



Universidad Politécnica
de Madrid



**Escuela Técnica Superior de
Ingenieros Informáticos**

Grado en Ingeniería Informática

Trabajo Fin de Grado

**Desarrollo de una Aplicación Web para el
Análisis y Visualización de Datos de
Películas**

Autor: Sergio Redondo Carracedo

Tutor: Antonio Jesús Díaz Honrubia

Madrid, junio 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Título: Desarrollo de una Aplicación Web para el Análisis y Visualización de
Datos de Películas

Junio 2025

Autor: Sergio Redondo Carracedo

Tutor:

Antonio Jesús Díaz Honrubia

LENGUAJES Y SISTEMAS INFORMÁTICOS E INGENIERÍA DE
SOFTWARE

ETSI Informáticos

Universidad Politécnica de Madrid

Resumen

En este Trabajo de Fin de Grado, se explica detalladamente el proceso seguido para el desarrollo de una aplicación web que permite el análisis y la visualización de datos de películas, obtenidos de diversas fuentes. En ella se engloban películas extraídas de distintas páginas especializadas del sector, mostrando datos detallados de cada una de ellas y pudiéndose filtrar por diversos campos. Además, también permite comparar ciertos datos entre las fuentes de extracción, como las críticas, puntuaciones y número de votos que tiene una película en cada una de ellas, y muestra el ranking de las películas mejor valoradas de cada fuente, todo ello con una interfaz intuitiva para el usuario final.

Para su implementación, el proceso se ha dividido en tres fases claras. En primer lugar, se realizó la extracción exhaustiva de los datos de tres fuentes distintas, utilizando una *API* y técnicas de *Web Scraping*. Una vez obtenidos los datos, se procedió con la fase de almacenamiento en una base de datos relacional, realizando una limpieza previa para conseguir un formato uniforme y evitar duplicados. Por último, tuvo lugar el desarrollo de la aplicación web mediante la herramienta *Django*, incluyendo distintas funciones y representando los datos almacenados mediante una interfaz con la que el usuario puede interactuar.

Además de poder aprender el proceso de desarrollo de una aplicación desde cero y las distintas técnicas a emplear, este proyecto surge con la idea de que los aficionados al cine cuenten con una aplicación sencilla de utilizar mediante la que puedan obtener fácilmente los datos de interés de una película y buscar según los filtros deseados (género, país, actores...), y además puedan comparar y analizar las distintas opiniones y puntuaciones dadas por los usuarios de diversas webs de cine, englobadas en una misma aplicación.

A lo largo de esta memoria se explicarán las distintas herramientas empleadas durante el proyecto y el diseño completo de la aplicación, además de detallar cada una de las etapas del proceso mencionadas previamente e incluir otras secciones de interés.

Abstract

This Final Degree Project explains in detail the process followed for the development of a web application that allows the analysis and visualization of films data, obtained from various sources. It includes films extracted from different specialized websites in the sector, displaying detailed data for each one of them and allowing to filter by various fields. It also allows to compare certain data across sources, such as reviews, ratings and the number of votes a film receives from each one, and shows the ranking of the highest-rated films from each source, all within an intuitive interface for the final user.

For its implementation, the process has been divided into three clear phases. Firstly, data was exhaustively extracted from three different sources, using *APIs* and *Web Scraping* techniques. Once the data was obtained, it was stored in a relational database, after previously cleaning them to achieve a uniform format and avoid duplicates. Finally, the web application was developed using *Django*, including various functions and representing the stored data through an interactive user interface.

In addition to being able to learn the process of developing an application from the beginning and the different techniques to use, this project emerged with the idea of giving film fans an easy-to-use application through which they could easily obtain relevant data about a film and search for them according to the desired filters (genre, country, actors...), and could also compare and analyse the different opinions and ratings given by users from different film websites, included in a single application.

Throughout this report, will be explained the different tools used during the project and the complete design of the application, and also it will be detailed each of the previously mentioned stages of the process and other relevant sections.

Tabla de contenidos

1	Introducción.....	1
1.1	Motivación y necesidad del proyecto.....	1
1.2	Objetivos	2
1.3	Planificación.....	2
1.4	Estructura de la memoria	3
2	Webs cinematográficas existentes.....	5
3	Tecnologías empleadas	8
3.1	Extracción de datos.....	8
3.1.1	API.....	8
3.1.2	Web Scraping.....	9
3.1.2.1	Web Scraping de páginas estáticas	10
3.1.2.2	Web Scraping de páginas dinámicas	11
3.1.3	Python.....	11
3.2	Almacenamiento de datos.....	12
3.2.1	CSV	12
3.2.2	MySQL.....	13
3.2.3	Pandas.....	13
3.3	Desarrollo web	14
3.3.1	HTML.....	14
3.3.2	CSS	15
3.3.3	Django	15
4	Diseño de la aplicación web	17
4.1	Análisis de requisitos	17
4.2	Arquitectura de la aplicación.....	18
4.3	Diseño de la base de datos	20
4.4	Estructura de la aplicación	21
5	Desarrollo de la aplicación web	23
5.1	Extracción de datos.....	23
5.1.1	API.....	23
5.1.2	Web Scraping.....	24
5.2	Almacenamiento de datos.....	26
5.2.1	Limpieza y unificación del formato de datos.....	26
5.2.2	Implementación de la base de datos.....	27
5.3	Desarrollo web de la aplicación	28
5.3.1	Implementación	28
5.3.2	Página principal.....	33
5.3.3	Lista de películas	34
5.3.4	Datos de una película	36

5.3.5	Rankings de una película.....	37
5.3.6	Críticas de una película	37
5.3.7	Buscador	38
5.3.8	Rankings	40
5.4	Evaluación	40
6	Conclusiones y trabajo futuro	42
7	Análisis de Impacto	45
	Bibliografía	46
	Anexo	48
	Informe de Originalidad	48

Índice de figuras

Figura 1. Representación del plan de trabajo mediante Diagrama de Gantt	3
Figura 2. Motivos de las personas para elegir la película de cine la última vez que asistieron [4]	6
Figura 3. Esquema de funcionamiento de una API [8]	8
Figura 4. Esquema de funcionamiento de Web Scraping [10]	9
Figura 5. Ejemplo de archivo CSV	12
Figura 6. Ejemplo de código de una estructura HTML [19]	14
Figura 7. Esquema del diseño MVT [21]	16
Figura 8. Diseño de la arquitectura de la aplicación	19
Figura 9. Diagrama Entidad-Relación de la base de datos	20
Figura 10. Esquema de la estructura de la aplicación	22
Figura 11. Fragmento de respuesta de la API [5]	24
Figura 12. URL para acceder a los créditos de una película en TMDB	24
Figura 13. Creación de la tabla de la relación entre Película y Género	27
Figura 14. Fragmento de "urls.py"	29
Figura 15. Fragmento de "models.py"	30
Figura 16. Fragmento de "views.py"	31
Figura 17. Fragmento de "listado_peliculas.html", correspondiente a la cabecera	32
Figura 18. Fragmento de "listado_peliculas.html", correspondiente al manejo de datos	33
Figura 19. Pantalla principal de la aplicación	34
Figura 20. Carga de imágenes a través de sus URL	34
Figura 21. Página de "Lista de Películas"	35
Figura 22. URL dependiente del "id" de la película seleccionada	36
Figura 23. Página de "Datos" de una película	36
Figura 24. Página de "Rankings" de una película	37
Figura 25. Página de "Críticas" de una película	38
Figura 26. Fragmento de la plantilla "buscador.html", correspondiente al formulario	39
Figura 27. Página de "Buscador"	39
Figura 28. Página de "Rankings"	40

1 Introducción

En este capítulo se hará una breve introducción del Trabajo de Fin de Grado realizado. Aquí se expondrá la motivación que llevó a realizar este trabajo y las necesidades que pretende cubrir, además de enumerar los objetivos y tareas del proyecto, e indicar su planificación y la estructura del resto de la memoria.

1.1 Motivación y necesidad del proyecto

El cine, considerado como “el séptimo arte”, es una de las actividades de ocio más practicadas en todo el mundo, surgiendo distintos grupos y comunidades que comparten su mismo gusto por ver películas. En la actualidad existen numerosas páginas web especializadas en películas (como veremos más adelante), donde además de mostrar los datos técnicos de miles de ellas, usuarios de todo el mundo pueden interactuar valorándolas mediante una puntuación o aportando una crítica u opinión personal. Sin embargo, estas páginas pueden resultar complicadas de navegar para usuarios no familiarizados con ellas o con el mundo del cine, pues al mostrar una gran cantidad de datos sobre cada película, estos pueden ser confusos o difíciles de acceder y visualizar. Además, al contar con miles de títulos registrados, puede resultar compleja la búsqueda de una cinta concreta o que posea unas características en particular. A esto se le suma el hecho de que cada página puede ser frecuentada por usuarios de una comunidad específica o de ciertos países y culturas concretas, lo que podría alterar las valoraciones dadas y su puntuación recibida según la web a la que accedamos.

Es por ello que surge la idea de este proyecto, con la intención de realizar una aplicación web cuya interfaz sea accesible y fácil de comprender para cualquier tipo de usuario que quiera obtener información sobre una película, y a la vez englobe y permita comparar las opiniones y puntuación de los usuarios de distintas webs especializadas, unificando sus datos en una misma aplicación.

Además de la afición por el cine, este trabajo también surge con la intención de conocer cada una de las etapas de desarrollo de una aplicación web desde cero, aprendiendo nuevas tecnologías y herramientas que pueden ser de utilidad para el futuro a la vez que se aplican los conocimientos adquiridos en la universidad. Este proyecto cuenta con las fases diferenciadas de extracción de datos, empleando *API* [1] y técnicas de *Web Scraping* [2]; su posterior almacenamiento en una base de datos relacional; y el desarrollo de la aplicación mediante la herramienta *Django* [3], permitiendo la visualización de la información a través de una interfaz simple e intuitiva.

1.2 Objetivos

El objetivo principal de este trabajo es el desarrollo de una aplicación web para el análisis y visualización de datos de películas, que permita comparar la información obtenida de distintas fuentes, mediante una interfaz intuitiva y fácil de usar para cualquier tipo de usuario. Además de mostrar una gran cantidad de datos de las películas, incluyendo críticas y opiniones procedentes de diversas páginas contrastadas del sector, la aplicación permitirá al usuario buscar películas filtrando por diversos campos, y se podrán comparar y analizar las valoraciones dadas por cada una de las fuentes.

Para conseguir completar este objetivo se ha dividido el proyecto en diversos objetivos específicos, que son necesarios para su correcta realización. Estos son los siguientes:

1. Estudio y familiarización con las tecnologías a emplear.
2. Realización de un análisis de los requisitos del sistema.
3. Diseño de la arquitectura de la aplicación.
4. Extracción de los datos de películas a emplear en la aplicación.
5. Implementación de la aplicación.
6. Realización de pruebas sobre el desarrollo para comprobar su correcto funcionamiento.

1.3 Planificación

Una vez marcados los objetivos a conseguir, se ha realizado una planificación del proyecto mediante distintas tareas que se deben completar y que marcarán el progreso realizado:

1. Estudiar y familiarizarse con las tecnologías a emplear
 - 1.1. Análisis e investigación de posibles herramientas
 - 1.2. Elección de herramientas a utilizar en el proyecto
 - 1.3. Estudio y prácticas con las tecnologías empleadas
2. Realizar un análisis de los requisitos del sistema y de las posibles fuentes de datos
 - 2.1. Investigación de las necesidades de la aplicación
 - 2.2. Definición de los requisitos del sistema
 - 2.3. Análisis de las posibles fuentes de extracción de datos
 - 2.4. Elección de las fuentes de datos
3. Diseñar la arquitectura de la aplicación
 - 3.1. Diseño de la estructura de la aplicación
 - 3.2. Diseño y modelado de la base de datos
 - 3.3. Diseño de la interfaz web

4. Realizar la extracción de los datos a emplear en la aplicación
 - 4.1. Extracción de datos mediante una *API*
 - 4.2. Extracción de datos mediante técnicas de *Web Scraping*
5. Implementar la aplicación
 - 5.1. Implementación de la base de datos relacional diseñada previamente
 - 5.2. Almacenamiento en la base de datos de los datos extraídos
 - 5.3. Desarrollo de la aplicación y de la interfaz web con *Django*
6. Realizar pruebas sobre el desarrollo, comprobando el correcto funcionamiento de la aplicación
7. Escritura de la memoria
8. Preparación de la defensa

A continuación, en la *Figura 1*, se puede observar el plan de trabajo representado mediante un Diagrama de Gantt.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14
Tarea 1														
Tarea 2														
Tarea 3														
Tarea 4														
Tarea 5														
Tarea 6														
Tarea 7														
Tarea 8														

Figura 1. Representación del plan de trabajo mediante Diagrama de Gantt

1.4 Estructura de la memoria

El resto de la memoria se va a estructurar de la siguiente manera:

Capítulo 2. Webs cinematográficas existentes: en este capítulo se explorarán y analizarán aplicaciones o plataformas web ya existentes sobre películas, y se verá su uso e impacto en la sociedad. También se explicarán sus carencias, que hacen necesaria la aplicación propuesta en este trabajo.

Capítulo 3. Tecnologías empleadas: aquí se abordarán las distintas tecnologías y herramientas usadas en cada una de las etapas del proyecto, explicando por qué se han elegido y qué beneficios aportan.

Capítulo 4. Diseño de la aplicación web: en este capítulo se analizará el diseño de la aplicación y de los distintos componentes que la conforman, explicando cada una de las decisiones de diseño tomadas durante el proyecto.

Capítulo 5. Desarrollo de la aplicación web: aquí se detallará cada uno de los pasos seguidos en el proceso para implementar la aplicación, explicando las etapas en las que se ha dividido su desarrollo hasta su finalización.

Capítulo 6. Conclusiones y trabajo futuro: en este capítulo se expondrán las conclusiones obtenidas tras la realización del proyecto y se revisará la consecución de los objetivos que fueron planteados al inicio. Además, se propondrá el posible trabajo futuro de ampliación y mejoras del proyecto.

Capítulo 7. Análisis de impacto: aquí se mencionarán los aportes que el proyecto realizado proporciona a la sociedad y su impacto en ella.

2 Webs cinematográficas existentes

Según un estudio realizado por el Ministerio de Cultura de España [4], la visualización de contenidos audiovisuales, entre los que se encuentran las películas, es la actividad cultural y de ocio más practicada en España, donde casi el 50% de la población fue al cine al menos una vez el año pasado. Esta tendencia es aplicable también al resto del mundo, surgiendo grupos y comunidades de personas que comparten el mismo gusto por el cine.

