|  |  |
| --- | --- |
|  | ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - ВАРНА  ФАКУЛТЕТ ПО ИЗЧИСЛИТЕЛНА ТЕХНИКА И АВТОМАТИЗАЦИЯ |

**КУРСОВ ПРОЕКТ**

по

Обектно-ориентирано програмиране­ – 2. част

Група 6

Михаил Бисеров Георгиев – СИТ 1. група, 19621820

Серкан Билгин Иляз – СИТ 3. група, 19621673

Тихомир Иванов Иванов – СИТ 3. група, 19621677

В А Р Н А

2 0 2 2 год.

## V. Склад с наличности

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

Системата поддържа два вида потребители администратор и оператори (складов агент) с различни роли за достъп до функционалностите в системата.

Операции за работа с потребители:

* Създаване на складови оператори от администратор;
* Създаване на доставчици;
* Създаване на клиенти;
* Създаване на каса (Парична наличност).

Системата поддържа операции за работа със събития:

* Създаване на номенклатури;
* Работа с фактури o Приемане на стока от доставчик на доставна цена; o Изписване на стока на продажна цена;
* Наблюдение за наличност на стоки в склада;
* Наблюдение за наличност на пари в касата;

Системата поддържа справки по произволен период за:

* Доставки и доставчици;
* Изписване и клиенти;
* Дейност на складовите оператори;
* За наличности в склада;
* Разходи, приходи, печалба.
* Движение на наличността в касата.

Системата поддържа Известия за събития:

* Критичен минимум и липса на стока;
* Критичен минимум и липса на парична наличност;

