



TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN



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

Estudiante: David Rodríguez Bacelar

Dirección: Patricia Martín Rodilla, David Otero Freijeiro

A Coruña, agosto de 2023.

Dedicatoria

Agradecimientos

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

Resumen

Abstract

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

Palabras clave:

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

Keywords:

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

Índice general

1	Introducción	1
1.1	Importancia de las redes sociales	2
1.2	Perfilado de autor	2
2	Estado del arte del perfilado de autor	3
2.1	Conceptos básicos	4
2.1.1	TF-IDF	4
2.1.2	Etiquetas POS	5
2.1.3	N-gramas	5
2.1.4	<i>Bag-of-words</i>	5
2.1.5	Análisis Semántico Latente	6
2.2	Competiciones analizadas	6
2.2.1	PAN <i>Author Profiling</i> 2014	6
2.2.2	PAN Author Profiling 2015	9
2.2.3	Martinc	12
2.2.4	Griwas	12
3	Herramientas, técnicas y lenguajes	13
3.1	<i>Backend</i>	13
3.2	<i>Frontend</i>	14
3.3	Algoritmos de perfilado	14
3.4	Soporte	15
4	Metodología	17
4.1	Roles	17
4.2	Eventos	18

5 Análisis	19
5.1 Requisitos funcionales	19
5.2 Requisitos no funcionales	21
6 Diseño	22
6.1 Arquitectura	22
6.2 Prototipado	24
7 Implementación	30
7.1 <i>Backend</i>	30
7.1.1 <i>Endpoints</i>	30
7.1.2 Clases	33
7.2 <i>Frontend</i>	35
8 Caso de uso: #BLM	38
8.1 Puesta en marcha del sistema	38
8.2 <i>Dataset</i> del movimiento BLM	38
8.3 Subida del <i>dataset</i>	39
8.4 Selección del algoritmo	40
8.5 Proceso de perfilado	41
8.6 Visualización de resultados	42
8.7 Análisis de resultados	43
9 Planificación y costes	47
9.1 Planificación temporal	47
9.2 Recursos y costes	48
9.3 Viabilidad del proyecto	49
10 Conclusiones	50
10.1 Lecciones aprendidas	50
10.2 Trabajo futuro	50
Bibliografía	52

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> del <i>PAN Author Profiling 2014</i>	8
2.2	Comparativa del número de autores por idioma en el <i>corpus</i> del <i>PAN Author Profiling 2015</i>	10
5.1	Diagrama de casos de uso	20
6.1	Diagrama de capas del patrón de diseño DDD. Adaptado de Ł, Ryś [1]	23
6.2	Diagrama C4 de Contenedor de la arquitectura del sistema.	23
6.3	Prototipo de la pantalla de inicio	24
6.4	Prototipo de la selección del algoritmo de perfilado	25
6.5	Prototipo del resumen del perfilado	26
6.6	Prototipo de la pantalla de ejemplos de dataset	27
6.7	Prototipo del dashboard utilizando el algoritmo Martinc	28
6.8	Prototipo del dashboard utilizando el algoritmo Grivas	29
7.1	Estructura del JSON devuelto por el <i>endpoint</i> /profilings/{id}	32
7.2	Diagrama de <i>endpoints</i> del <i>backend</i>	33
7.3	Diagrama de clases de la implementación del <i>backend</i>	35
7.4	Diagrama de clases de la implementación del <i>frontend</i>	37
8.1	Página de inicio de la aplicación	39
8.2	Página de ejemplos de <i>datasets</i>	40
8.3	Página de selección algoritmos	41
8.4	Tooltip de información sobre los algoritmos	41
8.5	Página de resumen del perfilado	42
8.6	Dashboard con los resultados obtenidos por el algoritmo de Martinc [2]	42

8.7 Dashboard con los resultados obtenidos por el algoritmo de Grivas [3] 43

Índice de tablas

2.1	Distribución del número de autores en cada rango de edad en el <i>corpus</i> del <i>PAN Author Profiling 2014</i>	7
2.2	Mejores cuatro clasificados en la competición <i>PAN Author Profiling 2014</i>	9
2.3	Distribución de los autores por característica y lenguaje en el <i>corpus</i> del <i>PAN Author Profiling 2015</i>	10
2.4	Cuatro mejores clasificados en la competición <i>PAN Author Profiling 2015</i>	12
2.5	Clasificación en la competición <i>PAN Celebrity Profiling 2019</i>	12
5.1	Historias de usuario	20
5.2	Requisitos no funcionales	21
8.1	Distribución de género obtenida en ambos algoritmos	44
8.2	Distribución de edad obtenida por Martinc [2]	44
8.3	Distribución de edad obtenida por Grivas [3]	45
8.4	Distribución de ocupación obtenida por Martinc [2]	45
8.5	Distribución de fama obtenida por Martinc [2]	45
9.1	Especificaciones del portátil empleado para el desarrollo del proyecto	48
9.2	Coste del proyecto	49

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 reporte "Digital 2023 Global Overview Report" (We Are Social et al., 2023) [4], a principios del año 2013 existían alrededor de 1.7 millones de usuarios en las redes sociales, mientras que a principios del año 2023, esta cifra aumentó hasta los 4.7 millones, con una variación anual media del 10.8% (se puede ver más información en la Figura 1.1). Así, plataformas como Facebook, Instagram, Twitter o YouTube, cuentan con millones de usuarios activos diariamente, donde se comparte información o se crea contenido de entretenimiento. Además, las redes sociales se han convertido en un lugar donde se debate sobre temas políticos, sociales o económicos, y donde se comparten diversas opiniones y noticias. Este hecho se puede ver, de nuevo, en el reporte mencionado anteriormente, donde se muestra que el 34.2% de los usuarios de redes sociales las utilizan para informarse sobre noticias, el 28.8% para saber cuales son los temas de actualidad y el 23.4% para compartir opiniones y debatir con otros usuarios.

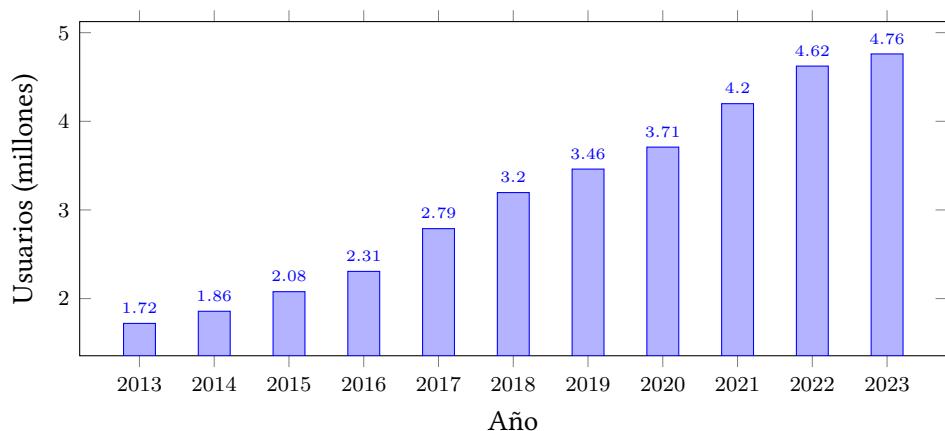


Figura 1.1: Evolución del número de usuarios en redes sociales.

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 la redes sociales ofrece la posibilidad a los ciudadanos de estar informados sobre política, mientras que a las instituciones de poder les permite conocer el estado de la opinión pública (Strömbäck, 2008) [6], pudiendo llegar a influenciar "mucho" o "bastante" la intención de voto (Gallardo-Páuls, 2016) [7]. En segundo lugar, la información generada en las redes sociales también tiene un gran impacto a nivel social. ¿Quiénes son los usuarios más activos? ¿Qué género predomina en las redes sociales? ¿Qué edad tienen los usuarios?. Cabe resaltar también el impacto que tienen las redes sociales a nivel económico, ya que estas plataformas se han convertido en un lugar donde las empresas publicitan sus productos y servicios y a las que destinan gran parte de sus presupuestos en publicidad (Saxena et al., 2013)[8]. Finalmente, destacar el impacto que tienen las redes sociales a nivel de seguridad, ya que estas plataformas se han convertido en un lugar donde se comparte información personal, y donde se pueden cometer delitos como el *cyberbullying* o el *grooming* (Machimbarrena et al., 2018) [9].

1.2 Perfilado de autor

Surge de esta forma la necesidad de analizar toda la información que se genera y proponer así herramientas útiles en todos los niveles vistos en la Sección 1.1. En este contexto, el perfilado de autor en redes sociales se ha convertido en un área de investigación de gran interés en los últimos años, posicionándose como una herramienta de creciente importancia en áreas como la seguridad, el *marketing* o la investigación forense (Rangel et al., 2013) [10].

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

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) [11] se desmarcaban de la tendencia de la época, la cual se basaba en la clasificación de textos en base a su contenido, para centrarse en la clasificación de textos **en base a su estilo**. En este caso, se centraban en la obtención del género del autor mediante el análisis de 920 documentos de carácter formal escritos en inglés con una media de alrededor de 34.300 palabras cada uno, obteniendo una precisión en la clasificación de aproximadamente el 77%.

Así, la demostración de la existencia de rasgos diferenciadores en la escritura que permitían perfilar ciertos aspectos del individuo, especialmente del género, supuso un gran avance en el campo del perfilado de autor y dió pie a la realización de trabajos como (Argamon et al., 2003) [12], (Corney et.al, 2002) [13] o (Otterbacher et al., 2010) [14], así como también permitió el inicio de una clasificación más compleja en base a otras características como la edad, la orientación sexual o la personalidad.

Más tarde, en el año 2011 se celebraría el primer evento organizado por el *PAN (Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection)* [15], un foro de investigación que organiza eventos científicos y tareas anuales relacionadas con el análisis forense de textos digitales y rasgos estilométricos. La primera de estas tareas centrada en el perfilado de autor se celebraría en el año 2013 (Rangel et al., 2013) [10], en la que se pedía a los participantes que obtuvieran, a partir de una serie de *tweets*, la edad y el género de su autor. El ganador de este concurso obtuvo una precisión del 60% en la clasificación de género y del 67% en la clasificación de edad, haciendo uso, la mayor parte de los participantes, de técnicas de aprendizaje supervisado como los Árboles de Decisión (en inglés *Decision Trees*) o las Máquinas de Soporte Vectorial (en inglés *Support Vector Machines*) e incluyendo en sus modelos características basadas en el TF-IDF, n-gramas, etiquetas POS u otras como el número de emoticonos o la frecuencia de signos de puntuación. En los siguientes años se celebrarían nuevas ediciones

de esta tarea (Rangel et al., 2014 [16]; Rangel et al., 2015 [17]; Rangel et al., 2016 [18]...), añadiendo nuevas sub-tareas como el reconocimiento de rasgos personales, la ocupación o los dialectos del lenguaje, así como también alcanzando mejores resultados en la clasificación.

