



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, septiembre de 2023.

Agradecimientos

En primer lugar, me gustaría agradecerles a mis tutores, Patricia y David, por su ayuda durante todo el desarrollo del trabajo y por sentar las bases de lo que es este proyecto.

Agradecerle a mi familia por su incansable paciencia, su constante apoyo y sus incontables palabras de ánimo durante todo este tiempo, sin las cuales todo habría sido mucho más difícil.

Quería agradecerle también a mi pareja todo lo que me da día a día, todo el apoyo que hace que pueda seguir adelante ante cualquier problema y todo lo que significa para mí una persona tan maravillosa en todos los sentidos y sin la que no habría llegado a ser quien soy.

Por último, agradecerles a todos mis amigos, tanto a los que tuve la oportunidad de conocer durante la carrera como a los de toda la vida, haciendo mención especial a mi gran amigo y compañero de piso, por todo lo que me ha aportado desde que lo conocí.

Resumen

Con el objetivo de proporcionar una herramienta útil basada en el perfilado automático de autores, a lo largo de este trabajo se profundizará más en este campo en el contexto de las redes sociales, analizando cuáles son los mejores algoritmos, las técnicas más utilizadas y las características más comunes que se suelen emplear. Asimismo, se desarrollará, bajo la metodología SCRUM, una aplicación web donde se muestren los resultado del perfilado en un *dashboard* intuitivo y accesible. Finalmente, para mostrar un caso de uso real de la aplicación se hará uso de la colección de referencia sobre el movimiento *Black Lives Matter* (#BLM), puesta a disposición para este TFG por los tutores del mismo, con el objetivo de analizar el perfil de los usuarios que participaron en los debates sobre este movimiento en las redes sociales.

Abstract

To provide a useful tool based on automatic author profiling, throughout this work, we will delve deeper into this field in the context of social networks, analyzing the best algorithms, the most commonly used techniques, and the typical features that are often employed. Furthermore, under the SCRUM methodology, we will develop a web application that displays the profiling results on an intuitive and accessible dashboard. Finally, to illustrate a real-use case of the application, the reference collection related to the "Black Lives Matter" movement (#BLM) will be utilized. This collection has been made available for this Bachelor's Thesis by its tutors, to analyze the profiles of users who participated in discussions about this movement on social media.

Palabras clave:

- *Perfilado de autor*
- *Redes sociales*
- *Procesado de lenguaje natural*
- *Aplicación web*
- *Aprendizaje automático*

Keywords:

- *Author profiling*
- *Social networks*
- *Natural language processing*
- *Web application*
- *Machine learning*

Índice general

1	Introducción	1
1.1	Importancia de las redes sociales	2
1.2	Perfilado de autor	2
1.3	Objetivos del trabajo	3
1.4	Estructura de la memoria	3
2	Estado del arte del perfilado de autor	5
2.1	Conceptos básicos	6
2.1.1	TF-IDF	6
2.1.2	Etiquetas POS	7
2.1.3	N-gramas	8
2.1.4	<i>Bag-of-words</i>	8
2.1.5	Análisis Semántico Latente	8
2.2	Tareas analizadas	8
2.2.1	<i>PAN Author Profiling 2014</i>	9
2.2.2	<i>PAN Author Profiling 2015</i>	11
2.2.3	<i>PAN Celebrity Profiling 2019</i>	14
2.2.4	Conclusiones	18
2.3	Algoritmos seleccionados	19
2.3.1	Algoritmo de Grivas et al. [1]	19
2.3.2	Algoritmo de Martinc et al. [2]	21
2.3.3	Pruebas	23
3	Herramientas, técnicas y lenguajes	26
3.1	<i>Backend</i>	26
3.2	<i>Frontend</i>	27
3.3	Algoritmos de perfilado	28
3.4	Soporte	28

4 Metodología	30
4.1 Roles	30
4.2 Adaptaciones metodológicas	31
5 Análisis	32
5.1 Requisitos funcionales	32
5.2 Requisitos no funcionales	34
6 Diseño	35
6.1 Arquitectura	35
6.2 Prototipado	37
6.3 <i>Backend</i>	42
6.3.1 <i>Endpoints</i>	42
6.3.2 Clases	45
6.4 <i>Frontend</i>	47
7 Implementación	50
7.1 Adaptación de los algoritmos	50
7.2 Estructura del proyecto	51
7.3 Publicación del código fuente	53
8 Caso de uso: #BLM	55
8.1 Puesta en marcha del sistema	55
8.2 <i>Dataset</i> del movimiento #BLM	55
8.3 Subida del <i>dataset</i>	56
8.4 Selección del algoritmo	57
8.5 Proceso de perfilado	58
8.6 Visualización de resultados	59
8.7 Análisis de resultados	60
8.7.1 Distribución de género	60
8.7.2 Distribución de edad	61
8.7.3 Otras características	64
9 Planificación y costes	66
9.1 Planificación temporal	66
9.2 Recursos y costes	67

10 Conclusiones	69
10.1 Trabajo futuro	69
10.2 Lecciones aprendidas	70
Bibliografía	73

Índice de figuras

1.1	Evolución del número de usuarios en redes sociales	1
2.1	Comparativa del número de autores por idioma en el <i>corpus</i> de la competición <i>PAN Author Profiling 2014</i>	10
2.2	Comparativa del número de autores por idioma en el <i>corpus</i> de la competición <i>PAN Author Profiling 2015</i>	12
2.3	Distribuciones de género y fama en el <i>corpus</i> de la competición <i>PAN Celebrity Profiling 2019</i>	16
2.4	Distribución de ocupaciones en el <i>corpus</i> de la competición <i>PAN Celebrity Profiling 2019</i>	16
2.5	Distribución del año de nacimiento en el <i>corpus</i> de la competición <i>PAN Celebrity Profiling 2019</i>	16
2.6	Características extraídas en el algoritmo de Grivas et al. [1],	20
2.7	Niveles del preprocesado en el algoritmo de Martinc et al. [2]	22
5.1	Diagrama de casos de uso	33
6.1	Diagrama de capas del patrón de diseño DDD. Adaptado de Ł, Ryś [3]	36
6.2	Diagrama C4 de Contenedor de la arquitectura del sistema	36
6.3	Prototipo de la pantalla de inicio	37
6.4	Prototipo de la selección del algoritmo de perfilado	38
6.5	Prototipo del resumen del perfilado	39
6.6	Prototipo de la pantalla de ejemplos de dataset	40
6.7	Prototipo del dashboard utilizando el algoritmo de Martinc et al. [2]	41
6.8	Prototipo del dashboard utilizando el algoritmo de Grivas et al. [1]	42
6.9	Estructura del JSON devuelto por el endpoint /profilings/{id}	44
6.10	Diagrama de endpoints del backend	45
6.11	Diagrama de clases del backend	47

6.12	Diagrama de clases del <i>frontend</i>	49
7.1	Estructura de directorios del proyecto	53
8.1	Página de inicio de la aplicación	56
8.2	Página de ejemplos de <i>datasets</i>	57
8.3	Página de selección algoritmos	58
8.4	Tooltip de información sobre los algoritmos	58
8.5	Página de resumen del perfilado	59
8.6	<i>Dashboard</i> con los resultados obtenidos por el algoritmo de Martinc [2]	59
8.7	<i>Dashboard</i> con los resultados obtenidos por el algoritmo de Grivas [1]	60
8.8	Distribución de género obtenida por ambos algoritmos	61
8.9	Distribución de edad obtenida por el algoritmo de Martinc at al. [2]	63
8.10	Distribución de edad obtenia por el algoritmo de Grivas at al. [1]	64

Índice de tablas

2.1	Distribución del número de autores en cada rango de edad en el <i>corpus</i> de la competición <i>PAN Author Profiling 2014</i>	10
2.2	Cuatro mejores clasificados en la competición <i>PAN Author Profiling 2014</i>	11
2.3	Distribución de los autores por característica y lenguaje en el <i>corpus</i> de la competición <i>PAN Author Profiling 2015</i>	13
2.4	Cuatro mejores clasificados en la competición <i>PAN Author Profiling 2015</i>	14
2.5	Cuatro mejores clasificados en la competición <i>PAN Celebrity Profiling 2019</i>	18
2.6	Relación de características utilizadas en cada subtarea en el algoritmo de Grivas et al. [1]	21
2.7	Resultados de las pruebas realizadas con el algoritmo de Grivas et al. [1]	23
2.8	Resultados de las pruebas realizadas con el <i>dataset</i> de 2014 para el algoritmo de Grivas et al. [1]	24
2.9	Resultados de las pruebas realizadas con el algoritmo de Martinc et al. [2]	25
5.1	Historias de usuario	33
5.2	Requisitos no funcionales	34
8.1	Distribución de género obtenida por ambos algoritmos	61
8.2	Distribución de edad obtenida por el algoritmo de Martinc et al. [2]	63
8.3	Distribución de edad obtenida por el algoritmo de Grivas et al. [1]	64
8.4	Distribución media de los rasgos personales obtenidos por el algoritmo de Grivas et al. [1]	65
8.5	Distribución de ocupación obtenida por el algoritmo de Martinc et al. [2]	65
8.6	Distribución de fama obtenida por algoritmo de Martinc et al. [2]	65
9.1	Especificaciones del portátil empleado para el desarrollo del proyecto	67
9.2	Coste del proyecto	68

Capítulo 1

Introducción

APARTIR 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 informe *Digital 2023 Global Overview Report* (We Are Social et al., 2023) [4], a principios del año 2013 existían alrededor de 1.700 millones de usuarios en las redes sociales, mientras que a principios del año 2023, esta cifra aumentó hasta los 4.700 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 informe 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.

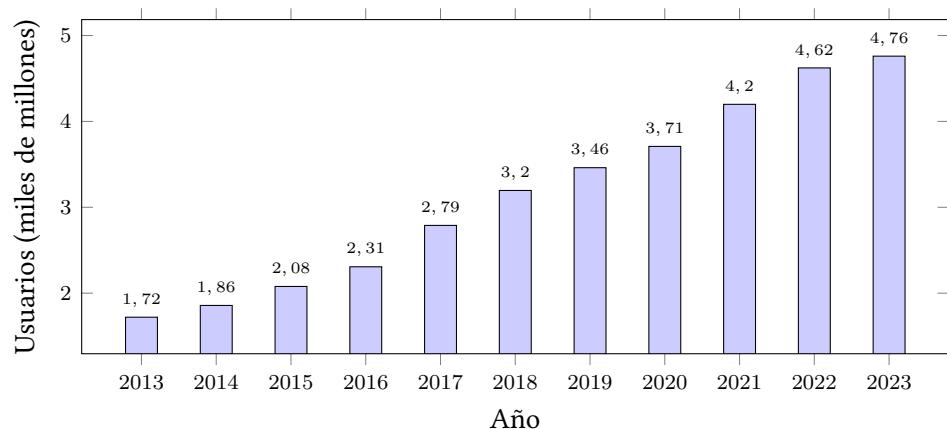


Figura 1.1: Evolución del número de usuarios en redes sociales

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) [5], donde el acceso a estas plataformas 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) [6], pudiendo llegar a influenciar "mucho" o "bastante" la intención de voto (Gallardo-Páuls, 2016) [7]. En segundo lugar, el contenido que se genera en las redes sociales es de gran relevancia a nivel sociológico ya que sirve para analizar, entre muchas cosas, la representación que tienen en ellas la sociedad o la forma en la que interactuamos (Mislove et al., 2011) [8]. Cabe resaltar también la influencia que tienen las plataformas digitales a nivel económico, ya que 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) [9]. Finalmente, destacar el rol fundamental que juega la seguridad en las redes sociales, ya que se han convertido en lugares donde se comparte información personal, y donde se pueden cometer delitos como el *cyberbullying* o el *grooming* (Machimbarrena et al., 2018) [10].

1.2 Perfilado de autor

El perfilado de autor, también conocido como perfilado de usuario o *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 el procesado del lenguaje natural (*Natural Language Processing* o NLP en inglés), que permiten extraer características lingüísticas del texto y utilizarlas para una posterior clasificación.

De esta forma, 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 campos como la seguridad, el *marketing* o la investigación forense (Rangel et al., 2013) [11]. Así, la evolución y el perfeccionamiento del perfilado de autor propiciaría la creación de una herramienta de gran ayuda para los sociólogos a la hora de realizar análisis sobre temas que dispongan de información digital; ofrecería a las empresas una forma de 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 respaldaría el trabajo de los cuerpos de seguridad a la hora de la evaluación de sospechosos en base a su perfil lingüístico.

1.3 Objetivos del trabajo

Tras la introducción al perfilado de autor y las motivaciones que existen para su estudio a nivel de las redes sociales, se expondrán a continuación los objetivos principales de este trabajo.

El primero de ellos es la investigación del estado del arte del perfilado automático de autor, enmarcado en el análisis de documentos digitales. Para acotar el estudio, la investigación se centrará en el perfilado de autor en el ámbito de la lengua inglesa y se priorizarán aquellos algoritmos que perfilen los rasgos generacionales del autor. Asimismo, se profundizará en cuáles son las técnicas de aprendizaje automático más utilizadas en el campo, cuáles son las características que se extraen comúnmente y cuáles son los algoritmos que mejores resultados obtienen.

Como segundo objetivo, una vez se hayan logrado ejecutar satisfactoriamente los algoritmos seleccionados tras el estudio, se pretende desarrollar una aplicación que permita a los usuarios el perfilado automático de una serie de características demográficas de una colección de textos, mostrando un tablero web visual de datos (*dashboard* en inglés) accesible y usable con los resultados obtenidos. Además, se liberará el código de la aplicación en un repositorio público de GitHub [12], para que pueda ser utilizado de forma gratuita por cualquier persona y pueda crecer gracias a la colaboración de la comunidad.

Por último, empleando la aplicación previamente desarrollada, se utilizará como caso de uso la colección sobre el movimiento *Black Lives Matter* (#BLM) proporcionada para este TFG. Se busca, por lo tanto, realizar un análisis con un componente sociológico de los perfiles obtenidos, con el objetivo de obtener conclusiones acerca del movimiento y de los usuarios que participaron debatiendo sobre el mismo en las redes sociales.

Se trata, en definitiva, de un trabajo que pretende profundizar en el perfilado de autor, tanto desde un punto de vista teórico como práctico, haciendo accesible mediante la creación de una aplicación, una herramienta basada en técnicas de aprendizaje automático, y demostrando su utilidad con el análisis de un fenómeno social de gran relevancia.

1.4 Estructura de la memoria

La memoria se estructura en diez capítulos, que se describen a continuación:

- **Introducción:** Se trata del capítulo actual; en él se contextualiza el perfilado automático de autores junto a su utilidad para el análisis de la información generada en las redes

sociales. Asimismo, se exponen los objetivos que tiene este trabajo y la estructura de esta memoria.

- **Estado del arte del perfilado de autor:** En este capítulo se realiza una revisión bibliográfica desde los inicios del perfilado de autor y se explican los conceptos básicos que se utilizan habitualmente en el campo. Posteriormente, se analizan las tareas sobre el perfilado de autor que se han llevado a cabo, seleccionando y evaluando diferentes algoritmos presentados en las mismas.
- **Herramientas, técnicas y lenguajes:** Tercer capítulo de la memoria, donde se exponen las herramientas utilizadas en todo el proceso de desarrollo y se argumenta la elección de cada una de ellas.
- **Metodología:** En este capítulo se explica la metodología que se ha seguido para el desarrollo de la aplicación, así como los roles que desempeña cada uno de los miembros del equipo junto a las adaptaciones que se han realizado a la metodología propuesta.
- **Análisis:** A lo largo de este capítulo se detallan los requisitos funcionales y no funcionales identificados para la aplicación.
- **Diseño:** En este capítulo se muestran los diagramas de arquitectura y de clases diseñados, así como los prototipos de la interfaz web de usuario.
- **Implementación:** Séptimo capítulo de la memoria; en él se explica a bajo nivel las adaptaciones que fueron necesarias para integrar los algoritmos seleccionados en la aplicación. Además, se detalla la estructura de directorios del proyecto y se justifica la decisión de liberar el código fuente.
- **Caso de uso #BLM:** En este capítulo se utiliza como ejemplo de caso de uso de la aplicación el análisis de la colección de referencia ofrecida para este TFG sobre el movimiento *Black Lives Matter*.
- **Planificación y costes:** A lo largo de este capítulo se explican las tareas realizadas en cada *sprint* y se desglosan los costes económicos del proyecto.
- **Conclusiones:** Capítulo final de la memoria, en el que se exponen las líneas de trabajo futuras y se realiza una valoración de los conocimientos adquiridos durante todo el proceso.

Capítulo 2

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 [13] se desmarcaban de la tendencia de la época, la cual se basaba en la clasificación de textos en base a su contenido, para enfocarse 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% (*accuracy=0,773*).

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 dio pie a la realización de trabajos como Argamon et al., 2003 [14], Corney et al., 2002 [15] u Otterbacher et al., 2010 [16]. Además, este hecho también permitió el inicio de una clasificación más compleja en base a otras características demográficas.

Posteriormente, tal como recoge Wiegmann et al., 2019 [17], se continuarán llevando a cabo estudios en esta línea, diversificando las características a clasificar. En este sentido, se realizarían trabajos como Álvarez-Carmona et al., 2018 [18], donde se buscaba predecir el lugar de residencia del autor junto a su ocupación; otros como Volkova et al., 2015 [19], en el que se trataba de perfilar la orientación política, el salario, el optimismo del autor o su sentimiento de satisfacción vital; Preoțiuc-Pietro et al., 2018 [20], en el que se llevaba a cabo la tarea de clasificar la raza y la etnia del autor; u otros como Fatima et al., 2017 [21] que buscaban extender el perfilado de autor a otros idiomas. Asimismo, la mayor parte de los *datasets* utilizados para la creación de los modelos con estos algoritmos, se basaban en textos de redes sociales como Twitter (Burger et al., 2011) [22], Facebook (Fatima et al., 2017) [21] o Reddit (Gjurković et al., 2018) [23], reflejando así la importancia de estas plataformas en la actualidad.

Por otro lado, en el año 2011, se celebraría el primer evento organizado por el *PAN (Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection)* [24], uno de los foros de investigación más importantes que organiza eventos científicos y tareas anuales relacionadas con el análisis forense de textos digitales y rasgos estilométricos, así como uno de los grandes impluses del perfilado de autor. La primera de las tareas centradas en este campo se celebraría en el año 2013 (Rangel et al., 2013) [11], 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% (*accuracy=0,5921*) en la clasificación de género y del 65% (*accuracy=0,6491*) en la clasificación de edad, haciendo uso, la mayor parte de los participantes, de técnicas de aprendizaje supervisado como los Árboles de Decisión (*Decision Trees* en inglés) o las Máquinas de Soporte Vectorial (*Support Vector Machines* o SVMs en inglés). Además, la mayoría de ellos incluyeron en sus modelos características basadas en el TF-IDF, n-gramas o etiquetas POS, explicadas en detalle en la Sección 2.1, junto a otras como el número de emoticonos o la frecuencia de signos de puntuación. En los siguientes años, se celebrarían nuevas ediciones centradas en perfilado (Rangel et al., 2014 [25], Rangel et al., 2015 [26] o Rangel et al., 2016 [27]), en las que se añadirían nuevas subtareas como el reconocimiento de rasgos personales, la ocupación o los dialectos del lenguaje, así como también alcanzando mejores resultados en la clasificación.

2.1 Conceptos básicos

Cuando hablamos del perfilado automático de autores, nos referimos a la tarea del análisis detallado de un texto para la obtención de ciertas características que nos permitan identificar a su autor. Dicha tarea de análisis está enmarcada dentro del campo del procesado del lenguaje natural y, más concretamente, dentro de la rama de la estilometría. De esta forma, para las tareas de NLP se hace uso de una serie de algoritmos y técnicas que permiten la extracción de características formales de un texto que puedan ser procesadas por un ordenador. Las más utilizadas y relevantes para la comprensión de los análisis posteriores y, en general, de este trabajo son las siguientes:

2.1.1 TF-IDF

El TF-IDF (del inglés, *Term Frequency-Inverse Document Frequency*) es una medida estadística muy utilizada en el campo de la recuperación de información (en inglés *Information Retrieval* o IR) que expresa cuán relevante es una palabra para un documento en una colección.

