UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

SOFTWARE-MODELING

JUAN DIEGO LOZADA GONZÁLEZ

EDISON ALVAREZ

20222020014

20222020043

CARLOS ANDRES VIRGUEZ

PARTIAL PROJECT

BOGOTA D.C

2024

ÍNDICE

Introduction

Stakeholders

The stakeholders are the people who will be participating in an active way on the app, impulsing the app to be better, those are:
1) Users
The users improve the app, making the app more visible, generating posts that serve to the rest of the community, making more attractive the ”boards”, where the people get motivation with ideas found it on it. The users can upload and create boards and pins as they want (respecting rules)
2) Manager
The manager will be checking the correct function of the app-user, where the users must respect the rules that the app contain, for that reason in the business rules on Readme from the folder Backend, show the rules that all users must follow. Following those rules, the app will be for everyone

Business Model

The main concept of our app is to help users discover photos shared by others on various boards. Through these visual collections, individuals can find inspiration and ideas for their own projects and plans. The purpose of creating this app is to assist people who encounter difficulties in generating ideas, experiencing moments where their minds go blank Imagine having all communities gathered in one place, where users can easily explore and discover photos relevant to their interests. Whether it’s cooking, art, fashion, home decoration, or any other passion, our platform ensures users can find images with their preferences. Our app’s primary goal is to offer a seamless experience, allowing users to dive into a world of visual inspiration with just a few clicks. By providing access to a vast collection of high-quality images, we aim to reignite users’ creative spark and empower them to bring their ideas to life.

Tools to Use

As a main programming lenguage will be python, due to its simplification and its libraries who can complement the app successfully, such as Pandas, Numpy and SQLalchemy. The last one is one of the most interesting libraries we will use, SQLalchemy is an ORM (Object Relational Mapper), which serves to map and convert data between oriented objects and relational data. We are checking if it is necessary to use another library such as Faker, which generates databases with fake information with the purpose of filling the table and test properly the funcionality of the program. Meanwhile, we use some tools we manage previously due to database foundations, so, we use PowerDesign to make the Diagram ER and generate a file SQL for create the tables and relationships on PostgreSQL. We test the tables making queries of INSERT

User Stories

The user stories help to know what are the objectives that people want for the app.

- As a user, I want to see boards, so what I can get inspiration.
- As a user, I want upload pins, so what I can share my art.
- As a user, I want to have an account, so what I can save my boards.

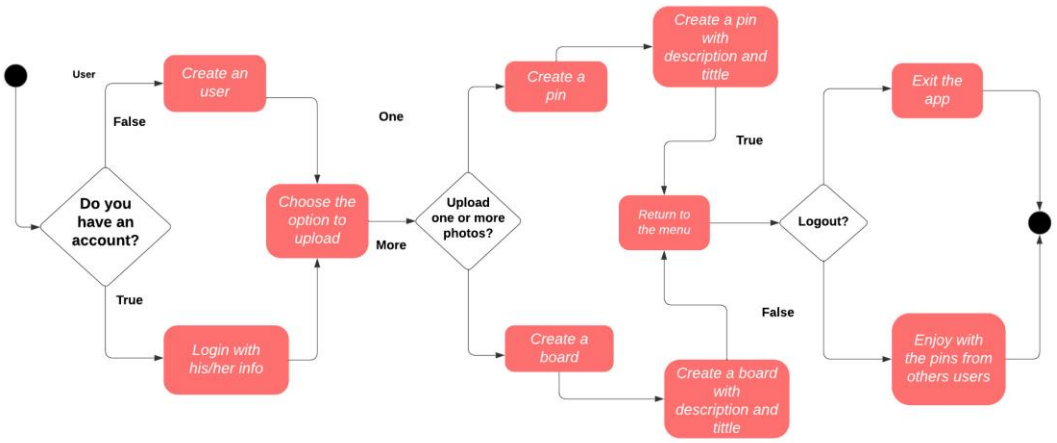
- As a user, I want to see the catalogs, so what I can see other people's work.
- As a manager, I want to see all the users have the app, so what I can make analysis about it.
- As a manager, I want to see boards, so what I can decide to delete depending on business rules.
- As a manager, I want to have a special account, so I can have special permissions such as deleting or viewing the content of users.

