Universidad Distrital Francisco José de Caldas

Faculty of engineering

Technical Report Digital wallet

Software modelling

Cristian Santiago López Cadena - 20222020027

Carlos Alberto Barriga Gámez - 20222020179

Bogotá, D.C.

1. Project description

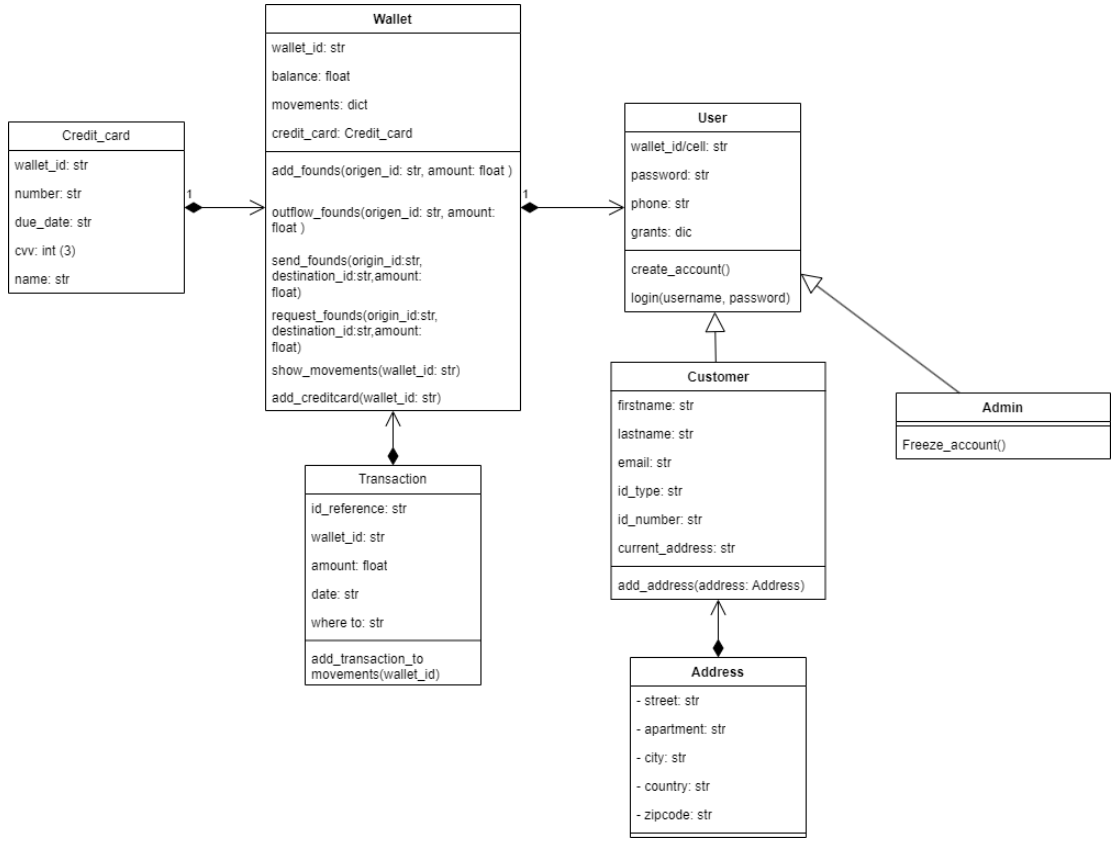
This project consists of developing a monolithic software for a digital wallet application. Regarding stakeholders, we consider that our software can be functional for those banking companies that want to expand their capabilities and reach with the public. Likewise, those users who require a simple and efficient way to manage their money may be interested in the virtual wallet.
Now, among the tools used to develop the software, it was decided to use python and java for the development of the backend. For the database, it was decided to use the ORM Sqlalchemy, because it allows managing the database with a syntax similar to that of programming languages.
Likewise, GraphQL will be used, in order to fully manage the application services.
Regarding the hardware that will be used to make the software, it was decided that it will need 1 GB of RAM, 1 GB of hard drive and a tenth generation Intel Core i5 processor.