El cálculo de dicha medida para cada palabra sigue la siguiente fórmula:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t) \quad (2.1)$$

Donde:

- t es la palabra de la que se quiere calcular el TF-IDF.
- d es el documento en el que se encuentra la palabra.
- $\text{TF}(t, d)$ es la frecuencia de la palabra t en el documento d , que se calcula como:

$$\text{TF}(t, d) = \frac{n_{t,d}}{n_d} \quad (2.2)$$

Donde:

- $n_{t,d}$ es el número de veces que la palabra t aparece en el documento d .
- n_d es el número total de palabras en el documento d .
- $\text{IDF}(t)$ es la frecuencia inversa de documentos que representa la importancia de la palabra t en la colección de documentos y se calcula como:

$$\text{IDF}(t) = \log \left(\frac{N}{\text{DF}(t)} \right) \quad (2.3)$$

Donde:

- N es el número total de documentos en la colección.
- $\text{DF}(t)$ es el número de documentos en los que aparece la palabra t .

Por lo tanto, la importancia aumenta proporcionalmente al número de veces que una palabra aparece en el documento, pero se compensa con la frecuencia de la palabra en la colección, lo que permite manejar el hecho de que algunas palabras son generalmente más comunes que otras.

2.1.2 Etiquetas POS

Las etiquetas POS (del inglés, *Part-Of-Speech*) son etiquetas gramaticales que se asignan a las palabras de un texto según su función sintáctica, es decir, si se trata de un sustantivo, un verbo, un adjetivo, etc. Existen varios tipos de etiquetas POS, pero las más empleadas son las descritas en el *Penn Treebank Project* [28].

2.1.3 N-gramas

Los n-gramas son una técnica de modelado de lenguaje que consiste en la extracción de secuencias de n elementos de un texto. Estos elementos pueden ser de varios tipos, entre ellos, caracteres, palabras o etiquetas POS. Las longitudes de dichas secuencias de elementos más habituales suelen oscilar entre uno y cuatro, dando como resultado los unigramas, bigramas, trigramas y tetragramas. Asimismo, los n-gramas suelen combinarse con el TF-IDF para proporcionar vectores de características que puedan ser utilizados por los algoritmos de clasificación. Cabe destacar también que el TF-IDF, en su versión más común, es un caso particular de uso de los n-gramas, en concreto, de unigramas de palabras.

2.1.4 Bag-of-words

La *bag-of-words* o BoW es una técnica simple de representación de documentos que consiste en la extracción de las palabras que aparecen en un texto, sin tener en cuenta el orden en el que aparecen ni la estructura gramatical de la que forman parte. De este modo, se obtiene un vector de características en el que cada posición representa una palabra y que tiene como valores el número de veces que aparece dicha palabra en el documento.

2.1.5 Análisis Semántico Latente

El Análisis Semántico Latente (*Latent Semantic Analysis* o LSA en inglés) es una técnica de IR (del inglés, *Information Retrieval*) desarrollada por Deerwester et al., 1990 [29] que busca descubrir relaciones latentes entre las palabras y los documentos que forman una colección. Esta técnica se basa en la idea de que las palabras que aparecen en contextos similares tienden a tener significados similares. Para aplicarla, se construye una matriz en la que se almacena la frecuencia de cada palabra del vocabulario en cada documento y a continuación, se aplica una descomposición en valores singulares (*Singular Value Decomposition* o SVD en inglés) a dicha matriz para reducir su dimensionalidad y encontrar relaciones palabras-documentos.

Esta técnica tiene limitaciones, sobre todo a la hora de manejar sinónimos y polisemias, ya que al basarse únicamente en patrones estadísticos, no tiene en cuenta el significado real de las palabras.

2.2 Tareas analizadas

Una vez explicados los conceptos básicos, procederemos en esta sección al análisis de las tareas más relevantes en el contexto del perfilado automático de autores. En este sentido, cabe destacar que, dado que el objetivo de este trabajo estaba encuadrado en la investigación sobre

algoritmos de perfilado de autores de habla inglesa, se obvian otras competiciones relevantes como pueden ser las organizadas por IberLEF [30], un foro de investigación que promueve el desarrollo de sistemas basados en NLP en español y otras lenguas ibéricas. Asimismo, puesto que el perfilado de la edad era otra característica fundamental que debía incluir este trabajo, se seleccionaron aquellas tareas en las que se incluía dicha característica, en este caso, las celebradas en los años 2014 (Rangel et al.) [25], 2015 (Rangel et al.) [26] y 2019 (Wiegmann et al.) [17] por PAN.

2.2.1 PAN Author Profiling 2014

Esta segunda edición de la competición internacional de perfilado de autores celebrada en 2014 por PAN, estaba enfocada en la clasificación de género y edad de los autores, en la que se consideraban, para esta última, las siguientes clases: 18-24, 25-34, 35-49, 50-64 y 65-XX; donde esta última representa cualquier edad igual o superior a 65 años.

Corpus

En lo referente al *dataset* de entrenamiento y evaluación utilizado, la organización de la competición elaboró un *corpus* con documentos de diferentes tipos que incluían: *tweets*, *posts* de blogs, *posts* de otras redes sociales y *reviews* de hoteles. Todos ellos contenían documentos en inglés y en español a excepción de las *reviews* de hoteles, las cuales solo estaban disponibles en inglés.

Como se puede observar en la distribución de los autores, disponible tanto en la Tabla 2.1 como en la Figura 2.1, el número de autores de habla inglesa es muy superior a los de habla española, lo que provoca potencialmente que los algoritmos entrenados con este *dataset* tengan una mejor generalización y obtengan, por lo tanto, mejores predicciones para la lengua inglesa. Asimismo, existe un amplio desbalanceo de clases con respecto a la edad en el *corpus*, donde las clases 18-24 y 65-XX cuentan con un número de autores significativamente inferior al resto de clases. Destacar finalmente que, en cuanto al género, el *dataset* está perfectamente balanceado, existiendo el mismo número de autores masculinos y femeninos, lo que ayuda a evitar un posible sesgo.

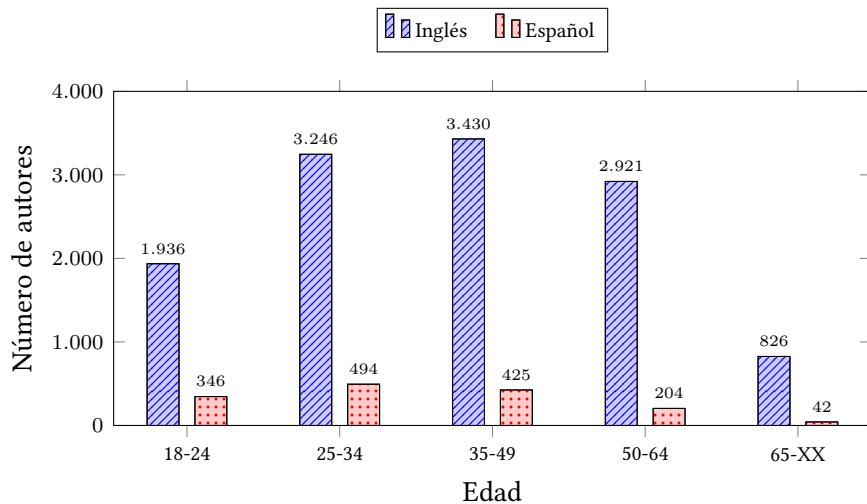


Figura 2.1: Comparativa del número de autores por idioma en el *corpus* de la competición *PAN Author Profiling 2014*

Edad	Entrenamiento		Test	
	Inglés	Español	Inglés	Español
18-24	1.936	346	850	158
25-34	3.246	494	1.380	218
35-49	3.430	425	1.470	210
50-64	2.921	204	1.226	92
65-XX	826	42	324	32
Total	12.359	1.511	5.250	710

Tabla 2.1: Distribución del número de autores en cada rango de edad en el *corpus* de la competición *PAN Author Profiling 2014*

Resultados

Basándonos en los resultados de la competición, reflejados en la Tabla 2.2, podemos observar que los mejores datos de precisión en cuanto a género fueron de alrededor de un 66% para el inglés ($accuracy=0,6630$) y de un 61% para el español ($accuracy=0,6108$), mientras que para la edad fueron de aproximadamente un 40% para el inglés ($accuracy=0,3950$) y de un 50% para el español ($accuracy=0,5010$).

En cuanto a las aproximaciones algorítmicas llevadas a cabo por los equipos mejor clasi-

ficados en la competición, podemos destacar lo siguiente:

- **Preprocesado:** En cuanto al preprocesado, tanto Maharjan et al. [31] como Weren et al. [32] realizaron un limpiado de los documentos (del inglés, *Extensible Markup Language*) para obtener texto plano, escapando caracteres inválidos en el caso del segundo equipo.
- **Características:** En relación a las características (*features* en inglés) extraídas de los documentos para la clasificación, Maharjan [31], Weren et al. [32] y Siang et al., consideraron diferentes rasgos estilométricos como la frecuencia de los signos de puntuación, el tamaño de las frases o la frecuencia de las palabras, entre otros. Con respecto a los rasgos basados en el contenido, Siang et al. y Maharjan et al. [31], modelizaron el lenguaje con *bag-of-word* y n-gramas, respectivamente, mientras que López-Monroy et al. [33] emplearon representaciones más complejas basadas en las relaciones entre palabras y documentos, explicadas a fondo en su trabajo.
- **Clasificación:** En cuanto a los algoritmos de clasificación utilizados, todos los equipos que aparecen en la Tabla 2.2 emplearon regresión logística. López-Monroy et al. [33] hizo uso, más concretamente, de una librería de regresión logística y de SVMs lineales llamada LIBLINEAR (*Large Linear Classification*) [34].

Equipo participante	Precisión							
	Global				Inglés		Español	
	Conjunta	Género	Edad	Género	Edad	Género	Edad	
López-Monroy et al. 2014 [33]	0,2790	0,6310	0,4421	0,6512	0,3950	0,6108	0,4892	
Siang et al.	0,2758	0,6344	0,4348	0,6630	0,3909	0,6057	0,4786	
Maharjan et al., 2014 [31]	0,2624	0,5948	0,4411	0,6132	0,3811	0,5764	0,5010	
Weren et al., 2014 [32]	0,2266	0,5866	0,3863	0,6066	0,3690	0,5666	0,4035	

Tabla 2.2: Cuatro mejores clasificados en la competición *PAN Author Profiling 2014*

2.2.2 PAN Author Profiling 2015

En esta competición, a mayores de la clasificación por género y edad del autor, se añadió la predicción de rasgos de la personalidad. En concreto, se buscaba predecir los cinco rasgos descritos en el modelo *Big Five* (Goldberg, 1990) [35]: extroversión (*extroversion*), estabilidad emocional / neuroticismo (*stability / neuroticism*), apertura a la experiencia (*openness to experience*), amabilidad (*agreeableness*) y responsabilidad (*conscientiousness*). En cuanto a la edad,

y a diferencia de la edición anterior de 2014, se eliminó la categoría de 65 años o más, resultando en cuatro categorías: 18-24, 25-34, 35-49 y 50-XX; donde esta última representa cualquier edad igual o superior a los 50 años.

Corpus

En lo que respecta al *corpus* proporcionado para la competición, los organizadores recopilaron *tweets* de la red social Twitter de 632 usuarios en cuatro idiomas diferentes: inglés, español, italiano y holandés. Dicho *dataset* está etiquetado con el género y la edad (solamente para inglés y español) de los autores, que ellos mismos reportaban en sus perfiles de Twitter. Para determinar los rasgos de la personalidad, los autores completaron el test BFI-10 (Rammsstedt et al., 2007) [36], el cual proporcionaba valores para cada rasgo normalizados entre -0,5 y 0,5. En este sentido, es importante destacar la gran reducción en cuanto al número de autores con los que cuenta el *dataset* en comparación, por ejemplo, con competición celebrada en año anterior, la cual cuenta con cerca de 12.000. Esto puede generar problemas de sobreajuste (*overfitting* en inglés) en los modelos entrenados con esta colección, fenómeno que limita en gran medida la generalización y empeora notablemente los resultados en caso de uso reales.

Así, como se puede ver en la Figura 2.2, el *dataset* vuelve a estar desbalanceado en cuanto a la edad, ya que existen clases como la 50-XX la cual apenas cuenta con un 7% de representación. Sin embargo, a excepción de la clase 18-24, existe un número similar de autores de habla española e inglesa, por lo que cabe esperar una generalización semejante para ambos idiomas. En lo referente a la distribución de género, como se puede ver en la Tabla 2.3, está perfectamente balanceada, contando con un 50% de autores de cada género para todos los idiomas.

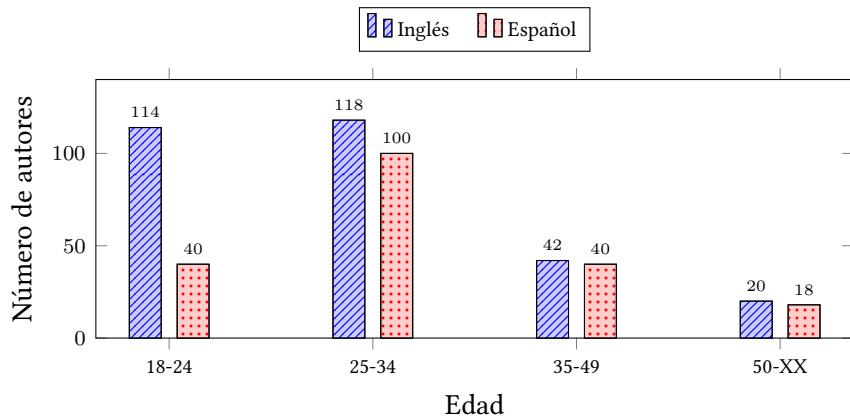


Figura 2.2: Comparativa del número de autores por idioma en el *corpus* de la competición *PAN Author Profiling 2015*

Característica	Entrenamiento				Test			
	Inglés	Español	Italiano	Holandés	Inglés	Español	Italiano	Holandés
18-24	58	22	-	-	56	18	-	-
25-34	60	56	-	-	58	44	-	-
35-49	22	22	-	-	20	18	-	-
50-XX	12	10	-	-	8	8	-	-
Masculino	76	55	19	17	71	44	18	16
Femenino	76	55	19	17	71	44	18	16
Extrovertido	0,16	0,18	0,17	0,24	0,17	0,16	0,15	0,24
Estabilidad	0,14	0,07	0,20	0,21	0,13	0,09	0,20	0,22
Amabilidad	0,12	0,14	0,22	0,13	0,14	0,14	0,19	0,15
Responsabilidad	0,17	0,24	0,18	0,14	0,17	0,21	0,21	0,17
Apertura	0,24	0,18	0,23	0,29	0,26	0,19	0,25	0,28

Tabla 2.3: Distribución de los autores por característica y lenguaje en el *corpus* de la competición *PAN Author Profiling 2015*

Resultados

A la vista de los datos de precisión obtenidos por los primeros equipos clasificados, mostrados en la Tabla 2.4, podemos determinar que existe una gran mejora con respecto a los resultados de la competición anterior del año 2014, mejorando la clasificación conjunta de edad y género en aproximadamente un 40% ($\Delta accuracy=0,4326$).

Más en concreto, cabe destacar los grandes resultados que ofrecen los algoritmos presentados con respecto al género en los dos idiomas principales del *corpus*, llegando a alcanzar precisiones del 96% ($accuracy=0,9659$) para el español en el caso de Alvarez-Carmona et al. [37]. Con respecto a la edad, se puede ver como se obtienen unos mejores resultados, de alrededor de un 4,5% ($\Delta accuracy=0,0444$), en la clasificación en inglés que en español, algo que hipotéticamente puede ser un reflejo de las pequeñas diferencias existentes entre el número de autores en cada uno de los idiomas en el *dataset* de entrenamiento. Sin embargo, estos tan buenos resultados, pueden ser también un reflejo del reducido número de autores con los que cuenta el *dataset*, de apenas 600 autores, lo que puede provocar un sobreajuste (*overfitting* en inglés) de los modelos.

Con respecto a la implementación de los algoritmos, podemos destacar lo siguiente:

- **Preprocesado:** De forma similar a los algoritmos de años anteriores, la mayoría realizaron una limpieza del HTML incluído en los *tweets* y, tanto González-Gallardo et al.

[38] como Grivas et al. [1], utilizaron las menciones, los *hashtags* y los enlaces como características adicionales.

- **Características:** En la mayor parte de los algoritmos presentados, se combinaba el uso de características basadas en el estilo y en el contenido. Así, en cuanto a contenido, González-Gallardo et al. [38] hizo uso de n-gramas de caracteres mientras que Grivas et al. [1] empleó TF-IDF con trigramas. Además, en el caso de Grivas et al. [1], se analizaron otras características basadas en el estilo como la longitud de las palabras o el número de mayúsculas. En el caso de Alvarez-Carmona et al. [37], se hizo uso del Análisis Semántico Latente, explicado en la Sección 2.1.5, junto a una técnica similar a la empleada por López-Monroy et al. [33] en el año 2014.
- **Clasificación:** En lo referente a los métodos utilizados para realizar las predicciones, destaca el uso de la librería LIBLINEAR [34] por parte de Alvarez-Carmona et al. [37] y González-Gallardo et al. [38]. En el caso de Grivas et al. [1], se hizo uso de SVMs para la clasificación del género y la edad mientras que se emplearon SVRs (Drucker et al., 1996) [39] para el reconocimiento de los rasgos personales.

Equipo participante	Precisión									
	Global			Inglés		Español		Italiano	Holandés	
	Conjunta	Género	Edad	Género	Edad	Género	Edad	Género	Género	
Alvarez-Carmona et al. 2015 [37]	0,7116	0,8712	0,8168	0,8592	0,8380	0,9659	0,7955	0,7222	0,9375	
González-Gallardo et al., 2015 [38]	0,6693	0,8871	0,7545	0,8521	0,7817	0,8977	0,7273	0,8611	0,9375	
Grivas et al., 2015 [1]	0,6487	0,9011	0,7199	0,8592	0,7465	0,9432	0,6932	0,8333	0,9688	
Kocher et al., 2015 [40]	0,5655	0,7800	0,7250	0,7113	0,7113	0,8182	0,7386	0,7778	0,8125	

Tabla 2.4: Cuatro mejores clasificados en la competición *PAN Author Profiling 2015*

2.2.3 PAN Celebrity Profiling 2019

En esta última competición analizada, la primera de ellas celebrada específicamente para el perfilado de celebridades, la tarea consistía en predecir cuatro características demográficas a partir de sus publicaciones en Twitter:

- **Género:** Masculino, femenino o no binario.
- **Año de nacimiento:** A la hora de calcular la precisión del algoritmo, se contempló una ventana de error adaptativa, entre los 2 y los 9 años, en función de la edad real de la celebridad.

- **Ocupación o motivo de fama:** Deportista (*sports*), actor (*performer*), político (*politics*), creador de contenido (*creator*), director (*manager*), científico (*science*), profesional (*professional*) o religioso (*religious*).
- **Fama:** Revelación (*rising*), estrella (*star*) o superestrella (*superstar*), determinado en función del número de seguidores en Twitter.

Corpus

Como se mencionó anteriormente, el *corpus* empleado está compuesto por publicaciones en inglés de celebridades en la red social Twitter. En este caso, un usuario era considerado una celebridad si tenía un perfil verificado en Twitter y superaba cierto umbral de notoriedad en Wikidata [41]. El *corpus*, que contenía un total de 48.335 celebridades con aproximadamente 2.180 *tweets* cada una, fue obtenido a partir del *Webis Celebrity Profiling Corpus* (Wiegmann et al., 2019) [42], en el cual se relacionaban los perfiles de Twitter de cada celebridad con sus entradas en Wikidata. De esta forma, se obtuvieron los *tweets* de la red social y se simplificaron las características demográficas que ofrecía Wikidata, excluyendo las celebridades nacidas antes de 1940 y que no tuvieran el inglés como lengua de uso principal. Finalmente, para generar un conjunto de entrenamiento y otro de validación, se optó por una división 70-30, obteniendo 33.836 y 14.499 celebridades para cada uno, respectivamente.

Con todo, las distribuciones de las celebridades en las distintas características demográficas se muestran en las Figuras 2.3, 2.4 y 2.5. Como se puede observar, existe un gran desbalanceo en el género, fama y ocupación, propiciando un posible sesgo hacia las clases mayoritarias (masculino, estrella y deportista) en las predicciones de los algoritmos. En lo que respecta al año de nacimiento, la mayor parte se concentran alrededor de los años 80 y 90, acumulando un 50% de las celebridades. Además, como se puede ver en dicho gráfico, existe una anomalía para el año 2000, la cual se debe a un error de los datos proporcionados por Wikidata, según apunta la organización de la competición.

