UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Universidad Distrital Francisco José de Caldas

Facultad de ingeniería

Ingeniería en sistemas

programming models I

David Martínez cod: 20222020167

Julian David Pulido cod: 20231020064

4 de Octubre de 2024

Technical Report

User stories

- As a user, I want to log in using my email and password so that I can access my digital wallet and perform transactions.
- As a user, I want to view my account balance so that I can know how much money I have available before making a transaction.
- As a user, I want to deposit funds into my account so that I can increase my available balance in the wallet.
- As a user, I want to withdraw funds from my account so that I can transfer money from my wallet to an external bank or account.
- As a user, I want to view my transaction history so that I can track all the transactions I have made over time.
- As a user, I want to know each transaction in the transaction history so that know where my money went.
- As a user, I want to receive a notification after each transaction so that I can be informed whether it was successful or not.
- As a user, I want to check the status of a transaction so that I can confirm whether it was completed, pending, or failed.
- As a user, I want to reset my password via email so that I can regain access to my account if I forget my login details.
- As a user, I want to update my profile details like phone number and address so that my personal information stays up to date in the system.
- As a user, I want to log out of the application so that my session is securely terminated.
- As a user, I want to mark notifications as read so that I can organize my notification history and know which ones are new.
- As a user, I want to deactivate my account so that I can close my digital wallet and stop using the service if I choose.

Object-oriented principles analysis

Encapsulation

Encapsulation is the principle that enforces restricting access to an object's internal state and only allowing access through well-defined interfaces (methods). In the provided class diagram:

- Each class encapsulates its data (attributes) and behavior (methods).

- For example, the Account class has private attributes like id and balance, and the behavior to access these details is through methods like getAccountDetails().

- Similarly, Transaction has its attributes (id, amount, date, and status) and provides methods like getTransactionDetails() and cancelTransaction(), which control how the transaction details are accessed or modified.

By using encapsulation, the internal details of the class (like balance updates or transaction statuses) are hidden from the outside, promoting modularity and maintainability.

Inheritance

Inheritance allows a class to inherit properties and behavior from another class, promoting reusability.

For example:

- Send and Withdraw share commonalities with Transaction since they are specialized transaction types. A improvement was to create a base class TransactionType, from which Send, Withdraw and inherit common behavior.

- Similarly, if there are multiple types of notifications EmailNotification or SMSNotification, they could inherit from a common Notification base class.

Polymorphism

Polymorphism allows objects of different types to be treated as objects of a common super type.

As Withdraw, and send are subclasses of Transaction, polymorphism could be used to treat them all as Transaction objects, allowing more flexible and extensible transaction handling. For example, a method could accept any type of Transaction without needing to know the specific type (Send or Withdraw).

Single Responsibility Principle (SRP)

The SRP states that a class should only have one reason to change, meaning each class should focus on one specific responsibility. In this case

- Authentication handles login, logout, and password management.

- Transaction is responsible for transaction-specific behavior (details, cancellation, success checks).

- Notification is responsible for sending and managing notifications.

- Account deals with balance and account details.

By adhering to SRP, each class remains focused on a single task, making the system easier to maintain, understand, and extend.

Liskov Substitution Principle (LSP)

The Liskov Substitution Principle ensures that objects of a subclass should be replaceable with objects of a superclass without affecting the correctness of the program. LSP can be considered in inheritance into the model, with Transaction as a base class for Send or Withdraw these should be interchangeable with the Transaction class, ensuring the system continues to function correctly regardless of the specific transaction type.

CRC cards

User

Responsabilidades:	Colaboradores
• Manejar la autenticación del usuario • Permitir el cambio de contraseña. • Desactivar la cuenta del usuario	`Profile`: Para acceder a los detalles del perfil del usuario. `Authentication`: Para la validación de credenciales.