2.1 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 (*Natural Language Processing* o NLP en inglés) y, más concretamente, dentro de la rama de la Estilometría, la cual se encarga del estudio de los rasgos estilísticos de un texto. De esta forma, para las tareas de NLP se hace uso de una serie de algoritmos y técnicas básicas que permiten la extracción de características numéricas y objetivas de un texto. Las más utilizadas y relevantes para la comprensión de este trabajo son las siguientes:

2.1.1 TF-IDF

El TF-IDF (del inglés *Term Frequency-Inverse Document Frequency*) es una medida numérica 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 (*Part-Of-Speech* en inglés) 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* [19].

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. Existen n-gramas de varios tipos en los que se pueden extraer secuencias de caracteres, palabras, etiquetas POS, etc. Los más habituales son los bigramas y los trigramas (*bigrams* y *trigrams* en inglés).

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 grammatical de la que forman parte. De esta 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 desarrollada por Deerwester et al., 1990 [20] 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 Competiciones analizadas

Una vez explicados los conceptos básicos sobre NLP, procederemos en esta sección al análisis de las competiciones más relevantes en el contexto del perfilado automático de autores. Asimismo, dado que el perfilado de la edad era una característica fundamental que debía incluir este trabajo, se seleccionaron aquellas que contaban con dicha tarea, en este caso, las celebradas en los años 2014 (Rangel et al.) [16], 2015 (Rangel et al.) [17] y 2019 (Wiegmann et al.) [21], de las que se extraerá la mayor parte de la información mostrada a lo largo de esta sección.

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, 65-XX.

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.

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* del PAN *Author Profiling 2014*

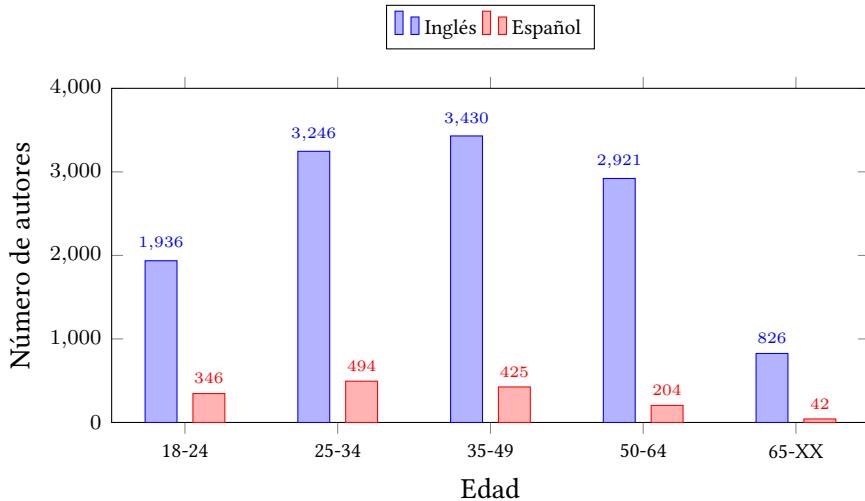


Figura 2.1: Comparativa del número de autores por idioma en el *corpus* del *PAN Author Profiling 2014*

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 60% para ambos idiomas, mientras que para la edad fueron de aproximadamente un 40% para el inglés y de un 50% para el español.

En cuanto a las aproximaciones algorítmicas llevadas a cabo por los equipos mejor clasificados en la competición, podemos destacar lo siguiente:

- **Preprocesado:** En cuanto al preprocesado, tanto Maharjan et al. [22] como Weren et al. [23] realizaron un limpiado de los documentos XML para obtener texto plano, escapando caracteres inválidos en el caso del segundo equipo.
- **Features:** En relación a las características extraídas de los documentos para la clasificación, Maharjan [22], Weren et al. [23] 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. [22], modelizaron el lenguaje con *bag-of-word* y n-gramas, respectivamente, mientras que López-Monroy et al. [24] 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. [24]

hizo uso, más concretamente, de una librería de regresión logística y de SVMs lineales llamada LIBLINEAR (*Large Linear Classification*) [25].

Equipo participante	Precisión							
	Global				Inglés		Español	
	Conjunta	Género	Edad	Género	Edad	Género	Edad	
López-Monroy et al. 2014 [24]	0,2790	0,6310	0,4421	0,6512	0,3950	0,6108	0,4892	
Siang et al.	0,2758	0,6344	0,4348	0,663	0,3909	0,6057	0,4786	
Maharjan et al., 2014 [22]	0,2624	0,5948	0,4411	0,6132	0,3811	0,5764	0,5010	
Weren et al., 2014 [23]	0,2266	0,5866	0,3863	0,6066	0,3690	0,5666	0,4035	

Tabla 2.2: Mejores cuatro 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, 1993) [26]: 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.

Corpus

En lo que respecta al *corpus* proporcionado para la competición, los organizadores recopilaron *tweets* de la red social Twitter de 726 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 reportaron en sus perfiles de Twitter. Para determinar los rasgos de la personalidad, los autores completaron el test BFI-10 (Rammstedt et al., 2007) [27], proporcionando valores para cada rasgo normalizados entre -0.5 y 0.5.

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 tiene un 7% de la representación con respecto al total. 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 similar 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.

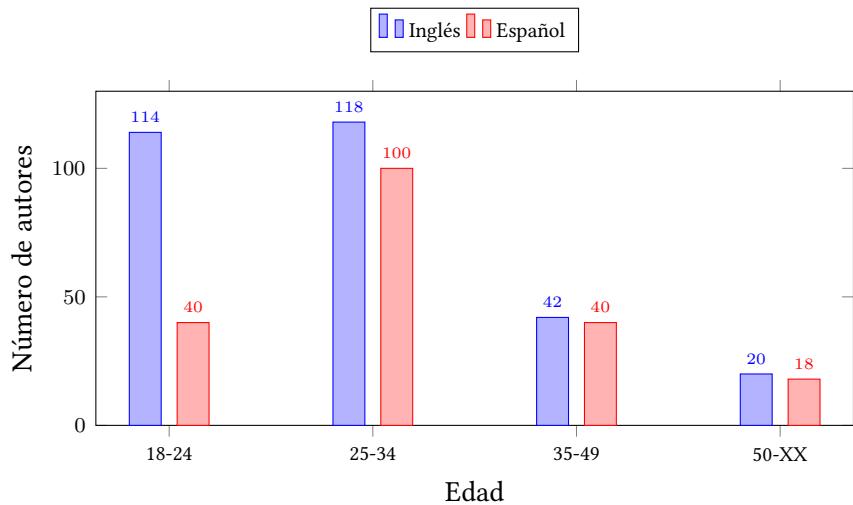


Figura 2.2: Comparativa del número de autores por idioma en el *corpus* del *PAN Author Profiling 2015*

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* del *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%.

Más en concreto, 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% para el español en el caso de Alvarez-Carmona et al. 2015 [28]. Con respecto a la edad, se puede ver como se obtienen unos mejores resultados, de alrededor de un 5%, 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.

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. [29] como Grivas et al. [3], utilizaron las menciones, los *hashtags* y los enlaces como características adicionales para una posterior clasificación.
- **Features:** En cuanto a las características, la mayor parte de los algoritmos combinaron el uso de características basadas en el estilo y en el contenido. Así, en cuanto a contenido, González-Gallardo et al. [29] hizo uso de n-gramas de caracteres mientras que Grivas et al. [3] empleó n-gramas basados en TF-IDF. Además, en el caso de Grivas et al. [3], 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. [28], hicieron 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. [24] en el año 2014.
- **Clasificación:** En lo referente a los métodos utilizados para realizar las predicciones, destaca el uso de la librería LIBLINEAR [25] por parte de Alvarez-Carmona et al. [28] y González-Gallardo et al. [29]. En el caso de Grivas et al. [3], se hizo uso de SVMs para la clasificación del género y la edad junto a SVRs (Drucker et al., 1996) [30] 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 [28]	0,7116	0,8712	0,8168	0,8592	0,8380	0,9659	0,7955	0,7222	0,9375
González-Gallardo et al., 2015 [29]	0,6693	0,8871	0,7545	0,8521	0,7817	0,8977	0,7273	0,8611	0,9375
Grivas et al., 2015 [3]	0,6487	0,9011	0,7199	0,8592	0,7465	0,9432	0,6932	0,8333	0,9688
Kocher et al., 2015 [31]	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*

Equipo participante	Precisión		
	Conjunta	Género	Edad
Radivchev et al. 2019 [32]	0,477	0,928	0,514
Martinc et al., 2019 [2]	0,410	0,906	0,453
Fernquist et al., 2019	0,362	0,774	0,468
Moreno-Sandoval et al., 2019 [33]	0,320	0,862	0,371

Tabla 2.5: Clasificación en la competición *PAN Celebrity Profiling 2019*

2.2.3 Martinc

2.2.4 Grivas

Capítulo 3

Herramientas, técnicas y lenguajes

A lo largo de este capítulo se describirán las herramientas utilizadas para el desarrollo del proyecto y se expondrán las razones de su uso. A su vez, para una mejor estructuración, 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 [34] junto con el *framework* FastAPI [35], el cual permite crear APIs REST de forma sencilla. La decisión de utilizar Python, viene condicionada por el hecho de que los algoritmos de perfilado de autor utilizados, así como también la mayor parte de los algoritmos de aprendizaje automático y procesamiento de lenguaje natural, están ya programados en Python, evitando así crear nuevos *endpoints*, *sockets* o *bindings* para la ejecución de dichos algoritmos. Destacar también que el desarrollo ágil en Python favorecía mucho al trabajo debido a su tipado dinámico, a su ejecución interpretada y a su sintaxis sencilla.