## Анализ на проблема

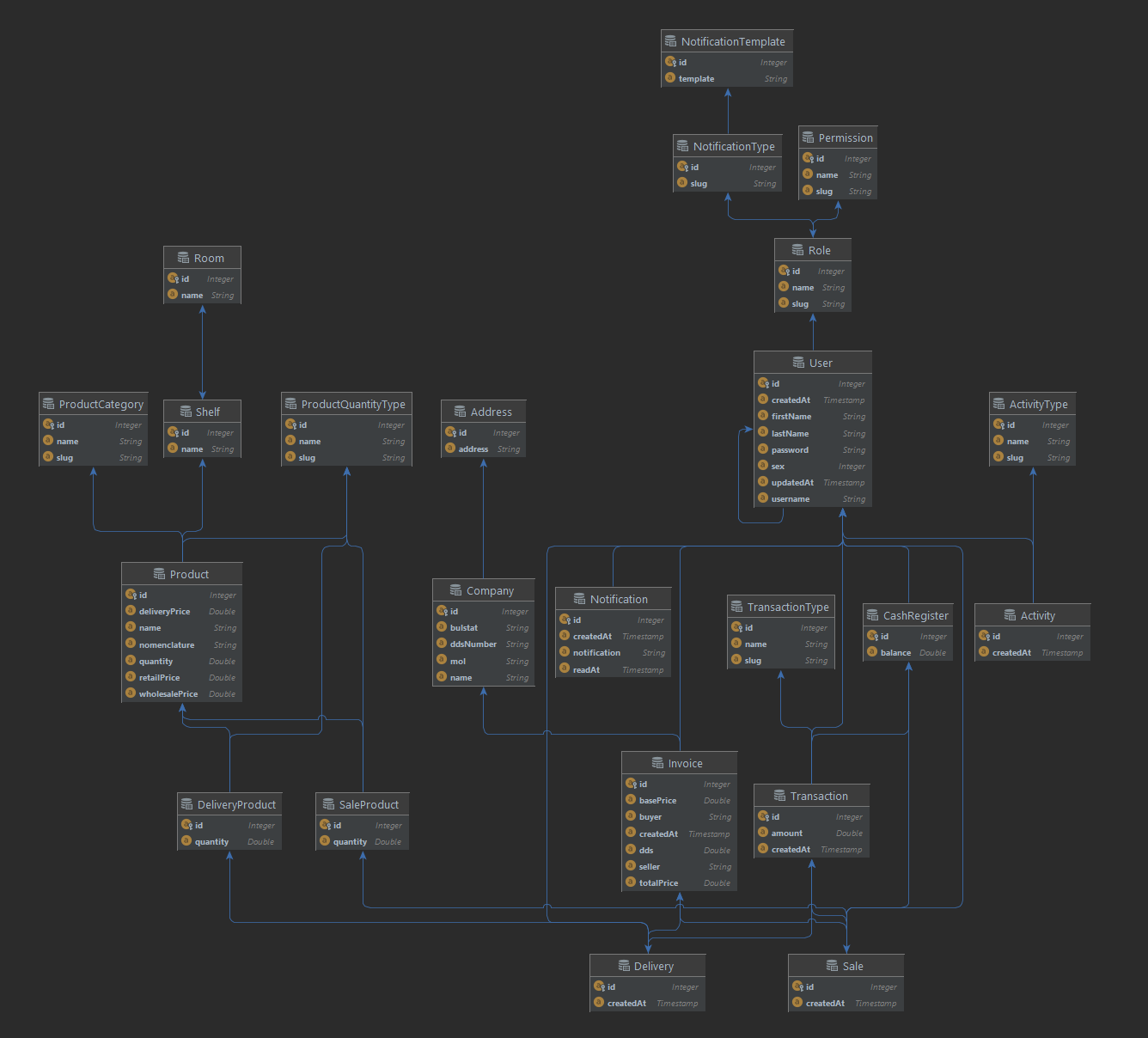
1. **Функционални изисквания:**

Софтуерът трябва да може да управлява и работи с множество потребители, да съхранява и обработва информация за складови помещения, каси и продукти. От него се изисква да поддържа създаване на фактури за доставка и продажба на стока, също и известия за критичен минимум на стоки и пари. На потребителя е необходимо да се предостави възможност за справка на стоки и пари в склада и касата съответно.

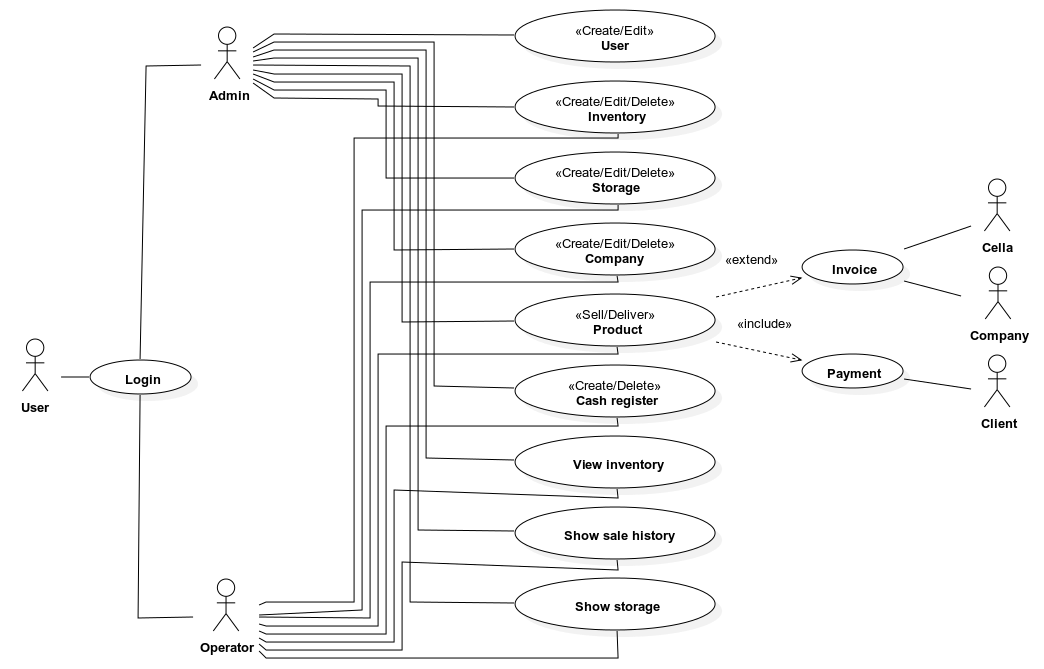
1. **Структура на проекта:** 
   * Контролери (controllers) – Всички класове, които управляват FXML файловете. Те заедно образуват презентационния слой.
   * Модели (entities) – Класовете, които прадставляват таблиците и техните връзки от базата данни.
   * Хранилища (repositories) – Всички класове, които управляват данните в базата данни. Заедно с моделите формират данновия слой.
   * Услуги (services) – Всички класове, които работят с данните от хранилищата. Те образуват бизнес логиката.
   * Общи услуги (utils) – Класове, които реализиращи ключови механизми за приложението.
   * Начална точка (warehouse) – Класовете за стариране на проекта.
   * Активи (assets) – Файлове за стилизация и изображения.
   * Конфигурационни файлове (configurations, META-INF)
   * Изгледи (views) – Всички FXML файлове.