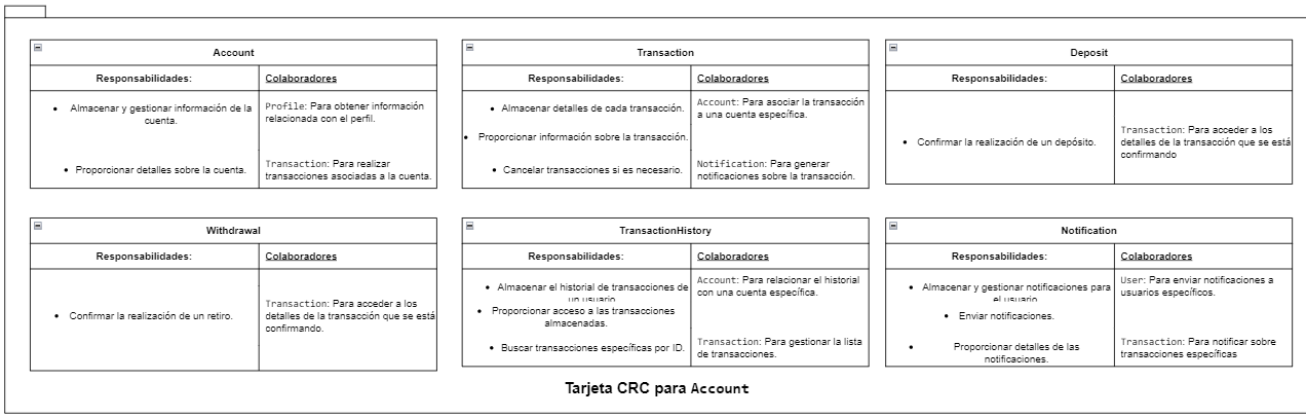
Profile

Responsabilidades:	Colaboradores
• Almacenar y gestionar los detalles del perfil del usuario. • Actualizar la información del perfil • Validar el número de teléfono del usuario.	`User`: Para obtener el email y estado de la cuenta.

Authentication

Responsabilidades:	Colaboradores
• Manejar el inicio y cierre de sesión del usuario. • Gestionar la recuperación de contraseñas. • Validar si un usuario está actualmente autenticado.	`User`: Para verificar las credenciales del usuario

Tarjetas CRC para User Service



Account

Responsabilidades:	Colaboradores
• Almacenar y gestionar información de la cuenta. • Proporcionar detalles sobre la cuenta.	`Profile`: Para obtener información relacionada con el perfil. `Transaction`: Para realizar transacciones asociadas a la cuenta.

Transaction

Responsabilidades:	Colaboradores
• Almacenar detalles de cada transacción. • Proporcionar información sobre la transacción. • Cancelar transacciones si es necesario.	`Account`: Para asociar la transacción a una cuenta específica. `Notification`: Para generar notificaciones sobre la transacción.

Deposit

Responsabilidades:	Colaboradores
• Confirmar la realización de un depósito.	`Transaction`: Para acceder a los detalles de la transacción que se está confirmando

Withdrawal

Responsabilidades:	Colaboradores
• Confirmar la realización de un retiro.	`Transaction`: Para acceder a los detalles de la transacción que se está confirmando.

