Vibe: Development of a Social Platform for Microblogging

Tellez Melo David Santiago

Lopez Hernandez Johan Nicolas

Universidad Distrital Francisco José de Caldas

May 15, 2025

Contents

List of Figures

List of Tables

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.

Chapter 1

Introduction

Currently, social networks play a fundamental role in the way people communicate, access information, and build digital interaction spaces. Among the various existing formats, microblogging has emerged as one of the most dynamic and effective. Platforms like Twitter have shown that it is possible to convey ideas, news, and opinions in brief, accessible messages that are easy to spread.

This context served as the inspiration for the development of Vibe, a social platform designed to facilitate the posting of brief, visual, and real-time content. The system allows users to register, share posts up to 280 characters, react to content, follow other users, and receive notifications about relevant interactions. The application has been designed from its technical foundations to provide a simple, efficient, and modular experience, focused on replicating the core logic of modern microblogging.

The main goal of the project is to build a functional platform that replicates the fundamental mechanisms of microblogging social networks, using an object-oriented programming architecture. Through the design of entities such as users, posts, feeds, and notifications, a solid structure was established to represent the basic operations that users expect from a modern social network.

During the development of the project, various modeling and documentation tools will be used, including UML diagrams, CRC cards, and interaction flows. These elements will facilitate the definition of main classes, their attributes, methods, and relationships, as well as the planning of posting, reaction, and following processes. Additionally, mockups will be

designed to visually represent the different user interface screens, allowing for an anticipation of the system's final user experience.

Vibe emerges as a clear proposal, aimed at simplifying the structure and functionality of a traditional social network, while still keeping in mind the key elements that define the user experience. The modular approach, combined with a clean design focused on essential interaction, ensures that the platform is easily scalable and adaptable to future needs or integrations.

Chapter 2

Literature Review

The development of social platforms based on microblogging has been extensively studied in recent years, particularly in terms of their architecture, scalability, user experience design, and technical structure. Numerous studies have explored how to design and implement these systems to support a large number of users, ensure acceptable response times, and maintain the integrity of shared data.

Smith and Doe (2020) propose a scalable architecture for microblogging services, which relies on distributed clusters, REST APIs, and decoupled controllers. Their research highlights the importance of organizing system components modularly to enhance fault tolerance and horizontal scalability. They also emphasize the use of clearly defined logical layers to separate client-side operations, server logic, and data management — principles that also guide the design of the Vibe platform [1].

Furthermore, the work of Gamma et al. (1994) on object-oriented design patterns has become a foundational reference in the structuring of complex software. Their contributions, particularly in code reuse, abstract classes, and responsibility separation, allow for the creation of cleaner, more maintainable, and scalable applications. While Vibe does not directly implement specific patterns documented by these authors, the general principles of modular design, encapsulation, and inheritance play a central role in its development [2].

While platforms like Mastodon and Pleroma offer similar services in a federated and decentralized context, they often present significant technical barriers for new developers. Their codebases are designed for complex production environments, making them challenging

to adapt or understand without advanced technical experience. In contrast, Vibe seeks to address this gap with a simpler and more approachable solution, without compromising on a well-structured architecture or best development practices.

Additionally, the literature indicates that user experience (UX) is a critical factor in the adoption of social networks. Studies in this area recommend simple interfaces, intuitive workflows, and immediate feedback as essential elements to encourage user engagement. In this regard, Vibe relies on visual mockups that help anticipate and evaluate user interactions and feature flows, even before the final implementation.

Chapter 3

Background

Building a social network like Vibe requires understanding a range of technical concepts and structures that form the foundation of both its design and functionality. Among the most important are object-oriented programming (OOP) principles, user interaction flows, and the modular design of software.

Object-oriented programming allows for modeling real-world entities as classes within the code, making it easier to represent users, posts, notifications, interactions, and feeds as independent objects with defined attributes and behaviors. This methodology encourages component reuse, isolates functionalities, and enables future system scalability. In Vibe, each class has been designed with a clear responsibility, following the single responsibility principle, and its relationships are defined through inheritance, composition, and aggregation mechanisms.

The architecture of Vibe is based on a logical separation into layers: a model layer that contains the core domain classes, a controller layer that manages business logic, and a view layer responsible for the user interfaces. This structure enhances system maintainability and allows each component to evolve independently.

The planning tools used also contribute to the understanding of the project. UML class diagrams provide a visual representation of the relationships between objects and the methods available in each class. Meanwhile, CRC cards help define precisely what responsibilities each class holds and with whom it must collaborate. These graphical representations were crucial in establishing a clear design before implementation.

Finally, the development of Vibe is also supported by basic digital interaction concepts, such as user flows and visual mockups. These elements help anticipate how a user will navigate the application, what actions they can perform, and how information will be displayed on the screen. All of this ensures that the system not only functions well internally but also offers a coherent and engaging user experience.

Chapter 4

Objectives

Among the objectives we have for this application are:

-to provide a simple to understand interface

-basic user information should be able to be entered: name, age, profile picture, etc.

-Possibility to publish texts of up to 280 characters, links and images.

-Have a button with which users can react to each other's content.

-Have a main feed where you can see the content of other users.

Chapter 5

Scope

The goal of this project is to develop a lightweight social platform focused on the microblogging model, similar to networks like Twitter. The scope of this work includes the conceptual, structural, and functional design of the application, ranging from the definition of core classes to the planning of the user interface, interaction flows, and the internal structures needed for its operation.

Included in the scope are basic functionalities such as user registration, creating and viewing posts with text, links, or images, interacting with posts via a reaction button, managing a personalized main feed, and receiving notifications when relevant actions occur between users. Additionally, the project involves applying object-oriented programming principles to structure the entire system in a clear and maintainable manner.

