Aruppi, Japan in one place

Xiomara Salome Arias Arias-20222020028, Member 1
Carlos Andres Celis Herrera-20222020051, Member 2
Universidad Distrital
Professor: Carlos Andrés Sierra Virgüez
Bogotá, Colombia
Date: June 11, 2024

ABSTRACT– Japanese culture has become a worldwide interest, so many of those interested want to know a little more about it, especially anime, news and music. However, access to this type of content is often limited or has some problems, such as publicity. In contrast to this, Aruppi is an application that looks to collect all the Japanese culture in an application the already mentioned contents. To implement this, a rigorous planning and documentation is required, making use of different tools and methods, such as design patterns, which in this context the following were used: "factory", "state", "strategy" and "facade" for the development of the app. This work shows all the diagrams made to understand the operation of the Aruppi backend, and how is the data flow, activities, states and sequence between objects and classes; in addition, a mapping of the database is made, with their respective entities and relationships. of classes that synthesizes all the logic of the platform. The above, will allow a correct implementation in the application, looking for a solution to the problem posed, using the available technologies.

I. INTRODUCTION

The world has become a space where there are a lot of cultures, with different traditions and customs. In that context, Japanese Culture has been a social interesting around the world, because that culture stands out for its rich historical depth, its vibrant traditions, and its global influence on art, music and literature.

Aruppi is a tool that makes a connection between users and Japanese culture, through three main types of content which are news, anime and music. In these types of contents there is a clear example of their traditions, characteristic places, gastronomy, etc. Another important aspect of Japanese culture is the cinema, where interesting plots and excellent film production stand out, however, it has been shown that the most demanded is the animated cinema, that is to say, anime, which has been successful worldwide; an interesting fact is that all the information given above is reflected in its cinema, which allows the world to know a little more about Japanese culture, customs and point of view on life.

Aruppi was built with the implementation of development patterns in the search to fulfill its requirements in a better way, the following patterns were used: factory, strategy, state and facade. These are divided into three main groups: creative, structural and behavioral. Aruppi has to provide solutions to a series of problems that present similar structures to those that the patterns seek to solve.

Likewise, the Python FastAPI web framework was used to build fast and efficient RESTful APIs. That is, web services through which Aruppi can deploy its services. In these services deployed with FastAPI we can obtain information from the user and also display data without needing to know the internal details of Aruppi's implementation. It should be noted that these services can be connected to other systems not only to users through the web.

Aruppi is deployed in a container with the help of Docker, packaging all its dependencies. So that, it can run optimally on another machine or environment; such as the cloud. In this way, Aruppi presents an agile deployment and a quick response to user demands. It is of utmost importance that the database is deployed in this same container, since it is an essential part of the project. All Aruppi's data types are stored in it.

In this project the stakeholders are principally the final users, who are a community known particularly as "otakus", who are the main consumers of Japanese cultural content. The term otaku in Japan is used to refer to a fan of anything, while outside Japan it is used to refer to all those who have a great interest in manga and anime of Japanese origin. However, they are not the only ones to whom the application is designed for, because, a great part of teenagers and children are interested in the consume of this content, without being otaku. That is to say, the target audience of this project will be those interested in Japanese culture, taking this into account, there are features or functionalities that they want to do in the application, that is, the user stories, these are:

- As an administrator, I want to add anime, so what to have updated content.
- As an administrator, I want to add news, so what to update the content
- As an administrator, I want to remove news, so what to have updated information.

- As a user, I want to pause a station, so what to stop listening music.
- As a user, I want to play a station, so what to listen to music.
- As a user, I want to watch anime, so what I hang out.
- As a user, I want listen Japanese stations, so what I learn Japanese music and know facts of their country.
- As a user, I want know the news about Japanese culture, so what I keep me informed.
- As a user, I want add my favorite content, so what I categorize them according to my preferences.
- As a user, I want see my history of episodes, so what I don't lose the thread of it.
- As a user, I want add content to the list queue, so what I can see it after.

