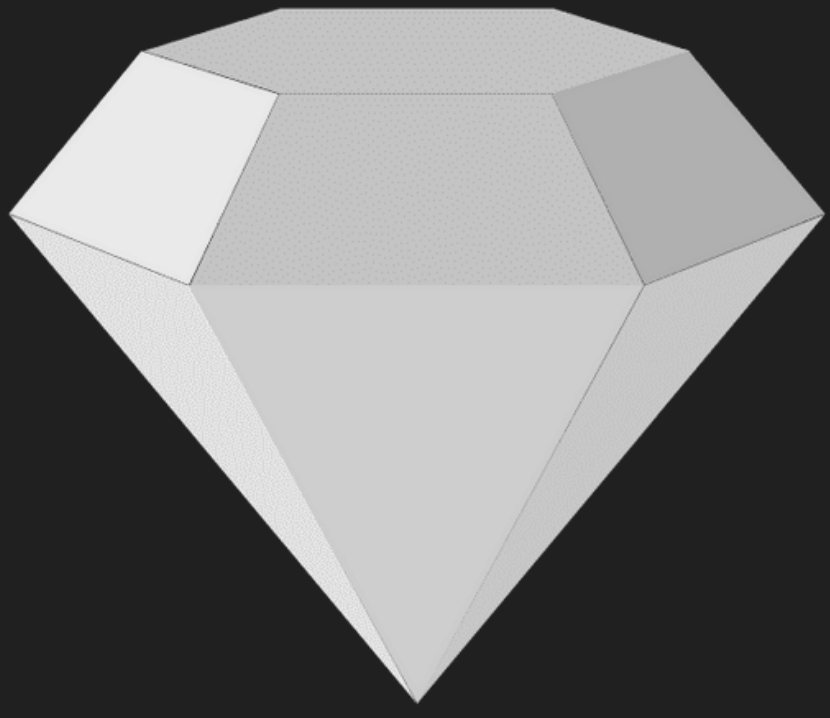
hELPP

a crypto market developed in java



Giorgio Zucconelli

hElpp is a fictional crypto market where users can buy and sell cryptocurrencies with real-time data updates on prices

Software Requirements: java 19+ installed.
Hardware Requirements: no particular hardware required to run this application

Related Systems:
Etoro, bitstamp
pros: more functionalities, more secure, better ui
cons: ...

User Stories:
1) As a crypto investor, I want to browse a variety of cryptocurrencies and their current market prices, so that I can make investments based on real-time data

2) As a crypto holder, I want to monitor the cryptocurrencies that I own, so that I can keep a close watch on my investments

3) As a financial advisor, I want to have followers who view and copy my investments, so that I can grow in popularity and reputation *