2. User stories

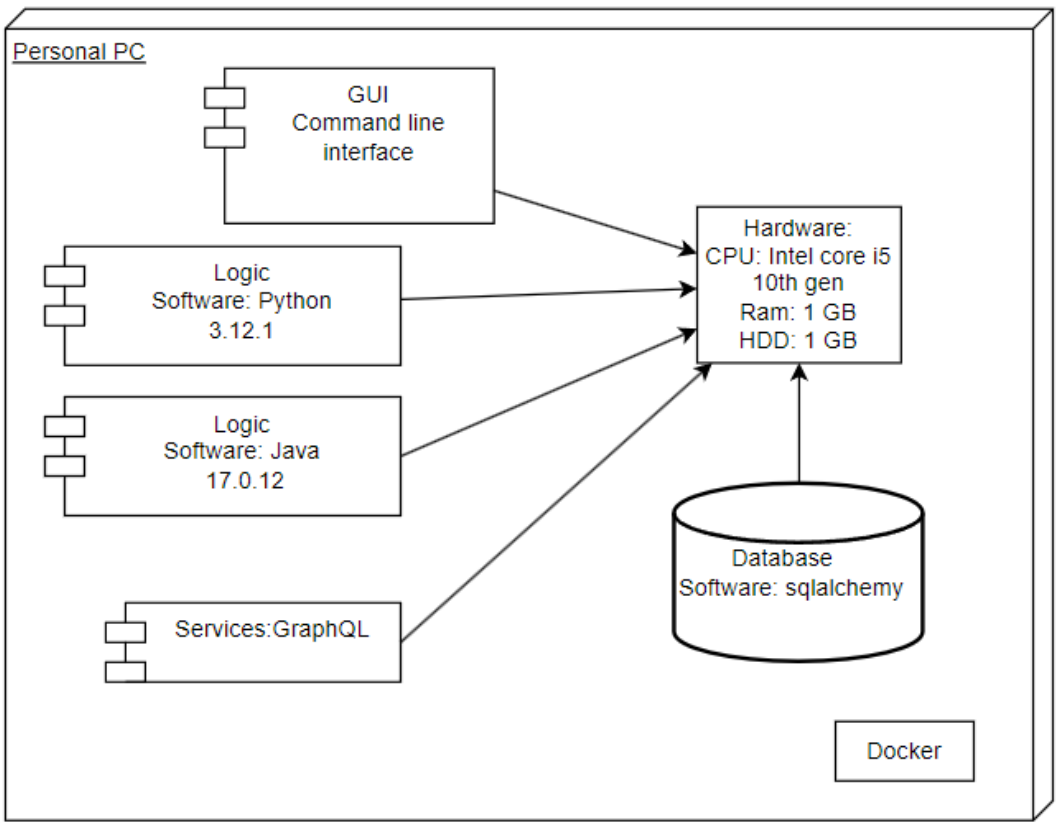
UH_1: As CTO, I want to keep a record of the users who enter the application. I want to know their first and last name, cell phone number, email address, and ID, to know who is using our service.
UH_2: As CTO I want to offer a simple way for users to access their digital wallet, so that everyone can access the service easily and immediately.
UH_3: As CTO I want to be able to send money between different wallets so that users can make fast and secure transfers within the ecosystem.
UH_4: As CTO I want users to be able to link a debit card to their wallet so they can make payments easily.
UH_5: As a user, I want to be able to withdraw money from my wallet at a physical location so that I can access my cash quickly and conveniently when I need it.
UH_6: As a user, I want to be able to load money into my account in different ways so that I can have flexibility and ease when adding funds.
UH_7: As a user, I want to be able to request money from a friend who also has the app so that I can receive the money quickly and easily without complications.
UH_8: As a user, I want to be able to see how much money I currently have and a history of all the transactions made so that I can keep track and monitor my finances.
UH_9: As CTO, I want to know the user's place of residence and their ID to ensure that all legal requirements are met.

