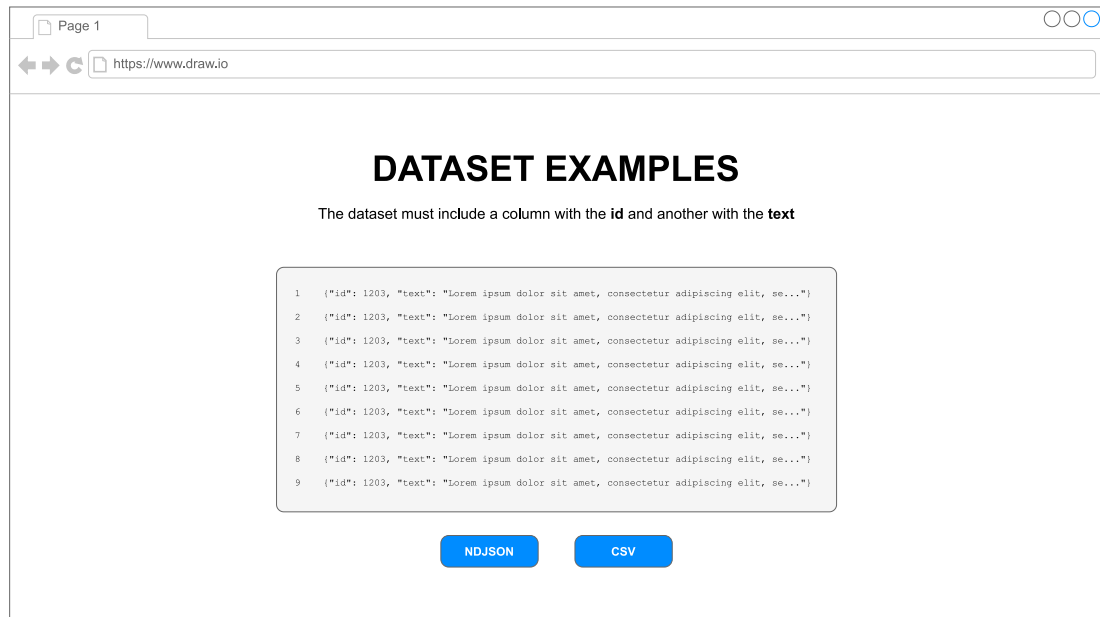


Figura 6.6: Prototipo de la pantalla de ejemplos de dataset

Cuando el perfilado haya finalizado, se mostrará al usuario un *dashboard* con los resultados obtenidos, como se puede ver en las Figuras 6.7, 6.8. Este *dashboard* contendrá los siguientes gráficos y elementos, los cuales variarán en función del algoritmo utilizado:

- **Datos generales:** El *dashboard* contiene una primera sección en la que aparece información de carácter general como son: el número total de personas perfiladas, el tiempo total del perfilado y el algoritmo utilizado.
- **Lista detallada de personas:** En esta sección, se muestra la lista de personas perfiladas de forma paginada. Además, la lista permitirá ser ordenada por cada uno de los diferentes campos, según se indica en la historia de usuario con identificador 6 de la Sección 5.1. Para ello, el usuario deberá hacer clic en el nombre de dicho campo, teniendo la opción de, volviendo a clicar, cambiar el sentido de ordenación (ascendente o descendente).
- **Distribución de edad:** Para la distribución de edad, se ha optado por un gráfico de barras sobre otras opciones de representación categóricas como el gráfico circular. Esto se debe a que, teniendo cinco clases diferentes, el gráfico de barras permite una mejor visualización de los datos y una comparativa más clara entre ellos, dado que es más fácil comparar longitudes que áreas o ángulos. En la parte inferior del gráfico, se mostrará una tarjeta con la edad mediana.

