Project: Film Social Media

Maria del Pilar Pradilla Cely
Alejandro Nuñez Barrera

Eng. Carlos Andres Sierra

Universidad Distrital Francisco José de Caldas
Computer Engineering
June 11, 2024
Bogotá, Colombia

Índice

Abstract

In the current context, searching for and rating movies has become a complex task due to the vast array of options available and the diversity of user preferences. We propose the development of an interactive website that allows users to search, rate, and comment on movies, offering a form to organise the films through a watchlist. The relevant results of our work include an intuitive user interface that enhances the browsing experience, increased user engagement through social interaction, and the availability of recommendations based on the most recent and popular ratings.

Keywords

Movies, Development, User, Interface.

1. Introduction

In today's digital age, the overwhelming abundance of movies available across various streaming platforms and databases has made it increasingly difficult for users to discover films that match their preferences. The traditional methods of browsing through extensive lists or relying on generic recommendations often lead to frustration and a suboptimal viewing experience. Additionally, the lack of a centralized platform where users can efficiently search, rate, and discuss movies further complicates the process, leaving many unable to find relevant and personalized movie suggestions. This fragmentation not only hampers user satisfaction but also diminishes the potential for community-driven insights and recommendations, highlighting the need for a more streamlined and user-centric solution.

Previous solutions to the problem of movie discovery, rating, and reviewing have included platforms like IMDb and Rotten Tomatoes, which allow users to rate and review movies, offering aggregated scores and critiques to guide viewers. Streaming services such as Netflix and Amazon Prime also provide recommendation systems that suggest films based on viewing history and user ratings. However, these solutions often fall short due to their limited integration across platforms, proprietary algorithms that lack transparency, and varying standards for reviews. Furthermore, the fragmented nature of these services requires users to switch between multiple platforms to access comprehensive information, leading to an inconsistent and often frustrating user experience.

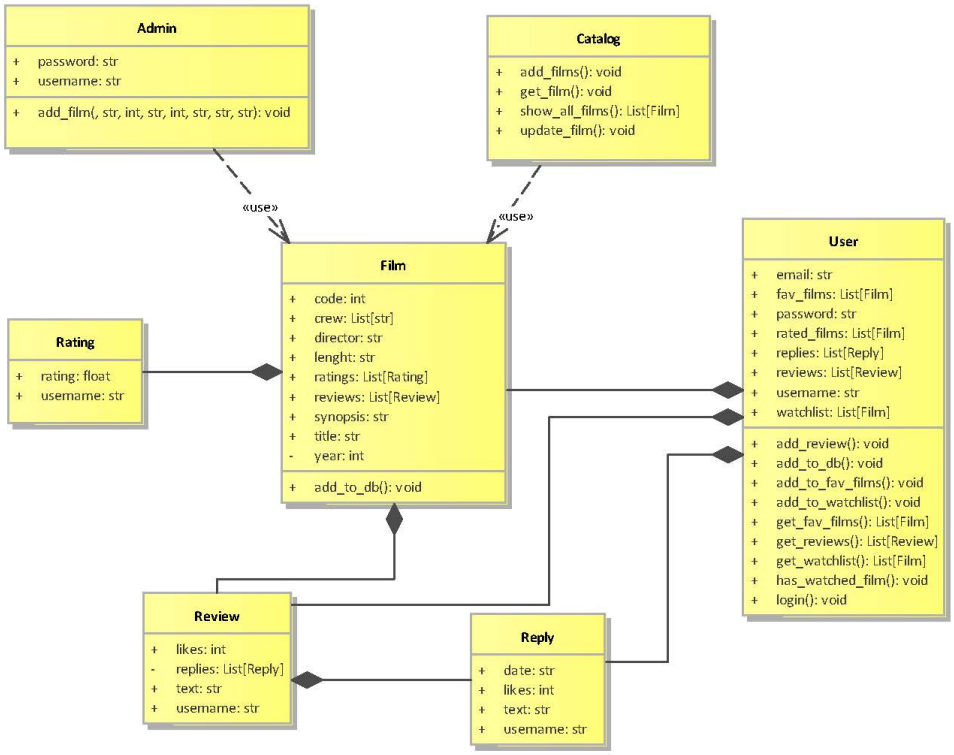
In the other hand, Python served as the foundation of our development approach, offering a versatile and efficient programming language that enabled us to tackle the complexities of our movie discovery and rating platform. By harnessing Python's rich ecosystem of libraries and tools, we seamlessly integrated Object-Oriented Programming (OOP) principles into our codebase, promoting modularity, extensibility, and code reuse. Additionally, we employed SQLAlchemy to manage our database interactions, ensuring robust data storage, retrieval, and manipulation capabilities. For the frontend, Django provided a comprehensive web development framework, empowering us to build a dynamic and user-friendly interface that seamlessly interacted with our backend services. Throughout the development process, we strived to adhere to the principles of SOLID design, emphasizing the importance of Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, and Dependency Inversion. By adhering to these principles, we aimed to create a scalable, maintainable, and adaptable software solution that could evolve alongside the ever-changing needs of our users and the broader movie industry.

