

Технически университет Варна

Проект по ООП2

на:

Габриел Георгиев Ф.Н.:19621668 и Христомиир Желев Ф.Н.:19621713

Факултет: ФИТА Катедра: КНТ Специалност: СИТ

**VII. Куриерска фирма**

Да се разработи информационна система, предоставяща услуга за изпращане и

получаване на пратки по куриер. Програмата съхранява и обработва данни за

създадени пратки, обработени от куриер и доставени до адрес.

Системата поддържа три вида потребители администратор, куриери и клиенти (с

различни роли за достъп до функционалностите в системата.

**Операции за работа с потребители:**

* създаване на куриерска фирма(и) от администратор(име, офиси в градове и
* др.);
* създаване на куриер към фирма от администратор;
* регистриране на клиенти на куриерската фирма от куриер.

**Системата поддържа операции за работа със изпращане и получаване на пратки:**

* регистриране на пратка в офис на фирмата. Различните категории пратки
* (плик, колет, голям пакет, товар) имат различни цени;
* транзит на пратката до местоназначението(извършва се автоматично от
* системата на базата на времезаделяща техника!);
* статус на получаването – получено, неполучена, отказана;

**Системата поддържа Справки по произволен период за:**

* пратки и тяхното състояние;
* възможност за проследяване на статуса на дадена пратка;
* статистика на куриерската фирма;
* работа на куриерите;
* статистика за клиентите на куриерските фирми.

**Системата поддържа Известия за:**

* неполучени пратки;
* отказани пратки;

**Функционални изисквания**

1. Работа с различни потребителски нива на правомощия

1.1. потребител/клиент:

1.1.1. подаване на пратка

1.1.2. отказване на пратка

1.1.3. проследяване на пратка

1.1.4. приемане на пратка

1.1.5. заплащане за услугата

1.2. куриер:

1.2.1. приема пратка от подател

1.2.2. регистрира клиенти

1.2.3. обработва данни за пратка

1.2.4. доставя до получател

1.3. администратор

1.3.1. добавя потребители/куриери

1.3.2. променя правомощия на други потребители

1.3.3. въвежда нови куриерски фирми във системата

1.3.4. добавя/редактира офиси на куриерските фирми

2. Поддръжка на операции за работа с пратки:

2.1. регистриране на пратка

2.2. транзит на пратката

2.3. статус на получаването

3. Изготвяне на справки по произволен период

3.1. пратки и тяхното състояние

3.2. възможност за проследяване на статуса на дадена пратка

3.3. статистика на куриерската фирма

3.4. работа на куриерите

3.5. статистика за клиентите на куриерските фирми

4. поддържа Известия

4.1. неполучени пратки

4.2. отказани пратки

**Структура на проекта**

****

**проектиране на системата**

**База Данни:**

**CREATE DATABASE `deliverydb` /\*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci \*/ /\*!80016 DEFAULT ENCRYPTION='N' \*/;**

**CREATE TABLE `admins` (**

**`id\_admin` int NOT NULL AUTO\_INCREMENT,**

**`name` varchar(45) NOT NULL,**

**`phone` char(12) NOT NULL,**

**`password` varchar(30) NOT NULL,**

**PRIMARY KEY (`id\_admin`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**CREATE TABLE `categories` (**

**`id\_category` int NOT NULL AUTO\_INCREMENT,**

**`category` varchar(45) DEFAULT NULL,**

**PRIMARY KEY (`id\_category`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**CREATE TABLE `cities` (**

**`id\_city` int NOT NULL AUTO\_INCREMENT,**

**`city` varchar(45) DEFAULT NULL,**

**PRIMARY KEY (`id\_city`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**CREATE TABLE `companies` (**

**`id\_company` int NOT NULL AUTO\_INCREMENT,**

**`company` varchar(45) DEFAULT NULL,**

**PRIMARY KEY (`id\_company`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=20 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**CREATE TABLE `couriers` (**

**`id\_courier` int NOT NULL AUTO\_INCREMENT,**

**`name` varchar(45) NOT NULL,**

**`phone` char(12) NOT NULL,**

**`password` varchar(30) NOT NULL,**

**`id\_office` int NOT NULL,**

**PRIMARY KEY (`id\_courier`),**

**KEY `couriers\_ibfk\_1` (`id\_office`),**

**CONSTRAINT `couriers\_ibfk\_1` FOREIGN KEY (`id\_office`) REFERENCES `office` (`id\_office`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=16 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**CREATE TABLE `customers` (**

**`id\_customer` int NOT NULL AUTO\_INCREMENT,**

**`name` varchar(45) NOT NULL,**

**`phone` char(12) NOT NULL,**

**`password` varchar(30) NOT NULL,**

**`id\_city` int DEFAULT NULL,**

**`address` varchar(45) CHARACTER SET utf8 COLLATE utf8\_general\_ci DEFAULT NULL,**

**PRIMARY KEY (`id\_customer`),**

**KEY `id\_city` (`id\_city`),**

**CONSTRAINT `customers\_ibfk\_1` FOREIGN KEY (`id\_city`) REFERENCES `cities` (`id\_city`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**CREATE TABLE `office` (**

**`id\_office` int NOT NULL AUTO\_INCREMENT,**

**`id\_company` int DEFAULT NULL,**

**`id\_city` int DEFAULT NULL,**

**`address` varchar(45) DEFAULT NULL,**

**PRIMARY KEY (`id\_office`),**

**KEY `id\_company` (`id\_company`),**

**KEY `id\_city` (`id\_city`),**

**CONSTRAINT `office\_ibfk\_1` FOREIGN KEY (`id\_company`) REFERENCES `companies` (`id\_company`),**

**CONSTRAINT `office\_ibfk\_2` FOREIGN KEY (`id\_city`) REFERENCES `cities` (`id\_city`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=15 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**CREATE TABLE `orders` (**

**`id\_order` int NOT NULL AUTO\_INCREMENT,**

**`id\_category` int DEFAULT NULL,**

**`id\_office\_sender` int DEFAULT NULL,**

**`id\_office\_recipient` int DEFAULT NULL,**

**`id\_customer\_sender` int DEFAULT NULL,**

**`id\_customer\_recipient` int DEFAULT NULL,**

**`id\_courier` int DEFAULT NULL,**

**`id\_status` int DEFAULT NULL,**

**`fragile` tinyint(1) DEFAULT NULL,**

**`paid` tinyint(1) DEFAULT NULL,**

**`cash\_on\_delivery` double DEFAULT NULL,**

**`delivery\_to\_address` tinyint(1) DEFAULT NULL,**

**`address` varchar(45) DEFAULT NULL,**

**`acceptance\_by\_sender` datetime DEFAULT NULL,**

**`customer\_delivery` datetime DEFAULT NULL,**

**PRIMARY KEY (`id\_order`),**

**KEY `id\_category` (`id\_category`),**

**KEY `id\_office\_sender` (`id\_office\_sender`),**

**KEY `id\_office\_recipient` (`id\_office\_recipient`),**

**KEY `id\_customer\_sender` (`id\_customer\_sender`),**

**KEY `id\_customer\_recipient` (`id\_customer\_recipient`),**

**KEY `id\_courier` (`id\_courier`),**

**KEY `id\_status` (`id\_status`),**

**CONSTRAINT `orders\_ibfk\_1` FOREIGN KEY (`id\_category`) REFERENCES `categories` (`id\_category`),**

**CONSTRAINT `orders\_ibfk\_2` FOREIGN KEY (`id\_office\_sender`) REFERENCES `office` (`id\_office`),**

**CONSTRAINT `orders\_ibfk\_3` FOREIGN KEY (`id\_office\_recipient`) REFERENCES `office` (`id\_office`),**

**CONSTRAINT `orders\_ibfk\_4` FOREIGN KEY (`id\_customer\_sender`) REFERENCES `customers` (`id\_customer`),**

**CONSTRAINT `orders\_ibfk\_5` FOREIGN KEY (`id\_customer\_recipient`) REFERENCES `customers` (`id\_customer`),**

**CONSTRAINT `orders\_ibfk\_6` FOREIGN KEY (`id\_courier`) REFERENCES `couriers` (`id\_courier`),**

**CONSTRAINT `orders\_ibfk\_7` FOREIGN KEY (`id\_status`) REFERENCES `statuses` (`id\_status`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**CREATE TABLE `price\_list` (**

