Book Trader

Analysis and Design Document

Student: Paiu Alexandra

**Group: 30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
| 02/may/19 | 0.1 | Iteration 2 | Paiu Alexandra |
| 14/may/19 | 0.2 | Diagram Correction | Paiu Alexandra |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

*[Present the project specification]*

# Elaboration – Iteration 1.1

# Domain Model

# Architectural Design

## Conceptual Architecture

The pattern that I used to implement the application is Layer Pattern Architecture, it is used to separate the classes into 3 main categories, presentation, business and database access. In each of those categories will be stored classes related to it’s main purpose, for example, I will not store the updates and deletes for users profile in presentation layer, where will be stored the main implementation of how the application will look like. In the second layer, the service one I will try and put the users and their trading, follow by working with the data extracted from database access layer, lastly I might try to store the messages in a database but in the first steps it will not be a main purpose and with this being said, I will not have a repository for message in data layer. The reason for choosing this pattern is the way you view classes and the flow of the project, being more simple and easy to identify whatever I would need in the future for an upgrade or a modification for the project.

Another pattern which will be implemented in the project will be a Client Server Architecture, it will be useful when a client send a request about trading or messaging another user, even when he wants to browse books in this case, the server will receive the request, process it and send it back to the client. In this way we can use more clients connected to the same network.

The reason for using those two pattern in my project is the way I can integrate them in it, meaning that the Layer Pattern is used to manage easily the data contained and viewing it, in this category it is included the way I, as a developer, can update or modify the data. As for the second pattern used, I think it work hand in hand with the Layer style, because it is focused on the client – system part, which it is not included in the other one, in this way I can assure that my clients will not have a hard time using the application and they can enjoy it’s features.