TransactionHistory

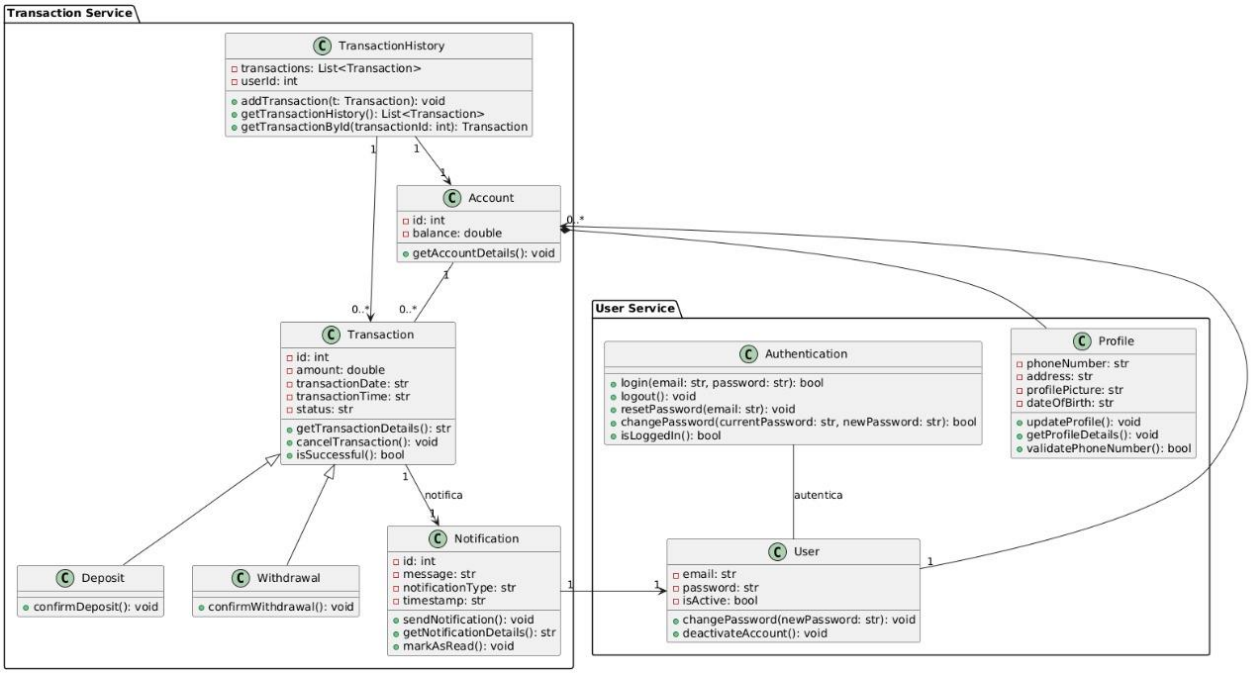
Responsabilidades:	Colaboradores
• Almacenar el historial de transacciones de un usuario. • Proporcionar acceso a las transacciones almacenadas. • Buscar transacciones específicas por ID.	`Account`: Para relacionar el historial con una cuenta específica. `Transaction`: Para gestionar la lista de transacciones.

Notification

Responsabilidades:	Colaboradores
• Almacenar y gestionar notificaciones para el usuario. • Enviar notificaciones. • Proporcionar detalles de las notificaciones.	`User`: Para enviar notificaciones a usuarios específicos. `Transaction`: Para notificar sobre transacciones específicas