3. UML DIAGRAMS

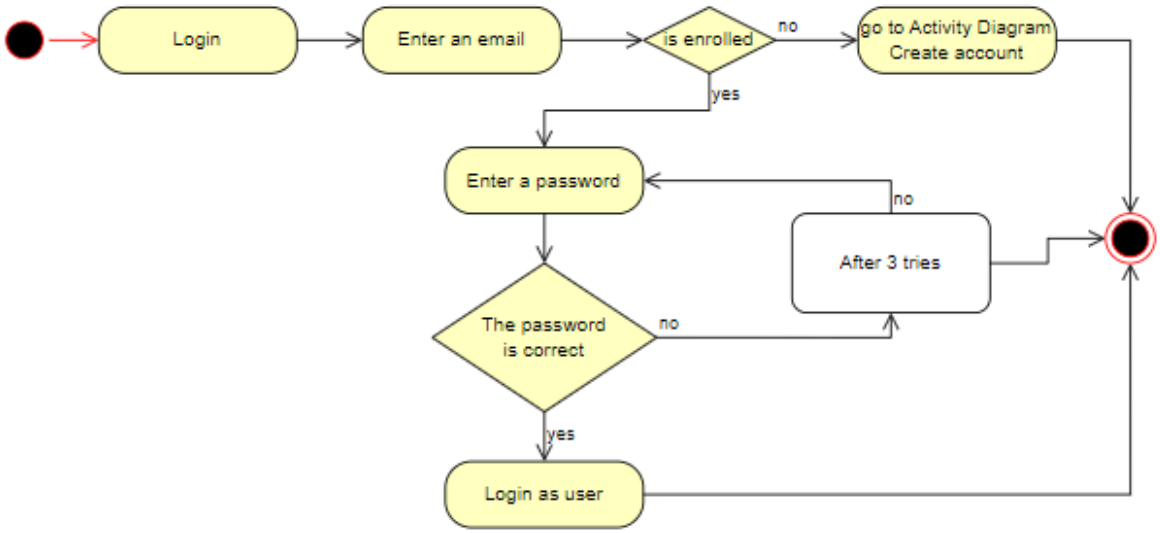
Class Diagram



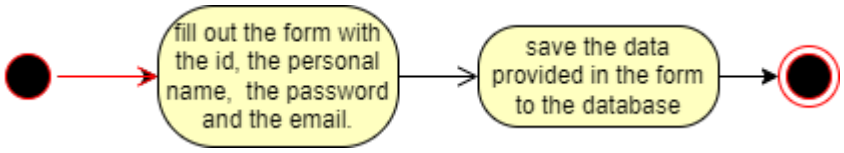
Deployment Diagram



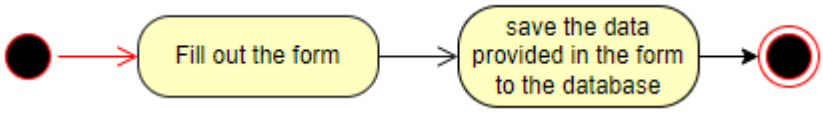
Login: This diagram shows how the user enters an email to log in. If the user is not registered, he is asked to create his account. On the other hand, if he is registered, he enters a password, and if it is correct, the user logs into the application.



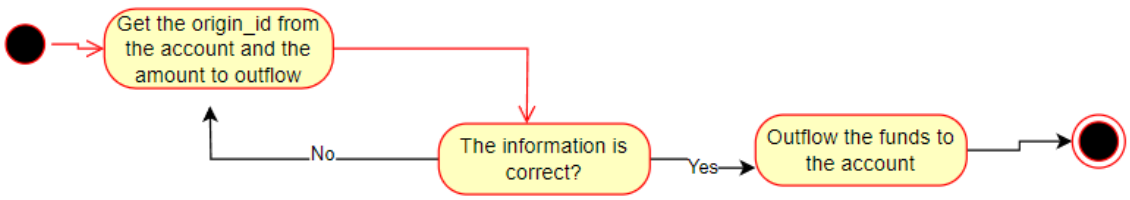
Create account: In this case, the user fills out the form with the ID, the personal name, the password and the email and this data is saved in the database.



Add address: In this diagram the user fills out the form to add the address and this information is saved in the database

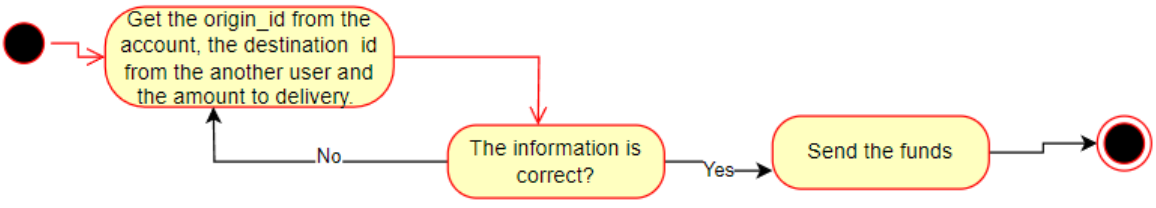


Outflow funds: In this diagram the system receives the account ID and the amount to be outflow, then the system requests verification of the data and when they are correct, the money is outflow from the account.

