Furniture Deals

Analysis and Design Document

Student:Urcan Miruna - Stefania

**Group:30238**

Table of Contents

[**1. Requirements Analysis**](#_heading=h.aaw7mub2lp4j) **2**

[**Assignment Specification**](#_heading=h.tyjcwt) **3**

[**Functional Requirements**](#_heading=h.3dy6vkm) **3**

[**Non-functional Requirements**](#_heading=h.1t3h5sf) **3**

[**2. Use-Case Model**](#_heading=h.9t1mvtqcf893) **4**

[**3. System Architectural Design**](#_heading=h.1ci93xb) **5**

[**4. UML Sequence Diagrams**](#_heading=h.3whwml4) **7**

[**5. Class Design**](#_heading=h.2bn6wsx) **7**

[5.1 Design Patterns Description](#_heading=h.qsh70q) 7

[5.2 UML Class Diagram](#_heading=h.3as4poj) 8

[**6. Data Model**](#_heading=h.23ckvvd) **12**

[**7. System Testing**](#_heading=h.ihv636) **12**

[**8. Bibliography**](#_heading=h.2grqrue) **12**

# 1. Requirements Analysis

## Assignment Specification

The task is about building a search engine for furniture products. The client ( or user) is able to login or create an account in case he does not have one. He can search for different products added by a staff member. For an easier search, he can filter the products by their type, price or name. In this case, the product list will change, in order to have a better vision for what the user is looking for.

A deal can be proceeded only if available, which means there are enough products to cover the order for user. The payment process is automatically requested when the user validate the order.

The payment is validated by the staff. In this case, the order will remain in the system until a staff member validate it. This process can be seen by the user via Invoice The payment is validated by the staff. In this case, the order will remain in the system until a staff member validate it. This process can be seen by the user via Invoice: when a user member logs in for invoice, a message will be sent to him with the total price of the orders that are not validated yet. THe page with the invoice has two tables: one table for the history orders - orders that have been already validated by a staff member, and a table with Invoice orders - the orders that have to be validated.

## Functional Requirements

There are two kinds of clients for this application: users and staff .

The users operations required are: login, addOrder, filter by name, type ( kitchen, office, livingroom, bedroom) or price, invoice, order history, logout.

The staff operations required are: login, validate order, delete product, add product,, apply discount, logout.

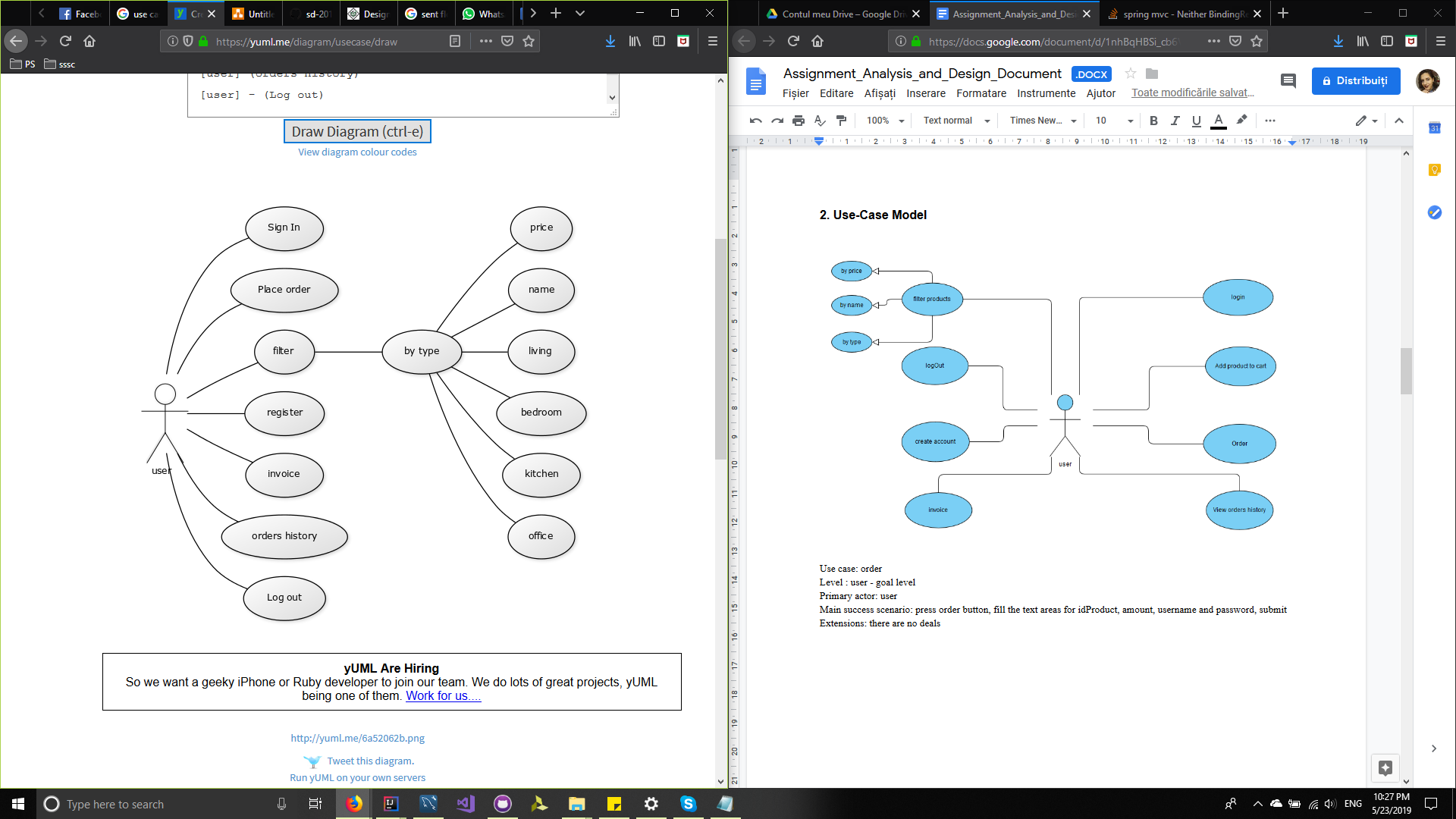
Both categories of clients are able to logIn. The user has to login only for the invoice. Also, his logIn data will be required when placing an order. The staff has to log in when opening the page, due to some operations, like delete product and validate order. The logout is made automatically when the client closes the window.