- **Distribución de género:** En cuanto a la distribución de género, en cambio, se ha optado por un gráfico circular, puesto que solo existen dos clases y se desea resaltar la proporción de cada clase con respecto al total. A mayores, se mostrarán debajo del gráfico dos tarjetas con el número de personas de cada género junto con su porcentaje exacto.
- **Distribución de fama (*Martinc*):** De forma similar a la distribución de género, solo contamos con tres clases diferentes, por lo que se ha elegido un gráfico circular. Resaltar que este gráfico solo se mostrará en el caso de que se haya utilizado el algoritmo de Martinc.
- **Distribución de ocupación (*Martinc*):** En el caso de la distribución de ocupación, debido a que existen ocho clases distintas, se ha optado por un gráfico de barras. Este gráfico, al igual que el anterior, solo aparecerá si se ha empleado el algoritmo de Martinc para el perfilado.
- **Características personales (*Grivas*):** Ya que en este caso cada característica personal tendría asociado un valor decimal entre -0.5 y 0.5, se ha optado por un gráfico de barras. En este sentido, cabe resaltar que para mostrar dicho gráfico, es necesario implementar una funcionalidad que permita seleccionar a una persona de la lista detallada para mostrar sus características personales. Además, este gráfico solo estará disponible si se hace uso del algoritmo de Grivas.