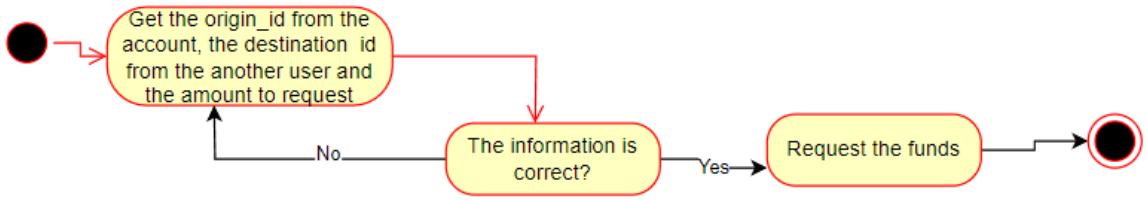


Send funds:

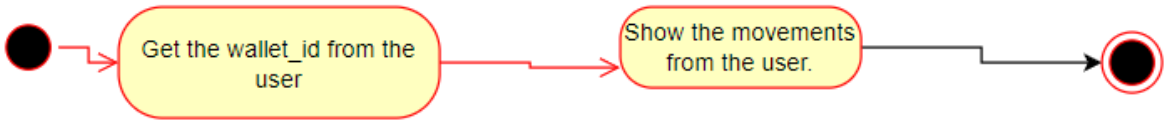
In this diagram the system receives the account ID, the amount to be send to another user and the id from the receiver user, then the system requests verification of the data and when they are correct, the money is sent to another account.



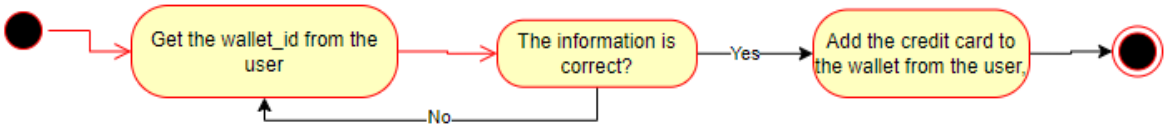
Request funds: In this diagram the system receives the ID of the account that sends the money, the amount to be requested from another user and the ID of the receiving user, then the system requests verification of the data and when they are correct the money is sent to another account.



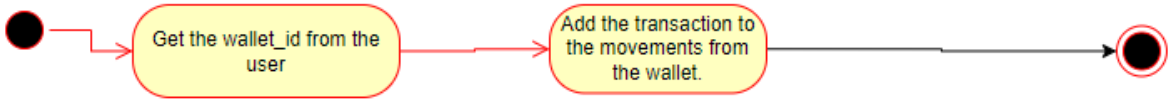
Show movements: In this diagram the wallet ID is received, and with it the movements made with the wallet are shown.



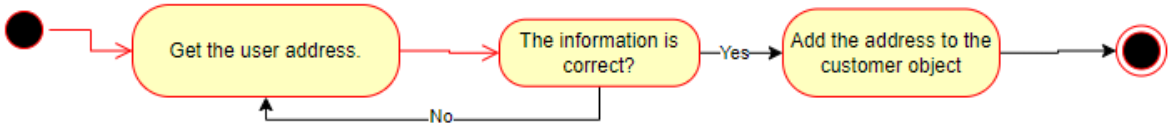
Add credit card: In this diagram the wallet id is received, and if the information is correct, the credit card is added to the user's wallet.



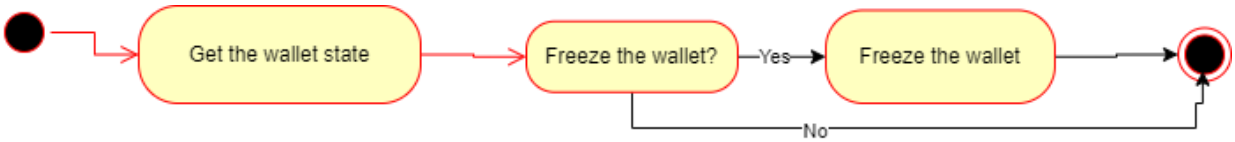
Add transactions to movements: In this diagram the wallet id is received, and if the information is correct, the transactions are sent to the wallet movements.



Add user address: In this diagram the wallet id is received, and if the information is correct, the transactions are sent to the wallet movements.