## Non-functional Requirements

We were asked to use an OOP language (Java), implement and test the application. Another requirement was to use a layered architecture and to use a database for storage. All the inputs of the application has to be validated. The next request was to build a feature for the staff member, in order to apply discounts. This feature can change the overall price.

# 

# 2. Use-Case Model



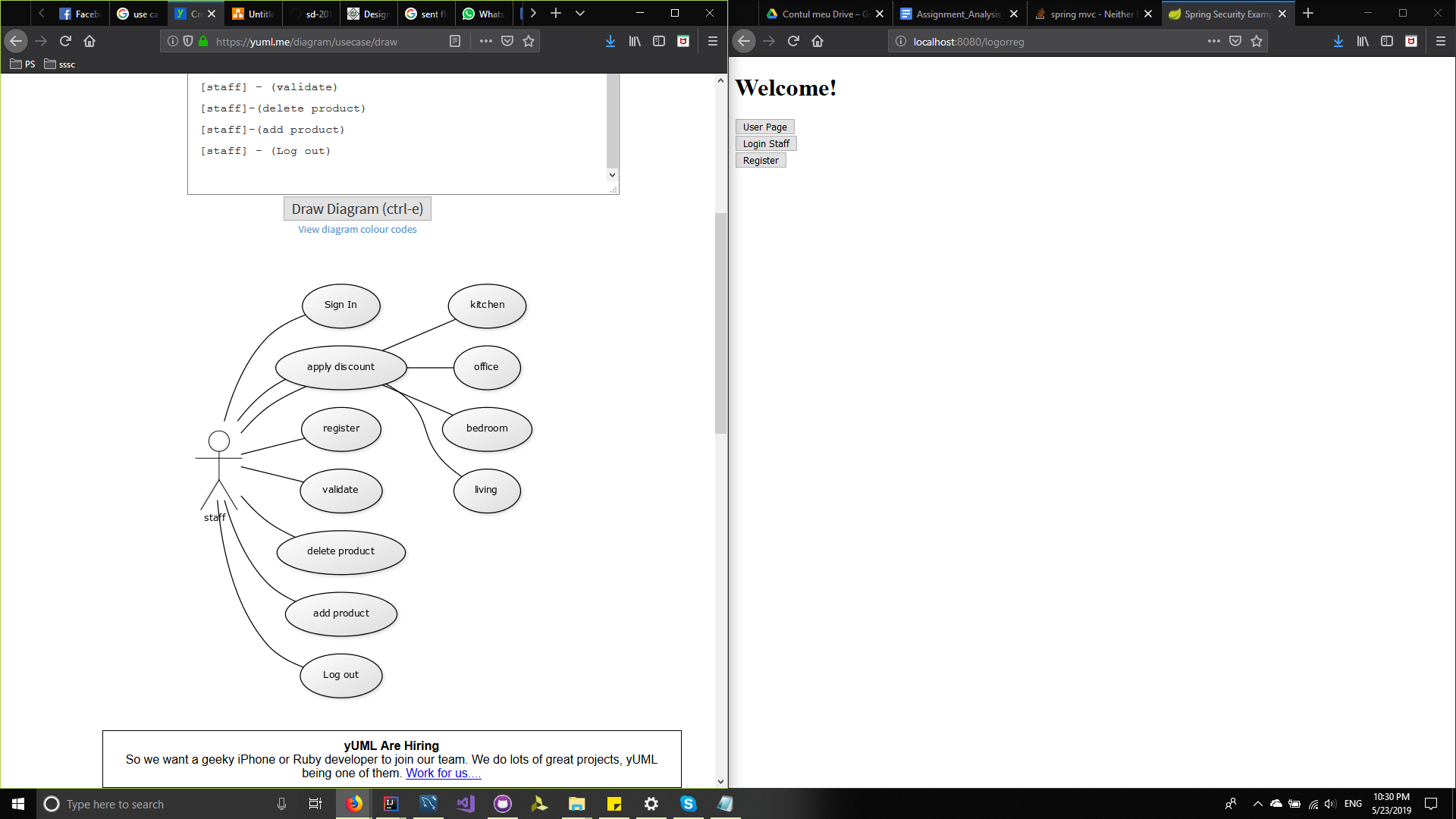
Use case: order

Level : user - goal level

Primary actor: user

Main success scenario: press order button, fill the text areas for idProduct, amount, username and password, submit

Extensions: there are no deals



Use case: validate order

Level : staff-goal level

Primary actor: staff

Main success scenario: login, choose order from user and validate data

Extensions: there are no orders

# 3. System Architectural Design

**3.1 Architectural Pattern Description**

An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture. he most common pattern is the layered pattern. Each layer has a specific responsibility.

I organized the application using client-server architecture. This architecture shares the data processing chores between a server and clients : the server hosts, delivers and manages most of the resources and services that are consumed by the client.

Spring Data’s mission is to provide a familiar and consistent, Spring-based programming model for data access while still retaining the special traits of the underlying data store.

For this assignment, I used a CQRS architecture. CQRS stands for **Command Query Responsibility Segregation**. This architecture segregate operations that read data from operations that update data by using separate interfaces. This can maximize performance, scalability, and security.

For this application I implemented an Observer Pattern. The observer pattern is a [software design pattern](https://en.wikipedia.org/wiki/Design_pattern_(computer_science)) in which an [object](https://en.wikipedia.org/wiki/Object_(computer_science)#Objects_in_object-oriented_programming), called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their [methods](https://en.wikipedia.org/wiki/Method_(computer_science)). This pattern adressess to the following problems: A one to many dependency between objects should be defined without making the objects tightly coupled, it ensured that when an object changes state, a various number of dependent objects are updated.

