<Numismatic Application>

Analysis and Design Document

Student:Urda Sebastian George

**Group:30433**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <1/Apr/18> | <1.0> | <Update Iteration 1.1> | <Urda Sebastian George> |
| <28/May/18> | <2.0> | <Update Iteration 2> | <Urda Sebastian George> |
|  |  |  |  |
|  |  |  |  |

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

The purpose of this application will be to serve as my final project at Software Design. The main goal of the Numismatic Application App will be to create a common platform for the collectors who want to share the information they know about the coins they own. The application will give the coin collectors the opportunity to keep track of their coins and communicate with others who share the same passion. A user could create an account, log in, add coins, se the coins that another user shared.

# Elaboration – Iteration 1.1

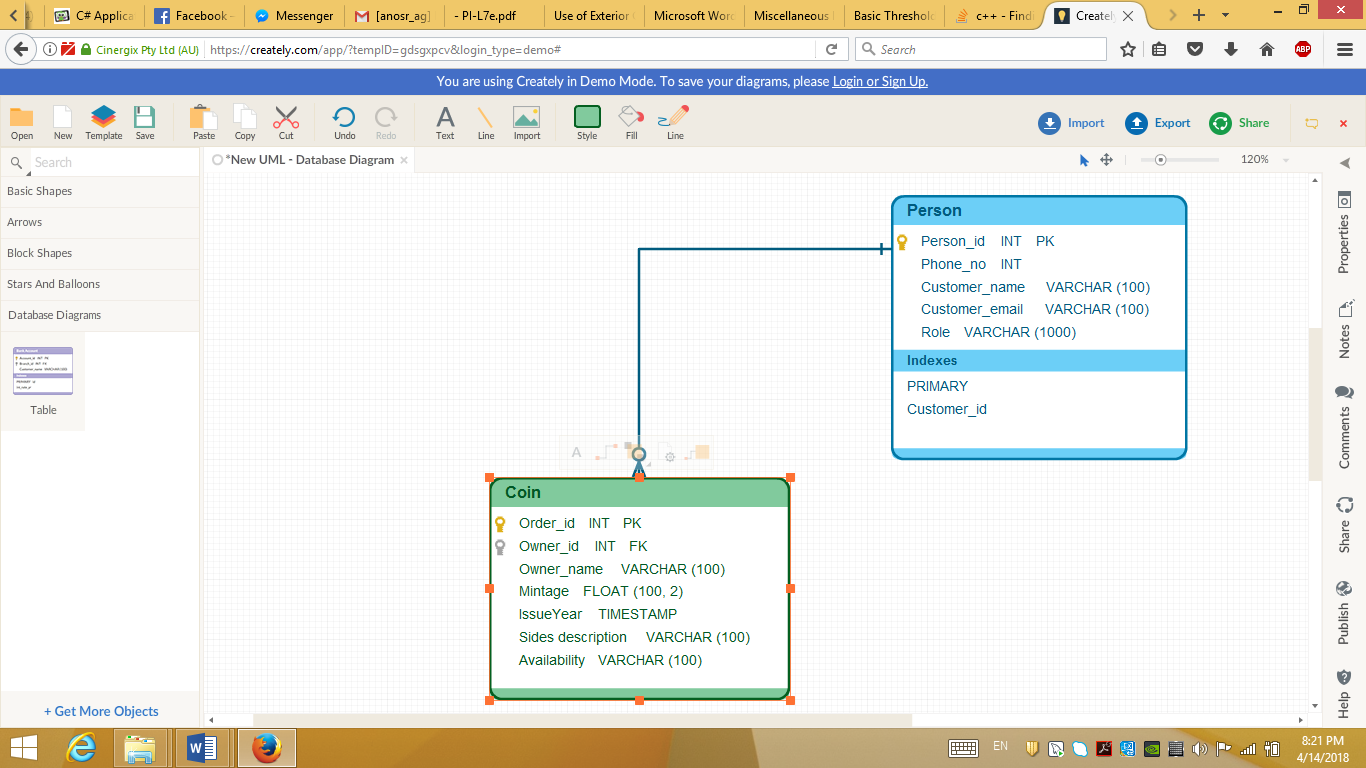
# Domain Model

The database will consist of at least 2 tables: Person and Coin.