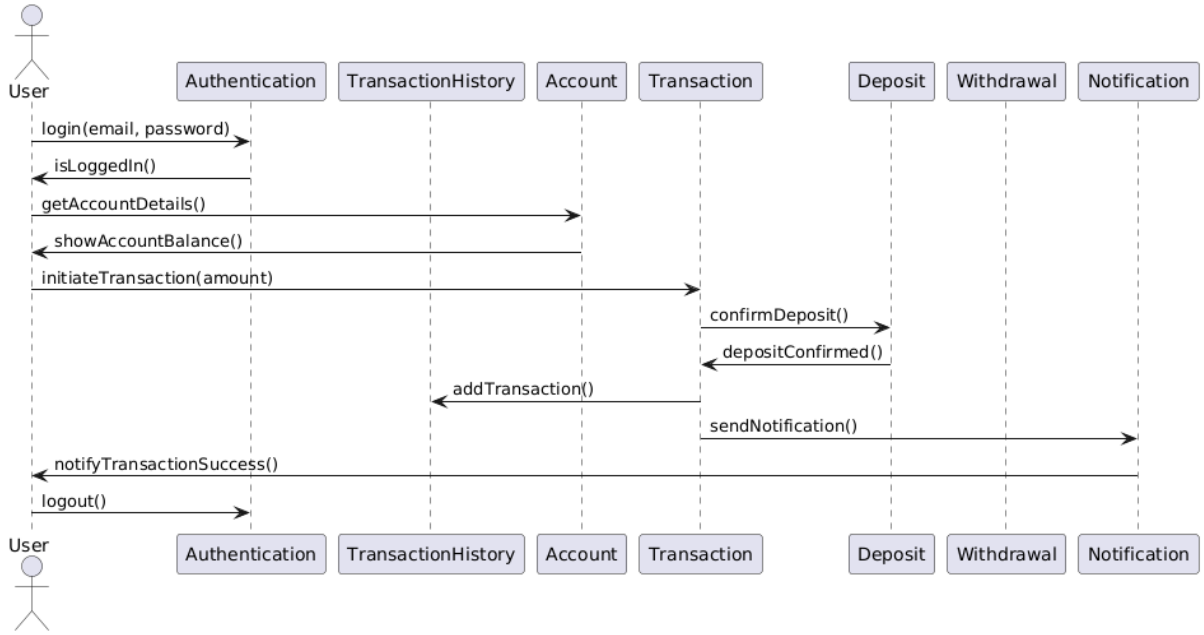
Tarjeta CRC para `Account`

Class diagram

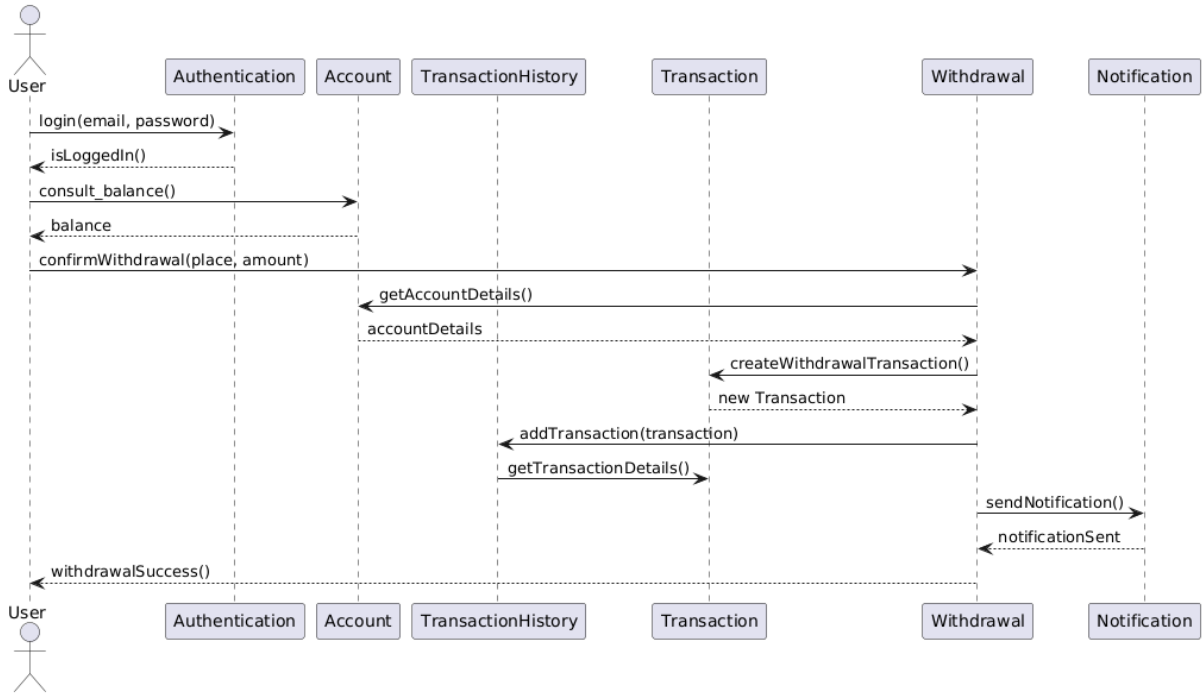


Sequence diagrams