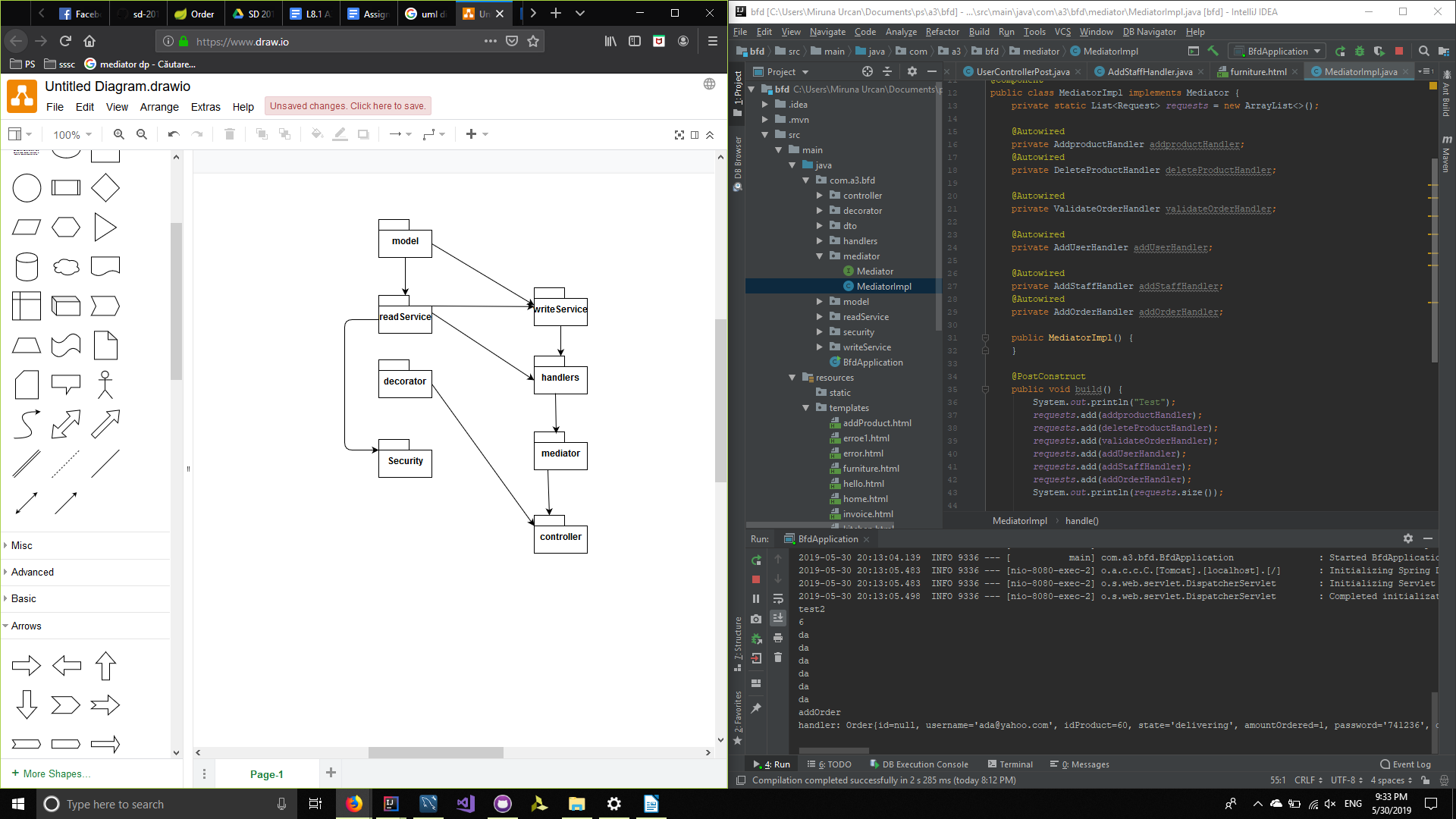
Mediator pattern is used to reduce communication complexity between multiple objects or classes. This pattern provides a mediator class which normally handles all the communications between different classes and supports easy maintenance of the code by loose coupling. Mediator pattern falls under behavioral pattern category. We used mediator pattern to handle the communication between events and the interface.

Decorator pattern allows a user to add new functionality to an existing object without altering its structure. This type of design pattern comes under structural pattern as this pattern acts as a wrapper to existing class. I used the decorator pattern to apply discounts.

**3.2 Diagrams**

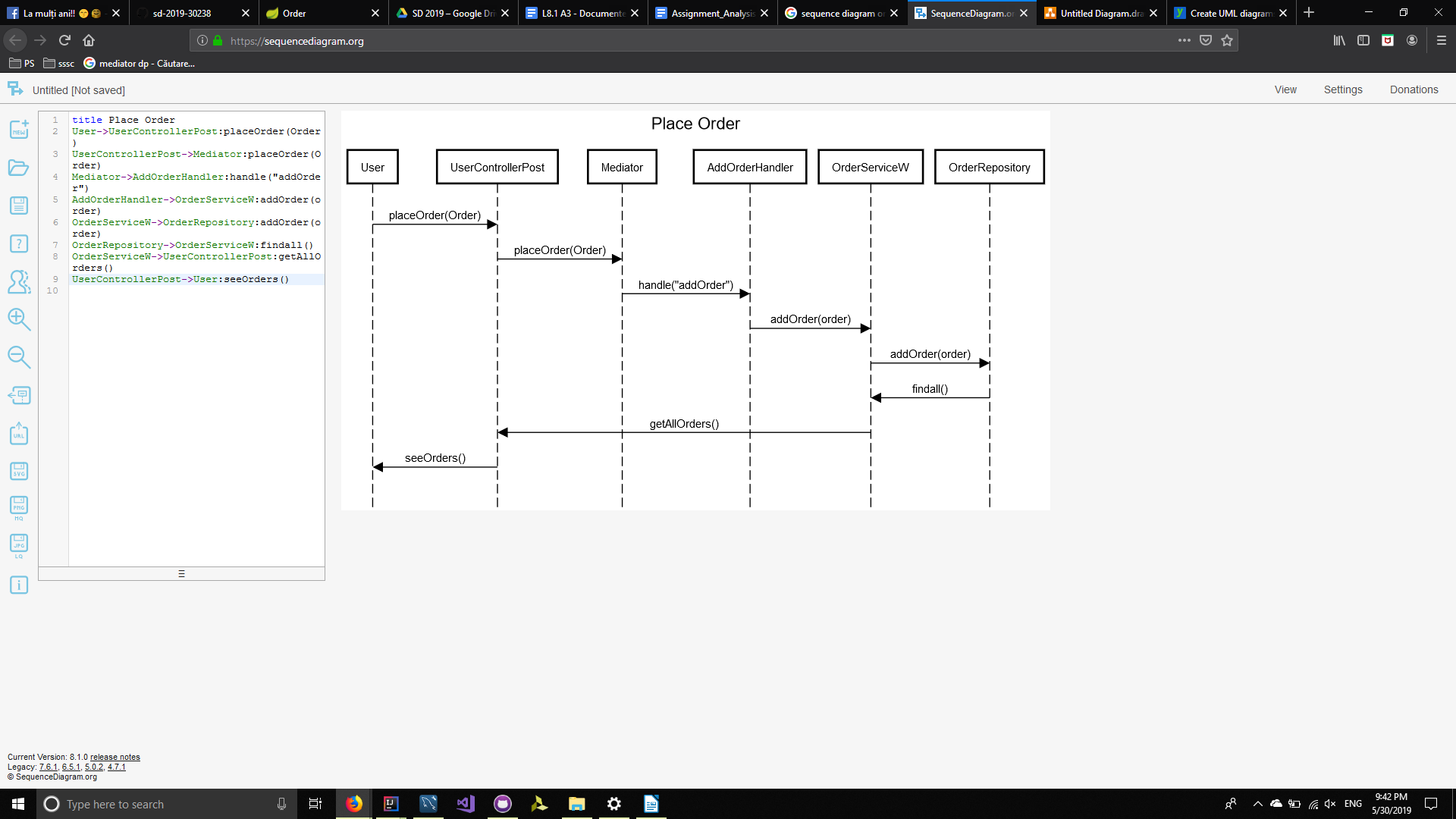
*[Create the system’s conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]*

