Development of a digital wallet using object-oriented programming and software design patterns

Cristian Santiago López Cadena[1] 20222020027[1]
Carlos Alberto Barriga Gámez[2] 20222020179[2]

Universidad Distrital Francisco José de Caldas

***Abstract*— Electronic commerce refers to any commercial transaction that uses the transfer of information over electronic networks to buy and sell goods or services.[2] In this way, the purpose of this research is to show how a digital wallet can be developed using the concepts of object-oriented programming, as well as software design patterns. In order to achieve this objective, the application design process was developed, in which the user stories necessary to meet the needs of the application were determined. Likewise, class, activity, sequence, and deployment diagrams were created in order to guide the implementation of the software.**

Now, for the implementation of the software, Python version 3.12.1 was used, and Java version 17.02.12 was used for the development of the software backend. Likewise, for the data layer, it was decided to use JSON files, due to their ease of handling and access.

Because the digital wallet is designed with 2 backends (Java and Python), it was decided to use the FastApi and Spring Boot tools to develop the application services and carry out the respective information exchanges between the backends. It is important to note that the testing of the services of both backends was carried out using the postman software.

I. INTRODUCTION

Object-oriented programming is defined as a programming paradigm that attempts to simulate things in the real world through elements called objects. These objects have some characteristics such as inheritance, polymorphism, encapsulation, and abstraction. [3]

Abstraction is defined as a process where the essential characteristics of an object are extracted depending on their importance in solving the problem.

On the other hand, encapsulation is the process of hiding information from the end user, establishing visible details that contribute to the user experience, and hidden details that support the software.

Modularity aims to reduce a system into parts, in this way it is sought that the parts of the system have high connectivity (cohesion) and reduced association (coupling).

Inheritance is a mechanism with which new classes are created that reuse and modify the behavior of previously created classes. Inheritance is defined hierarchically where a parent class is defined over several child classes.

Polymorphism is defined as the ability of objects belonging to a class to generate a different response depending on the parameters sent to it. [3]

Universidad Distrital Francisco José de Caldas

Likewise, these objects are defined as a series of behaviors called methods and properties known as attributes. [1].

Within the object-oriented programming paradigm, the SOLID principles are presented, aimed at improving the understandability and maintainability of software. [5]. The SOLID principles are discussed below:

The single responsibility principle (S) determines that the software module must have a single reason for changing.

Now, the principle of open and closed (O) states that the modules must be open for extension and closed for modification.

The Liskov substitution principle (L) states that the classes of a program can be replaced by the instances of their subclasses, without altering the functionality of the program.

The principle of interface segregation (I) states that if we create an interface we must ensure that the class that is going to implement this interface will have the ability to implement all the methods.

Dependency inversion (D) means that parent and child modules should not depend on implementations, but on abstractions.

According to the authors [7], design patterns are proven solutions to certain problems encountered when designing software.

This project search to develop an digital wallet using the object-oriented programming paradigm and the software patterns. The author [4] defines digital wallets as software that allows us to store electronic money for later use in online commerce. Likewise, the author states that digital wallets are used as a means of payment to send and receive money safely.

Starting from these concepts, it is important to highlight that this research aims to understand how object-oriented programming can be used to develop software for an virtual wallet platform. In the same way, this project will be developed with a monolithic architecture, which is defined as an architecture in which all the layers of the application are compiled into a single unit. [6]

So, in this document, the second chapter will discuss the methods and materials necessary to develop object-oriented software from the virtual wallet. In the next chapter, the experiments to be carried out to meet the objective of developing a virtual wallet will be discussed and the results of this experimental practice will also be discussed.

In a final chapter, the conclusions about the functionality of object-oriented programming and the software design patterns in the creation of a virtual wallet will be developed.

II. METHODS AND MATERIALS

A. Methods

The aim of this project was to develop a functional virtual wallet platform. To this end, the following technical decisions were made:
For the implementation of the software, Python version 3.12.1 and Java version 17.02.12 were used for the development of the logical part of the software. Likewise, for the development of the data layer, it was decided to use Json files to perform all the necessary operations with information.

