Development and Design of a Microblogging Platform:Vibe

Tellez Melo David Santiago
Lopez Hernandez Johan Nicolas
Universidad Distrital Francisco José de Caldas
dstellezm@udistrital.edu.co
jnlopezh@udistrital.edu.co

***Abstract*—The growing complexity of modern social media platforms presents challenges for developers seeking to create lightweight and scalable systems. This paper proposes the design and development of a microblogging application that focuses on brief and multimedia content, leveraging object-oriented programming principles to build a modular and efficient system. The result is an application with intuitive user interactions, visually appealing components, and an optimized user experience, ensuring rapid performance, ease of maintenance, and scalability to accommodate future growth.**

***Index Terms*—Microblogging, Social Networks, Object Oriented Programming, Java.**

I. Introduction

Social networks have become one of the most influential digital tools of the 21st century. Through them, millions of people around the world interact, share information, express opinions, and actively participate in the construction of public discourse. Among all these platforms, Twitter has stood out for its focus on short messages—known as *tweets*—that enable fast, direct, and accessible communication for any user. This dynamic, known as *microblogging*, has been adopted not only by individuals but also by institutions, media outlets, and public figures as an effective method for instant content distribution.

The idea of building a social network similar to Twitter, even in a simplified version, requires thinking about how data is structured, how different elements (users, posts, reactions, notifications) relate to each other, and how to provide a coherent experience for those who interact with the application. In our case, as students learning object-oriented programming, we decided to take on this challenge in order to apply and reinforce our knowledge through a collaborative, functional, and meaningful project.

Our project is called *Vibe*. The goal behind this application is not only to recreate a familiar social network, but to build a digital platform from scratch that allows us to apply fundamental programming principles. In *Vibe*, users can register, create posts of up to 280 characters, follow other users, and react to content. Although this set of functionalities may seem simple, it requires a well-structured system design to maintain clarity, scalability, and cohesion between its components.

To organize this system, we applied the core principles of object-oriented programming: encapsulation, inheritance, abstraction, and polymorphism. These concepts guided us in defining the classes that form the foundation of the application, such as `User`, `Post`, `Feed`, and `Notification`, as well as the implementation of methods that enable interaction between them. We used tools like UML class diagrams and CRC cards to visualize and define the responsibilities of each class, their attributes, and their relationships with other entities.

In addition, we developed usage flows that represent the most common processes in a social network: publishing content, reacting with a ”like,” following another user, and viewing a feed of posts. Each of these flows was diagrammed to identify the main actors, the actions involved, and the possible outcomes. To complement this, we created mockups or visual drafts of the user interface, which helped us imagine and agree on how the final version of the application should look.

Projects like this have previously been documented by authors such as Smith and Doe [1], who propose scalable architectures for microblogging services using REST APIs, and by Gamma et al. [2], whose work on object-oriented design patterns has been a key reference for building reusable software structures.

The main classes have already been coded, and basic functionality tests have begun. The development of *Vibe* is progressing with the clear goal of building a fully functional social network, designed and implemented by students from the ground up.

II. Methods and Materials

The development of *Vibe* was structured based on the principles of object-oriented programming, which form a fundamental part of the system's design approach. This methodology is essential for organizing and properly representing the entities that make up a social network, allowing us to model objects such as users, posts, notifications, and interactions in a logical and coherent way. By using classes, attributes, and methods, we established a clear structure that facilitates data management and the relationships between system elements.

The system architecture is composed of several main classes. The `User` class encapsulates each user's data, including name, age, email address, password, published posts, and followed users. To protect sensitive information, access modifiers were used to restrict direct access to certain attributes. Interaction with this data is handled through public methods,

which allow controlled access and updates. This class also defines methods for actions such as following other users, reacting to content, and receiving notifications.

The `Post` class represents the publications within the platform. It was defined as an abstract class to allow the creation of specialized subclasses, such as image posts or video posts, which can extend the basic functionality without altering the general structure. Each post contains attributes such as the author, content, creation date, and number of reactions. Actions such as publishing, editing, or generating previews are managed from this class and its derivatives.