Diagrams

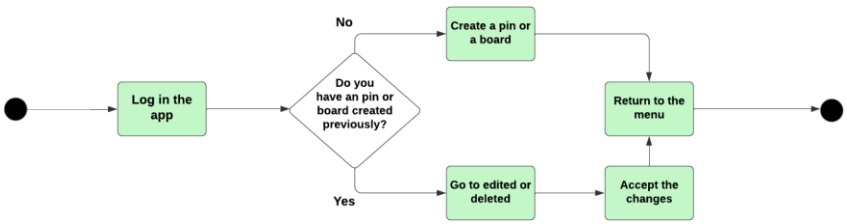
Activity Diagram

The diagrams from the different processes are:

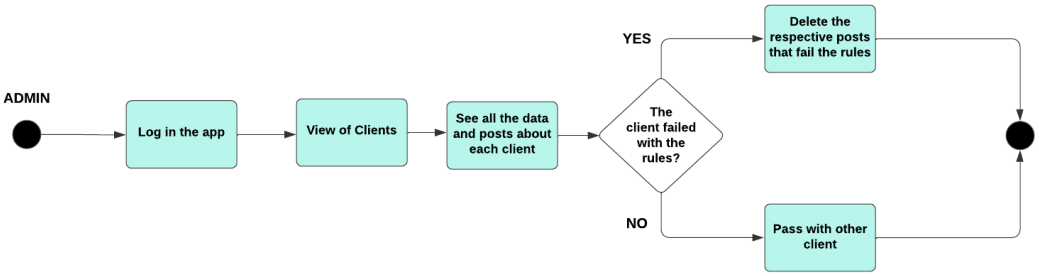
- 1) The user login and upload a pin or create a board



- 2) The user wants to delete a photo that upload previously



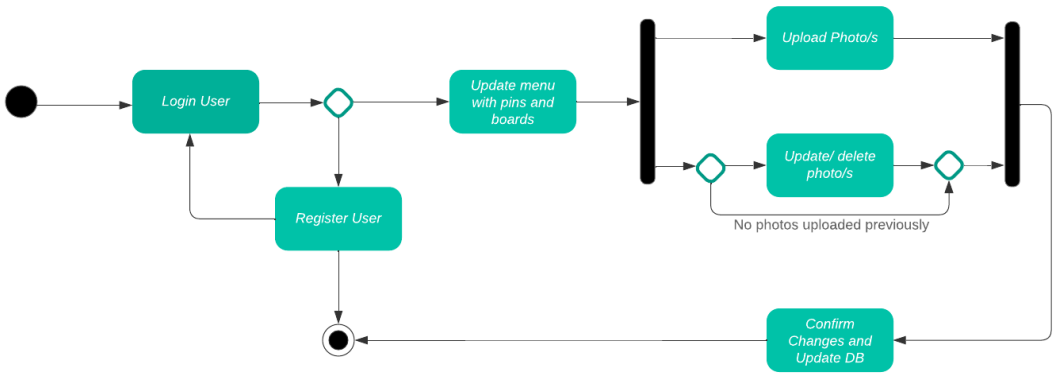
- 3) The manager will see the clients of the app



The Activity Diagram complete of the users is the next one:

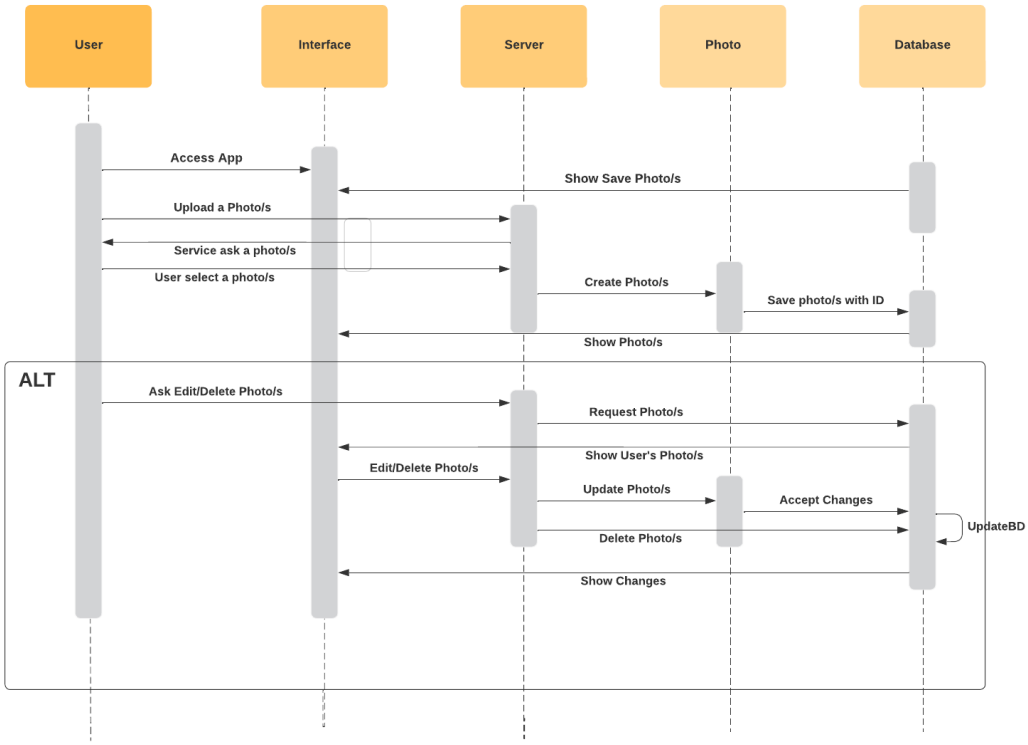
Diagram State:

Show the different states that the users pass through the app

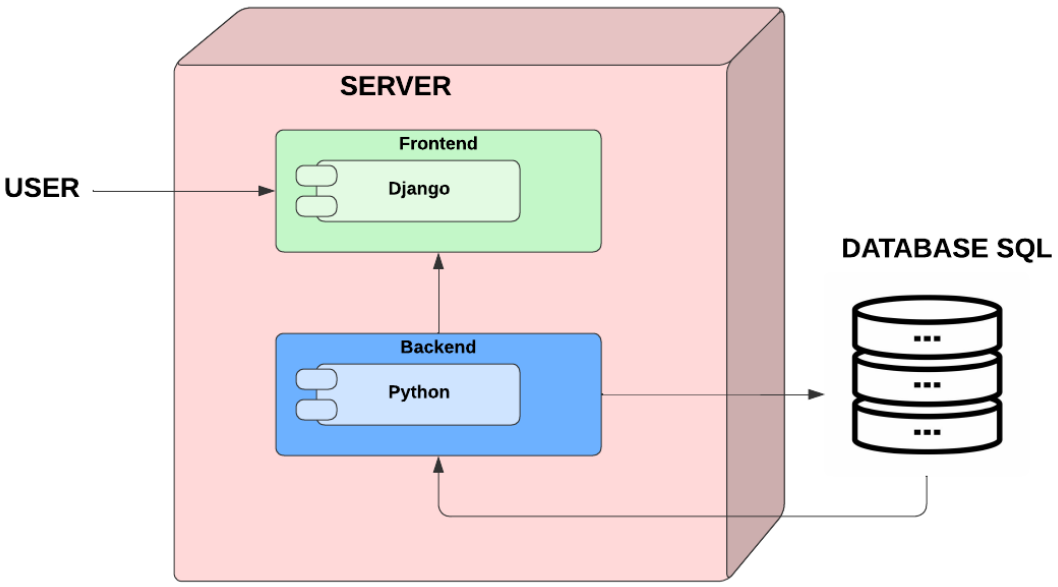


Sequence Diagram

It's necessary highlight that photo is the same Pin, but Photo is for the people understand easier the business model



