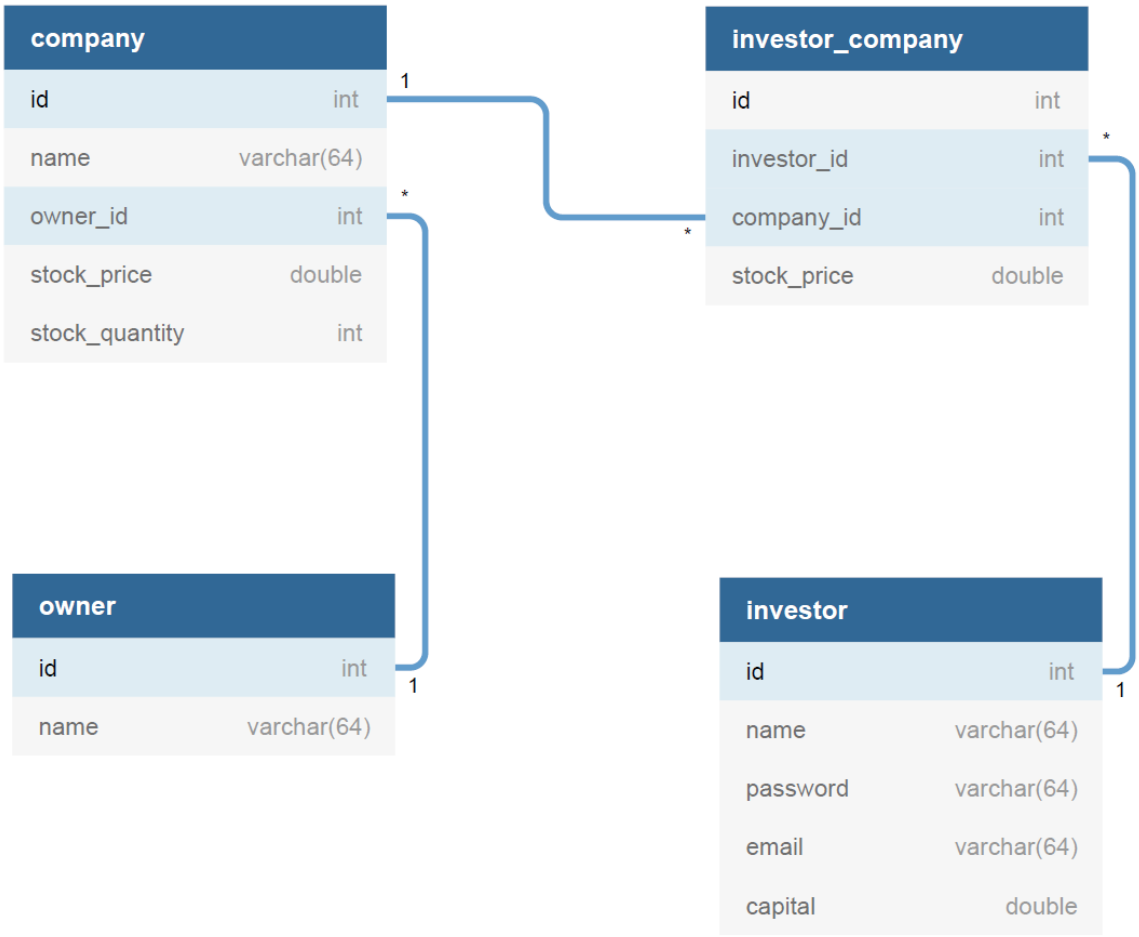
Martuneac
Alexandru
30431

Stock market platform

Analysis and Design Document

I. Elaboration – Iteration 1.1

Domain Model



Architectural Design

Conceptual Architecture

For the back-end, a layered architecture will be implemented to better organize and differentiate between entities, services, business logic, etc. Furthermore, abstract factory (creational pattern) will be present, since it allows to easily switch between the concrete implementation of the objects instantiated. Also, it is easier to write, understand, test and extend.

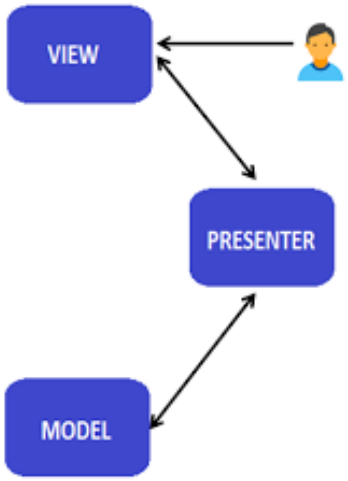
Observer pattern will be implemented to announce the investors that a new company was listed.