**`id\_company` int NOT NULL,**

**`id\_category` int NOT NULL,**

**`price` double DEFAULT NULL,**

**PRIMARY KEY (`id\_company`,`id\_category`),**

**KEY `id\_category` (`id\_category`),**

**CONSTRAINT `price\_list\_ibfk\_1` FOREIGN KEY (`id\_company`) REFERENCES `companies` (`id\_company`),**

**CONSTRAINT `price\_list\_ibfk\_2` FOREIGN KEY (`id\_category`) REFERENCES `categories` (`id\_category`)**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**CREATE TABLE `statuses` (**

**`id\_status` int NOT NULL AUTO\_INCREMENT,**

**`status\_name` varchar(45) DEFAULT NULL,**

**PRIMARY KEY (`id\_status`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**GUI:**

**fxml**

**<?xml version="1.0" encoding="UTF-8"?>**

**<?import javafx.scene.control.Button?>**

**<?import javafx.scene.control.Hyperlink?>**

**<?import javafx.scene.control.Label?>**

**<?import javafx.scene.control.PasswordField?>**

**<?import javafx.scene.control.TextField?>**

**<?import javafx.scene.image.Image?>**

**<?import javafx.scene.image.ImageView?>**

**<?import javafx.scene.layout.AnchorPane?>**

**<?import javafx.scene.text.Font?>**

**<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="447.0" prefWidth="309.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="bg.tu\_varna.sit.group17.controllers.LoginController">**

**<children>**

**<ImageView fitHeight="116.0" fitWidth="299.0" layoutX="15.0" layoutY="13.0" pickOnBounds="true" preserveRatio="true">**

**<image>**

**<Image url="@../img/GandHlogo.png" />**

**</image>**

**</ImageView>**

**<Label id="title" layoutX="71.0" layoutY="133.0" text="ВЛИЗАНЕ В АКАУНТ" textFill="WHITE" />**

**<Button id="register" alignment="CENTER\_RIGHT" contentDisplay="BOTTOM" layoutX="127.0" layoutY="278.0" mnemonicParsing="false" onAction="#registerFx" prefHeight="119.0" prefWidth="139.0" text="Регистрирай се" />**

**<Hyperlink id="forgot" fx:id="forgotPassword" layoutX="61.0" layoutY="405.0" text="забравих си паролата" />**

**<TextField fx:id="phone" layoutX="29.0" layoutY="182.0" prefHeight="38.0" prefWidth="235.0" promptText="телефон" />**

**<PasswordField fx:id="pass" layoutX="29.0" layoutY="229.0" prefHeight="38.0" prefWidth="235.0" promptText="парола" />**

**<Button id="login" alignment="CENTER\_RIGHT" contentDisplay="BOTTOM" layoutX="29.0" layoutY="279.0" mnemonicParsing="false" onAction="#loginFx" prefHeight="81.0" prefWidth="82.0" text="Влез" textOverrun="CENTER\_ELLIPSIS" />**

**<Label id="errorMessage" layoutX="126.0" layoutY="159.0" textFill="RED">**

**<font>**

**<Font size="13.0" />**

**</font>**

**</Label>**

**</children>**

**</AnchorPane>**

| **<?xml version="1.0" encoding="UTF-8"?>** |  |
| --- | --- |
|  |  |
|  | **<?import javafx.scene.control.Button?>** |
|  | **<?import javafx.scene.control.ComboBox?>** |
|  | **<?import javafx.scene.control.Label?>** |
|  | **<?import javafx.scene.control.PasswordField?>** |
|  | **<?import javafx.scene.control.TextField?>** |
|  | **<?import javafx.scene.image.Image?>** |
|  | **<?import javafx.scene.image.ImageView?>** |
|  | **<?import javafx.scene.layout.AnchorPane?>** |
|  | **<?import javafx.scene.text.Font?>** |
|  |  |
|  | **<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="620.0" prefWidth="309.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="bg.tu\_varna.sit.group17.controllers.RegisterController">** |
|  | **<children>** |
|  | **<PasswordField fx:id="repeatPassword" layoutX="25.0" layoutY="432.0" prefHeight="38.0" prefWidth="243.0" promptText="потвърди паролата" />** |
|  | **<ImageView fitHeight="116.0" fitWidth="299.0" layoutX="15.0" layoutY="13.0" pickOnBounds="true" preserveRatio="true">** |
|  | **<image>** |
|  | **<Image url="@../img/GandHlogo.png" />** |
|  | **</image>** |
|  | **</ImageView>** |
|  | **<Label id="title" layoutX="92.0" layoutY="129.0" text="РЕГИСТРАЦИЯ" textFill="WHITE" />** |
|  |  |
|  | **<Button id="register" alignment="TOP\_CENTER" contentDisplay="BOTTOM" layoutX="29.0" layoutY="479.0" mnemonicParsing="false" onAction="#register" prefHeight="119.0" prefWidth="142.0" text="Регистрирай се" />** |
|  | **<TextField fx:id="phone" layoutX="25.0" layoutY="244.0" prefHeight="38.0" prefWidth="243.0" promptText="телефон" />** |
|  | **<Button id="login" alignment="TOP\_CENTER" contentDisplay="BOTTOM" layoutX="187.0" layoutY="481.0" mnemonicParsing="false" onAction="#login" prefHeight="81.0" prefWidth="82.0" text="Влез" textOverrun="CENTER\_ELLIPSIS" />** |
|  | **<TextField fx:id="name" layoutX="25.0" layoutY="197.0" prefHeight="38.0" prefWidth="243.0" promptText=" име" />** |
|  | **<PasswordField fx:id="password" layoutX="23.0" layoutY="385.0" prefHeight="38.0" prefWidth="243.0" promptText="парола" />** |
|  | **<ComboBox fx:id="cBox0" layoutX="25.0" layoutY="290.0" prefHeight="38.0" prefWidth="243.0" promptText="град" />** |
|  | **<TextField fx:id="address" layoutX="23.0" layoutY="336.0" prefHeight="38.0" prefWidth="243.0" promptText="адрес" />** |
|  | **</children>** |
|  | **</AnchorPane>** |