Package diagram:



# 4. UML Sequence Diagrams

*[Create a sequence diagram for a relevant scenario.]*

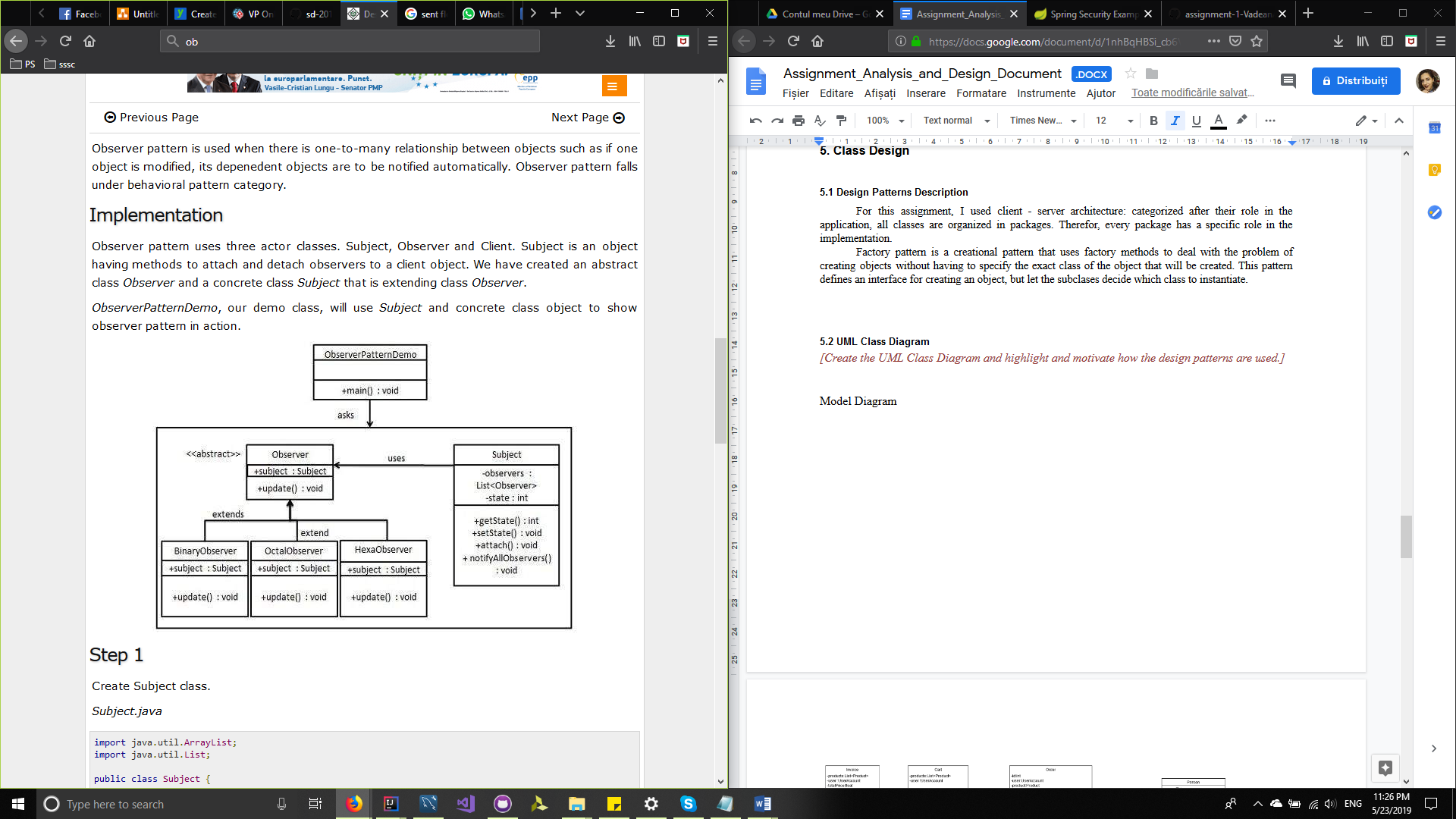


# 5. Class Design

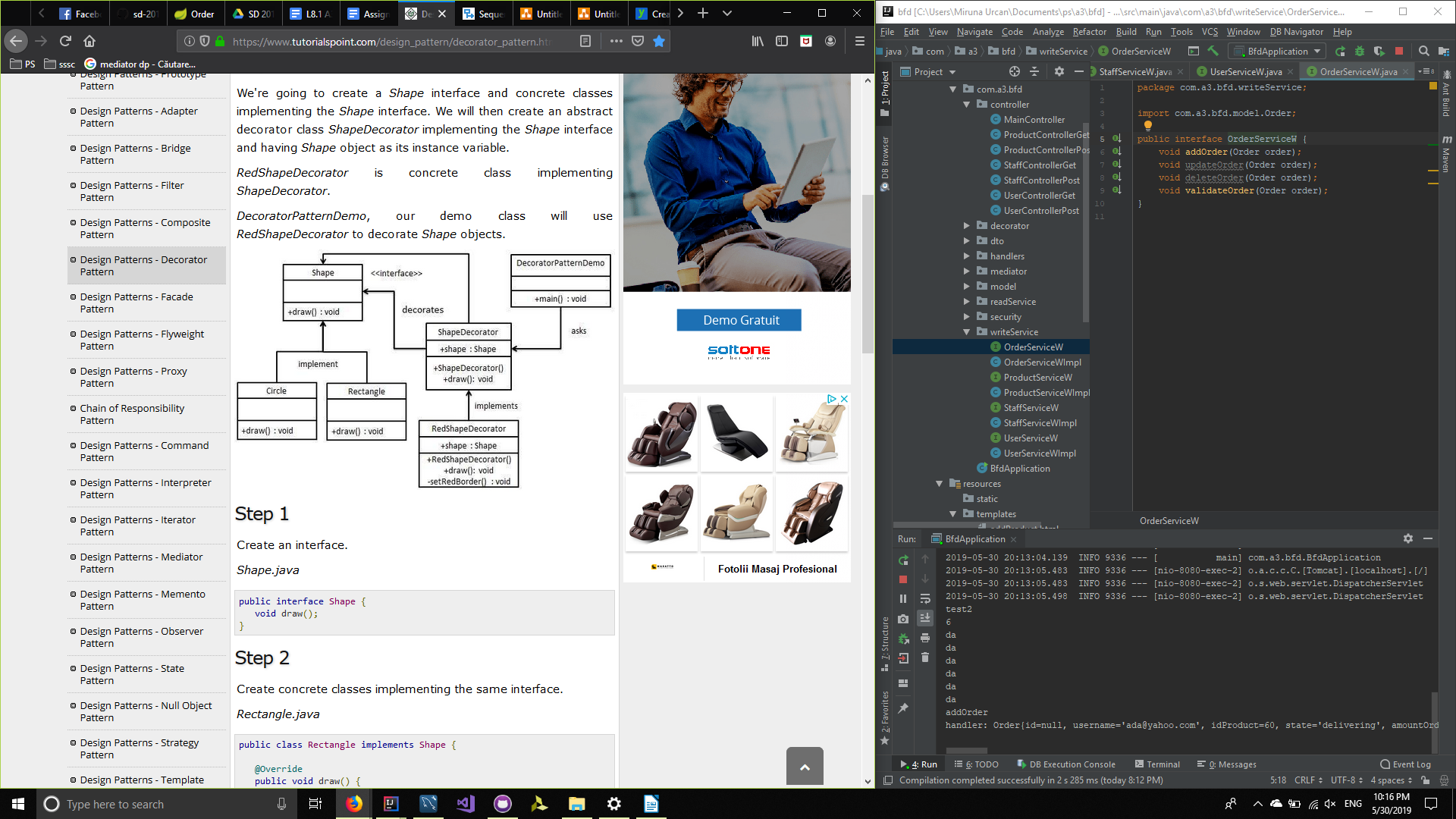
## 5.1 Design Patterns Description

For this assignment, I used client - server architecture: categorized after their role in the application, all classes are organized in packages. Therefor, every package has a specific role in the implementation.

Observer pattern is used when there is one-to-many relationship between objects: if one objects id modified, all its dependent objects are notified.

**

The Decorator pattern is used to apply discount for different types of products. For our example, the interface is defined by the Discount, The subjects are the types of products.

