**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ ВАРНА**



**Курсов проект**

**по Обектно-ориентирано програмиране- 2 част**

**Тема №23: Програма за обработка на складове**

Катедра: ФИТА

Специалност: СИТ

Курс III, Група 3а

Реализиран от: Белин Герджиков(Фак. № 19621672) и Румен Макриев (Фак. № 19621669)

Сорс код достъп на: <https://github.com/BelinGerdjikov/TuVarna_Project_War_Warehouses.git>

**ЗАДАНИЕ НА ПРОЕКТА**

Да се разработи информационна система, предоставяща услуга склад. Програмата съхранява и обработва данни за складови помещения. Системата позволява множествен достъп.

Системата поддържа два вуда потребители администратор и клиенти (складов агент, собственик) с различни роли за достъп до функционалностите в системата.

Операции за работа с потребители

* Създаване на собственици от администратор.
* Създаване на складов агент от администратор.
* Създаване, редактиране на склад(ове) с избор на един или списък складови агента от собственик за отдаване на складовете под наем.
* Поддържане на профили с характеристика на собственици и складови агенти(хонорар и др...).
* Рейтинговане на складови агенти.

Системата поддържа операции за работа със складови помещения

* Добавяне на ново помещение от собственик (размери, климатични условия вид на съграняваната стока, категория склад и др. ...).
* Отдаване на складово помещение от складов агент, създаване на формуляр за наем (информация за наемателя, период и цена).

Системата поддържа справки по произволен период за:

* Складов агент(сключени договори за наем, данните на уредник, рейтинг).
* Налични складовер за отдаване под наем (дата, статус, адрес, вид на склада).
* Складове (на даден собственик).

Собствениците на складове достъпват справки само за складове, на които е собственик. Собственика достъпва справки за всички притежаващи складове.

Складовите агенти има право на справки за налични складове и история на наемателите.

Системата поддържа Известия за събития :

* Новопостъпила заявка за отдаване под наем(в профила на складовия агент).
* Отдаден(и) склад(ове) (в профила на собственика).
* Изтичащ договор (собственик, складов агент).

**ФУНКЦИОНАЛНИ ИЗИСКВАНИЯ КЪМ ПРОЕКТА**

|  |  |
| --- | --- |
| № | Описание на функционалните изисквания. |
| 1 | Създаване на собственици от администратор (създава потребилско име и парола). |
| 2 | Създаване на складов агент от администратор (създава потребителско име и парола). |
| 3 | Администратор поддържа профилите с характеристики за сообсtвениците и складовите агенти. |
| 4 | Администратор може да създава склад за потребител. |
| 5 | Собственик създава склад на негово име и може да променя информацията за склада |
| 6 | Администаротор прави справка с най-добре справящите се агенти посредством брой сделки(рейтинговане). |
| 7 | Складов агент създава формуляр за отдаване на склад под наем(информация за наемателя, период, цена). |
| 8 | Функционалност за складов агент(сключени договори за наем и данните на удредник). |
| 9 | Функционалност за складов агент за налични складове за отдаване под наем(дата, адрес, вид на склада и цена). |
| 10 | Валидация при създаване на склад |
| 11 | Валидация при създаване на догово от агент |
| 12 | Агентът може да контролира своята комисионна |
| 13 | Валидация при логин: непразни полета, неправилен вход. |
| 14 | Логин функционалност: пренасочване на потребителя към съответното ниво на достъп. |
| 15 | Функционалност за регистрация: (само от админ). |
| 16 | Валидация при регистрация на потребител. |

**СТРУКТУРА НА ПРОЕКТ**

Проектът е реализиран на **Maven**. Използвани са следните dependency-та:

* **MySQL** – връзка с базата данни
* **JDBC** – слой за работа с бази данни
* **JavaFX** – реализация на графичен интерфейс и бизнес логика чрез съответните контролери
* **Log4J** – логове за всички контролери за проследяване на логиката на приложението. Допълнително използване, ако се налага да се дебъгва
* **JUnit** – Unit тестове

**ДЕФИНИЦИЯ НА МОДУЛИТЕ НА СИСТЕМАТА**

аpplication- стартиране на прозорците в програмата.