Es por ello que en la actualidad existen numerosas páginas especializadas en proporcionar información de películas, donde se recopila una gran cantidad de datos de películas que han sido estrenadas desde hace más de 100 años hasta la actualidad, siendo actualizadas constantemente. Además, estas webs no solo muestran datos objetivos, sino que en la gran mayoría de ellas participan los usuarios aportando sus opiniones y valoraciones sobre las películas, y proporcionando una puntuación numérica y críticas personales que podrán leer los demás. A continuación, se destacan tres ejemplos de dichas webs especializadas en el sector, que se han empleado como fuentes de obtención de los datos para realizar la aplicación de este proyecto.

- The Movie Database (TMDb) [5]. Se trata de una gran base de datos de películas (y series) que surgió en 2008, y está construida y actualizada por la propia comunidad. Los usuarios de esta web son quienes proporcionan y verifican cada dato, pudiendo incluir también una puntuación según la opinión personal de cada uno de ellos y escribir una reseña. Además, proporciona una *API* abierta que permite que sus datos puedan ser accesibles por desarrolladores.
- Internet Movie Database (IMDb) [6]. Consiste en una base de datos en línea surgida en 1990 en la que los propios usuarios, denominados “contribuidores”, añaden, editan y validan los datos que aparecen en su web. Cuenta con un apartado de búsqueda avanzada que permite filtrar por distintos campos, y ofrece rankings obtenidos según la valoración dada por sus usuarios. Además de películas, también ofrece información de series de televisión, premios de cine, celebridades y noticias relacionadas con el mundo audiovisual, y ofrece una versión de pago orientada a profesionales de la industria que cuenta con información más completa.
- FilmAffinity [7]. Se trata de una web de recomendación de películas y series nacida en España en 2002. En ella se muestra información sobre las películas en cartelera en España y las plataformas de streaming donde encontrar cada título, pero también actúa como una base de datos que incluye películas de todo el mundo. Los usuarios pueden puntuar las películas, escribir críticas y votar por sus títulos preferidos, pudiendo así encontrar personas con gustos similares.

La existencia de webs que actúan como bases de datos de películas, en las que cualquier persona puede colaborar y opinar sobre ellas, provoca un gran

impacto en el ocio y entretenimiento de la sociedad. Dichas páginas recogen la crítica que recibe una película de parte de un gran número de personas, lo que puede influir en el éxito y la acogida de la cinta. Según el mismo estudio mencionado al principio de este capítulo y como se puede observar en la *Figura 2*, existen distintos motivos por los que una persona elige la película que asiste a ver al cine. Entre ellos destacan el tema de la película y los actores, pero también tienen gran importancia las opiniones de otras personas sobre la película, ya sean de familiares y amigos o de usuarios de redes sociales y críticas profesionales. Es decir, la crítica tiene cierta influencia en la elección sobre el visionado de una película. Por tanto, dichas webs actúan como referencia para la población, donde el usuario puede buscar películas para ver que contengan una serie de características deseadas (como el tema o los actores), y también incluyen valoraciones que pueden afectar a la hora de elegir la visualización de una película.

(Distribución porcentual del total que asistieron en un año)

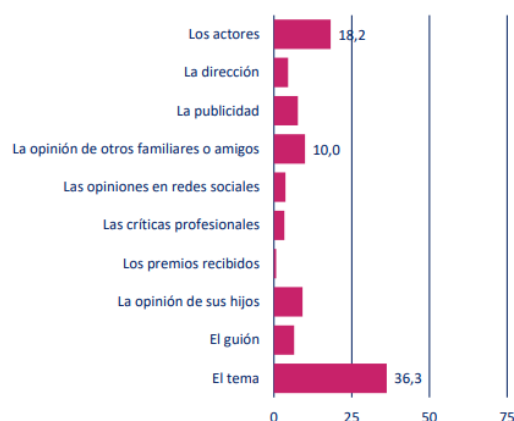


Figura 2. Motivos de las personas para elegir la película de cine la última vez que asistieron [4]

Sin embargo, tanto las páginas mencionadas como el resto de webs especializadas en datos y críticas de películas, tienen ciertas carencias que hacen necesaria la aplicación propuesta en este proyecto. Como se ha indicado previamente, la gran mayoría de datos que se incluyen en ellas provienen del resto de usuarios de la comunidad. Por tanto, estos datos pueden no estar contrastados correctamente o pueden variar de una página a otra, lo que puede ser motivo de confusión, o incluso puede haber datos o películas que estén incluidas en algunas páginas pero no en otras. Es por ello por lo que surge la necesidad de tener una web que englobe y unifique estas películas y sus datos correspondientes. Además, puede ser complicado para algunos usuarios navegar por ellas y llegar a ciertos datos deseados, pues hay una gran cantidad de información y algunos apartados son accedidos a través de otros enlaces contenidos en la propia página, haciendo necesaria una interfaz simple y accesible para todo tipo de usuarios. Por otro lado, según la web que se visite, esta puede ser más frecuentada por personas de una comunidad específica o de ciertos países o culturas concretas, pudiendo variar de esta forma la puntuación recibida o las críticas dadas por los usuarios. Por ejemplo, la web de *FilmAffinity*,

al ser originaria de España, contendrá más películas españolas entre las mejor puntuadas, y estas contarán con un mayor número de críticas y valoraciones que en la web de *IMDb*, la cuál es más internacional y sus películas españolas no tendrán puntuaciones tan exactas ni críticas tan variadas. Es por ello que, al obtener valoraciones de distintas fuentes como se propone en esta aplicación y poder compararlas entre sí, se contaría con información más contrastada y variada disponible para el usuario.

3 Tecnologías empleadas

Durante este capítulo, se mencionarán y detallarán las distintas tecnologías y herramientas empleadas en cada etapa durante la realización del proyecto. También se explicarán los motivos de la elección de dichas tecnologías y qué beneficios aportan, además de por qué no se eligieron otras herramientas candidatas.

3.1 Extracción de datos

La primera etapa a realizar durante la implementación de la aplicación propuesta fue la obtención de los datos que se manejan en ella, siendo un proceso exhaustivo y cuidadoso para extraer correctamente los datos. Una vez elegidas las fuentes de las que obtener los datos, siendo estas tres páginas fiables y especializadas en el sector como son *TheMovieDatabase (TMDB)*, *IMDb* y *FilmAffinity*, se debía analizar qué tecnologías estaban disponibles para extraer los datos de cada una de ellas por separado.

3.1.1 API

Una *API* (Application Programming Interface) es una herramienta basada en un conjunto de protocolos y definiciones que permite conectar dos aplicaciones o componentes entre sí, de modo que pueden comunicarse y compartir datos, información y otras funciones entre ellas. Funciona mediante un modelo cliente-servidor, donde una de las aplicaciones o componentes envía una solicitud a la *API* actuando como cliente, y la otra entidad envía una respuesta actuando como servidor, siendo la *API* un intermediario entre ambas. Además, una *API* puede compartir sólo la información que se desee, de manera controlada. De este modo, simplifica el trabajo tanto a usuarios finales como a desarrolladores, pues a través de ellas pueden incorporar elementos software ya existentes en otras aplicaciones u obtener los datos que necesiten de forma rápida y sencilla.

En la *Figura 3* se puede apreciar una representación del funcionamiento de una *API*.

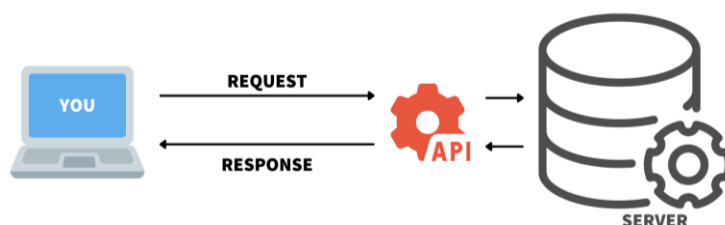


Figura 3. Esquema de funcionamiento de una API [8]

En un primer momento no se consideraba el empleo de una *API* en este proyecto, pero se descubrió que la fuente de datos *TheMovieDatabase* proporciona una *API* propia que facilita la obtención de los datos de su página web, por lo que se decidió emplear esta herramienta para la extracción de dicha fuente.

Esta se trata de una *API* de tipo *RESTful* [9], en la que a través de las llamadas *peticiones HTTP*, en concreto *peticiones GET*, se realiza una solicitud para obtener los datos deseados. En este caso, la *API* devuelve una respuesta en lenguaje *JSON* con la información solicitada. Para poder hacer uso de esta herramienta, fue necesario registrarse en la página web de *TheMovieDatabase* para obtener una clave privada y un token de acceso de lectura, lo que permite realizar las *peticiones* correspondientes.

Mediante la utilización de esta herramienta se consiguieron extraer los datos deseados de forma más rápida y sencilla que mediante el uso de otras técnicas, como las que veremos a continuación, ya que toda la parte interna de búsqueda en su base de datos es realizada por la *API* y solo se deben realizar las *peticiones* correspondientes. Además, dicha web también proporciona la documentación necesaria para conocer el funcionamiento de la *API*, cuya lectura fue imprescindible para poder hacer uso de ella correctamente.

3.1.2 Web Scraping

Esta técnica consiste en el proceso automático de recolección de datos y contenido directamente de un sitio web, permitiendo extraer la información necesaria simulando la interacción humana mediante software. Esto permite obtener y posteriormente almacenar una gran cantidad de datos masivos de forma rápida y automática, en lugar de tener que realizarlo manualmente, como se puede observar en el esquema representado en la *Figura 4*.

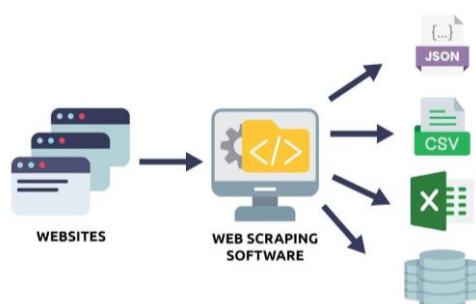


Figura 4. Esquema de funcionamiento de Web Scraping [10]

Generalmente, para extraer la información deseada de un sitio web mediante esta técnica, primero se realizan *peticiones HTTP* a la dirección de la web para obtener su estructura *HTML* [11]. Este es un lenguaje de diseño de páginas web basado en etiquetas, que representan e identifican los distintos elementos que la componen. Una vez obtenido, se recorre el *HTML* y se analiza en busca de los elementos deseados, ya sea a través de las etiquetas que los representan o por distintos atributos que puedan identificarlos correctamente. Por tanto, a pesar

de ser una tecnología de obtención rápida y precisa, se debe hacer un trabajo previo exhaustivo analizando la web y cómo extraer los datos requeridos de ella, buscando cuáles son los elementos que los identifican.

En caso de existir una API a través de la cual se puedan recolectar los datos, es recomendable emplearla en lugar de usar *Web Scraping*, ya que esta última requiere de un proceso más complejo y algunas webs pueden contener mecanismos de seguridad que impidan el uso de esta técnica. Sin embargo, tanto la web de *IMDb* como *FilmAffinity*, no cuentan con API accesibles para los usuarios, por lo que se optó por usar *Web Scraping*. Además, existen dos tipos diferenciados de *Web Scraping* que se explicarán a continuación, ambos usados en este proyecto.

3.1.2.1 Web Scraping de páginas estáticas

Este caso se corresponde con la extracción de datos de webs cuyo contenido es estático, es decir, que se puede obtener directamente desde su *HTML* una vez cargado. Para ello, basta con identificar los elementos buscados a través de sus etiquetas o atributos y recorrer el *HTML* para su obtención.

Existen varias maneras de aplicar esta técnica, pero en el caso de este proyecto se ha realizado empleando las bibliotecas *Requests* [12] y *BeautifulSoup* [13] del lenguaje *Python* [14], al estar ya familiarizado con este lenguaje de programación y ser dos bibliotecas sencillas de aprender a manejar y aplicar en el código.

Requests

Permite realizar peticiones *HTTP* a una página web mediante su *URL* de forma sencilla, pudiendo personalizar algunos aspectos como pueden ser las cabeceras (y así evitar ser bloqueados por la página web), e incluye distintos métodos que facilitan la realización de dichas peticiones.

BeautifulSoup

Esta biblioteca facilita el recorrido y la extracción de datos de archivos *HTML* y *XML*. A partir de estos archivos, crea un “árbol” conteniendo todos sus elementos, simplificando así su obtención, en el que se puede buscar un elemento concreto recorriendo el árbol en busca de una etiqueta o característica que lo identifique.

Ambas bibliotecas se han empleado trabajando con las fuentes *IMDb* y *FilmAffinity*. Se ha utilizado *Requests* para realizar peticiones *HTTP* y obtener el *HTML* completo de la página individual de cada película, y una vez obtenido, se ha extraído cada dato buscado (título, año de estreno, país...) mediante *BeautifulSoup*.

3.1.2.2 Web Scraping de páginas dinámicas

Este caso de *Web Scraping* se aplica cuando el contenido de la web de la que queremos extraer los datos no es estático, es decir, si cogemos el *HTML* sólo obtendremos una parte de la información contenida en la página. Esto se debe a que los datos no son totalmente accesibles en un primer momento, sino que requiere de alguna forma de interacción por parte del usuario con la página web, o lo que se denomina “página dinámica”. Dicha interacción puede ser la necesidad de pulsar un botón para cargar ciertos datos, o incluso simplemente un tiempo de espera hasta cargar la página.

Sin embargo, existen técnicas de *Web Scraping* para solucionar este problema, como *Selenium* [15], que se decidió emplear al ser la más popular en este ámbito y la que cuenta con mayor documentación detallada para aprender a usarla.

Selenium

Se trata de una herramienta que permite simular y automatizar interacciones con el navegador, como si de una persona se tratase, pues incluye métodos que realizan distintas acciones en la web cuya *URL* le suministremos. No contiene un navegador propio, sino que admite ser integrado en distintos navegadores ya existentes mediante un denominado *WebDriver*, y puede incluirse en varios lenguajes de programación. En este caso, se ha empleado en el navegador *Google Chrome* mediante lenguaje *Python*.

En un primer momento no se consideraba el uso de *Web Scraping* dinámico, ya que se trata de una técnica más compleja que el estático. Sin embargo, para obtener las distintas *URL* de las películas de cada fuente, fue imprescindible su uso, ya que sus webs se encuentran paginadas mostrando sólo un reducido número de películas por página y se debía pulsar un botón para mostrar más títulos, como se detallará a lo largo de esta memoria.