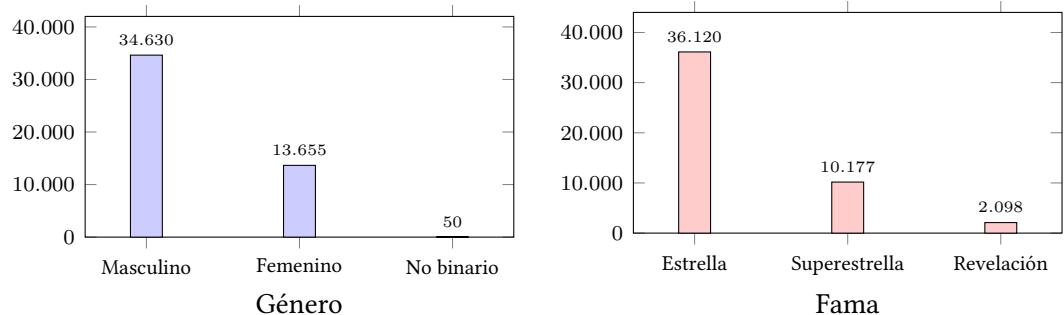


Figura 2.3: Distribuciones de género y fama en el *corpus* de la competición *PAN Celebrity Profiling 2019*

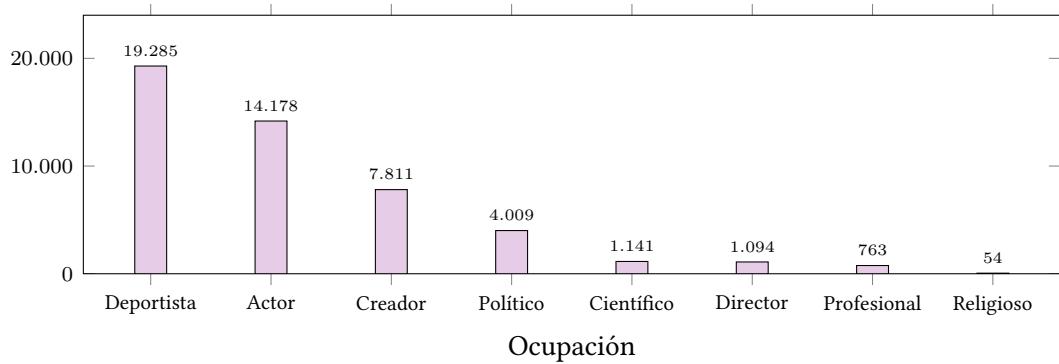


Figura 2.4: Distribución de ocupaciones en el *corpus* de la competición *PAN Celebrity Profiling 2019*

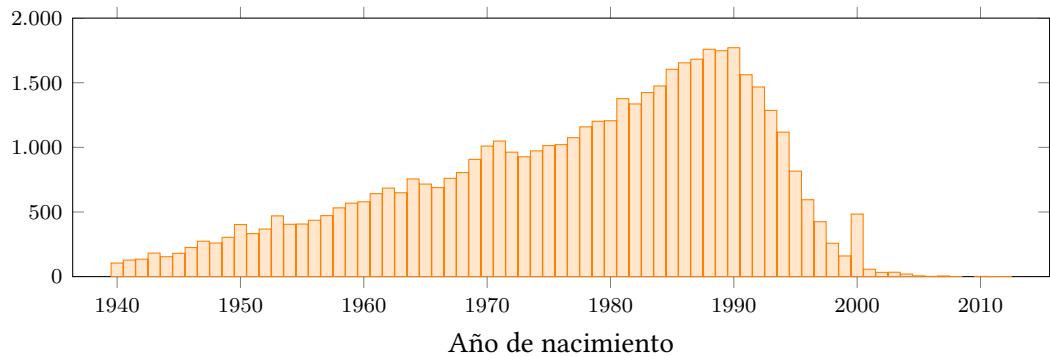


Figura 2.5: Distribución del año de nacimiento en el *corpus* de la competición *PAN Celebrity Profiling 2019*

Resultados

Como se aprecia en la Tabla 2.5, la precisión conjunta en edad y género obtenida por el algoritmo mejor clasificado es de aproximadamente un 20% ($\Delta accuracy=0,2346$) inferior a la alcanzada en la competición del año 2015. Esto es debido a que, como se comentó anteriormente, la clasificación de edad no se realizó en base a rangos en esta edición, sino que se buscó una predicción exacta de la edad del autor, junto a una grado de tolerancia adaptativo, provocando así precisiones mucho más bajas para este atributo. En lo que respecta a la clasificación del género, vemos como los dos mejores participantes obtienen resultados de aproximadamente un 90% ($accuracy=0,928$; $accuracy=0,906$), superior a los obtenidos en la edición de 2015. Por último, en cuanto a la clasificación de ocupación, destacar el gran resultado que ofrece el algoritmo de Radivchev et al. [43], obteniendo una precisión del 75% para una característica con ocho clases diferentes ($accuracy=0,757$).

Centrándonos en las implementaciones realizadas en esta competición, podemos destacar lo siguiente:

- **Preprocesado:** Para el preprocesado, Martinc et al. [2], Radivchev et al. [43] y Moreno-Sandoval et al. [44], optaron por sustituir los enlaces, las menciones y los *hashtags* por etiquetas especiales. Por otro lado, Martinc et al. [2] se decantó por eliminar los signos de puntuación y las *stopwords*, es decir, las palabras poco significativas, para mejorar el modelado de los documentos con TF-IDF puro basado en unigramas de palabras.
- **Características:** La mayor parte de los algoritmos presentados emplearon técnicas basadas en TF-IDF y n-gramas. Más concretamente, Radivchev et al. [43] utilizó como características los vectores obtenidos mediante TF-IDF sobre los 10.000 bigramas más frecuentes; Martinc et al. [2] obtuvo los vectores mediante el TF-IDF de unigramas de palabras y tetragramas de caracteres; Moreno-Sandoval et al. [44], en cambio, además de emplear n-gramas como características, calculó el número medio de emojis, *hashtags*, menciones, enlaces, *retweets*, palabras y longitud de las palabras por *tweet*.
- **Clasificación:** Al igual que en las pasadas ediciones, los métodos de clasificación más frecuentes fueron las SVMs y la regresión logística. Radivchev et al. [43] utilizó SVMs para la predicción de la fama y la ocupación, mientras que para la edad y género hizo uso de la regresión logística; Moreno-Sandoval et al. [44], en cambio, empleó la regresión logística para la clasificación de todos los atributos excepto para la ocupación, que obtuvo mejores resultados haciendo uso de un clasificador multinomial Bayesiano Ingenuo (*Multinomial Naive Bayes* o MNB en inglés). Por último, Martinc et al. [2] empleó regresión logística para la clasificación de las cuatro características demográficas.

Equipo participante	Conjunta (Género y Edad)	Precisión			
		Género	Edad	Ocupación	Fama
Radivchev et al., 2019 [43]	0,477	0,928	0,514	0,757	0,777
Martinc et al., 2019 [2]	0,410	0,906	0,453	0,732	0,755
Fernquist et al., 2019	0,362	0,774	0,468	0,642	0,773
Moreno-Sandoval et al., 2019 [44]	0,320	0,862	0,371	0,703	0,545

Tabla 2.5: Cuatro mejores clasificados en la competición *PAN Celebrity Profiling 2019*

2.2.4 Conclusiones

Tras el análisis de estas tres competiciones, en lo que concierne al objetivo de este trabajo, podemos concluir que la clasificación tanto del de género como de la edad son tareas que ofrecen resultados sorprendentes, llegando a superar precisiones de hasta un 90% en el caso del género y un 80% en el caso de la edad, convirtiéndose así en herramientas de gran fiabilidad para el perfilado automático de autores.

En lo que respecta a los algoritmos, no apreciamos grandes diferencias entre ediciones, ya que el preprocesado sigue involucrando el limpiado del texto, la eliminación de palabras poco significativas o la sustitución de ciertos elementos por *tokens* especiales; las características continúan estando basadas en medidas estadísticas como el TF-IDF y los n-gramas, junto a otras que se apoyan en rasgos estilísticos como la media de palabras o el número de signos de puntuación; y los métodos de clasificación más utilizados siguen estando fundamentados en técnicas de aprendizaje supervisado clásicas como los SVMs y la regresión logística.

En este sentido, autores de las competiciones más recientes como Martinc et al., 2019 [2] o Radivchev et al., 2019 [43], destacan la gran efectividad que tienen estos métodos clásicos con respecto a técnicas más modernas como las basadas en Aprendizaje Profundo (*Deep Learning* en inglés). Martinc et al., 2019 [2], por ejemplo, apunta que, haciendo uso de BERT, una red neuronal basada en Transformers propuesta por Devlin et al., 2018 [45], los resultados obtenidos fueron significativamente peores en comparación con la regresión logística, de alrededor de un 10% en F1. En el caso de Radivchev et al., 2019 [43], se reportaron resultados similares, en este caso haciendo uso de DPCNN (*Deep Pyramid Convolutional Neural Network* en inglés), propuesta por Johnson et al., 2017 [46] para el procesado de textos.

2.3 Algoritmos seleccionados

Con todo, para seleccionar los algoritmos que formarán parte de la aplicación que se expondrá en los siguientes capítulos, se ha optado por buscar aquellos que contasen con una implementación pública y que, además, ofreciesen un buen rendimiento en las tareas. Por ello, dado que los algoritmos de la edición de 2014 no obtenían resultados comparables a las otras dos competiciones, no se ha tenido en cuenta ninguno de ellos. Así, de los cuatro algoritmos mejor clasificados del *PAN Author Profiling 2015* y del *PAN Celebrity Profiling 2019* se ha seleccionado uno de cada competición, concretamente las implementaciones propuestas por Grivas et al. [1] (disponible en <https://github.com/pan-webis-de/grivas15>) y Martinc et al. [2] (disponible en <https://github.com/EMBEDDIA/PAN2019>).

En este punto, cabe destacar que la mayor parte de los trabajos analizados en la Sección 2.2 no ofrecían implementaciones públicas de sus algoritmos, sino que se limitaban a explicar a alto nivel su funcionamiento. Así, de los tres únicos casos encontrados que contaban con el código fuente publicado, solo los dos algoritmos mencionados anteriormente disponían de una implementación funcional y bien documentada. Concretamente, se intentó ejecutar la implementación ofrecida por Pizarro, 2019 [47] presentada en la competición *PAN Bots and Gender Profiling 2019*, lo que resultó en errores continuados e intentos fallidos por lograr su correcto funcionamiento. Este problema ha sido ampliamente reconocido por la comunidad, especialmente en el campo del aprendizaje automático y la inteligencia artificial. Tal es así, que en trabajos como el presentado por Hutson, 2018 [48], se critica la falta de reproducibilidad de los algoritmos y se apuntan varios problemas, entre ellos: la inexistencia de código fuente, la ausencia de los *datasets* utilizados para el entrenamiento del modelo y las diferencias conforme al rendimiento esperado de los mismos.

2.3.1 Algoritmo de Grivas et al. [1]

Este algoritmo sigue una aproximación particular basada en la creación de dos grupos de características, unas basadas en el contenido (estructurales) y otras basadas en el estilo (estilométricas). Como se puede ver en la Figura 2.6, las características estructurales se corresponden con el número de *hashtags*, enlaces y menciones que aparecen en los *tweets*, mientras que las características estilométricas se corresponden con otras como el TF-IDF de n-gramas o el número de mayúsculas. A estas últimas se le suman algunas características menos utilizadas como la bolsa de emoticonos (*bag-of-smileys* en inglés), una técnica similar a la *bag-of-words* explicada en la Sección 2.1.4 o los grafos de n-gramas, propuestos por Giannakopoulos et al., 2008 [49].

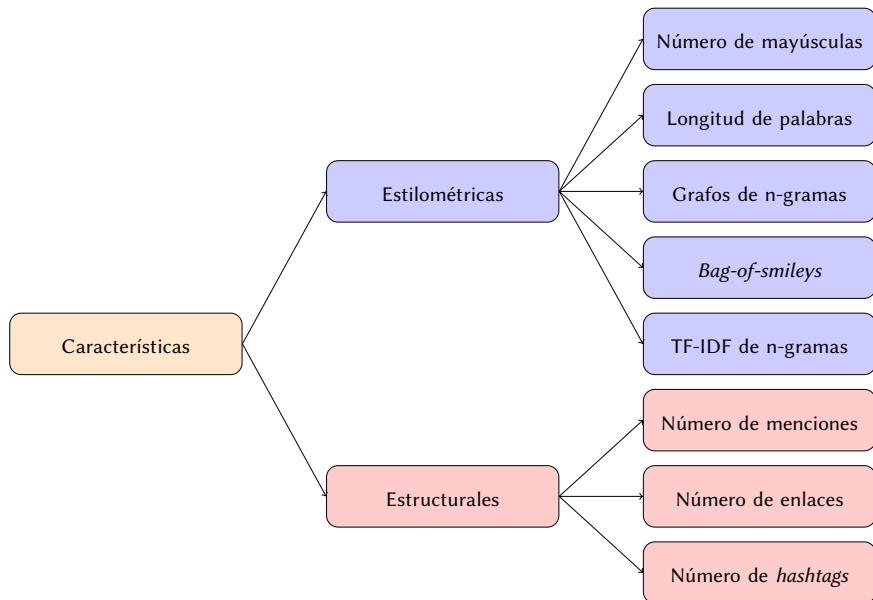


Figura 2.6: Características extraídas en el algoritmo de Grivas et al. [1],

A mayores, para cada uno de los grupos de *features*, se aplicaron preprocesados diferentes debido a la naturaleza de las propias características. De esta forma, para la extracción del grupo de *features* estructurales, no se realizó ningún tipo de preprocesado. Sin embargo, para reducir el ruido en el grupo de *features* estilométricas, se eliminó cualquier tipo de HTML presente en los *tweets*, ignorando así elementos como las menciones o los enlaces.

Por otro lado, como se muestra en la Tabla 2.6, solo algunas de las características extraídas anteriormente son utilizadas en cada una de las subtareas, basándose para ello en una serie de heurísticas y de pruebas realizadas con el *dataset* de entrenamiento.

Grupo de características		
Subtarea	Estructurales	Estilométricas
Género	-	TF-IDF de trigramas
Edad	Número de menciones Número de <i>hashtags</i> Número de enlaces	TF-IDF de trigramas Longitud de palabras
Rasgos personales	Número de menciones Número de <i>hashtags</i> Número de enlaces	TF-IDF de trigramas Longitud de palabras Número de mayúsculas

Tabla 2.6: Relación de características utilizadas en cada subtarea en el algoritmo de Grivas et al. [1]

Finalmente, en lo que concierne a las técnicas de clasificación, se utilizaron SVMs con un kernel RBF (*Radial Basis Function* en inglés) y con un kernel lineal para las subtareas de predicción de la edad y el género, respectivamente. En el caso de la predicción de los rasgos personales, se empleó un SVR (*Support Vector Regression* en inglés) con un kernel lineal. Destacar que, para la ejecución de ambos algoritmos de aprendizaje automático, se hizo uso de las librerías ofrecidas por scikit-learn [50].

2.3.2 Algoritmo de Martinc et al. [2]

En lo que respecta a este segundo algoritmo seleccionado, lo primero que se realiza es un concatenado de los primeros 100 *tweets* de cada usuario para la creación de cada documento que formará la colección. La razón de este límite se justifica en la reducción considerable de la complejidad temporal y espacial, manteniendo a su vez una muestra representativa que garantizaba una buena clasificación.

Una vez realizado este paso, se continúa con el preprocessado de los documentos, el cual se divide en tres niveles incrementales, como se aprecia en la Figura 2.7.

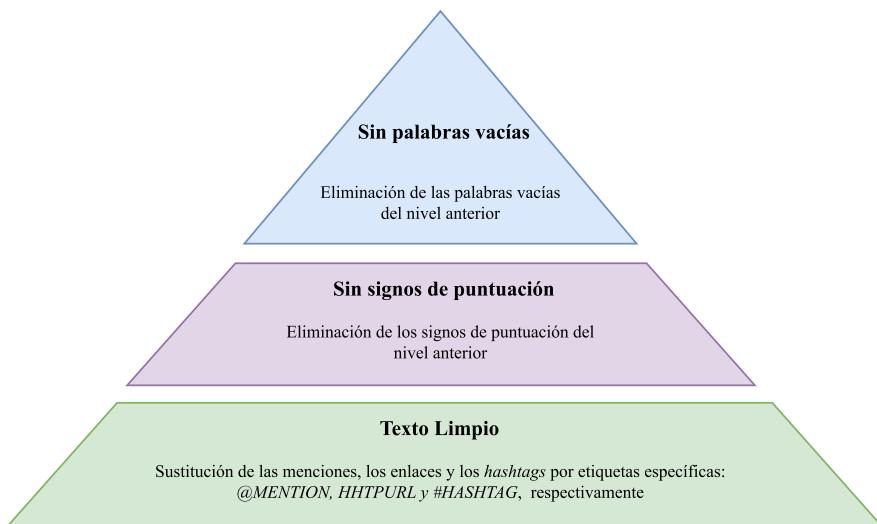


Figura 2.7: Niveles del preprocesado en el algoritmo de Martinc et al. [2]

Posteriormente, en la fase de extracción de características, se obtuvieron diferentes vectores de n-gramas basados en TF-IDF, calculados cada uno sobre un nivel diferente de preprocesado:

- **Unigramas de palabras:** Calculado sobre el documento del Nivel 3 (sin palabras vacías) transformado a minúsculas.
- **Tetragramas de caracteres:** Calculado sobre el documento del Nivel 1 (texto limpio) transformado a minúsculas.
- **Tetragramas de caracteres sufijos:** Calculado sobre el documento del Nivel 1 (texto limpio) transformado a minúsculas. Aclarar que para esta representación se consideran las cuatro últimas letras de cada palabra con una longitud suficiente, es decir, sufijos de cuatro caracteres.

Finalmente, se hicieron pruebas con varios clasificadores, entre ellos SVMs con diferentes *kernels*, *random forest* o *gradient boosting*, pero el que mejores resultados obtuvo fue el de regresión logística, siendo empleado en todas las subtareas de clasificación de la competición. Para hallar dicha solución y obtener los hiparámetros óptimos para cada clasificador, se aplicó una búsqueda en rejilla (*grid search* en inglés), en la que se evaluaron todas las posibles combinaciones de parámetros. Destacar también el uso de la clase *FeatureUnion* de scikit-learn [50], la cual permite la concatenación de diferentes características en un mismo vector asignándoles un diferente peso a cada una de ellas, obteniendo para este caso un peso de 0,8 sobre 1 para los unigramas de palabras y los tetragramas de caracteres, mientras que tan solo un 0,4 para los tetragramas de caracteres sufijos, lo que implica una menor relevancia.

2.3.3 Pruebas

Una vez seleccionados los algoritmos y explicado su funcionamiento, es necesario llevar a cabo una serie de pruebas que nos permitan determinar si obtienen los mismos resultados en la clasificación que los reportados en sus respectivos trabajos. Para ello, se hará uso de los *datasets* de entrenamiento y test originales que proporcionaba públicamente cada competición disponibles en <https://zenodo.org/record/3745945> y <https://zenodo.org/record/3885373> para las ediciones de 2015 y 2019, respectivamente.

En cuanto al algoritmo de Grivas et al. [1], como se observa en la Tabla 2.7, existen pequeñas variaciones en cuanto a la precisión, cerca de un 3,5% ($\Delta accuracy=0,0352$), lo que se puede deber a cambios en el propio *dataset* de entrenamiento o test ofrecidos. A mayores, se muestra una columna con el Macro F1 obtenido. Esta métrica, basada en el F1 tradicional, se calcula como la media de los F1 de cada clase, reflejando los problemas subyacentes en casos de conjuntos de entrenamiento desbalanceados. Así, vemos como los resultados obtenidos para el género son buenos, mientras que para la edad son bastante peores. Esto puede deberse al gran desbalanceo de clases de la edad, junto a las pequeñas diferencias que pueden existir entre rangos contiguos y que provocan errores en la clasificación.