These allow to synthesize the processes and to understand how they work, in order to perform a correct solution of the problem.

II. METHODS AND MATERIALS

The tools that will be used in the construction of this project are: Visual Studio code, a code editor through which we will program with the python programming language, docker to facilitate the implementation and execution of the project, FastAPI to deploy web services, Lucidchart and draw.io for the creation of the different conceptual models of the project and finally Github through which deliveries will be made and relevant information will be shared with others. Finally, phpMyAdmin for creating and testing SQL databases..

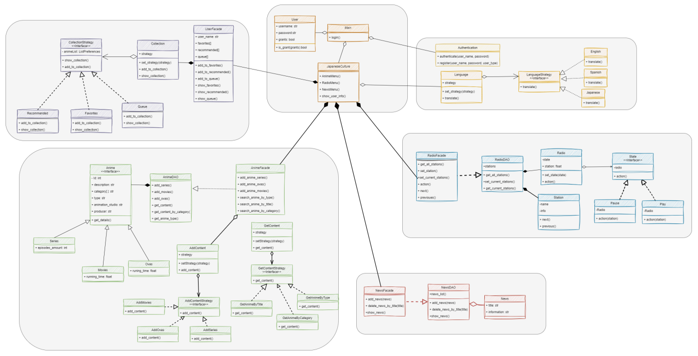
First, it is important to understand what design patterns fit together to solve the problems of this project. The creative design pattern "factory" is implemented in the creation of anime type objects, since anime has three types: movies, series and ovas. On the other hand, the behavioral design patterns "strategy" and "state" were implemented. The first one in order to solve the searches in the anime content, since in this content searches are made by title, category and type. The second to control the two states that a radio station has when playing music, "Play" and "Pause". Finally, we implemented the structural design pattern "facade", since the backend divides the responsibilities by subsystems, this pattern helps us to access what the user needs without the need to know the code.

Figure 1 shows the Aruppi class diagram, highlighting the division of the project by subsystems. In each of these there is a different type of content, i.e., each one manages and is responsible for a content, providing an app easier to maintain and scalable.

UML (Unified Modeling Language) is a standard modeling language used to visualize, specify, build and document software systems, which is very useful for understanding Aruppi's processes. The diagrams used will be presented below.

One of the methods used corresponds to the UML diagrams

Figure 1: Class Diagram

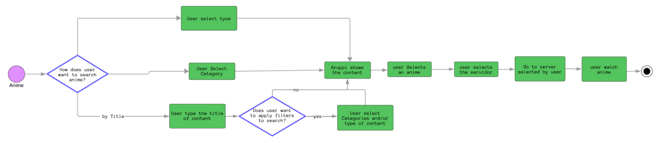


that allow the analysis of Aruupi's functionality and processes.

Activity Diagrams: is used to visualize and specify the dynamic behavior of Aruppi.In this case, there are various activities that can be performed in the application, so a diagram was created for each activity. The figures used are the circle, which represents the beginning of a process, while the end is a circle with a filler; in the same time, the arrows show the flow of the different activities or processes, where decisions or different ways to take can be found, which are joined to a rhombus.

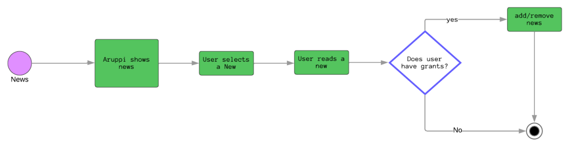
Figure 2, shows the process flow that occurs when the user wants to watch Anime, where it can be seen that three types of search can be performed, by title, category or type, Aruppi shows the content found, there the user can choose the server to watch it, then it will redirect to this one

Figure 2: Anime Activity Diagram


Anime
How does user want to search anime?
User select tyoe
User Select Category
by Title
User type the title of content
Does user want to apply filters to search?
no
yes
User select Categories and/or type of content
Aruppi shows the content
user Selects an anime
user selects the servidor
Go to server selected by user
user watch anime

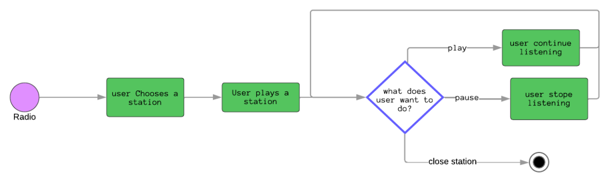
In the activity diagram shown in Figure 3, which corresponds to News, it can be seen that the process flow is given with the available news, the user who has grants is considered admin and will be able to add and remove news.