| **<?xml version="1.0" encoding="UTF-8"?>** |  |
| --- | --- |
|  |  |
|  | **<?import javafx.scene.control.Button?>** |
|  | **<?import javafx.scene.control.ComboBox?>** |
|  | **<?import javafx.scene.control.DatePicker?>** |
|  | **<?import javafx.scene.control.Label?>** |
|  | **<?import javafx.scene.control.MenuButton?>** |
|  | **<?import javafx.scene.control.MenuItem?>** |
|  | **<?import javafx.scene.control.TableView?>** |
|  | **<?import javafx.scene.control.TextField?>** |
|  | **<?import javafx.scene.image.Image?>** |
|  | **<?import javafx.scene.image.ImageView?>** |
|  | **<?import javafx.scene.layout.AnchorPane?>** |
|  | **<?import javafx.scene.layout.Pane?>** |
|  | **<?import javafx.scene.text.Font?>** |
|  |  |
|  | **<Pane id="mainPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="458.0" prefWidth="780.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="bg.tu\_varna.sit.group17.controllers.HomeController">** |
|  | **<children>** |
|  | **<AnchorPane id="filterPane" layoutY="109.0" prefHeight="351.0" prefWidth="206.0">** |
|  | **<children>** |
|  | **<Button id="vij" alignment="CENTER" contentDisplay="CENTER" layoutX="66.0" layoutY="210.0" mnemonicParsing="false" onAction="#filter" prefHeight="50.0" prefWidth="60.0" textOverrun="CENTER\_WORD\_ELLIPSIS" />** |
|  | **<Label layoutX="9.0" layoutY="83.0" text="от" />** |
|  | **<DatePicker fx:id="dateFrom" layoutX="32.0" layoutY="79.0" prefHeight="26.0" prefWidth="159.0" />** |
|  | **<Label layoutX="8.0" layoutY="127.0" text="до" />** |
|  | **<DatePicker fx:id="dateTo" layoutX="32.0" layoutY="122.0" prefHeight="26.0" prefWidth="159.0" />** |
|  | **<Label layoutX="65.0" layoutY="4.0" prefHeight="26.0" prefWidth="70.0" text="функция " />** |
|  | **<ComboBox fx:id="functions" layoutX="9.0" layoutY="30.0" prefHeight="42.0" prefWidth="185.0" />** |
|  | **<TextField fx:id="phone" layoutX="24.0" layoutY="166.0" prefHeight="40.0" prefWidth="167.0" promptText="Телефон на клиент" />** |
|  | **<TextField fx:id="IdOrder" layoutX="20.0" layoutY="268.0" prefHeight="40.0" prefWidth="159.0" promptText="ID пратка" />** |
|  | **<Button layoutX="20.0" layoutY="308.0" mnemonicParsing="false" onAction="#cancelOrder" prefHeight="40.0" prefWidth="159.0" text="Откажи пратка" />** |
|  | **</children>** |
|  | **</AnchorPane>** |
|  | **<TableView fx:id="table" layoutX="206.0" layoutY="108.0" prefHeight="351.0" prefWidth="574.0" />** |
|  | **<Pane id="navPane" prefHeight="110.0" prefWidth="780.0">** |
|  | **<children>** |
|  | **<Pane id="acPane" prefHeight="99.0" prefWidth="200.0">** |
|  | **<children>** |
|  | **<MenuButton id="userName" alignment="TOP\_LEFT" layoutX="1.0" layoutY="65.0" mnemonicParsing="false" prefHeight="40.0" prefWidth="199.0" text="user name" textFill="#333333">** |
|  | **<items>** |
|  | **<MenuItem mnemonicParsing="false" text="смени аватара" />** |
|  | **<MenuItem mnemonicParsing="false" text="излез" />** |
|  | **</items>** |
|  | **<font>** |
|  | **<Font size="18.0" />** |
|  | **</font>** |
|  | **</MenuButton>** |
|  | **<ImageView fitHeight="60.0" fitWidth="60.0" layoutX="14.0" layoutY="5.0" pickOnBounds="true" preserveRatio="true">** |
|  | **<image>** |
|  | **<Image url="@../img/defalut%20avatar.png" />** |
|  | **</image>** |
|  | **</ImageView>** |
|  | **</children>** |
|  | **</Pane>** |
|  | **<Button id="spravki" disable="true" layoutX="206.0" mnemonicParsing="false" prefHeight="110.0" prefWidth="159.0" text="справки" />** |
|  | **<Button id="pratka" layoutX="359.0" mnemonicParsing="false" onAction="#registerPratka" prefHeight="110.0" prefWidth="159.0" text="+пратка" />** |
|  | **<Button id="izvestia" fx:id="notificationBell" layoutX="696.0" layoutY="14.0" minHeight="10.0" minWidth="53.0" mnemonicParsing="false" onAction="#notificationBellClick" prefHeight="71.0" prefWidth="70.0" />** |
|  | **<Button id="firma" layoutX="518.0" mnemonicParsing="false" onAction="#firma" prefHeight="110.0" prefWidth="159.0" text="фирма" />** |
|  | **</children>** |
|  | **</Pane>** |
|  | **</children>** |
|  | **</Pane>** |

| **<?xml version="1.0" encoding="UTF-8"?>** |  |
| --- | --- |
|  |  |
|  | **<?import javafx.scene.control.Button?>** |
|  | **<?import javafx.scene.control.CheckBox?>** |
|  | **<?import javafx.scene.control.ComboBox?>** |
|  | **<?import javafx.scene.control.DatePicker?>** |
|  | **<?import javafx.scene.control.MenuButton?>** |
|  | **<?import javafx.scene.control.MenuItem?>** |
|  | **<?import javafx.scene.control.TextField?>** |
|  | **<?import javafx.scene.image.Image?>** |
|  | **<?import javafx.scene.image.ImageView?>** |
|  | **<?import javafx.scene.layout.Pane?>** |
|  | **<?import javafx.scene.text.Font?>** |
|  |  |
|  | **<Pane id="mainPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="458.0" prefWidth="780.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="bg.tu\_varna.sit.group17.controllers.PratkaRegisterController">** |
|  | **<children>** |
|  | **<ComboBox fx:id="category" layoutX="316.0" layoutY="197.0" prefHeight="41.0" prefWidth="217.0" promptText="категория" />** |
|  | **<CheckBox fx:id="fragile" layoutX="316.0" layoutY="243.0" mnemonicParsing="false" text="чупливо" />** |
|  | **<ComboBox fx:id="officeSender" layoutX="42.0" layoutY="235.0" prefHeight="41.0" prefWidth="217.0" promptText="офис на подаване" />** |
|  | **<ComboBox fx:id="officeReceiver" layoutX="316.0" layoutY="301.0" prefHeight="41.0" prefWidth="217.0" promptText="офис за получаване" />** |
|  | **<CheckBox fx:id="sendToAddress" layoutX="316.0" layoutY="271.0" mnemonicParsing="false" text="доставка до адрес" />** |
|  | **<ComboBox fx:id="companySender" layoutX="42.0" layoutY="197.0" prefHeight="41.0" prefWidth="217.0" promptText="фирма доставчик" />** |
|  | **<TextField fx:id="phoneSender" layoutX="43.0" layoutY="270.0" prefHeight="39.0" prefWidth="217.0" promptText="Телефон на Подател" />** |
|  | **<TextField fx:id="phoneReceiver" layoutX="43.0" layoutY="303.0" prefHeight="39.0" prefWidth="217.0" promptText="Телефон на Получател" />** |
|  | **<CheckBox fx:id="isPaid" layoutX="555.0" layoutY="243.0" mnemonicParsing="false" text="предплатена доставка" />** |
|  | **<DatePicker fx:id="receiveDate" layoutX="43.0" layoutY="339.0" prefHeight="41.0" prefWidth="217.0" promptText="приемане от подател" />** |
|  | **<DatePicker fx:id="clientReceiveDate" layoutX="555.0" layoutY="199.0" prefHeight="41.0" prefWidth="217.0" promptText="доставка на клиент" />** |
|  | **<Button layoutX="579.0" layoutY="404.0" mnemonicParsing="false" onAction="#registerOrder" prefHeight="39.0" prefWidth="193.0" text="регистрирай пратката" />** |
|  | **<TextField fx:id="address" layoutX="316.0" layoutY="359.0" prefHeight="39.0" prefWidth="217.0" promptText="адрес на доставка" />** |
|  | **<TextField fx:id="sendPrice" layoutX="555.0" layoutY="276.0" prefHeight="39.0" prefWidth="217.0" promptText="сума на доставка" />** |
|  | **<Pane id="navPane" prefHeight="110.0" prefWidth="780.0">** |
|  | **<children>** |
|  | **<Pane id="acPane" prefHeight="99.0" prefWidth="200.0">** |
|  | **<children>** |
|  | **<MenuButton id="userName" alignment="TOP\_LEFT" layoutX="1.0" layoutY="65.0" mnemonicParsing="false" prefHeight="40.0" prefWidth="199.0" text="user name" textFill="#333333">** |
|  | **<items>** |
|  | **<MenuItem mnemonicParsing="false" text="смени аватара" />** |
|  | **<MenuItem mnemonicParsing="false" text="излез" />** |
|  | **</items>** |
|  | **<font>** |
|  | **<Font size="18.0" />** |
|  | **</font>** |
|  | **</MenuButton>** |
|  | **<ImageView fitHeight="60.0" fitWidth="60.0" layoutX="14.0" layoutY="5.0" pickOnBounds="true" preserveRatio="true">** |
|  | **<image>** |
|  | **<Image url="@../img/defalut%20avatar.png" />** |
|  | **</image>** |
|  | **</ImageView>** |
|  | **</children>** |
|  | **</Pane>** |
|  | **<Button id="pratka" disable="true" layoutX="359.0" mnemonicParsing="false" onAction="#registerOrder" prefHeight="110.0" prefWidth="159.0" text="+пратка" />** |
|  | **<Button id="izvestia" layoutX="696.0" layoutY="14.0" minHeight="10.0" minWidth="53.0" mnemonicParsing="false" prefHeight="71.0" prefWidth="70.0" />** |
|  | **<Button id="firma" layoutX="518.0" mnemonicParsing="false" prefHeight="110.0" prefWidth="159.0" text="фирма" />** |
|  | **<Button id="spravki" layoutX="200.0" mnemonicParsing="false" onAction="#queries" prefHeight="110.0" prefWidth="159.0" text="справки" />** |
|  | **</children>** |
|  | **</Pane>** |
|  | **</children>** |
|  | **</Pane>** |