3.1.3 Python

Python es un lenguaje de programación de alto nivel, siendo uno de los más usados en todo el mundo. Esto se debe en gran parte a su eficiencia, legibilidad y su facilidad de aprender, con una sintaxis clara y sencilla. Además, es un lenguaje de propósito general, es decir, que no está enfocado en un único ámbito sino que puede ser empleado en una gran variedad de casos, como puede ser el desarrollo web o de aplicaciones, el análisis de datos o la administración de sistemas.

Aparte de su sencillez y variedad de usos, se ha elegido usar este lenguaje para la realización del proyecto ya que cuenta con numerosas bibliotecas y marcos que facilitan la programación y aportan una gran cantidad de funciones potentes. Un ejemplo son las bibliotecas *Requests* y *BeautifulSoup* mencionadas previamente, además de otras que se verán más adelante que también han sido

empleadas. Esto hizo que se descartaran otros lenguajes de programación como *Java*, que también fue pensado inicialmente como lenguaje a emplear.

Python se ha utilizado durante la etapa completa de extracción de datos, tanto en la *API* para realizar las peticiones *HTTP* y extraer los datos de sus respuestas, como durante las técnicas de *Web Scraping* usando sus librerías y la herramienta *Selenium*. Sin embargo, no solo se ha hecho uso de *Python* en esta etapa, sino que ha estado presente a lo largo de todo el proyecto como se verá a lo largo de la memoria.

3.2 Almacenamiento de datos

Una vez extraídos los datos se debía proceder con su almacenamiento, para tener toda la información estructurada y poder manejarlos y mostrarlos posteriormente a través de la aplicación web. Existe una gran variedad de tecnologías dedicadas a almacenar datos, y a continuación se explicarán las elegidas en este proyecto.

3.2.1 CSV

Un archivo CSV [16] (comma-separated values) se refiere a un tipo de formato de documento en el que los distintos datos o valores aparecen separados por comas. Esto permite almacenar grandes cantidades de datos de forma ordenada, a modo de tabla, donde en la primera línea del archivo generalmente se indican los campos a guardar. A continuación, cada línea del archivo representa un elemento o registro, donde los valores de los campos siguen siempre el mismo orden (indicado en la cabecera) y se separan por comas, como se puede apreciar en la *Figura 5*.

```
Nombre, Apellido, Edad, País, Profesión
Juan, García, 38, Argentina, Panadero
Peter, Smith, 55, Estados Unidos, Comercial
Lucía, Petrillo, 22, España, Periodista
```

Figura 5. Ejemplo de archivo CSV

Por lo general, este tipo de archivos se emplea para transferir o intercambiar datos entre distintos programas y aplicaciones, o para exportar datos hacia una hoja de cálculo o una base de datos, debido a su estructura sencilla y compatibilidad con variedad de programas.

En el caso de este proyecto se han empleado archivos CSV para almacenar, en un primer momento, los datos extraídos de las fuentes de datos, y posteriormente transferirlos a una base de datos relacional. Estos se podrían haber almacenado directamente en la base de datos, pero se ha preferido contar con un “paso intermedio” y guardarlos primero en estos archivos. De esta forma, se ha facilitado la limpieza previa y revisión de los datos y ha permitido una

mejor estructuración, simplificando el almacenamiento posterior en la base de datos automatizando el proceso. Además, *Python* cuenta con distintos módulos y funciones que facilitan el guardado y lectura de archivos CSV. Otro beneficio que nos aporta su uso es que, en caso de fallos o borrado de la base de datos, se cuenta con una copia de seguridad al tener también los datos almacenados en estos archivos.

3.2.2 MySQL

La herramienta *MySQL* [17] consiste en un sistema de gestión de bases de datos de tipo relacional, es decir, que almacenan los datos y la información en formato de tablas (con filas y columnas) que se pueden relacionar entre sí. También pueden crearse esquemas, para organizar dichas tablas y sus relaciones. De esta forma permite el almacenamiento de datos masivos de forma clara y ordenada, y está basado en el lenguaje de consulta *SQL* mediante el que se pueden crear, editar y borrar tablas, además de realizar consultas para obtener y manejar datos. Está soportada por *Oracle*, y es uno de los sistemas de bases de datos más usados en todo el mundo estando presente en muchas de las aplicaciones utilizadas en el día a día. Se caracteriza por su buen rendimiento y su facilidad y rapidez de uso, además de por poder aplicarse en una gran variedad de casos de uso (como el desarrollo de aplicaciones) y por su escalabilidad, adaptándose según la cantidad de datos que se quieran almacenar.

En cuanto al proyecto, se tuvieron que tomar distintas decisiones con respecto al almacenamiento de datos. En primer lugar, se decidió emplear una base de datos relacional en lugar de una no relacional, ya que se conocían previamente los tipos de datos a almacenar y estos se debían relacionar entre ellos de forma cuidadosa. Por tanto, era más adecuado almacenar los datos en formato tabla para lograr una mayor coherencia, además de no contar con una excesiva cantidad de datos como para tener que emplear una base de datos no relacional.

Una vez determinado esto, se tenía que elegir qué herramienta concreta emplear. Se decidió utilizar *MySQL* ya que, además de ser adecuado para el desarrollo de aplicaciones y contar con las características mencionadas previamente, es directamente compatible con la tecnología empleada para el desarrollo web (que se mencionará a continuación), al contrario de otros sistemas de gestión de bases de datos como *Microsoft SQL Server*, que también se tenía en mente en un principio.

Por tanto, *MySQL* se ha empleado para almacenar los datos obtenidos de las películas, transferidos desde los archivos CSV, y actuar de esta forma como la base de datos de la aplicación a desarrollar.

3.2.3 Pandas

Pandas [18] se trata de una biblioteca de *Python* que proporciona herramientas potentes de análisis y manipulación de datos. Esta incluye estructuras de datos

que permiten una limpieza y análisis de datos más eficiente, como por ejemplo el denominado *DataFrame*, una estructura tabular de dos dimensiones que facilita el acceso a sus campos y proporciona numerosas funciones para operar y trabajar con ellos.

Esta herramienta se ha empleado en la limpieza y transferencia de los datos extraídos desde los archivos *CSV* hacia la base de datos *MySQL*. A pesar de que su uso no es imprescindible, se ha decidido emplearla ya que cuenta con métodos y funciones que facilitan la lectura de *CSV* y el acceso y limpieza de sus datos. Además, ya se contaba con conocimientos previos sobre esta herramienta, por lo que no supuso tener que emplear tiempo en aprenderla.

3.3 Desarrollo web

Contando con los datos correspondientes extraídos y almacenados, falta la etapa de desarrollo e implementación de la aplicación web en sí, tanto del *frontend* como del *backend*. A continuación, se explicarán las tecnologías empleadas durante este período.

3.3.1 HTML

El *HTML* (HyperText Markup Language) es el lenguaje en el que se basa el desarrollo de páginas web. Se compone de diversas etiquetas que definen la estructura y los distintos componentes y elementos que definen una web, como se puede apreciar en la *Figura 6*.

```
<!DOCTYPE html>
<html>
<head>
  <!-- Metainformación y referencias -->
  <title>Título de la página web</title>
</head>
<body>
  <!-- Contenido visible -->
  <h1>Encabezado</h1>
  <p>Este es un párrafo de texto.</p>
</body>
</html>
```

Figura 6. Ejemplo de código de una estructura HTML [19]

Según la etiqueta empleada para cada elemento, este obtendrá visualmente una forma u otra; por ejemplo, la etiqueta `<ol>` se emplea para definir una lista ordenada, o la etiqueta `<p>` se refiere a un párrafo de texto. Además, mediante este lenguaje también se pueden incluir ciertos elementos como enlaces, títulos, imágenes o botones interactivos, entre otros. Una vez se ha definido el contenido

del *HTML*, el navegador es capaz de interpretarlo y mostrar la página web tal y como la vemos normalmente.

3.3.2 CSS

Por otro lado, *CSS* [20] (Cascading Style Sheets) es un lenguaje empleado para definir el diseño y la apariencia de una página web, y generalmente se usa como complemento al lenguaje *HTML*. De esta forma, se pueden detallar por separado el contenido y estructura de una página web de su aspecto, y contar así con un código más legible y fácil de modificar. Entre los elementos que permite definir el lenguaje *CSS* se encuentran tamaños y tipos de letra, espaciados, márgenes de página, y colores, entre otros.

Se han empleado *HTML* y *CSS* en el proyecto ya que son los lenguajes básicos para la creación del contenido y diseño de una página web. También se podrían haber usado otras tecnologías como *Javascript*, que incluye diversas funciones para mejorar la interacción entre páginas web, o ciertos entornos y bibliotecas que proporcionan plantillas y ayudas en el diseño web, pero no se ha considerado necesario hacer uso de ellas. Sin embargo, sí que fue imprescindible emplear alguna herramienta como *Django* para desarrollar una aplicación dinámica y completa, pudiendo englobar varias páginas *HTML* relacionándolas entre sí y enlazándolas con la base de datos.

3.3.3 Django

Django es un *framework* (marco de trabajo) de *Python* que permite el desarrollo de una aplicación de forma rápida y sencilla. Este se encarga de realizar internamente funciones complejas del desarrollo web, como la autenticación y gestión de usuarios o la conexión y gestión de las bases de datos, pudiéndose centrar únicamente en la creación de la aplicación. Además, dispone de diversos paquetes y módulos reutilizables para evitar tener que repetir código, e incluye otras ventajas como interaccionar con la base de datos por medio de objetos de *Python* en vez de emplear *SQL*, convirtiendo las tablas de datos en clases de código u objetos.

Django funciona con un diseño *MVT* (Model-View-Template), que es una variante del modelo-vista-controlador. Mediante esta arquitectura, se separan las funciones en tres componentes:

- Modelo: proporciona los datos provenientes de la base de datos, empleando la técnica denominada como *ORM* la cual facilita su interacción con el resto del código sin tener que utilizar lenguaje *SQL*.
- Vista: recoge las peticiones *HTTP* realizadas a la aplicación, obtiene datos del modelo y se los proporciona a las plantillas, enviando una respuesta.
- Plantillas (templates): contienen el código *HTML* que define el contenido de las páginas web, determinando la forma de representar los datos.

En la *Figura 7* se puede apreciar un esquema que representa las relaciones entre estos elementos.

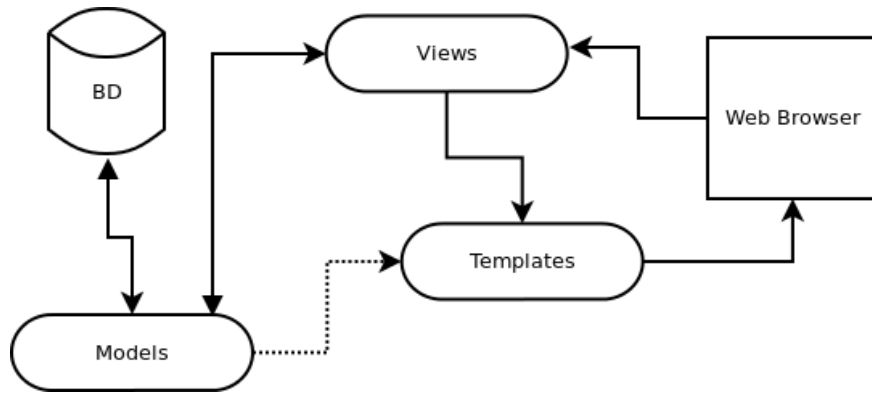


Figura 7. Esquema del diseño MVT [21]

Se ha decidido emplear *Django* para desarrollar esta aplicación debido a su rapidez de desarrollo y facilidad de uso, además de contar con una amplia documentación para aprender a utilizarlo y manejarlo. Otro *framework* similar para desarrollar aplicaciones web en *Python* de forma rápida y sencilla es la herramienta *Flask*. Sin embargo, esta no cuenta con las funciones y componentes que sí que incluye *Django*, por lo que se descartó su uso.

4 Diseño de la aplicación web

Una vez conocidas las tecnologías empleadas a lo largo del proyecto, en este capítulo se detallará el diseño de la aplicación web, siendo esta la fase previa a su implementación. Se abordará el diseño tanto de la arquitectura aplicada como de cada uno de los componentes que conforman la aplicación propuesta, además de indagar en el análisis de requisitos realizado, y se explicará cada una de las decisiones de diseño tomadas a lo largo del proyecto.

4.1 Análisis de requisitos

Tras explorar las distintas plataformas y aplicaciones web que existen actualmente similares a la que se propone, como se mencionó previamente en esta memoria, el primer paso a seguir fue analizar y determinar los requisitos que debía tener la aplicación a desarrollar. Estos se refieren a las distintas funcionalidades y características que debe poseer, lo que sirve como guía previa para realizar su implementación. Se buscaba que la aplicación tuviera las funciones básicas que ya poseen las webs del sector, como la búsqueda de películas y filtrado según sus campos, pero de forma simplificada para el usuario y que además unificara los datos de varias páginas en un mismo sitio. También se pretendía ofrecer funciones nuevas que aportaran valor a la aplicación, como la posibilidad de poder comparar los rankings y puntuaciones de las películas de distintas webs y englobar las críticas de usuarios de diversas fuentes. A continuación, se muestra la lista de requisitos que debía tener la aplicación:

- **RQ1.** La aplicación debe englobar datos de películas de distintas fuentes de datos. Esta debe contar con películas extraídas de varias webs, unificar sus datos globales, y contener datos únicos de cada una de las fuentes como su puntuación, número de votos, críticas y posición en el ranking de su web.
- **RQ2.** La aplicación debe contar con una página principal a través de la que se pueda acceder a las distintas pestañas que contiene.
- **RQ3.** La aplicación debe mostrar un listado completo de las películas disponibles en ella, incluyendo el título, año de estreno e imagen de cada una de ellas en dicha lista.
- **RQ4.** El usuario debe poder acceder a los datos detallados de cada película de forma individual. Estos datos deben incluir: título, año de estreno, imagen, duración, clasificación de edad, sinopsis, géneros, país de origen, empresas productoras, reparto, director, guionistas y compositor de la música/banda sonora de la película.
- **RQ5.** El usuario debe poder acceder a la información de los rankings de cada película de forma individual, mostrando los datos de cada fuente de

extracción. Se debe mostrar, por cada fuente, la posición de la película en el ranking, su puntuación media y el número de votos que ha obtenido por parte de usuarios de la fuente.