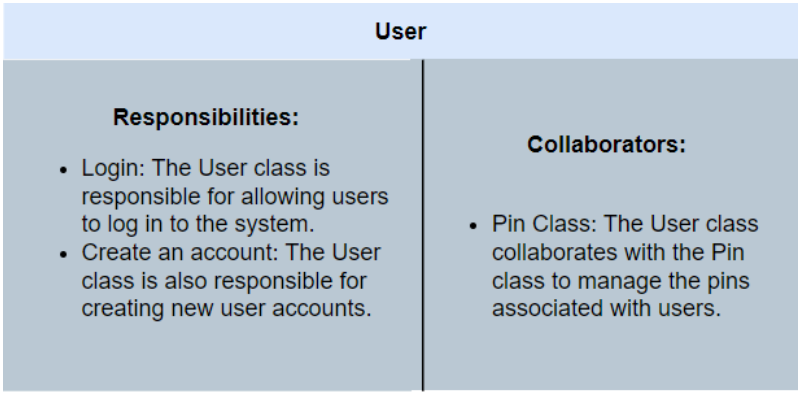
Deployment Diagram



CRC cards

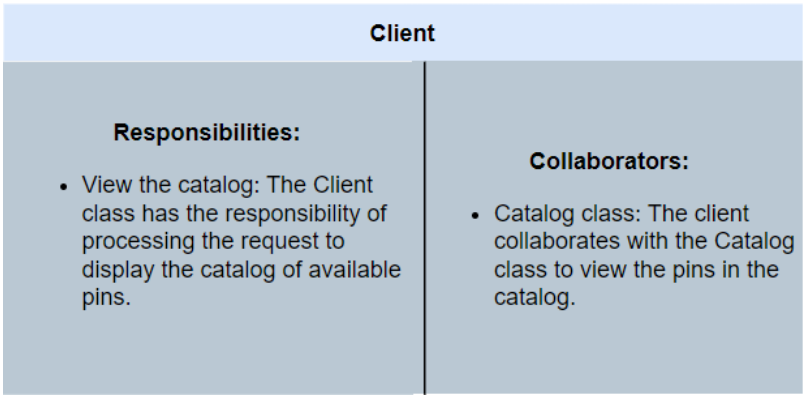
User Class

The User class performs an important role in the system, taking care of the tasks of authentication and creation of user accounts. Its design is based on the principle of encapsulation, keeping its responsibilities clear and promoting low coupling in the system. Collaboration with the Pin class is also important, since this relationship allows efficient management of the pins associated with each user account, ensuring a modular and flexible design.



Client class

The Client class, which inherits from User, focuses on providing a specific functionality or interface for the end users of the system. The main responsibility of this class is the presentation of the catalog's pin catalogue. Collaboration with the Pin and Catalog classes guarantees efficient access and proper presentation of pins, maintaining a clear separation of responsibilities and facilitating the management of information related to the pins.



Manager class:

The Manager class, which also inherits from User, has additional privileges for system management. This is responsible for extending user functionalities with specific operations for administrators. The responsibility of viewing the list of registered users is assigned to the

administrator as is the ability to modify the pin from catalog. Collaboration with the Pin and Catalog classes allows correct management.

Manager	
Responsibilities: • View Users: The administrator is responsible for viewing the list of users registered in the system. • Modify the catalog: The administrator can make modifications to the catalog pins.	**Collaborators:** • Catalog Class: Collaborates with the Catalog class to manage its pins.

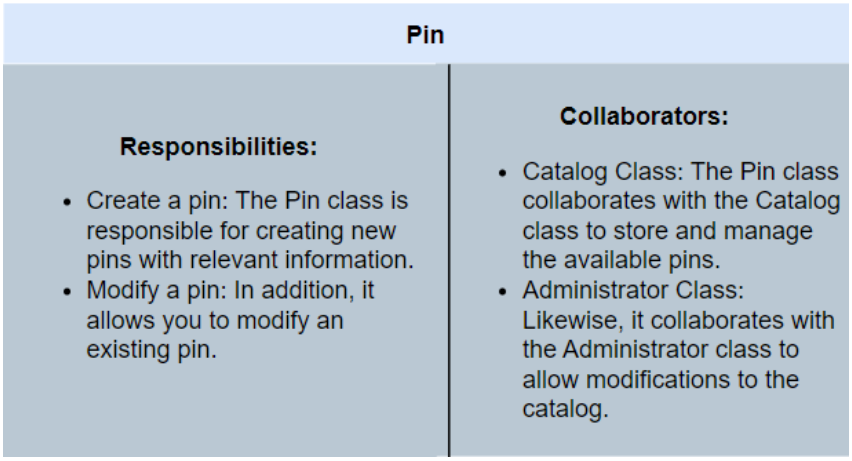
Catalog class:

The Catalog class assumes the responsibility of managing the catalog of pins available in the system. Its design, based on the Singleton pattern, guarantees the existence of a single catalog instance and provides controlled access methods. Collaboration with the Pin class ensures proper management of pin information, while interaction with the Maneger class allows authorization and consistency in catalog modifications. This approach promotes an efficient design, avoiding inconsistency problems when having a single instance and improves efficiency in accessing system resources.

Catalog	
Responsibilities: • Create a catalog instance: The Catalog class is responsible for creating a single catalog instance. • Get Catalog Instance: Also provides a method to get that instance.	**Collaborators:** • Pin Class: The catalog collaborates with the Pin class to store and manage available pins. • Administrator Class: Additionally, it collaborates with the Administrator class to allow modifications to the catalog.