*** = not implemented**

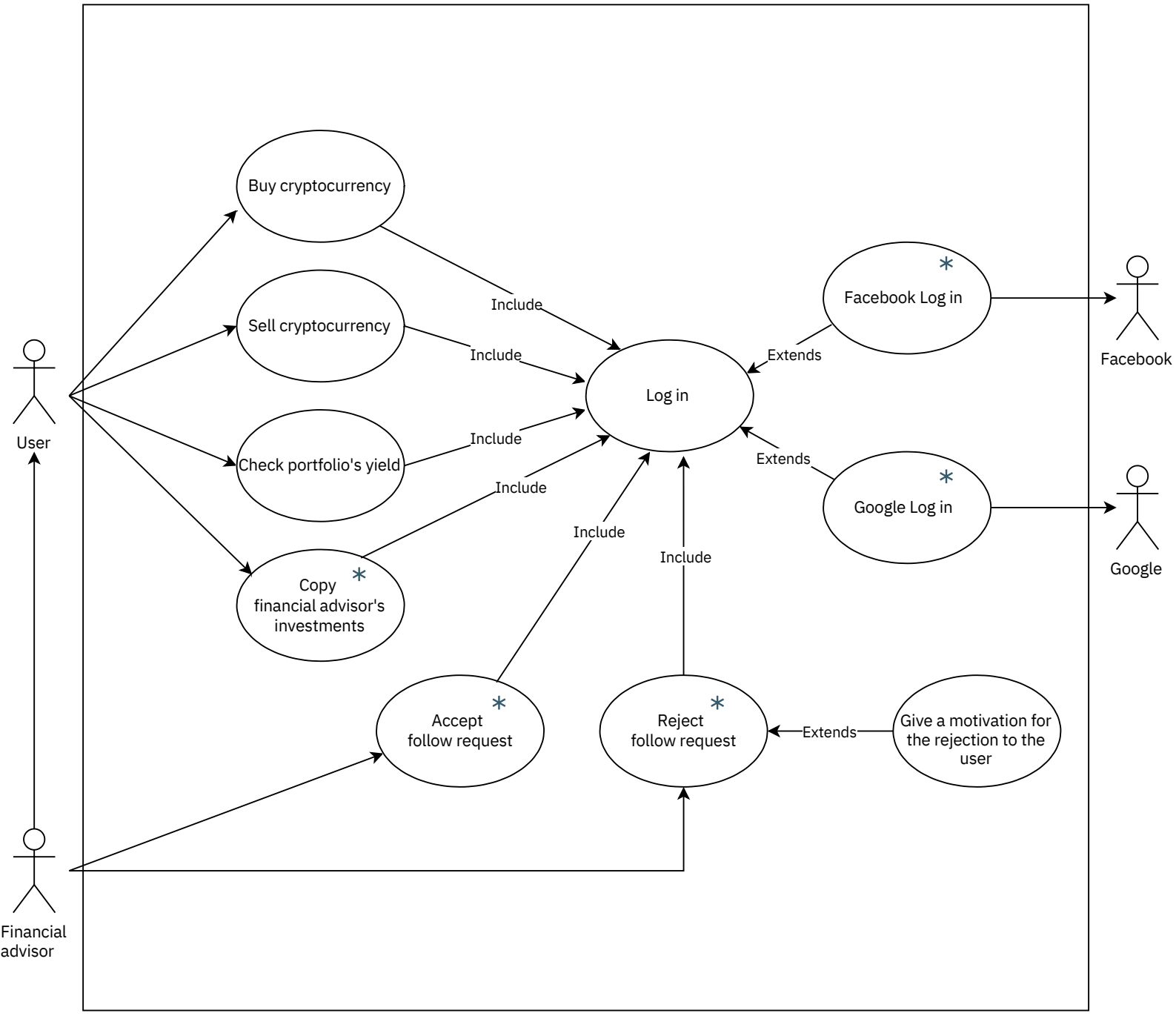
Functional Requirements:

1) The system shall provide multiple payment methods for purchasing cryptocurrencies, such as credit/debit cards and bank transfers *

2) The system shall display a list of available cryptocurrencies along with their current market prices

3) The system shall check the user's funds before proceeding with purchase and withdrawal operations

Use Case Diagram



StoryBoards

Login Page

Username:

Password:

Log in

Don’t have an account? Register

Main Menu Page

Profile

Buy Crypto

Portfolio

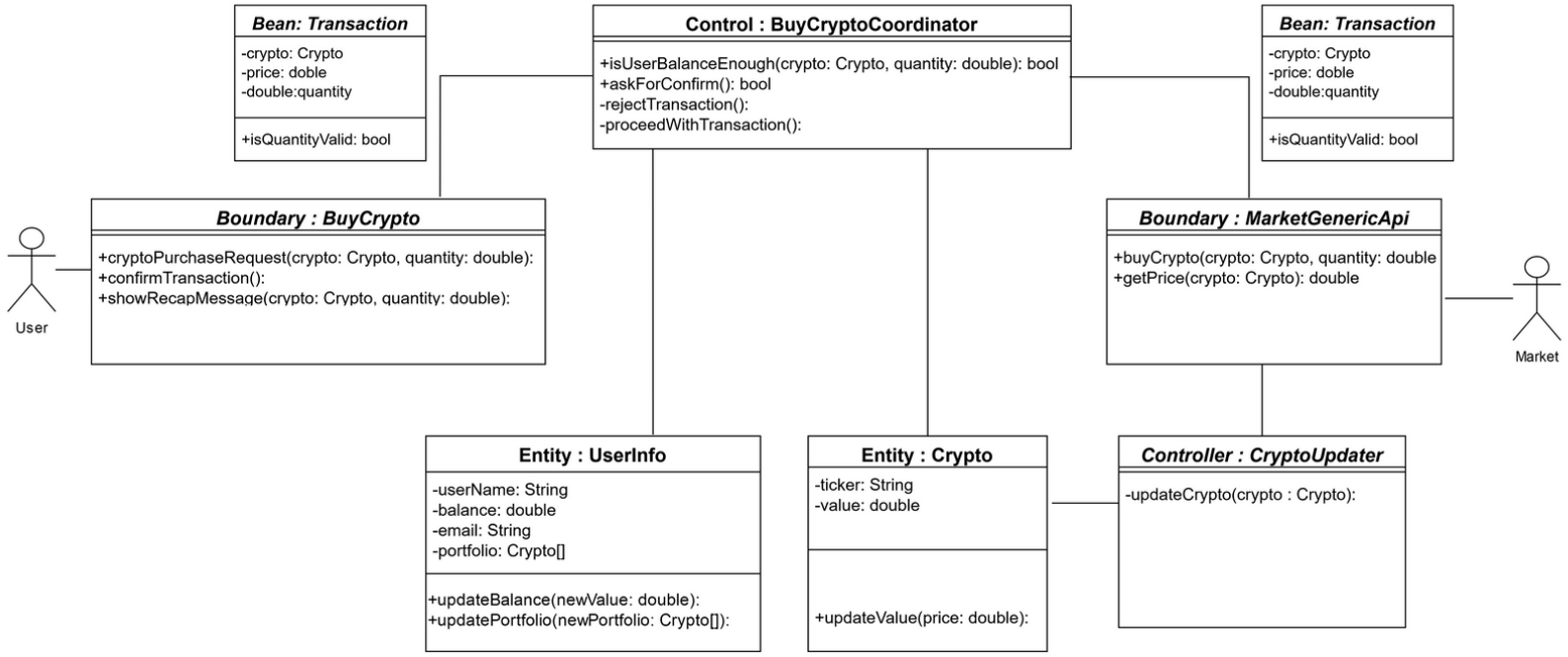
Deposit

Withdraw

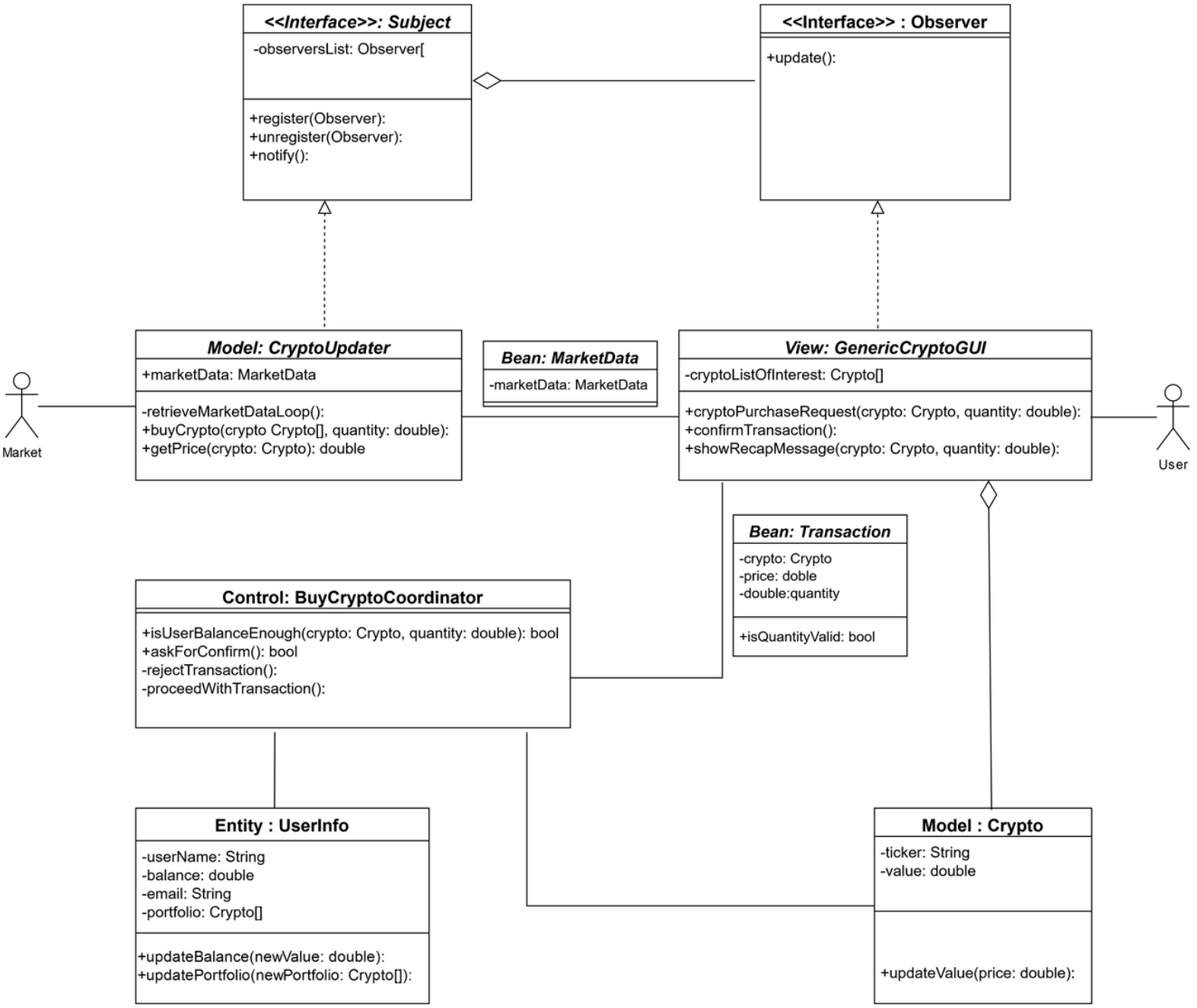
Balance: 1000 €

hELPP

Bce diagram


Bean: Transaction
-crypto: Crypto
-price: doble
-double:quantity
+isQuantityValid: bool
Control : BuyCryptoCoordinator
+isUserBalanceEnough(crypto: Crypto, quantity: double): bool
+askForConfirm(): bool
-rejectTransaction():
-proceedWithTransaction():
Bean: Transaction
-crypto: Crypto
-price: doble
-double:quantity
+isQuantityValid: bool
User
Boundary : BuyCrypto
+cryptoPurchaseRequest(crypto: Crypto, quantity: double):
+confirmTransaction():
+showRecapMessage(crypto: Crypto, quantity: double):
Boundary : MarketGenericApi
+buyCrypto(crypto: Crypto, quantity: double
+getPrice(crypto: Crypto): double
Market
Entity : UserInfo
-userName: String
-balance: double
-email: String
-portfolio: Crypto[]
+updateBalance(newValue: double):
+updatePortfolio(newPortfolio: Crypto[]):
Entity : Crypto
-ticker: String
-value: double
+updateValue(price: double):
Controller : CryptoUpdater
-updateCrypto(crypto : Crypto):