- **RQ6.** El usuario debe poder acceder a las críticas de diversos usuarios de cada una de las fuentes de extracción. Se debe mostrar el nombre de la fuente, el nombre del usuario que ha puesto la crítica y su contenido.
- **RQ7.** El usuario debe poder buscar películas filtrando por diversos campos. Se debe permitir la búsqueda por título, año de estreno, género, país, productora, actor, director, guionista, compositor y fuente, pudiendo filtrar por uno o varios de estos campos.
- **RQ8.** La aplicación debe mostrar un listado de las películas que contengan los filtros seleccionados por el usuario. Una vez el usuario ha definido los campos por los que quiere buscar, la aplicación le debe devolver una lista con las películas que incluyen las características deseadas.
- **RQ9.** La aplicación debe mostrar los rankings completos de las 1000 películas mejor valoradas de cada una de las fuentes. En una pestaña aparte, el usuario podrá ver y comparar los rankings de cada fuente, que deben incluir el título y año de las 1000 películas con mayor puntuación de cada una de forma ordenada, además de mostrar el número de votos y la propia puntuación para permitir su comparación.
- **RQ10.** La aplicación debe contar con una interfaz intuitiva y fácil de utilizar para cualquier tipo de usuario, tanto si está familiarizado con páginas web similares como si no.
- **RQ11.** La aplicación debe tener un diseño atractivo visualmente y con colores con un contraste adecuado, para evitar problemas de visibilidad. Además, debe contar con pestañas organizadas y bien estructuradas.

Una vez definida esta lista de requisitos, ya se contaba con una idea más clara de las distintas funcionalidades y aspectos que debía incluir la aplicación final.

4.2 Arquitectura de la aplicación

El siguiente paso a seguir consistía en realizar el diseño general de la aplicación, definiendo principalmente su arquitectura, sus componentes y sus formas de interacción entre ellos.

Dadas las características de la aplicación, siguiendo un modelo *cliente-servidor* donde la parte del *cliente* realiza las peticiones y la parte del *servidor* las procesa y envía una respuesta, se decidió seguir una arquitectura de tres capas. Esta se compone del *frontend*, *backend* y la base de datos.

Frontend: se refiere a la interfaz de usuario y parte visual de la aplicación. Es lo que se muestra en pantalla según lo implementado internamente, y es el componente con el que el usuario interactúa directamente.

Backend: es la parte lógica e interna de la aplicación que el usuario no ve. Se encarga de procesar las peticiones e interacciones del usuario con el *frontend*, realizar operaciones, acceder a los datos y enviar una respuesta.

Base de datos: aquí es donde se almacenan, se gestionan y se mantienen los diversos datos que se incluyen en la aplicación.

Esta arquitectura permite gestionar por separado cada componente de la aplicación, lo que facilita su desarrollo, mantenimiento y detección de errores, y proporciona seguridad y un rendimiento óptimo. Además, todas las partes están relacionados e interactúan entre sí, como se puede apreciar en la *Figura 8*.

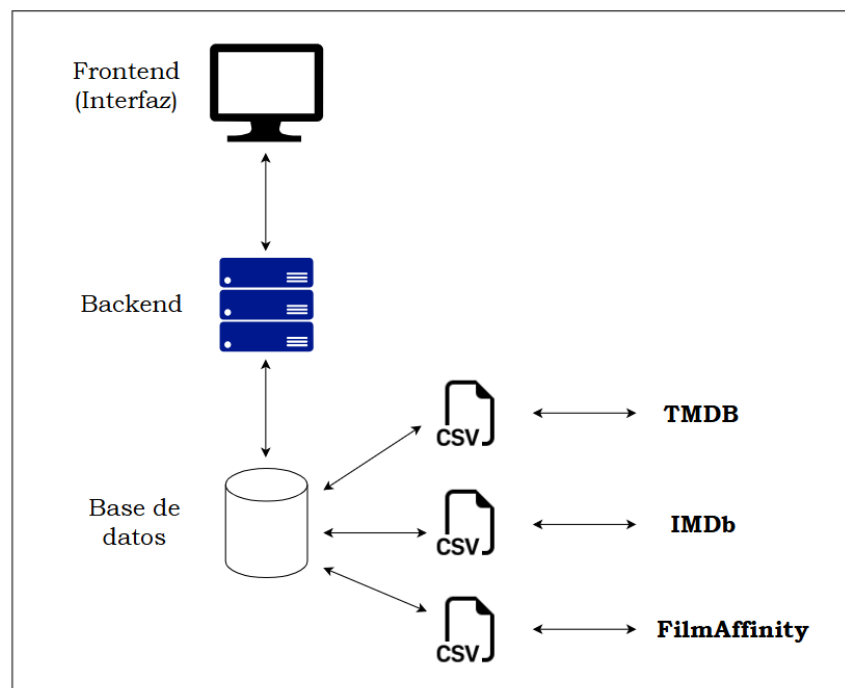


Figura 8. Diseño de la arquitectura de la aplicación

En este esquema se pueden observar las distintas relaciones presentes en la aplicación. El usuario interactúa directamente con el *frontend* a través de su interfaz, y cada acción que realiza (pulsar un botón, realizar una búsqueda, etc.,) se corresponde con una petición que el *backend* se encarga de procesar, obteniendo y transformando los datos provenientes de la base de datos. También se puede apreciar cómo, a través de las etapas de extracción y almacenamiento de los datos de la aplicación, primero se extraen los datos de las tres fuentes, guardándose en distintos archivos CSV, y después estos se transfieren a una única base de datos con la que interactúa la parte de *backend*.

4.3 Diseño de la base de datos

Como ya se explicó anteriormente, se decidió emplear una base de datos relacional, que tiene formato tabular, debido a la naturaleza de los datos y las distintas relaciones que se debían establecer entre las diversas entidades. Antes de crear la base de datos, fue imprescindible realizar un diseño previo para establecer todas las entidades que se debían crear, sus atributos y las relaciones entre ellas. Esta fue una de las partes más importantes del proyecto, pues un buen diseño de la base de datos es clave para manejar correctamente los datos de la aplicación y evitar problemas con su lógica y su estructura. En la *Figura 9* a continuación, se muestra el diagrama Entidad-Relación de la base de datos realizada.

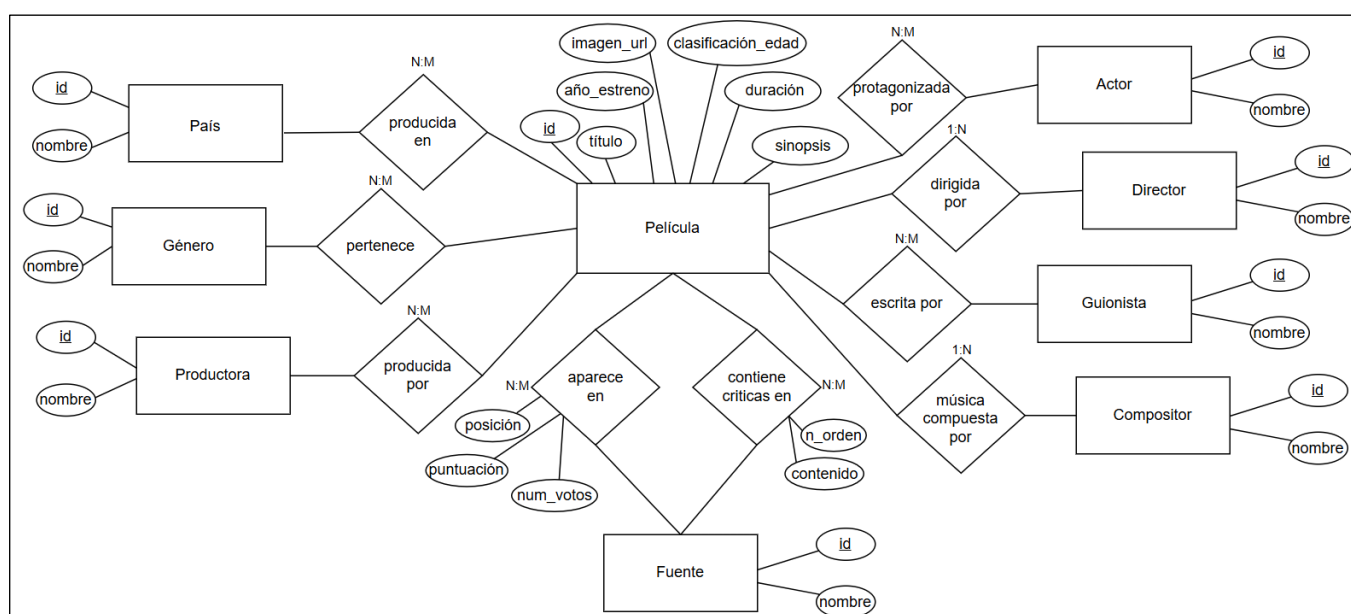


Figura 9. Diagrama Entidad-Relación de la base de datos

Como se puede observar, el núcleo principal de la base de datos es la entidad “Película”, la cual se relaciona con el resto de las entidades. En ella se almacenan algunas de las características de la película y se asigna un “id” que la identifica, y que se emplea para establecer las relaciones con el resto de tablas. Hay que reseñar que no se podía establecer el título de la película como su identificador único, ya que pueden existir películas con un mismo título (como nuevas versiones de películas antiguas), y en el resto de entidades también se empleó un “id” como su identificador.

Se puede apreciar como la mayoría de las relaciones tienen cardinalidad “N:M” pues, por ejemplo, en el caso de los géneros, una película puede pertenecer a varios de ellos, y un género puede englobar varias películas. Sin embargo, hay dos relaciones 1:N, correspondientes a la dirección de la película y la composición de la música o banda sonora. En estos casos se ha considerado que una película solamente puede ser dirigida por un director, y su música compuesta por un solo compositor. También hay que destacar las dos relaciones que tienen lugar entre “Película” y “Fuente”. En la relación “aparece en”, se

determina en qué fuentes de las empleadas en el proyecto aparece cada película, guardando como atributos de la relación la posición de la película en el ranking de dicha fuente, su puntuación y el número de votos obtenido. Por otro lado, en la relación “contiene críticas en”, se almacenan las críticas de los usuarios hacia una película en cada fuente en la que aparece, mediante los atributos “n_orden” y “contenido”. En “n_orden” se almacena el número de crítica por película y fuente, pues puede haber varias críticas de una película en una misma fuente, y en “contenido” se guarda el texto de la crítica junto con su autor.

4.4 Estructura de la aplicación

Previo a comenzar su implementación, se debía realizar un diseño de la estructura de la aplicación en cuanto a su interfaz. Ya se realizó el diseño de la arquitectura general de la aplicación, con los componentes y sus relaciones, pero se debía determinar el número de pestañas con las que contaría la aplicación y las funcionalidades que proporcionaría, atendiendo a los requisitos ya establecidos y a las características deseadas.

Se decidió dividir la aplicación en tres pestañas generales, todas ellas precedidas de una página principal a modo de portada mediante la que se podría acceder a cualquiera de las pestañas. La primera de ellas proporcionaría el listado total de las películas con las que cuenta la aplicación, pudiendo acceder a características de cada película individual como son sus datos detallados, sus rankings según cada fuente de extracción en la que aparece, y las críticas de usuarios de cada web. Otra pestaña permitiría al usuario buscar películas, pudiendo introducir diversos filtros y obteniendo una lista de películas que cumplan con las características deseadas. Por último, la tercera pestaña mostraría el ranking completo de las tres fuentes de extracción, indicando las puntuaciones y número de votos de sus películas para poder compararlas entre ellas. Además, se planteó la introducción de una cabecera fija en todas las pestañas, que permitiera navegar por cualquiera de ellas o volver a la página anterior.

A continuación, en la *Figura 10*, se muestra una representación del diseño planteado.

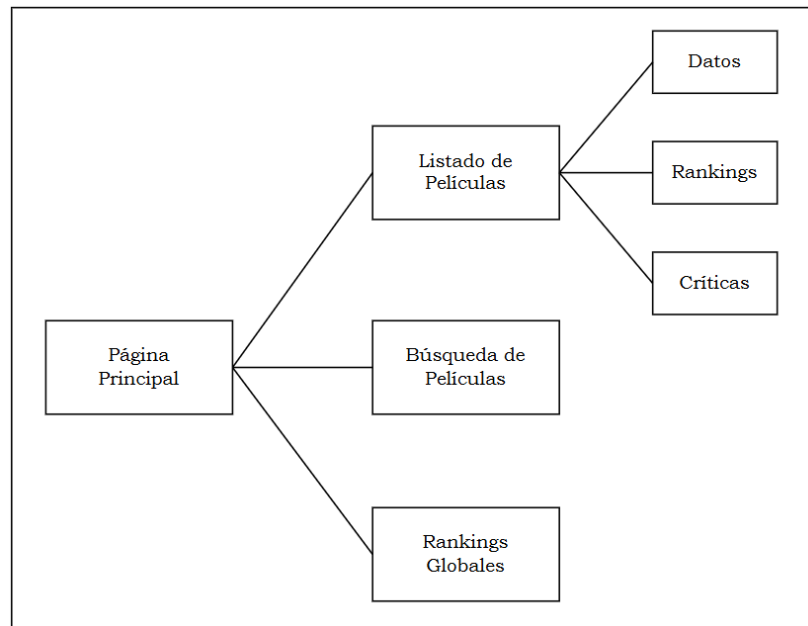


Figura 10. Esquema de la estructura de la aplicación

5 Desarrollo de la aplicación web

Tras establecer el diseño tanto de la arquitectura de la aplicación como de los distintos elementos que la componen, en este capítulo se explicará detalladamente cada uno de los pasos seguidos para lograr el desarrollo de la aplicación web que se propone. Se explorará cada una de las etapas en las que se divide su implementación: extracción de datos, almacenamiento y desarrollo web.

5.1 Extracción de datos

Tras obtener los requisitos de la aplicación y decidir el diseño de su arquitectura, ya se podía comenzar con el proceso de implementación en sí. El primer paso o etapa a realizar consistía en la extracción de los datos que conforman la aplicación, con los que se trabaja a lo largo del proyecto.

Previamente se realizó un análisis de las posibles fuentes de obtención de datos, explorando las distintas aplicaciones y páginas web especializadas en el sector del cine, como ya se explicó en el *Capítulo 2* de esta memoria. Se buscaba que estas webs tuvieran accesibles los datos principales de cada película, todos o gran parte de los que se habían descrito en la parte de requisitos. Además, estas debían contar con una puntuación numérica y con críticas realizadas por otros usuarios o expertos de cine, para poder compararlas posteriormente. Finalmente se optó por las páginas ya mencionadas: *TheMovieDatabase*, *IMDb* y *FilmAffinity*, ya que contenían todos los datos y características deseadas. De esta forma se descartaron otras webs destacadas, como la página *RottenTomatoes* [22], que no disponía de toda la información deseada de las películas y no mostraba unas puntuaciones numéricas claras, o *Sensacine* [23], que al contener menos usuarios no contaba con un sistema de puntuaciones adecuado y contrastado.