En cuanto a la persistencia de los datos, se optó por MongoDB [36], 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 [37] o MySQL [38], se debe a dos factores principales. Uno de ellos es la fácil implementación de MongoDB en Python haciendo uso de la librería PyMongo [39], dado que permite realizar operaciones sobre las colecciones de forma muy intuitiva. El otro factor tiene que ver con la extensibilidad de la aplicación a largo plazo, es decir, frente a la creación de nuevas funcionalidades, campos o relaciones, MongoDB permitiría integrarlos gracias a la flexibilidad que ofrece sobre los esquemas de datos, algo imposible si se emplease una base de datos relacional.

3.2 *Frontend*

En cuanto al *frontend*, se decidió utilizar NextJS [40] 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 [41], es decir, en la construcción de interfaces dinámicas e interactivas mediante la composición de elementos que pueden tener estado. Además, este *framework* implementa varias mejoras sobre React como por ejemplo el *server-side rendering*, algo que ayuda en gran medida al SEO (del inglés *Search Engine Optimizations*), esto es, a que los motores de búsqueda como Google puedan indexar mejor la página y, por tanto, que esta aparezca en una posición superior en los resultados de búsqueda. Además, también cuenta con optimizaciones para la carga y el renderizado de imágenes o fuentes, entre otras.

Todo ello se desarrolló utilizando TypeScript [42], un lenguaje de programación que añade tipado estático a JavaScript y que está ganando mucha popularidad con respecto a la mantención, compresión y escalabilidad que proporciona a los proyectos en los que se usa (Stack Overflow, 2023) [43]. 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 [44], una librería que permite definir esquemas de validación de datos, incluyendo el tipo específico de cada campo.

En cuanto el estilado de la página, se optó por emplear SASS (del inglés *Syntactically Awesome Style Sheets*) [45], un preprocesador de CSS (del inglés *Cascading Style Sheets*) que añade funcionalidades extra como son el uso de variables, bucles o anidamiento de clases. Además, dado que se estaba desarrollando una aplicación novedosa, se buscó crear un estilo propio haciendo uso de CSS "nativo", desmarcándose así de librerías que proporcionan estilos predefinidos como Bootstrap [46] o componentes ya implementados como Material UI [47] o Chakra UI [48]. Por otra parte, para la creación de gráficos se utilizó la librería ChartJS [49], 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 [50], una librería que permite crear animaciones de una mayor complejidad desde JavaScript/TypeScript.

3.3 Algoritmos de perfilado

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

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

En cuanto a la parte nuclear del aprendizaje automático, se utilizó Scikit-Learn [51], 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 [52], la cual permite mostrar el progreso de entrenamiento o predicción de forma visual, Pickle [53], que permite serializar objetos Python (en nuestro caso los modelos ya entrenados), o NumPy [54], 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 [55], 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 realizar las reuniones necesarias para la planificación de las tareas, se utilizó Microsoft Teams [56], 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 [57] como herramienta de control de versiones junto a GitHub [58], una plataforma que permite almacenar repositorios Git de forma remota, similar a otras como GitLab [59] o BitBucket [60].

En cuanto al entorno de desarrollo o IDE (del inglés *Integrated Development Environment*), se utilizó Visual Studio Code [61], un editor gratuito y de código semi-aberto 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 [62] o Black [63], las cuales permiten formatear el código de forma automática para JavaScript o Python, respectivamente; ESLint [64], una herramienta que permite detectar errores en el código JavaScript/TypeScript; o GitLens [65], 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 [66], 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 [67] 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 [68], 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 [69]. 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, comentadas en la Sección 4.2, que facilitaron en gran medida el desarrollo del proyecto.

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 Eventos

Para agilizar el desarrollo y no interferir en el trabajo diario del equipo por las obligaciones externas de cada uno, tanto las reuniones de planificación como las de revisión y retrospectiva se 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 son, por lo tanto, de una mayor extensión que las *dailies* clásicas y son también empleadas para realizar una revisión del incremento desarrollado en el último *sprint*. 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 emplean para la configuración del entorno de trabajo y para la parte de 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
1	Como usuario quiero poder subir mi propio dataset de textos para su perfilado
2	Como usuario quiero conocer ejemplos del formato de los datasets aceptados
3	Como usuario quiero poder seleccionar el algoritmo de perfilado que se va a utilizar
4	Como usuario quiero poder visualizar los datos obtenidos tras el perfilado
5	Como usuario quiero poder ver una lista detallada de cada persona perfilada
6	Como usuario quiero poder ordenar la lista de personas por cada uno de los campos perfilados
7	Como usuario quiero poder saber como funcionan los diferentes algoritmos de perfilado disponibles
8	Como usuario quiero ver el rendimiento de los algoritmos de perfilado disponibles
9	Como usuario quiero poder reentrenar los algoritmos con diferentes datasets

Tabla 5.1: Historias de usuario

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

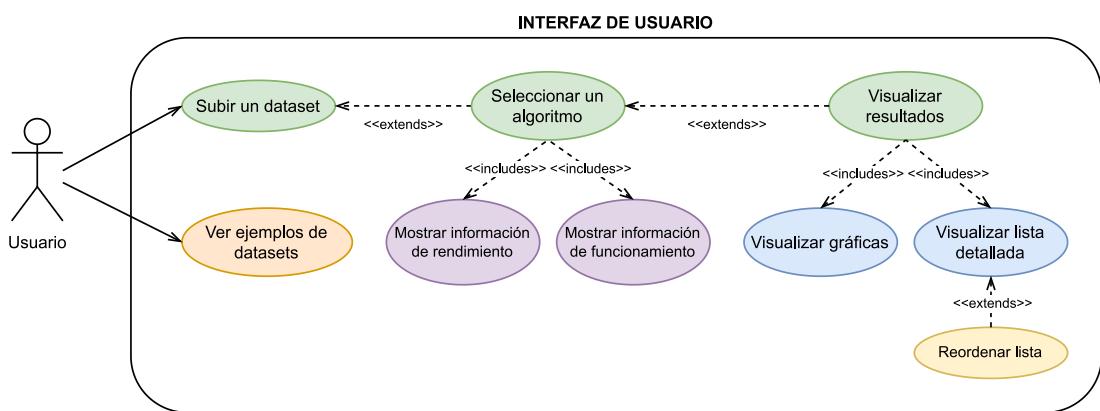


Figura 5.1: Diagrama de casos de uso

5.2 Requisitos no funcionales

Una vez definidas las funcionalidades que debe tener la aplicación, es necesario definir cuales van a ser sus requisitos no funcionales, esto es, aquellos que están relacionados 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.

Requisito	Descripción
<i>Usabilidad</i>	La aplicación debe ser fácil de usar, de forma que cualquier usuario pueda utilizarla sin necesidad de tener conocimientos previos sobre el perfilado de autores.
<i>Escalabilidad</i>	La aplicación debe ser capaz de procesar <i>datasets</i> de cualquier tamaño.
<i>Portabilidad</i>	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 y se mostrarán los prototipos de la interfaz de usuario.

6.1 Arquitectura

Teniendo en cuenta los requisitos definidos en las Secciones 5.1 y 5.2, así como también las limitaciones temporales del proyecto, se ha optado por una arquitectura cliente-servidor. Esta arquitectura se ha elegido por la facilidad en su implementación y por su capacidad de intercomunicación con otros sistemas, especialmente con clientes web. Como se aprecia en la Figura 6.2, el servidor cuenta con un controlador REST capaz de redirigir las peticiones al servicio correspondiente. En este sentido, a pesar de solo contar con un único servicio 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*) [70], el cual permite separar el código en tres capas: la capa de aplicación, la capa de dominio y la capa de infraestructura. La capa de aplicación es la encargada de gestionar la entrada y salida de la aplicación que, en nuestro caso, es el controlador que maneja los *endpoints* REST; la capa de dominio es la que contiene la lógica de negocio y las entidades; y la capa de infraestructura es la responsable de administrar las interacciones internas de la aplicación que, en nuestro caso, se encarga de la comunicación con la base de datos.

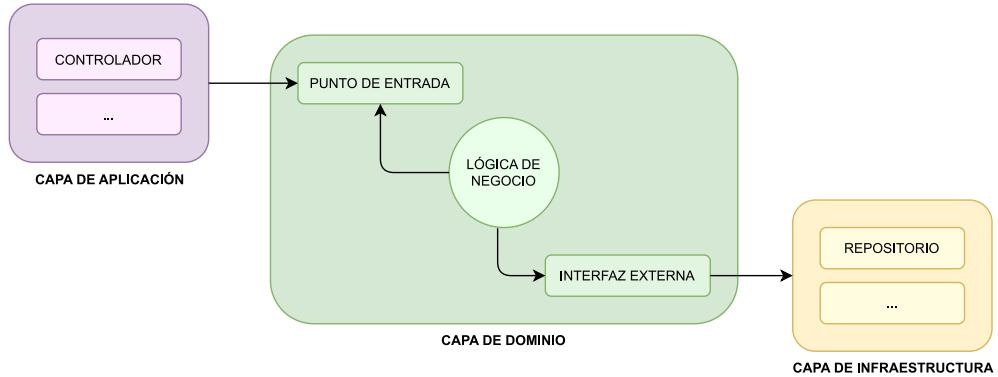


Figura 6.1: Diagrama de capas del patrón de diseño DDD. Adaptado de Ł, Ryś [1]

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

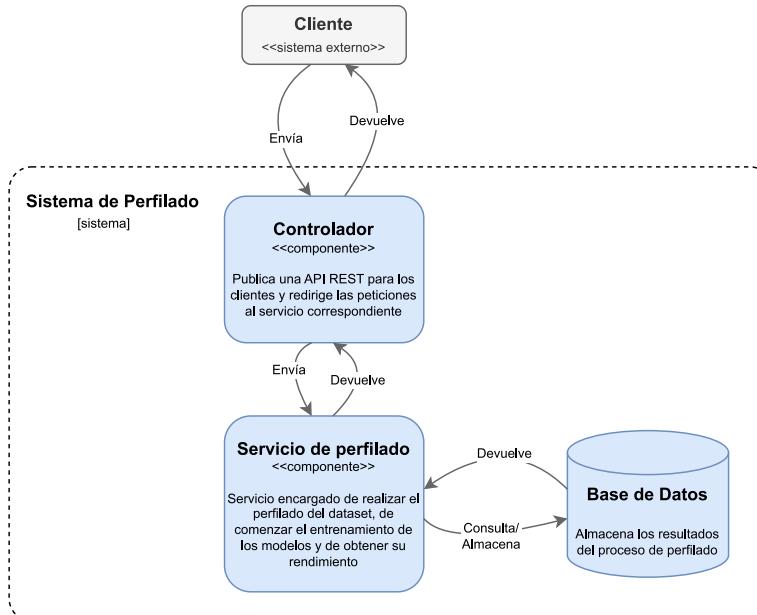


Figura 6.2: Diagrama C4 de Contenedor de la arquitectura del sistema.