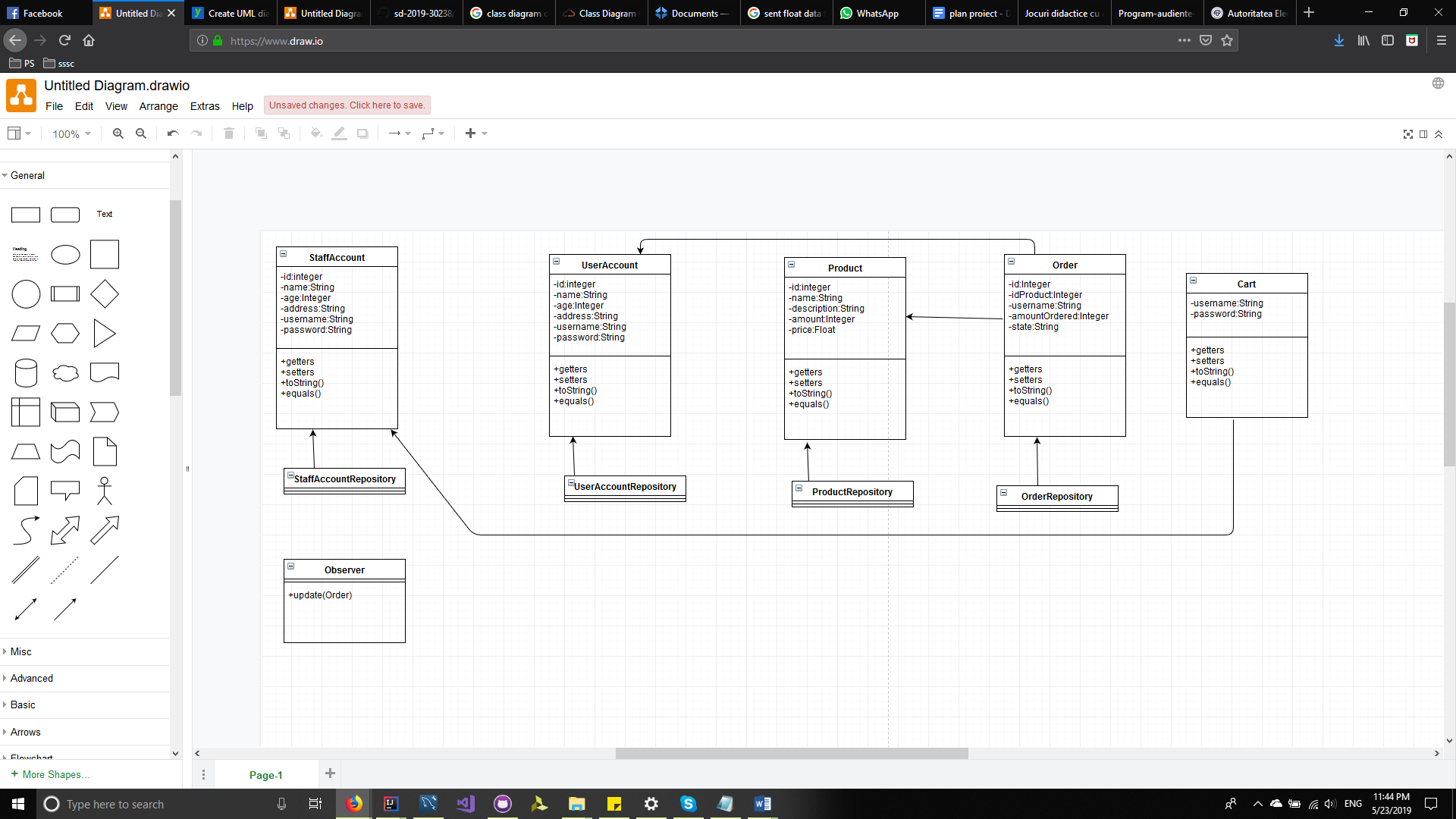


The Mediator pattern is used to handle all the events that writes in the database. In the mediator package is defined the interface and implementation. This implementation contains the handlers for the events that are needed.