Tras haber decidido las fuentes, se debía analizar la forma de extraer los datos de cada una de ellas. Hay que reseñar que, debido a la gran cantidad de películas existentes en cada fuente (cerca de un millón), se decidió obtener una muestra de películas más pequeña, correspondiente a las 1000 películas con mayor puntuación de cada una de las webs accedidas.

5.1.1 API

En el caso de *TheMovieDatabase*, esta contaba con una *API* pública disponible para usuarios registrados, por lo que se decidió usar esta herramienta debido a las facilidades que suponía su uso. Para poder emplearla se tuvo que realizar un registro en su web, obteniendo una clave privada y un token de acceso de lectura que permite realizar peticiones. Una vez recibidos, se escribió una clase de código en lenguaje *Python* y empleando la biblioteca *Requests*. En ella se accedía a la *URL* que contiene las películas mejor valoradas de la web, y se

realizaba una petición *GET* incluyendo una cabecera con el token de acceso de lectura obtenido. La *API* devolvía una respuesta en lenguaje *JSON*, que incluía ciertos datos de cada película. A continuación, en la *Figura 11*, se puede ver un fragmento de la respuesta devuelta por la *API*.

```
"id": 238,  
"original_language": "en",  
"original_title": "The Godfather",  
"overview": "Don Vito Corleone, conocido dentro de",  
"popularity": 33.9816,  
"poster_path": "/5HLLUsmsv60cZVTzVns9ICZD6zU.jpg",  
"release_date": "1972-03-14",  
"title": "El padrino",
```

Figura 11. Fragmento de respuesta de la API [5]

Sin embargo, esta primera respuesta *JSON* no incluía todos los datos que se querían extraer, por lo que a través del “id” obtenido de cada película se debían realizar otras peticiones *GET* a diversas *URL* más concretas, para que la *API* mostrara todos los datos requeridos.

Para ello, se establecía cada *URL* a la que debíamos acceder incluyendo en ella el “id” único de cada película, y se realizaban peticiones a la *API*. Por ejemplo, en la *Figura 12* se puede observar la *URL* accedida para obtener los datos de los créditos de una película (reparto, director, guionista...).

```
url_credits = f"https://api.themoviedb.org/3/movie/{id_pelicula}/credits?language=es-ES"
```

Figura 12. URL para acceder a los créditos de una película en TMDb

De este modo, la *API* devolvía otra respuesta *JSON* de la película indicada, la cual se debía recorrer en busca de los datos concretos que se querían extraer de ella. Con cada *URL* explorada se obtenían solamente unos datos determinados, por lo que este proceso debía repetirse con varias de ellas (una mostraba los detalles de la película, otra el reparto y el resto de los créditos, otra las críticas...).

Fue un proceso largo, pues se tuvo que emplear una gran cantidad de tiempo en explorar la documentación de la *API* para consultar las numerosas *URL* a las que se debía acceder, y qué datos nos proporcionaba cada una, así como el nombre de cada campo a extraer. También hubo que enfrentarse a ciertos retos, como el tratamiento de la paginación, pues en ocasiones la *API* devolvía una respuesta por páginas en vez de mostrar todos los resultados de una sola vez.

5.1.2 Web Scraping

Para extraer los datos de *IMDb* y *FilmAffinity* no se contaba con una *API*, por lo que se emplearon técnicas de *Web Scraping*. En ambas fuentes el proceso seguido fue el mismo: en primer lugar, se accedía a la página de cada una de las webs donde obtener las películas ordenadas por mayor puntuación, y así

extraer la *URL* individual de cada película; y después se accedía a cada una de las *URL* para obtener los datos de cada película individual.

Para conseguir las *URL* de las películas mejor valoradas, se debió utilizar *Web Scraping dinámico*, empleando la herramienta *Selenium* ya mencionada. Esto se debe a que el listado de películas, tanto de *IMDb* como de *FilmAffinity*, se encontraba paginado mostrando una cantidad reducida de películas por página, y se debía pulsar un botón para cargar la siguiente muestra de películas de forma sucesiva. De este modo, se creó una clase en *Python* en la que se accedía a la *URL* del listado de películas y, analizando el *HTML* de la web, se localizaba el botón que debíamos pulsar buscando según los elementos de la etiqueta que lo identificaban. Una vez encontrado, se configuraban las acciones que se debían realizar mediante métodos proporcionados por *Selenium*, es decir, se movía el cursor hacía el botón localizado y una vez allí este debía pulsarse. De esta forma, y mediante un bucle, se simulaba el pulsado del botón de forma repetida, permitiendo así la carga de todas las películas del listado y la obtención de la *URL* de cada una de ellas. Tras ello, las *URL* obtenidas se iban almacenando en un archivo *CSV* para poder acceder a ellas posteriormente.

Teniendo todas las *URL* de las películas, ya se podía acceder a cada una de ellas individualmente para extraer sus datos. Además, ya no era necesario usar *Selenium*, pues al no ser estas páginas dinámicas y no tener que interactuar con ellas, se podían emplear las técnicas de *Web Scraping* estático. Para ello, se hizo una clase en *Python* (una para *IMDb* y otra diferente para *FilmAffinity*) en la que se tomaban las *URL* almacenadas previamente en el *CSV* de la fuente correspondiente, y mediante la biblioteca *Requests* se realizaba una petición *GET* a cada una de ellas. En dicha petición, fue necesario incluir una variable en la cabecera para simular que se realizaba desde un navegador, y evitar así que la página web la bloqueara por posibles mecanismos de seguridad que pudiera tener.

Con cada petición, empleando la biblioteca *BeautifulSoup* se obtenía el *HTML* de la página con los datos de la película. El siguiente paso a realizar era recorrer dicho *HTML* en busca de cada uno de los datos a extraer, mediante los métodos “find” y “find_all” que proporciona dicha biblioteca. Para ello, se debía localizar cada dato mediante un elemento que lo identificara, que podía ser directamente su etiqueta o alguno de sus atributos. Al ser páginas extensas con numerosa información, se tuvo que realizar un análisis exhaustivo y detallado del *HTML* obtenido para conocer cómo identificar cada dato y no extraer información errónea. Entre los elementos que permitieron distinguirlos se encontraban atributos dedicados a indicar su “id”, texto o atributos de propiedades que solo poseía un dato concreto, metainformación, o incluso elementos del diseño *CSS* que también lograban su distinción. Además, en ocasiones debían realizarse varias búsquedas sucesivas dentro de cada etiqueta examinada para localizar el elemento concreto a extraer.

Por ejemplo, el dato de las empresas productoras de una película en la página *IMDb*, se encontraba dentro de una etiqueta “” cuyo atributo denominado “data-testid” era igual a “title-details-companies”. Sin embargo, dentro de dicha etiqueta aparecían numerosos datos además de los deseados, por lo que una vez localizada se debía realizar otra búsqueda en su interior para extraer

únicamente los nombres de las productoras, que era lo que se buscaba, mediante sus etiquetas y atributos correspondientes.

Este proceso de búsqueda debía realizarse con el resto de datos, pero cada uno con el elemento que lo identificara, lo que podía ser más o menos complejo. Por ejemplo, en el caso de las productoras se contaba con un atributo único que lo identificaba, pero otros datos no contaban con dicho atributo y debía buscarse otra forma de localizarlos. De ahí la razón de lo laborioso que supone este proceso de extracción.

Hay que reseñar que, en el caso particular de la obtención de la imagen de la película, no se obtuvo la imagen como tal, sino que se extrajo la *URL* de dicha imagen para facilitar su almacenamiento posterior en la base de datos.

5.2 Almacenamiento de datos

Los datos extraídos en el apartado anterior se almacenaron primero en distintos archivos CSV, uno para cada fuente, antes de transferirlos a la base de datos. De esta forma se facilitaba el proceso de almacenamiento, evitando tener que conectar con la base de datos en el mismo proceso de extracción, y también permitía una revisión previa y limpieza de datos más sencilla y eficiente. Además, los archivos CSV actuaban como copia de seguridad en caso de errores o borrado de la base de datos.

5.2.1 Limpieza y unificación del formato de datos

Como se ha comentado, al haber obtenido la información de tres fuentes distintas, era necesario realizar una revisión y limpieza de los datos antes de transferirlos a la base de datos, para tener una unificación del formato y evitar duplicados.

Uno de los aspectos que más costó realizar fue la unificación de la medida del tiempo de duración, pues una fuente lo mostraba en horas y minutos mientras que las otras dos tenían el tiempo solo en minutos. Se decidió tenerlo todo en minutos, por lo que se tuvo que realizar una conversión de la medida en la fuente donde no se encontraba de esta forma. También hubo que transformar de ciertas maneras los países de las películas, pues *TheMovieDatabase* los contenía en inglés y hubo que traducirlos al castellano. De igual modo, había ciertas incongruencias entre los propios países en español, con leves diferencias en algún carácter (por ejemplo, “Bosnia y Herzegovina” y “Bosnia-Herzegovina”) que también había que unificar para que no se contaran como dos países distintos. Otro problema encontrado fue que, en *IMDb*, los títulos de las películas que contenían alguna palabra entre paréntesis no las detectaba, por lo que hubo que solucionarlo cambiando la etiqueta de búsqueda en la fase de *Web Scraping*. También hubo problemas con la parte de la puntuación, pues en algunas se marcaban los decimales con un punto y en otras con una coma, decidiéndose sustituir las comas por los puntos. Además, en la extracción del número de votos, en *IMDb* se mezclaban número y texto (por ejemplo, se

mostraba “79 mil votos”) por lo que hubo que convertirlo todo a número. Por último, hubo que realizar transformaciones en las críticas, para que en todos los casos se mostrara primero el nombre del autor y a continuación el texto de la crítica, y hubo que realizar una limpieza de ciertos caracteres o emoticonos que introducían los usuarios en su crítica pero que al extraerlos no se reconocían.

Una vez limpios y revisados los datos, estos ya se podían transferir desde los distintos CSV a una sola base de datos que se encargara del almacenamiento de la aplicación.

5.2.2 Implementación de la base de datos

Una vez hecho el diseño de la base de datos, como se ha visto en el capítulo anterior, ya podía realizarse su implementación. Esta se realizó en *MySQL*, en concreto empleando su herramienta visual *MySQL Workbench*. En ella se creó la base de datos en local con cada una de sus tablas, mediante la ejecución de un fichero con código y consultas en *SQL* (con sentencias como “CREATE TABLE” o “INSERT INTO”). Se creó una tabla para cada una de las entidades que aparecen en el diagrama Entidad-Relación, cada una con sus atributos correspondientes, y también una tabla para representar cada una de las relaciones. En las tablas de relación, además de sus posibles atributos, se almacenaban el “id” de cada una de las entidades que vinculaba, referenciadas mediante claves foráneas, y la combinación de ambos “id” actuaba como clave primaria de la tabla. En la *Figura 13* se puede apreciar la creación de la tabla de la relación entre “Película” y “Género”, donde se relaciona el “id” de cada película con cada “id” de los géneros a los que pertenece.

```
-- Tabla que relaciona las películas con sus géneros (N:M)
CREATE TABLE rela_pelicula_genero (
    id_pelicula INT NOT NULL,
    id_genero INT NOT NULL,
    PRIMARY KEY (id_pelicula, id_genero),
    FOREIGN KEY (id_pelicula) REFERENCES pelicula (id),
    FOREIGN KEY (id_genero) REFERENCES genero (id)
);
```

Figura 13. Creación de la tabla de la relación entre Película y Género

De esta manera se creaba la estructura vacía de la base de datos, pero faltaba transferir la información extraída de cada fuente desde los CSV hasta ella.

Para realizarlo, se creó una clase en *Python* donde se establecía una conexión con la base de datos creada, y mediante la biblioteca *Pandas* se leía el CSV de cada fuente y se separaban sus columnas de datos por campos. A continuación se analizaba cada campo por separado, realizando consultas en la base de datos a través de *Python* para insertar los datos en su tabla correspondiente. Primero

se realizaba una consulta “SELECT” para comprobar si el dato ya había sido introducido previamente en la tabla, consultando su “id” mediante el nombre del elemento que queríamos introducir (por ejemplo, consultando el “id” del elemento con nombre “España” en la tabla “País”). De esta manera se evitaban los duplicados, ya que ese elemento podría haberse introducido previamente tanto desde la fuente que se estaba analizando como de otras fuentes. En caso de no encontrarlo, significaba que todavía no se había añadido dicho elemento, por lo que se realizaba una consulta “INSERT INTO” para añadir el dato en su tabla. A cada dato nuevo introducido se le asignaba un “id” automáticamente mediante un autoincremento, y posteriormente para insertarlos en las tablas de relaciones, se consultaban los “id” correspondientes y se introducían en la tabla.

Una vez introducidos los datos de cada CSV en la base de datos, esta englobaba la información extraída de cada una de las fuentes en un solo sitio, contando con un sistema de almacenamiento estructurado y listo para conectarse con el resto de la aplicación a implementar.

5.3 Desarrollo web de la aplicación

Tras haber extraído y almacenado los datos a emplear, la siguiente etapa a realizar fue el desarrollo de la aplicación web en sí. Se debía implementar tanto la parte del *frontend* con su interfaz web, como la parte del *backend*, además de la estructura que conectaba y relacionaba todos los componentes entre sí.

5.3.1 Implementación

Para implementar la aplicación conteniendo todas las pestañas y características deseadas, se usó la herramienta *Django*. Tal y como se explicó en el *Capítulo 3* de esta memoria, *Django* emplea un diseño “Modelo-Vista-Template”, lo que se aplica también en la programación del código mediante esta herramienta. Una vez creado un proyecto en *Django*, este nos genera una serie de clases que se deben completar manualmente mediante el código deseado, pero la configuración y funciones internas de la aplicación ya se encuentran realizadas por la propia herramienta.