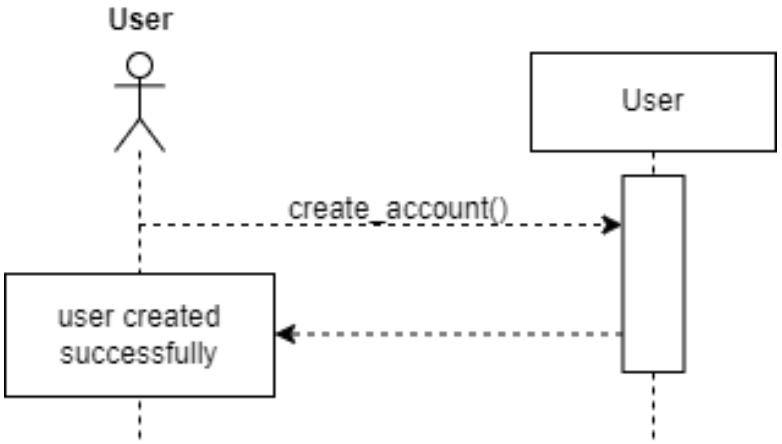


Freeze account: In this diagram the admin receives the wallet state and if it’s necessary, the admin freezes the account

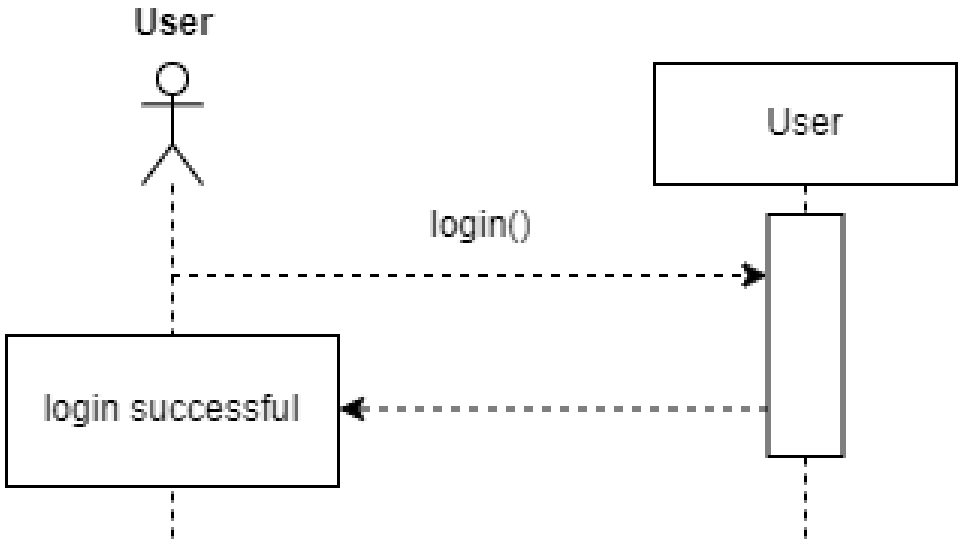


Sequence Diagrams

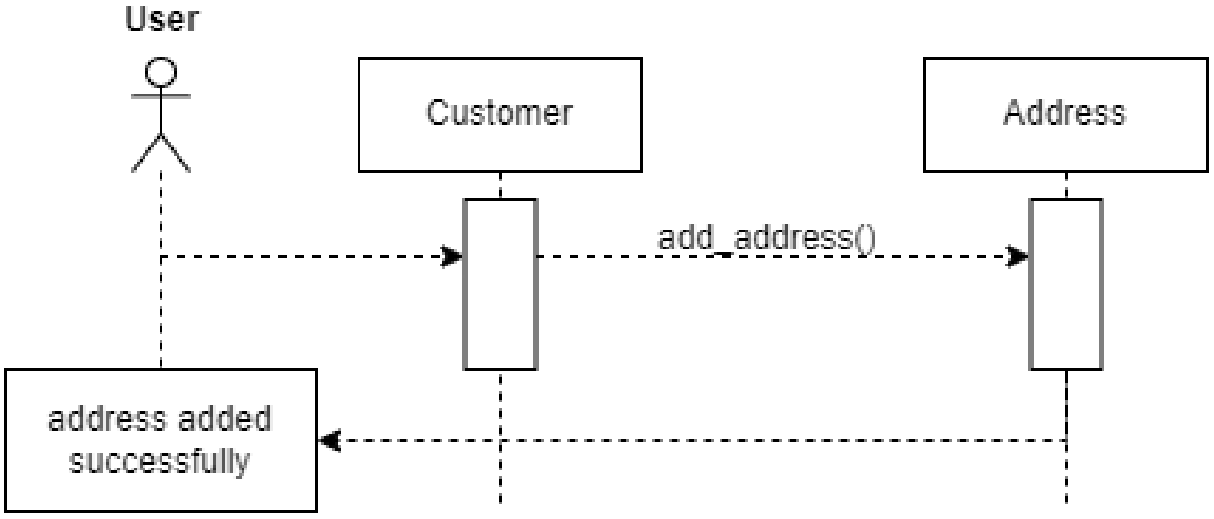
Create Account: This diagram shows how the customer tries to create an account, and the system returns a registration message if he has entered the expected data.