## Проектиране на системата

1. **Entity-Relationship диаграма:**

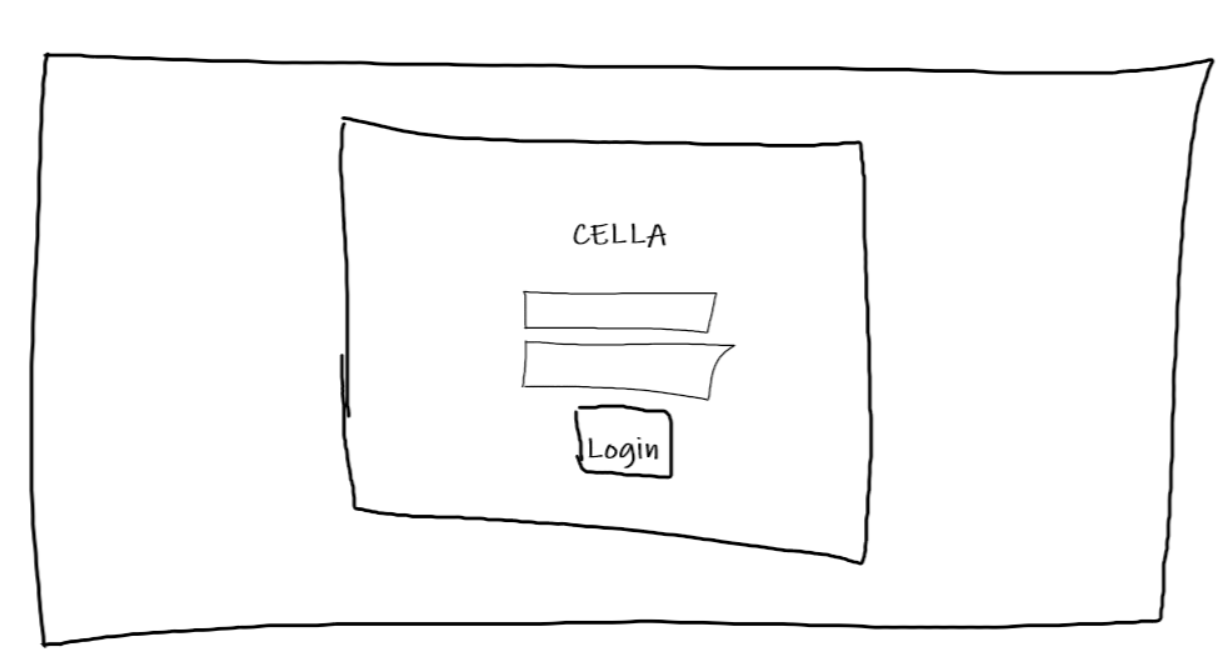


1. **Use case диаграма**