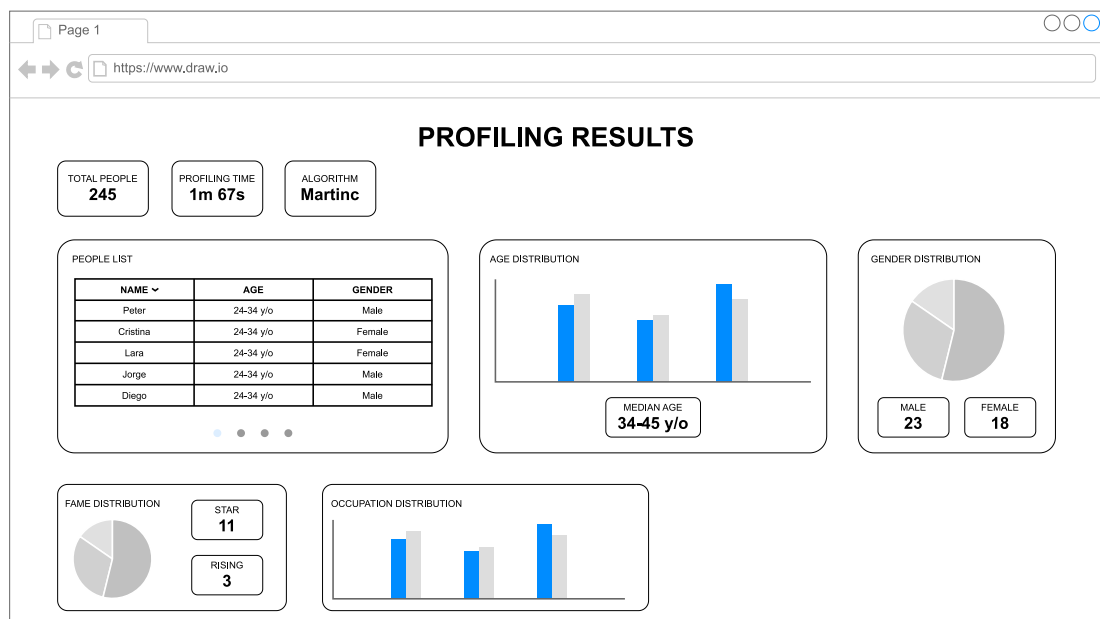


Figura 6.7: Prototipo del dashboard utilizando el algoritmo Martinc

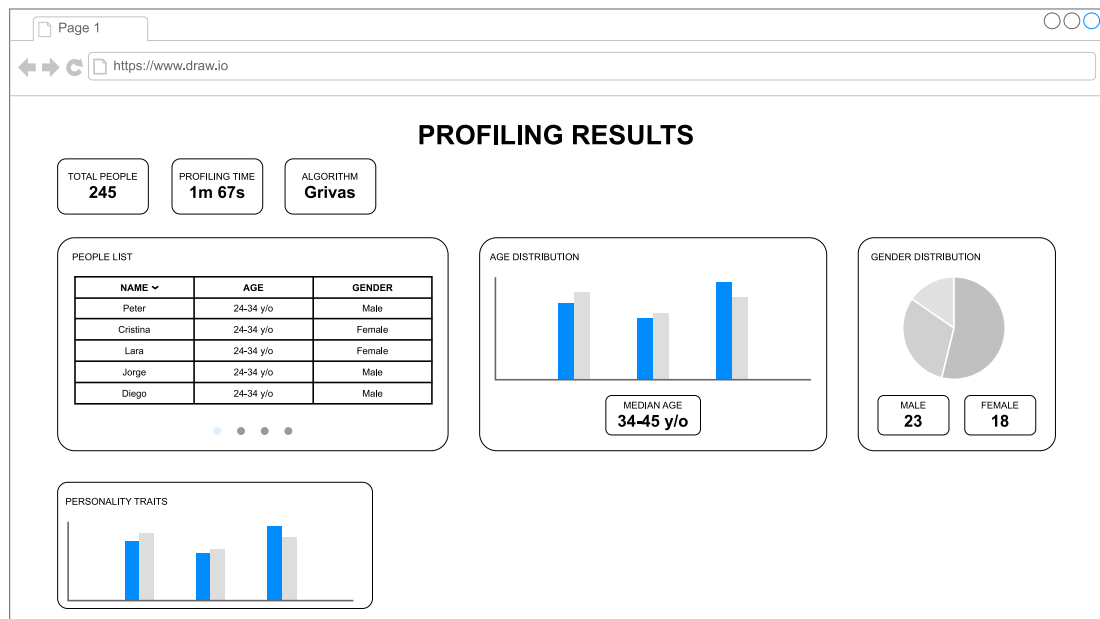


Figura 6.8: Prototipo del dashboard utilizando el algoritmo Grivas

# Implementación

---

A lo largo de este capítulo, se expondrán las decisiones de implementación más relevantes tomadas durante el desarrollo del proyecto.

## 7.1 *Backend*

Para analizar la implementación del *backend*, se tomará como referencia el diagrama de clases de la Figura 7.3 y se profundizará en las clases más importantes que lo componen. Asimismo, se estudiará y se justificará cuales han sido los *endpoints* expuestos por el servidor, representados en la Figura 7.2.

### 7.1.1 *Endpoints*

Dado que el *backend* seguirá una arquitectura REST, se expondrán los *endpoints* que se han implementado para el servidor, así como los parámetros que reciben. Como se puede observar en la Figura 7.2, el servidor expone un total de cuatro *endpoints*:

- **/predict**: Como indica la historia de usuario con identificador 1 de la Sección 5.1, el usuario debe poder subir su propio dataset de textos para su perfilado. Para ello, se ha implementado este *endpoint* que recibe como parámetros de la URL el nombre del algoritmo de perfilado a utilizar junto al *dataset* utilizado para entrenar el modelo. Además, ya que el método de la petición es POST, en el cuerpo de la petición será donde se envíe el archivo con los textos y usuarios a perfilar. En caso de que el archivo no sea correcto, que el algoritmo no exista o que el *dataset* de entrenamiento no sea válido, se devolverá un error HTTP 400 (Bad Request). En caso contrario, se devolverá un identificador correspondiente a la tarea de perfilado que se ha creado de forma asíncrona, generado dinámicamente con la librería uuid [55] de Python (según el estándar RFC 4122 [56]) y que coincide con el identificador del documento almacenado en la base de datos. El

hecho de no esperar a que el perfilado se complete para devolver una respuesta al usuario evita tener problemas con el tiempo de espera de la petición o *timeout* del *socket* TCP, permitiendo así procesar grandes volúmenes de datos y cumplir con el requisito de escalabilidad establecido en la Sección 5.2.