| **<?xml version="1.0" encoding="UTF-8"?>** |  |
| --- | --- |
|  |  |
|  | **<?import com.gluonhq.charm.glisten.control.ToggleButtonGroup?>** |
|  | **<?import javafx.scene.control.Button?>** |
|  | **<?import javafx.scene.control.ComboBox?>** |
|  | **<?import javafx.scene.control.MenuButton?>** |
|  | **<?import javafx.scene.control.MenuItem?>** |
|  | **<?import javafx.scene.control.PasswordField?>** |
|  | **<?import javafx.scene.control.TextField?>** |
|  | **<?import javafx.scene.image.Image?>** |
|  | **<?import javafx.scene.image.ImageView?>** |
|  | **<?import javafx.scene.layout.Pane?>** |
|  | **<?import javafx.scene.text.Font?>** |
|  |  |
|  | **<Pane id="mainPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="458.0" prefWidth="780.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="bg.tu\_varna.sit.group17.controllers.FirmaController">** |
|  | **<children>** |
|  | **<ComboBox fx:id="companies" layoutX="13.0" layoutY="115.0" prefHeight="36.0" prefWidth="218.0" promptText="Име на фирма" />** |
|  | **<ComboBox fx:id="offices" layoutX="282.0" layoutY="115.0" prefHeight="38.0" prefWidth="218.0" promptText="Град и адрес на офис" />** |
|  | **<ComboBox fx:id="city" layoutX="282.0" layoutY="151.0" prefHeight="41.0" prefWidth="218.0" promptText="Град" />** |
|  | **<TextField fx:id="address" layoutX="282.0" layoutY="192.0" prefHeight="39.0" prefWidth="218.0" promptText="Адрес" />** |
|  | **<TextField fx:id="courierTextField" layoutX="551.0" layoutY="151.0" prefHeight="38.0" prefWidth="218.0" promptText="Име на куриер" />** |
|  | **<TextField fx:id="phone" layoutX="551.0" layoutY="187.0" prefHeight="38.0" prefWidth="218.0" promptText="Телефон" />** |
|  | **<Button layoutX="14.0" layoutY="354.0" mnemonicParsing="false" onAction="#addFirma" prefHeight="39.0" prefWidth="92.0" text="+фирма" />** |
|  | **<Button layoutX="283.0" layoutY="355.0" mnemonicParsing="false" onAction="#addOffice" prefHeight="39.0" prefWidth="92.0" text="+офис" />** |
|  | **<Button layoutX="551.0" layoutY="355.0" mnemonicParsing="false" onAction="#addCourier" prefHeight="39.0" prefWidth="108.0" text="+служител" />** |
|  | **<PasswordField fx:id="password" layoutX="550.0" layoutY="224.0" prefHeight="39.0" prefWidth="218.0" promptText="парола" />** |
|  | **<ComboBox fx:id="couriers" layoutX="550.0" layoutY="115.0" prefHeight="38.0" prefWidth="218.0" promptText="Име на куриер" />** |
|  | **<Button layoutX="282.0" layoutY="403.0" mnemonicParsing="false" onAction="#changeOffice" prefHeight="39.0" prefWidth="218.0" text="редактирай офис" />** |
|  | **<Button layoutX="551.0" layoutY="403.0" mnemonicParsing="false" onAction="#changeCourier" prefHeight="36.0" prefWidth="218.0" text="редактирай служител" />** |
|  | **<TextField fx:id="firmaName" layoutX="14.0" layoutY="151.0" prefHeight="26.0" prefWidth="218.0" promptText="Име на фирма" />** |
|  | **<Button layoutX="13.0" layoutY="408.0" mnemonicParsing="false" onAction="#changeFirma" prefHeight="39.0" prefWidth="218.0" text="редактирай фирма" />** |
|  | **<Button layoutX="140.0" layoutY="354.0" mnemonicParsing="false" onAction="#deleteFirma" prefHeight="39.0" prefWidth="92.0" text="- фирма" />** |
|  | **<Button layoutX="408.0" layoutY="354.0" mnemonicParsing="false" onAction="#deleteOffice" prefHeight="39.0" prefWidth="92.0" text="-офис" />** |
|  | **<Button layoutX="674.0" layoutY="354.0" mnemonicParsing="false" onAction="#deleteCourier" prefHeight="39.0" prefWidth="92.0" text="-куриер" />** |
|  | **<TextField fx:id="plik" layoutX="14.0" layoutY="187.0" prefHeight="27.0" prefWidth="218.0" promptText="Цена за плик" />** |
|  | **<TextField fx:id="kolet" layoutX="14.0" layoutY="224.0" prefHeight="23.0" prefWidth="218.0" promptText="Цена за колет" />** |
|  | **<TextField fx:id="paket" layoutX="14.0" layoutY="261.0" prefHeight="23.0" prefWidth="218.0" promptText="Цена за голям пакет" />** |
|  | **<TextField fx:id="tovar" layoutX="14.0" layoutY="297.0" prefHeight="36.0" prefWidth="218.0" promptText="Цена за товар" />** |
|  | **<ToggleButtonGroup selectionType="SINGLE" />** |
|  | **<Pane id="navPane" prefHeight="110.0" prefWidth="780.0">** |
|  | **<children>** |
|  | **<Pane id="acPane" prefHeight="99.0" prefWidth="200.0">** |
|  | **<children>** |
|  | **<MenuButton id="userName" alignment="TOP\_LEFT" layoutX="1.0" layoutY="65.0" mnemonicParsing="false" prefHeight="40.0" prefWidth="199.0" text="user name" textFill="#333333">** |
|  | **<items>** |
|  | **<MenuItem mnemonicParsing="false" text="смени аватара" />** |
|  | **<MenuItem mnemonicParsing="false" text="излез" />** |
|  | **</items>** |
|  | **<font>** |
|  | **<Font size="18.0" />** |
|  | **</font>** |
|  | **</MenuButton>** |
|  | **<ImageView fitHeight="60.0" fitWidth="60.0" layoutX="14.0" layoutY="5.0" pickOnBounds="true" preserveRatio="true">** |
|  | **<image>** |
|  | **<Image url="@../img/defalut%20avatar.png" />** |
|  | **</image>** |
|  | **</ImageView>** |
|  | **</children>** |
|  | **</Pane>** |
|  | **<Button id="spravki" layoutX="206.0" mnemonicParsing="false" onAction="#queries" prefHeight="110.0" prefWidth="159.0" text="справки" />** |
|  | **<Button id="pratka" layoutX="359.0" mnemonicParsing="false" onAction="#registerOrder" prefHeight="110.0" prefWidth="159.0" text="+пратка" />** |
|  | **<Button id="izvestia" layoutX="696.0" layoutY="14.0" minHeight="10.0" minWidth="53.0" mnemonicParsing="false" prefHeight="71.0" prefWidth="70.0" />** |
|  | **<Button id="firma" disable="true" layoutX="518.0" mnemonicParsing="false" prefHeight="110.0" prefWidth="159.0" text="фирма" />** |
|  | **</children>** |
|  | **</Pane>** |
|  | **</children>** |
|  | **</Pane>** |

**css**

| **\*{** |  |
| --- | --- |
|  | **-fx-text-fill: white;** |
|  | **-fx-background-color: black;** |
|  | **}** |
|  | **TextField,PasswordField,ComboBox{** |
|  | **-fx-border-color: white;** |
|  | **-fx-border-radius: 3;** |
|  | **}** |
|  | **Button{** |
|  | **-fx-border-color: white;** |
|  | **-fx-border-radius: 3;** |
|  | **}** |
|  | **Hyperlink{** |
|  | **-fx-text-fill: #60cdf7;** |
|  | **}** |
|  | **#login{** |
|  | **-fx-graphic: url("../img/GandHlogin0.png");** |
|  | **}** |
|  | **#login:hover {** |
|  |  |
|  | **-fx-graphic: url("../img/GandHlogin1.png");** |
|  | **}** |
|  | **#register{** |
|  | **-fx-graphic: url("../img/register0.png");** |
|  | **}** |
|  | **#register:hover {** |
|  |  |
|  | **-fx-graphic: url("../img/register1.png");** |
|  | **}** |
|  | **#error{** |
|  | **-fx-text-fill: red;** |
|  | **}** |