Pin class:

The Pin class plays a fundamental role in managing pins within the system. Its design focuses on providing functionality for the creation and modification of pins, ensuring the integrity and relevance of the associated information.



Entity-Relationship Diagram

The design of the Entity-Relationship Diagram is based on a series of decisions aimed at accurately and efficiently reflecting the relationships between system entities. Below, the justifications behind each decision are explained:

TypeOfUser

The inclusion of the TypeOfUser entity is justified by the need to distinguish between different user roles, such as clients and administrators. This separation facilitates the implementation of specific functionalities for each type of user, resulting in greater flexibility in the database.

Usuario (User)

The one-to-many relationship between TypeOfUser and Usuario was chosen to allow the assignment of a user type to multiple users, reflecting the system's capability to support multiple users with different roles. This enables efficient management of registered users and facilitates the implementation of role-based access control.

Pin

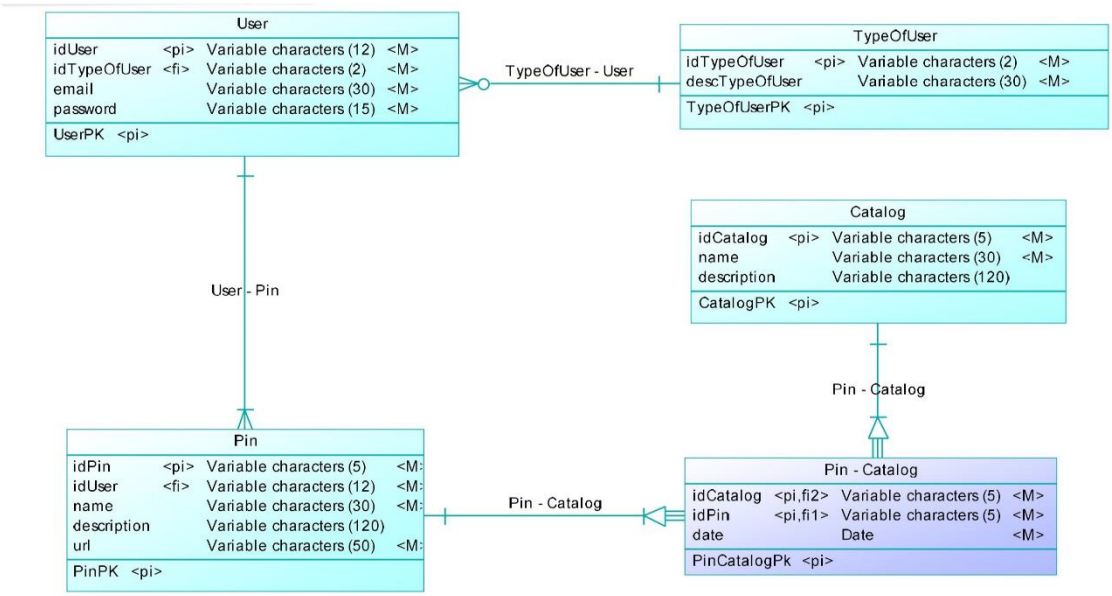
Associating each pin with a specific user was done to maintain a clear record of pin ownership in the database. This ensures that each pin is correctly attributed to its creator, facilitating management and tracking of activity in the system.

Catálogo (Catalog)

The Catalog entity was included to organize and group pins. The many-to-many relationship between Pin and Catálogo is due to the possibility of a pin belonging to multiple catalogs and viceversa. This allows flexibility in pin organization and facilitates navigation and search.

In summary, the design of the Entity-Relationship Diagram is guided by the need to accurately represent relationships between entities, promoting efficient management and

navigation within the system. Each decision contributes to the overall flexibility and functionality of the database.



Some of the queries we're going to use for the app with the DB are the next one:

. Show all the users:
```sql
SELECT *
FROM "USER";
```

2. Show the catalog with its pins:

```sql
SELECT c.IDCATALOG, c.NAME AS CATALOG_NAME, c.DESCRIPTION AS CATALOG_DESCRIPTION,
      p.IDPIN, p.NAME AS PIN_NAME, p.DESCRIPTION AS PIN_DESCRIPTION, p.URL
FROM CATALOG c
JOIN USER u ON c.IDCATALOG = u.IDCATALOG
JOIN PIN p ON up2.IDPIN = p.IDPIN;
```

3. Show all the pins

```sql
SELECT *
FROM PIN;
```

UML Diagram

The association between User and Pin reflects the link between users and the pins they have created in the system. This relationship allows a user to have one or more pins associated with them. On the other hand, the composition from Pin to Catalog establishes that the Catalog is composed of elements from the Pin class. This design decision implies that the catalog has, as an attribute, a list of all the pins it is composed of.