Figure 3: News Activity Diagram



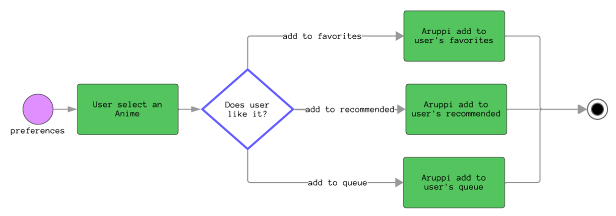
On the radio side, the application offers different stations, so the user chooses one and starts listening, he/she can pause, play or close the station, then the process is finished, as shown in figure 54

Figure 4: Radio Activity Diagram



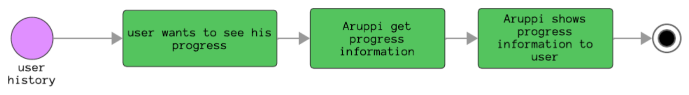
When the user wants to add an Anime to favorites, to the queue or to recommended, the process that follows, according to figure 5, is to select an anime and select to which list to add it, Aruppi will save all the user's preferences.

Figure 5: Preferences Activity Diagram



The last activity diagram is the one in figure 6, which shows the user's history, where it can be seen that if the user wants to know his history, Aruppi will obtain his history and will show it to the user.

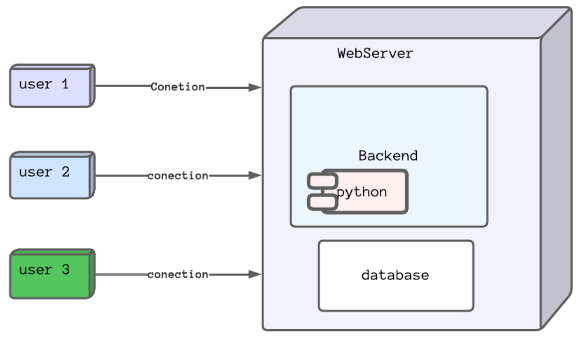
Figure 6: User History Activity Diagram


user history
user wants to see his progress
Aruppi get progress information
Aruppi shows progress information to user

Deployment Diagram: The project will be executed locally, that is to say, in the computer where the program is installed, in Figure 7 the backend is observed in the same space as the database, this in order to contemplate a database inside the machine and not external to the computer, it is hosted and it is only possible to access from the same program located in the same machine. This in order not to use an external service. In the same way, the python component is inside the backend making allusion to the construction of this by means of this programminglanguage that today is one of the most used.

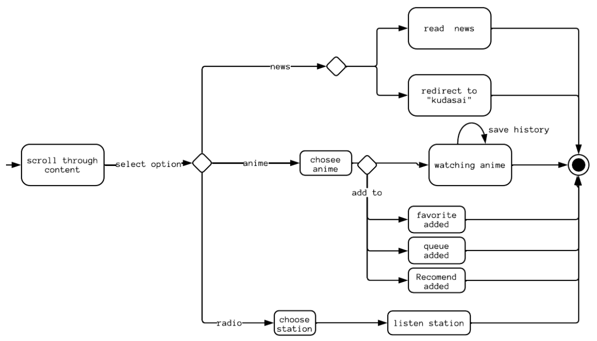
Finally, the diagram gives us an idea of the relationship between software and hardware. In this project the hardware is minimal and the execution of the software is reinforced with tools like docker.