Likewise, it was decided to use the FastApi and Spring Boot tools to manage the application services and carry out the necessary information exchanges between the two backends.
The application uses Postman software as a means of testing web services, which is also the graphical interface of the application.

Regarding the development process, we decided to use the GitHub storage software to facilitate the cooperative development of the application and the management of its versions. We sought to make atomic updates to the application to easily recognize errors and perform the necessary maintenance.

Class diagram decisions:

When we developed the class diagram, the following decisions were made:

Initially we have a UserFacade class that represents the endpoint of the service that connects the Java backend with the Main class. Likewise, the User class has an aggregation with the UserFacade. It was decided to create a user class that contains the attributes of type id, id Number, password and phone.
This class will function as an interface for the customer class and the Admin class.
The customer class was created in order to specialize each customer according to the permissions they have, in this way the user will have a higher level when registering certain data in the application. This class will have the attributes name, wallet ID, email,current address, haveTicket and tickets. It will also have the methods changeAddress, createWallet, getWalletid, getStatus and setStatus.
The admin class was developed in order to have more complex permissions, such as the ability to freeze digital wallets, view the customers movements, report accounts and create tickets.

We also have a WalletFacade class that represents the service that connects the wallet module with the Main class. The Wallet class has an aggregation with the WalletFacade class. The wallet class contains the attributes necessary to represent a digital wallet. It will have the wallet id, wallet balance and the movements that have been made in it.
It will also have the necessary methods to withdraw, send, add and request funds and it will also have a method to show the movements of the wallet and add a credit card to the wallet.
The transaction class has the attributes account reference id, wallet ID, transaction amount, date and account that receives the money.

The credit card will contain the attributes necessary to represent a credit card such as the number, expiration date, and cvv , as well as a wallet ID attribute to link it to a specific digital wallet. We also have the pocket factory class, which works as an interface to create specialized pockets (bills, savings, investments). The complete class diagram is attached at the end of the document (Fig 1. Class diagram). Similarly, regarding the application components, we have the Main component, the Users module component, and the wallets module component. The component diagram is located in the appendices section (Fig 2. Components diagram)

B. Materials

In the project, some hardware elements were used for the development of the online sales platform, among them 1 GB of ram were used for the execution of the software since we consider it necessary for the efficient performance of the application. A 10th generation Intel core i5 processor were also used because this processor can run the processes to meet the demand of the platform.
Finally, a 1 GB storage space intended only for the execution of the application were used, because we consider that this can store the software information and other software necessary for development, without generating problems with the space reserved for the operating system.

III. RESULTS

To verify the correct functioning of the software, it was decided to use user stories as a way of verifying the effectiveness of the software. In this way, it will be determined whether each of these stories can be fulfilled with the digital wallet at the end of the project.

On the other hand, tests were carried out on the application services in order to determine whether they generate the requested information in each case. This process was carried out with the Postman software, in which each of the Java and Python backend services were tested.
Unit tests of the code were also carried out to determine whether the software fulfills its intended purpose.
To do this, the persistence of the information was tested by checking whether the information can be stored and manipulated correctly in the json files.

Tools such as pylint and black formatter were also used to verify and avoid code smell errors in the project, allowing the generation of well-structured code.
We performed a detailed analysis of the design patterns

we implemented, making sure to implement only those that really added value to the project. This allowed us to avoid anti-patterns that could add unnecessary complexity. Regarding the patterns applied, we prioritized low coupling and the use of interfaces, which allowed us to develop decoupled, scalable, flexible, and easy-to-maintain code.

IV. CONCLUSIONS

The development of a digital wallet using object-oriented programming and software design patterns demonstrated the applicability and advantages of these approaches in creating modular, scalable, and maintainable systems. During implementation, SOLID principles and design patterns were applied to ensure proper code organization and reduced dependency between modules, thus facilitating their reuse and evolution.

The combination of Python and Java as backends, along with the use of FastAPI and Spring Boot for communication between the two, demonstrated the viability of an architecture that allows taking advantage of the strengths of each technology. Data persistence through JSON files proved to be an efficient and easy-to-use solution for storing information, although in higher scalability scenarios it would be advisable to consider more robust databases.