* HelloApplication – стартиране на проекта
* CreatingNewWindows- отваря нов прозорец при дадени UML path и заглавие на самия прозорец

common

* Constants – константи на всички UML path-ове.

Data – SQL communication

* DataBaseConnectioн – Създава JDBC връзка с MySQL сървар.
* Repositories – Всеки един обект в repositories комуникира със съответната за него База данни.

Presentation

* Controllers - контролерите извършват цялата бизнес логика в приложението. Абсолютно всеки изглед от JavaFX е обвързан със съответен контролер. Тук има валидации, тук се изобразява графично информацията на потребителя и всички заявки тръгват и се връщат тук.
* Modules – Модули с които контролерите комуникират.

Бизнес логика

Service layer

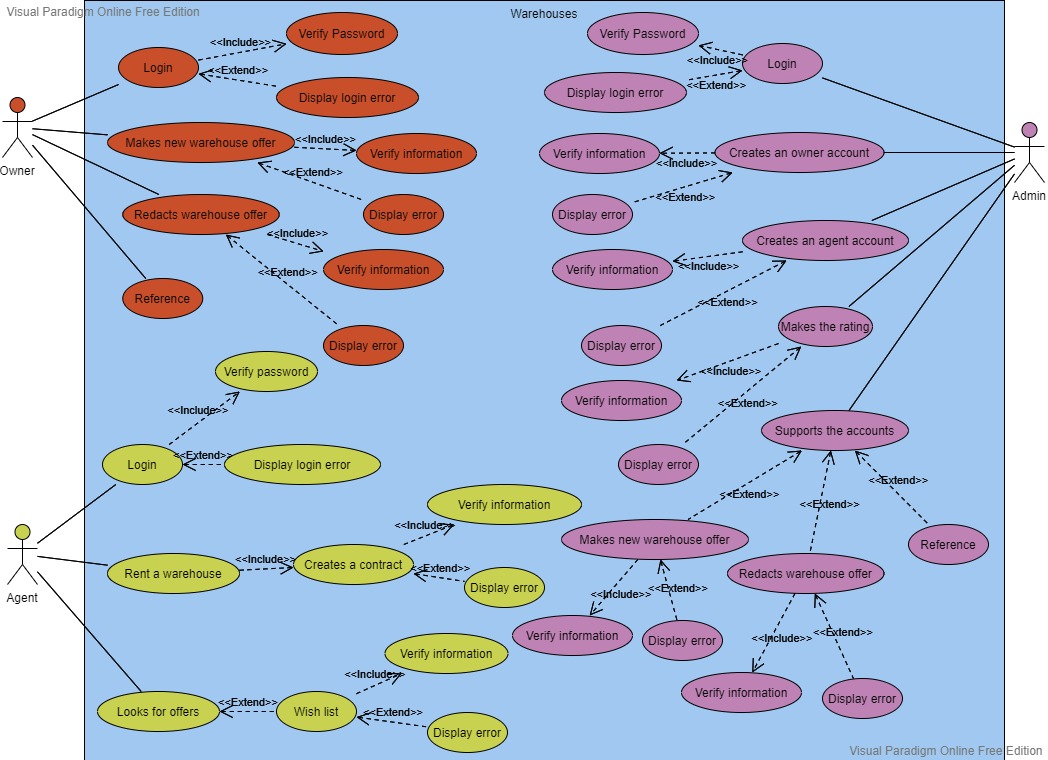
JavaFX Controllers