Característica	Precisión		Macro F1
	Reportada	Obtenida	Obtenida
Género	0,8592	0,8310	0,8304
Edad	0,7465	0,7887	0,5631

Tabla 2.7: Resultados de las pruebas realizadas con el algoritmo de Grivas et al. [1]

A mayores, ya que los *datasets* de la edición de 2014 y de 2015 tienen una estructura similar, esto permitió utilizar la colección del *PAN Author Profiling 2014*, disponible en <https://zenodo.org/record/3716008>, para llevar a cabo experimentos y comparar los resultados que ofrece el algoritmo de Grivas et al. [1] con cada uno. Con todo, fue necesario realizar las siguientes modificaciones para que tanto el algoritmo como la colección, fuesen completamente compatibles:

- **Transformación de etiquetas:** Para que el conjunto de clases presentes en cada subtabla fuese el mismo que el presentado en la edición de 2015, se modificaron las etiquetas del fichero de *truth*. De este modo, en el caso de la edad, se agruparon los rangos 50-64 y 65-XX en uno solo (50-XX), mientras que en el género se sustituyeron las etiquetas *MALE* y *FEMALE* por sus iniciales.

- **Modificación de código:** En cuanto al código fuente, fue necesario ignorar todo lo referente a la subtarea de la predicción de los rasgos de la personalidad expuesta en la competición de 2015, ya que el *dataset* de 2014 no contaba con dichas características.

De esta forma, haciendo uso del *dataset* de 2014 para entrenar el algoritmo de Grivas et al. [1], se obtuvieron los resultados mostrados en la Tabla 2.8. Destacar que, debido a que la colección proporcionada en 2014 no contaba con un conjunto específico de test, se realizó la validación con el conjunto de test de la competición de 2015, obteniendo así resultados directamente comparables con los reportados en la Tabla 2.7. En este sentido, vemos como la precisión y el Macro F1 de ambas características se reduce drásticamente a causa, hipotéticamente, de dos factores. El primero de ellos tiene que ver con heterogeneidad de los textos en el conjunto de entrenamiento, ya que se unificaron las publicaciones de los blogs, las *reviews* de hoteles y los *posts* de otras redes sociales, provocando, por un lado, una dificultad añadida en el proceso de extracción de características y, por otro lado, un gran contraste con los *tweets* utilizados para la validación. El segundo factor tiene que ver con la gran diferencia en el tamaño de los *datasets*, dado que la edición de 2014 contaba con 12,053 autores frente a 142 de la de 2015.

Colección del PAN 2014		
Característica	Precisión	Macro F1
Género	0,6408	0,6328
Edad	0,5282	0,3587

Tabla 2.8: Resultados de las pruebas realizadas con el *dataset* de 2014 para el algoritmo de Grivas et al. [1]

En la misma línea, el algoritmo de Martinc et al. [2] obtiene también resultados muy similares a los reportados en su trabajo, como se muestra en la Tabla 2.9, con una variación media que no llega al 1% en la precisión ($\Delta accuracy=0,0085$) y de cerca del 3%, algo más elevadas, en el Macro F1 ($\Delta Macro F1=0,0305$). Destacar que, para esta última métrica, al algoritmo obtiene resultados bastante pobres en comparación con la precisión, lo que demuestra un posible sesgo hacia las clases mayoritarias en el conjunto de entrenamiento.

Característica	Precisión		Macro F1	
	Reportada	Obtenida	Reportada	Obtenida
Género	0,906	0,904	0,587	0,584
Edad	0,453	0,433	0,354	0,308
Fama	0,755	0,755	0,512	0,480
Ocupación	0,743	0,731	0,468	0,427

Tabla 2.9: Resultados de las pruebas realizadas con el algoritmo de Martinc et al. [2]

Sin embargo, la predicción exacta del año de nacimiento, tal y como se exponía en la competición de 2019, supone una utilidad real limitada para el perfilado de autores, por lo que se buscó mejorar la precisión del algoritmo mediante el *binning*. Esta técnica consiste en agrupar los valores de una característica en rangos, de forma que se reduzca la complejidad del problema y se obtengan así mejores resultados. En este caso, como criterio para realizar la agrupación se siguió el mismo que el utilizado en la competición del año 2015, añadiendo el rango de los menores de edad, es decir, se consideraron los siguientes grupos: XX-17, 18-24, 25-34, 35-49, 50-XX; donde, de igual forma que anteriormente, XX-17 representa cualquier edad igual o inferior a 17 años y 50-XX cualquier edad igual o superior a 50 años. Con todo, aplicando el *bining* se consiguió aumentar la precisión para el perfilado de la edad un 16,48%, hasta llegar al 59,68%, lo que, aún siendo inferior al obtenido por Grivas et al. [1], se considera un gran resultado teniendo en cuenta la gran diferencia en tamaño de los datasets empleados en cada competición.

Capítulo 3

Herramientas, técnicas y lenguajes

Una vez abordada la tarea de investigación del estado del arte del perfilado de autor y de la selección de los algoritmos que formarán parte fundamental de la aplicación, nos centraremos en el desarrollo de la misma. Así, a largo de este capítulo se describirán las herramientas utilizadas en todo el proceso y se expondrán las razones de su uso. A su vez, para una mejor estructuración, se dividirán en cuatro grupos diferentes: herramientas del *backend*, *frontend*, algoritmos de perfilado y soporte.

3.1 *Backend*

Ya que para el *backend* no se necesitaba nada excesivamente complejo, se decidió utilizar el lenguaje de programación Python [51] junto con el FastAPI [52], un *framework* el cual permite crear APIs (del inglés, *Application Programming Interface*) REST (del inglés, *Representational State Transfer*) 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, lo que evitó 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 simple.

En cuanto a la persistencia de los datos, se optó por MongoDB [53], 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 [54] o MySQL [55], se debe a dos factores principales. Uno de ellos es la fácil implementación de MongoDB en Python haciendo uso de la librería PyMongo [56], 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, y es que, fren-

te a la creación de nuevos campos o relaciones, MongoDB permitiría una integración simple 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 [57] 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 [58], es decir, en la construcción de interfaces dinámicas e interactivas mediante la composición de elementos que pueden tener estado. Además, esta herramienta 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 [59], un lenguaje de programación que añade tipado estático a JavaScript y que está ganando mucha popularidad con respecto a la mentibilidad, compresión y escalabilidad que proporciona a los proyectos en los que se usa (Stack Overflow, 2023) [60]. 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 [61], una librería que permite definir esquemas de validación de datos, incluyendo el tipo específico que debe tener cada campo.

En cuanto el estilado de la página, se optó por emplear SASS (del inglés, *Syntactically Awesome Style Sheets*) [62], 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 innovadora, se buscó crear un estilo propio haciendo uso de CSS "puro", desmarcándose así de librerías que proporcionan estilos predefinidos como Bootstrap [63] o componentes ya implementados como Material UI [64] o Chakra UI [65]. Por otra parte, para la creación de gráficos, se utilizó la librería ChartJS [66], 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 [67], 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 empleadas en las implementaciones de los algoritmos seleccionados en la Sección 2.3.

En cuanto a la parte nuclear del aprendizaje automático, se utilizó Scikit-Learn [50], 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 [68], la cual permite mostrar el progreso de entrenamiento o predicción de forma visual, Pickle [69], que permite serializar objetos Python (en nuestro caso los modelos ya entrenados), o NumPy [70], 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 [71], una herramienta que permite crear tableros con listas de tareas, las cuales pueden moverse entre ellas según su estado: pendiente, en progreso o completada. En la misma línea, para efectuar las reuniones necesarias para la planificación de las tareas, se utilizó Microsoft Teams [72], una plataforma de comunicación que permite realizar videollamadas, compartir la pantalla o enviar archivos, entre otras funcionalidades.

Además, debido a la importancia de mantener un historial sobre los cambios realizados en el código, se optó por utilizar Git [73] como herramienta de control de versiones junto a GitHub [12], una plataforma que permite almacenar repositorios Git de forma remota, similar a otras como GitLab [74] o BitBucket [75].

En cuanto al entorno de desarrollo o IDE (del inglés, *Integrated Development Environment*), se utilizó Visual Studio Code [76], 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 [77] o Black [78], las cuales permiten formatear el código de forma automática para JavaScript o Python, respectivamente; ESLint [79], una herramienta que permite

detectar errores en el código JavaScript/TypeScript; o GitLens [80], 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 [81], la cual permite crear todo tipo de diagramas *online* de forma gratuita, sin la necesidad de registrarse ni de instalar ningún programa. Junto a esta herramienta, se utilizó también la librería pgfplots [82] para la creación de las gráficas en el propio L^AT_EX.

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 [83], una herramienta que permite crear contenedores, esto es, 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 [84]. 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, a mayores de tener experiencia anterior en su uso, se realizaron ligeras modificaciones que facilitaron en gran medida el desarrollo del proyecto, comentadas en la Sección 4.2.

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. Esto tiene una implicación directa en el transcurso del proyecto, ya que, al solo contar con un desarrollador, cualquier imposibilidad de este para realizar su trabajo conlleva inevitablemente a una parada en todo el desarrollo, aumentando así de forma considerable el riesgo del proyecto.

4.2 Adaptaciones metodológicas

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 realizaron el mismo día que se iniciaba cada *sprint*, esto es, aproximadamente cada tres semanas, lo que se ajusta a la duración recomendada por la metodología Scrum. Estas reuniones fueron, por lo tanto, de una mayor extensión que las *dailies* clásicas y se emplearon también para revisar el incremento desarrollado en el *sprint* anterior. Asimismo, para facilitar el hecho de que todos los miembros del equipo pudiesen asistir a dichas reuniones, se realizaban de forma telemática haciendo uso de Microsoft Teams, como se comentó en la Sección 3.4.

En lo que respecta a los *sprints*, como se detallará en la Sección 9.1, comentar que los dos primeros no atienden específicamente al desarrollo de ninguna historia de usuario, sino que se emplearon para la configuración del entorno de trabajo y para la investigación y experimentación con los algoritmos de perfilado. Por lo tanto, es a partir del tercer *sprint* cuando ya se sigue de forma habitual el desarrollo clásico bajo la metodología Scrum.

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.

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
H1	Como usuario quiero poder subir mi propio <i>dataset</i> de textos para su perfilado
H2	Como usuario quiero conocer ejemplos del formato de los <i>datasets</i> aceptados
H3	Como usuario quiero poder seleccionar el algoritmo de perfilado que se va a utilizar
H4	Como usuario quiero poder visualizar los datos obtenidos tras el perfilado
H5	Como usuario quiero poder ver una lista detallada de cada persona perfilada
H6	Como usuario quiero poder ordenar la lista de personas por cada uno de los campos perfilados
H7	Como usuario quiero poder saber como funcionan los diferentes algoritmos de perfilado disponibles
H8	Como usuario quiero ver el rendimiento de los algoritmos de perfilado disponibles
H9	Como usuario quiero poder reentrenar los algoritmos con diferentes <i>datasets</i>

Tabla 5.1: Historias de usuario

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

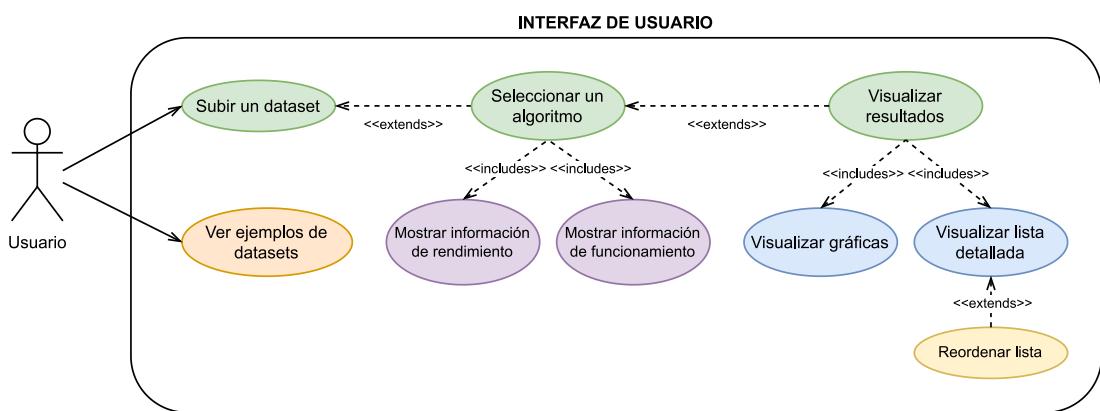


Figura 5.1: Diagrama de casos de uso

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 con cómo debe funcionar la aplicación. En este sentido, a pesar de que la propia metodología Scrum no contempla la definición de requisitos no funcionales más allá de las historias de usuario, se ha considerado adaptar esta parte y definirlos por separado para dotarlos de una mayor importancia. De esta forma, se han determinado los requisitos recogidos en la Tabla 5.2.

ID	Requisito	Descripción
RNF1	Usabilidad	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.
RNF2	Escalabilidad	La aplicación debe ser capaz de procesar <i>datasets</i> de cualquier tamaño.
RNF3	Portabilidad	La aplicación debe ser capaz de ejecutarse en cualquier sistema, haciendo 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 se detallará, a lo largo de este capítulo, la arquitectura software elegida para el sistema, se mostrarán los prototipos de la interfaz de usuario elaborados y se explicará a fondo mediante diagramas de clases la estructura a bajo nivel tanto del *frontend* como del *backend*.

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 en la versión actual (*Servicio de perfilado*), 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*) [85], 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.

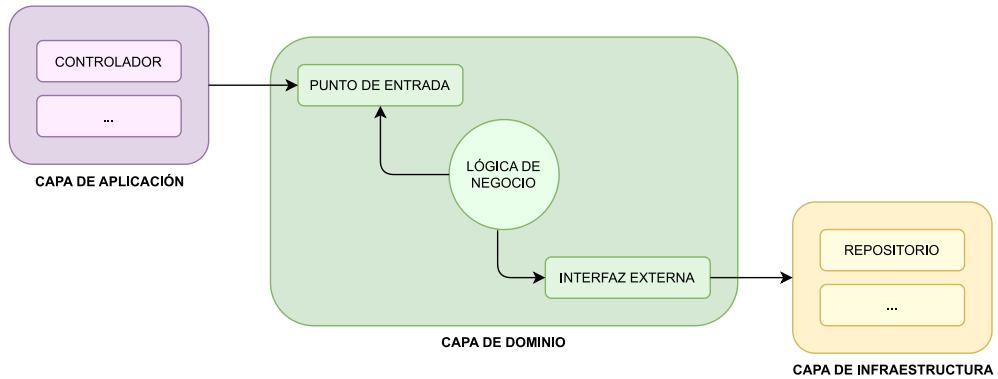


Figura 6.1: Diagrama de capas del patrón de diseño DDD. Adaptado de Ł, Ryś [3]

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 ([RNF3](#)), puesto que, de esta forma, la aplicación podría ser utilizada por cualquier dispositivo con un navegador, evitando desarrollar aplicaciones nativas para cada sistema operativo.

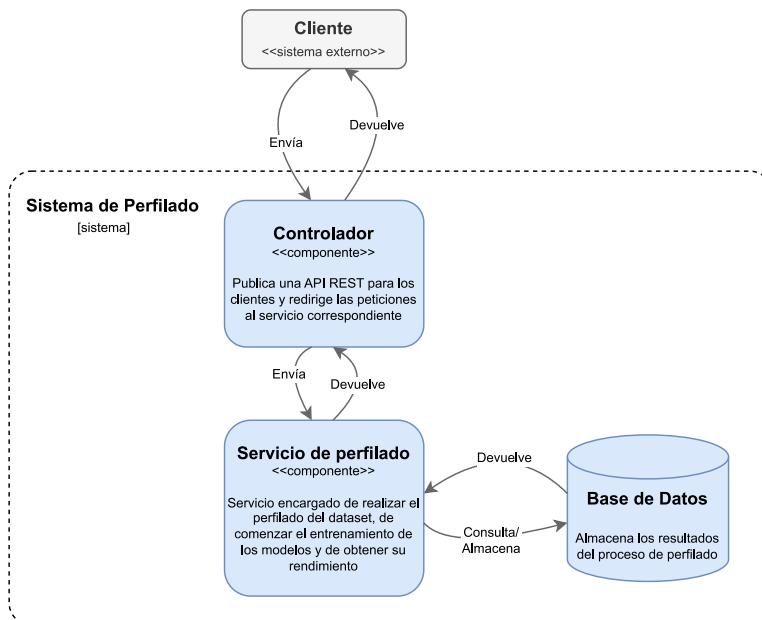


Figura 6.2: Diagrama C4 de Contenedor de la arquitectura del sistema

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 [81] debido a su sencillez y rapidez en la elaboración con respecto a otros programas más avanzados como pueden ser Adobe XD [86] o Figma [87].

La filosofía de diseño de la interfaz se centró en el minimalismo y la accesibilidad, como se establece en el RNF1 recogido en la Sección 5.2, por lo que todo está 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 dará la opción al usuario de subir un *dataset*, tanto mediante un elemento *drag and drop* como mediante un botón que permitirá la selección de un archivo almacenado en el dispositivo.

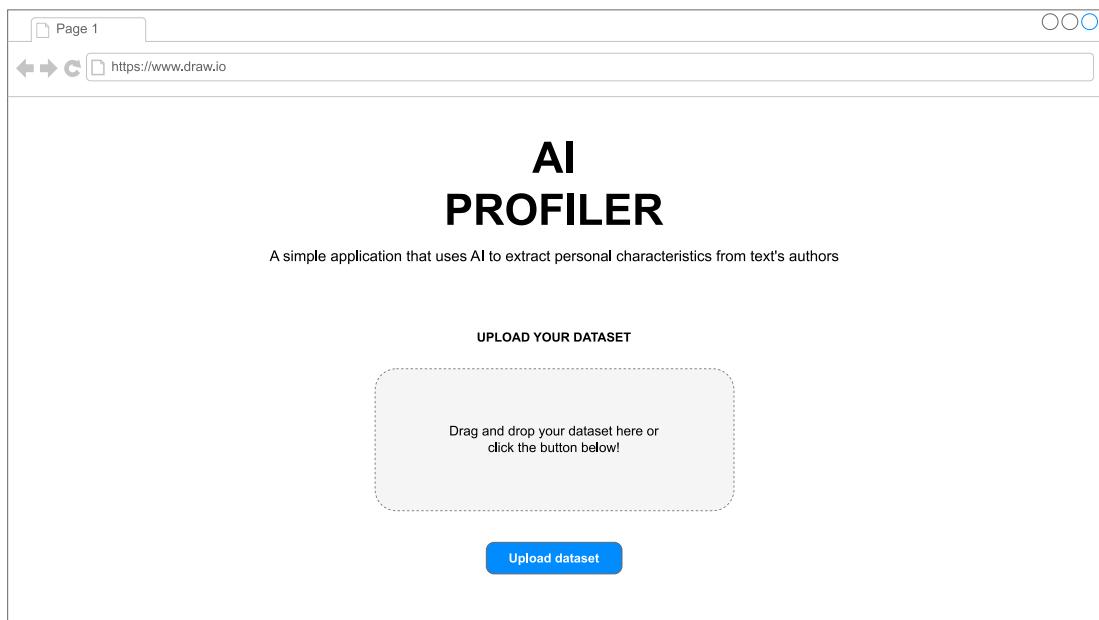


Figura 6.3: Prototipo de la pantalla de inicio

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, su nombre junto a las características demográficas que es capaz de extraer. Además, ya que según indican las historias de usuario H7 y H8, era necesario mostrar información del funcionamiento y del rendimiento de los algoritmos, se decidió crear

un mensaje emergente (*tooltip* en inglés) para ello. Un *tooltip* es una ventana que aparece al pasar el ratón por encima de un elemento y que muestra información adicional sobre el mismo sin ocupar espacio en la interfaz. Finalmente, si el usuario desease cambiar el *dataset* seleccionado, se permitirá retroceder mediante la flecha situada junto al título del paso actual.