The `Notification` class is responsible for generating and distributing notifications among users when a relevant interaction occurs, such as a "like" or a new follower. This class centralizes the alert logic and keeps users informed about activity within their network. Notifications are also modeled through subclasses, making it possible to customize their behavior and presentation depending on the type of event.

The `Feed` component organizes and displays the most relevant posts to each user. This module retrieves, sorts, and filters content according to defined criteria, such as publication date or type of interaction. Its main function is to present a continuous stream of posts that match the user's interests, based on their activity and the accounts they follow.

To plan and define the system structure, several visual tools were used to support design and organization. UML class diagrams made it possible to graphically represent inheritance, association, and aggregation relationships between different classes. Additionally, CRC cards (Class, Responsibility, Collaborator) helped identify the specific responsibilities of each class and the objects with which they should interact. These tools facilitated teamwork and allowed us to maintain a shared vision of the code structure.

We also defined usage flows that represent common actions in a social network, such as creating posts, reacting with likes, following other users, and viewing a content feed. Each of these flows was diagrammed to clarify the sequence of events and the actors involved in each functionality. In addition, mockups were designed to represent the general appearance of the user interfaces, including the home screen, login form, main feed, and registration section.

Figures 1, 2, and 3 show the CRC cards, UML class diagram, and main interaction flows within the application, respectively. These visual representations were key to consolidating the functional design of the platform.

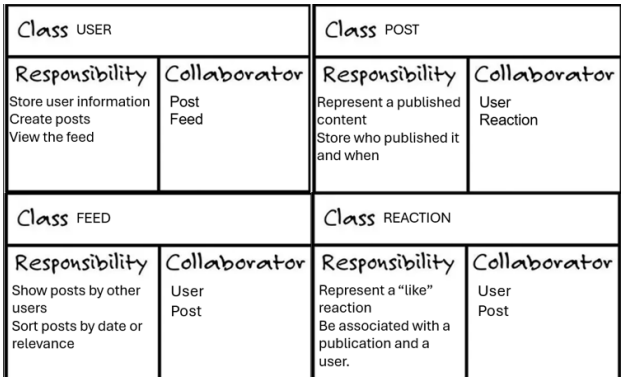

Figure 1. CRC cards defining class responsibilities and collaborators.

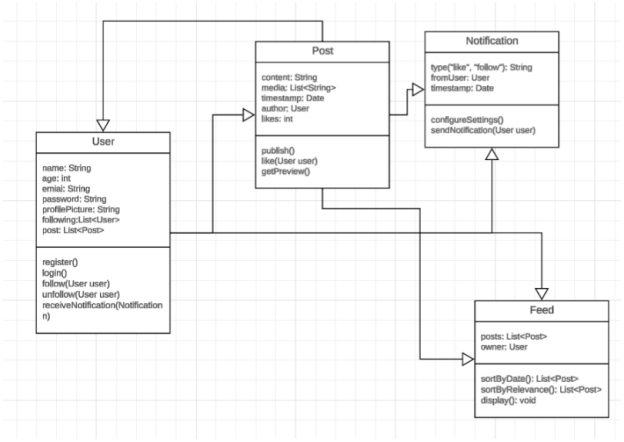

Figure 2. UML diagram showing main classes and their relationships.

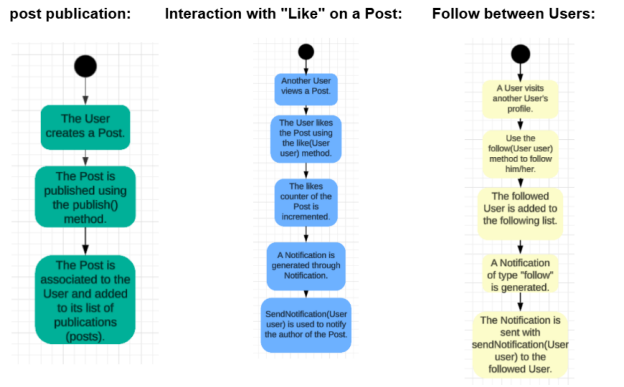

Figure 3. Usage flows illustrating post creation, reactions, and user following.

III. Results

IV. Conclusions

References

[1] J. Smith and A. Doe, "Building Scalable Microblogging Services," *Journal of Web Systems*, vol. 12, no. 3, pp. 123–145, 2020.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.