A continuación, se explicarán las principales clases implementadas y las finalidades de cada una.

urls.py

En esta clase se han definido todas las *URL* que se emplean dentro de la propia aplicación desarrollada, que se corresponden con cada una de las pestañas o apartados a los que redirige la aplicación cuando el usuario final navega a través de ella mediante su interfaz. En cada una de ellas se ha definido su ruta, la vista con la que se corresponde y un nombre que la identifica, como se puede ver en la *Figura 14*.

```
urlpatterns = [
    path('', views.principal, name='principal'),
    path('listado/', views.listado_peliculas, name='listado_peliculas'),
    path('buscador/', views.buscador, name='buscador'),
    path('rankings/', views.rankings, name='rankings'),
    path('listado/<int:id_pelicula>/datos/', views.datos, name='datos'),
    path('listado/<int:id_pelicula>/ranking/', views.ranking, name='ranking'),
    path('listado/<int:id_pelicula>/criticas/', views.criticas, name='criticas'),
    path('listado/<int:id_pelicula>/datos-filtros/', views.datos_filtros, name='datos-filtros'),
    path('listado/<int:id_pelicula>/ranking-filtros/', views.ranking_filtros, name='ranking-filtros'),
    path('listado/<int:id_pelicula>/criticas-filtros/', views.criticas_filtros, name='criticas-filtros'),
    path('listado/<int:id_pelicula>/datos-rankings/', views.datos_rankings, name='datos-rankings'),
]
```

Figura 14. Fragmento de "urls.py"

De esta forma se tienen agrupadas en una misma clase todas las *URL*, y estas se pueden referenciar desde otras clases del proyecto cuando sea necesario, mediante el nombre definido.

models.py

En la parte de configuración del proyecto *Django* se debió importar la base de datos ya implementada que se deseaba en la aplicación. Una vez hecho se generó esta clase que incluye el modelo de datos producido, determinando cada una de las tablas creadas, con sus atributos, claves primarias y claves foráneas; y también las tablas de relaciones indicando las entidades que enlazan y sus atributos en caso de haberlos.

De esta forma se transformaron todas las tablas en clases u objetos de *Python*, incorporándolas directamente en la estructura del proyecto y pudiendo interactuar con ellas más fácilmente, evitando así tener que emplear consultas *SQL* y tener que realizar conexiones con la base de datos de forma continua. Además, esto también facilitaba la creación, modificación y borrado de tablas desde *Django*, así como la variación de ciertas características de la base de datos en caso de ser necesario.

En la *Figura 15* se muestra un fragmento de esta clase, donde se aprecian algunas de las tablas de la base de datos transformadas en clases de *Python*.

```

class Pelicula(models.Model):
    titulo = models.CharField(max_length=100)
    anio = models.IntegerField()
    imagen_url = models.TextField(blank=True, null=True)
    sinopsis = models.TextField(blank=True, null=True)
    duracion = models.CharField(max_length=30, blank=True, null=True)
    clasificacion_edad = models.CharField(max_length=30, blank=True, null=True)
    class Meta:
        managed = False
        db_table = 'pelicula'

class Productora(models.Model):
    nombre = models.CharField(unique=True, max_length=150)
    class Meta:
        managed = False
        db_table = 'productora'

class RelaPeliculaActor(models.Model):
    pk = models.CompositePrimarykey('id_pelicula', 'id_actor')
    id_pelicula = models.ForeignKey(Pelicula, models.DO_NOTHING, db_column='id_pelicula')
    id_actor = models.ForeignKey(Actor, models.DO_NOTHING, db_column='id_actor')
    class Meta:
        managed = False
        db_table = 'rela_pelicula_actor'
        unique_together = (('id_pelicula', 'id_actor'),)

```

Figura 15. Fragmento de "models.py"

En ellas se pueden ver los atributos de cada tabla tal y como se definieron en *MySQL*, con sus características definidas como la longitud máxima o la posibilidad de ser un valor nulo, y las claves primarias y foráneas indicando sus tablas de origen.

views.py

En esta clase es donde se realizan la mayoría de las operaciones en cuanto a la lógica de la aplicación. En ella se definen distintas funciones, las llamadas “vistas”, que se llaman desde otras clases como “urls.py”, donde cada *URL* se corresponde con una vista. Las vistas se encargan de recibir las peticiones *HTTP* realizadas por el usuario, y obtener los datos correspondientes desde los modelos. Dichos datos se operan o se transforman, en caso de ser necesario, y se devuelven mediante una respuesta que se recogerá en las plantillas. En la

Figura 16 se puede observar una de las vistas programadas en esta clase, correspondiente a la página que muestra los rankings.

```
#Página rankings
def rankings(request):
    #Obtenemos las películas de cada fuente, ordenadas según su posición en cada una de ellas
    pelis_TMDB = RelaPelículaFuente.objects.filter(id_fuente_nombre = "TMDB").order_by("posicion")
    pelis_IMDb = RelaPelículaFuente.objects.filter(id_fuente_nombre = "IMDb").order_by("posicion")
    pelis_FA = RelaPelículaFuente.objects.filter(id_fuente_nombre = "FilmAffinity").order_by("posicion")
    return render(request, 'rankings/rankings.html', {
        "pelis_TMDB": pelis_TMDB,
        "pelis_IMDb": pelis_IMDb,
        "pelis_FA": pelis_FA
    })
```

Figura 16. Fragmento de "views.py"

En ella se puede apreciar cómo se recibe una petición como parámetro de la función, se obtienen las películas de cada una de las fuentes ordenadas por posición, y estas se devuelven como respuesta en la plantilla "rankings.html".

Plantillas (templates)

En cuanto a las plantillas, se creó un directorio dentro del proyecto donde están incluidas todas ellas. En cada plantilla se establece el código *HTML* de una página o pestaña de la aplicación, donde se define su contenido, determinando la estructura de la web y describiendo sus distintos componentes y elementos con los que puede interactuar el usuario (como pueden ser botones, enlaces...). Es decir, en ellas se establece lo que posteriormente será la interfaz visual de la aplicación. Además, cada vista de "views.py" envía los datos obtenidos del modelo a una plantilla correspondiente, donde se puede determinar la forma de representar dichos datos.

En las plantillas también se define el diseño gráfico de cada página, lo que se corresponde con su "decoración", a través de lenguaje *CSS*. En este proyecto, en lugar de mezclar los elementos *CSS* dentro de las plantillas *HTML*, se decidió realizarlo en un archivo aparte. De esta forma, en dicho archivo se definen en diversas clases de *CSS* los diseños deseados para cada página o componente de dentro de ella, como pueden ser los colores, márgenes, tipo y tamaño de letra o la disposición de sus distintos elementos. Posteriormente, en cada plantilla se importa este archivo y se indica la clase *CSS* a emplear en las etiquetas *HTML*. Al realizarlo de esta manera, se permite la reutilización de código *CSS* en distintos elementos donde se requiera el mismo diseño, y permite modificar de forma más fácil tanto el diseño como la estructura del *HTML* al tenerlos en

archivos separados. A continuación, en la *Figura 17* y la *Figura 18*, se muestran fragmentos de la plantilla que define la página del listado de películas.

```
{% load static %}

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Listado de Películas</title>
    <link rel="stylesheet" href="{% static '/styles/main.css' %}">
</head>
<body class="fondo-paginas">
    <div class="cabecera-paginas">
        <div class="cabecera-paginas-izq">
            <a href="{% url 'principal' %}" class="boton-cabecera">Menú principal</a>
        </div>
        <div class="cabecera-paginas-centro">
            <h1 class="titulo">TopPelisDB 🍿🎬</h1>
        </div>
        <div class="cabecera-paginas-der">
            <a href="{% url 'listado_peliculas' %}" class="cabecera-enlace">Lista</a>
            <a href="{% url 'buscador' %}" class="cabecera-enlace">Búsqueda</a>
            <a href="{% url 'rankings' %}" class="cabecera-enlace">Rankings</a>
        </div>
    </div>
```

Figura 17. Fragmento de "listado_peliculas.html", correspondiente a la cabecera

Esta parte de código se corresponde con el inicio de la plantilla, donde se aprecia la importación del archivo CSS (main.css) y el código *HTML* que define la cabecera, en la que se puede observar la incorporación de diversas *URL* que permiten navegar por la aplicación. En la parte izquierda de la cabecera se estableció el enlace a la página principal, en el medio el título de la aplicación, y en la parte derecha tres enlaces que se corresponden con cada una de las pestañas con las que cuenta la aplicación. También se puede distinguir la incorporación de las distintas clases CSS mediante los atributos “class” de las etiquetas, donde en el archivo “main.css” se establece la posición de cada elemento, el color del texto o la característica que provoca que la cabecera se mantenga fija, entre otros, mientras que en la plantilla solo se muestra el nombre de cada clase empleada. Esto permite tener un código más limpio y legible, pudiéndose reutilizar los diseños al definir la cabecera en otras plantillas.

```

<div class="listado-pelis">
<h1 class="titulo-pagina">Listado de Películas</h1>
{% for pelicula in peliculas %}
  <div class="peli">
    <h2 class="titulo-peli">{{ pelicula.titulo }} - {{ pelicula.anio }}</h2>

    <div class="contenido-peli">

      {% if pelicula.imagen_url %}
        
      {% else %}
        <p>[Imagen no disponible]</p>
      {% endif %}

      <div class="conjunto">

        <div class="conjunto-botones">
          <a href="{% url 'datos' pelicula.id %}" class="botones">Datos</a>
          <a href="{% url 'ranking' pelicula.id %}" class="botones">Ranking</a>
          <a href="{% url 'criticas' pelicula.id %}" class="botones">Críticas</a>
        </div>

        <div class="sinopsis">
          <p>{{ pelicula.sinopsis }}</p>
        </div>
      </div>
    </div>
  </div>
{% endfor %}
</div>

```

Figura 18. Fragmento de "listado_peliculas.html", correspondiente al manejo de datos

En este otro fragmento se puede apreciar cómo se manejan ciertos datos para mostrarlos en la aplicación. Estos se han obtenido a través de la respuesta dada por la vista correspondiente de "views.py", que se los proporciona a esta plantilla y permite así poder emplearlos en bucles "for" o condicionales "if" para determinar el contenido de la web. Posteriormente en esta memoria, se mostrará una imagen que muestra el resultado visual de ambos fragmentos de la plantilla.

Las clases mencionadas han sido las principales sobre las que se ha trabajado en el proyecto de *Django*, pero hay numerosas clases adicionales que son necesarias para la implementación y configuración del proyecto. A continuación, se abordarán por separado las distintas pestañas de la aplicación.

5.3.2 Página principal

A pesar de no ser una pantalla totalmente necesaria, se decidió realizar una página principal a modo de "portada". Su *URL* se corresponde con la ruta de la "raíz" de la aplicación, a partir de la cual se derivan el resto de rutas para acceder a las distintas pantallas de las que se dispone. De esta forma, cuando el usuario accede a la aplicación, la primera página a la que se le dirige es a esta página inicial. En ella se puede observar el título de la aplicación en la parte de arriba, "TopPelisDB", y tres botones en el medio mediante los que se accede a las distintas pestañas de las que dispone la aplicación, mediante sus *URL*. En

cuanto a su diseño visual, se pretendía crear en el usuario la sensación de adentrarse en una sala de cine, de acorde con la temática del proyecto. Para ello, se escogió un fondo oscuro y, en el medio, los botones se disponen sobre fondo blanco para simular la pantalla. Este diseño de “pantalla de cine” se propagó al resto de las pestañas de la aplicación, como se verá a lo largo de este capítulo. En la *Figura 19* se muestra el diseño final de la pantalla principal de la aplicación.

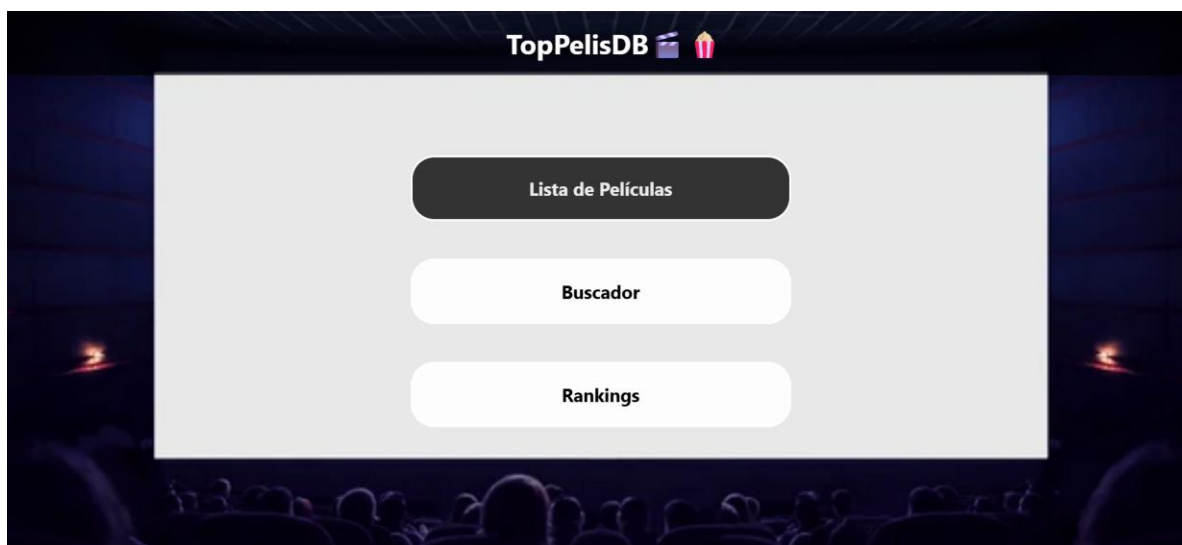


Figura 19. Pantalla principal de la aplicación

5.3.3 Lista de películas

Al acceder a la pestaña de “Lista de Películas”, la aplicación muestra un listado completo de todas las películas presentes en la aplicación. Estas aparecen en orden según sus “id” correspondientes en la tabla “Película” de la base de datos, es decir, en orden de adición a la tabla, por lo que indirectamente estarán ordenadas según el ranking de cada fuente conforme se iban añadiendo.

En esta lista se exponen algunos de los datos destacados de cada película, como son el título, el año de estreno, la imagen y la sinopsis. Como se mencionó anteriormente, en la base de datos no se guarda la imagen como tal, sino su *URL*, para no tener una base de datos pesada y mejorar su rendimiento. Por tanto, al cargar cada imagen se hace referencia a su *URL* (en su plantilla *HTML* correspondiente) mediante una etiqueta, como se puede ver en la *Figura 20*, y se cargan las imágenes en tiempo de ejecución.

```
{% if pelicula.imagen_url %}
  
{% else %}
  <p>[Imagen no disponible]</p>
{% endif %}
```

Figura 20. Carga de imágenes a través de sus URL

Además de los datos mencionados, en la lista se incluyen tres botones distintos por cada película por si el usuario quiere obtener más datos de cada una: uno correspondiente a sus datos más detallados, otro para obtener sus rankings en las distintas páginas de las que se han extraído, y otro para mostrar las críticas de usuarios de diversas fuentes. Estos se abordarán con detalle más adelante.