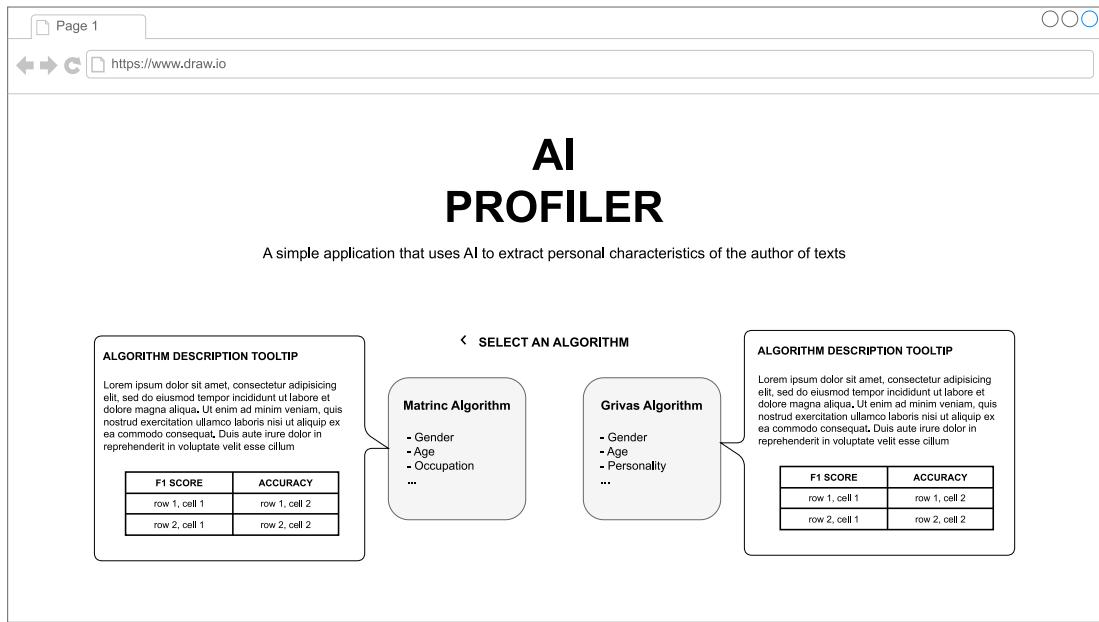


Figura 6.4: Prototipo de la selección del algoritmo de perfilado

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 comenzar con el proceso. 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.

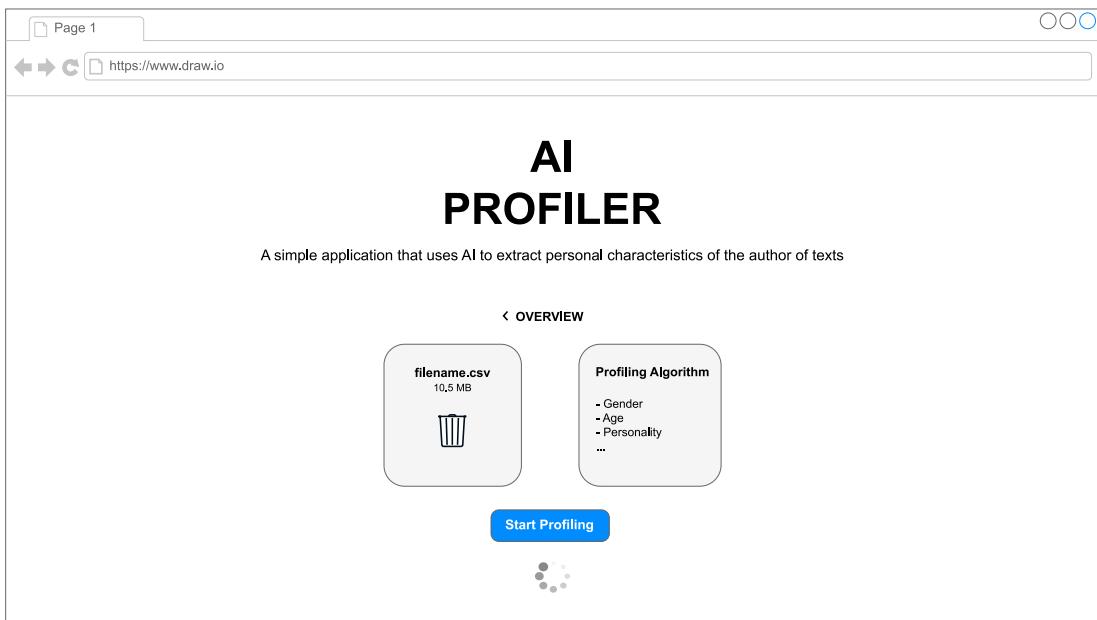


Figura 6.5: Prototipo del resumen del perfilado

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*.

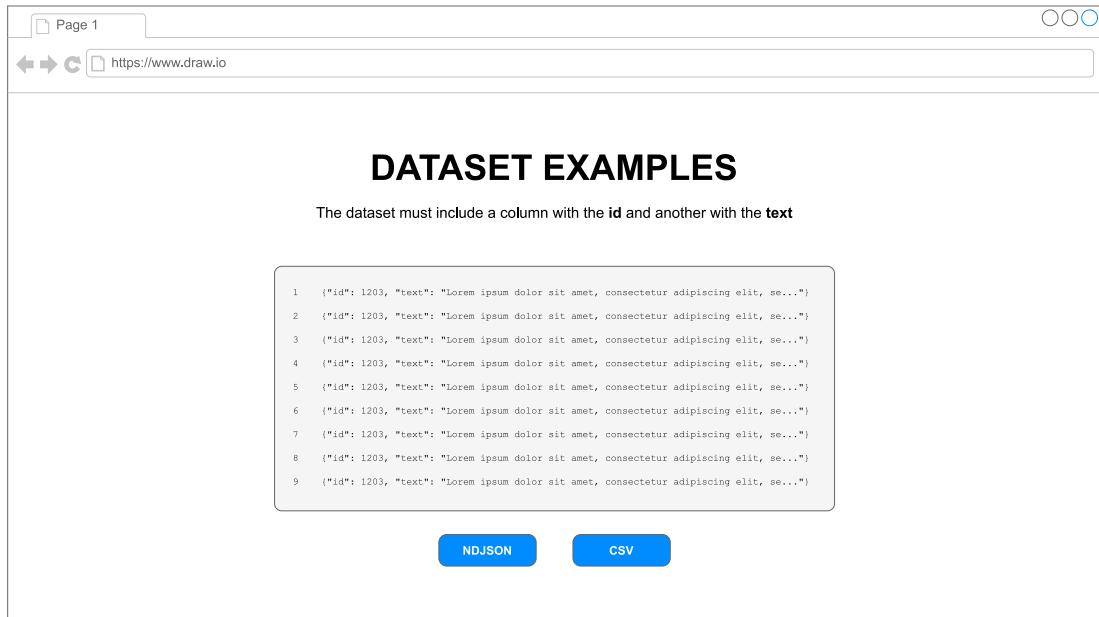


Figura 6.6: Prototipo de la pantalla de ejemplos de dataset

Cuando el perfilado haya finalizado, se mostrará al usuario un *dashboard* con los resultados obtenidos, como se puede ver en las Figuras 6.7 y 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, como se establece en la historia de usuario H5. Además, la lista permitirá ser ordenada por cada uno de los diferentes campos, según se indica en la historia de usuario H6. 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 o el gráfico de anillo. 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 tendrá 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*.

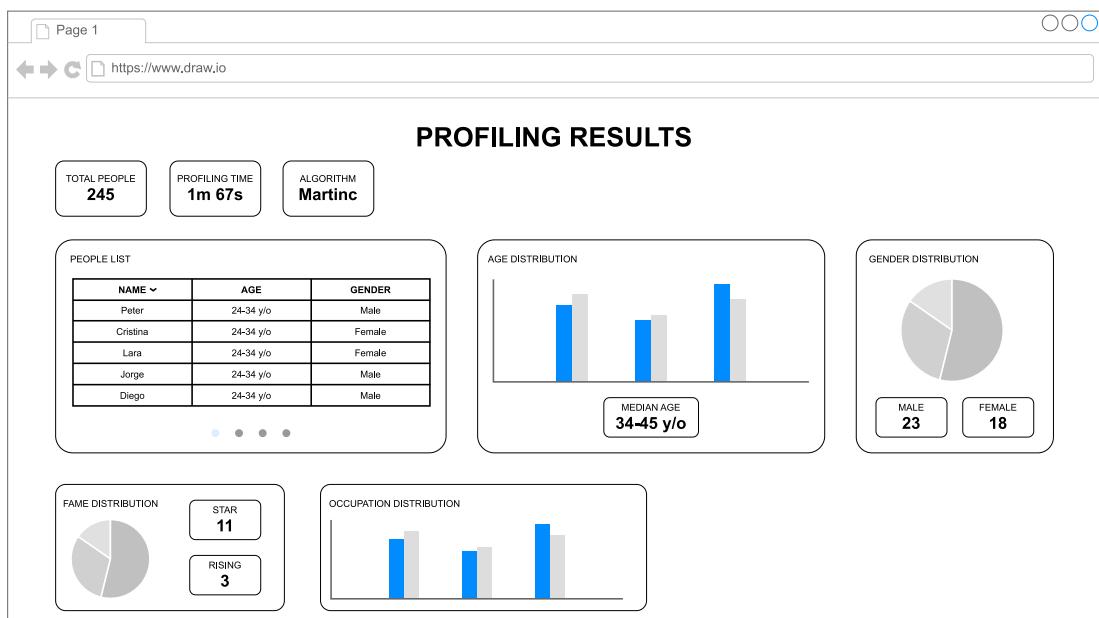


Figura 6.7: Prototipo del dashboard utilizando el algoritmo de Martinc et al. [2]

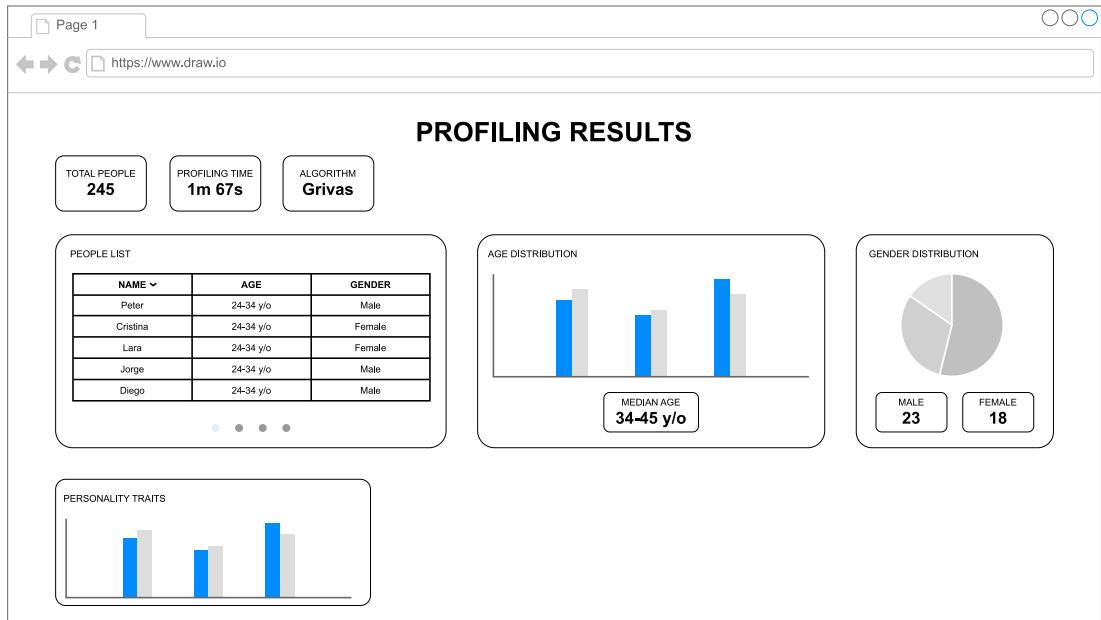


Figura 6.8: Prototipo del dashboard utilizando el algoritmo de Grivas et al. [1]

6.3 Backend

Para analizar la estructura del *backend*, se tomará como referencia el diagrama de clases de la Figura 6.11 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 6.10.

6.3.1 Endpoints

Dado que el *backend* seguirá una arquitectura REST, se explicarán a continuación los *endpoints* que se han implementado para el servidor, así como los parámetros que recibe cada uno. Como se puede observar en la Figura 6.10, 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, será en el cuerpo donde se envíe el archivo con los textos 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` [88] de Python (según el estándar RFC 4122 [89]) 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 [RNF2](#).

- **/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 da 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 6.9.