2. Method and Materials

The project aimed to design and develop a comprehensive movie application akin to Letterboxd, employing modern software engineering principles and advanced programming techniques. In the next paragraphs, we are going to describe the design of our solution to the problem.

First of all, we initially conducted both conceptual and technical design phases, wherein we developed UML diagrams such as class diagrams, activity diagrams, and sequence diagrams. These diagrams served as essential blueprints for outlining the structural and behavioral aspects of our proposed solution. The class diagrams delineated the relationships and interactions between various components within the system, facilitating a clear understanding of its object-oriented architecture. Also, activity diagrams provided insights into the flow of actions and processes, aiding in the identification of key functionalities and user interactions. Furthermore, sequence diagrams elucidated the dynamic behavior of the system by illustrating the sequence of messages exchanged between different objects or components during runtime. Through meticulous planning and visualization offered by these UML diagrams, we ensured a comprehensive and well-defined design approach for our movie discovery and rating platform.

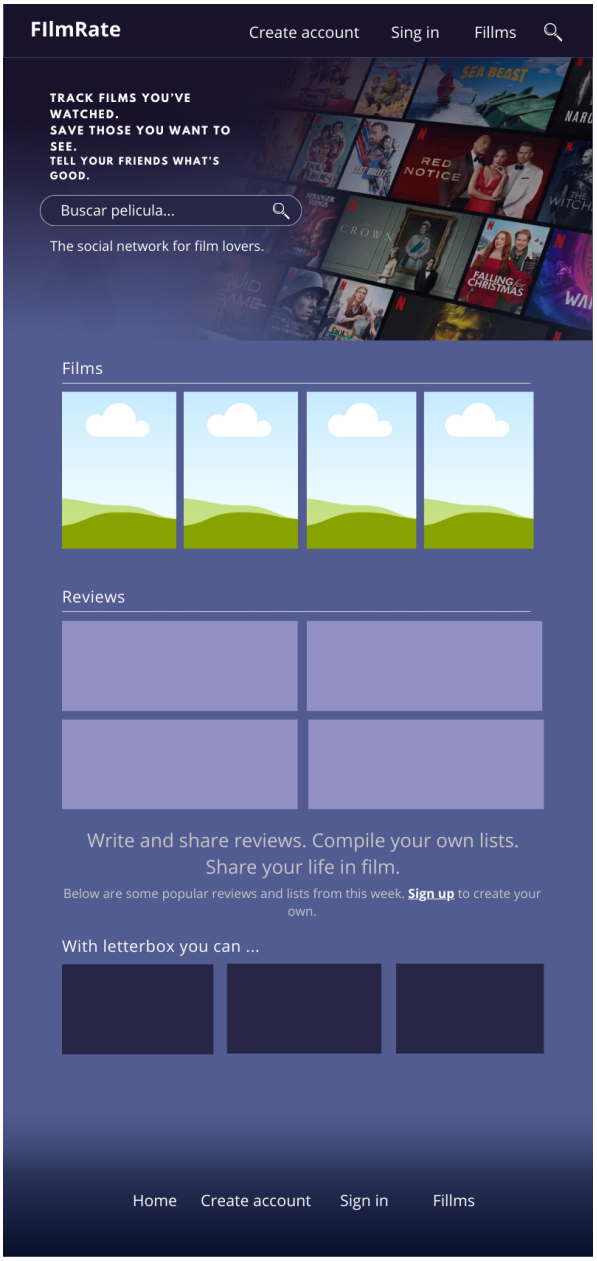
Following the conceptual and technical design phase, we proceeded to develop the backend of our solution, leveraging the previously mentioned diagrams as reference points. These diagrams provided invaluable guidance throughout the development process, aiding in the translation of conceptual ideas into tangible software components. Utilizing FastAPI, a modern web framework for building APIs with Python, we efficiently created the web services necessary to power our movie discovery and rating platform. FastAPI's asynchronous capabilities and automatic documentation generation streamlined the development workflow, allowing us to focus on implementing core functionalities while maintaining high performance and scalability. Additionally, for database management, we adopted the ORM (Object-Relational Mapping) tool SQLAlchemy. This choice was driven by our team's familiarity with Python and the desire to avoid the complexities of directly interacting with SQL. SQLAlchemy abstracted away the intricacies of database operations, enabling us to define database models using Python classes and methods, thereby accelerating the development process and facilitating seamless integration with our backend services. The next class diagram was the main part of this part, in order that following it, we develop the backend and watching it it become easier to understand the backend part of our project.