The Person table will also contain the role of the person: collector, historian, seller .

The historian will also act like an admin user.

The Coin table will have the specifications of the Coin: the year it was issued, mintage, alloy and 2 pictures one for each side, the number of coins the collector owns and weather is available for sale or not.



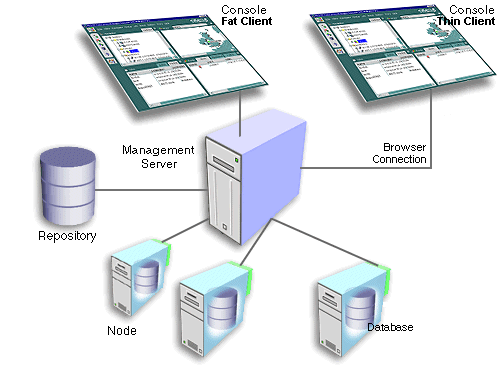
# Architectural Design

## Conceptual Architecture

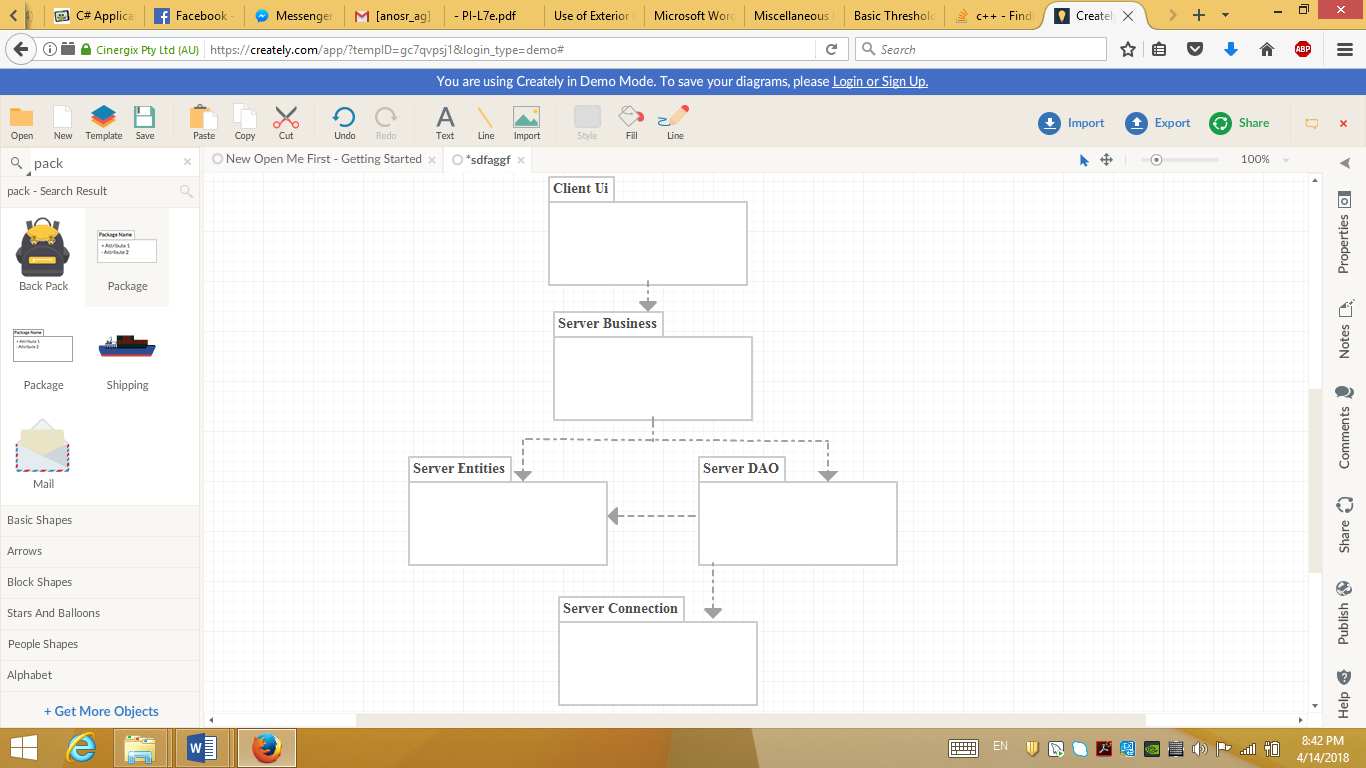
The system uses the client-server architecture. It is an obvious choice because of the need of many users and one processing unit which needs to provide services for each of these users.