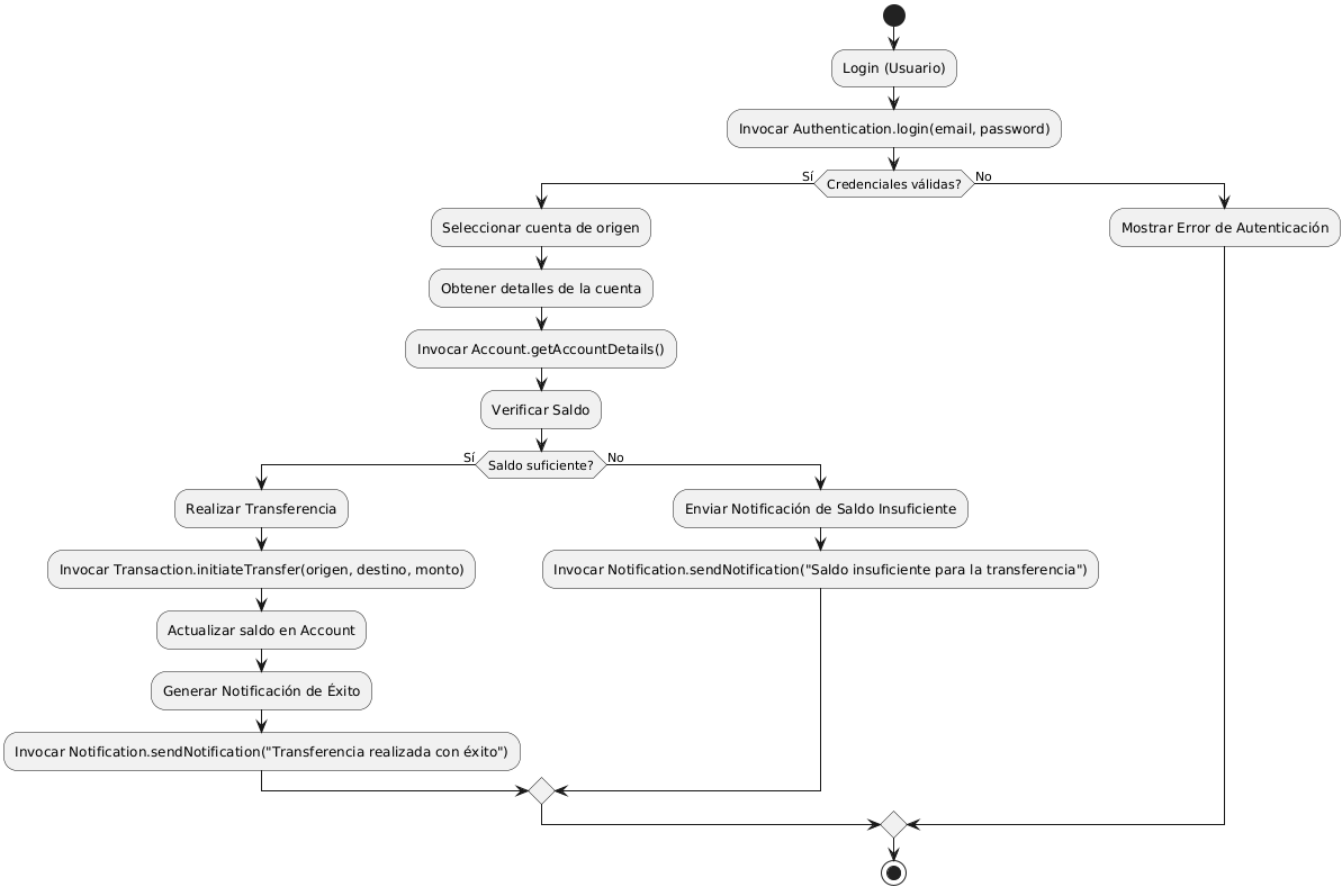
Example with a deposit



Example with a withdrawal



activity diagram



Deployment diagram