Abstract factory pattern for creating families of related objects without specifying their type. This allows us to change the implementation of different classes, in case a new technology appears or we have to use multiple implementations.

The front-end will use a variation of the famous MV* design pattern, more specifically MVP (model-view-presenter).

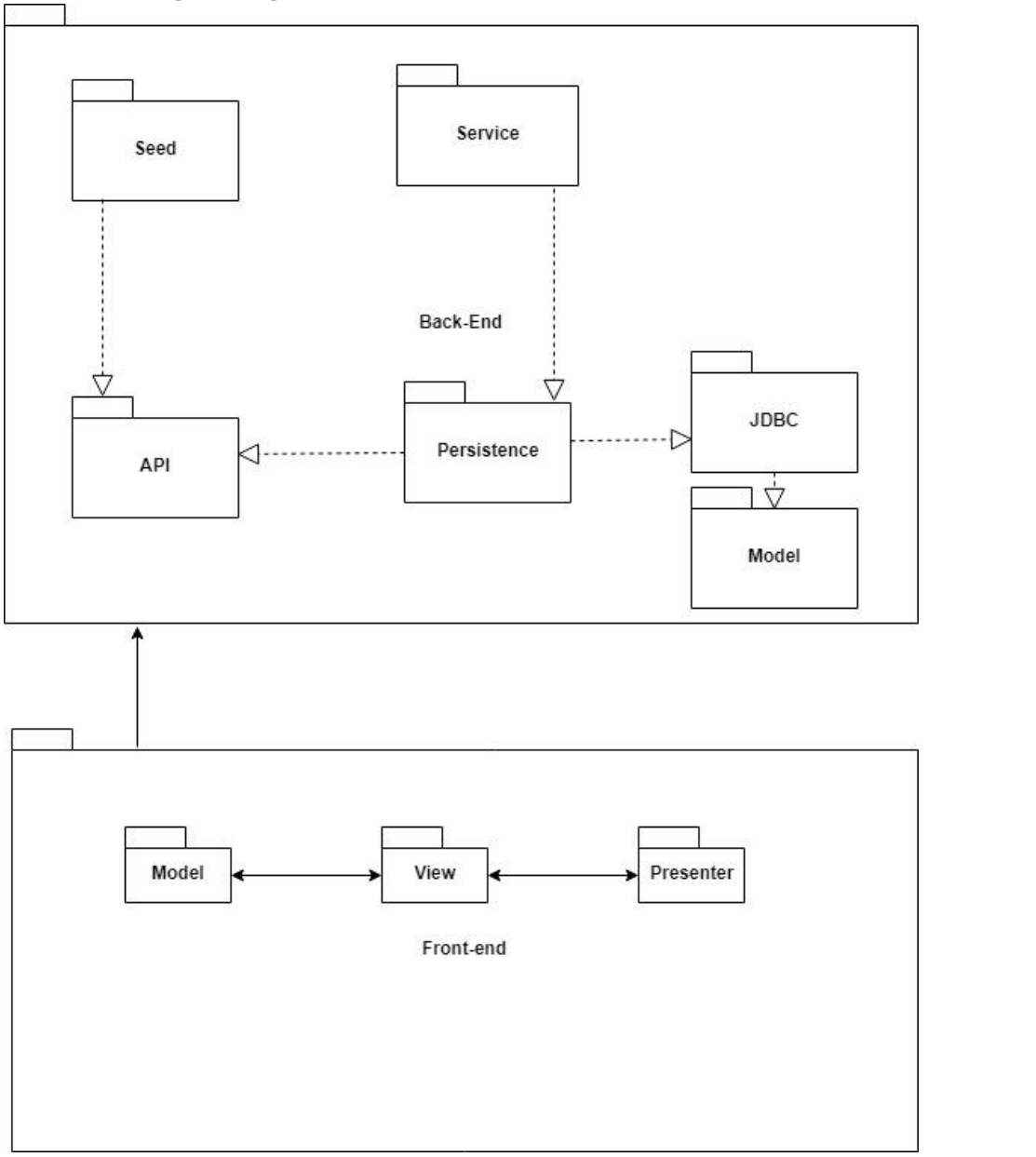
The Presenter receives the input from users via View, then process the user's data with the help of Model and passing the results back to the View.