| **Button{** |  |
| --- | --- |
|  | **-fx-text-fill: #B0B079;** |
|  | **-fx-background-color: black;** |
|  | **-fx-border-width: 2;** |
|  | **-fx-border-color: white;** |
|  | **}** |
|  | **Button:hover{** |
|  | **-fx-border-color: #B0B079;** |
|  | **}** |
|  | **#acPane{** |
|  | **-fx-text-fill: white;** |
|  | **-fx-background-color: black;** |
|  | **-fx-effect: null;** |
|  | **-fx-border-radius: 0;** |
|  | **-fx-border-width: 0 1 0 0;** |
|  | **-fx-border-color: transparent black transparent transparent;** |
|  | **}** |
|  | **#spravki,#pratka,#firma** |
|  | **{** |
|  | **-fx-text-fill: white;** |
|  | **-fx-background-color: black;** |
|  | **-fx-effect: null;** |
|  | **-fx-font-size: 24px;** |
|  | **-fx-font-style: Italic;** |
|  |  |
|  | **-fx-border-color: white;** |
|  | **-fx-border-width: 0 5 0 5;** |
|  | **-fx-border-color: transparent white transparent white;** |
|  | **}** |
|  | **#spravki:hover,#pratka:hover,#firma:hover{** |
|  | **-fx-text-fill: #B0B079;** |
|  | **-fx-border-color: transparent #B0B079 transparent #B0B079;** |
|  | **}** |
|  | **#navPane{** |
|  | **-fx-background-color: black;** |
|  | **}** |
|  | **#mainPane{** |
|  | **-fx-text-fill: black;** |
|  | **-fx-background-color: white;** |
|  | **}** |
|  | **#filterPane {** |
|  | **-fx-background-color: black;** |
|  | **-fx-text-fill: #B0B079;** |
|  | **-fx-border-width: 0 5 0 0;** |
|  | **-fx-border-color: transparent white transparent transparent;** |
|  |  |
|  | **}** |
|  | **#filterPane>\*{** |
|  | **-fx-text-fill: #B0B079;** |
|  | **-fx-background-color: black;** |
|  | **}** |
|  | **#userName, #userName .context-menu{** |
|  | **-fx-border: none;** |
|  | **-fx-text-fill: white;** |
|  | **-fx-background-color: black;** |
|  | **}** |
|  | **#userName .label{-fx-text-fill: white;}** |
|  |  |
|  | **.menu-item:focused {** |
|  | **-fx-background-color: #B0B079;** |
|  | **}** |
|  | **#izvestia{** |
|  | **-fx-background-color: black;** |
|  | **-fx-background-image: url("../img/izv.png");** |
|  | **-fx-background-position: center, center;** |
|  | **-fx-background-repeat: no-repeat;** |
|  | **-fx-background-size: 70% 90%;** |
|  | **-fx-border-radius: 50;** |
|  | **}** |
|  | **#vij{** |
|  | **-fx-background-color: black;** |
|  | **-fx-graphic: url("../img/filter.png");** |
|  | **}** |
|  | **#vij:hover{** |
|  | **-fx-graphic: url("../img/filter1.png");** |
|  | **}** |
|  | **.button,.textField,.comboBox** |
|  | **{** |
|  | **-fx-spacing: 100;** |
|  | **}** |
|  |  |
|  |  |
|  | **#spravki:disabled,#pratka:disabled,#firma:disabled{** |
|  |  |
|  | **-fx-opacity: 0.96;** |
|  | **-fx-text-fill: black;** |
|  | **-fx-background-color: white;** |
|  | **-fx-effect: null;** |
|  | **}** |
|  | **TextField,PasswordField,ComboBox,.date-picker{** |
|  | **-fx-prompt-text-fill: #B0B079;** |
|  | **-fx-border-width: 2;** |
|  | **-fx-border-color: #B0B079;** |
|  | **-fx-border-radius: 0;** |
|  | **}** |
|  | **.date-picker >\*{** |
|  | **-fx-prompt-text-fill: #B0B079;** |
|  | **}** |
|  | **.check-box{** |
|  | **-fx-prompt-text-fill: #B0B079;** |
|  | **-fx-focus-color: white;** |
|  | **}** |
|  | **ComboBox .cell{-fx-text-fill: #B0B079;}** |
|  |  |
|  | **.check-box:selected .mark {** |
|  | **-fx-background-color: #B0B079;** |
|  | **}** |

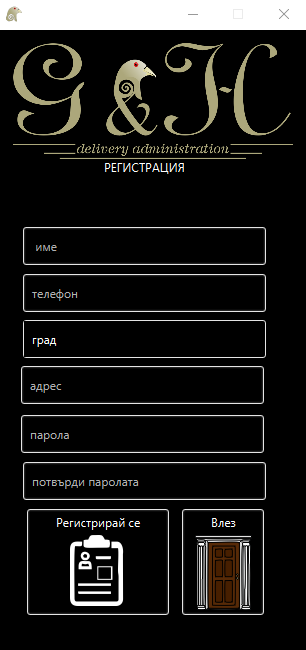
| **\*{** |  |
| --- | --- |
|  | **-fx-text-fill: white;** |
|  | **-fx-background-color: black;** |
|  | **}** |
|  | **TextField,PasswordField,ComboBox{** |
|  | **-fx-border-color: white;** |
|  | **-fx-border-radius: 3;** |
|  | **}** |
|  | **Button{** |
|  | **-fx-border-color: white;** |
|  | **-fx-border-radius: 3;** |
|  | **}** |
|  | **Hyperlink{** |
|  | **-fx-text-fill: #60cdf7;** |
|  | **}** |
|  | **#login{** |
|  | **-fx-graphic: url("../img/GandHlogin0.png");** |
|  | **}** |
|  | **#login:hover {** |
|  |  |
|  | **-fx-graphic: url("../img/GandHlogin1.png");** |
|  | **}** |
|  | **#register{** |
|  | **-fx-graphic: url("../img/register0.png");** |
|  | **}** |
|  | **#register:hover {** |
|  |  |
|  | **-fx-graphic: url("../img/register1.png");** |
|  | **}** |
|  | **#error{** |
|  | **-fx-text-fill: red;** |
|  | **}** |

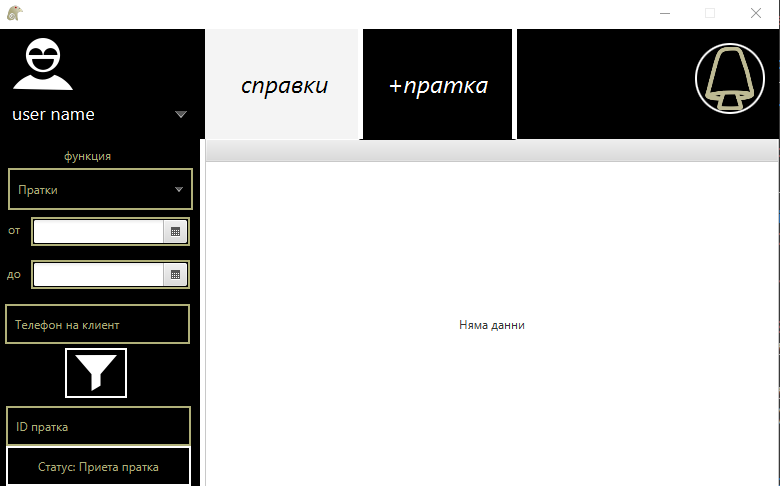
| **#pratka{** |  |
| --- | --- |
|  | **visibility: hidden;** |
|  | **}** |

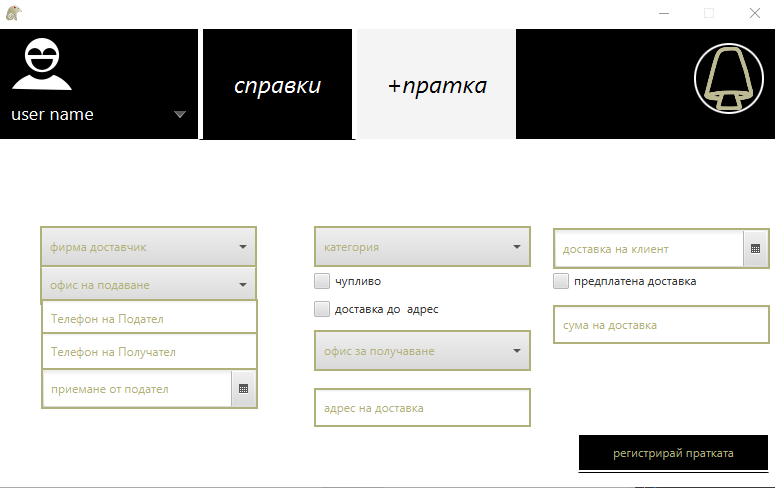
| **#firma{** |  |
| --- | --- |
|  | **visibility: hidden;** |
|  | **}** |