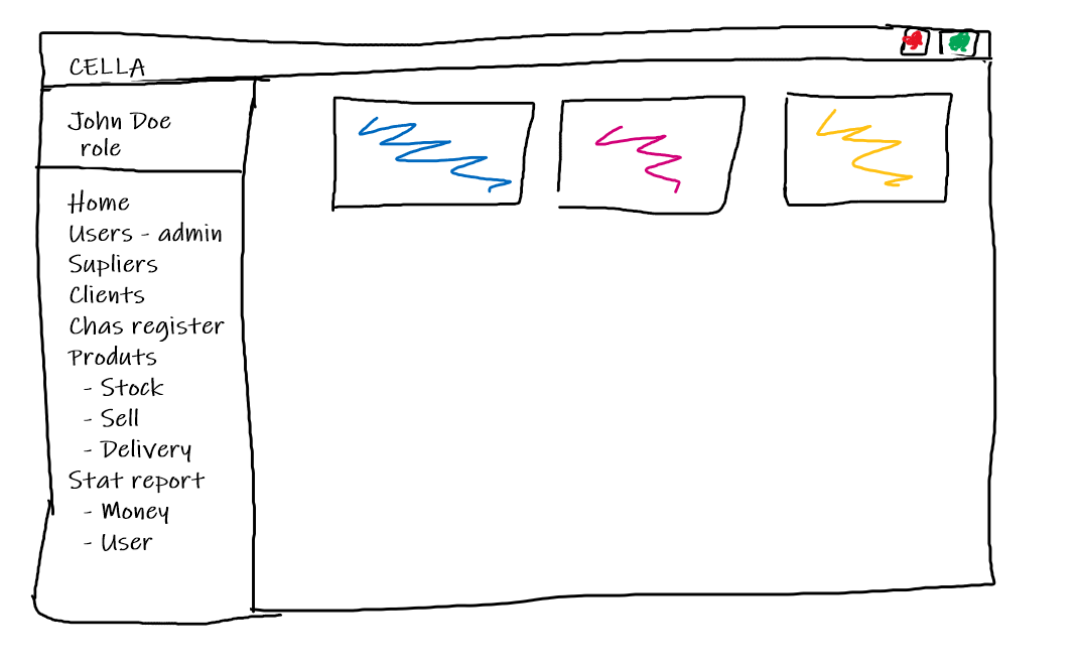


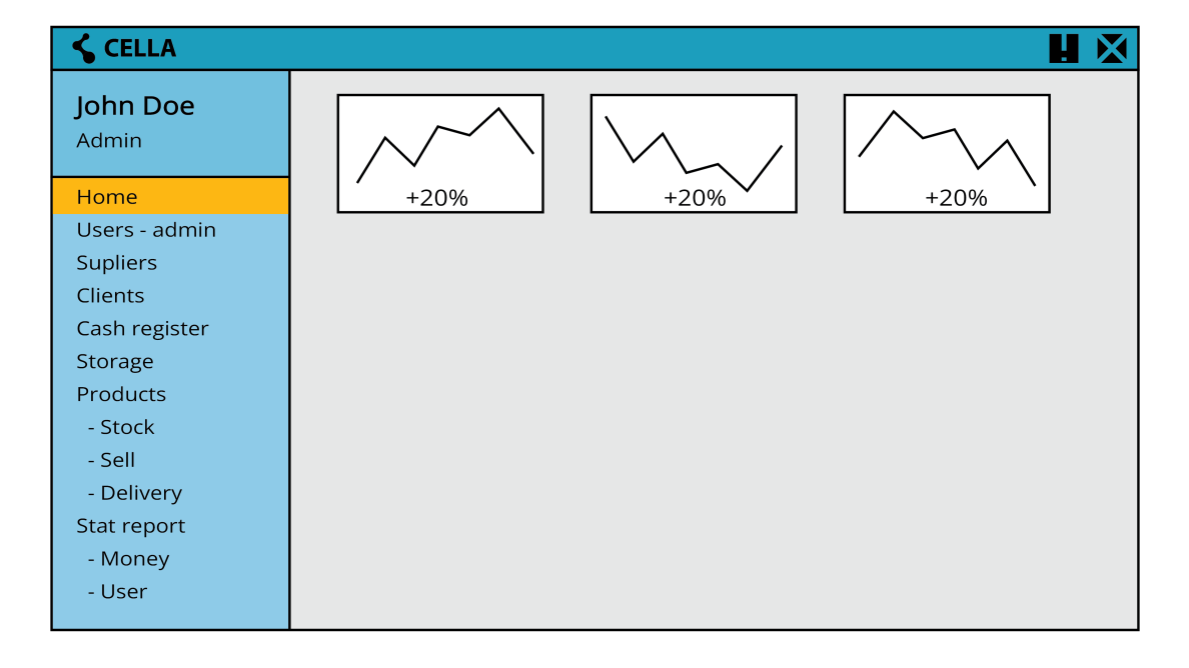
1. **Class диаграма**
2. **Sequence диаграма**
3. **Прототипи**