Presenter communicates with view through interface. Interface is defined in presenter class, to which it passes the required data.

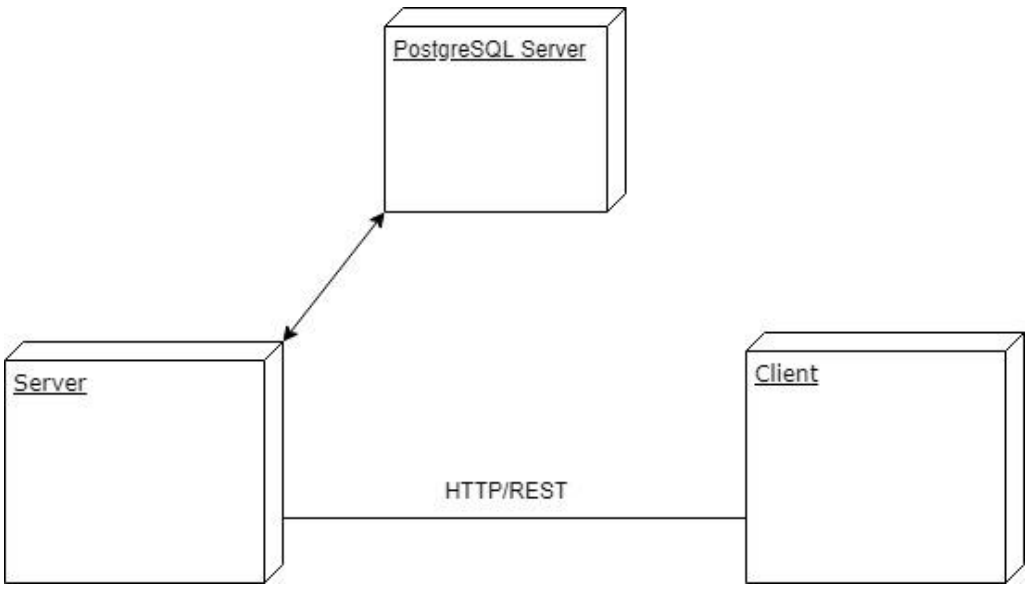


MVP

Package Design



Component and Deployment Diagrams

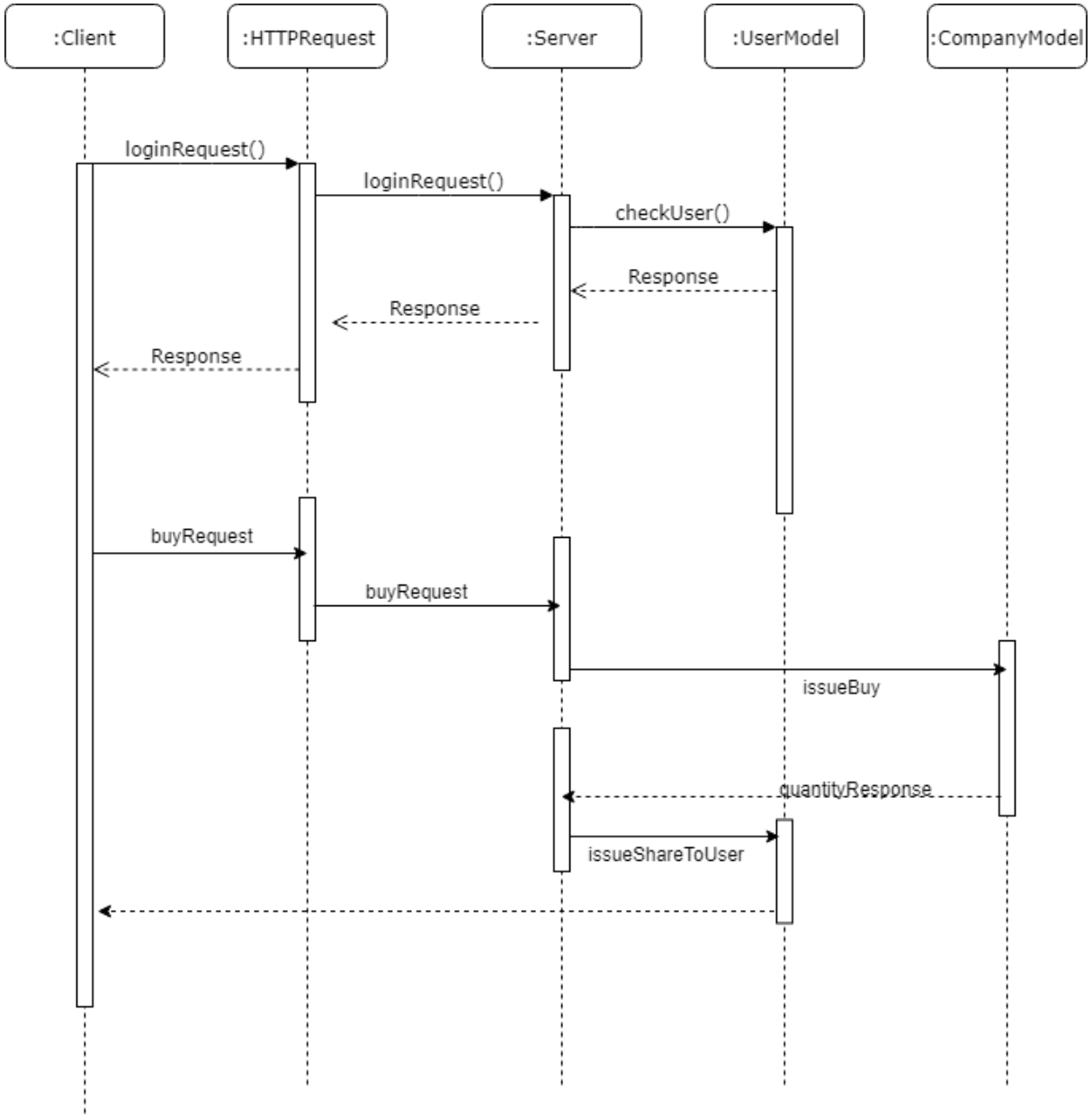


II. Elaboration – Iteration 1.2

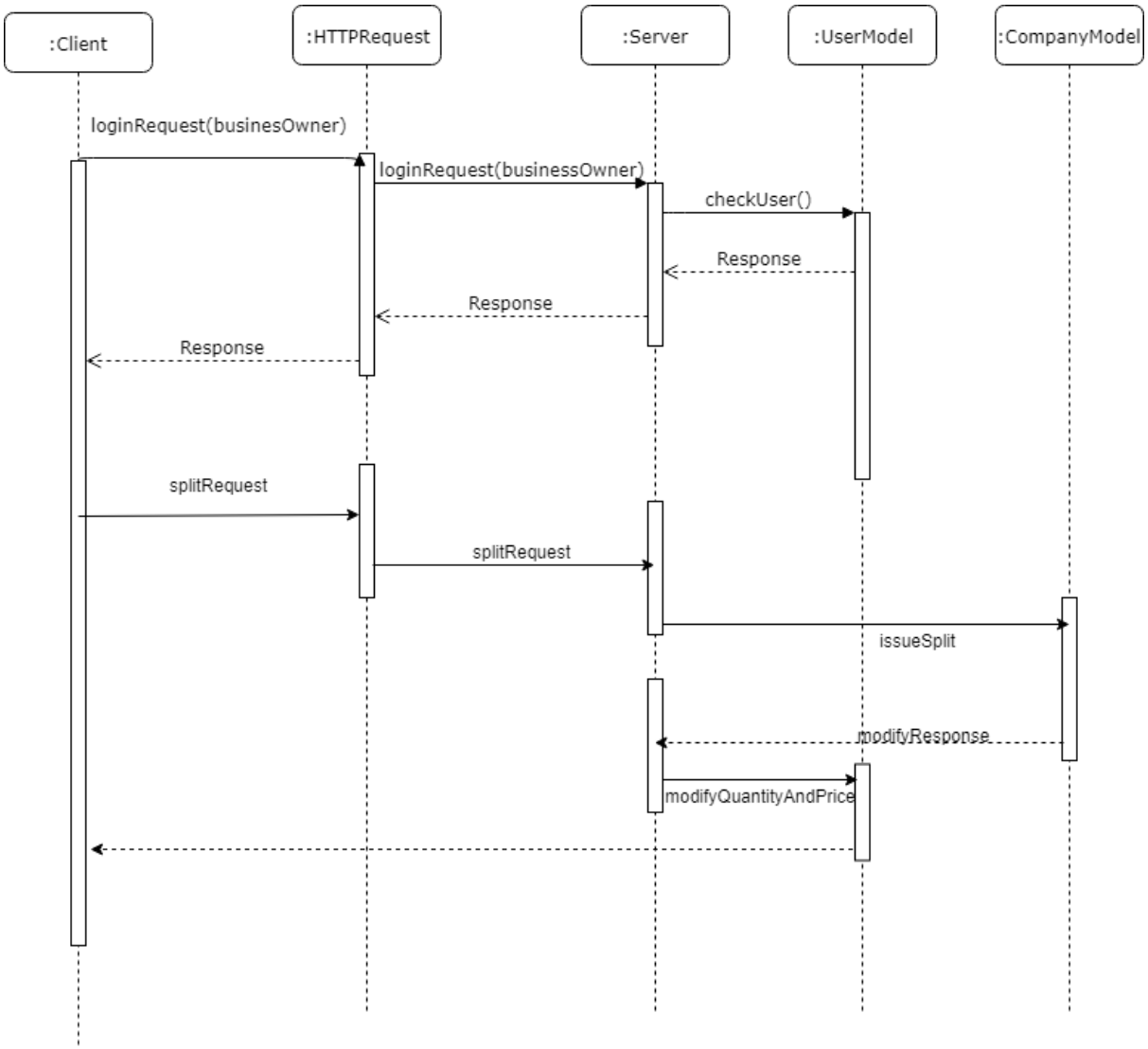
Design Model

Dynamic Behavior

1. *Investor authenticates and performs a buy.*

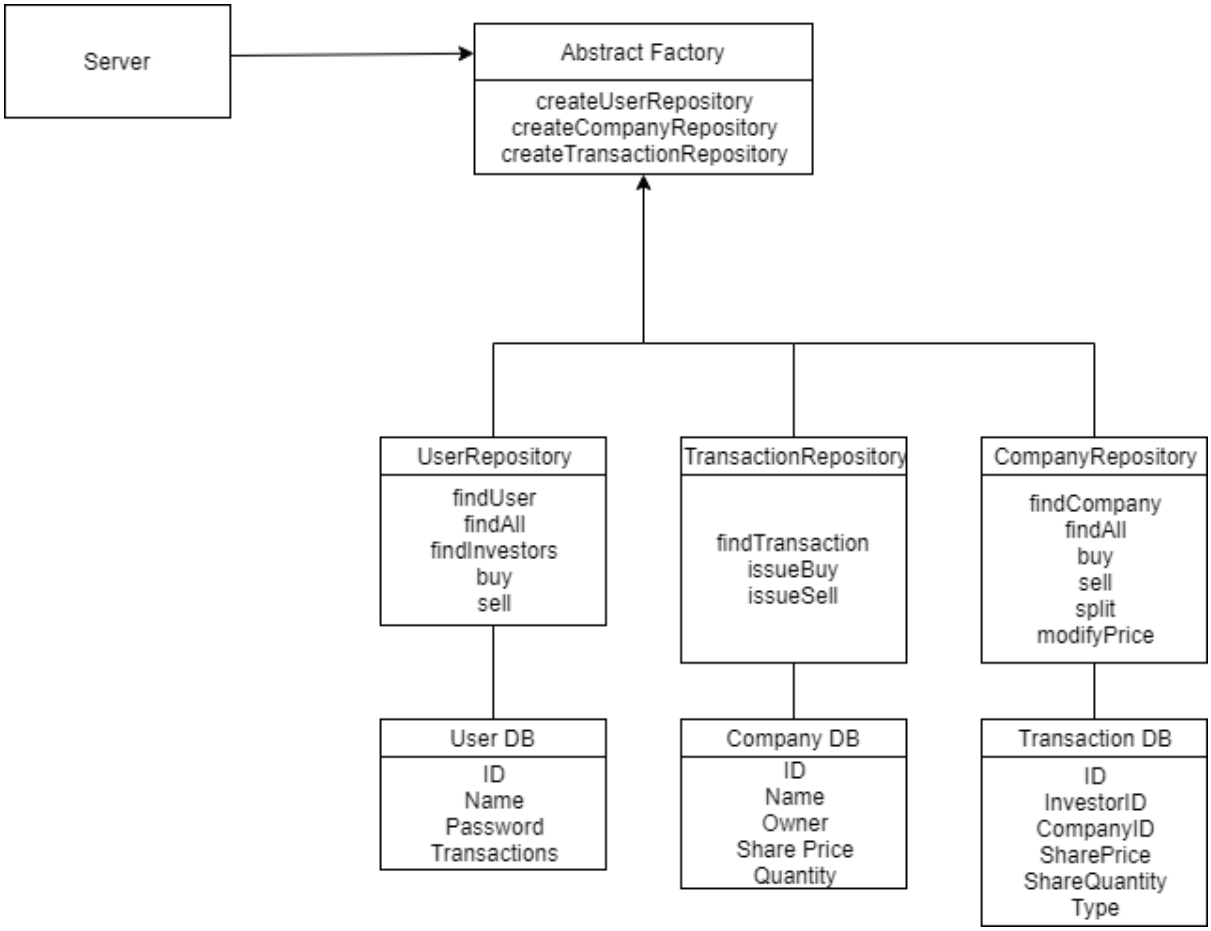


2. *Business owner authenticates and issues a split.*

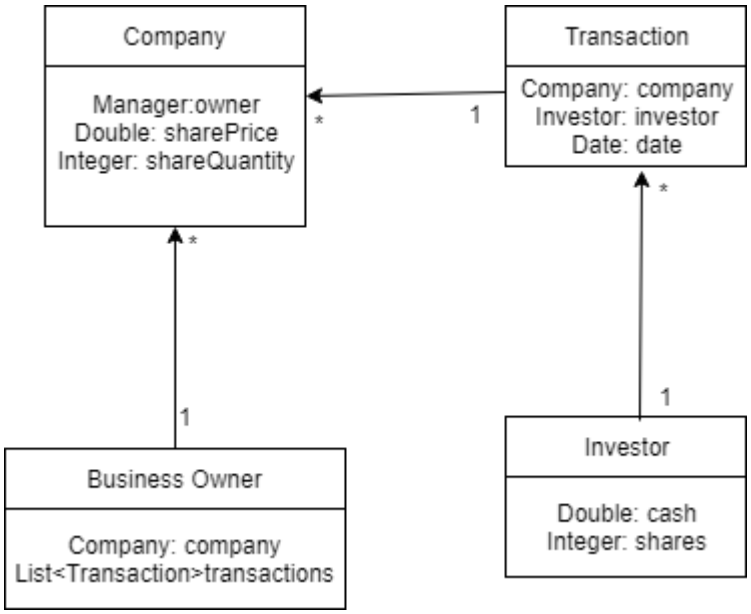