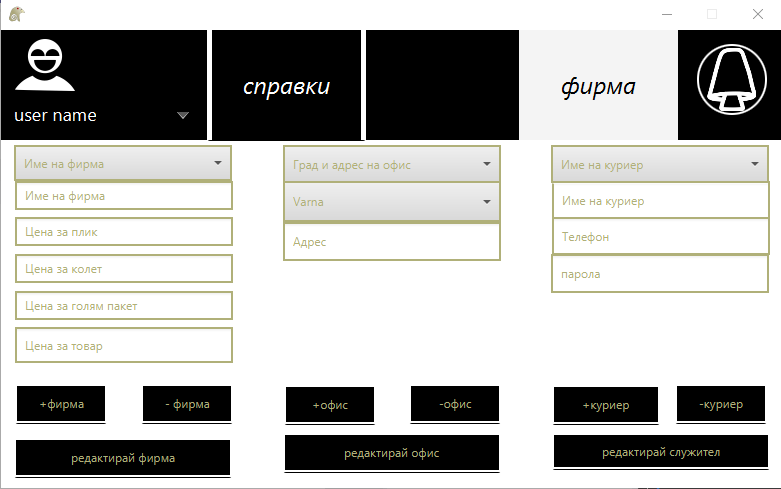
**скрийншоти**

****

****

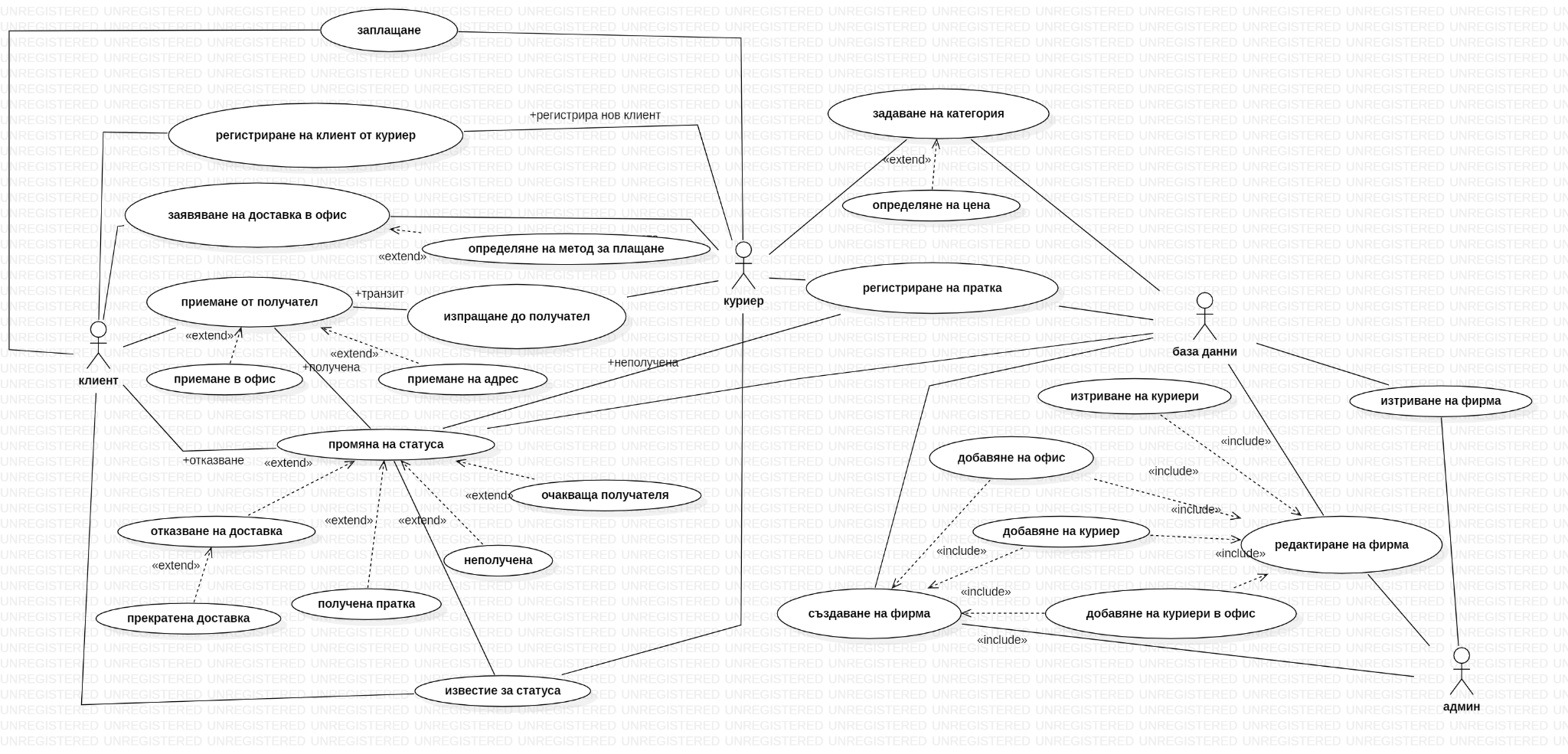
****

****

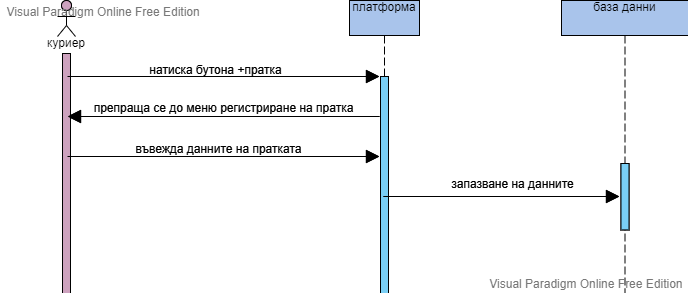
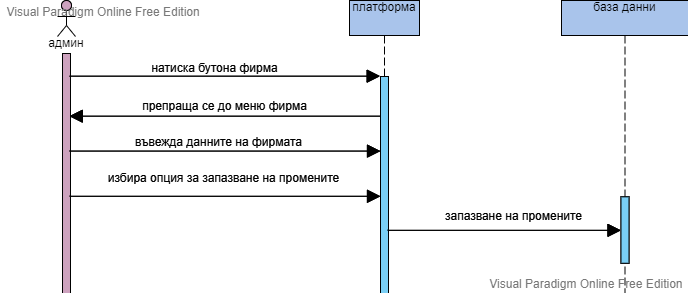
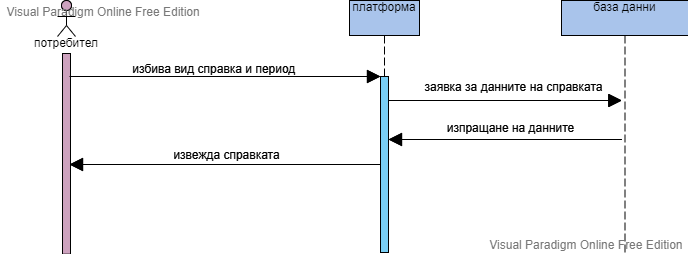
****