* Екран за вход

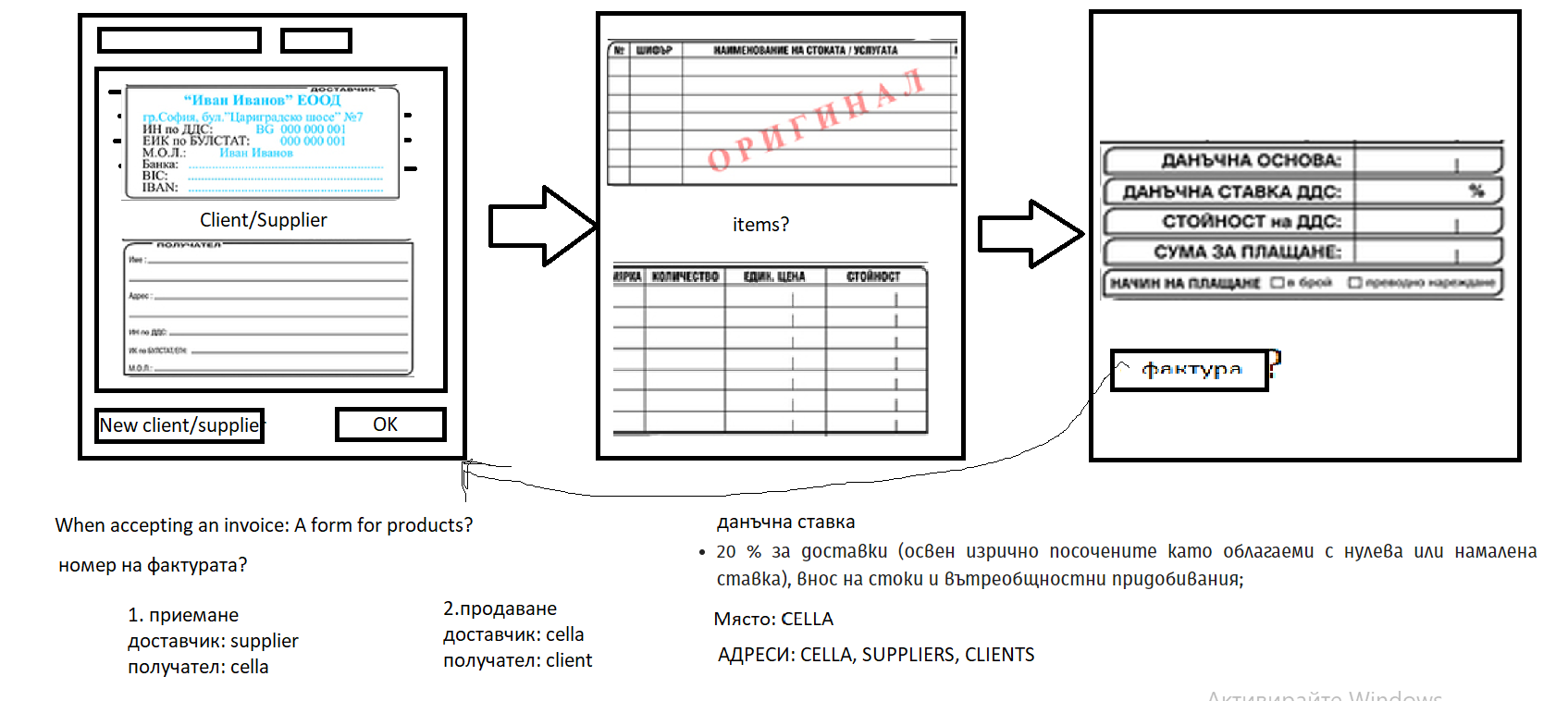


* Главен енран

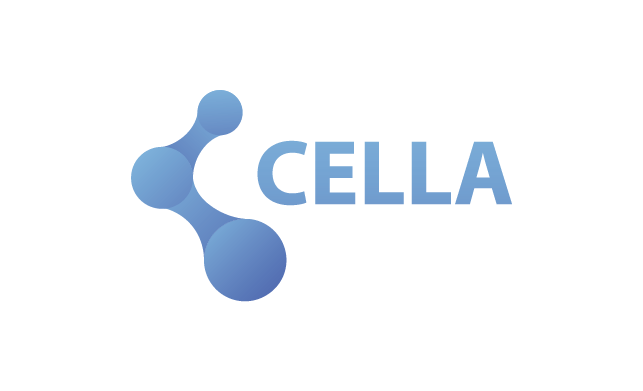




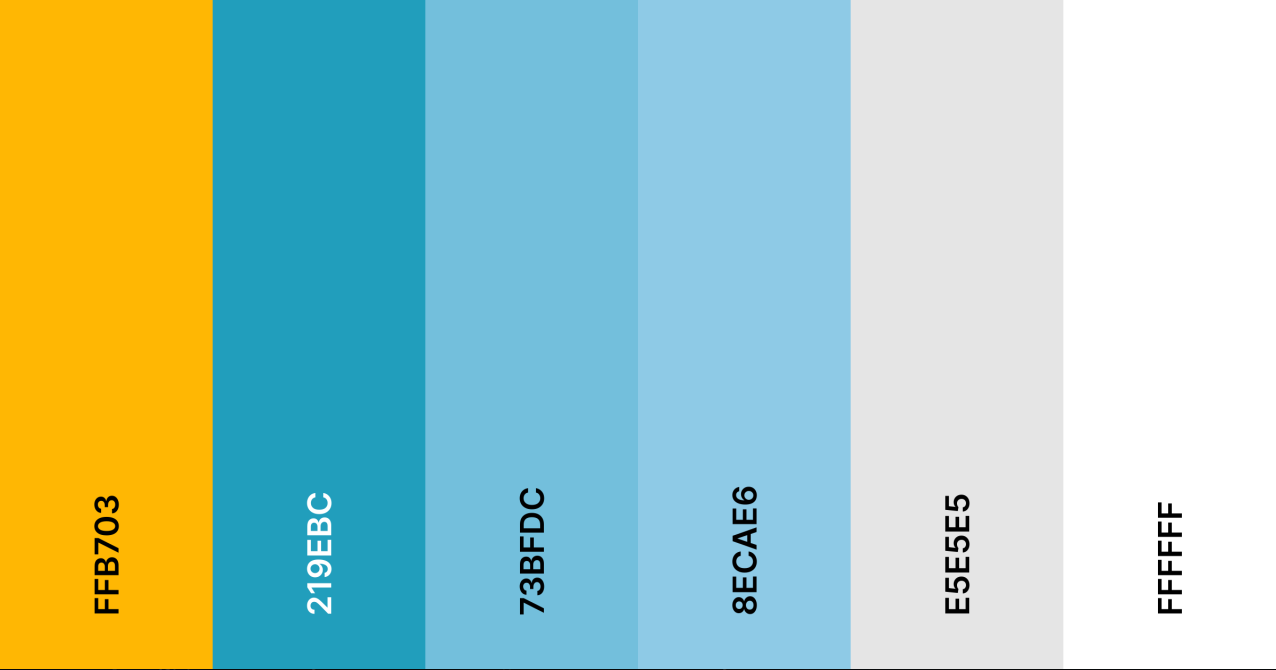
* Съветник (wizard) за доставка и продажба



1. **Лого и палитра**

****

****

****

## Реализация на системата

1. **База от данни**

Всички полета „id“ са целочислени (int) и са винаги положителни (unsigned), те задължително имат стойност (not null). Тя се калкулира автоматично (auto increment) и никога не се повтаря (unique) за самата таблица. Тези полета винаги са първичен ключ (primary key) за таблицата.

Всички полета, които завършват на „\_id“ съдържат стойност от полетата „id“, затова аналогично те също са int, unsigned.

Всички полета „created\_at“ и „updated\_at“ са времеви мерки (timestamp) и not null.

Всички полета „created\_by“ и „updated\_by“ съдържат стойност от полето „id“ от таблицата „users“, затова аналогично са int, unsigned.

Всички полета „slug“ и „name“ са низови (varchar) с дължина 45.

Полетата „username“, „first\_name“, „last\_name“ и „nomenclature“ също са varchar с дължина 45.

Полето „password“ е varchar с дължина 100, защото в него се съдържа криптирана парола.