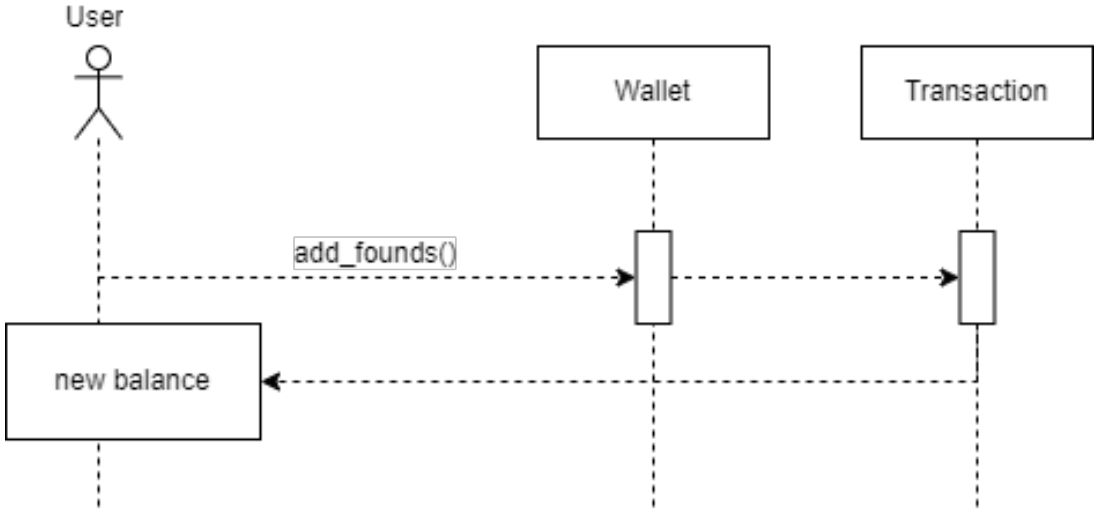
Login: This diagram shows how the user tries to log in to the system, and the system returns a message of validity or invalidity depending on the data entered.



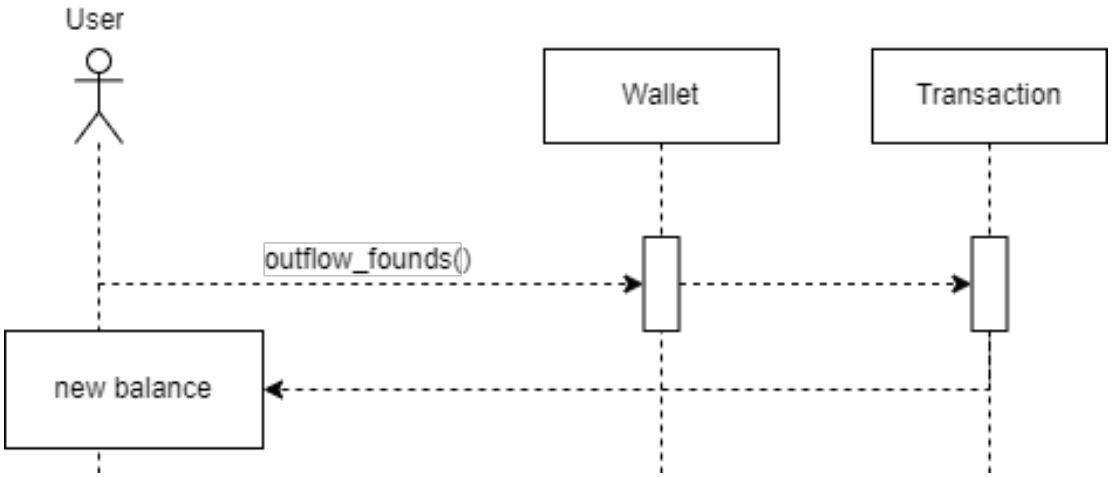
Add Address: This sequence diagram shows how the user add an address and the system response with a confirmation message.