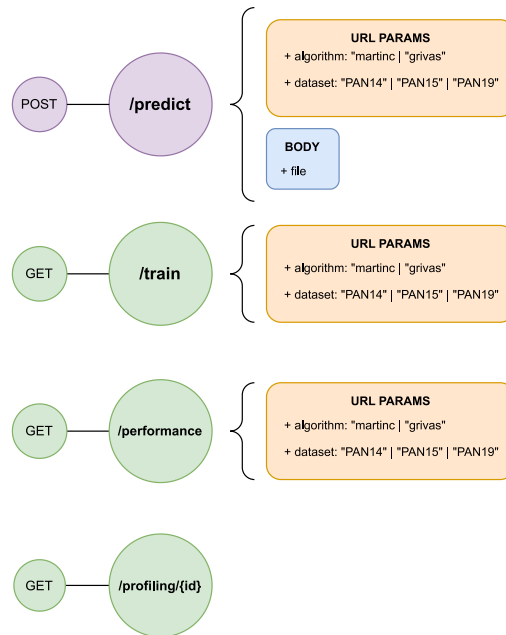
- **/train:** Además, ya que el usuario debe poder reentrenar los modelos con los algoritmos disponibles, se ha implementado este *endpoint* de tipo GET. En este caso, se recibe como parámetros de la URL el nombre del algoritmo de perfilado a utilizar junto al *dataset* de entrenamiento. De la misma forma, se validarán los parámetros y, en caso de que sean correctos, se iniciará una tarea asíncrona en el *backend*.
- **/performance:** Puesto que otra de las historias de usuario nos indica que es necesario conocer el rendimiento que tiene los algoritmos, el *backend* expondrá otro *endpoint* de tipo GET que recibe los mismos parámetros que el anterior. La diferencia es que, en este caso, se devolverá un JSON con los valores de rendimiento del algoritmo en lugar de iniciar una tarea asíncrona.
- **/profilings/{id}:** Para poder obtener los resultados del perfilado creado de forma asíncrona, este *endpoint* de tipo GET permite recuperarlos de la base de datos haciendo uso del identificador devuelto por la tarea de predicción. En caso de que la tarea no exista, se devolverá un error HTTP 404 (Not Found). La respuesta devuelta será un JSON con la estructura mostrada en la Figura 7.1.

```
1 {
2   "status": "SUCCESS",
3   "algorithm": "martinc",
4   "time": 4291,
5   "output": [
6     {
7       "id": "29502",
8       "result": {
9         "gender": "male",
10        "fame": "star",
11        "occupation": "sports",
12        "age": "35-49"
13      }
14    },
15    {
16      "id": "38991",
17      "result": {
18        "gender": "female",
19        "fame": "rising",
20        "occupation": "performer",
21        "age": "50-XX"
22      }
23    },
24  ]
25 }
```

Figura 7.1: Estructura del JSON devuelto por el *endpoint* /profilings/{id}

Destacar que en todos los casos, tanto el algoritmo de Grivas como el de Martinc, tienen asociado un *dataset* de entrenamiento por defecto (que coincide con el utilizado por los autores originales para su publicación), el cual se utilizará en caso de no especificar ninguno en la URL.



Figura 7.2: Diagrama de *endpoints* del *backend*

### 7.1.2 Clases

Como se explicó en la Sección 3.1, la estructura del *backend* sigue los principios del patrón de diseño DDD [52], en el que se distinguen tres capas: la capa de aplicación, la capa de dominio y la capa de infraestructura.

Nuestro punto de entrada al sistema, es decir, el componente que forma parte de la **capa de aplicación**, es la clase *Controller*, el cual se encarga de realizar las siguientes tareas:

- Exponer los *endpoints* y recibir los parámetros de entrada, ya sea a través de la URL o del cuerpo de la petición haciendo uso de FastAPI [18].
- Validar dichos parámetros y devolver los errores HTTP correspondientes en caso de que no sean correctos.
- Parsear los parámetros de entrada a los tipos de datos que se necesiten.
- Realizar la llamada al servicio de la capa de dominio y devolver el valor de retorno encapsulado en una respuesta HTTP.

En la **capa de dominio**, una vez los datos han sido validados y parseados, el *ProfilingService* es el encargado de orquestrar las diferentes llamadas y delegar la lógica de negocio a las distintas entidades.