Figure 7: Deployment Diagram



State Diagram: It is possible to observe the different states through which the "user" object is involved in the use of the project.
As shown in Figure 8, the user, after inspecting Arupi, is ready to select one of the four options. By closing the initial state, the user can start four other different events within Arupi. It is worth mentioning one of the events with more work within Arupi and is the option of anime, since, within this there is a special event that is to add content that is divided into three states: add to favorites, queue and recommended.

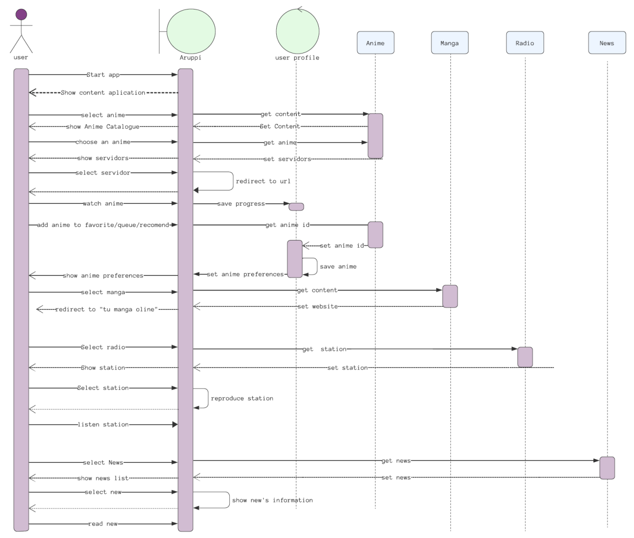
Figure 8: State Diagram



Sequence Diagram: The sequence diagram, allows to give a solution focused on the lifelines together with the objects that exist together, along with the messages exchanged between them. This allows to know the requirements and functionality that Aruppi must have.

In the diagram in Figure 9, the representation of classes,

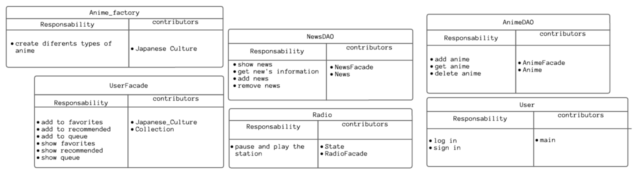
Figure 9: Sequence Diagram



objects and users is shown; in this case, Aruppi represents a boundary, because it is the interface with the user will interact; on the other hand, the user profile is represented as a control class, because it has a management of the user's information. The different activation boxes are also visible on the lifelines of each object or class, where the user will always be active while inside the application as well as the interface, while the content objects will only have an activation once the user selects an option related to it. The messages that describe the process show how the content interacts with the user, and how the user's required information will be saved.

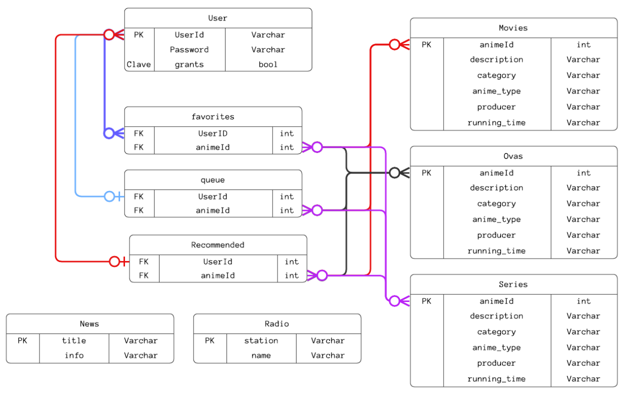
CRC Cards: In the context of Arupi we use these cards to represent the objects and classes that make up this project. Focused on the responsibilities and collaborations of each one of them. In order to have a clearer idea of the system.