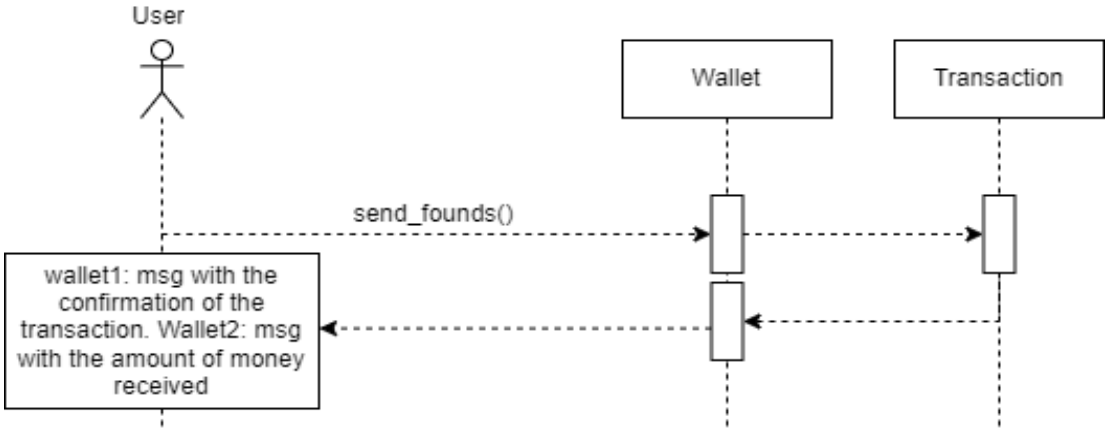
Add Founds: This sequence diagram shows the process of adding funds to the wallet, and the system response is the new wallet balance.



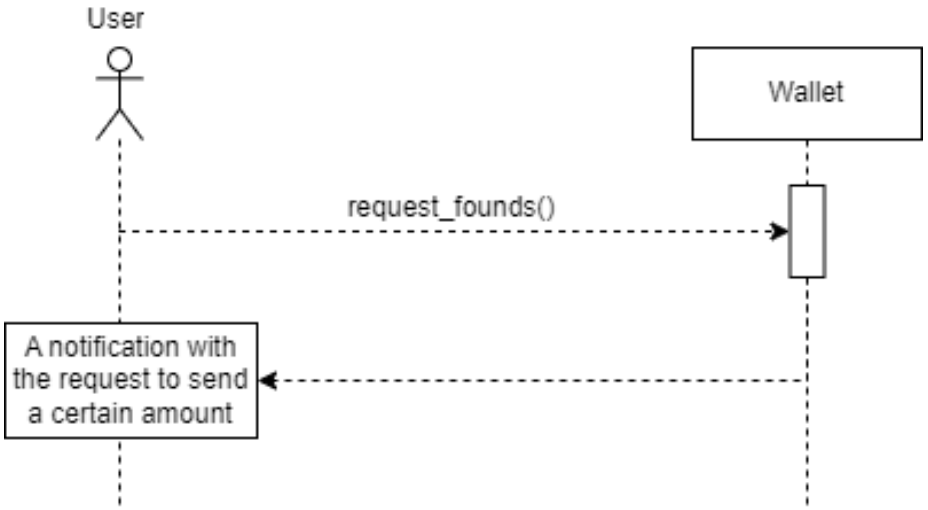
Outflow Founds: This sequence diagram shows the process of outflow Funds from the wallet and the response of the system that in this case is show the new balance