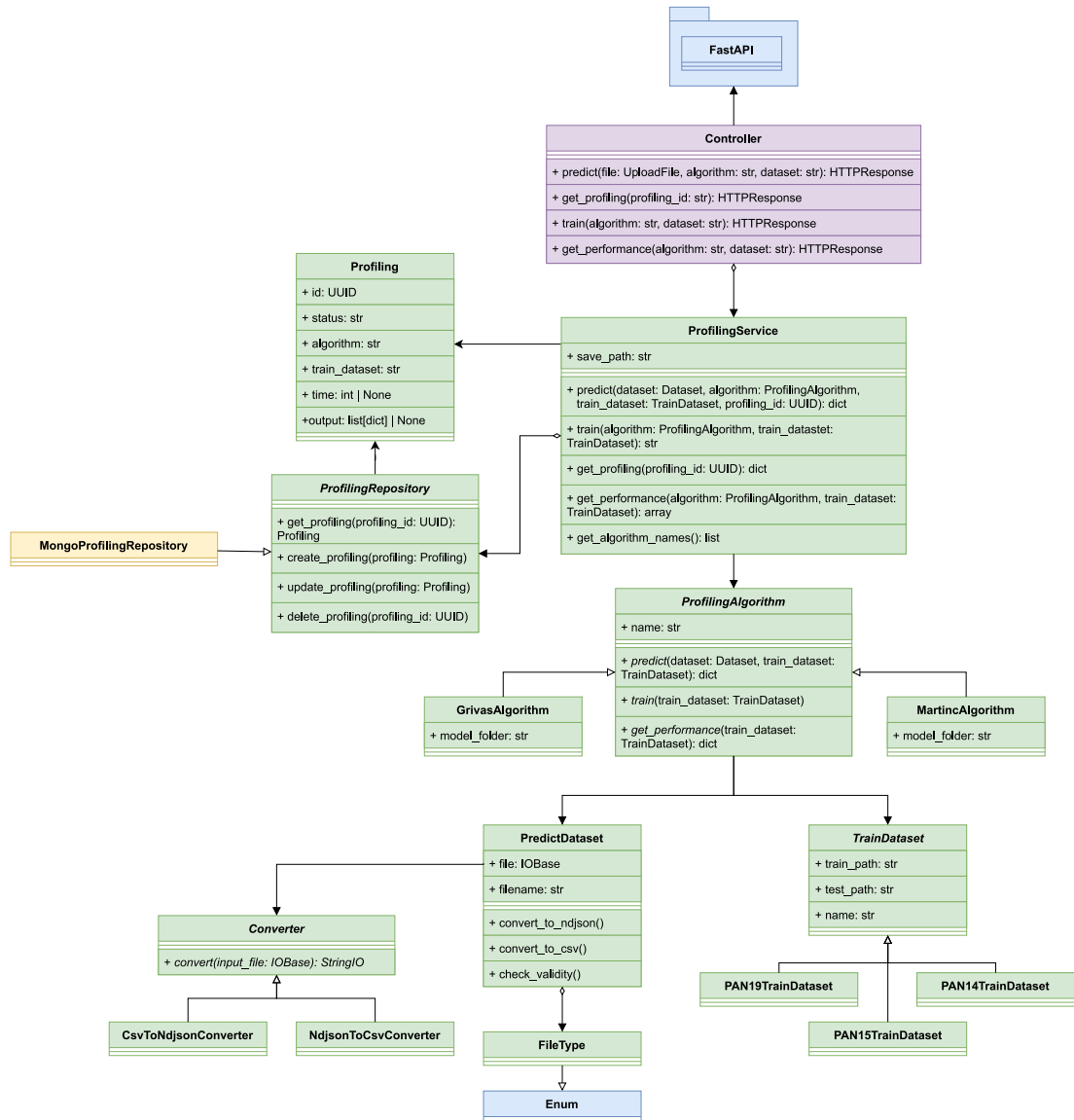
La más importante de ellas es la clase *ProfilingAlgorithm*, una clase abstracta que define la interfaz que deben implementar todos los algoritmos de perfilado incluyendo las tres funciones principales: *predict()*, *train()* y *get\_performance()*. En este sentido, es importante mencionar que los algoritmos elegidos no estaban implementados pensando en formar parte de una aplicación más grande, por lo que fue necesario realizar una adaptación manual para cumplir con la interfaz de la clase heredada, lo que implicó comprender a muy bajo nivel como estaban implementados dichos algoritmos. Esta adaptación del código fue, en el caso del algoritmo de Grivas, bastante compleja, ya que conllevó sustituir librerías obsoletas, reimplementar funciones y actualizar la sintaxis a la nueva versión de Python.

Otra entidad importante de la capa de dominio es la clase *PredictDataset*, que representa el *dataset* que proporciona el usuario para realizar la predicción. Dado que los algoritmos de perfilado requieren que el *dataset* tenga un formato específico (en este caso ambos solo procesan NDJSON), es necesario implementar conversores que transformen el archivo de entrada a dicho formato por lo que fue necesario crear las clases *CsvToNdjsonConverter* y *NdjsonToCsvConverter*.

También existe una clase llamada *TrainDataset*, que representa el *dataset* que se utiliza para entrenar el modelo y validarlo. Esta clase abstracta contiene la localización de los elementos del conjunto de entrenamiento y test, así como también un nombre que la identifica. Esto nos permite la incorporación de nuevos *datasets* creando simplemente una nueva clase que herede de *TrainDataset* y posibilita la generación de modelos haciendo uso de diferentes conjuntos de entrenamiento de forma sencilla y automática.