6.2 Prototipado

Como se mencionó anteriormente, para el diseño de los prototipos de pantallas, también conocidos como *wireframes*, se utilizó la herramienta Draw.io [66] debido a su sencillez y rapidez en la elaboración con respecto a otros programas más avanzados como pueden ser Adobe XD [71] o Figma [72].

La filosofía de diseño de la interfaz se centró en el minimalismo y la accesibilidad, como se establece en el requisito no funcional de usabilidad 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 permitirá al usuario subir un *dataset*, tanto mediante un elemento *drag and drop* como mediante un botón que abrirá el explorador de archivos del sistema operativo.

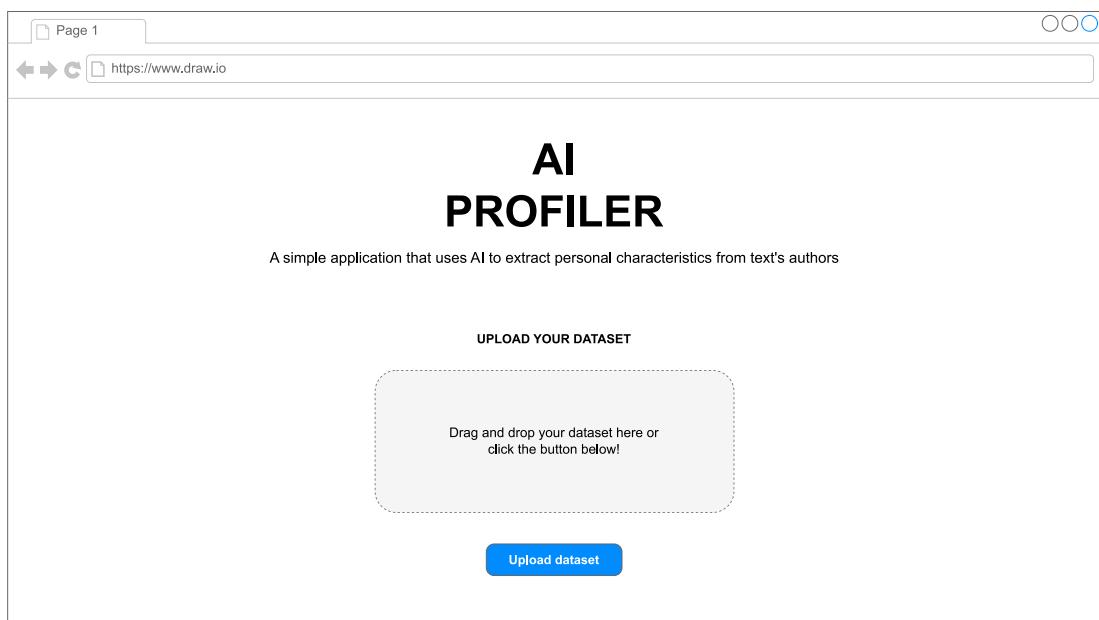


Figura 6.3: Prototipo de la pantalla de inicio

Una vez el usuario haya subido el *dataset*, el campo de subida se sustituirá por la lista de los algoritmos de perfilado disponibles, como se puede ver en la Figura 6.4. De cada algoritmo se mostrará, en una tarjeta, las características que es capaz de extraer del perfilado junto con

su título. Además, ya que uno de los requisitos era el de mostrar información del funcionamiento y del rendimiento de los algoritmos, se decidió crear un *tooltip* para ello. Un *tooltip* es una ventana emergente 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.

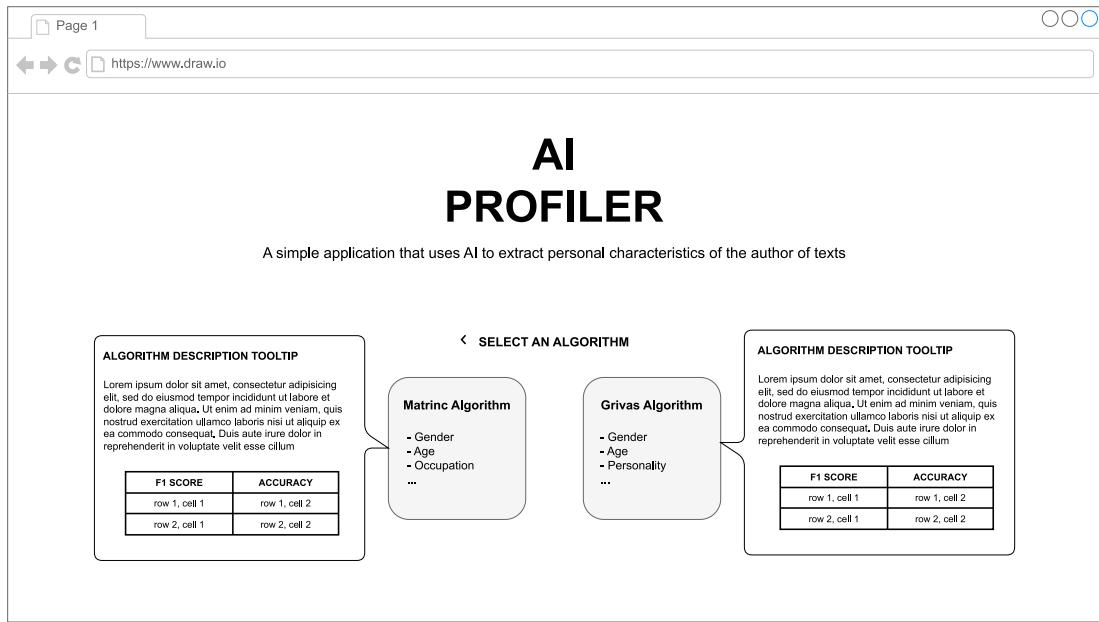


Figura 6.4: Prototipo de la selección del algoritmo de perfilado

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 de perfilado. Además, como se aprecia en la Figura 6.5, una vez comienza el perfilado, se mostrará una barra de progreso o un *spinner* que le indicará al usuario que el proceso está en marcha. De la misma forma que en el paso anterior, si el usuario decide cambiar de algoritmo, puede retroceder haciendo uso de la flecha situada junto al título del paso actual.

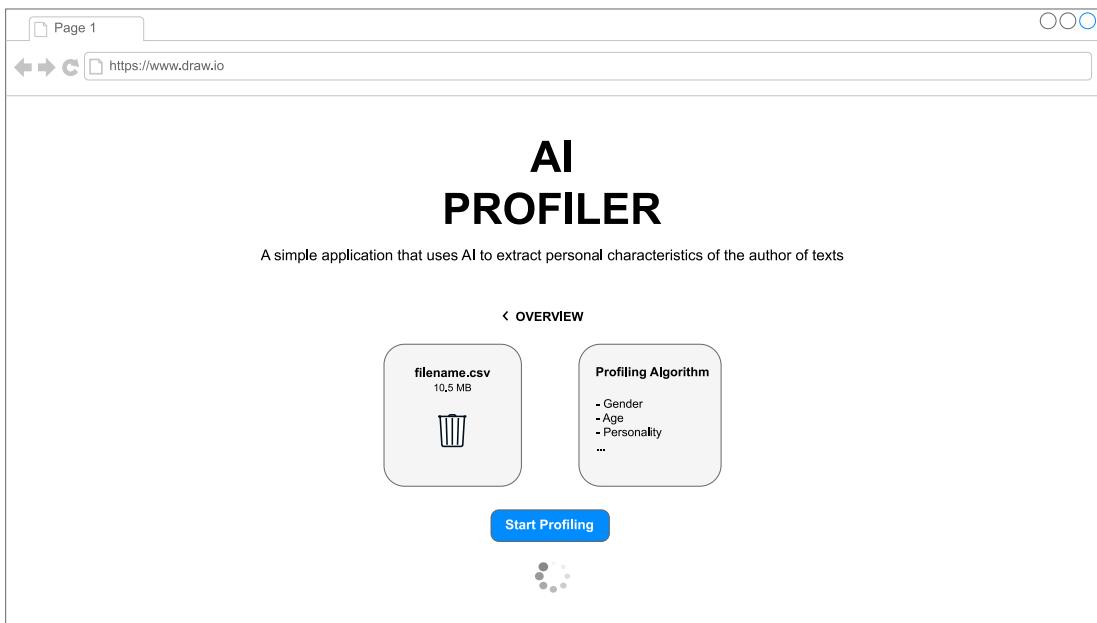


Figura 6.5: Prototipo del resumen del perfilado

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

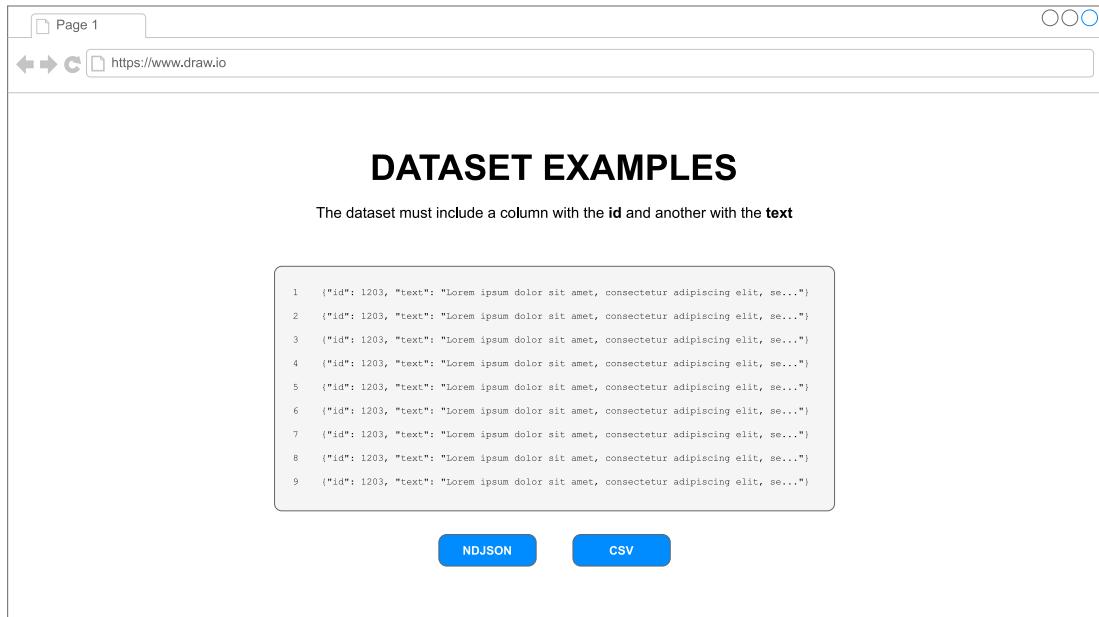


Figura 6.6: Prototipo de la pantalla de ejemplos de dataset

Cuando el perfilado haya finalizado, se mostrará al usuario un *dashboard* con los resultados obtenidos, como se puede ver en las Figuras 6.7 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. Además, la lista permitirá ser ordenada por cada uno de los diferentes campos, según se indica en la historia de usuario con identificador 6 de la Sección 5.1. Para ello, el usuario deberá hacer clic en el nombre de dicho campo, teniendo la opción de, volviendo a clicar, cambiar el sentido de ordenación (ascendente o descendente).
- **Distribución de edad:** Para la distribución de edad, se ha optado por un gráfico de barras sobre otras opciones de representación categóricas como el gráfico circular 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.