Como ya se ha adelantado, el diseño visual de esta página (y del resto) se buscaba que simulara la sensación de encontrarse en una sala de cine. Por tanto, mediante código CSS, se estableció un fondo oscuro y cada película se encasilló dentro de una tarjeta de color claro, con bordes brillantes y amarillentos para representar cada una como una pantalla de cine. Además, se estableció en la parte superior de la pantalla una cabecera, que se mantiene fija aunque el usuario se desplace hacia abajo. En ella se muestra un botón que permite retroceder al menú principal de la aplicación, en su parte izquierda; el título de la aplicación, en el centro; y tres enlaces que permiten navegar hacia las tres pestañas principales de la aplicación, en su parte derecha (esta cabecera se corresponde con el código de la plantilla *HTML* mostrado en la *Figura 17*). El diseño de la interfaz de esta página, tanto el fondo oscuro con tarjetas brillantes como la cabecera, se repite también en el resto de páginas de la aplicación, para mantener una coherencia visual a lo largo de ella.

A continuación, en la *Figura 21*, se muestra el resultado de esta página (que se corresponde con el código de la plantilla *HTML* mostrado en la *Figura 18*).



Figura 21. Página de "Lista de Películas"

Como se puede observar, cada tarjeta de una película contiene los tres botones mencionados anteriormente, que se explicarán en los siguientes apartados.

5.3.4 Datos de una película

Si en el listado de películas anterior el usuario pulsa el botón de “Datos”, la aplicación le redirige a una pantalla donde se muestran todos los datos detallados disponibles de dicha película. Estos son los que ya se mostraban previamente (título, año, imagen y sinopsis) junto con su duración, clasificación de edad, género(s), país(es) de origen, empresas productoras, reparto de actores, director, guionistas y compositor. En caso de no estar disponible alguno de los datos de la película, se muestra el texto “No disponible”.

Para lograr que se muestren los datos de cada película concreta, en la parte de “urls.py” se establece una *URL* que depende del “id” de la película seleccionada, como se puede apreciar en la *Figura 22*.

```
path('listado/<int:id_pelicula>/datos/', views.datos, name='datos'),
```

Figura 22. URL dependiente del “id” de la película seleccionada

Cuando el usuario pulsa el botón, se ha configurado la plantilla *HTML* para que se establezca el “id” de la película seleccionada en dicha *URL*, quedando así completa. De esta forma, en la vista correspondiente de “views.py” se recibe la petición *HTTP* junto con el “id” de la película, a través del que obtiene los datos correspondientes desde el modelo pudiendo así mostrar la información de la película concreta deseada.

En la *Figura 23* se muestra esta pantalla, donde se puede apreciar cómo la cabecera, en este caso, contiene un botón que retrocede a la pantalla anterior en lugar de regresar directamente al menú principal, por si el usuario quisiera seguir explorando la lista de películas u otros apartados disponibles de la película deseada.

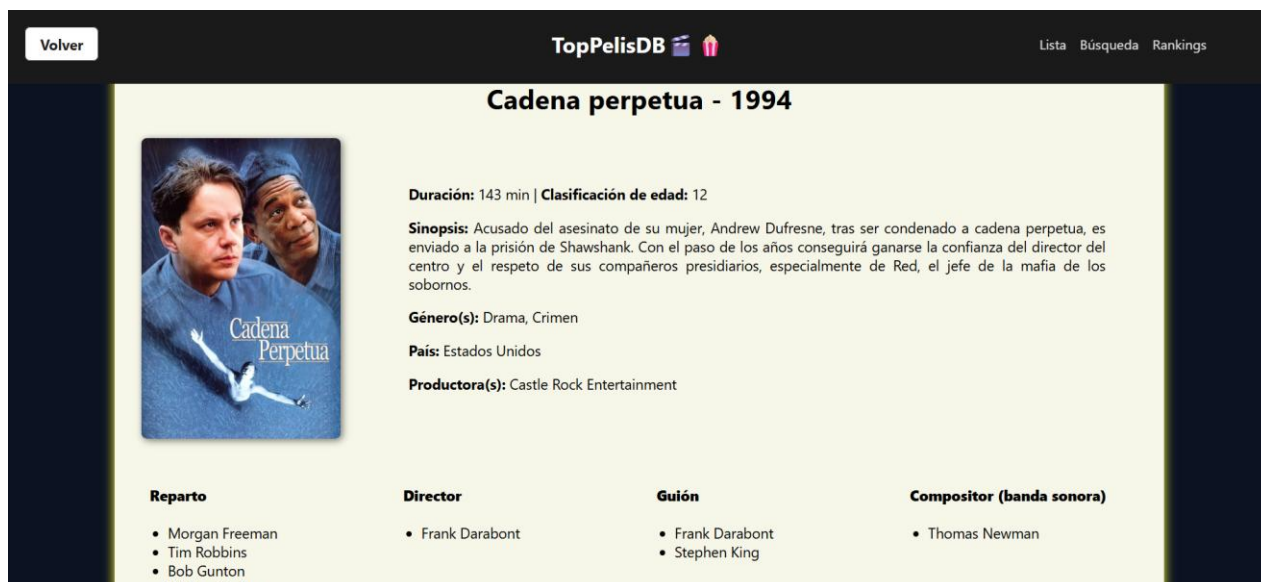


Figura 23. Página de “Datos” de una película

5.3.5 Rankings de una película

Otro de los botones disponibles en el listado de películas se corresponde con la visualización de los rankings de cada película. De esta forma, se redirige al usuario a una pestaña donde se indica cada una de las fuentes de extracción donde aparece la película, y se muestra su posición en el ranking, su puntuación y el número de votos obtenido por usuarios de cada una de las fuentes. En caso de no aparecer la película en alguna de las fuentes, no se mostrará su tarjeta correspondiente. Para obtener los datos de cada película concreta, se ha seguido la misma lógica que la explicada en la página anterior.

Esta pestaña se puede observar en la *Figura 24*.



The screenshot shows the 'Rankings' page for the movie 'Cadena perpetua - 1994' on the TopPelisDB website. The page features a dark blue header with a 'Volver' button, the site logo, and navigation links for 'Lista', 'Búsqueda', and 'Rankings'. The movie title is prominently displayed. To the left is the movie poster. To the right are three ranking cards, each with a source name and a list of statistics.

Fuente	Posición	Puntuación	Nº Votos
TMDB	1	8.71	28078
IMDb	1	9.30	3000000
FilmAffinity	6	8.60	172027

Figura 24. Página de "Rankings" de una película

5.3.6 Críticas de una película

La última pestaña a la que se puede acceder desde el listado de películas se corresponde con el botón de "Críticas". Este redirige al usuario a una pestaña donde se muestran diversas críticas de los usuarios de cada una de las fuentes de extracción, mostrando el nombre de usuario de quién realizó la crítica y seguidamente su contenido. Hay que reseñar que hay fuentes donde sí que aparece la película, pero no se dispone de ninguna crítica hacia ella, por lo que en este caso solo se mostrarán las fuentes que contengan críticas de la película seleccionada. En caso de no contar con ninguna fuente con críticas hacia una película, se mostrará el texto "No existen críticas para esta película". También hay que destacar que un gran número de películas sólo tienen disponibles críticas en inglés.

A continuación, en la *Figura 25*, se muestra esta pestaña.

Volver

TopPelisDB

Lista Búsqueda Rankings

Cadena perpetua - 1994



TMDB

- javimar: A total example of how we never have to lose HOPE.

IMDb

- Sleepin_Dragon: It is no wonder that the film has such a high rating, it is quite literally breathtaking. What can I say that hasn't said before? Not much, it's the story, the acting, the premise, but most of all, this movie is about how it makes you feel. Sometimes you watch a film, and can't remember it days later, this film loves with you, once you've seen it, you don't forget. The ultimate story of friendship, of hope, and of life, and overcoming adversity. I understand why so many class this as the best film of all time, it isn't mine, but I get it. If you haven't seen it, or haven't seen it for some time, you need to watch it, it's amazing. 10/10.
- EyeDunno: I'm trying to save you money; this is the last film title that you should consider borrowing. Renting

Figura 25. Página de "Críticas" de una película

5.3.7 Buscador

Una vez exploradas las distintas pestañas a las que da acceso la página correspondiente al listado de películas, se va a explicar otra de las páginas principales de la aplicación, correspondiente al "Buscador". En ella, el usuario puede establecer una serie de filtros para buscar las películas contenidas en la aplicación que cumplan con las características deseadas.

Para ello, se han implantado una serie de campos que el usuario puede rellenar con los datos que desee. Los ámbitos por los que el usuario puede filtrar son: título, año, género, país, productora, actor, director, guionista, compositor y fuente. Hay que destacar que el usuario puede rellenar todos los filtros disponibles, o únicamente los que desee. Además, los datos introducidos no tienen que ser estrictamente exactos a cómo están almacenados en la base de datos para que se tengan en cuenta, es decir, no discrimina según mayúsculas y minúsculas e incluso admite escribir solamente una parte del texto (por ejemplo, si en vez de escribir en el filtro de actor "Leonardo DiCaprio", se escribe únicamente "dicaprio", la aplicación devolverá igualmente las películas donde aparezca dicho actor). Una vez pulsado el botón de búsqueda con los campos deseados rellenos, la aplicación devuelve una lista de las películas que cumplen con dichas características de la misma forma que en la pestaña vista anteriormente, junto con los botones de "Datos", "Rankings" y "Críticas".

En cuanto a su funcionamiento interno, en la plantilla *HTML* correspondiente se ha definido lo que se denomina un "formulario", donde se han establecido los distintos campos por los que el usuario puede filtrar y las acciones a realizar con el conjunto de datos introducidos en ellos. En la *Figura 26*, se muestra un fragmento de código de la plantilla *HTML* correspondiente a dicho formulario, donde se establecen algunos de los campos junto con el tipo de datos que se espera del usuario.

```

<form action="{% url 'buscador' %}" method="get">
  <div class="campos-formulario">
    <label for="titulo">Título: </label>
    <input id="titulo" type="text" name="titulo" value="{ { filtros_introducidos.titulo } }">
    <label for="anio">Año: </label>
    <input id="anio" type="number" name="anio" min="1900" max="2025" value="{ { filtros_introducidos.anio } }">
    <label for="genero">Género: </label>
    <input id="genero" type="text" name="genero" value="{ { filtros_introducidos.genero } }">
    <label for="pais">País: </label>
    <input id="pais" type="text" name="pais" value="{ { filtros_introducidos.pais } }">
    <label for="productora">Productora: </label>
    <input id="productora" type="text" name="productora" value="{ { filtros_introducidos.productora } }">
  </div>
</form>

```

Figura 26. Fragmento de la plantilla "buscador.html", correspondiente al formulario

Una vez se pulsa el botón de buscar, se forma una *URL* con cada uno de los filtros rellenos. Estos datos se obtienen en la vista correspondiente (de "views.py") mediante métodos *GET*, donde se filtran las películas y se van seleccionando aquellas que cumplen con todos los filtros cumplimentados por el usuario. Una vez se han obtenido las películas filtradas, estas se devuelven a la plantilla, donde ya se podrán emplear para representar los datos de las películas que cumplen con las características deseadas.

A continuación, en la Figura 27, se puede observar un ejemplo de búsqueda con ciertos filtros en esta pestaña.

Figura 27. Página de "Buscador"

5.3.8 Rankings

En la última pestaña que ofrece la aplicación se exhiben los rankings de las tres páginas web de las que se han extraído los datos, uno al lado del otro, para que se puedan comparar entre sí y contrastar las puntuaciones y votos de cada fuente. En cada uno de los rankings se muestran, de forma ordenada de mayor a menor puntuación, las 1000 películas con mejor valoración. En ellos se indica el nombre y año de estreno de cada película, su puntuación y su número de votos, como se puede ver en la *Figura 28*.

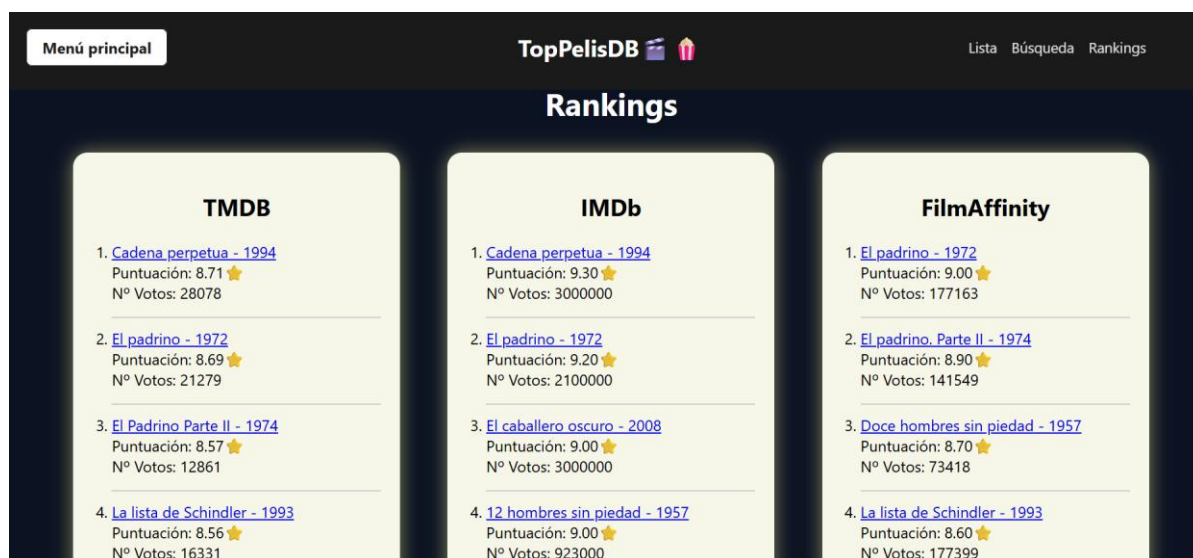


Figura 28. Página de "Rankings"

Además, se puede observar que el título de cada película consiste en un enlace, mediante el que se redirige a la página de datos detallados de cada película concreta.

5.4 Evaluación

Una vez implementada la aplicación en su totalidad y explicadas todas sus pantallas y funcionalidades, se puede realizar una evaluación del resultado final.

La aplicación fue probada por distintos usuarios, todos ellos sin altos conocimientos tecnológicos, para comprobar principalmente la facilidad de uso e intuición de la interfaz, que era uno de los principales objetivos a conseguir, así como su usabilidad. Ninguno de ellos presentó excesivos problemas navegando por la aplicación y consiguieron acceder a cada una de las pestañas disponibles, acceder a los distintos datos y comparaciones, y realizar búsquedas de películas mediante filtros, todo ello sin haber sido informados previamente de las funcionalidades y modo de uso de la aplicación. Además, el tiempo de respuesta y rendimiento de la aplicación fue bastante aceptable, sin problemas de carga de datos visibles. Sin embargo, es cierto que se detectaron pequeños desajustes en la aplicación, como por ejemplo al pulsar sobre el enlace de una

película en la pestaña de “Rankings”, si posteriormente se pulsaba el botón de volver hacia atrás, se redirigía al listado de películas en vez de retornar de nuevo a los rankings. Este problema se revisó y se solucionó, ajustando la redirección de los botones de la cabecera de forma individual en cada pestaña, y también se consiguieron solventar el resto de los conflictos detectados correspondientes a pequeños detalles de diseño.