Finalmente, en la capa de dominio, se encuentra la clase abstracta *ProfilingRepository*, la cual define la interfaz que deben implementar los repositorios de tecnologías concretas, ya sean relacionales, no relacionales o de otro tipo.

Ya en la **capa de infraestructura**, es decir, donde se almacenan todos los componentes externos con los que interactúa el dominio, se encuentran las clases que implementan la interfaz definida por la clase *ProfilingRepository*. En nuestro caso, ya que se decidió optar por MongoDB como base de datos, solo existe la clase *MongoProfilingRepository*, la cual se encarga de realizar las operaciones CRUD (del inglés *Create, Read, Update, Delete*) sobre la base de datos, así como de establecer y mantener la conexión con la misma.

Figura 7.3: Diagrama de clases del *backend*

## Caso de uso: #BLM

---

# Planificación y costes

---

En este capítulo se expondrá la planificación y distribución del trabajo en cada *sprint*. Asimismo, se detallará el coste asociado a todos los recursos involucrados en el proceso de desarrollo.

## 9.1 Planificación temporal

Con todo, el proyecto se ha dividido finalmente en 6 *sprints* de aproximadamente tres semanas de duración cada uno. Se estima que las horas diarias de trabajo de un desarrollador sean 4 horas de media por lo que, teniendo en cuenta los fines de semana, se obtiene un total de 60 horas por *sprint* y 360 horas en total. En cuanto al *Project Manager*, se estima que dedica 2 horas por *sprint* a la gestión del proyecto, lo que supone un total de 12 horas.

- **Sprint 1:** Durante el primer sprint, del 15 de abril al 5 de mayo, se realizó la instalación del entorno de desarrollo sobre el que se trabajaría posteriormente y se familiarizó con las librerías más comunes de NLP.
- **Sprint 2:** Del 6 al 26 de mayo, se llevó a cabo una investigación de los posibles algoritmos a utilizar junto a su rendimiento e implementación.
- **Sprint 3:** Durante el tercer *sprint*, del 27 de mayo al 16 de junio, se comenzó con el desarrollo de la interfaz de usuario, es decir, diseño de *mockups* e implementación de algunos componentes.
- **Sprint 4:** Del 17 de junio al 7 de julio, se continuó con el desarrollo de la interfaz de usuario y se integraron en el *backend* dos de los algoritmos seleccionados.
- **Sprint 5:** Durante el quinto *sprint*, del 8 al 28 de julio, finalizó el desarrollo de la interfaz web y el desarrollo se enfocó en estructurar el código del *backend* así como de agregar la integración con la base de datos. Además, se comenzó la redacción de esta memoria.

- **Sprint 6:** Finalmente, todo el mes de agosto se centró en la elaboración de la memoria y en pulir los últimos detalles para la entrega de la aplicación.

## 9.2 Recursos y costes

A lo largo del proyecto, se han empleado recursos de tres tipos diferentes: humanos, materiales y software. Como recursos de tipo humano asociados al proyecto contamos con un grupo de trabajo de tres personas, el autor de este documento y los tutores del trabajo, como se mencionó anteriormente en Sección 4.1. En cuanto a los recursos de tipo software, todos los programas, librerías y herramientas en general se han descrito en detalle en el Capítulo 3. Finalmente, en lo que respecta a recursos materiales, se ha hecho uso del portátil personal del autor, cuyas especificaciones se muestran en la Tabla 9.1.

Componente	Modelo
Procesador	Intel Core™ i5-9300H @ 2.40GHz 4C/8T
Memoria RAM	16GB 2667MHz DDR4
Disco duro	512GB SSD NVMe M.2
Tarjeta gráfica	NVIDIA GeForce GTX 1660Ti
Sistema operativo	Arch Linux 6.1.12

Tabla 9.1: Especificaciones del portátil empleado para el desarrollo del proyecto

Dado que todos los recursos de tipo software utilizados con gratuitos no se ha incurrido en ningún coste asociado a ellos. Por otro lado, ya que el portátil no se adquirió específicamente para el desarrollo de este proyecto, no se ha tenido en cuenta en la estimación de costes del proyecto. Por lo tanto, para estimar el coste de los recursos humanos utilizados se ha tenido en cuenta el salario medio de empleados con el mismo puesto en España. Según publica uno de los portales de empleo más populares a nivel mundial, Indeed, el salario medio de un ingeniero de software que desarrolla utilizando Python es de alrededor de 32.100€ anuales [57]. Además, el salario medio de un *Project Manager* del sector de las TIC es de unos 43.500€ anuales. Con estos datos, el coste del proyecto se puede ver detallado en la Tabla 9.2.