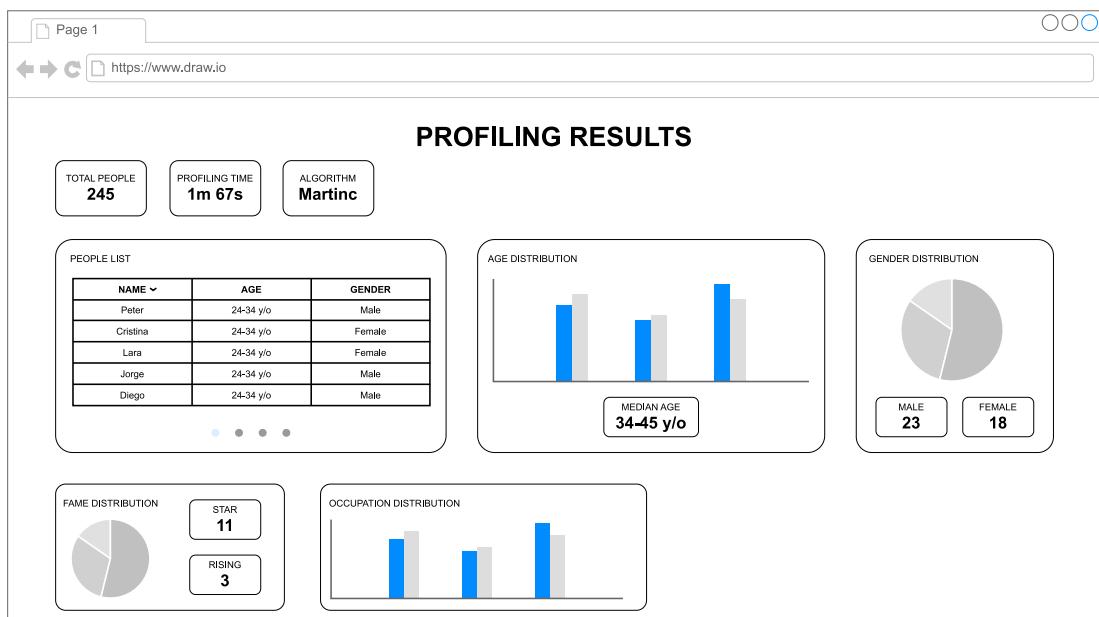


Figura 6.7: Prototipo del dashboard utilizando el algoritmo Martinc

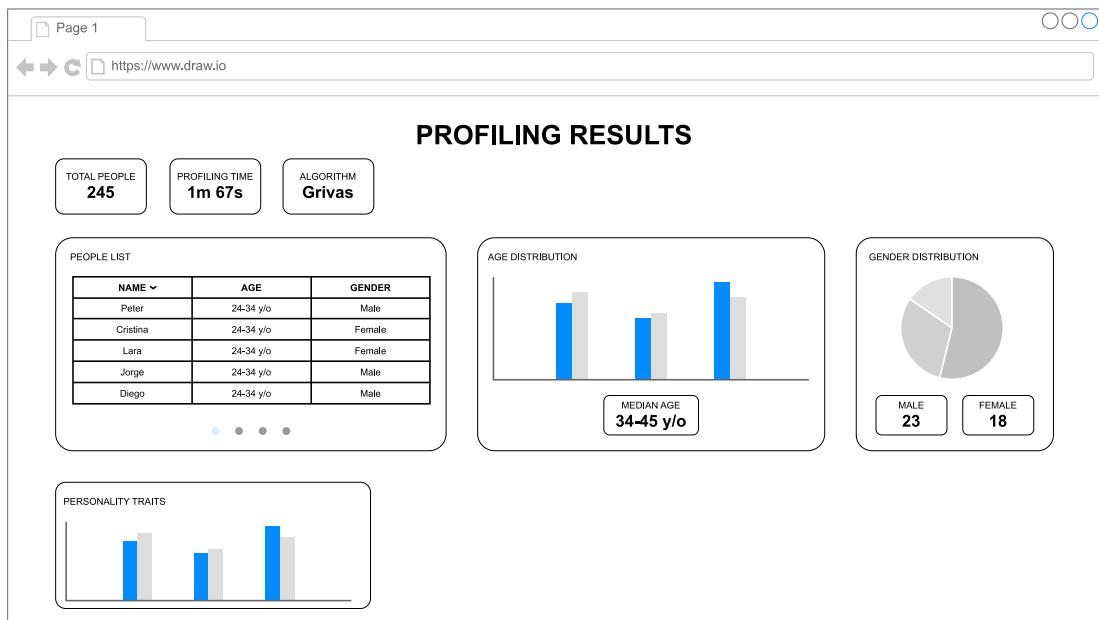


Figura 6.8: Prototipo del dashboard utilizando el algoritmo Grivas

Capítulo 7

Implementación

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

7.1 *Backend*

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

7.1.1 *Endpoints*

Dado que el *backend* seguirá una arquitectura REST, se 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 7.2, el servidor expone un total de cuatro *endpoints*:

- **/predict:** Como indica la historia de usuario con identificador 1 de la Sección 5.1, el usuario debe poder subir su propio dataset de textos para su perfilado. Para ello, se ha implementado este *endpoint* que recibe como parámetros de la URL el nombre del algoritmo de perfilado a utilizar junto al *dataset* utilizado para entrenar el modelo. Además, ya que el método de la petición es POST, 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` [73] de Python (según el estándar RFC 4122 [74]) y que coincide con el identificador del documento almacenado en la base de datos. El hecho de no esperar a que el

perfilado se complete para devolver una respuesta al usuario evita tener problemas con el tiempo de espera de la petición o *timeout* del *socket* TCP, permitiendo así procesar grandes volúmenes de datos y cumplir con el requisito de escalabilidad establecido en la Sección 5.2.

- **/train:** Además, ya que el usuario debe poder reentrenar los modelos con los algoritmos disponibles, se ha implementado este *endpoint* de tipo GET. En este caso, se recibe como parámetros de la URL el nombre del algoritmo de perfilado a utilizar junto al *dataset* de entrenamiento. De la misma forma, se validarán los parámetros y, en caso de que sean correctos, se iniciará una tarea asíncrona en el *backend*.
- **/performance:** Puesto que otra de las historias de usuario nos indica que es necesario conocer el rendimiento que tiene los algoritmos, el *backend* expondrá otro *endpoint* de tipo GET que recibe los mismos parámetros que el anterior. La diferencia es que, en este caso, se devolverá un JSON con los valores de rendimiento del algoritmo en lugar de iniciar una tarea asíncrona.
- **/profilings/{id}:** Para poder obtener los resultados del perfilado creado de forma asíncrona, este *endpoint* de tipo GET permite recuperarlos de la base 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 7.1.

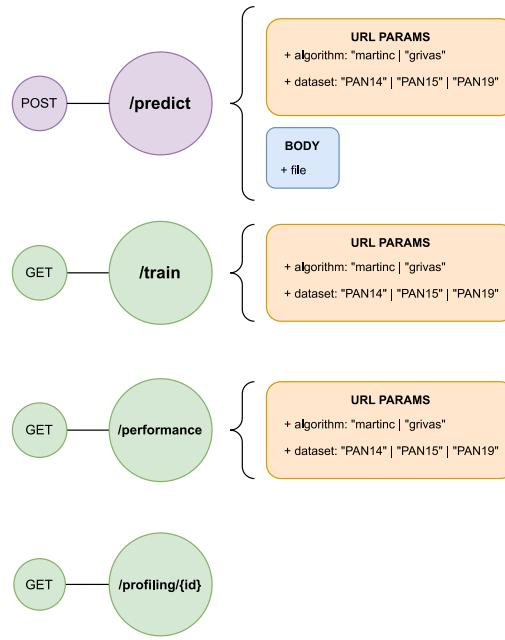
```

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

```

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

Destacar que en todos los casos, tanto el algoritmo de Grivas et al. [3] 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 7.2: Diagrama de *endpoints* del *backend*

7.1.2 Clases

Como se explicó en la Sección 3.1, la estructura del *backend* sigue los principios del patrón de diseño DDD [70], 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 [35].
- 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.