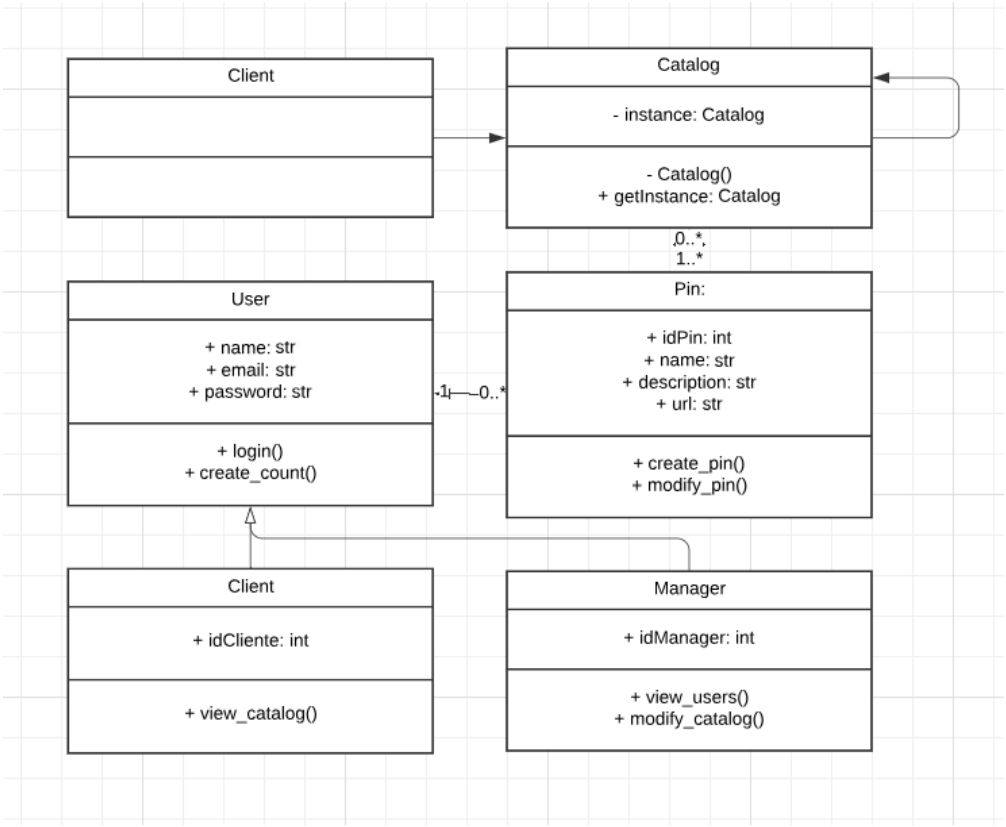
The use of the Singleton pattern for the Catalog class is due to this class assuming the responsibility of managing the catalog of available pins in the system, and by implementing this pattern, it ensures the existence of a single instance of Catalog throughout the system. This provides several benefits, which are:

- Instance Control: By restricting the creation of instances of the Catalog class to a single instance, the possibility of creating multiple instances that could lead to inconsistencies in the catalog data is avoided.

- Global Access: The Singleton pattern provides a global access point to the Catalog instance, facilitating its use in different parts of the system without the need to pass the instance from one object to another.

- Resource Efficiency: By having a single instance of Catalog in memory, the system's resource consumption is reduced since multiple copies of the same information are not needed.

- Data Consistency: By managing all operations related to the catalog through a single instance, data consistency is ensured, and potential update conflicts between different instances of the catalog are avoided.



References

- Pinterest, What is pinterest (2024) https://help.pinterest.com/es/guide/all-about-pinterest Atlassian, User Stories (2022) https://acortar.link/ymjOC9
- Unifed Model Lenguage, What is UML? (2022) https://www.uml-diagrams.org/
- Singleton (2024) https://refactoring.guru/es/designpatterns/singleton\\