Musicat: Development of a streaming music website

Tito Alejandro Burbano Plazas

Universidad Distrital Francisco José de Caldas

Abstract:
This work describes the development of a streaming music website, taking advantage of the knowledge acquired in the programming course. The project aims to provide a platform where users can listen to music online. The implementation was done using technologies like HTML, CSS, and JavaScript for the frontend, and Python along with FastApi for the backend. The use of responsive design principles is highlighted to ensure a consistent user experience across different devices. This project not only demonstrates the practical application of concepts learned in class, but also addresses common challenges in web application development, such as efficient database management and server performance optimization.

Keywords:
Music, Web development, Proyect

Introduction:

In the digital age, streaming music has become one of the most popular ways to consume music content. Users expect robust platforms that not only allow them to access a vast library of songs but also personalize their music experience. This project aims to develop a streaming music website using the knowledge acquired in the Web Development course, with special emphasis on object-oriented programming (OOP).
Object-oriented programming is a paradigm that organizes software into units called "objects," which are instances of "classes." This model is particularly suitable for developing complex web applications due to several key reasons:
First, OOP allows for better code organization and modularity. By dividing the system into objects with clearly defined responsibilities, it is easier to maintain and expand the system. For example, on a music streaming website, we can have classes like 'User', 'Song', 'PlayList' etc, each with specific attributes and methods. This structure makes it easy to add new functionality, without having to significantly restructure existing code.
Second, OOP promotes code reuse through inheritance and polymorphism. This is especially useful in a streaming project, where many objects share common characteristics.
The OOP model also makes it easier to implement design patterns. These patterns not only improve development efficiency but also the scalability and maintainability of the system.

Methods:
Use the tools that were seen in class such as; Python due to its wide adoption in web development and its powerful set of libraries.
FastAPI was used for the framework due to ease of use.
During the development of the backend, design principles were applied. Each component was designed to be independent and reusable, making it easy to scale the system as new features and functionality were added.
PostgreSQL was used as the database engine, due to its reputation for reliability and scalability in production environments.
Postman became an integral part of the development process thanks to its ability to simplify and automate web service testing.
Before starting the backend implementation, time was spent designing and planning the classes using UML diagrams. This approach allowed me to have a clear understanding of the system architecture and the relationships between different entities before I started writing code.
The UML diagrams provided a visual representation of the system structure, which facilitated communication between team members and helped avoid potential misunderstandings and design errors later in the development process.
Additionally, pre-designing classes made it possible to identify and address potential design issues before they became real problems. This resulted in a smoother and more efficient implementation of the backend and helped minimize development time and associated costs.

Experiments:
All methods were rigorously tested to ensure proper functionality. A comprehensive set of tests was designed to verify the behavior of each method in various scenarios, covering edge cases and expected results. These tests, implemented using frameworks such as Pytest, were intended to validate that the backend components were working as intended, delivering accurate results.
After the creation of the web services, an additional layer of testing was introduced using Postman, as mentioned above. Postman's intuitive interface and robust testing capabilities enabled extensive validation of web services functionality across different endpoints. By creating test collections spanning various scenarios and API endpoints, I was able to systematically verify the behavior of services, ensuring that they met specifications and provided expected responses.

Results:
It is expected to obtain a project that meets the client's expectations effectively and efficiently. This involves delivering a functional, robust and scalable product that meets specific customer requirements. Additionally, we seek to establish efficient development processes that enable the timely delivery of new features and updates, as well as validate our design and architectural decisions throughout the process.