On the other hand, certain more complex aspects are outside the scope of this project, as they require additional integrations or specialized environments. These exclusions include the development of a persistent database, the implementation of cloud servers, distributed version control, advanced authentication and security systems, and the creation of an independent mobile client. While these elements could be addressed in later stages, the current focus is on creating a functional version within a controlled and self-contained environment.

Chapter 6

Assumptions

For the development of the Vibe project, several assumptions were made to define the system's functional and technical boundaries, as well as to simplify certain design aspects without compromising coherence or functionality.

Firstly, it is assumed that all users interacting with the platform do so legitimately, and that advanced authentication controls or protections against malicious users will not be implemented at this stage of development. This allows the focus to remain on the system's functional logic rather than external security mechanisms.

It is also assumed that the data entered by users, such as name, age, or profile pictures, is valid and does not require external verification. Similarly, it is assumed that posts will not contain inappropriate content and will not require automatic moderation.

Lastly, it is assumed that the functionalities defined as objectives (a simple interface, brief posts, reactions, and a dynamic feed) are sufficient to demonstrate the proposed microblogging model. Any additional functionalities are outside the scope of the initial assumptions.

Chapter 7

Limitations

During the development of the Vibe platform, several limitations have been identified that affect its current scope and functionality. One of the main limitations is that the application is designed to run in a local environment, without integration with external databases or cloud services, which limits its ability to handle large numbers of users or data persistently.

Another important limitation is that, for now, the functionalities focus on basic operations such as user registration, posts, reactions, and following. Additional features like comments, content moderation, or advanced security and authentication systems have not yet been implemented.

Furthermore, since the project is still under development, no automated tests or formal performance or usability evaluations have been conducted, which may affect stability and the user experience in real-world scenarios.

Lastly, the lack of real users to test the application prevents a full validation of the interface and interaction flows, meaning some improvements may arise once user testing is carried out.

Chapter 8

Methodology

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. Figures 8.1, 8.2, and 8.3 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.

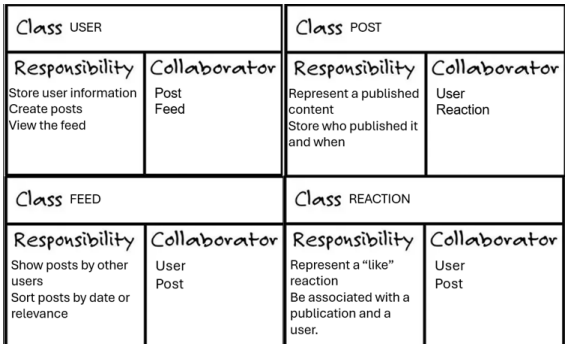

Figure 8.1: CRC cards defining class responsibilities and collaborators.

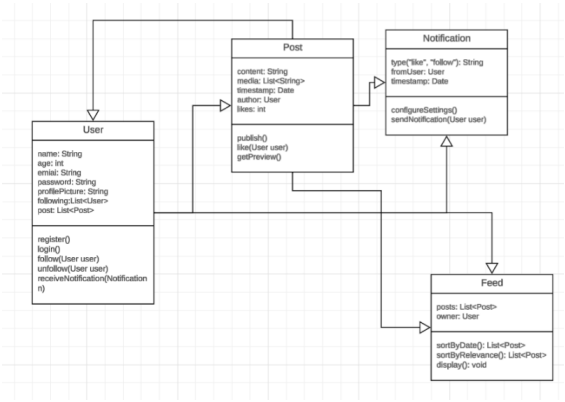

Figure 8.2: UML diagram showing main classes and their relationships.

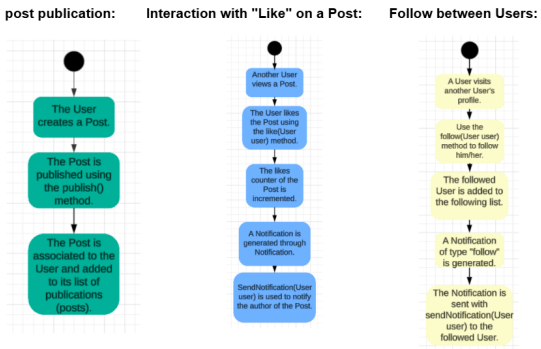

Figure 8.3: Usage flows illustrating post creation, reactions, and user following.

Chapter 9

Results

Chapter 10

Discussion

Chapter 11

Conclusion

Acknowledgements

Glossary

Microblogging A form of online communication where users share short messages, typically limited in length, that may include text, links, or images.

Object-Oriented Programming (OOP) A programming paradigm based on the concept of "objects", which contain data and behavior. Core principles include encapsulation, inheritance, abstraction, and polymorphism.

User An entity that interacts with the platform, capable of creating content, reacting to posts, and following other users.

Post A unit of content created by a user, which can include text, images, or links, and can be viewed and reacted to by others.

Feed A dynamic list of posts displayed to the user, generally ordered by recency or relevance, and composed of content from followed users.

Reaction An interaction (such as a "like") that a user can perform on another user's post to express acknowledgment or emotion.

Notification An alert generated by the system to inform users of interactions related to their content or profile (e.g., a new follower or a received reaction).

CRC Card A design tool used to identify and define the responsibilities of classes and how they collaborate with others.

UML Diagram A visual representation of a software system's structure, showing classes, attributes, methods, and relationships.

Mockup A visual prototype or sketch of a user interface, used to plan and preview the design and layout of the application.

Scalability The system's capacity to handle growth, such as increased users or data, without losing performance or stability.

Bibliography

[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.

Appendix A

Appendix A

Appendix B

Appendix B

List of Figures