**диаграми**

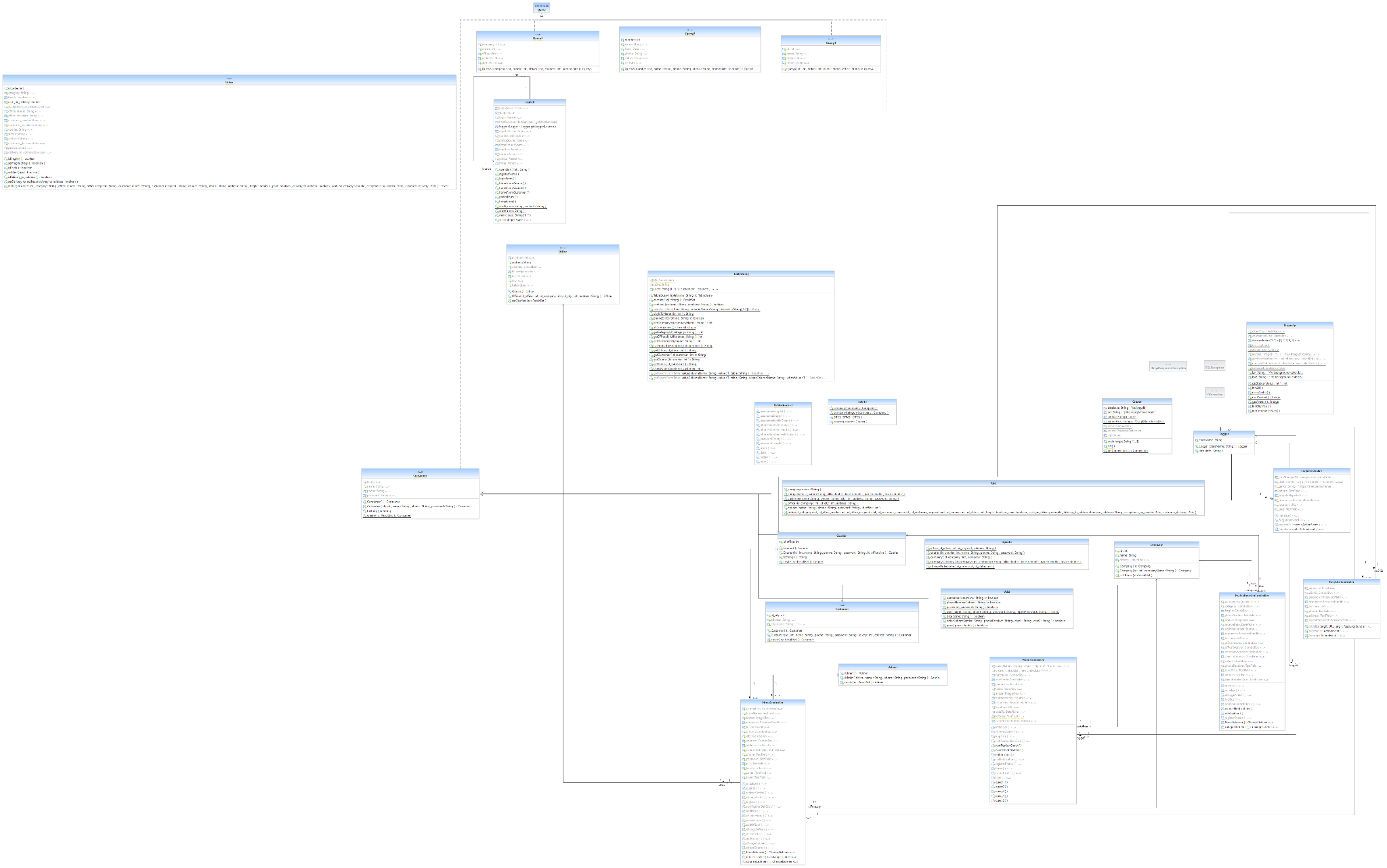
**use-case**

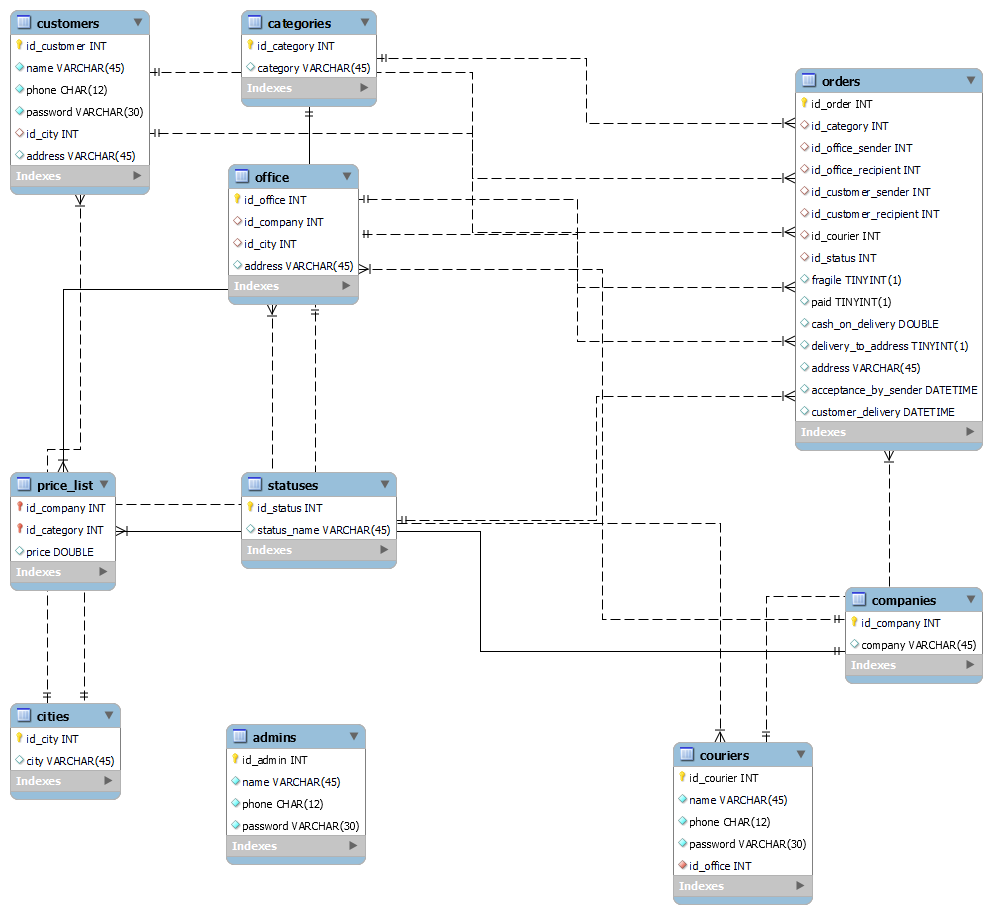
****

**Sequence Diagrams**

****

**Class Diagram**

****

**ER Diagram**

**Реализация на слоя за работа**

**Class Create**

**package bg.tu\_varna.sit.group17.database;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.sql.SQLException;**

**import java.util.Properties;**

**public final class Create {**

**private static final String database = "deliverydb";**

**private static final String url = "jdbc:mysql://localhost/" + database + "?useUnicode=true&characterEncoding=UTF-8";**

**private static final String admin = "root";**

**private static final String adminPass = "Mysql@localhost:3306";**

**private static Connection conn;**

**private static PreparedStatement create;**

**private static String sql;**

**public static void main(String[] args) {**

**try {**

**conn = getConnection();**

**DB();**

**} catch(Exception e) {**

**e.printStackTrace();**

**}**

**}**

**public static void DB() throws SQLException {**

**sql = "create database if not exists " + database;**

**create = conn.prepareStatement(sql);**

**create.executeUpdate(sql);**

**System.out.println("Database created successfully...");**

**}**

**public static Connection getConnection() throws SQLException {**

**try {**

**Properties info = new Properties();**

**info.put("user", admin);**

**info.put("password", adminPass);**

**info.put("userUnicode", "true");**

**Connection conn =**

**DriverManager.getConnection(url, info);**

**return conn;**

**} catch(SQLException e) {**

**System.out.println(e);**

**}**

**return null;**

**}**

**}**

**Class Add**

**package bg.tu\_varna.sit.group17.database;**

**import java.sql.Connection;**

**import java.sql.Date;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import bg.tu\_varna.sit.group17.application.Launch;**

**public final class Add {**

**public static void company(String name) throws SQLException {**

**Connection conn = Create.getConnection();**

**final String sql = "insert into companies(company) values(?)";**

**PreparedStatement st = conn.prepareStatement(sql);**

**st.setString(1, name);**

**st.execute();**

**st.close();**

**}**

**public static void companyPrice(String name, double plik, double kolet, double paket, double tovar) throws SQLException {**

**Connection conn = Create.getConnection();**

**String sql = "insert into companies(company) values(?)";**

**PreparedStatement ps = conn.prepareStatement(sql);**

**ps.setString(1, name);**

**ps.execute();**

**sql = "SELECT id\_company FROM companies ORDER BY id\_company DESC LIMIT 1";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) {**

**Launch.alert("Не се намира фирмата");**

**ps.close();**

**return;**

**}**

**double[] categories = {plik, kolet, paket, tovar};**

**int id\_company = rs.getInt("id\_company");**

**sql = "insert into price\_list(id\_company, id\_category, price) values(?, ?, ?)";**

**for(int i = 1; i < 5; ++i) {**

**ps = conn.prepareStatement(sql);**

**ps.setInt(1, id\_company);**

**ps.setInt(2, i);**

**ps.setDouble(3, categories[i-1]);**

**ps.execute();**

**}**

**ps.close();**

**}**

**public static void customer(String name, String phone, int city, String address, String password) throws SQLException {**