Полето „sex“ е int, not null и има стойност по подразбиране (default value) 0. В него се записва 1 за „Мъж“ и 0 – „Жена“.

Полетата „address“, „mol“, „buyer“, „seller“, „template“ и „notification“ са varchar с дължина 255.

Полетата „quantity“, „retail\_price“, „wholesale\_price“, „delivery\_price“, „base\_price“, „dds“, „total\_price“, „balance“ и „amount“ са реални числа с висока точност (double), not null, unsigned и имат default value „0.00“.

Полето „bulstat“ е varchar с дължина 13, защото това е максималната стойност по стандарт. Стойностите могат да бъдат 9, 10 и 13.

Полето „dds\_number“ e varchar с дължина 15, защото по стандарт формата е „BG000000000“, където 0 са числа.

Полето „read\_at“ е timestamp с default value null, в него се записва текущото време, когато се прочете известието.

1. **Слой за работа с базата данни**

За реализация на връзката с базата данни е използван Hibernate. В entity класовете са използва JPA annotations.

В интерфейса ObjectRepository са дефинирани функциите „save“, „update“, „delete“, „getById“ и „getAll“. Всички останали класове repository имплементират този интерфейс. Функцията „save“ запазва подаденият ѝ обект в базата. Функцията „update“ актуализира подаденият ѝ обект в базата. Функцията „delete“ изтрива подаденият ѝ обект от базата. Функцията „getById“ връща обект от базата с подаденото „id“. Функцията „getAll“ връща всички обекти от дадената таблица.

Отделно от тези функции, в някои от хранилищата има и допълнителни, които връщат обекти по дадени критерии.

1. **Бизнес логика и графичен интерфейс (GUI)**

В пакета services, се съдържат класове, които изпълняват следните функционалности:

* комуникират с базата данни чрез repositories;
* обработват данни;

В пакета utils се съдържат класове за:

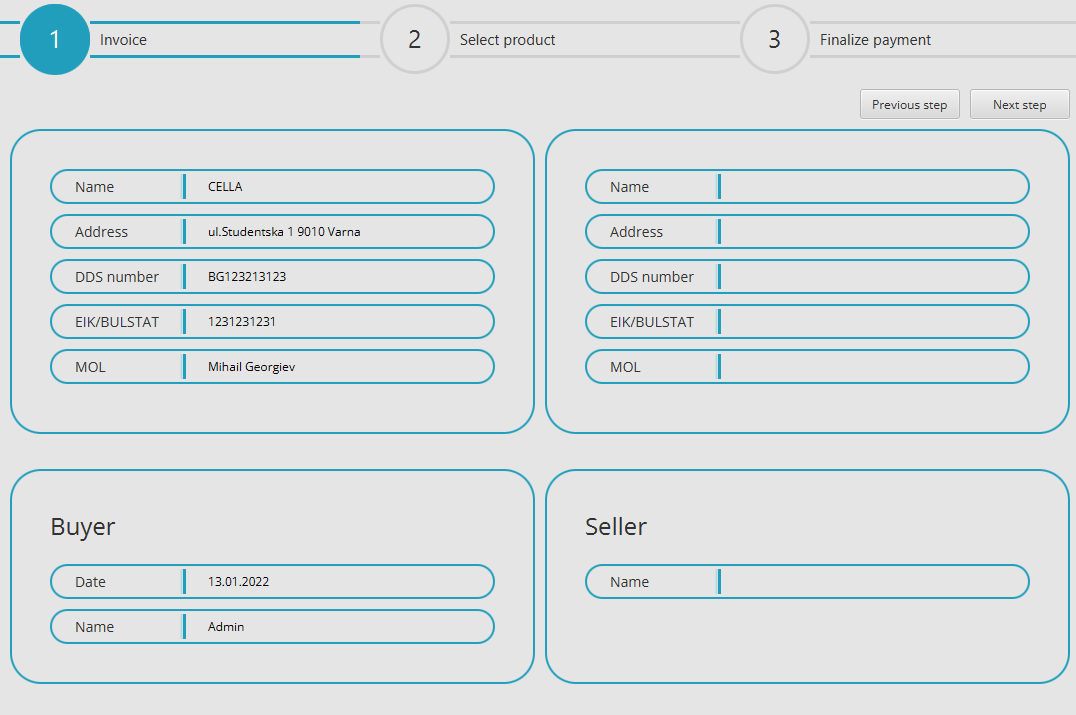
* сесия (session);
* връзка с базата (connection);
* диалози (dialogs);
* управление на прозорците (window management);
* форматиране на таблиците (table resizer);
* имплементация на wizard;
* криптиране на паролите (hasher);