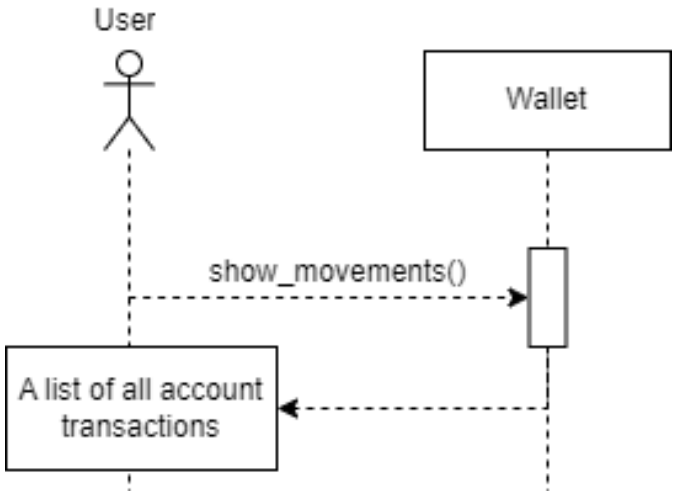
Send Founds: This sequence diagram shows the process of sending money between wallets and the system response consisting of a message with the status of the transaction for the person sending the money and a message with the amount of money received for the person to whom the transaction was directed.



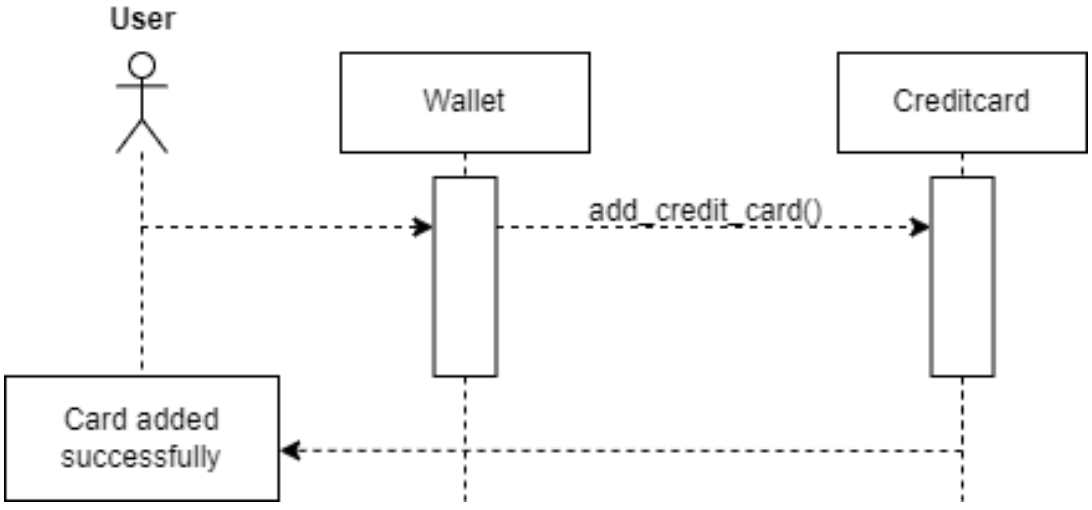
Request Founds: This sequence diagram shows the process of request money between user of the application.



Show movements: This sequence diagram shows the process to watch the history of movements from the account.



Add Credit Card: This sequence diagram shows the process to add a new credit card linked to the wallet.



4. CRC Cards

User	
Responsibility	Collaborators
Store and validate application user data like name, wallet_id and email. Log in the user in the application.	Customer Wallet

Customer	
Responsibility	Collaborator
Save and validate the customer data.	User

Transactions	
Responsibility	Collaborator
Validate the structure of transaction data and save it in the database.	Wallet

Credit card	
Responsibility	Collaborator
Validate the structure of credit card data	Wallet

Wallet	
Responsibility	Collaborator
Have the wallet information from the user. Have the credit card information. Develop the adding and sending of money. Show the movements from the wallet	User Transactions Credit card

Address	
Responsibility	Collaborator
Validate the integrity and structure of data related at credit card	Wallet

Admin	
Responsibility	Collaborator
Manage users and generate reports on various user activities in the application.	Wallet User Transactions

5. Technical decisions

In terms of technical decisions, it was decided to structure the project under the idea of a layered architecture, in order to develop a scalable, maintainable and flexible system.

In this way, the application will be made up of a backend that will have all the logic for the operation of the software, as well as the database layer. The backend will be made in two programming languages, Python and Java. On the other hand, it was decided that the frontend will be a command line interface, because the project is intended to focus on the use of design patterns and good practices of object-oriented programming.

In the same way, the goal is to use GraphQL to easily unify the backends made with Java and Python, as well as to have a complete tool for managing the application's services.

Finally, we decided to use Docker to perform the software integration, as well as to automate the deployment of the application.