Repositories

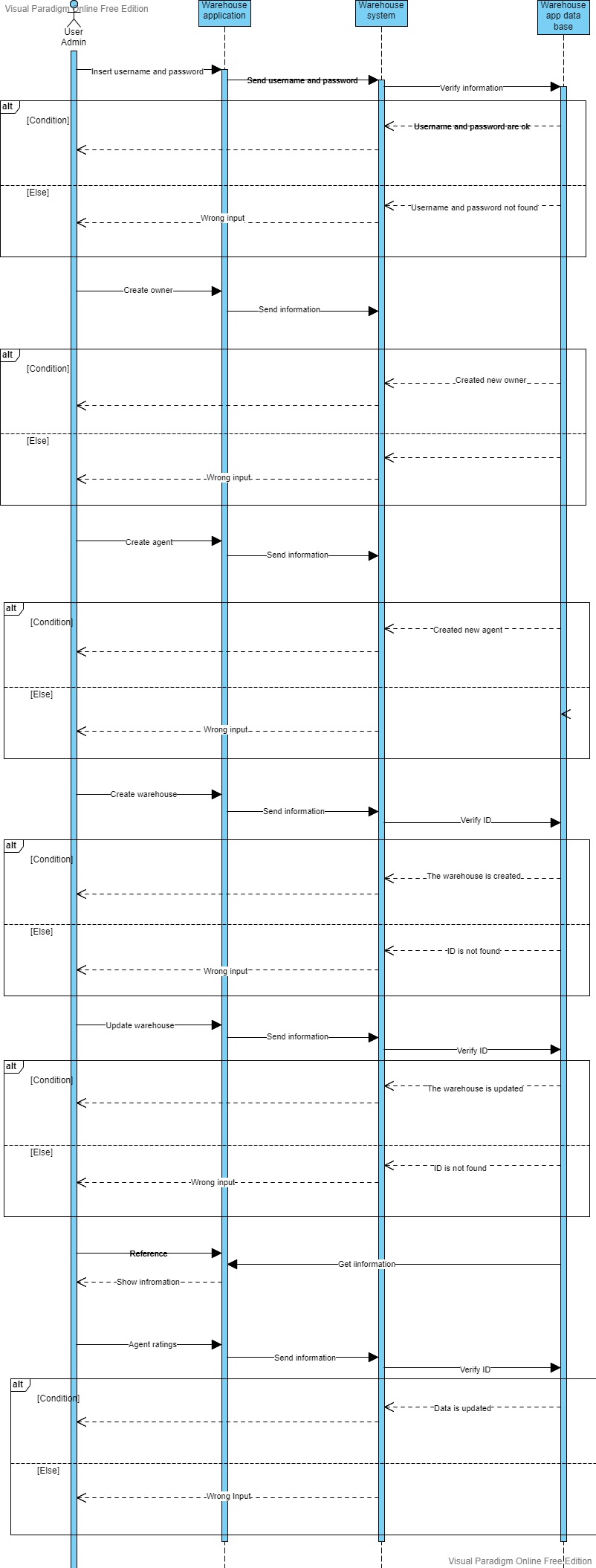
GUI

База данни

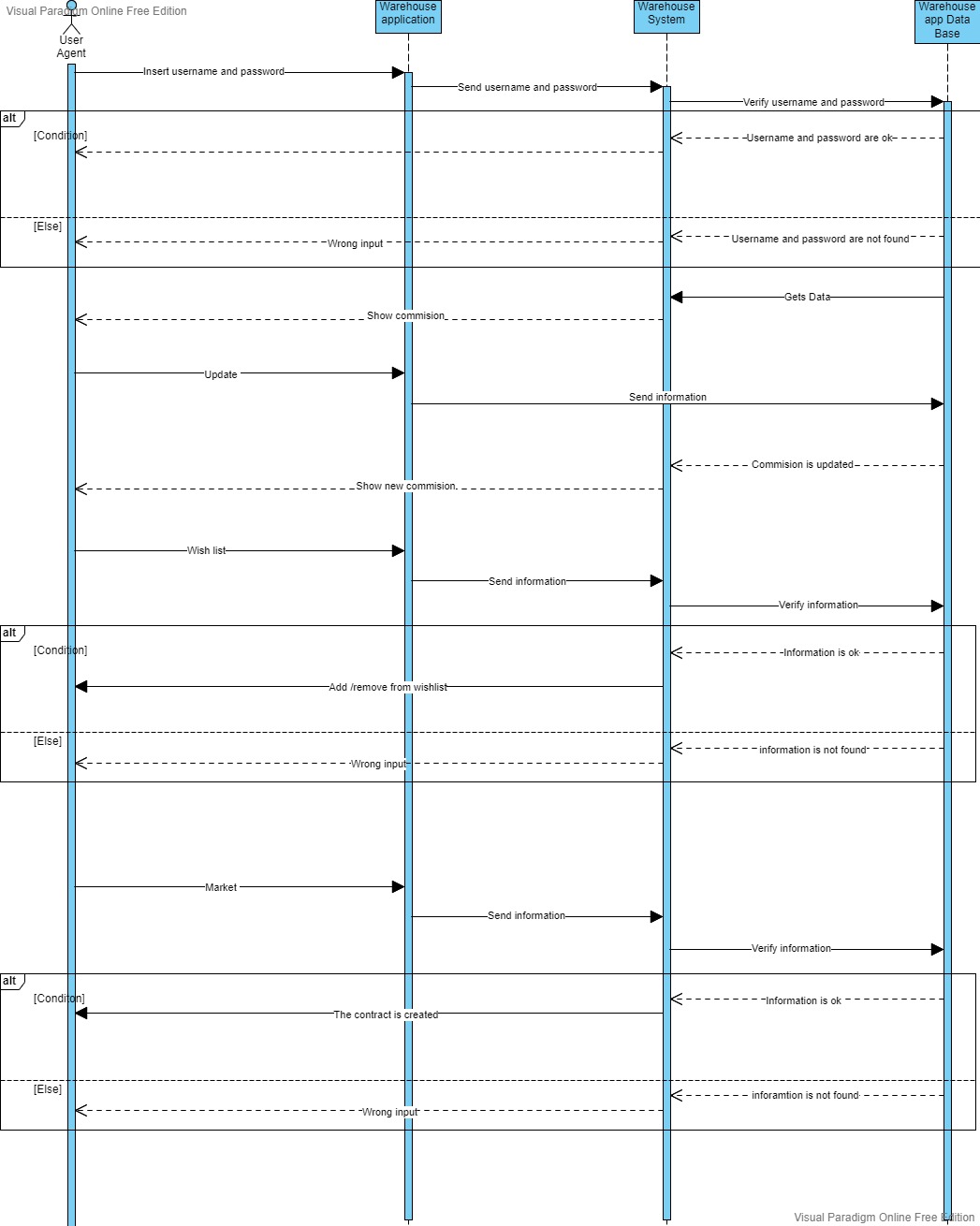
**Диаграми**

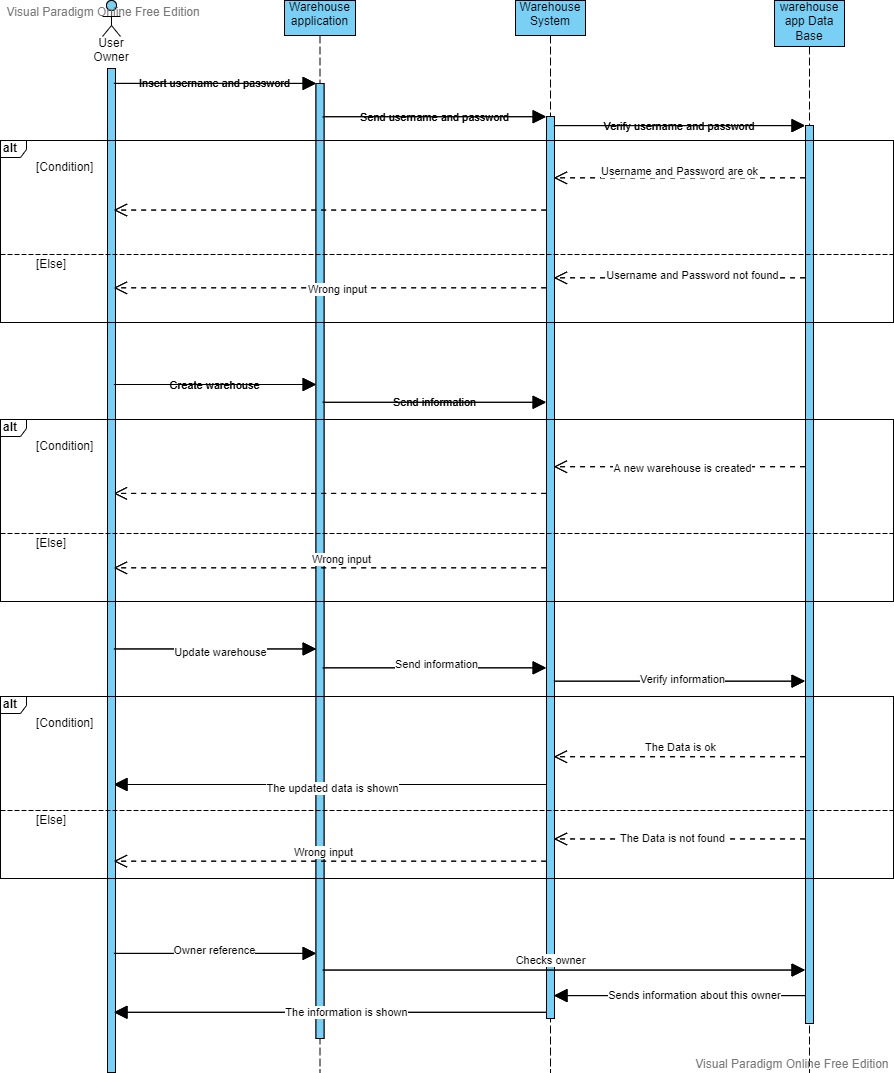
**UML-Use Case Diagram**

**UML-Sequence Diagram(за Администратор)**

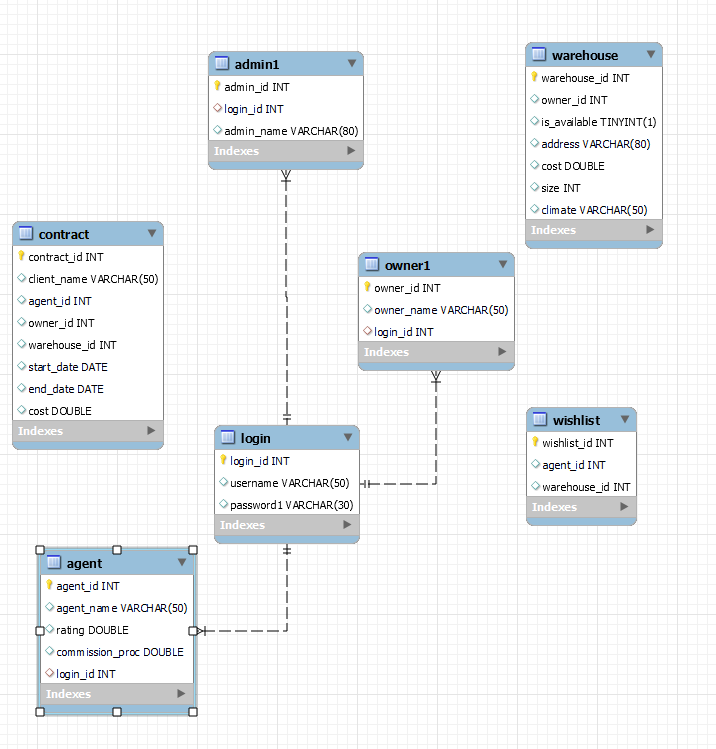