Figure 10: CRC Cards



ER Diagram: Is a tool used in database design to model the relationships between the entities that compose Aruppi. Figure 11 shows the Aruppi entity-relationship diagram, where the different tables of the database are displayed, with their respective names, fields, type of data and the primary and foreign key of each one of them is specified. In general, it is known that the primary key of the content is the id of

Figure 11: ER Diagram



the content. As the user has a profile, in which only its id is taken into account, together with its favorites, queue, and recommended list. Which are associated to another database, in which an association with the primary and foreign keys can be made. For example, the Favorites entity is associated with a user's list, which may be empty or contain many, but there is only one of these associated with each user. This is how the other bases work, the diagram shows the type of relationships they have, and those that have no relationship, but will serve to display the contents of the application.

III. EXPERIMENTS AND RESULTS

- **Implementation of the API with FastAPI** FastAPI was used to develop the API due to its efficiency and ease of use for creating web services. Definition of multiple endpoints for CRUD (Create, Read, Update, Delete) operations related to series, movies, and OVAs. Use of Pydantic models for validating incoming data. The API was successfully developed, allowing the management of anime content. The endpoints worked correctly during local testing with Uvicorn.
- **Connection to the MySQL Database** A MySQL database was configured to store information about series, movies, and OVAs. The tables were successfully created in the database. Persistent connection issues were encountered and not fully resolved.
- **Generation of Test Data with Faker** The Faker library was used to generate fake data during development and API testing. Generation of test data for different data models. Use of this data in automated tests to simulate various conditions. Test data was successfully generated and effectively used in testing. This allowed validation of the correct functionality of the API endpoints.
- **Automated Testing with PyTest** PyTest was employed to write and run automated tests for the API.
  Creation of unit and integration tests for the API endpoints. Execution of tests and analysis of the results. Tests

demonstrated that the endpoints worked correctly with the test data. Several errors were identified and fixed during the testing process.
- **Deployment with Docker** An attempt was made to deploy the application and the MySQL database using Docker. Creation of a Dockerfile and configuration of Docker Compose. Attempted deployment of the application and the database. Deployment in Docker was unsuccessful due to unresolved errors. Errors were investigated, but not resolved within the available time.

IV. CONCLUSIONS

To conclude, these are some of the conclusions that were obtained from the design of the Arupi backend:

- In this last installment we implemented creative, structural and behavioral patterns, which helped to solve common problems in the development of the project's code. In the creation of similar objects in the way they are created, in the organization and cohesion of the project and finally in the behavior of the objects within it..
- Implementing web services is essential for enhancing project flexibility and preparing for future collaborative projects, whether in front-end or back-end development. This approach not only increases the adaptability of the project but also builds crucial teamwork skills, fostering efficient collaboration in various software development roles
- Adhering to best practices in software development, including robust coding standards, efficient version control management, and the mitigation of code smells, is paramount for ensuring the reliability, maintainability, and scalability of software projects. By consistently applying these practices, teams can streamline collaboration, enhance code readability, and minimize the risk of errors or bugs.

V. BIBLIOGRAPHY

[1] Arisín, J. and Rodriguez, P. (2023). Japon Secreto. Available at: https://japon-secreto.com/cultura/. Accessed on: April 4, 2024.

[2] Lucidchart. (2024). State Machine Diagram Tutorial. Available at: https://www.lucidchart.com/pages/uml-state-machine-diagram. Accessed on: April 5, 2024.

[3] Lucidchart. (2024). UML Sequence Diagram Tutorial. Available at: https://www.lucidchart.com/pages/uml-sequence-diagram. Accessed on: April 5, 2024.

[4] Lucidchart. (2024). UML Class Diagram Tutorial. Available at:https://www.lucidchart.com/pages/uml-class-diagram. Accessed on: April 5, 2024.

[5] Lucidchart. (2024). UML Activity Diagram Tutorial. Available at: https://www.lucidchart.com/pages/uml-activity-diagram UML. Accessed on: April 5, 2024.

[6] Lucidchart. (2024). Deployment Diagram Tutorial. Available at: https://www.lucidchart.com/pages/uml-deployment-diagram. Accessed on: April 7, 2024.

[7] Lucidchart. (2024). What is an Entity Relationship Diagram (ERD). Available at: https://www.lucidchart.com/pages/er-diagrams. Accessed on: April 7, 2024.