* The server will take care of accessing the database, as well as perform the business logic for the user commands. Therefore, it will be a Fat Server.
* The client will display the user interface and send commands to the server via the Internet connection. Therefore, it will be a Thin Client.

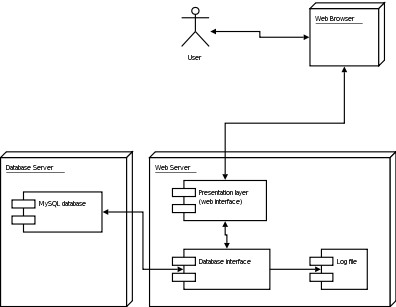
This type of approach is called **Sever-based processing**, since the server does the hard part while the client only displays data to the user and calls the server for any type of transaction that needs to be performed.



## Package Design



## Component and Deployment Diagrams



# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

## Class Design

# Data Model

# Unit Testing

The following are some relevant testing scenarios that need to be considered when performing the application testing.

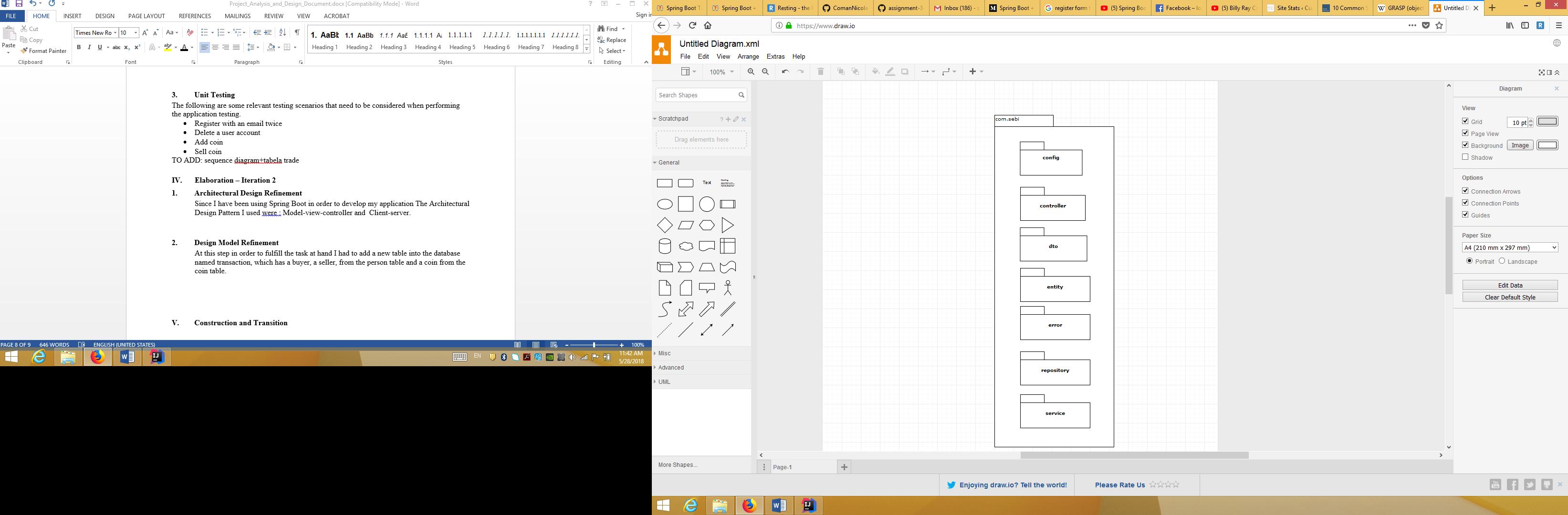
* Register with an email twice
* Delete a user account
* Add coin
* Sell coin