****

**UML-Sequence Diagram(за Агент)**

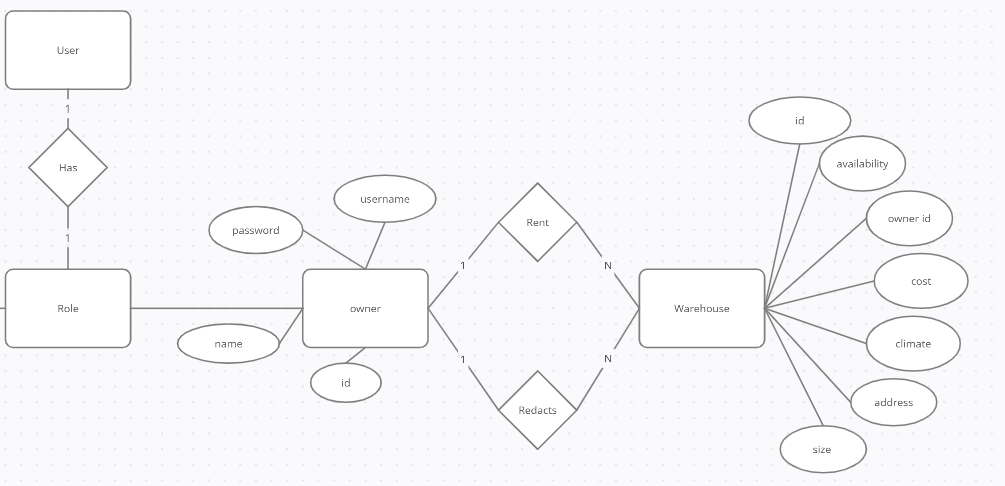
****

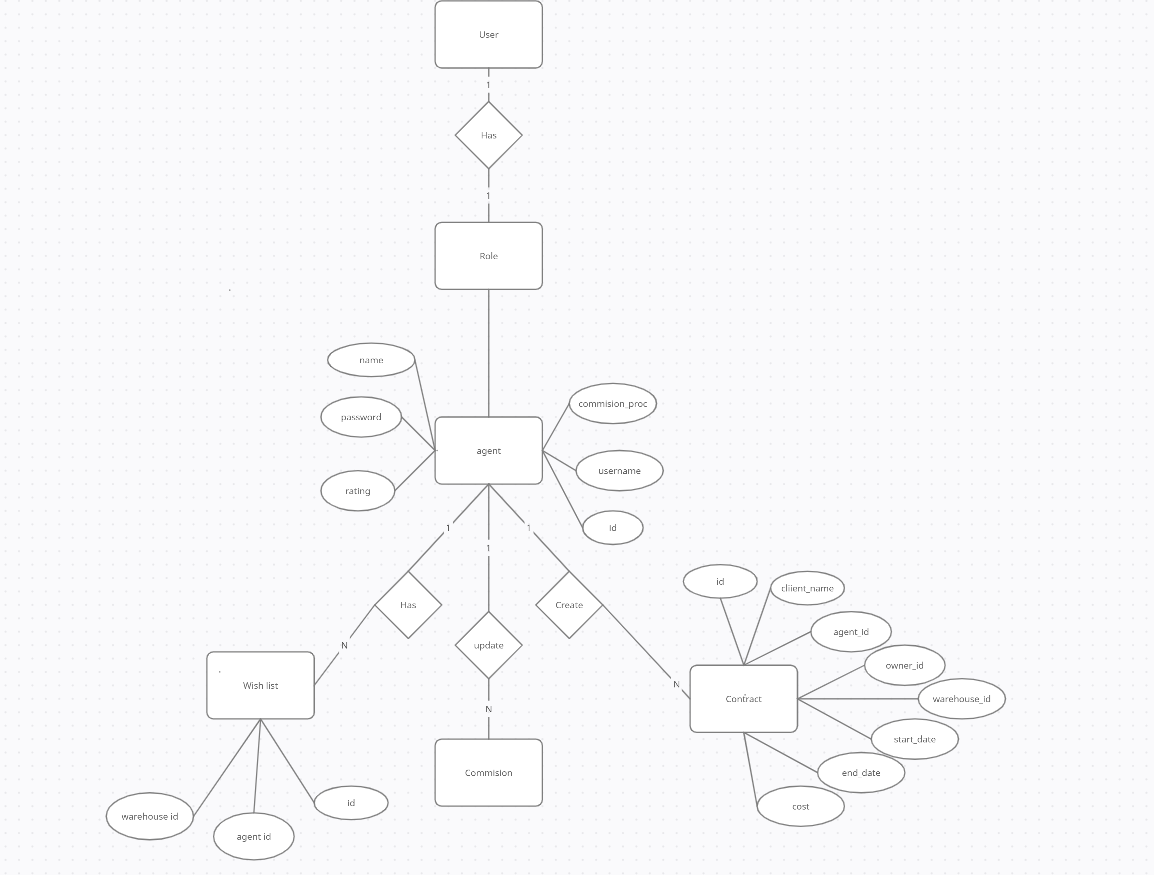
**UML Sequence Diagram(за Собственик)**

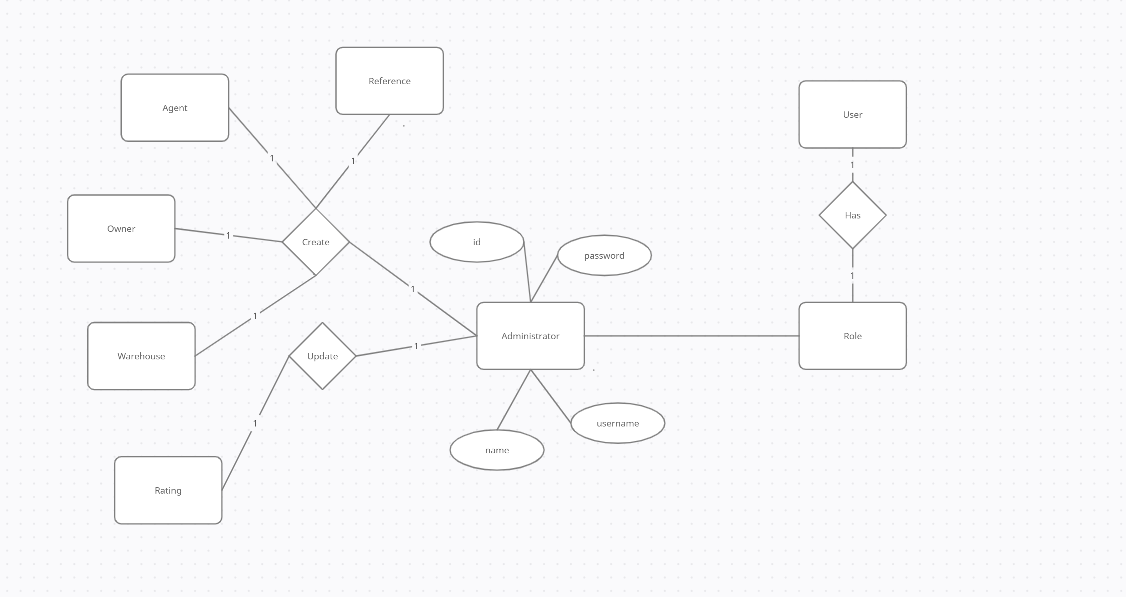
**Модел на База данни**

****

**Модел на Чен**

****

****

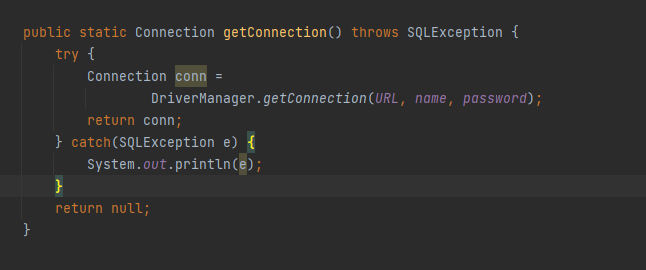
****

**Реализация на база данни**

Използваната база данние е MySQL. Избрана е, тъй като е една от най-популярните open source database management system която предлага лесна връзка с релационната база данни.