Tests performed with Postman and code analysis tools such as pylint and black formatter ensured the quality and functionality of the software, minimizing errors and guaranteeing its correct performance. Likewise, validation through user stories made it possible to verify that the digital wallet met the stated requirements and offered an adequate experience to users.

REFERENCES

[1] S. Valbuena and S. A. Cardona, “Object-oriented programming principles” Elizcom S.A.S, 2018, pp. 7.

[2] C. A. Robleto, “Electronic Commerce: Background, Definitions and Subjects”, 2004, pp. 6-8.

[3] J. B .Bermudez, “Oriented object programming with java”, 2012, pp. 7-8.

[4] G.Bellindo, “Electronic wallets: a tool for entrepreneurship in the digital age”, 2023,

[5] Autentia, “Software design, principles and patterns of the software development”, 2012, pp. 9-17.

[6] Amazon, “What is the difference between monolithic and microservices architecture?”, 2023.

[7] E.Gamma, R.Helm, R. Johnson, J.Vissides, “Design patterns. Elements of reusable object-oriented Software”, 1994.

V. ANNEXES

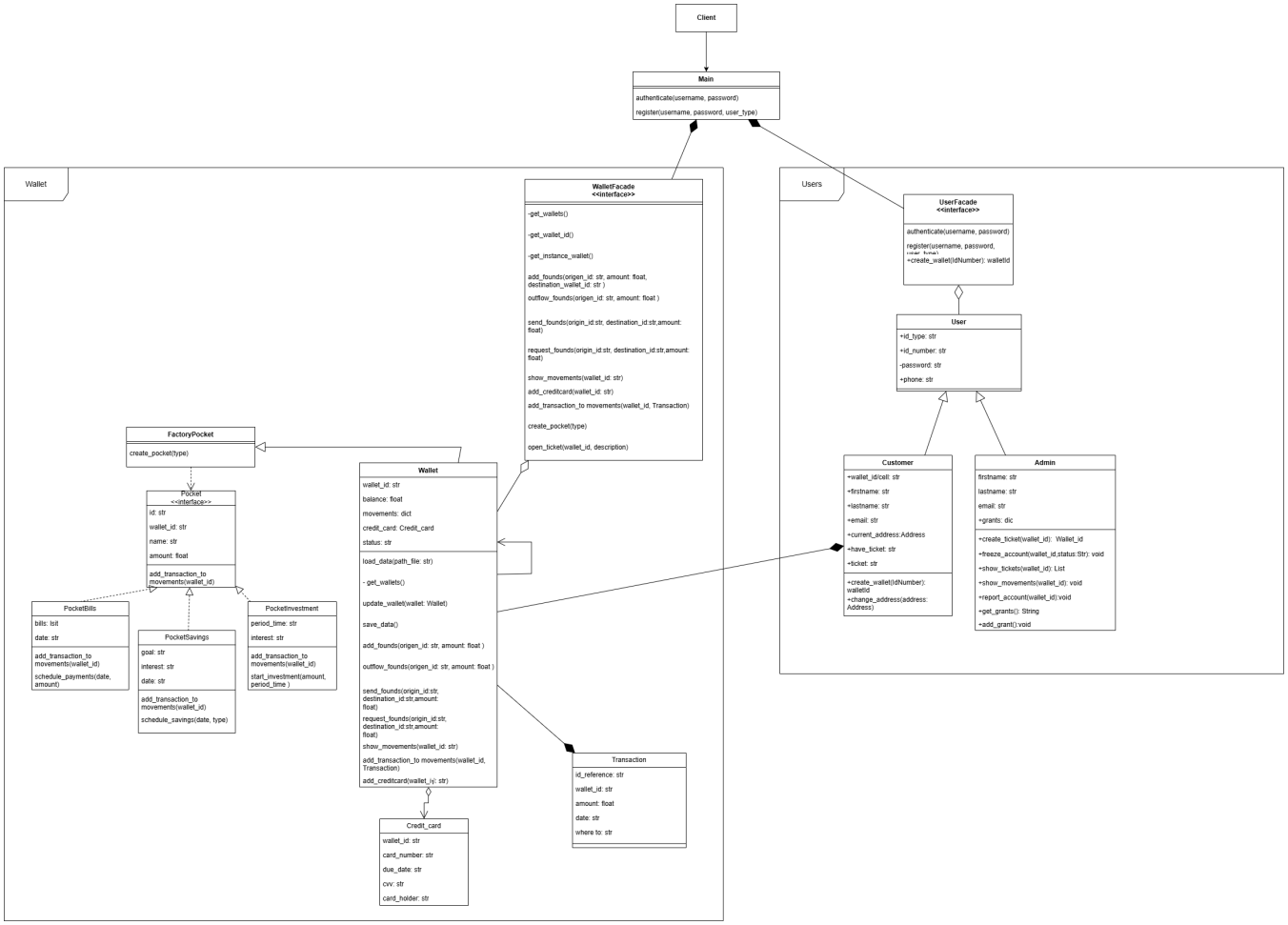

Fig. 1. Class diagram

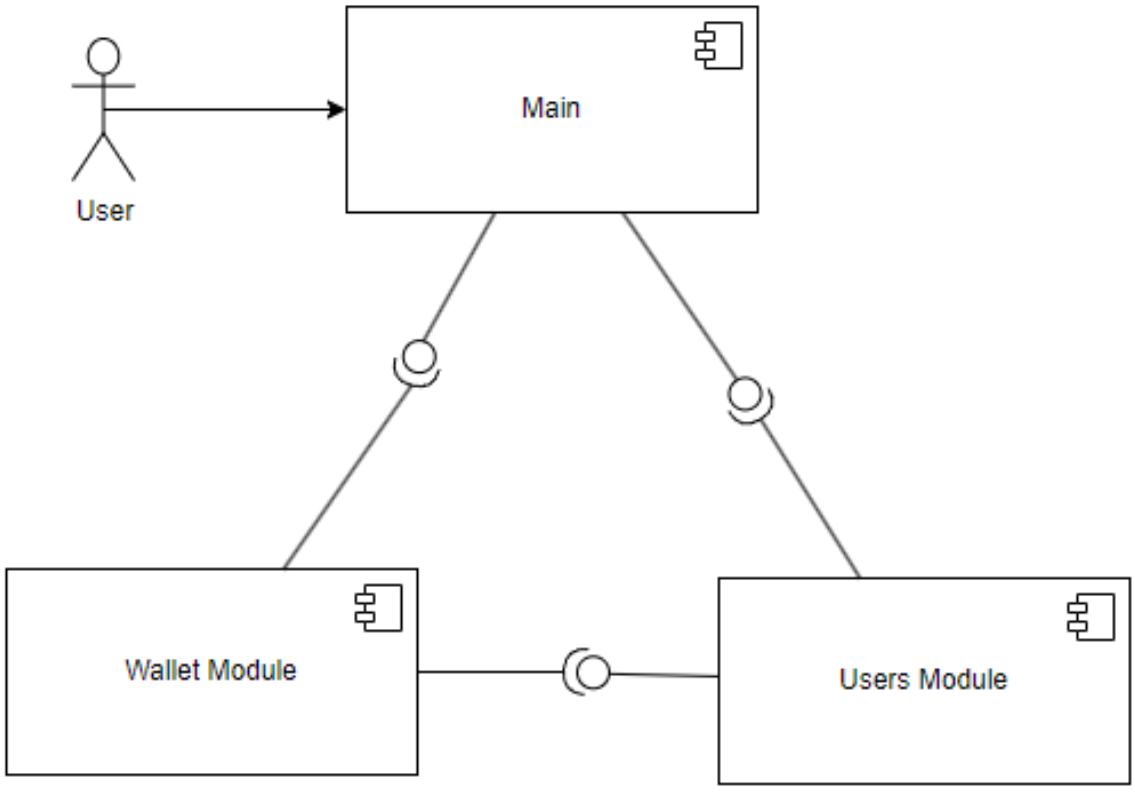

Fig. 2. Comopnents diagram