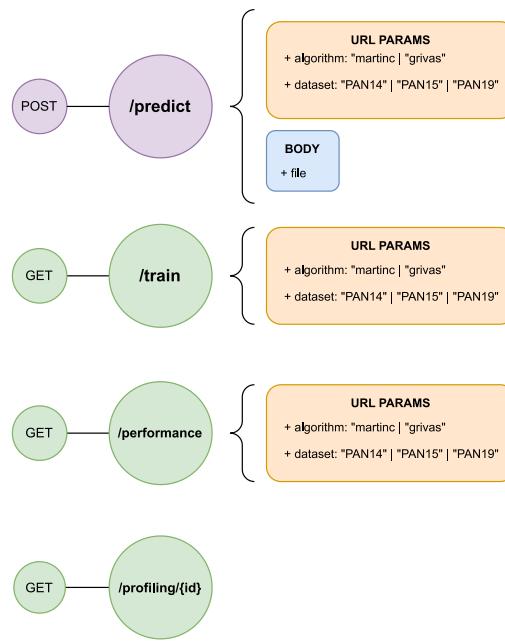
```

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 6.9: Estructura del JSON devuelto por el endpoint /profilings/{id}

Destacar que, en todos los casos, tanto el algoritmo de Grivas et al. [1] como el de Martinc et al. [2], tienen asociado un *dataset* de entrenamiento por defecto (que coincide con el utilizado por los autores originales en su publicación), el cual se utilizará en caso de no especificar ninguno en la URL.

Figura 6.10: Diagrama de *endpoints* del *backend*

6.3.2 Clases

Como se explicó en la Sección 3.1, la estructura del *backend* sigue los principios del patrón de diseño DDD [85], 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 [52].
- 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 orquestar las diferentes llamadas y delegar la lógica de negocio a las distintas entidades.

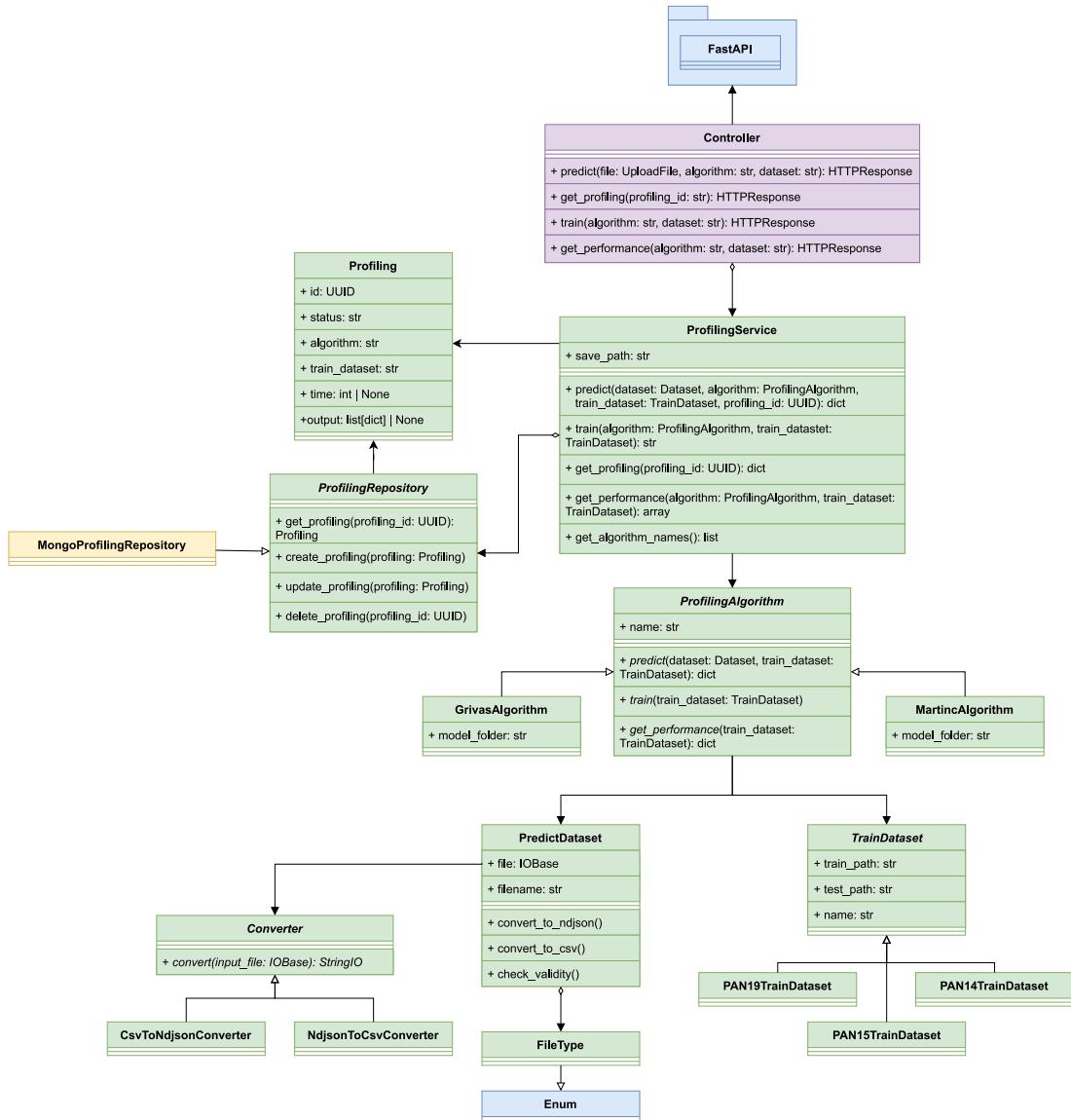
La más importante de ellas es la clase *ProfilingAlgorithm*, una clase abstracta que define la interfaz que deben implementar todos los algoritmos de perfilado incluyendo las tres funciones principales: *predict()*, *train()* y *get_performance()*. En este sentido, es importante mencionar que los algoritmos 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, explicada en profundidad en la Sección 7.1.

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 y validar el modelo. 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*, posibilitando 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 de bases de datos 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 el *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 las conexiones con la misma.

Figura 6.11: Diagrama de clases del *backend*

6.4 Frontend

En lo que respecta al *frontend*, como se explicó en la Sección 3.2, se optó por utilizar NextJS como *framework* de desarrollo, el cual está basado en React. Así, teniendo esto en cuenta y apoyándonos en el diagrama de clases de la Figura 6.12, analizaremos las clases más relevantes del proyecto y su relación con el resto de módulos y componentes.

Primero, como elemento principal que conforma la interfaz de usuario, estarían todas las páginas que se encuentran en el módulo *pages*, es decir, la página principal (*Home*) y la página

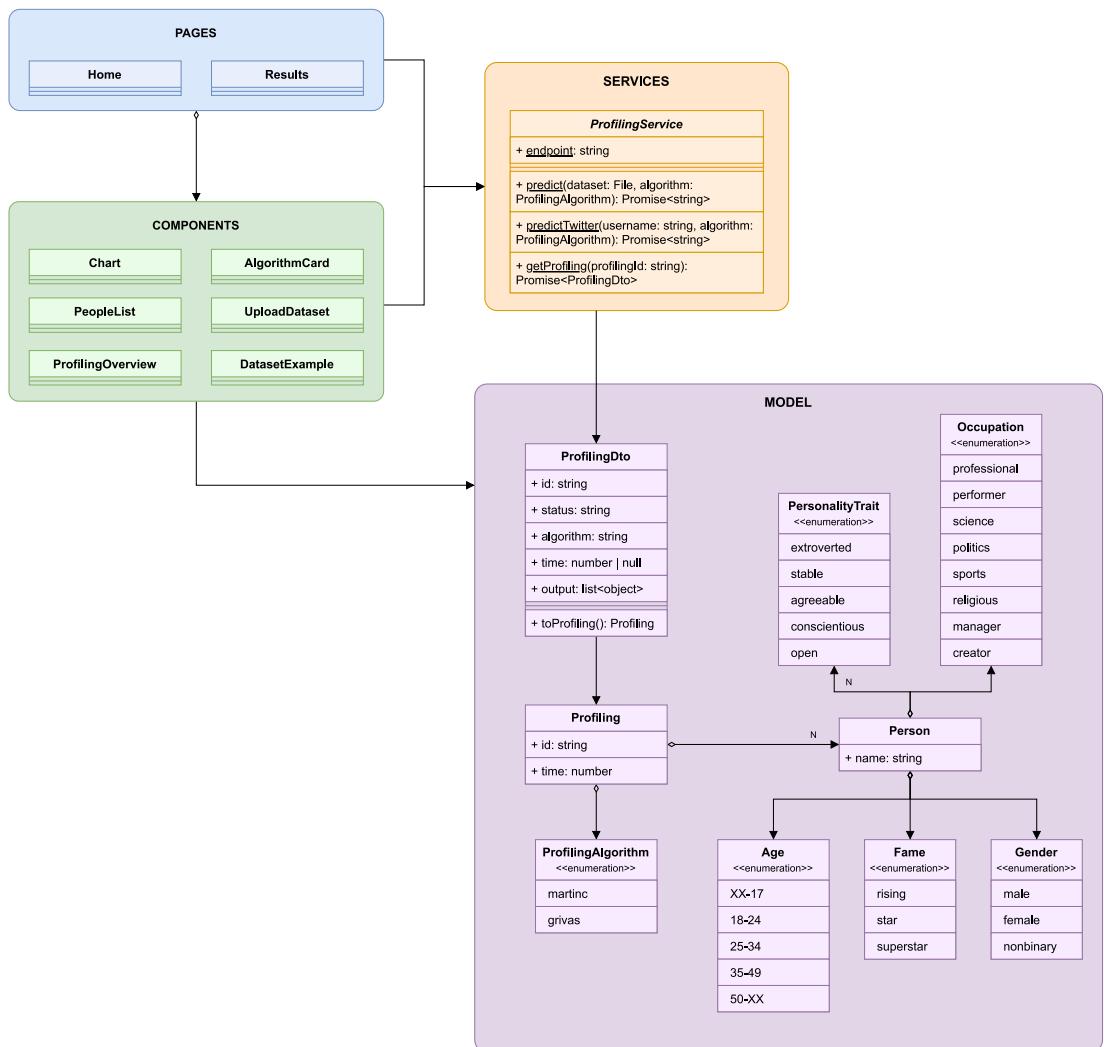
de resultados (*Results*).

Como se ve reflejado en el diagrama, cada página está formada por varios componentes, los cuales se almacenan dentro del módulo *components*. En este sentido, el componente *Chart* es uno de los más complejos e interesantes, dado que en él reside toda la lógica y la configuración común de los gráficos que se muestran en el *dashboard* de la página de resultados. Otros componentes importantes son el *UploadDataset*, el cual se encarga de gestionar los eventos de *drag and drop* y de la subida de archivos; el componente *AlgorithmCard*, que presenta la información de las características que perfila cada algoritmo y se ocupa de mostrar el *tooltip* con la información detallada; o el componente *PeopleList*, el cual se hace cargo de manejar la paginación, la ordenación y la selección de la lista de personas.

Es importante destacar aquí el hecho de que cada uno de los componentes tiene una hoja de estilos propia gracias a la utilización de los módulos CSS [90]. A su vez se buscó sacarle partido a SASS, haciendo uso de variables globales que definen colores o distancias; empleando el anidamiento para una mejor estructuración de los estilos; y utilizando los *mixins*, esto es, funciones que permiten reutilizar código CSS para, por ejemplo, simplificar la implementación de las *media queries*. En este sentido, para cumplir con el requisito no funcional de portabilidad RNF3, era necesario que la aplicación fuese *responsive*, es decir, adaptable a cualquier tamaño de pantalla, desde móviles a monitores.

Por otro lado, ya que algunos componentes y páginas necesitan realizar peticiones al *backend*, es necesario implementar un servicio que contenga la lógica de la comunicación y exponga funciones que faciliten su uso. De esta tarea se encarga la clase abstracta *ProfilingService*, la cual define funciones estáticas tales como *getProfiling()* o *predict()* que abstraen al resto de componentes de la lógica de las peticiones HTTP. Dicho servicio hace uso por debajo de *fetch*, una API nativa de JavaScript muy flexible que permite programar casi cualquier funcionalidad relacionada con peticiones HTTP. Asimismo, ya que la asincronía es un concepto muy extendido y necesario en el desarrollo web, se optó por usar las *Promises* de JavaScript, las cuales permiten realizar operaciones asíncronas de forma muy sencilla.

Por último, como toda aplicación tipada, es necesario definir un modelo de datos que represente cada entidad. En este sentido, se ha optado por el uso de enumeraciones o *enums* para representar la mayor parte de los datos, tales como los algoritmos de perfilado, los géneros, las edades o los rasgos personales. Asimismo, se ha creado una clase *Person* que aúna todas las características perfiladas de una persona junto a su nombre. Finalmente, para representar los datos enviados por el *backend*, se ha definido la clase *ProfilingDto*, la cual, en última instancia, permite realizar la conversión a la clase *Profiling* empleada por el resto de la aplicación.

Figura 6.12: Diagrama de clases del *frontend*

Capítulo 7

Implementación

Al lo largo de este capítulo, se expondrán las decisiones de implementación más relevantes tomadas durante el desarrollo de la aplicación, y se detallará la estructura de directorios que se ha seguido para organizar el código fuente del proyecto. Finalmente, se hablará de la importancia del código abierto y de como contribuye a la innovación en el campo de la informática.

7.1 Adaptación de los algoritmos

Dado que el código fuente de los algoritmos de perfilado seleccionados estaba disponible para su uso, fue necesario adaptarlo para ofrecer una interfaz común a través de la cual se pudiese comunicar el resto de la aplicación. Para ello se barajaron en un inicio dos aproximaciones diferentes. La primera de ellas pasaba por implementar APIs sencillas en cada uno de los algoritmos, es decir, crear microservicios que se encargasen de recibir las peticiones y devolver los resultados al *backend*. La segunda aproximación consistía en adaptar el propio código fuente para que cada algoritmo formase parte de una clase que implemente la interfaz mencionada en el diagrama de la Figura 6.11 (*ProfilingAlgorithm*). Finalmente, se optó por esta segunda opción ya que, a pesar de ser más compleja, permitía una mayor integración con el resto de la aplicación y un mejor rendimiento debido a la eliminación de la latencia de red. Además, podíamos aprovechar el hecho de que los algoritmos estaban programados en Python para utilizarlos directamente en el *backend* sin hacer uso de llamadas interlenguaje.

Sin embargo, esta decisión implicó comprender a muy bajo nivel como estaban implementados dichos algoritmos para realizar las modificaciones necesarias e implementar nuevas funciones. Esta adaptación del código fue, en el caso del algoritmo de Grivas et al. [1], bastante compleja, ya que conllevó sustituir librerías obsoletas, reimplementar funciones y resolver los cambios disruptivos (*breaking changes* en inglés) que suponía dar el salto desde la versión 2.7 de Python a la 3.10.

7.2 Estructura del proyecto

En relación a la estructura del proyecto, se ha optado por agrupar el código fuente de todo el sistema bajo un mismo directorio, como se muestra en el diagrama de la Figura 7.1, dado que facilita en gran medida su portabilidad y su mantenimiento al tratarse de un proyecto relativamente pequeño.

En primer lugar, situados en la raíz del proyecto, podemos diferenciar las carpetas principales que lo conforman, esto es, *backend*, *fronted* y *datasets*, así como dos archivos:

- ***README.md***: Contiene un resumen en formato Markdown [91] de lo que es la aplicación, de sus características y de la forma de ejecutarla.
- ***docker-compose.yml***: Es el archivo que almacena las configuraciones de los contenedores de Docker que se deben levantar para que el sistema funcione adecuadamente.

Continuando con el directorio *backend*, distinguimos varias carpetas y archivos:

- ***requirements.txt***: Contiene todas las dependencias de Python que necesita el *backend* para funcionar.
- ***Dockerfile***: Es el archivo en el que se definen los pasos necesarios, es decir, los comandos a ejecutar en una máquina recién instalada, para construir la imagen de Docker del *backend*.
- ***docker-compose.yml***: A diferencia del situado en la raíz, este archivo contiene las configuraciones de los contenedores necesarios para el correcto funcionamiento del entorno de desarrollo en el *backend*.
- ***/venv***: Dado que el proyecto está creado haciendo uso de un entorno virtual para aislar las dependencias de Python, este directorio contiene los archivos para su puesta en marcha.
- ***/src***: Aquí se almacena todo el código fuente del *backend*, donde podemos encontrar las siguientes carpetas principales:
 - ***/application***: Contiene los controladores y, en general, lo que se encarga de gestionar las comunicaciones exteriores con el sistema.
 - ***/domain***: Contiene las clases que conforman la capa de dominio, es decir, las entidades, los repositorios, los servicios, los algoritmos y los conversores.

- **/infrastructure:** Contiene los repositorios de tecnologías concretas que implementan las interfaces definidas en la capa de dominio.
- **/utils:** Contiene clases de utilidad requeridas en diversas partes del sistema.
- **main.py:** Se corresponde con el punto de entrada de la aplicación.

Por otro lado, la estructura que sigue el *frontend*, muy ligada al *framework* de desarollo NextJS, es la siguiente:

- **package.json:** En este archivo se almacenan todas las dependencias de NodeJS que requiere el *frontend*.
- **next.config.js:** La importancia de este archivo, más allá de contener las configuraciones básicas de NextJS, reside en la posibilidad de definir un *reverse proxy* capaz de redirigir las peticiones al *backend* a la máquina local durante el desarrollo o a la máquina remota, en este caso otro contenedor de Docker, en producción.
- **Dockerfile:** Archivo análogo al presente en el directorio *backend*, necesario para la construcción de la imagen del *frontend*.
- **/src:** Dentro de esta carpeta, nos encontramos la estructura típica de un proyecto en NextJS:
 - **/components:** Contiene los componentes que conforman las páginas de la aplicación.
 - **/model:** Contiene las clases que representan las entidades del sistema en el *frontend*.
 - **/pages:** Contiene las páginas navegables de la aplicación, estructuradas en directorios según su ruta.
 - **/services:** Contiene los servicios encargados de abstraer las peticiones que se realizan al *backend*.
 - **/styles:** Contiene las hojas de estilos globales de la aplicación.
 - **/utils:** Contiene funciones de diversa utilidad en el proyecto.

Finalmente, el directorio *datasets* se corresponde con el lugar donde se agrupan todas las colecciones que se han utilizado para el entrenamiento y la validación de los modelos. Entre ellos se incluye una gran parte de *datasets* ofrecidos por PAN [24] en las competiciones de perfilado de autor que organiza, además de otras como la del movimiento #BLM ofrecida en este trabajo y de la que se profundizará más en la Sección 8.2.

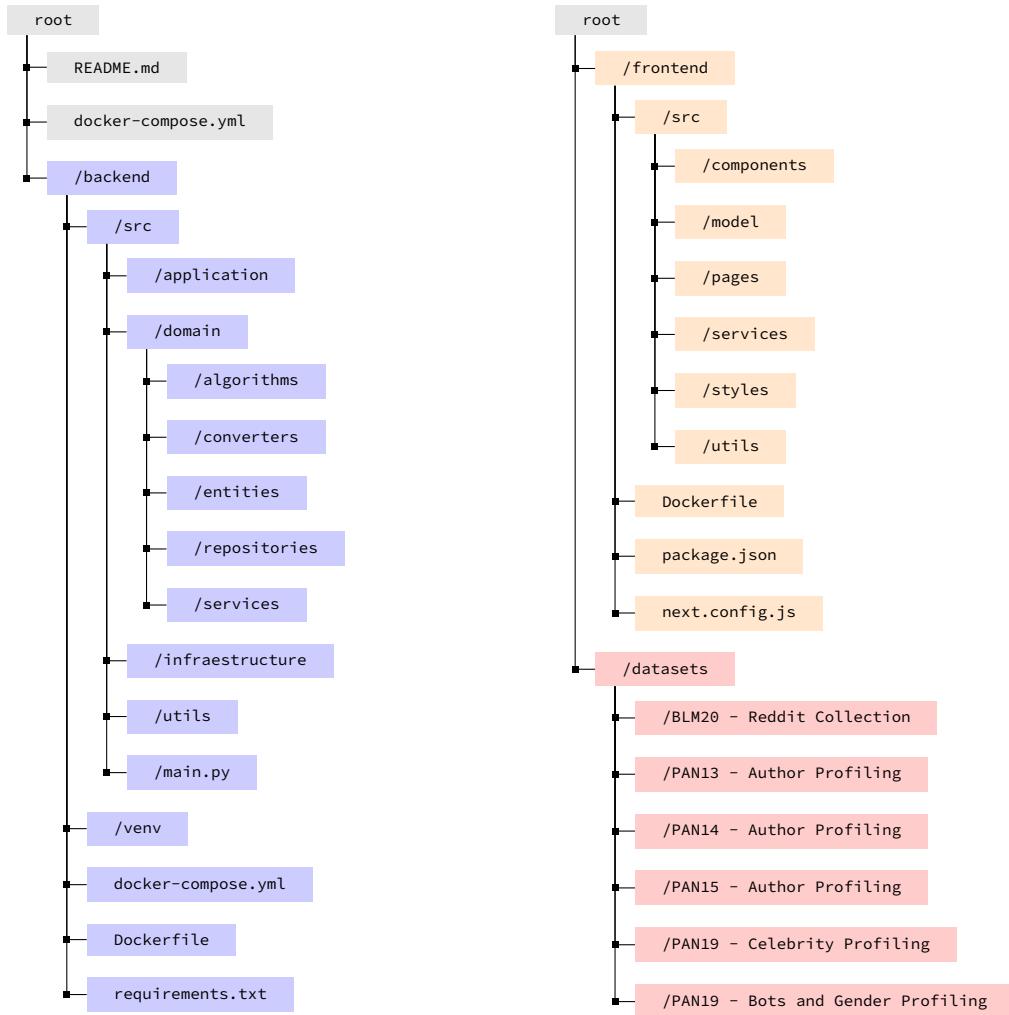


Figura 7.1: Estructura de directorios del proyecto

7.3 Publicación del código fuente

Una vez explicado el funcionamiento a bajo nivel de la aplicación, es necesario valorar la forma de publicar y distribuir el código fuente de la misma.

En este sentido, existen dos enfoques principales: el código abierto y el código privado. El código privado (*closed source* o *proprietary source* en inglés) es aquel que no está disponible para el público en general, es decir, que no se puede acceder a él ni modificarlo, por lo que normalmente se asocia al mundo empresarial y a la venta de licencias para su uso. Por otro lado, el código abierto (*open source* en inglés) es aquel que está disponible públicamente y puede ser modificado, utilizado y distribuido por cualquier persona en función de la licencia bajo la que esté publicado. Normalmente, el código abierto se asocia con la filosofía del *software libre*.

y con la gratuidad de la aplicación, aunque esto no siempre es así.

En nuestro caso, creemos que la mejor forma de publicar el código fuente de la aplicación es hacerlo de forma abierta, favoreciendo a una mejora continua de la misma por parte de la comunidad y a ofrecer la posibilidad de que cualquiera pueda utilizarla y adaptarla a sus necesidades.

Sin embargo, para que esto sea posible, como se mencionó anteriormente, es necesario elegir una licencia bajo la cual publicarla. Así, existen licencias como la GPL (*General Public License* en inglés) [92] que permiten a los usuarios hacer uso del código, modificarlo y distribuirlo siempre y cuando se mantenga la misma licencia. Por otro lado, existen licencias como la MIT [93] o la Apache [94] que ofrecen una mayor libertad, permitiendo incluso la utilización del código en proyectos privados. Por lo tanto, teniendo en cuenta que la filosofía de este proyecto es la de ofrecer una herramienta útil y gratuita a la mayor cantidad de usuarios, sin importar si sus fines son comerciales o no, se ha optado por utilizar la licencia MIT. Asimismo, también se consideró la opción de publicar el código bajo una doble licencia, pero se descartó por las complicaciones legales que suponía y por la confusión que podría generar.

En lo que respecta al lugar de publicación, se ha decidido hacer público el repositorio almacenado en GitHub [12], una de las plataformas más populares para el alojamiento de proyectos de software con más de 100 millones de desarrolladores [95]. Además, GitHub ofrece una gran cantidad de herramientas que facilitan el desarrollo colaborativo, como la posibilidad de crear *issues* para reportar errores o sugerir mejoras o la opción realizar *pull requests* para proponer cambios en el código.

Por último, para que la comunidad pueda contribuir al proyecto de la mejor forma posible, se ha creado una guía de contribución en la que se detallan los pasos a seguir para proponer cambios en el código, así como las normas de estilo que se deben cumplir.

Para acceder al repositorio, se puede hacer uso del siguiente enlace: <https://github.com/daavidrgz/ai-profiler>.

Capítulo 8

Caso de uso: #BLM

PARA mostrar el funcionamiento de la aplicación, a lo largo de este capítulo se desarrollará un caso de uso real y se comentará paso a paso.

8.1 Puesta en marcha del sistema

Como se mencionó en la Sección 3.4, para poder ejecutar la aplicación es necesario tener instalado Docker y Docker-Compose. Una vez instalados, simplemente se debe ejecutar el siguiente comando en la raíz del proyecto:

```
1 docker-compose up
```

A partir de este momento, Docker se encargará de descargar las imágenes necesarias para poner en marcha el sistema y de levantar los contenedores. Una vez finalizado el proceso, se podrá acceder a la aplicación de forma local en la URL: <http://localhost:3000>.

8.2 Dataset del movimiento #BLM

Como se comentó en el Capítulo 1, en las redes sociales se genera una gran cantidad de información y se debate sobre diversos temas. Por ello, numerosos investigadores han utilizado esta información para crear lo que se conoce como "archivos sociales" (Pybus et al., 2015 [96]; Acker et al., 2014 [97]; Ruiz et al., 2020 [98]), que buscan preservar determinados aspectos de la interacción en las redes sociales y servir como base para futuras investigaciones y estudios.

En este contexto, los tutores de este trabajo, utilizando una metodología propia para la creación de colecciones de referencia a modo de archivos sociales (Otero et al., 2021) [99], han elaborado un *dataset* sobre el fenómeno social que se produjo tras la muerte de George Floyd

en mayo de 2020 conocido como *Black Lives Matter*, que implicó protestas en todo el mundo en las que se denunciaba la brutalidad policial y el racismo sistémico en Estados Unidos. Esta colección, disponible tanto en español como en inglés, recoge más de 260.000 posts referentes a más de 90.000 usuarios de la red social Reddit que compartieron contenido sobre este fenómeno en el periodo de aproximadamente un año.

Sin embargo, debido a que la colección estaba formada por varios archivos diferentes en formato XML, donde unos contenían información de los hilos de conversación en Reddit y otros posts de los usuarios que interactuaban en dichos hilos, fue necesario procesarlos y unificarlos en un *dataset* con un formato sencillo y aceptado por la aplicación. Así, se generaron tres *datasets*, todos ellos en inglés, en formato NDJSON, cada uno con un número diferente de posts para cada usuario (50, 500 y 1000) con el fin de analizar las posibles diferencias en cuanto a los resultados obtenidos por el perfilado. Además, destacar que, ya que ninguno de estos tres *datasets* están etiquetados, no conocemos a priori la distribución de cada característica y, por lo tanto, no podemos obtener una medida del balanceo entre clases.

8.3 Subida del *dataset*

Una vez contamos con el *dataset* que vamos a procesar para el perfilado, el primer paso es subirlo a la aplicación. Para ello, en la página de inicio mostrada en la Figura 8.1, se puede observar un campo para la subida del *dataset*, que será el que empleemos. Destacar que, en este caso y a lo largo del resto de secciones de este capítulo, se mostrarán las capturas de pantalla tanto de la versión de escritorio (izquierda) como de la versión móvil (derecha).

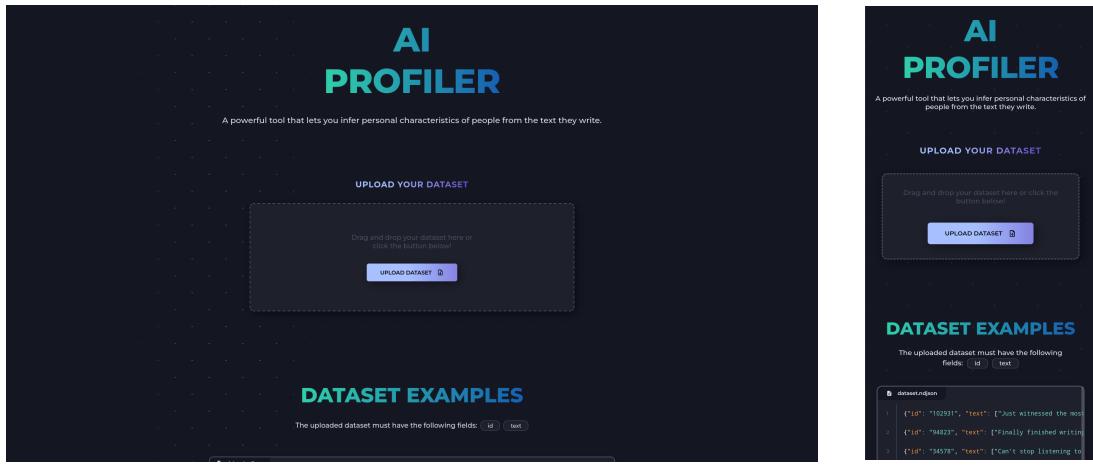


Figura 8.1: Página de inicio de la aplicación

Asimismo, si el usuario desea conocer cual es la estructura y los campos que debe contener

el *dataset* que va a subir, puede hacerlo consultando la sección de ejemplos, mostrada en la Figura 8.2, visible tras hacer *scroll* en la misma página de inicio.

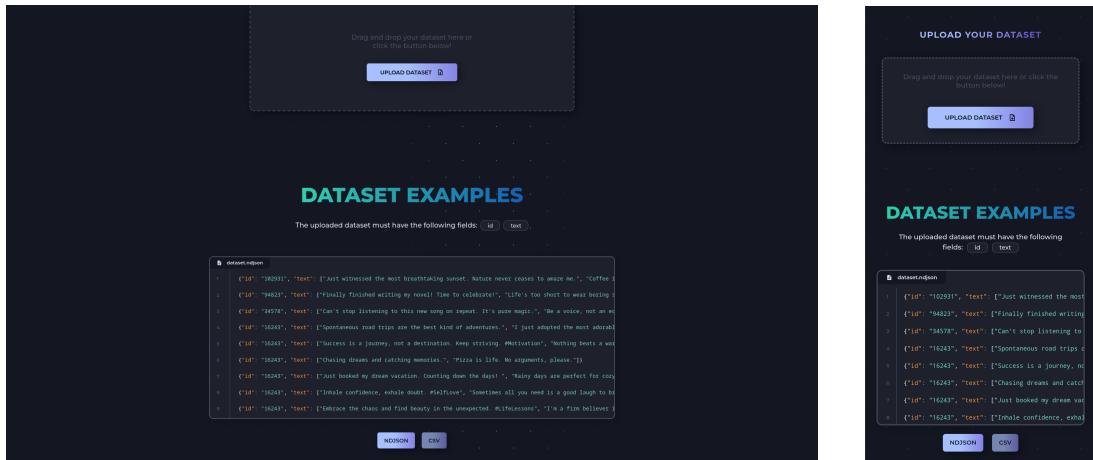


Figura 8.2: Página de ejemplos de *datasets*

8.4 Selección del algoritmo

En este punto, una vez subido el *dataset* deseado, el usuario debe seleccionar el algoritmo de perfilado que más se ajuste a sus necesidades. En este decisión, es fundamental tener en cuenta el lenguaje en el que se encuentra el *dataset*, las características que se buscan perfilar, el rendimiento del algoritmo o incluso el *dataset* con el que ha sido entrenado. Así, por un lado se muestran unas tarjetas con la información básica de cada algoritmo, como se puede ver en la Figura 8.3 y, por otro lado, se muestran *tooltips* con información más detallada sobre el funcionamiento del algoritmo, el *dataset* de entrenamiento y el rendimiento del mismo, como se distingue en la Figura 8.4.

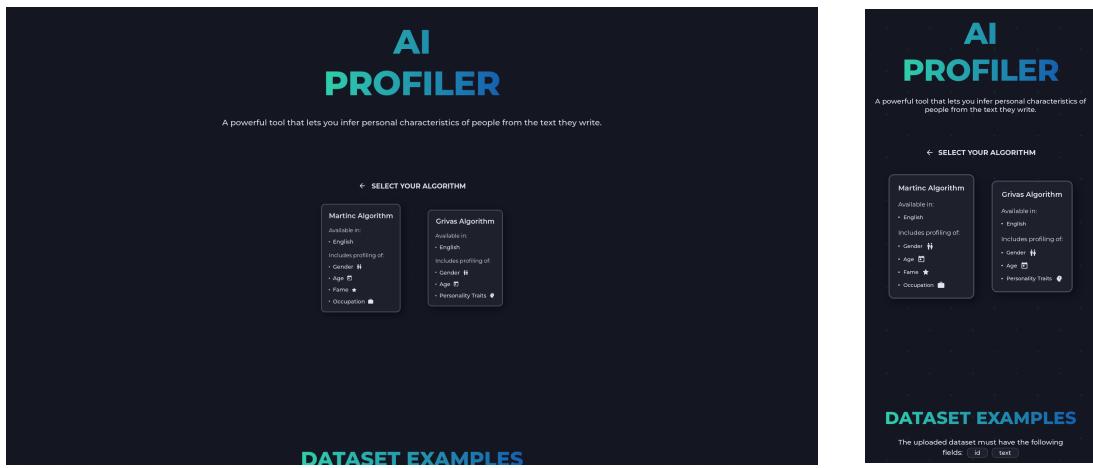


Figura 8.3: Página de selección algoritmos

This screenshot shows detailed information for the Martinc and Grivas algorithms. The Martinc section includes a table for classification scores and a reference to a paper. The Grivas section includes a table for regression scores and a reference to a paper. Both sections provide links to the original academic papers.

CLASS	ACCURACY	F1
Age	0.63988	0.63524
Gender	0.90211	0.90101
Occupation	0.73357	0.71881
Fame	0.75552	0.73296

CLASS	REGRESSION
Extroverted	0.12365
Stable	0.18683
Agreeable	0.13595
Conscientious	0.11283
Open	0.11656

Figura 8.4: Tooltip de información sobre los algoritmos

8.5 Proceso de perfilado

Después de seleccionar el algoritmo, se muestra una página a modo de resumen, en la que se puede observar el *dataset* subido junto al algoritmo seleccionado. Para comenzar el proceso de perfilado, el usuario debe pulsar el botón *Start profiling* y, para proporcionar *feedback* sobre la ejecución del proceso y mejorar la experiencia de usuario, se muestra una barra de progreso infinita. Esta página puede verse en la Figura 8.5.

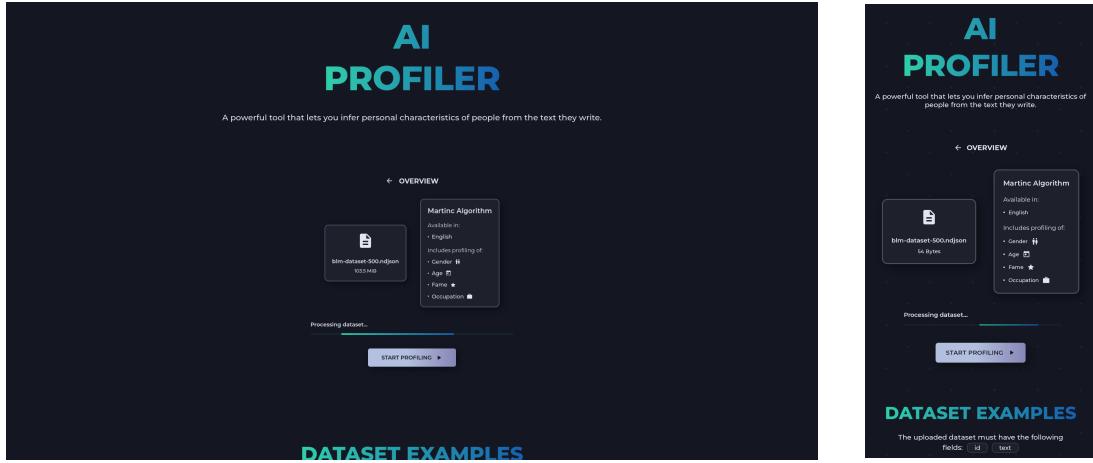


Figura 8.5: Página de resumen del perfilado

8.6 Visualización de resultados

Tras finalizar el proceso de perfilado, se muestra una página con los resultados obtenidos en formato *dashboard*, al igual que se especificaba en los prototipos de la Sección 6.2. De esta forma, el *dashboard* fue implementado teniendo como objetivo principal que toda la información estuviera disponible en una única página en la que se mostraran únicamente datos relevantes para el usuario, sin necesidad de hacer *scroll* para verla. Asimismo, en función del algoritmo empleado, se muestran unos gráficos u otros, como se puede ver en las Figuras 8.6 y 8.7.



Figura 8.6: Dashboard con los resultados obtenidos por el algoritmo de Martinc [2]



Figura 8.7: Dashboard con los resultados obtenidos por el algoritmo de Grivas [1]

8.7 Análisis de resultados

Para llevar a cabo el análisis de los resultados obtenidos, se abordará el estudio de cada característica por separado desde un marco sociológico y se compararán las predicciones realizadas por ambos algoritmos. A mayores, y a modo de experimentación para comprobar las diferencias subyacentes en el perfilado, se hará uso de tres *datasets* con diferente número de posts por usuario, como se comentó en la Sección 8.2.

8.7.1 Distribución de género

En cuanto a la distribución de género, como se aprecia en la Tabla 8.1, ambos algoritmos otorgan una gran mayoría al género masculino en cualquiera de los tres *datasets*.

En el caso del algoritmo de Grivas [1], observamos como, a medida que aumenta el número de posts por usuario, el número de usuarios clasificados como femeninos disminuye drásticamente. Como ya se adelantó en la Sección 2.2.2, este fenómeno se debe a que el *dataset* utilizado para entrenar dicho algoritmo era muy limitado, de apenas 150 personas, por lo que no es lo suficientemente representativo y no generaliza de forma óptima.

En lo que respecta al algoritmo de Martinc [2], vemos como, al aumentar el número de posts por usuario, aquellos clasificados como masculinos se estabilizan alrededor de 1,320 (94%), mientras que aquellos clasificados como femeninos se quedan en 82 (6%). Estos resultados, a simple vista, no tienen una relación clara con la realidad dado que, según Statista, la distribución de género en Reddit es de un 64% para los hombres y un 36% para las mujeres [100]. Sin embargo, es importante resaltar que la proporción de género puede variar ampliamente en función del tema del que se esté hablando. Así, como recoge Thelwall et al., 2019

[101], existen amplias diferencias en cuanto a la participación de cada género en temas políticos, y es que, la proporción de mujeres en *subreddits* como *Politics*, en el que se tratan temas de actualidad política en Estados Unidos, es de tan solo un 20,5%. A raíz de este hecho, podemos inferir que existió una menor participación de las mujeres en el movimiento #BLM en el marco de las redes sociales, lo que, a su vez, permite trazar como hipótesis de trabajo si en las manifestaciones y protestas sucedidas en la vida real existía también una menor participación del género femenino.

Por otro lado, entendiendo a las limitaciones del propio modelo, podemos destacar el hecho de que el conjunto de entrenamiento está formado por *tweets* de celebridades, haciendo que pueda no ser directamente aplicable a cualquier conjunto de usuarios o redes sociales diferentes.

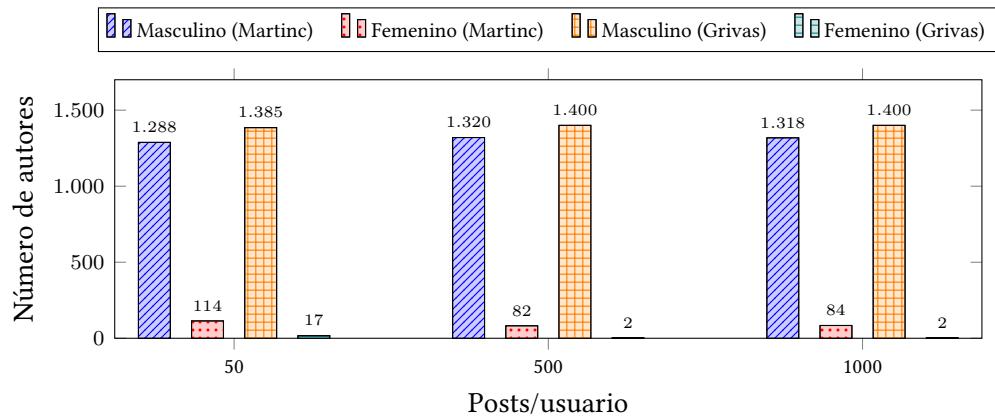


Figura 8.8: Distribución de género obtenida por ambos algoritmos

Posts/usuario	Martinc		Grivas	
	Masculino	Femenino	Masculino	Femenino
50	1.288	114	1.385	17
500	1.315	87	1.400	2
1.000	1.318	84	1.400	2

Tabla 8.1: Distribución de género obtenida por ambos algoritmos

8.7.2 Distribución de edad

En el caso de la distribución de edad, ambos algoritmos reflejan amplias diferencias en los resultados obtenidos, como se puede ver en las Tablas 8.2 y 8.3.

Para el algoritmo de Martinc et al. [2], observamos como, a medida que aumenta el número de posts por usuario, el número de usuarios pertenecientes a la franja de edad 25-34 disminuye considerablemente, mientras que los pertenecientes a los rangos 35-49 y 50-XX fluctúan. La principal conclusión que se extrae de esto es que el modelo, basado, recordemos, en TF-IDF, consigue encontrar patrones lingüísticos relacionados con rangos de edad más altos cuantos más posts tenga para predecir. Otra hipótesis gira entorno a la posibilidad de que al contar con más posts y, por ende, con más información por usuario, el algoritmo sea capaz de realizar una clasificación más precisa, debido al gran rendimiento que mostraba en cuanto a precisión y al elevado número de usuarios utilizados para el entrenamiento, cerca de 20.000. Sin embargo, es importante destacar que los rangos con más número de usuarios coinciden con aquellos más representados en el conjunto de entrenamiento, como se representaba en la Figura 2.5, lo que denota quizá problemas a causa del desbalanceo.

En cuanto a la distribución real de la edad, la encuesta llevada a cabo por Statista a la población estadounidense adulta en 2021, revela que los usuarios de entre 18 y 29 años conforman el 36% de la plataforma de Reddit, los usuarios de entre 30 y 49 años el 22% y los mayores de 50 el 13% [102]. Esto contrasta en gran medida con los resultados obtenidos, lo que puede tener varias implicaciones.

La primera de ellas gira en torno al hecho de que sea la gente más mayor la que más se exprese en Reddit sobre el #BLM y la que más empatize con el movimiento. Esto puede deberse a que la gente más joven no haya vivido tantos episodios de racismo y, por lo tanto, no se sienta tan identificada con las protestas. Esta hipótesis es respaldada por diferentes estudios como el realizado por Thernstrom et al., 1998 [103], en el que se puede observar un claro descenso de racismo en EEUU en diferentes ámbitos. Así, como se recoge en el estudio, en 1958 el 44% de los estadounidenses blancos afirmaban que se mudarían si una familia de raza negra se convirtiera en su vecina, mientras que en 1998 este porcentaje se redujo al 1%. Asimismo, en 1964 tan solo el 18% de los estadounidenses blancos admitían tener compañeros de raza negra; en 1998 ya el 86% afirmaba tenerlos.

La segunda hipótesis tiene que ver con el propio modelo utilizado, y es que, como se mencionó anteriormente, el conjunto de *tweets* de celebridades utilizados para el entrenamiento de este algoritmo, puede no ser directamente aplicable al análisis de este caso de uso. Esto es así debido a las diferencias notables que existen en la forma de expresarse de cada grupo demográfico, ya que las celebridades hacen uso, normalmente, de un lenguaje más correcto, claro y sin errores, al contrario de lo que puede suceder con un usuario medio de cualquier red social.

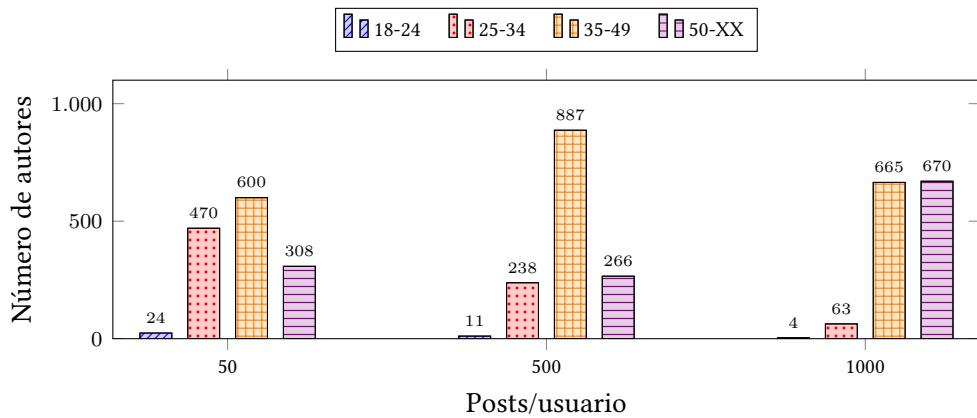


Figura 8.9: Distribución de edad obtenida por el algoritmo de Martinc et al. [2]

Martinc					
Posts/usuario	XX-17	18-24	25-34	35-49	50-XX
50	0	24	470	600	308
500	0	11	238	887	266
1,000	0	4	63	665	670

Tabla 8.2: Distribución de edad obtenida por el algoritmo de Martinc et al. [2]

Pasando al algoritmo de Grivas et al. [1], y de la misma forma que sucedía con la distribución de género, los resultados obtenidos parecen estar la mayor parte concentrados en dos clases, las cuales coinciden con las más representadas en la colección de entrenamiento para el algoritmo (ver Figura 2.2). Esto denota, lógicamente, un claro sobreajuste del modelo a dicha colección. **A pesar de ello, todas estas inferencias desde los datos a nivel sociológico representan en sí mismo contribuciones de este TFG, ya que abren nuevas hipótesis de trabajo acerca del fenómeno #BLM.**

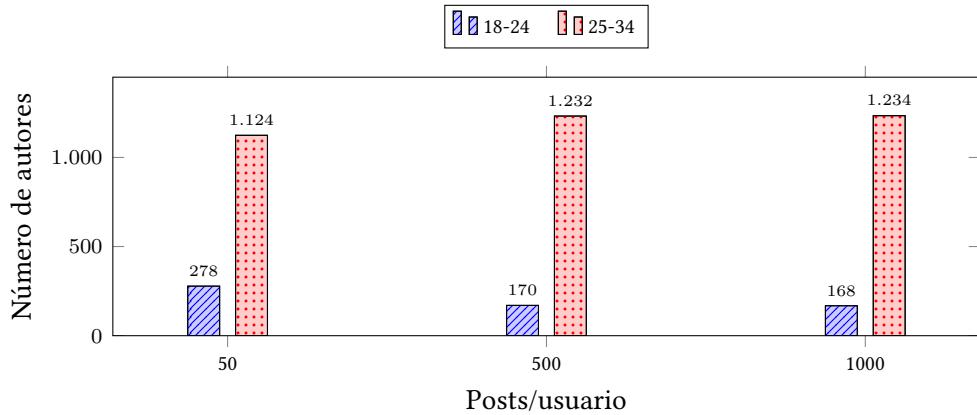


Figura 8.10: Distribución de edad obtenida por el algoritmo de Grivas et al. [1]

Grivas					
Posts/usuario	XX-17	18-24	25-34	35-49	50-XX
50	0	278	1.124	0	0
500	0	170	1.232	0	0
1.000	0	168	1.234	0	0

Tabla 8.3: Distribución de edad obtenida por el algoritmo de Grivas et al. [1]

8.7.3 Otras características

En lo que respecta al perfilado de los rasgos personales que ofrece el algoritmo de Grivas et al. [1], lo primero que observamos es que existe muy poca variación con respecto al número de posts por usuario, como se puede ver en la Tabla 8.4. A grandes rasgos, recordemos que los valores de cada rasgo personal oscilan entre -0,5 y 0,5, vemos como la característica que más destaca es claramente la de *Open* (abierto a nuevas experiencias), seguida de *Stable* (estable) y *Agreeable* (amable). Las implicaciones que tiene estos resultados con la realidad no son claras, pero podemos deducir que, en general, los usuarios que se lanzan a ser partícipes de los debates en Reddit sobre el #BLM son personas categorizadas en unos rasgos de personalidad abiertos y estables.

	Martinc				
Posts/usuario	<i>Extroverted</i>	<i>Stable</i>	<i>Agreeable</i>	<i>Conscientious</i>	<i>Open</i>
50	0,1630	0,1956	0,1877	0,1787	0,3271
500	0,1536	0,1974	0,1968	0,1771	0,3349
1,000	0,1529	0,1972	0,1969	0,1769	0,3358

Tabla 8.4: Distribución media de los rasgos personales obtenidos por el algoritmo de Grivas et al. [1]

Finalmente, en lo que respecta a las características únicas perfiladas por el algoritmo de Martinc et al. [2], es decir, la ocupación y el nivel de fama, creemos que no aportan información relevante para este análisis, dado que son características pensadas específicamente para el perfilado de celebridades. A pesar de ello, se pueden ver las distribuciones obtenidas en las Tablas 8.5 y 8.6.

	Martinc								
Posts/usuario	<i>Professional</i>	<i>Performer</i>	<i>Science</i>	<i>Politics</i>	<i>Manager</i>	<i>Creator</i>	<i>Sports</i>	<i>Religious</i>	
50	4	545	37	19	4	359	434	0	
500	0	680	34	2	1	396	289	0	
1,000	2	691	73	3	1	321	311	0	

Tabla 8.5: Distribución de ocupación obtenida por el algoritmo de Martinc et al. [2]

	Martinc		
Posts/usuario	<i>Rising</i>	<i>Star</i>	<i>Superstar</i>
50	2	1.202	198
500	1	1.226	175
1.000	1	1.261	140

Tabla 8.6: Distribución de fama obtenida por algoritmo de Martinc et al. [2]

Capítulo 9

Planificación y costes

EN este capítulo se expondrá la planificación y distribución del trabajo en cada *sprint*. Así mismo, 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 seis *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 librería 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. A mayores, se buscó replicar los resultados que reportaban dichos algoritmos en sus publicaciones y se seleccionaron los que mejor rendimiento mostraban. Este proceso se detalla más a fondo en las Secciones 2.2 y 2.3.
- **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 los componentes principales.
- **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* ambos algoritmos seleccionados.

- **Sprint 5:** Durante el quinto *sprint*, del 8 al 28 de julio, se 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, contamos con un grupo de trabajo de tres personas, el autor de este documento y los tutores del trabajo, como se mencionó anteriormente en la 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 los recursos materiales asociados al proyecto, 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 (Kernel 6.1.12)

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

En lo que respecta a los costes, dado que todos los recursos de tipo software utilizados son gratuitos, no se ha incurrido en ningún coste asociado a ellos.

Por otro lado, ya que se hizo uso de un recurso material, es necesario calcular su amortización asociada. Para ello, según la Ley 27/2014 del Impuesto sobre Sociedades (LIS) [104], los equipos para procesos de información tienen asociado un porcentaje anual de amortización del 25% sobre el coste inicial del bien, por lo que, dado que el valor en el momento de compra

del portátil fue de aproximadamente 1,000€, la amortización anual ascendería a 250€. Este número, sin embargo, tiene que ser proporcional a su tiempo de uso, cercano a los cuatro meses, por lo que el resultado final sería de 83.30€.