Mvc diagram



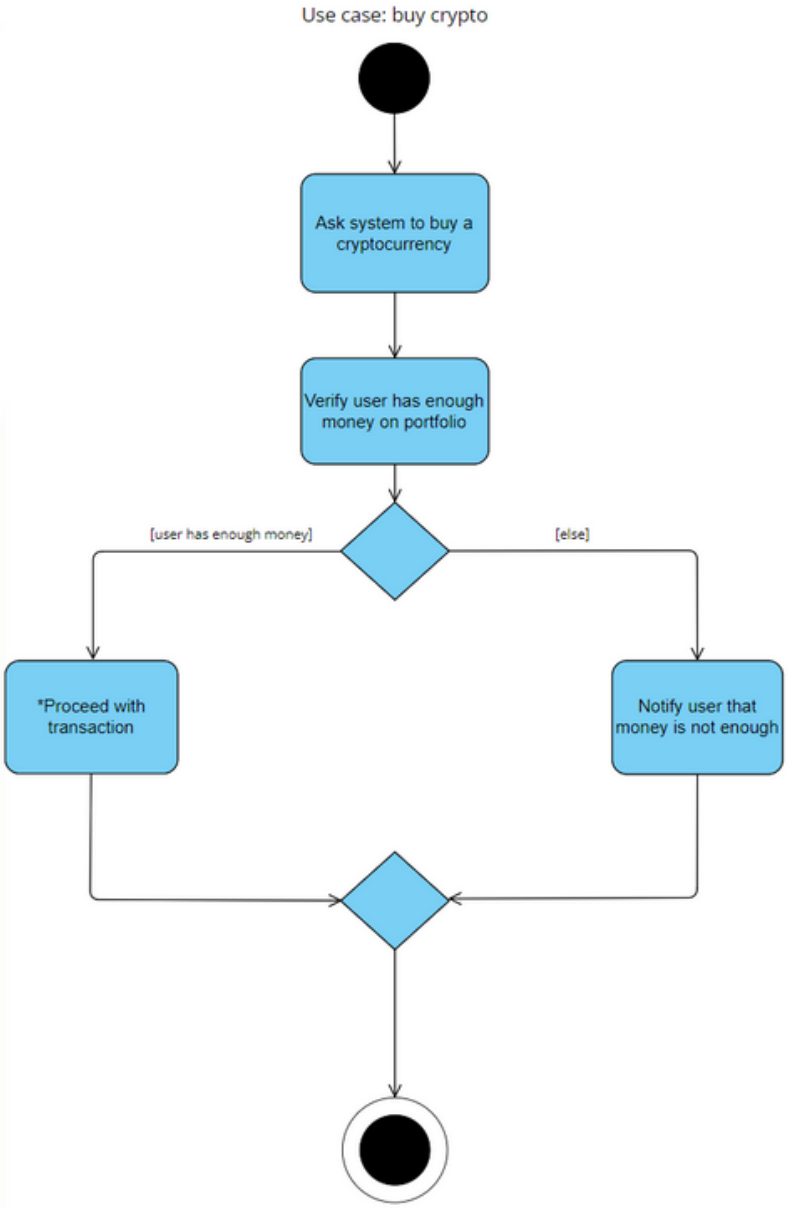
Primary pattern: Observer