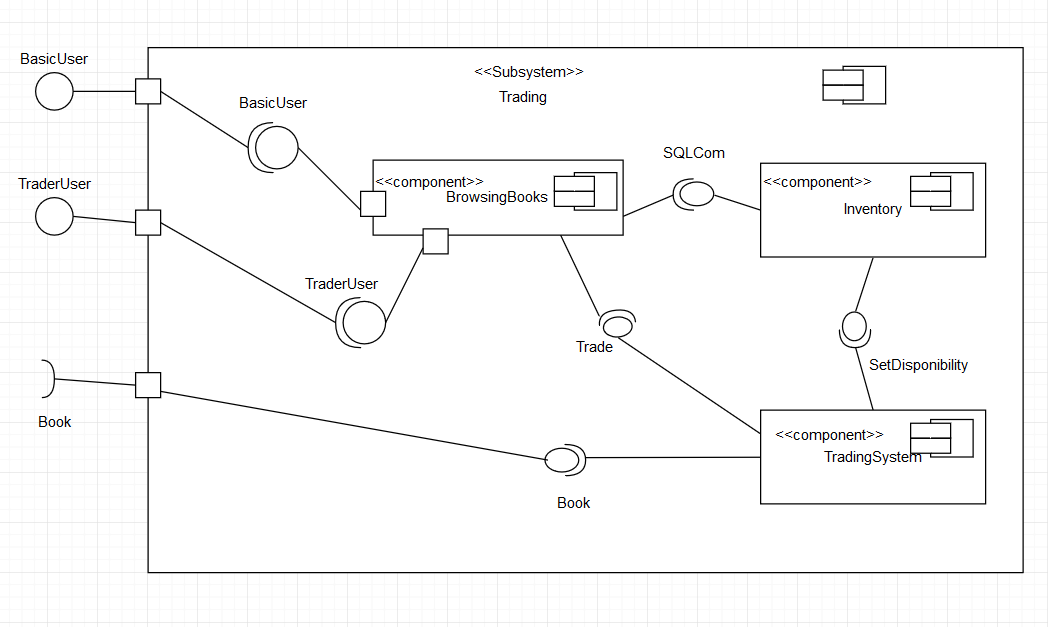
## Package Design

## 

## 

## Component and Deployment Diagrams

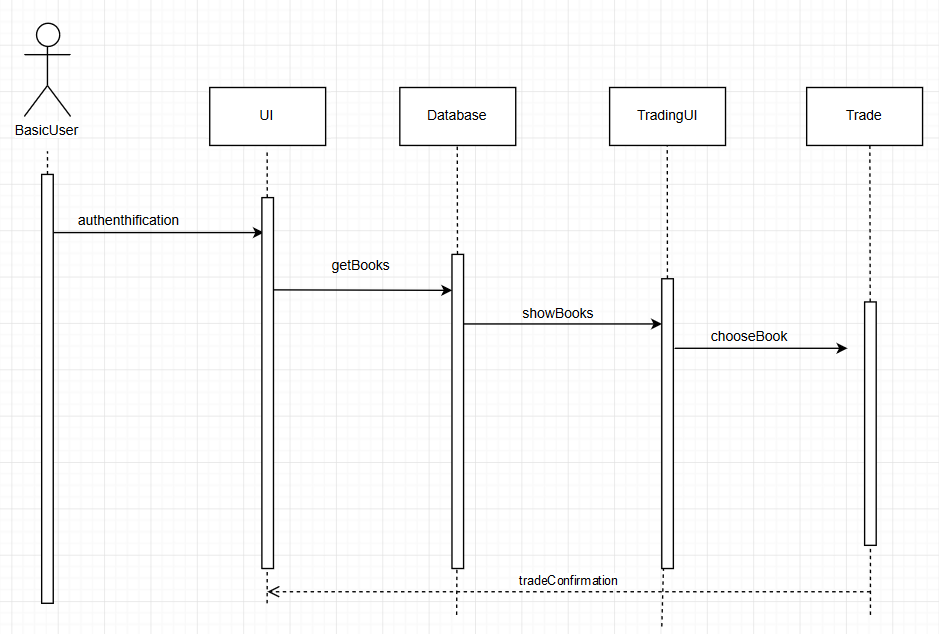
# 



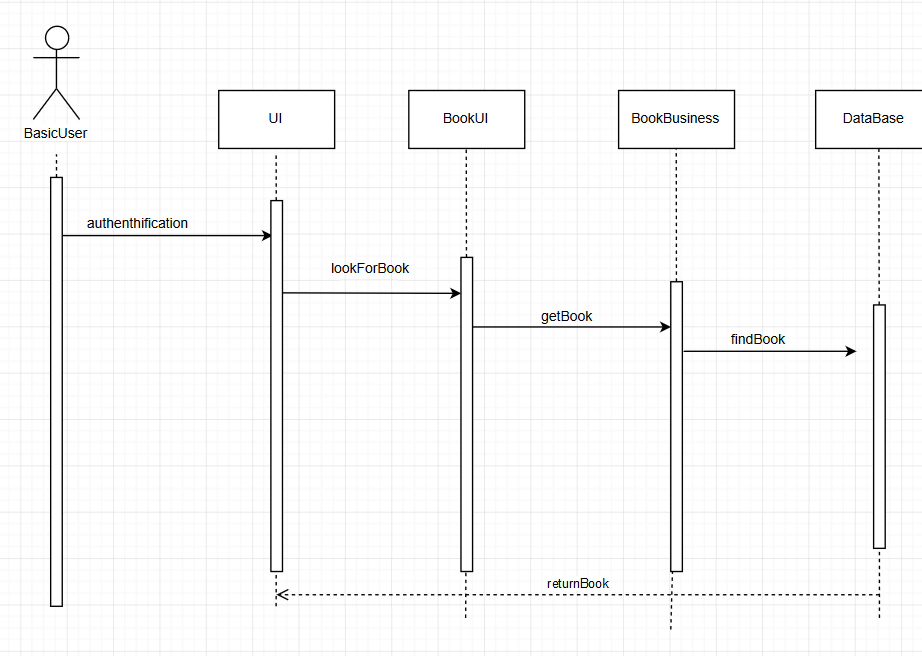
# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

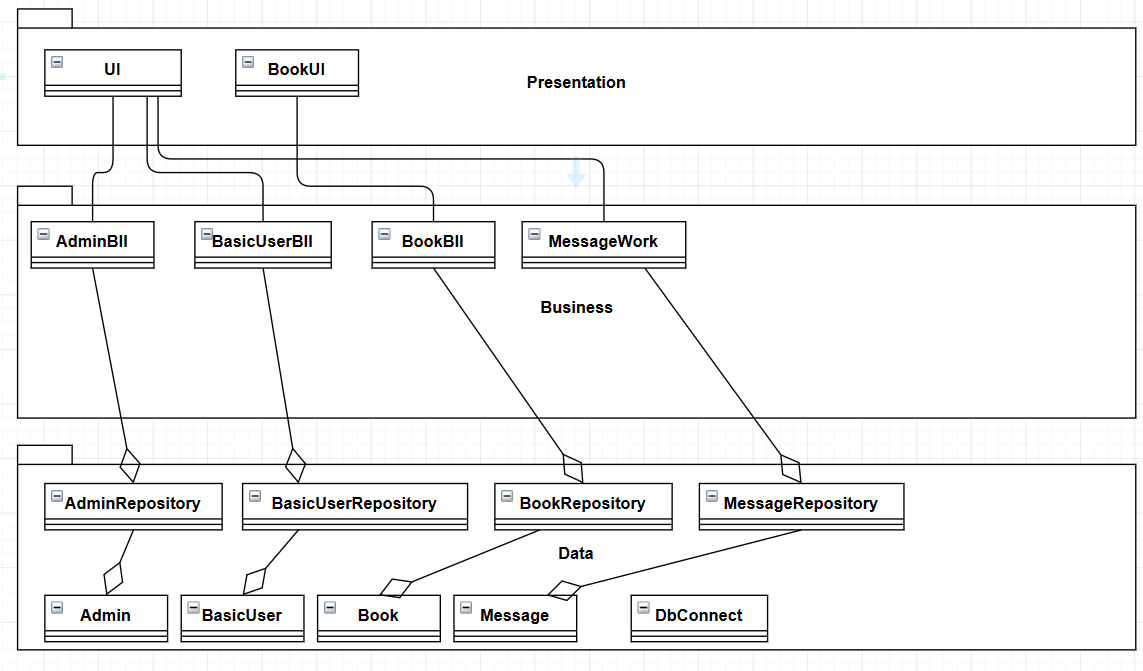


## 



## 

## Class Design



# Data Model

# 

# Unit Testing

The project will be tested using Junit and it will lean the login, add book, trade book and most likely the chatting. The Use Case Scenario for those can be found in the document Project\_UseCaseModel.

# Elaboration – Iteration 2

# Architectural Design Refinement

I have added the many-to-many resolved classes into the application for an easy way to watch the flow of the application and some repositories for them.

# Design Model Refinement

# Construction and Transition

# System Testing

Future improvements

For a future improvement, I might add some strong security protocols and a auto-generate algorithm for suggesting books and even suggesting people to trade with, even a picture preview of books.

# Bibliography