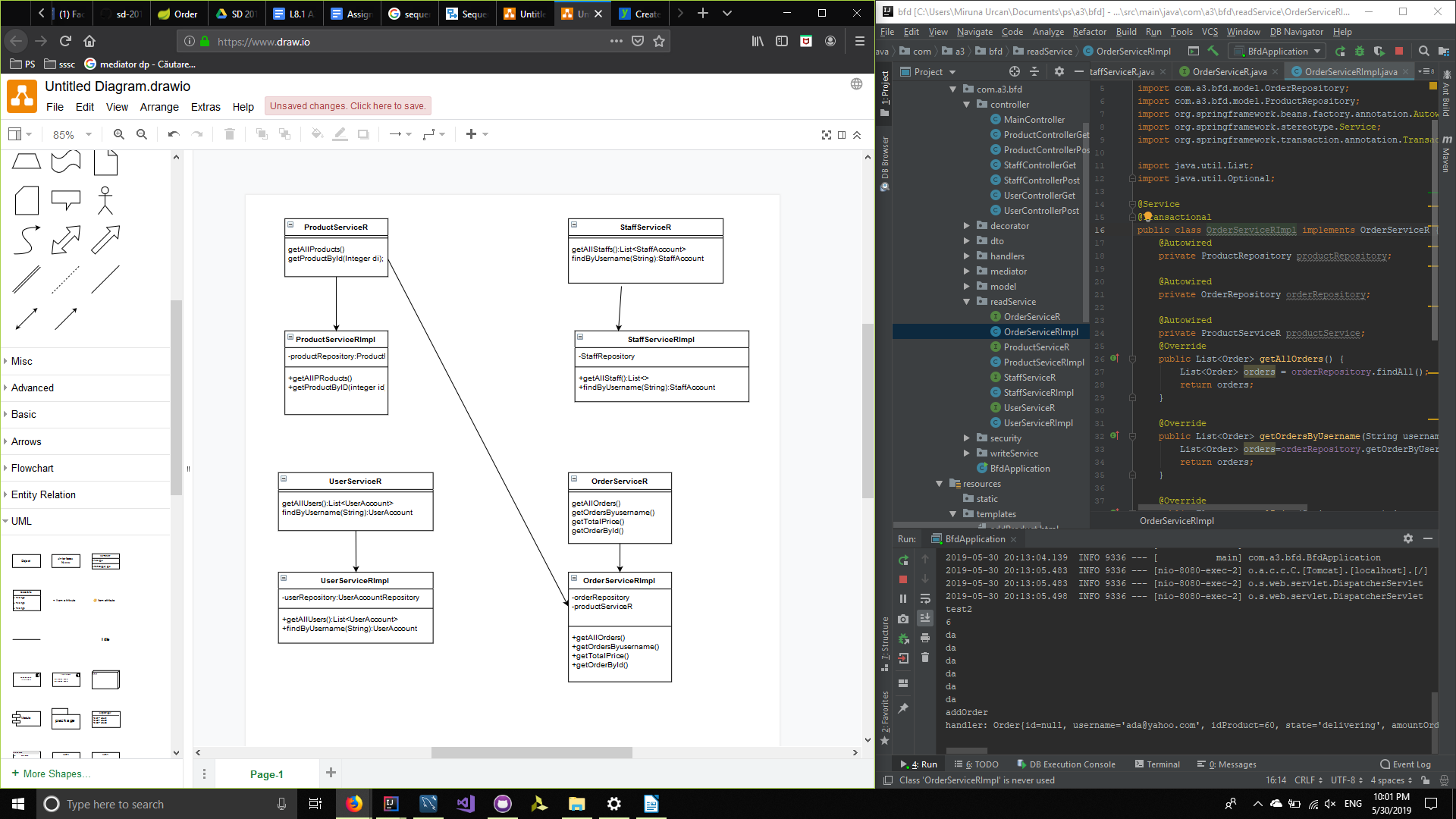
## 5.2 UML Class Diagram

*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*

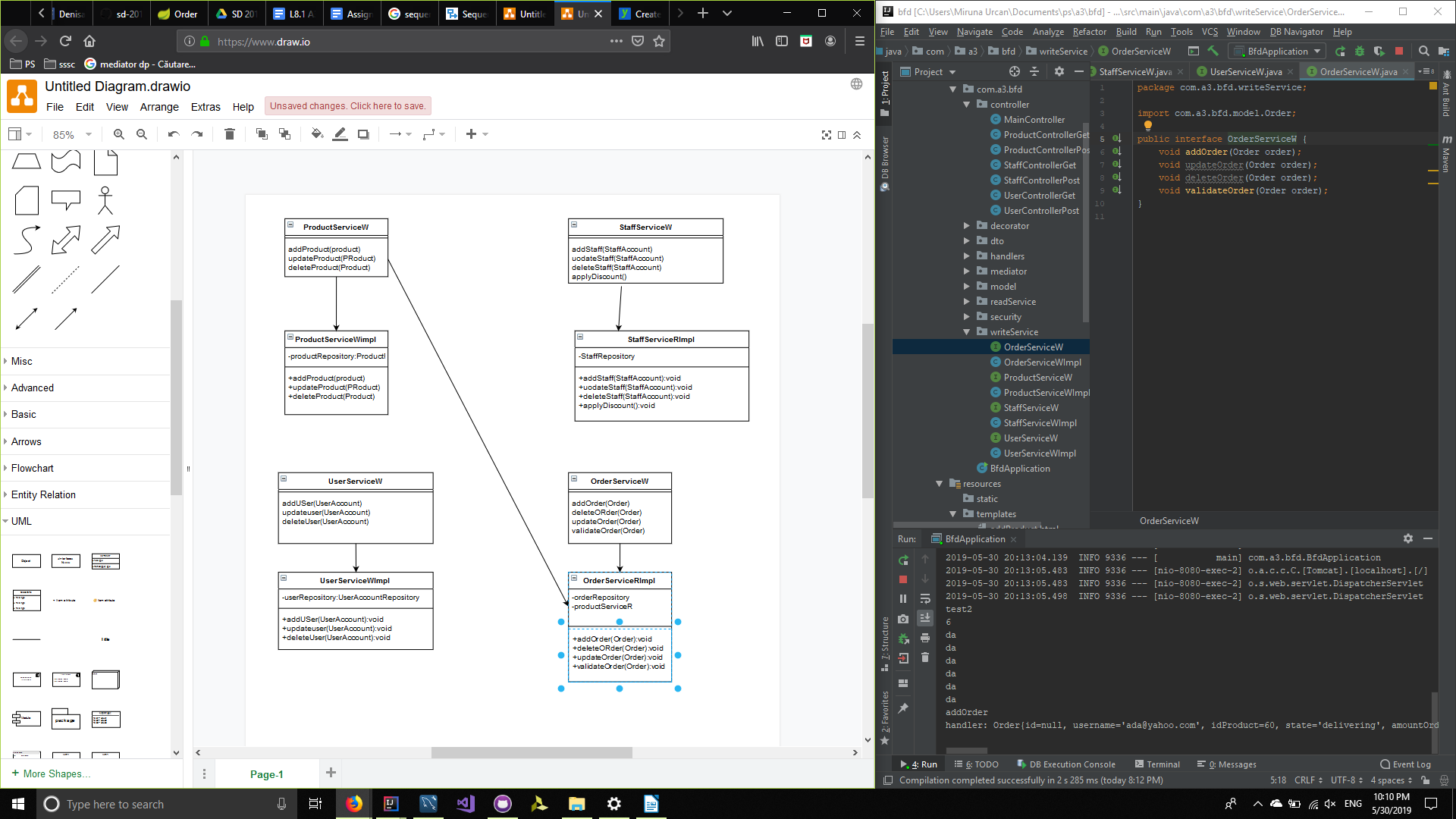
Model Diagram



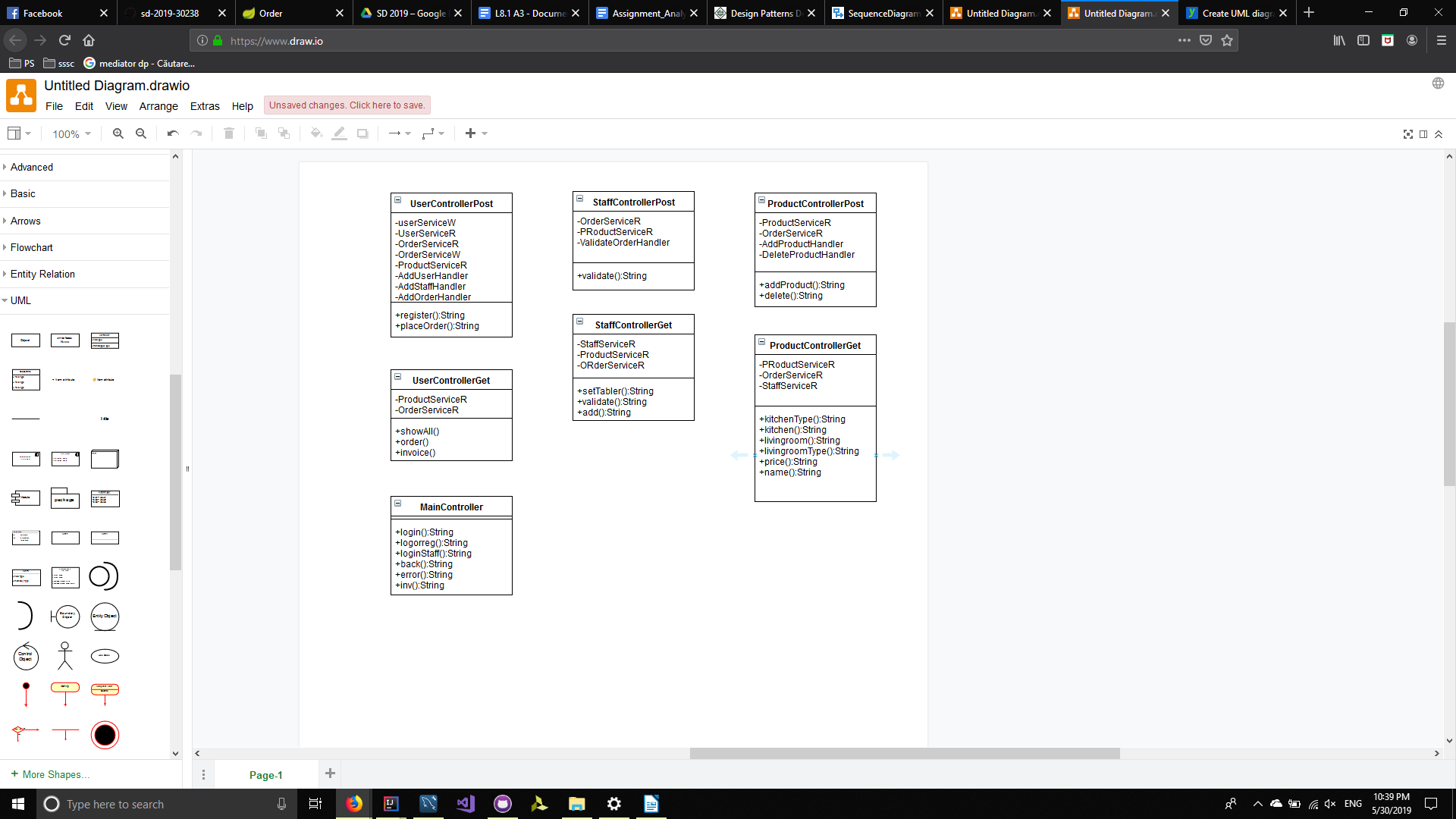
Read Service Diagram



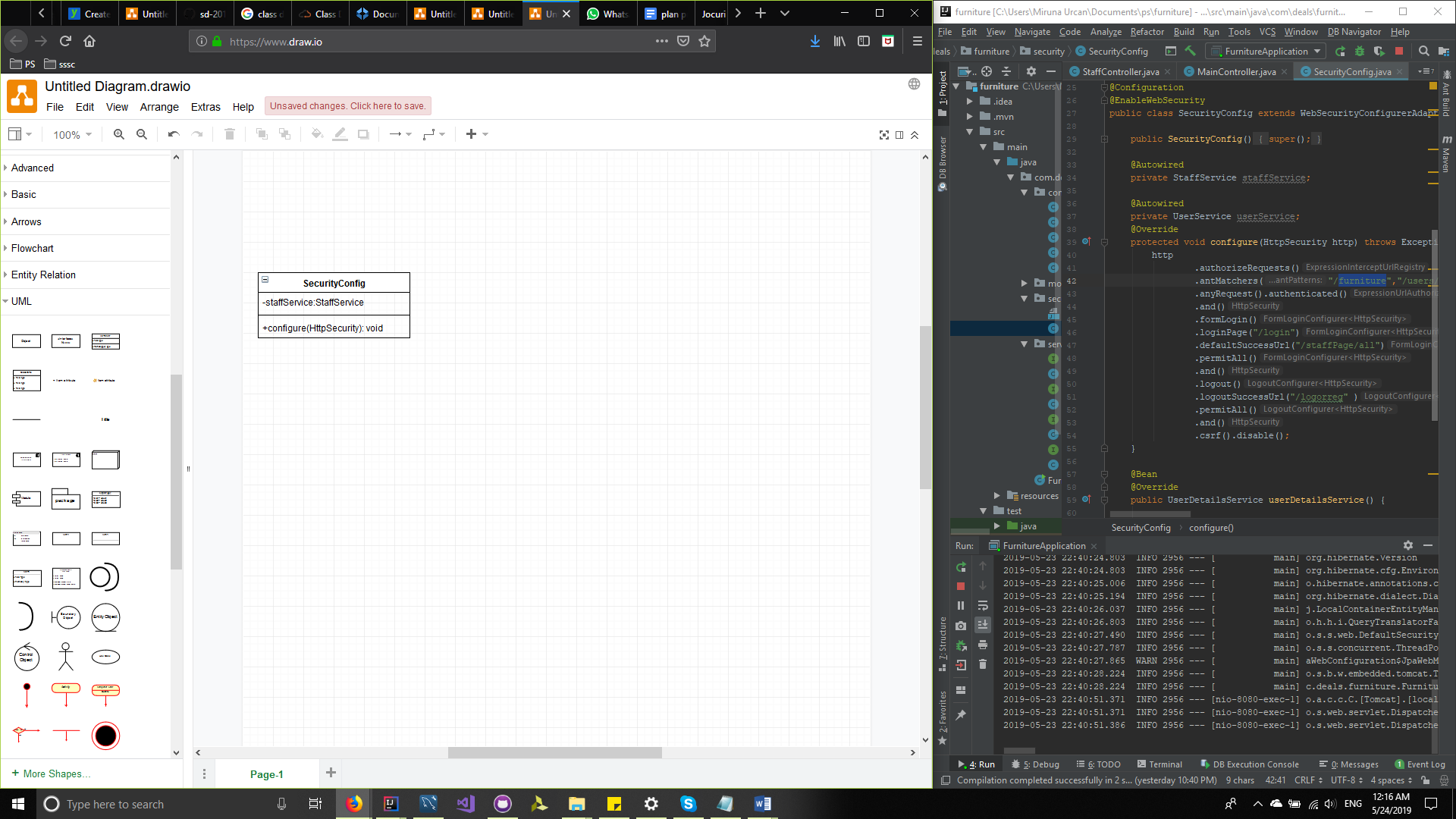
Write Service Diagram



Controller Diagram



Security



# 6. Data Model

Data Model is presented in model package. Each class(UserAccount, StaffAccount, Product, Invoice, Cart, Person and Order) are the objects used for the implementation of the application. All classes from this package are directly communicating with classes from the dao package, which realize the connection with the database. Each class has more simple attributes needed for representing the data.

In order to create this application, I used a database with five tables, describing the needed attributes to describe data .

# 7. System Testing

*[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]*

# 

# 8. Bibliography

Architectural pattern <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html>

Design Pattern

<https://en.wikipedia.org/wiki/Software_design_pattern>

Observer Pattern

<https://www.tutorialspoint.com/design_pattern/observer_pattern.htm>

Spring data

<https://spring.io/projects/spring-data>

Decorator Pattern

<https://www.tutorialspoint.com/design_pattern/decorator_pattern.htm>

Mediator pattern

<https://www.tutorialspoint.com/design_pattern/mediator_pattern.htm>

CQRS architecture

<https://martinfowler.com/bliki/CQRS.html>