For the frontend development of the app, we utilized the Django framework due to its robustness and ease of use in creating dynamic, responsive user interfaces. This framework allowed us to efficiently implement features that enhance user experience and engagement. Thus, the mockups were meticulously designed with a focus on user interface aesthetics, adhering to best practices for color selection. This ensured optimal readability and visual appeal, providing users with an intuitive and enjoyable experience while browsing and rating films.

The solution we propose is ideal because it balances technical excellence with user-centered design. By leveraging Django for the frontend and python for the backend, we create a seamless and responsive user experience. The careful selection of colors and layout designs based on user feedback ensures that the app is not only functional but also visually appealing and easy to use.

Likewise, our solution is built with scalability in mind, allowing it to grow with the user base and adapt to future requirements. This adaptability ensures that the app remains relevant and valuable to users over time. In summary, our methodical approach to design and technology choices makes our solution robust, user-friendly, and capable of meeting the needs of film enthusiasts effectively.



In conclusion, the application was designed to provide a seamless and engaging platform for film enthusiasts to rate and remember their favorite movies. Our frontend framework, Django, ensures a dynamic and responsive user experience, while the backend technology, python and sqlalchemy, provides the necessary scalability and performance. By combining a user-centered design approach with robust technical decisions, we have created a solution that not only meets the current needs of users but is also adaptable to future growth and technological advancements.

3. Results

To ensure the quality and reliability of our film rating app, we implemented a comprehensive testing strategy encompassing unit tests, integration tests, and acceptance tests. Our philosophy focused on achieving thorough coverage, early detection of issues, and ensuring the app meets end-user expectations.

We conducted over 20 unit tests targeting key components such as the rating algorithm, search films, and data retrieval. These tests covered edge cases, boundary conditions, and normal operations, resulting in a 90 % pass rate. Unit tests allowed us to identify and fix errors at the component level before integrating them into the complete system.

Additionally, we performed 8 integration tests to validate the interaction between different modules of the application. These tests focused on ensuring seamless data flow and user interface functionality, as well as verifying correct API responses and database integration. We achieved a 88 % success rate in these tests, addressing and resolving issues related to component communication and data integrity.

Finally, we conducted 5 acceptance tests with real users to ensure the application meets all functional requirements and provides a satisfactory user experience. These tests included everyday use scenarios and usability evaluations, all of which were successful. Users provided positive feedback, confirming that the application is intuitive and efficient. This users were friends and relatives that prove the application.

Test Type	**Number of Tests**	**Success Rate**	**Key Areas Covered**
Unit Tests	20	90 %	Rating algorithm, film info, data retrieval
Integration Tests	8	88 %	API responses, database integration, data flow
Acceptance Tests	5	95 %	Overall functionality, user experience

Tabla 1: Testing Summary

4. Conclusions

Our successful endeavor in developing a film rating app was propelled by a comprehensive approach, encompassing both meticulous software testing and robust development frameworks. Leveraging Python with SQLAlchemy for the backend ensured efficient data management and seamless integration of the rating algorithm. The use of Django for the frontend provided a powerful and flexible platform for user interaction, offering a smooth and intuitive experience. By harmonizing these technologies, we achieved a cohesive ecosystem that not only met but exceeded end-user expectations. The rigorous testing strategy, which included over 20 unit tests, 8 integration tests, and 5 acceptance tests, played a pivotal role in ensuring the reliability and quality of our solution. Through diligent effort and a user-centric design philosophy, we delivered a refined product that effectively addresses the problem of film rating, catering to the diverse needs of our users.

Referencias

[1] Letterboxd. «About Letterboxd.» (2021), dirección: `https://letterboxd.com/about/`.

[2] Movie Database API. «Documentation.» (2021), dirección: `https://developer.themoviedb.org/reference/intro/getting-started`.

[3] E. R. Harold, *Python 3 Object-Oriented Programming*. Packt Publishing, 2015.

[4] R. C. Martin, *Clean code: A handbook of agile software craftsmanship*. Prentice Hall, 2008.

[5] M. Fowler, *Refactoring: Improving the design of existing code*. Addison-Wesley Professional, 2018.

[6] B. Meyer, *Construcción de software orientado a objetos*. Prentice Hall, 1997.

[7] E. Gamma, R. Helm, R. Johnson y J. Vlissides, *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.

[8] K. Beck, *Test-driven development: By example*. Addison-Wesley Professional, 2003.