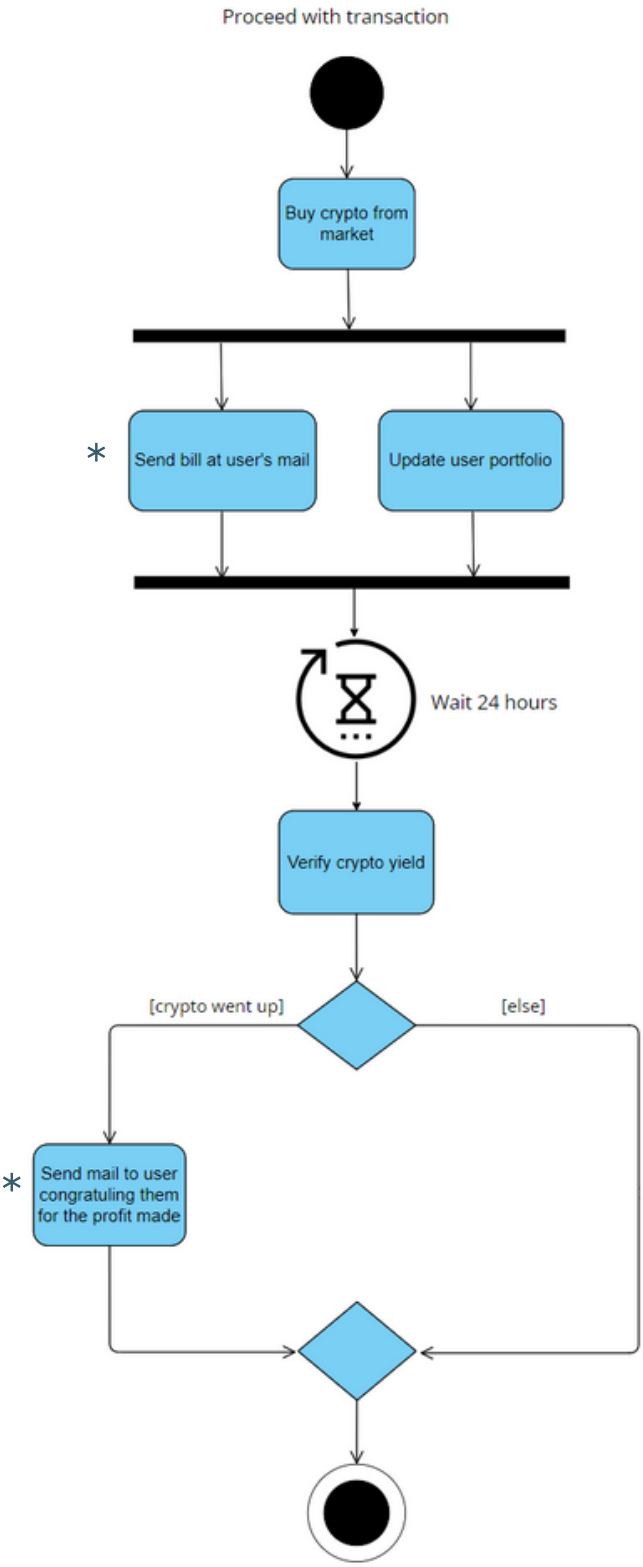
Secondary patterns used:

Strategy, Singleton

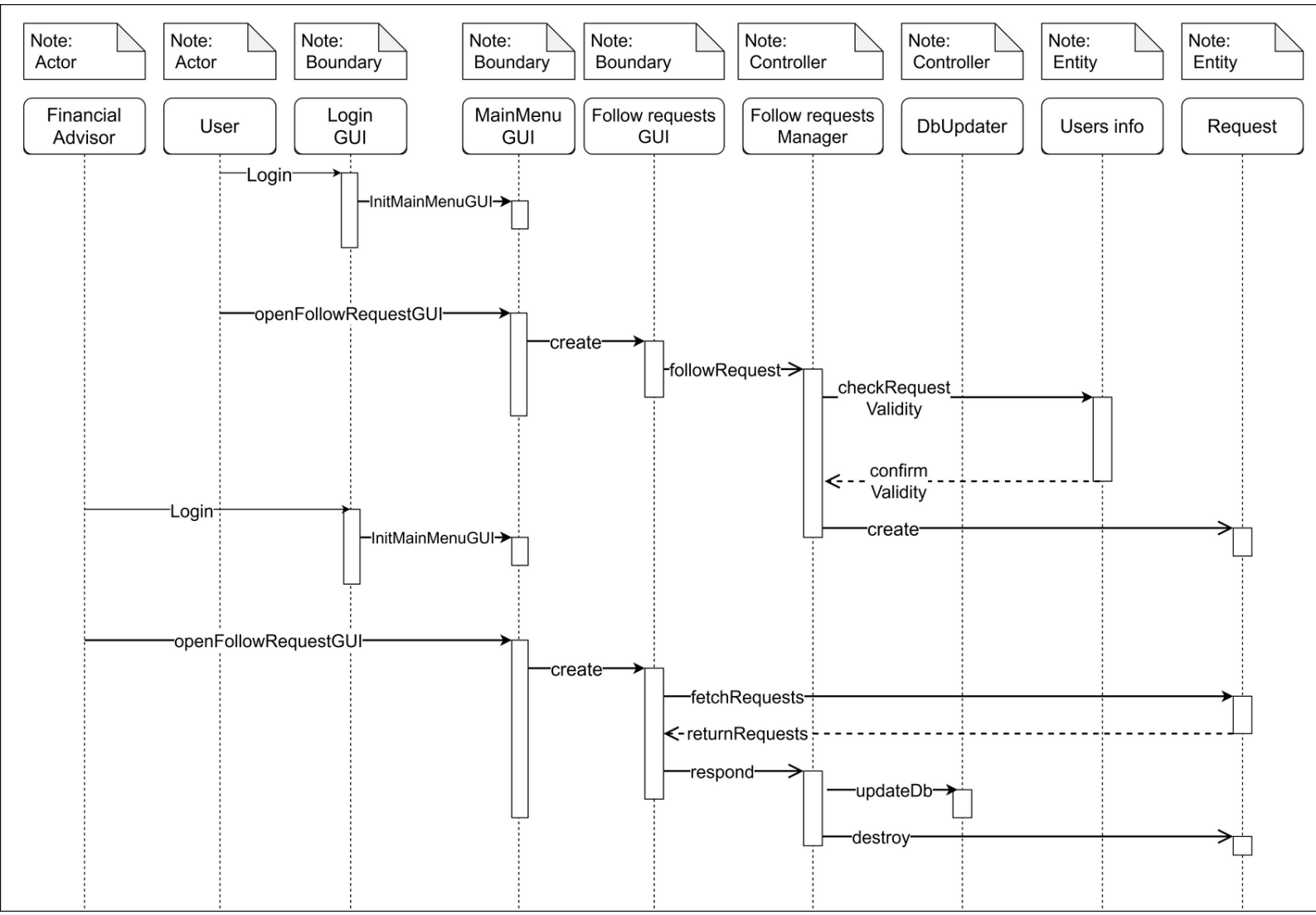
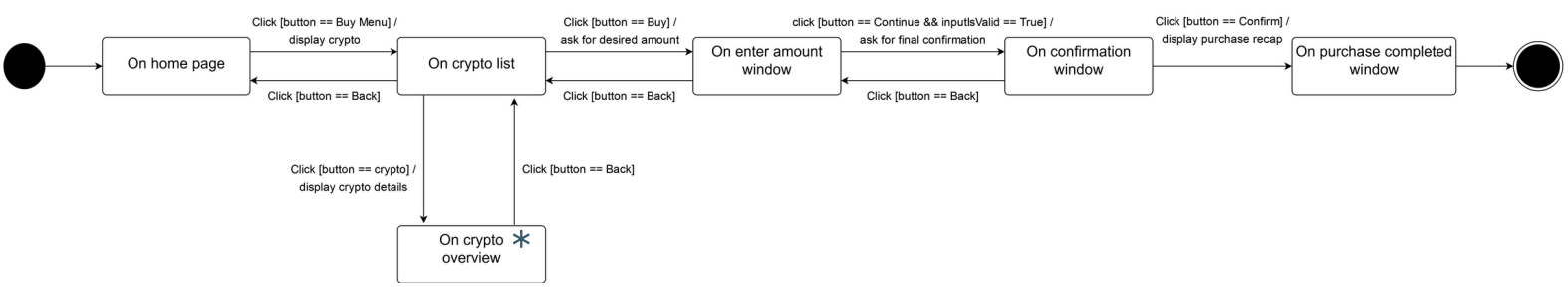
(in the project, not shown in this diagram)

Activity diagram





Sequence diagram



State diagram