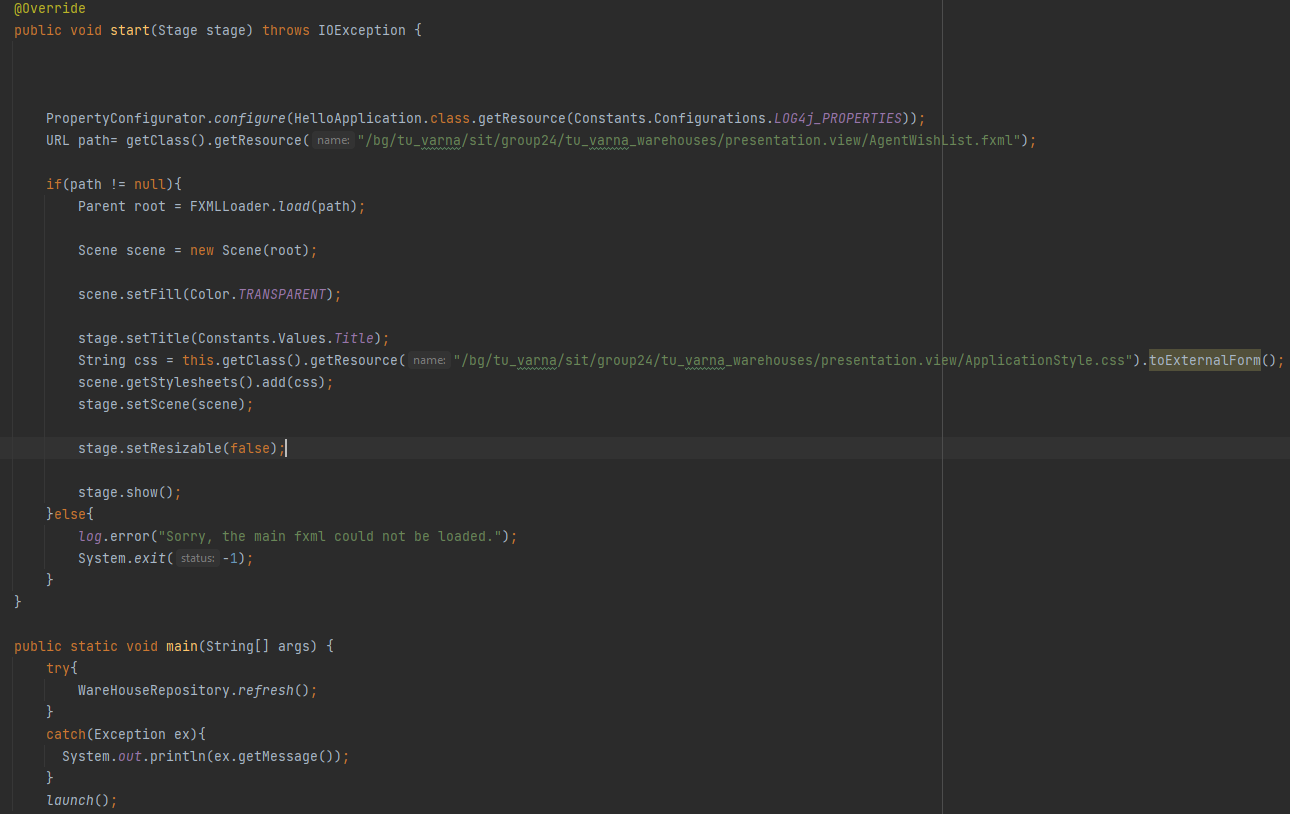
Цялата комуникация се извършва в директория Repositories, като за начина на комуникация използваме JDBC. От тази директория се викат функции с които се комуникира с базата данни.

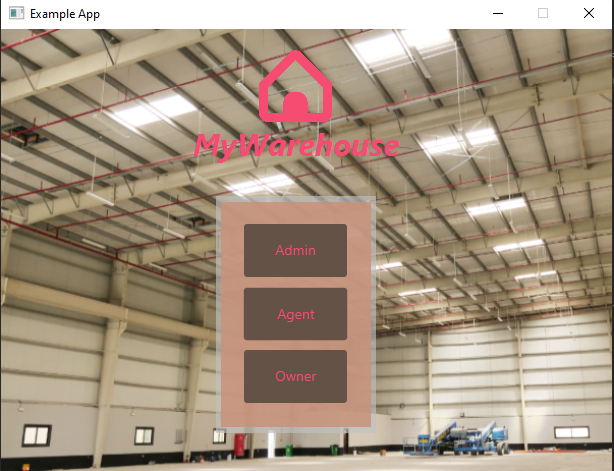
Връзка с MySQL:



**Бизнес логика и графичен интерфейс**

Бизнес логиката и графияния интерфейс се контролира от JavaFX. Принципа на работа е следния:



1. При стартиране се зарежда началната сцена в приложението за избор на потребител.
2. Всеки изглед е в папка ресурси и е с разширение .fxml.
3. Всеки изглед има съответен контролер, който реагира на събития(натискане нa бутон, валидация и текстови полета и др.)

Това е първоначалния изглед и показва различните достъпи в програмата.

**Модул за регистриране на събития**

Използва се библиотеката log4J, която е конфигурирана да разпечатва само в конзолата. Сметнато е за ненужно да се записва и в лог файл. Разполага с конфигурационен файл **log4j.properties:**