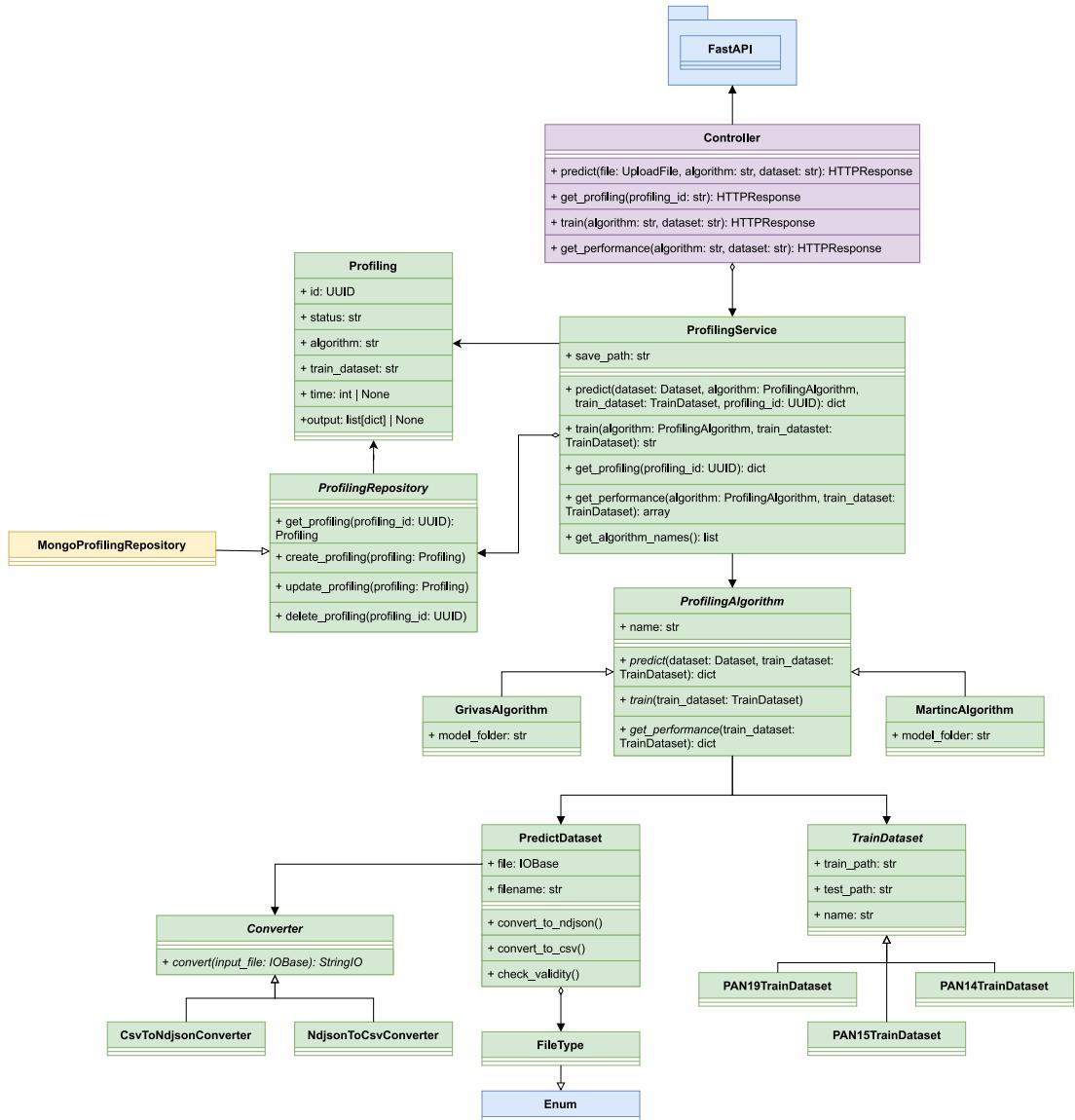
La más importante de ellas es la clase *ProfilingAlgorithm*, una clase abstracta que define la interfaz que deben implementar todos los algoritmos de perfilado incluyendo las tres funciones principales: *predict()*, *train()* y *get_performance()*. En este sentido, es importante mencionar que los algoritmos 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, lo que implicó comprender a muy bajo nivel como estaban implementados dichos algoritmos. Esta adaptación del código fue, en el caso del algoritmo de Grivas et al. [3], bastante compleja, ya que conllevó sustituir librerías obsoletas, reimplementar funciones y actualizar la sintaxis a la nueva versión de Python.

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

También existe una clase llamada *TrainDataset*, que representa el *dataset* que se utiliza para entrenar 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 la conexión con la misma.

Figura 7.3: Diagrama de clases de la implementación del *backend*

7.2 Frontend

En lo que respecta a la implementación del *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 7.4, 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 o *landing (Home)*

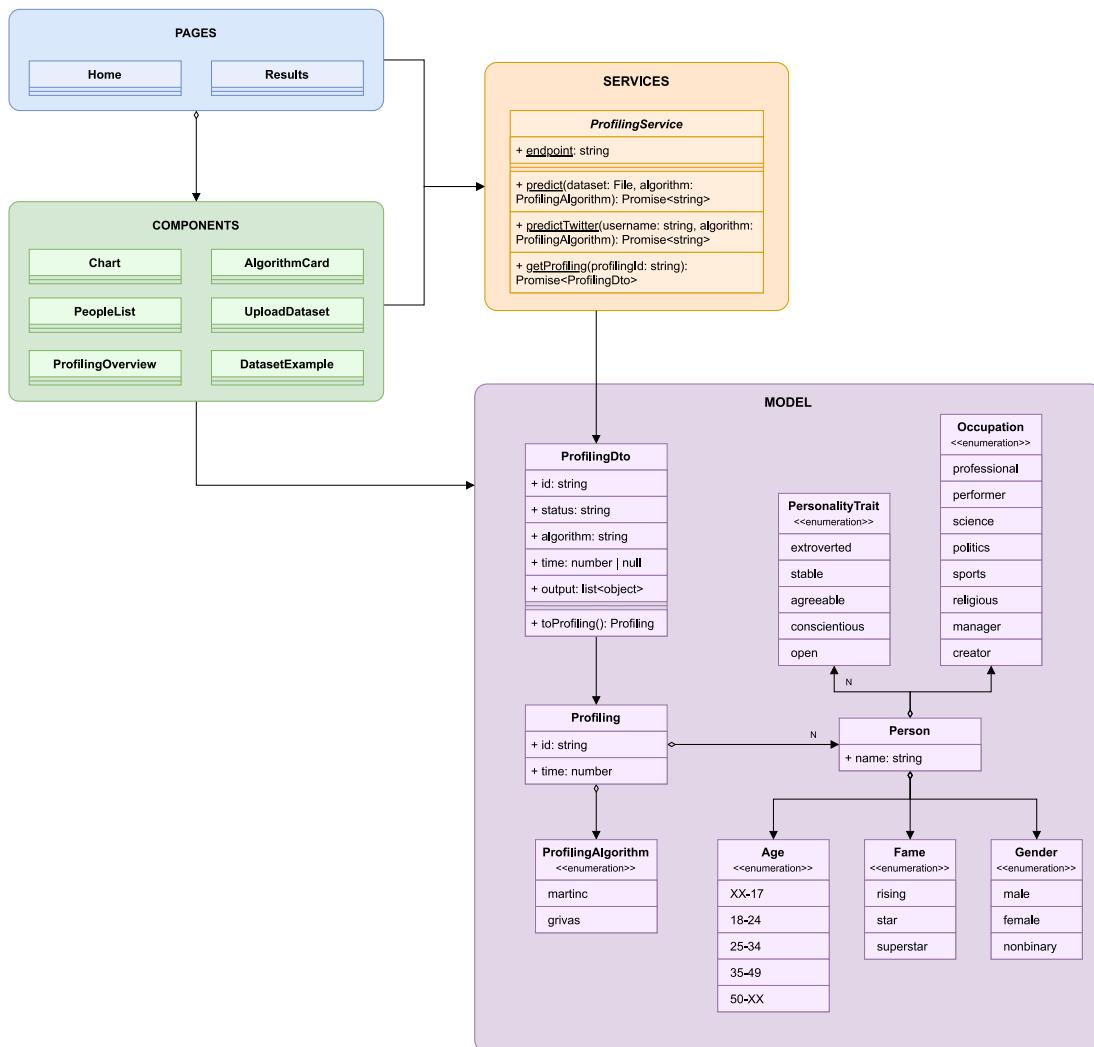
y la página de resultados (*Results*).

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 [75]. 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 explicado en la Sección 5.2, 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 mantenga la lógica de la comunicación y exponga funciones que faciliten su uso. De esta tarea se encarga la clase abstracta *Profiling-Service*, 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 ha optado 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 7.4: Diagrama de clases de la implementación del *frontend*

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 irá comentando paso a paso.

8.1 Puesta en marcha del sistema

Como se mencionó en la Sección 3.4, para poder ejecutar la aplicación simplemente es necesario tener instalado Docker y Docker-Compose. Una vez instalados, es necesario 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 [76]; Acker et al., 2014 [77]; Ruiz et al., 2020 [78]), 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) [79], 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 las 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, a mayores de un campo de búsqueda por usuario de Twitter, 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).

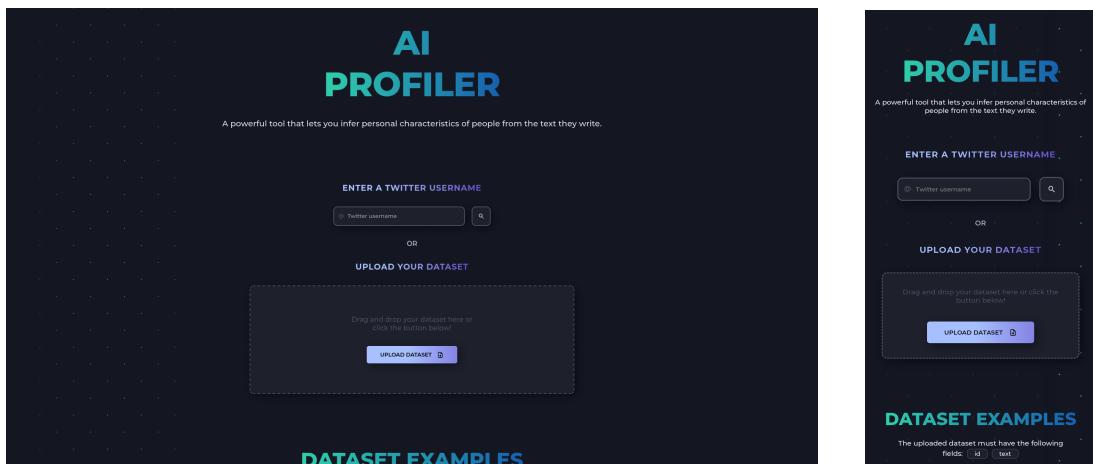


Figura 8.1: Página de inicio de la aplicación

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.