Finalmente, 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 [105]. Además, el salario medio de un *Project Manager* del sector de las TIC es de unos 43,500€ anuales. Con estos datos y junto al cómputo total de horas calculado en la Sección 9.1, 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>	-	-	83,30€
<i>Total</i>	-	-	7.071,30€

Tabla 9.2: Coste del proyecto

Capítulo 10

Conclusiones

En este capítulo final se presentan las posibles mejoras que se podrían realizar en un futuro a la herramienta desarrollada, así como las lecciones aprendidas durante el transcurso de este trabajo.

10.1 Trabajo futuro

Mirando hacia el futuro, existen varias inquietudes y mejoras que se podrían realizar con respecto a la herramienta desarrollada.

En primer lugar, el problema que suponen los conjuntos de datos desbalanceados y poco representativos estadísticamente, como se ha comprobado en el Capítulo 8, es algo que perjudica en gran medida a los resultados obtenidos y limita los casos de uso de la aplicación a aquellos en los que se quieran predecir características de autores similares a los utilizados para entrenar los modelos. Así, uno de los objetivos principales sería del de conseguir un mayor y más variado número de *datasets* con los que entrenar los modelos haciendo uso de los algoritmos previamente implementados.

En segundo lugar, se podría mejorar la experiencia de usuario ofreciendo más formas de especificar la fuente de textos para el perfilado. Así, a mayores de un campo que permita la subida de un *dataset* almacenado en local, se podría incluir un campo que permita la introducción de un usuario de una red social como Twitter o Reddit, de forma que se puedan obtener los textos automáticamente y clasificar así a dicho usuario.

Por otro lado, a raíz de la aparición de los LLMs (*Large Langauge Model* en inglés) como GPT-3 (Brown et al., 2020) [106] y, posteriormente, con modelos como GPT-4 en 2023, se ha visto un gran avance en las técnicas de *Deep Learning* y se ha demostrado el gran potencial que ofrecen estas herramientas para el procesado de lenguaje natural. Tal es así, que estos

modelos podrían ser utilizados en el campo del perfilado de autores. Sin embargo, debido a la gran cantidad de recursos computacionales que actualmente requieren, en el caso de que su código haya sido liberado, o a la inversión de dinero que supondría su uso, en el caso de que sea privado, irían en contra de los principios fundamentales de este trabajo: ofrecer una herramienta gratuita y accesible a todo tipo de usuarios.

10.2 Lecciones aprendidas

El hecho de profundizar tanto en el campo del procesado del lenguaje natural y, más específicamente, en las técnicas de perfilado automático de autores, supuso un gran reto tanto desde el punto de vista teórico como desde el punto de vista de la propia investigación, a lo que se le suma la poca experiencia en el campo. De esta forma, a lo largo del proceso de investigación, se comprendieron las bases teóricas de técnicas como el TF-IDF o los n-gramas y se entendió a bajo nivel el funcionamiento de los algoritmos que mejores resultados obtenían en las competiciones analizadas.

Asimismo, ya que los objetivos del proyecto, más ambiciosos, no se quedaban en el terreno de la investigación sino que buscaban implementar una aplicación completa con la información adquirida, el desarrollo implicó un trabajo mucho más amplio abordando diversas ramas de la informática. Desde el diseño y la implementación de una interfaz web gráfica, pasando por la creación de un servidor web y el entrenamiento de modelos basados en algoritmos clásicos de aprendizaje automático, hasta el seguimiento de metodologías de desarrollo ágiles como Scrum, conllevó un gran esfuerzo junto a un aprendizaje continuo.

Más concretamente, se aprendió a desarrollar haciendo uso de Python en el *backend* de la aplicación, consolidándose como una opción ágil y fiable; se comprendió también la importancia de diseñar una buena interfaz y de ofrecer una buena experiencia de usuario; y se profundizó más en el terreno del *open source* y de las licencias que, de alguna forma, promueven algo fundamental en nuestro campo.

Por último, destacar que se adquirieron conocimientos de carácter interdisciplinar tras realizar el análisis del caso de uso del fenómeno #BLM, uno de los movimientos sociales de mayor relevancia y complejidad en los últimos años. Además, esta tarea condujo a poder conocer en mayor profundidad las redes sociales y las distribuciones demográficas de los usuarios que las forman.

En conclusión, se puede decir que los objetivos establecidos al inicio del proyecto se han cumplido y que, además, su desarrollo ha contribuido a poner en práctica los conocimientos

adquiridos durante la carrera, demostrando la capacidad de investigación y de desarrollo de una aplicación completa.

Bibliografía

- [1] A. Grivas, A. Krithara, and G. Giannakopoulos, ``Author profiling using stylometric and structural feature groupings." in *CLEF (Working Notes)*, 2015.
- [2] M. Martinc, B. Skrlj, and S. Pollak, ``Who is hot and who is not? profiling celebs on twitter." in *CLEF (Working Notes)*, 2019.
- [3] Łukasz Ryś, ``Organizing layers using hexagonal architecture, ddd, and spring," <https://www.baeldung.com/hexagonal-architecture-ddd-spring>, 2023.
- [4] M. We Are Social, ``Digital 2023 global overview report," 2023. [En línea]. Disponible en: <https://datareportal.com/reports/digital-2022-global-overview-report>
- [5] B. Sydney, ``The permanent campaign: Inside the world of elite political operatives," 1980.
- [6] 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.
- [7] B. Gallardo-Paúls and S. Enguix Oliver, *Pseudopolítica: el discurso político en las redes sociales*. Universitat de València, 2016.
- [8] A. Mislove, S. Lehmann, Y.-Y. Ahn, J.-P. Onnela, and J. Rosenquist, ``Understanding the demographics of twitter users," in *Proceedings of the international AAAI conference on web and social media*, vol. 5, no. 1, 2011, pp. 554--557.
- [9] A. Saxena and U. Khanna, ``Advertising on social network sites: A structural equation modelling approach," *Vision*, vol. 17, no. 1, pp. 17--25, 2013.
- [10] 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.

- [11] 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.
- [12] GitHub, ``Github," <https://github.com/>, 2008.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] M. Wiegmann, B. Stein, and M. Potthast, ``Overview of the celebrity profiling task at pan 2019." in *CLEF (Working Notes)*, 2019.
- [18] M. Á. Álvarez-Carmona, E. Guzmán-Falcón, M. Montes-y Gómez, H. J. Escalante, L. Villasenor-Pineda, V. Reyes-Meza, and A. Rico-Sulayes, ``Overview of mex-a3t at ibereval 2018: Authorship and aggressiveness analysis in mexican spanish tweets," in *Notebook papers of 3rd sepln workshop on evaluation of human language technologies for iberian languages (ibereval), seville, spain*, vol. 6, 2018.
- [19] S. Volkova and Y. Bachrach, ``On predicting sociodemographic traits and emotions from communications in social networks and their implications to online self-disclosure," *Cyberpsychology, Behavior, and Social Networking*, vol. 18, no. 12, pp. 726--736, 2015.
- [20] D. Preoțiuc-Pietro and L. Ungar, ``User-level race and ethnicity predictors from twitter text," in *Proceedings of the 27th international conference on computational linguistics*, 2018, pp. 1534--1545.
- [21] M. Fatima, K. Hasan, S. Anwar, and R. M. A. Nawab, ``Multilingual author profiling on facebook," *Information Processing & Management*, vol. 53, no. 4, pp. 886--904, 2017.
- [22] J. D. Burger, J. Henderson, G. Kim, and G. Zarrella, ``Discriminating gender on twitter," in *Proceedings of the 2011 conference on empirical methods in natural language processing*, 2011, pp. 1301--1309.

- [23] M. Gjurković and J. Šnajder, ``Reddit: A gold mine for personality prediction," in *Proceedings of the second workshop on computational modeling of people's opinions, personality, and emotions in social media*, 2018, pp. 87--97.
- [24] P. Organizers, ``Pan (plagiarism analysis, authorship identification, and near-duplicate detection)," <https://pan.webis.de/>, 2023.
- [25] 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.
- [26] 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.
- [27] 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.
- [28] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, ``Building a large annotated corpus of english: The penn treebank," *Technical Reports (CIS)*, 1993.
- [29] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, ``Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391--407, 1990.
- [30] I. Organizers, ``Iberlef (iberian languages evaluation forum)," <https://sites.google.com/view/iberlef2022/home>, 2022.
- [31] S. Maharjan, P. Shrestha, and T. Solorio, ``A simple approach to author profiling in mapreduce—notebook for pan at clef 2014," *Cappellato et al.[6]*, 2014.
- [32] E. R. Weren, V. P. Moreira, and J. P. de Oliveira, ``Exploring information retrieval features for author profiling—notebook for pan at clef 2014," *Cappellato et al.[6]*, 2014.
- [33] A. P. López-Monroy, M. Montes-y Gómez, H. J. Escalante, and L. V. Pineda, ``Using intra-profile information for author profiling." in *CLEF (Working Notes)*, 2014, pp. 1116--1120.
- [34] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, ``Liblinear: A library for large linear classification," *the Journal of machine Learning research*, vol. 9, pp. 1871--1874, 2008.

- [35] L. R. Goldberg, ``An alternative" description of personality": the big-five factor structure." *Journal of personality and social psychology*, vol. 59, no. 6, p. 1216, 1990.
- [36] B. Rammstedt and O. P. John, ``Measuring personality in one minute or less: A 10-item short version of the big five inventory in english and german," *Journal of research in Personality*, vol. 41, no. 1, pp. 203--212, 2007.
- [37] M. A. Alvarez-Carmona, A. P. López-Monroy, M. Montes-y Gómez, L. Villasenor-Pineda, and H. Jair-Escalante, ``Inaoe's participation at pan'15: Author profiling task," *Working Notes Papers of the CLEF*, vol. 103, 2015.
- [38] C. E. González-Gallardo, A. Montes, G. Sierra, J. A. Núñez-Juárez, A. J. Salinas-López, and J. Ek, ``Tweets classification using corpus dependent tags, character and pos n-grams." in *CLEF (working notes)*, 2015.
- [39] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, ``Support vector regression machines," *Advances in neural information processing systems*, vol. 9, 1996.
- [40] M. Kocher and J. Savoy, ``Unine at clef 2015: author identification," *Working notes papers of the CLEF*, 2015.
- [41] Wikidata, ``Wikidata," <https://www.wikidata.org/>, 2012.
- [42] M. Wiegmann, B. Stein, and M. Potthast, ``Celebrity profiling," in *Proceedings of the 57th annual meeting of the Association for Computational Linguistics*, 2019, pp. 2611--2618.
- [43] V. Radivchev, A. Nikolov, A. Lambova, L. Cappellato, N. Ferro, D. Losada, and H. Müller, ``Celebrity profiling using tf-idf, logistic regression, and svm." in *CLEF (Working Notes)*, 2019.
- [44] L. G. Moreno-Sandoval, E. Puertas, F. Plaza-Del-Arco, A. Pomares-Quimbaya, J. A. Alvarado-Valencia, and A. Ureña-López, ``Celebrity profiling on twitter using socio-linguistic features notebook for pan at clef 2019," *Working Notes Papers of the CLEF*, 2019.
- [45] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ``Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [46] R. Johnson and T. Zhang, ``Deep pyramid convolutional neural networks for text categorization," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 562--570.
- [47] J. Pizarro, ``Using n-grams to detect bots on twitter." 2019.

- [48] M. Hutson, ``Artificial intelligence faces reproducibility crisis," 2018.
- [49] G. Giannakopoulos, V. Karkaletsis, G. Vouros, and P. Stamatopoulos, ``Summarization system evaluation revisited: N-gram graphs," *ACM Transactions on Speech and Language Processing (TSLP)*, vol. 5, no. 3, pp. 1--39, 2008.
- [50] S. learn Team, ``Scikit-learn," <https://scikit-learn.org/>, 2007.
- [51] P. S. Foundation, ``Python," <https://www.python.org/>, 1991.
- [52] S. Ramírez, ``Fastapi," <https://fastapi.tiangolo.com/>, 2018.
- [53] M. Inc., ``Mongodb," <https://www.mongodb.com/>, 2007.
- [54] P. G. D. Group, ``Postgresql," <https://www.postgresql.org/>, 1996.
- [55] O. Corporation, ``Mysql," <https://www.mysql.com/>, 1995.
- [56] M. Inc., ``Pymongo," <https://pymongo.readthedocs.io/en/stable/>, 2007.
- [57] Vercel, ``Next.js," <https://nextjs.org/>, 2016.
- [58] Facebook, ``React," <https://reactjs.org/>, 2013.
- [59] Microsoft, ``TypeScript," <https://www.typescriptlang.org/>, 2012.
- [60] S. O. Team, ``Stack overflow developer survey 2023," feb 2023. [En línea]. Disponible en: <https://survey.stackoverflow.co/2023/>
- [61] C. McDonnell, ``Zod," <https://zod.dev/>, 2020.
- [62] S. Team, ``Sass," <https://sass-lang.com/>, 2006.
- [63] B. Team, ``Bootstrap," <https://getbootstrap.com/>, 2011.
- [64] M.-U. Team, ``Material-ui," <https://material-ui.com/>, 2014.
- [65] C. U. Team, ``Chakra ui," <https://chakra-ui.com/>, 2019.
- [66] C. Team, ``Chart.js," <https://www.chartjs.org/>, 2013.
- [67] F. Team, ``Framer motion," <https://www.framer.com/motion/>, 2019.
- [68] T. Team, ``Tqdm," <https://tqdm.github.io/>, 2015.
- [69] P. S. Foundation, ``Pickle," <https://docs.python.org/3/library/pickle.html>, 1991.
- [70] N. Team, ``Numpy," <https://numpy.org/>, 2006.

- [71] Atlassian, ``Trello," <https://trello.com/>, 2011.
- [72] Microsoft, ``Microsoft teams," <https://www.microsoft.com/es-es/microsoft-teams/group-chat-software>, 2023.
- [73] G. Team, ``Git," <https://git-scm.com/>, 2005.
- [74] GitLab, ``Gitlab," <https://gitlab.com/>, 2011.
- [75] Bitbucket, ``Bitbucket," <https://bitbucket.org/>, 2008.
- [76] Microsoft, ``Visual studio code," <https://code.visualstudio.com/>, 2015.
- [77] P. Team, ``Prettier," <https://prettier.io/>, 2017.
- [78] B. Team, ``Black," <https://black.readthedocs.io/>, 2018.
- [79] E. Team, ``Eslint," <https://eslint.org/>, 2013.
- [80] E. Amodio, ``Gitlens," <https://gitlens.amod.io/>, 2017.
- [81] Jgraph, ``Draw.io," <https://www.diagrams.net/>, 2012.
- [82] C. Feuersänger, ``Pgfplots," <https://ctan.org/pkg/pgfplots>, 2007.
- [83] Docker, ``Docker," <https://www.docker.com/>, 2013.
- [84] J. Sutherland and K. Schwaber, ``Scrum," <https://www.scrum.org/>, 1995.
- [85] E. Evans, ``Domain-driven design," <https://www.domainlanguage.com/ddd/>, 2003.
- [86] Adobe, ``Adobe xd," <https://www.adobe.com/products/xd.html>, 2016.
- [87] Figma, ``Figma," <https://www.figma.com/>, 2012.
- [88] P. S. Foundation, ``uuid," <https://docs.python.org/3/library/uuid.html>, 1991.
- [89] P. J. Leach, M. Mealling, and R. Salz, ``Rfc 4122 - a universally unique identifier (uuid) urn namespace," <https://tools.ietf.org/html/rfc4122>, 2005.
- [90] C. M. Team, ``Css modules," <https://github.com/css-modules/css-modules>, 2015.
- [91] J. Gruber, ``Markdown," <https://daringfireball.net/projects/markdown/>, 2004.
- [92] F. S. Foundation, ``Gnu general public license," <https://www.gnu.org/licenses/gpl-3.0.en.html>, 2007.
- [93] M. I. of Technology, ``Mit license," <https://opensource.org/licenses/MIT>, 1988.

- [94] A. S. Foundation, ``Apache license," <https://www.apache.org/licenses/LICENSE-2.0>, 2004.
- [95] GitHub, ``Github reaches 100 million developers," <https://github.blog/2023-01-25-100-million-developers-and-counting/>, 2023.
- [96] J. Pybus, M. Coté, and T. Blanke, ``Hacking the social life of big data," *Big Data & Society*, vol. 2, no. 2, p. 2053951715616649, 2015.
- [97] A. Acker and J. R. Brubaker, ``Death, memorialization, and social media: A platform perspective for personal archives," *Archivaria*, pp. 1--23, 2014.
- [98] V. Ruiz Gómez and A. Maria Vallès, ``# cuéntalo: the path between archival activism and the social archive (s)," *Archives and Manuscripts*, vol. 48, no. 3, pp. 271--290, 2020.
- [99] D. Otero, P. Martin-Rodilla, and J. Parapar, ``Building cultural heritage reference collections from social media through pooling strategies: The case of 2020's tensions over race and heritage," *J. Comput. Cult. Herit.*, vol. 15, no. 1, dec 2021. [En línea]. Disponible en: <https://doi.org/10.1145/3477604>
- [100] Statista, ``Distribution of reddit users worldwide as of january 2022, by gender," <https://www.statista.com/statistics/1255182/distribution-of-users-on-reddit-worldwide-gender/>, 2022.
- [101] M. Thelwall and E. Stuart, ``She's reddit: A source of statistically significant gendered interest information?" *Information processing & management*, vol. 56, no. 4, pp. 1543--1558, 2019.
- [102] Statista, ``Percentage of u.s. adults who use reddit as of february 2021, by age group," <https://www.statista.com/statistics/261766/share-of-us-internet-users-who-use-reddit-by-age-group/>, 2021.
- [103] A. Thernstrom and S. Thernstrom, ``Black progress: How far we've come--and how far we have to go," *The Brookings Review*, vol. 16, no. 2, p. 12, 1998.
- [104] BOE, ``Ley 27/2014, de 27 de noviembre, del impuesto sobre sociedades," <https://www.boe.es/boe/dias/2014/11/28/pdfs/BOE-A-2014-12328.pdf>, 2014.
- [105] 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>
- [106] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, ``Language models are few-shot learners," 2020.