Recurso	Coste por hora	Horas	Total
<i>Desarrollador</i>	17.8€/h	360h	6.408€
<i>Project Manager</i>	2 x 24.17€/h	12h	580€
<i>Software</i>	-	-	0€
<i>Materiales</i>	-	-	0€
<i>Total</i>	-	-	<b>6.988€</b>

Tabla 9.2: Coste del proyecto

### 9.3 Viabilidad del proyecto

# Conclusiones

---

## 10.1 Lecciones aprendidas

## 10.2 Trabajo futuro



## **Apéndice**

---

Apéndice A

## Material adicional

---



# Bibliografía

---

- [1] Łukasz Ryś, ``Organizing layers using hexagonal architecture, ddd, and spring," <https://www.baeldung.com/hexagonal-architecture-ddd-spring>, 2022.
- [2] M. We Are Social, ``Digital 2023 global overview report," 2023. [En línea]. Disponible en: <https://datareportal.com/reports/digital-2022-global-overview-report>
- [3] B. Sydney, ``The permanent campaign: Inside the world of elite political operatives," 1980.
- [4] J. Strömbäck, ``Four phases of mediatization: An analysis of the mediatization of politics," *The international journal of press/politics*, vol. 13, no. 3, pp. 228--246, 2008.
- [5] B. Gallardo-Paúls and S. Enguix Oliver, *Pseudopolítica: el discurso político en las redes sociales*. Universitat de València, 2016.
- [6] A. Saxena and U. Khanna, ``Advertising on social network sites: A structural equation modelling approach," *Vision*, vol. 17, no. 1, pp. 17--25, 2013.
- [7] J. M. Machimbarrena, E. Calvete, L. Fernández-González, A. Álvarez-Bardón, L. Álvarez-Fernández, and J. González-Cabrera, ``Internet risks: An overview of victimization in cyberbullying, cyber dating abuse, sexting, online grooming and problematic internet use," *International journal of environmental research and public health*, vol. 15, no. 11, p. 2471, 2018.
- [8] F. Rangel, P. Rosso, M. Koppel, E. Stamatatos, and G. Inches, ``Overview of the author profiling task at pan 2013," in *CLEF conference on multilingual and multimodal information access evaluation*. CELCT, 2013, pp. 352--365.
- [9] M. Koppel, S. Argamon, and A. R. Shimoni, ``Automatically categorizing written texts by author gender," *Literary and linguistic computing*, vol. 17, no. 4, pp. 401--412, 2002.

- [10] S. Argamon, M. Koppel, J. Fine, and A. R. Shimoni, "Gender, genre, and writing style in formal written texts," *Text & talk*, vol. 23, no. 3, pp. 321--346, 2003.
- [11] M. Corney, O. De Vel, A. Anderson, and G. Mohay, "Gender-preferential text mining of e-mail discourse," in *18th Annual Computer Security Applications Conference, 2002. Proceedings.* IEEE, 2002, pp. 282--289.
- [12] J. Otterbacher, "Inferring gender of movie reviewers: exploiting writing style, content and metadata," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 369--378.
- [13] P. Organizers, "Pan (plagiarism analysis, authorship identification, and near-duplicate detection)," <https://pan.webis.de/>, 2023.
- [14] F. Rangel, P. Rosso, I. Chugur, M. Potthast, M. Trenkmann, B. Stein, B. Verhoeven, and W. Daelemans, "Overview of the 2nd author profiling task at pan 2014," in *CLEF 2014 Evaluation Labs and Workshop Working Notes Papers, Sheffield, UK, 2014*, 2014, pp. 1--30.
- [15] F. Rangel, F. Celli, P. Rosso, M. Potthast, B. Stein, W. Daelemans *et al.*, "Overview of the 3rd author profiling task at pan 2015," in *CLEF2015 Working Notes. Working Notes of CLEF 2015-Conference and Labs of the Evaluation forum.* Notebook Papers, 2015.
- [16] F. Rangel, P. Rosso, B. Verhoeven, W. Daelemans, M. Potthast, and B. Stein, "Overview of the 4th author profiling task at pan 2016: cross-genre evaluations," in *Working Notes Papers of the CLEF 2016 Evaluation Labs. CEUR Workshop Proceedings/Balog, Krisztian [edit.]; et al.*, 2016, pp. 750--784.
- [17] P. S. Foundation, "Python," <https://www.python.org/>, 1991.
- [18] S. Ramírez, "Fastapi," <https://fastapi.tiangolo.com/>, 2018.
- [19] M. Inc., "Mongodb," <https://www.mongodb.com/>, 2007.
- [20] P. G. D. Group, "Postgresql," <https://www.postgresql.org/>, 1996.
- [21] O. Corporation, "Mysql," <https://www.mysql.com/>, 1995.
- [22] M. Inc., "Pymongo," <https://pymongo.readthedocs.io/en/stable/>, 2007.
- [23] Vercel, "Next.js," <https://nextjs.org/>, 2016.
- [24] Facebook, "React," <https://reactjs.org/>, 2013.
- [25] Microsoft, "Typescript," <https://www.typescriptlang.org/>, 2012.