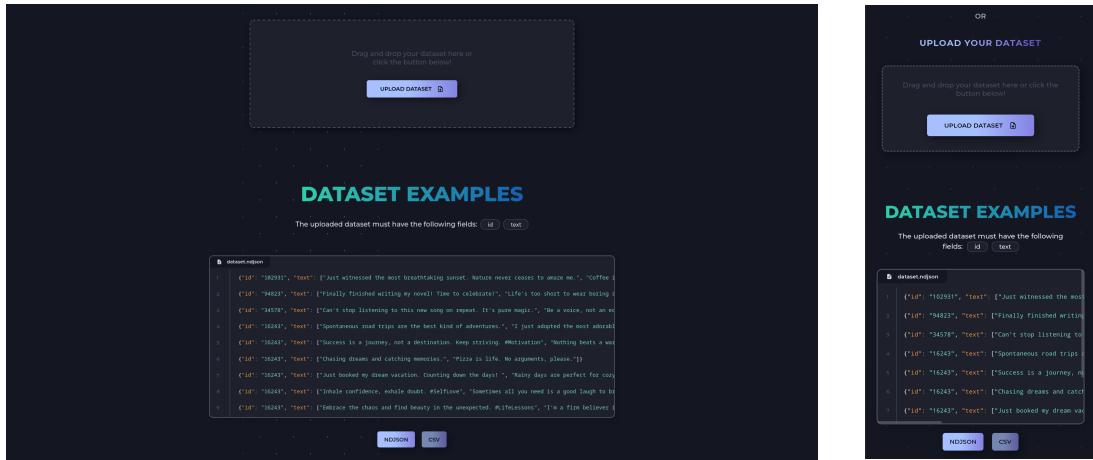


Figura 8.2: Página de ejemplos de *datasets*

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.

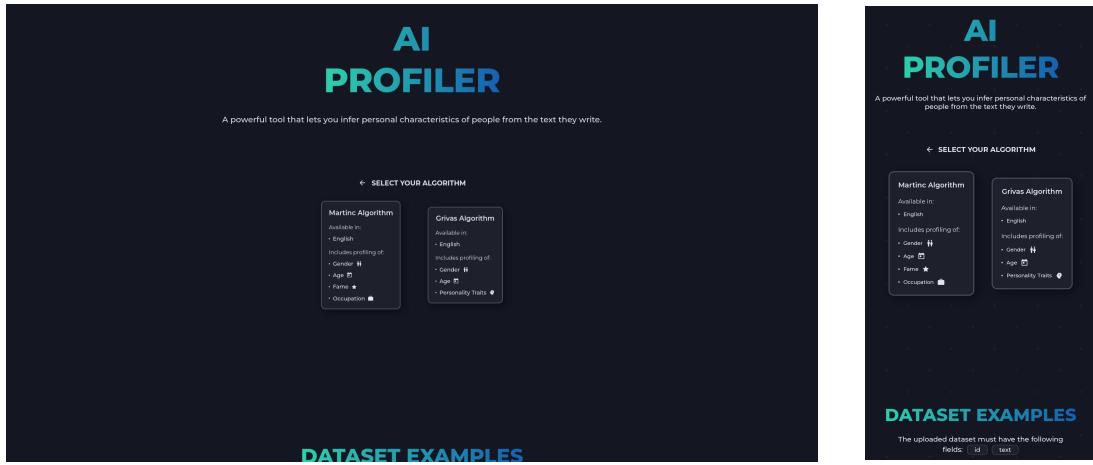


Figura 8.3: Página de selección algoritmos

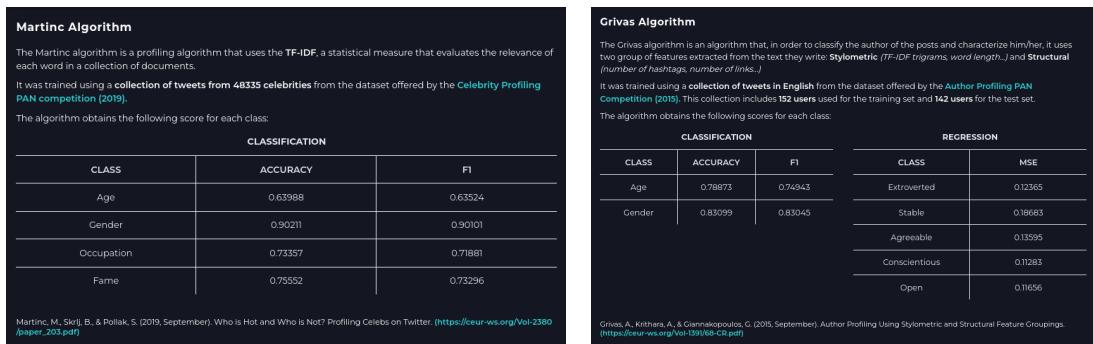


Figura 8.4: Tooltip de información sobre los algoritmos

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.

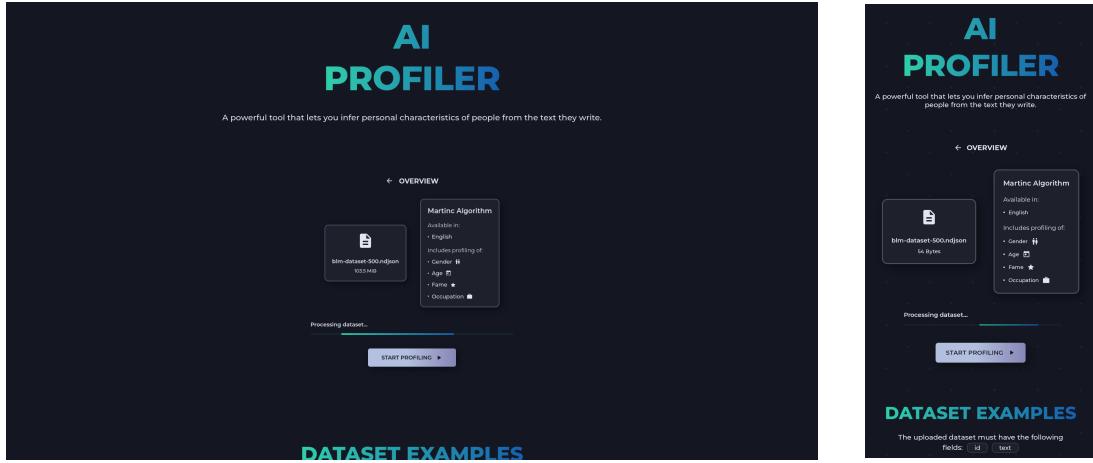


Figura 8.5: Página de resumen del perfilado

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]

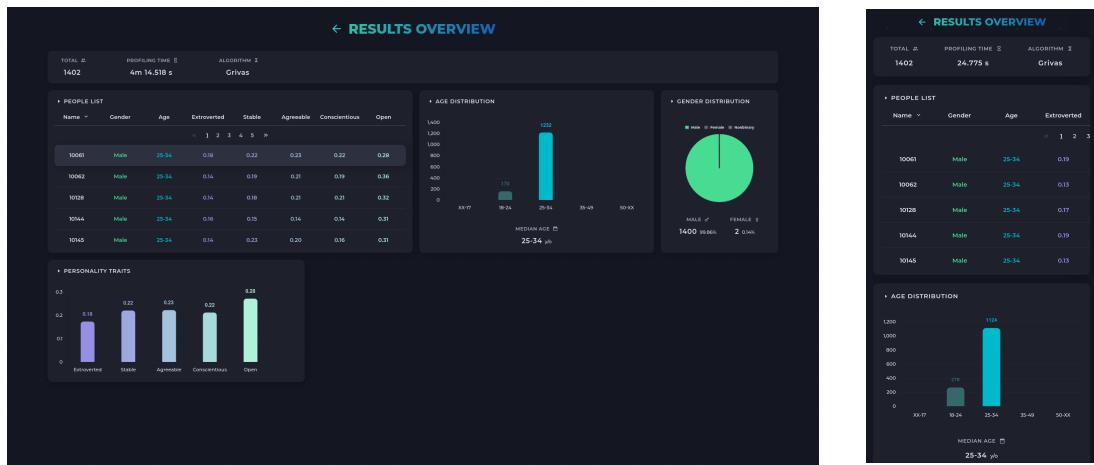


Figura 8.7: Dashboard con los resultados obtenidos por el algoritmo de Grivas [3]

8.7 Análisis de resultados

Para llevar a cabo el análisis de los resultados obtenidos, se abordará el estudio de cada característica por separado y se compranarán 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.

En cuanto a la distribución de género, como se aprecia en la Tabla 8.1, vemos como ambos algoritmos otorgan una gran mayoría al género masculino en cualquiera de los tres *datasets*. En el caso del algoritmo de Grivas [3], observamos como a medida que aumenta el número de posts por usuario el número de usuarios clasificados como femeninos disminuye excesivamente. Esto quizás es debido al limitado *dataset* utilizado para entrenar dicho algoritmo, de apenas 150 personas, por lo que es posible que no sea lo suficientemente representativo. En lo que respecta al algoritmo de Martinc [2], vemos como al aumentar el número de posts por usuario, el número de usuarios clasificados como masculinos se estabiliza alrededor de 1320 (94%), mientras que el número de usuarios clasificados como femeninos se queda en 82 (6%). Esta diferencia tan amplia, sin embargo, no tiene una clara relación con la realidad ya que según Statista, la distribución de género en Reddit es de un 64% para hombres y un 36% para mujeres [80], aunque esta proporción puede variar en función de la comunidad o del tema que se trate.

Posts/usuario	Martinc		Grivas	
	Male	Female	Male	Female
50	1294	108	1385	17
500	1320	82	1400	2
1000	1318	84	1400	2

Tabla 8.1: Distribución de género obtenida en ambos algoritmos

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. En el caso del algoritmo de Martinc [2], observamos como, a medida que aumenta el número de posts por usuario, el número de usuarios pertenecientes a la franja de edad 50-XX aumenta considerablemente, mientras que el número de usuarios pertenecientes a la franja de edad 35-49 disminuye. En este caso, debido al gran número de usuarios utilizados para el entrenamiento, cerca de 20.000, junto al gran rendimiento que demostraba en cuanto a precisión, es posible que, al contar con más posts y, por ende, con más información por usuario, el algoritmo sea capaz de realizar una clasificación más precisa. En cuanto a los datos obtenidos en la encuesta llevada a cabo por Statista a la población estadounidense adulta en 2021, la distribución de edad en Reddit es de un 36% para los usuarios de entre 18 y 29 años, un 22% para los usuarios de entre 30 y 49 años y un 13% para los usuarios de entre 50 y más [81]. Esto puede indicar que, debido a que el número de usuarios perfilados por el algoritmo entre 18 y 34 años es de tan solo un 5%, son los usuarios con más edad los que participan en debates políticos como el caso del BLM. Pasando al algoritmo de Grivas [3] y de la misma forma que sucedía con la distribución de género, los resultados obtenidos parecen estar la mayor parte concentrados en una única clase, en este caso la de 25-34 años, por lo que esto puede deberse, de nuevo, al limitado número de usuarios utilizados para entrenar el algoritmo.