Class Design

The pattern used will be the repository pattern. It uses an abstract factory to identify between different repos and allows for a convenient access to the data. Moreover, repository has the job to deliver to the business layer the required objects.


Server
Abstract Factory
createUserRepository
createCompanyRepository
createTransactionRepository
UserRepository
findUser
findAll
findInvestors
buy
sell
TransactionRepository
findTransaction
issueBuy
issueSell
CompanyRepository
findCompany
findAll
buy
sell
split
modifyPrice
User DB
ID
Name
Password
Transactions
Company DB
ID
Name
Owner
Share Price
Quantity
Transaction DB
ID
InvestorID
CompanyID
SharePrice
ShareQuantity
Type

1. Data Model


Company
Manager:owner
Double: sharePrice
Integer: shareQuantity
Transaction
Company: company
Investor: investor
Date: date
*
1
*
*
1
1
Business Owner
Company: company
List<Transaction>transactions
Investor
Double: cash
Integer: shares

The database tables will be integrated based on the above specified diagram,

2. Test Strategy

The project will use several unit tests and some integration tests. The tests will focus mainly on getting correct responses out of the api(correct splitting and its effect, invalid login credentials, etc.).

III. Elaboration – Iteration 2

1. Architectural Design Refinement

[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]

2. Design Model Refinement

[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]

IV. Construction and Transition

1. System Testing

[Describe how you applied integration testing and present the associated test case scenarios.]

2. Future improvements

[Present future improvements for the system]