Графичният интерфейс е реализиран чрез JavaFX и се състои от контролери и FXML файлове. За стилизация на приложението се използва CSS. Текстът в приложението има специално зададен шрифт „Open Sans“, който върви комплектовано с приложението. За икони e използвана библиотеката „fontawesomefx“. За известията е използвана библиотеката „controlsfx“.

Реализирани прототипи:

* Картина, която съдържа текст

  Описанието е генерирано автоматичноЕкран за вход
* Картина, която съдържа маса

  Описанието е генерирано автоматичноГлавен екран
* Wizard

1. **Регистриране на събитията**

Регистрирането на събитията е реализирано чрез log4j 2, като само грешките при операциите с базата данни се регистрират.

## Тестови резултати

Картина, която съдържа текст

Описанието е генерирано автоматично За реализацията на тестовете е използван jUnit 5.

1. **UserServiceTest**

* Should get admin user (функционален тест) – проверява дали при подадено „id“ 1 връща потребител с роля „Администратор“.
* Should get operator user (функционален тест) – проверява дали при подадено „id“ 2 връща потребител с роля „Оператор“.
* Should add new user (интеграционен тест) – проверява дали е възможно създаването на потребител
* Should update user (функционален тест) – проверява дали е възможна актуализация на новосъздадения потребител

1. **RoleServiceTest**

* Should get role by slug (функционален тест) – проверява вземането на роля по предварително подадени slugs

1. **TransactionTypeServiceTest**

* Should get transaction by slug (функционален тест) – проверява вземането на тип транзакция по предварително подадени slugs

1. **ProductServiceTest**

* Should add product (интеграционен тест) – проверява добавянето на продукти
* Should update product (функционален тест) – проверява актуализацията на новодобавения продукт
* Should delete product (интеграционен тест) – проверява изтриването на новодобавения продукт

1. **CashRegisterServiceTest**

* Should add new cash register (интеграционен тест) – проверява добавянето на каса
* Should delete cash register (интеграционен тест) – проверява изтриването на новодобавената каса

1. **StorageServiceTest**

* Should add new room (интеграционен тест) – проверява добавянето на стая
* Should add new shelf (интеграционен тест) – проверява добавянето на рафт към новодобавената стая
* Should set room name (функционален тест) – проверява актуализацията на името на новодобавената стая
* Should set shelf name (функционален тест) – проверява актуализацията на името на новодобавения рафт
* Should delete shelf (интеграционен тест) – проверява изтриването на новодобавения рафт
* Should delete room (интеграционен тест) – проверява изтриването на новодобавената стая

1. **ActivityTypeServiceTest**
   * Should get activity type by slug (функционален тест) – проверява вземането на тип дейност по предварително подадени slugs
2. **CompanyServiceTest**

* Should add new company (интеграционен тест) – проверява добавянето на компания
* Should update company (функционален тест) – проверява актуализацията на новодобавената компания
* Should delete company (интеграционен тест) – проверява изтриването на новодобавената компания

## Използвани ресурси

* <https://docs.oracle.com/>
* <https://coolors.co/ffb703-219ebc-73bfdc-8ecae6-e5e5e5-ffffff>
* <https://logging.apache.org/log4j/2.x/manual/configuration.html>
* <https://www.devglan.com/spring-mvc/storing-hashed-password-database-java>
* <https://www.youtube.com/watch?v=occdZWprRjg>
* <https://www.youtube.com/watch?v=9XJicRt_FaI>
* <https://www.youtube.com/watch?v=4JSaH0dNK5M>
* <https://www.youtube.com/watch?v=QJddHc41xrM>
* <https://www.youtube.com/watch?v=rk2zcyzeK3U>
* Препоръки за дизайна и изработка на лого – специални благодарности на Емил Митев, ИД 2. курс – 2. група.