- [26] S. O. Team, ``Stack overflow developer survey 2023," feb 2023. [En línea]. Disponible en: <https://survey.stackoverflow.co/2023/>
- [27] Vercel, ``Zod," <https://zod.dev/>, 2020.
- [28] S. Team, ``Sass," <https://sass-lang.com/>, 2006.
- [29] B. Team, ``Bootstrap," <https://getbootstrap.com/>, 2011.
- [30] M.-U. Team, ``Material-ui," <https://material-ui.com/>, 2014.
- [31] C. U. Team, ``Chakra ui," <https://chakra-ui.com/>, 2019.
- [32] C. Team, ``Chart.js," <https://www.chartjs.org/>, 2013.
- [33] F. Team, ``Framer motion," <https://www.framer.com/motion/>, 2019.
- [34] S. learn Team, ``Scikit-learn," <https://scikit-learn.org/>, 2007.
- [35] T. Team, ``Tqdm," <https://tqdm.github.io/>, 2015.
- [36] P. S. Foundation, ``Pickle," <https://docs.python.org/3/library/pickle.html>, 1991.
- [37] N. Team, ``Numpy," <https://numpy.org/>, 2006.
- [38] Atlassian, ``Trello," <https://trello.com/>, 2011.
- [39] G. Team, ``Git," <https://git-scm.com/>, 2005.
- [40] GitHub, ``Github," <https://github.com/>, 2008.
- [41] GitLab, ``Gitlab," <https://gitlab.com/>, 2011.
- [42] Bitbucket, ``Bitbucket," <https://bitbucket.org/>, 2008.
- [43] Microsoft, ``Visual studio code," <https://code.visualstudio.com/>, 2015.
- [44] P. Team, ``Prettier," <https://prettier.io/>, 2017.
- [45] B. Team, ``Black," <https://black.readthedocs.io/>, 2018.
- [46] E. Team, ``Eslint," <https://eslint.org/>, 2013.
- [47] E. Amodio, ``Gitlens," <https://gitlens.amod.io/>, 2017.
- [48] Jgraph, ``Draw.io," <https://www.diagrams.net/>, 2012.
- [49] C. Feuersänger, ``Pgfplots," <https://ctan.org/pkg/pgfplots>, 2007.

- [50] Docker, ``Docker," <https://www.docker.com/>, 2013.
- [51] J. Sutherland and K. Schwaber, ``Scrum," <https://www.scrum.org/>, 1995.
- [52] E. Evans, ``Domain-driven design," <https://www.domainlanguage.com/ddd/>, 2003.
- [53] Adobe, ``Adobe xd," <https://www.adobe.com/products/xd.html>, 2016.
- [54] Figma, ``Figma," <https://www.figma.com/>, 2012.
- [55] P. S. Foundation, ``uuid," <https://docs.python.org/3/library/uuid.html>, 1991.
- [56] P. J. Leach, M. Mealling, and R. Salz, ``Rfc 4122 - a universally unique identifier (uuid) urn namespace," <https://tools.ietf.org/html/rfc4122>, 2005.
- [57] Indeed, ``Lenguajes de programación mejor pagados en españa," jul 2023. [En línea]. Disponible en: <https://es.indeed.com/orientacion-laboral/remuneracion-salarios/lenguajes-programacion-mejor-pagados>