Posteriormente, se propuso a los mismos usuarios navegar por las aplicaciones y páginas web ya existentes mencionadas en el capítulo de “Webs cinematográficas existentes”. Se pudo observar cómo los usuarios evaluados tuvieron más dificultades usando estas aplicaciones, y emplearon más tiempo para acceder a todos los datos de las películas deseadas. Además, se confirmó que la aplicación propuesta en este proyecto contiene ciertas funcionalidades de las que dichas webs no disponen, como la comparación de las puntuaciones, críticas y valoraciones de las películas entre diversas fuentes.

6 Conclusiones y trabajo futuro

Una vez finalizado el proyecto, se pueden sacar varias conclusiones sobre el trabajo realizado.

Tras haber completado la implementación de la aplicación propuesta, se puede determinar que se ha conseguido el objetivo principal del trabajo, que era el desarrollo de una aplicación web para el análisis y visualización de datos de películas, permitiendo comparar la información obtenida de distintas fuentes mediante una interfaz intuitiva y fácil de usar para cualquier tipo de usuario. A lo largo de todo el proceso, se han podido explorar las distintas etapas de diseño y de desarrollo seguidas para conseguir la aplicación propuesta, logrando implementar todas las funcionalidades deseadas inicialmente y obteniendo un resultado accesible y sencillo de utilizar para los usuarios que deseen acceder a la aplicación.

A continuación, se va a hacer un repaso de los distintos objetivos específicos propuestos en la fase inicial del proyecto:

1. Estudio y familiarización con las tecnologías a emplear: se puede afirmar que se ha empleado una gran cantidad de tiempo aprendiendo a utilizar la gran variedad de herramientas utilizadas en el proyecto. Este tiempo de aprendizaje fue imprescindible para poder realizar la aplicación propuesta, pues se han empleado tecnologías no usadas previamente y que era necesario utilizar a lo largo del proyecto. Por tanto, se puede concluir que se ha adquirido una gran cantidad de conocimiento para poder aplicarlo en este trabajo.
2. Realización de un análisis de los requisitos del sistema: este objetivo también se ha realizado, pues se debían conocer las aplicaciones similares ya existentes actualmente y analizar tanto sus aportaciones como sus carencias, para así poder determinar qué funcionalidades incluir en la aplicación y cómo proporcionar nuevas características.
3. Diseño de la arquitectura de la aplicación: previo a la implementación de la aplicación, se realizó un diseño adecuado de la arquitectura a emplear en base al funcionamiento y características deseadas. Se establecieron todos sus componentes y la forma de relacionarse entre ellos, y también se realizaron diseños más concretos como de la base de datos o del modelo de la aplicación, facilitando el posterior desarrollo.
4. Extracción de los datos de películas a emplear en la aplicación: este objetivo fue de los que más tiempo llevó completar. Una vez decididas las fuentes de extracción, se tuvo que realizar un proceso detallado y exhaustivo donde se analizó la forma de extraer cada uno de los datos requeridos, para posteriormente aplicar de forma precisa las tecnologías pertinentes. Finalmente se consiguieron obtener todos los datos de cada una de las fuentes, para posteriormente emplearlos en la aplicación.

5. Implementación de la aplicación: se realizó el desarrollo completo de la aplicación, consiguiendo implementar todas las características deseadas y cumpliendo con los requisitos determinados previamente. Se completó con éxito cada una de las etapas del proceso, desarrollando una interfaz visual intuitiva enlazada de forma óptima con la parte de *backend* y la base de datos, obteniendo así una aplicación potente y con un gran rendimiento.
6. Realización de pruebas sobre el desarrollo para comprobar su correcto funcionamiento: se realizaron numerosas pruebas a lo largo de todo el proyecto para verificar su correcta implementación. Estas no solo se realizaron una vez finalizada la aplicación, sino que a lo largo de las distintas etapas del proyecto se fueron realizando las pruebas necesarias para comprobar que su funcionamiento fuera correcto, ya que la existencia de fallos en cualquier parte de la aplicación influye en el desempeño global.

Por tanto, se puede determinar que se han completado con éxito todos los objetivos propuestos al inicio del proyecto, consiguiendo obtener una aplicación potente y bien implementada mediante la que visualizar, analizar y comparar diversos datos sobre películas.

Sin embargo, existen ciertas propuestas de mejora y trabajos futuros a realizar para seguir aumentando las capacidades y el rendimiento de la aplicación.

En primer lugar, como ya se ha comentado, en la base de datos de la aplicación se almacena solo una muestra de las películas totales disponibles en cada fuente de extracción empleada. Por tanto, la primera propuesta a realizar sería la adición del total de películas disponibles para contar con una muestra completa, para lo que sería necesario contar con una base de datos más grande y potente para que, al disponer de tantos datos, esto no afecte a su rendimiento ni al tiempo de carga de la aplicación. También se propone como futura mejora la incorporación de más datos disponibles por película, como pueden ser sus premios y nominaciones, plataformas de streaming donde visualizarlas, un apartado de créditos completo o incluso el tráiler de la película. Otra propuesta sería una mejora en cuanto a las puntuaciones de las películas, con la incorporación de un ranking global según la puntuación media calculada sobre todas las fuentes de extracción, y una optimización de las críticas disponibles traduciendo al castellano aquellas que aparecen en inglés. Como última mejora, se propone el registro de usuarios en la aplicación para que estos puedan interactuar de forma más directa con ella, añadiendo sus propias valoraciones y críticas.

A continuación, se van a exponer algunas conclusiones personales sobre el proyecto.

Una vez finalizado este Trabajo de Fin de Grado, puedo afirmar que he quedado bastante satisfecho con el resultado conseguido, habiendo podido realizar la aplicación tal y como tenía pensado en un primer momento incluyendo todas

sus características y funcionalidades, así como su diseño visual. Es cierto que ha sido un proceso largo y duro, con mucho trabajo realizado, y dónde he tenido que emplear largas horas en aprender todas las tecnologías necesarias y aplicarlas en la realización del proyecto. Sin embargo, considero que todo ese trabajo ha merecido la pena y me enorgullece haber podido ampliar mis conocimientos, pues en estos meses he conocido y aprendido numerosas herramientas y tecnologías que inicialmente no sabía de su existencia y que me pueden ser de gran utilidad en el futuro. Además, he podido mejorar y ampliar mis habilidades sobre diversos lenguajes de programación, como *Python* o el lenguaje *HTML*, y aprender el uso de nuevas bibliotecas. También he podido emplear numerosas de las competencias adquiridas durante la carrera en un proyecto largo y complejo como ha sido la implementación de esta aplicación, y he podido aplicar mis capacidades y conocimientos de forma práctica. De esta forma, he podido evaluarme a mí mismo y darme cuenta de que mis niveles de programación son más altos de los que pensaba, no solo con respecto al manejo del código en sí y de sus estructuras, sino también en cuanto al razonamiento lógico y cómo enfocar el proyecto, las pruebas y depuración de errores y la limpieza del código.

Por último, quiero agradecer a todas las personas que han hecho posible la realización del proyecto. A mi tutor Antonio y a todos los profesores de la carrera, por orientarme, resolver mis dudas y enseñarme durante todos estos años; a mi familia y a Lucía, por aguantarme y apoyarme siempre que lo he necesitado; y a mis amigos y compañeros de la UPM, por haber conseguido que estudiar una ingeniería fuera más divertido junto a ellos.

7 Análisis de Impacto

Una vez finalizado el proyecto, se puede realizar un análisis de su impacto en la sociedad.

Principalmente, la aplicación implementada supone un impacto cultural, al incluir y mostrar numerosos datos de películas de distintos países, épocas y géneros. De esta forma, no solo permite acceder a las películas más conocidas y populares sino que se fomenta el descubrimiento de nuevas culturas, pudiendo buscar películas del resto del mundo que son menos conocidas por sus componentes culturales. También se puede investigar sobre la cultura de épocas anteriores, pues la aplicación incluye títulos de hace más de 100 años, pudiendo observar las características, temáticas y distintos patrones que contenían las películas antiguamente, y descubriendo los cambios que se han producido en la tendencia del cine a lo largo de los años. Además, al incluir datos de distintas fuentes de extracción internacionales, cada una más frecuentada por usuarios de diversos países y culturas, se promueve una diversidad cultural pudiendo comparar las críticas y valoraciones que aportan usuarios de distintas partes del mundo. Este ámbito fue considerado al desarrollar la aplicación, pues se podrían haber incluido únicamente películas españolas o de años recientes, pero se decidió contar con películas de cualquier país, cultura o época para hacer una aplicación más global y diversa.

La aplicación también implica un cierto impacto social. El cine es una de las actividades de ocio más practicadas en la sociedad, como se demostró al inicio de esta memoria, con numerosos usuarios consumidores de películas. Es por ello que, al mostrar diversos datos y opiniones de las películas, el usuario puede disponer de más información y tomar una decisión más acertada sobre la elección de la película a visualizar. Además, al no depender de una única fuente e incluir valoraciones de varias de ellas, se promueve una diversidad de opiniones teniendo en cuenta aquellas de todos los usuarios, y se fomenta la formación de comunidades que compartan el gusto por el cine e intercambien opiniones.

En cuanto a los Objetivos de Desarrollo Sostenible, se puede considerar que la aplicación propuesta se vincula con los siguientes:

- ODS 10, Reducción de las Desigualdades. Como ya se ha comentado, la aplicación cuenta con una interfaz intuitiva, accesible y abierta para cualquier tipo de usuario, ya sean de distinta edad, país o cultura, sin ninguna discriminación. Además, no solo muestra películas de países potencialmente económicos o de grandes productoras, sino que también abarca películas que cuentan con menos presupuesto o de culturas menos conocidas, ayudando a darles visibilidad.
- ODS 11, Ciudades y Comunidades Sostenibles. Una de las metas de este objetivo consiste en proteger y salvaguardar el patrimonio cultural, hecho al que el cine, y por tanto la aplicación propuesta, ayudan a conseguir.

Bibliografía

- [1] B. De, *API Management: An Architect's Guide to Developing and Managing APIs for Your Organization*. Berkeley, CA: Apress, 2023.
- [2] R. Lawson, *Web Scraping with Python*. Packt Publishing Ltd., 2015.
- [3] W. S. Vincent, *Django for Beginners, 5th Edition: Build Modern Web Applications with Python*. Still River Press, 2024.
- [4] Ministerio de Cultura de España, *Encuesta de Hábitos y Prácticas Culturales en España 2024-2025. Avance de Resultados 2024*. Diciembre 2024.
- [5] *The Movie Database (TMDB)*. [En línea]. Disponible: <https://www.themoviedb.org/> [Último acceso: 28 mayo 2025]
- [6] *Internet Movie Database (IMDb)*. [En línea]. Disponible: <https://www.imdb.com/> [Último acceso: 28 mayo 2025]
- [7] *FilmAffinity*. [En línea]. Disponible: <https://www.filmaffinity.com/> [Último acceso: 28 mayo 2025]
- [8] I. Velásquez, *Creating APIs for data science with plumber*. [En línea]. Disponible: <https://posit.co/blog/creating-apis-for-data-science-with-plumber/> [Último acceso: 18 mayo 2025]
- [9] M. Biehl, *RESTful API design (Vol. 3)*. API-University Press, 2016.
- [10] [En línea]. Disponible: <https://primalcom.com/web-data-scraping/> [Último acceso: 18 mayo 2025]
- [11] A. C. Luna, *Creación de páginas web: HTML 5*. ICB, SL (Interconsulting Bureau SL), 2024.
- [12] R. V. Chandra & B. S. Varanasi, *Python requests essentials*. Packt Publishing, 2015.
- [13] V. G. Nair, *Getting started with beautiful soup*. Packt Publishing Ltd., 2014.
- [14] A. Fernández, *Python 3 al descubierto*. Alfaomega Grupo Editor, 2013.
- [15] A. Chapagain, "Chapter 8: Using Selenium to Scrape the Web," en *Hands-On Web Scraping with Python: Perform advanced scraping operations using various Python libraries and tools such as Selenium, Regex, and others*. Packt Publishing Ltd., 2019.
- [16] Y. Shafranovich, *Common format and MIME type for comma-separated values (CSV) files*. 2005.
- [17] L. A. C. Santillán, M. G. Ginestà & Ó. P. Mora, *Bases de datos en MySQL*. Universitat oberta de Catalunya, 2014.
- [18] W. McKinney & P. D. Team, *Pandas-Powerful python data analysis toolkit*. Pandas-Powerful Python Data Analysis Toolkit (vol. 1625), 2012. [En línea]. Disponible: <https://pandas.pydata.org/pandas-docs/version/0.7.3/pandas.pdf> [Último acceso: 21 mayo 2025]

- [19] [En línea]. Disponible: <https://www.adobe.com/es/acrobat/resources/document-files/web-files/html.html> [Último acceso: 20 mayo 2025]
- [20] A. Durango, *Diseño Web con CSS*. IT Campus Academy, 2014.
- [21] [En línea]. Disponible: <https://hektorprofe.github.io/django/web-personal/patron-mvt-modelo-vista-template/> [Último acceso: 21 mayo 2025]
- [22] *RottenTomatoes*. [En línea]. Disponible: <https://www.rottentomatoes.com/> [Último acceso: 22 mayo 2025]
- [23] *SensaCine*. [En línea]. Disponible: <https://www.sensacine.com/> [Último acceso: 22 mayo 2025]

Anexo

Informe de Originalidad

SERGIO REDONDO CARRACEDO

TFG_SERGIO_REDONDO_CARRACEDO.pdf

Turnitin Memoria Final
TFG ETSIINF (Moodle PP)
Universidad Politecnica de Madrid

Detalles del documento

Identificador de la entrega
trn:oid::1:3267771752

Fecha de entrega
3 jun 2025, 5:00 p.m. GMT+2

Fecha de descarga
3 jun 2025, 5:07 p.m. GMT+2

Nombre de archivo
8301_SERGIO_REDONDO_CARRACEDO_TFG_SERGIO_REDONDO_CARRACEDO_83714_727564795.pdf

Tamaño de archivo
1.9 MB

54 Páginas
16.260 Palabras
87.227 Caracteres

2% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Top Sources

- 0% Internet sources
- 0% Publications
- 2% Submitted works (Student Papers)

Este documento esta firmado por



Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
Fecha/Hora	Tue Jun 03 17:30:49 CEST 2025
Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
Numero de Serie	561
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)