Posts/usuario	Martinc				
	XX-17	18-24	25-34	35-49	50-XX
50	0	9	193	815	385
500	0	4	78	769	551
1000	0	4	63	665	670

Tabla 8.2: Distribución de edad obtenida por Martinc [2]

	Grivas				
Posts/usuario	XX-17	18-24	25-34	35-49	50-XX
50	0	278	1124	0	0
500	0	170	1232	0	0
1000	0	168	1234	0	0

Tabla 8.3: Distribución de edad obtenida por Grivas [3]

Por último, en cuanto a las características perfiladas únicamente por el algoritmo de Martinc [2], vemos como apenas hay diferencias notables en función del número de posts por usuario. Como se puede ver en la Tabla 8.4, la mayoría de los usuarios son clasificados como *Performer*, *Creator* o *Sports*, mientras que el resto de ocupaciones apenas tienen presencia. En cuanto al nivel de fama o popularidad de los usuarios, reflejado en la Tabla 8.5, vemos como la gran mayoría de los usuarios son clasificados como *Star* y el número de usuarios clasificados como *Rising* es prácticamente nulo. Ambas características, al estar específicamente pensadas para el perfilado de celebridades no son, por lo tanto, determinantes para el *dataset* que estamos analizando.

	Martinc								
Posts/usuario	Professional	Performer	Science	Politics	Manager	Creator	Sports	Religious	
50	3	483	43	9	3	390	471	0	
500	2	633	65	3	1	347	351	0	
1000	2	691	73	3	1	321	311	0	

Tabla 8.4: Distribución de ocupación obtenida por Martinc [2]

	Martinc		
Posts/usuario	Rising	Star	Superstar
50	3	1191	208
500	1	1254	147
1000	1	1261	140

Tabla 8.5: Distribución de fama obtenida por Martinc [2]

Como conclusión, podemos decir que, basándonos en las predicciones obtenidas por el algoritmo de Martinc [2], el número de posts por usuario es un factor determinante a la hora de la clasificación. Con todo, se ha obtenido que aproximadamente el 94% de los usuarios son clasificados como masculinos y que una gran parte de los usuarios tienen edades mayores a los 35 años.

Capítulo 9

Planificación y costes

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

9.1 Planificación temporal

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

- **Sprint 1:** Durante el primer sprint, del 15 de abril al 5 de mayo, se realizó la instalación del entorno de desarrollo sobre el que se trabajaría posteriormente y se familiarizó con las 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 la Sección ??.
- **Sprint 3:** Durante el tercer *sprint*, del 27 de mayo al 16 de junio, se comenzó con el desarrollo de la interfaz de usuario, es decir, diseño de *mockups* e implementación de algunos componentes.
- **Sprint 4:** Del 17 de junio al 7 de julio, se continuó con el desarrollo de la interfaz de usuario y se integraron en el *backend* dos de los algoritmos seleccionados.

- **Sprint 5:** Durante el quinto *sprint*, del 8 al 28 de julio, finalizó el desarrollo de la interfaz web y el desarrollo se enfocó en estructurar el código del *backend* así como de agregar la integración con la base de datos. Además, se comenzó la redacción de esta memoria.
- **Sprint 6:** Finalmente, todo el mes de agosto se centró en la elaboración de la memoria y en pulir los últimos detalles para la entrega de la aplicación.

9.2 Recursos y costes

A lo largo del proyecto, se han empleado recursos de tres tipos diferentes: humanos, materiales y software. Como recursos de tipo humano, 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 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) [82], 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 fue de aproximadamente 1.000€, la amortización anual ascendería a 250€. Este número,

sin embargo, tiene que ser proporcional al tiempo de uso del portátil, de aproximadamente 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 [83]. 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

9.3 Viabilidad del proyecto

Capítulo 10

Conclusiones

10.1 Lecciones aprendidas

10.2 Trabajo futuro

Bibliografía

- [1] Łukasz Ryś, ``Organizing layers using hexagonal architecture, ddd, and spring," <https://www.baeldung.com/hexagonal-architecture-ddd-spring>, 2023.
- [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] A. Grivas, A. Krithara, and G. Giannakopoulos, ``Author profiling using stylometric and structural feature groupings." in *CLEF (Working Notes)*, 2015.
- [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. Saxena and U. Khanna, ``Advertising on social network sites: A structural equation modelling approach," *Vision*, vol. 17, no. 1, pp. 17--25, 2013.
- [9] 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.
- [10] 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.

- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] P. Organizers, ``Pan (plagiarism analysis, authorship identification, and near-duplicate detection)," <https://pan.webis.de/>, 2023.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, ``Building a large annotated corpus of english: The penn treebank," *Technical Reports (CIS)*, 1993.
- [20] 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.
- [21] M. Wiegmann, B. Stein, and M. Potthast, ``Overview of the celebrity profiling task at pan 2019," in *CLEF (Working Notes)*, 2019.
- [22] 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.

- [23] 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.
- [24] 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.
- [25] 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.
- [26] L. R. Goldberg, ``The structure of phenotypic personality traits." *American psychologist*, vol. 48, no. 1, p. 26, 1993.
- [27] 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.
- [28] 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.
- [29] 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.
- [30] 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.
- [31] M. Kocher and J. Savoy, ``Unine at clef 2015: author identification," *Working notes papers of the CLEF*, 2015.
- [32] 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.
- [33] 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 sociolinguistic features notebook for pan at clef 2019," *Working Notes Papers of the CLEF*, 2019.
- [34] P. S. Foundation, ``Python," <https://www.python.org/>, 1991.
- [35] S. Ramírez, ``Fastapi," <https://fastapi.tiangolo.com/>, 2018.
- [36] M. Inc., ``Mongodb," <https://www.mongodb.com/>, 2007.

- [37] P. G. D. Group, ``Postgresql," <https://www.postgresql.org/>, 1996.
- [38] O. Corporation, ``Mysql," <https://www.mysql.com/>, 1995.
- [39] M. Inc., ``Pymongo," <https://pymongo.readthedocs.io/en/stable/>, 2007.
- [40] Vercel, ``Next.js," <https://nextjs.org/>, 2016.
- [41] Facebook, ``React," <https://reactjs.org/>, 2013.
- [42] Microsoft, ``Typescript," <https://www.typescriptlang.org/>, 2012.
- [43] S. O. Team, ``Stack overflow developer survey 2023," feb 2023. [En línea]. Disponible en: <https://survey.stackoverflow.co/2023/>
- [44] C. McDonnell, ``Zod," <https://zod.dev/>, 2020.
- [45] S. Team, ``Sass," <https://sass-lang.com/>, 2006.
- [46] B. Team, ``Bootstrap," <https://getbootstrap.com/>, 2011.
- [47] M.-U. Team, ``Material-ui," <https://material-ui.com/>, 2014.
- [48] C. U. Team, ``Chakra ui," <https://chakra-ui.com/>, 2019.
- [49] C. Team, ``Chart.js," <https://www.chartjs.org/>, 2013.
- [50] F. Team, ``Framer motion," <https://www.framer.com/motion/>, 2019.
- [51] S. learn Team, ``Scikit-learn," <https://scikit-learn.org/>, 2007.
- [52] T. Team, ``Tqdm," <https://tqdm.github.io/>, 2015.
- [53] P. S. Foundation, ``Pickle," <https://docs.python.org/3/library/pickle.html>, 1991.
- [54] N. Team, ``Numpy," <https://numpy.org/>, 2006.
- [55] Atlassian, ``Trello," <https://trello.com/>, 2011.
- [56] Microsoft, ``Microsoft teams," <https://www.microsoft.com/es-es/microsoft-teams/group-chat-software>, 2023.
- [57] G. Team, ``Git," <https://git-scm.com/>, 2005.
- [58] GitHub, ``Github," <https://github.com/>, 2008.
- [59] GitLab, ``Gitlab," <https://gitlab.com/>, 2011.

- [60] Bitbucket, ``Bitbucket," <https://bitbucket.org/>, 2008.
- [61] Microsoft, ``Visual studio code," <https://code.visualstudio.com/>, 2015.
- [62] P. Team, ``Prettier," <https://prettier.io/>, 2017.
- [63] B. Team, ``Black," <https://black.readthedocs.io/>, 2018.
- [64] E. Team, ``Eslint," <https://eslint.org/>, 2013.
- [65] E. Amodio, ``Gitlens," <https://gitlens.amod.io/>, 2017.
- [66] Jgraph, ``Draw.io," <https://www.diagrams.net/>, 2012.
- [67] C. Feuersänger, ``Pgfplots," <https://ctan.org/pkg/pgfplots>, 2007.
- [68] Docker, ``Docker," <https://www.docker.com/>, 2013.
- [69] J. Sutherland and K. Schwaber, ``Scrum," <https://www.scrum.org/>, 1995.
- [70] E. Evans, ``Domain-driven design," <https://www.domainlanguage.com/ddd/>, 2003.
- [71] Adobe, ``Adobe xd," <https://www.adobe.com/products/xd.html>, 2016.
- [72] Figma, ``Figma," <https://www.figma.com/>, 2012.
- [73] P. S. Foundation, ``uuid," <https://docs.python.org/3/library/uuid.html>, 1991.
- [74] P. J. Leach, M. Mealling, and R. Salz, ``Rfc 4122 - a universally unique identifier (uuid) urn namespace," <https://tools.ietf.org/html/rfc4122>, 2005.
- [75] C. M. Team, ``Css modules," <https://github.com/css-modules/css-modules>, 2015.
- [76] J. Pybus, M. Coté, and T. Blanke, ``Hacking the social life of big data," *Big Data & Society*, vol. 2, no. 2, p. 2053951715616649, 2015.
- [77] A. Acker and J. R. Brubaker, ``Death, memorialization, and social media: A platform perspective for personal archives," *Archivaria*, pp. 1--23, 2014.
- [78] 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.
- [79] 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>

- [80] 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.
- [81] -----, ``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.
- [82] 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.
- [83] 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>