# Elaboration – Iteration 2

# Architectural Design Refinement

Since I have been using Spring Boot in order to develop my application The Architectural Design Pattern I used were : Model-view-controller and Client-server.

# Design Model Refinement

At this step in order to fulfill the task at hand I had to add a new table into the database named transaction, which has a buyer, a seller, from the person table and a coin from the coin table. The Data Model had to be updated and I chose to use only two roles USER and ADMIN

Packages: com.sebi:

Config

Controller

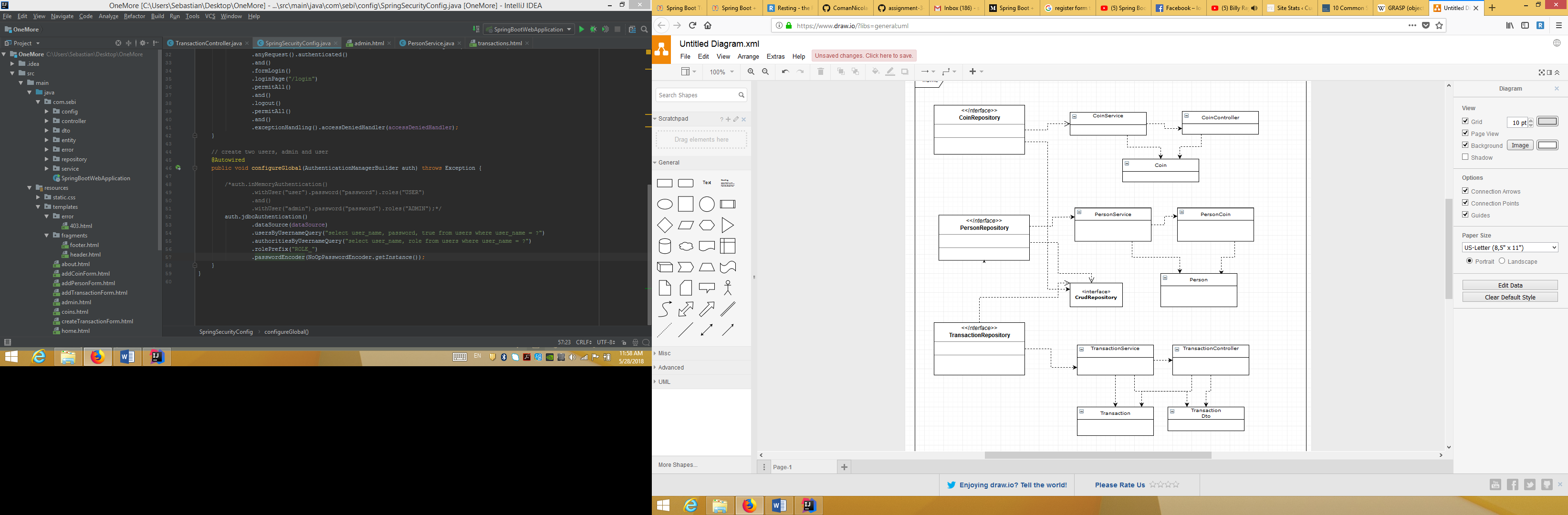
Dto

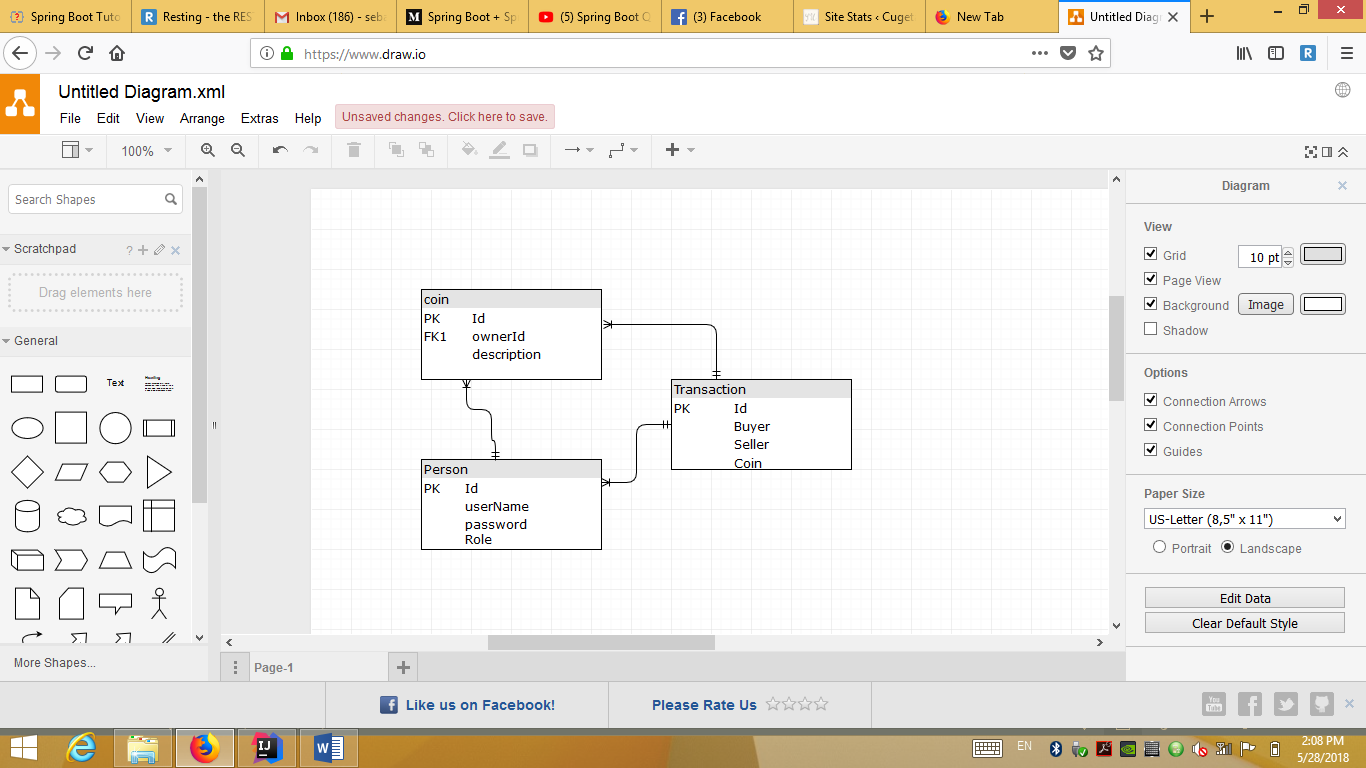
Entity

Error

Repository

Service





Updated DataBase

# Construction and Transition

# System Testing

The system was tested “on the way”, meaning that every new feature added was tested manually.

Teste scenario:

log in with a user name that has a USER role-> view his coins-> make a transaction-> check if the id of the coin has chanced-> check if the user who received the coin has it

log in as an admin-> create a new person with USER role->log in as that user-> add a coin to his collection.

# Future improvements

As a future improvement I would like to add pictures of the coins and maybe new roles for the users.

# Bibliography

<http://www.mkyong.com/spring-boot/spring-boot-spring-security-thymeleaf-example/>

<https://medium.com/@gustavo.ponce.ch/spring-boot-spring-mvc-spring-security-mysql-a5d8545d837d>

<https://www.youtube.com/watch?v=_Jnu_jHfQbM&t=0s&list=PLqq-6Pq4lTTbx8p2oCgcAQGQyqN8XeA1x&index=32>

<https://github.com/buzea/SoftwareDesign2018>