**Connection conn = Create.getConnection();**

**final String sql =**

**"insert into customers(customer, phone, "**

**+ "id\_city, address, password)"**

**+ " values (?, ?, ?, ?, ?)";**

**PreparedStatement st = conn.prepareStatement(sql);**

**st.setString(1, name);**

**st.setString(2, phone);**

**st.setInt(3, city);**

**st.setString(4, address);**

**st.setString(5, password);**

**st.execute();**

**st.close();**

**}**

**public static void office(int id\_company, int id\_city, String address) throws SQLException {**

**Connection conn = Create.getConnection();**

**final String sql = "insert into office(id\_company,"**

**+ " id\_city, address) values(?, ?, ?)";**

**PreparedStatement st = conn.prepareStatement(sql);**

**st.setInt(1, id\_company);**

**st.setInt(2, id\_city);**

**st.setString(3, address);**

**st.execute();**

**st.close();**

**}**

**public static void courier(String name, String phone, String password, int id\_office) throws SQLException {**

**Connection conn = Create.getConnection();**

**final String sql = "insert into couriers(name, phone,"**

**+ " password, id\_office) values(?, ?, ?, ?)";**

**PreparedStatement st = conn.prepareStatement(sql);**

**st.setString(1, name);**

**st.setString(2, phone);**

**st.setString(3, password);**

**st.setInt(4, id\_office);**

**st.execute();**

**st.close();**

**}**

**public static void order(int id\_category,**

**int id\_office\_sender, int id\_office\_recipient,**

**int id\_customer\_sender, int id\_customer\_recipient,**

**int id\_courier, int id\_status, boolean fragile,**

**boolean paid, double cash\_on\_delivery, boolean delivery\_to\_address, String address,**

**Date acceptance\_by\_sender, Date customer\_delivery) throws SQLException {**

**Connection conn = Create.getConnection();**

**final String sql = "insert into orders(id\_category,"**

**+ " id\_office\_sender, id\_office\_recipient,"**

**+ " id\_customer\_sender, id\_customer\_recipient,"**

**+ " id\_courier, id\_status, fragile,"**

**+ " paid, cash\_on\_delivery, delivery\_to\_address, address,"**

**+ " acceptance\_by\_sender, customer\_delivery) values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";**

**PreparedStatement st = conn.prepareStatement(sql);**

**st.setInt(1, id\_category);**

**st.setInt(2, id\_office\_sender);**

**st.setInt(3, id\_office\_recipient);**

**st.setInt(4, id\_customer\_sender);**

**st.setInt(5, id\_customer\_recipient);**

**st.setInt(6, id\_courier);**

**st.setInt(7, id\_status);**

**st.setBoolean(8, fragile);**

**st.setBoolean(9, paid);**

**st.setDouble(10, cash\_on\_delivery);**

**st.setBoolean(11, delivery\_to\_address);**

**st.setString(12, address);**

**st.setDate(13, acceptance\_by\_sender);**

**st.setDate(14, customer\_delivery);**

**st.execute();**

**st.close();**

**}**

**}**

**class Update**

**package bg.tu\_varna.sit.group17.database;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.SQLException;**

**public class Update {**

**public static void office(int id\_office, int id\_city, String address) throws SQLException {**

**Connection conn = Create.getConnection();**

**final String sql = "update office set id\_city=?, address=? where id\_office=?";**

**PreparedStatement st = conn.prepareStatement(sql);**

**st.setInt(1, id\_city);**

**st.setString(2, address);**

**st.setInt(3, id\_office);**

**st.executeUpdate();**

**st.close();**

**}**

**public static void courier(int id\_courier, String name, String phone, String password) throws SQLException {**

**Connection conn = Create.getConnection();**

**final String sql = "update couriers set name=?, phone=?, password=? where id\_courier=?";**

**PreparedStatement st = conn.prepareStatement(sql);**

**st.setString(1, name);**

**st.setString(2, phone);**

**st.setString(3, password);**

**st.setInt(4, id\_courier);**

**st.executeUpdate();**

**st.close();**

**}**

**public static void company(int id\_company, String company) throws SQLException {**

**Connection conn = Create.getConnection();**

**final String sql = "update companies set company=? where id\_company=?";**

**PreparedStatement ps = conn.prepareStatement(sql);**

**ps.setString(1, company);**

**ps.setInt(2, id\_company);**

**ps.executeUpdate();**

**ps.close();**

**}**

**public static void companyCategory(int id\_company, String company, double plik, double kolet, double paket, double tovar) throws SQLException {**

**Connection conn = Create.getConnection();**

**String sql = "update companies set company=? where id\_company=?";**

**PreparedStatement ps = conn.prepareStatement(sql);**

**ps.setString(1, company);**

**ps.setInt(2, id\_company);**

**ps.executeUpdate();**

**sql = "update price\_list set price=? where id\_company=? and id\_category=?";**

**double[] categories = {plik, kolet, paket, tovar};**

**for(int i = 1; i < 5; ++i) {**

**ps = conn.prepareStatement(sql);**

**ps.setDouble(1, categories[i-1]);**

**ps.setInt(2, id\_company);**

**ps.setInt(3, i);**

**ps.executeUpdate();**

**}**

**ps.close();**

**}**

**public static void changeOrderStatus(int id\_order, int id\_status) throws SQLException {**

**Connection conn = Create.getConnection();**

**String sql = "update orders set id\_status=? where id\_order=?";**

**PreparedStatement ps = conn.prepareStatement(sql);**

**ps.setInt(1, id\_status);**

**ps.setInt(2, id\_order);**

**ps.executeUpdate();**

**ps.close();**

**}**

**}**

**class TableQuery**

**package bg.tu\_varna.sit.group17.database;**

**import java.sql.Connection;**

**import java.sql.DatabaseMetaData;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.util.LinkedList;**

**import bg.tu\_varna.sit.group17.application.Property;**

**import bg.tu\_varna.sit.group17.database.property.Company;**

**import bg.tu\_varna.sit.group17.database.property.Office;**

**public class TableQuery {**

**private Connection DB;**

**private String table;**

**public static final String[] users =**

**{"customers", "couriers", "admins"};**

**public TableQuery(String tableName) {**

**try {**

**DB = Create.getConnection();**

**table = tableName;**

**DatabaseMetaData md = DB.getMetaData();**

**ResultSet rs = md.getTables(null, null, table, null);**

**if(!rs.next())**

**throw new IllegalArgumentException(**

**"Table " + tableName + " not exists");**

**} catch(Exception e) {**

**e.printStackTrace();**

**}**

**}**

**public static ResultSet execute(String sql) throws SQLException {**

**Connection DB = Create.getConnection();**

**PreparedStatement ps = DB.prepareStatement(sql);**

**ResultSet rs = ps.executeQuery();**

**if(!rs.next()) return null;**

**return rs;**

**}**

**public boolean contains(String column, String contains) {**

**try {**

**String sql = "select " + column + " from " + table +**

**" where " + column + " = '" + contains + "'";**

**PreparedStatement ps = DB.prepareStatement(sql);**

**ResultSet rs = ps.executeQuery();**

**return rs.next();**

**} catch(SQLException e) {**

**e.printStackTrace();**

**return false;**

**}**

**}**

**public static <T> ResultSet getRecordFromTable(**

**String valueColumnName, T value, String table) throws SQLException {**

**Connection DB = Create.getConnection();**

**PreparedStatement ps = null;**

**ResultSet rs = null;**

**String sql = "";**

**sql = "select \* from " + table +**

**" where " + valueColumnName + " = '" + value + "'";**

**ps = DB.prepareStatement(sql);**

**rs = ps.executeQuery();**

**if(!rs.next()) return null;**

**return rs;**

**}**

**public static <T> ResultSet getRecordFromTable(**

**String valueColumnName, T value, String table, String whereColumnName, T whereValue) throws SQLException {**

**Connection DB = Create.getConnection();**

**PreparedStatement ps = null;**

**ResultSet rs = null;**

**String sql = "";**

**sql = "select \* from " + table +**

**" where " + valueColumnName + " = '" + value + "' and " + whereColumnName + "='" + whereValue + "'";**

**ps = DB.prepareStatement(sql);**

**rs = ps.executeQuery();**

**if(!rs.next()) return null;**

**return rs;**

**}**

**public static String searchInTables(String find, String columnName, String... toSearch) {**

**TableQuery query;**

**for(int i = 0; i < toSearch.length; ++i) {**

**query = new TableQuery(toSearch[i]);**

**if(query.contains(columnName, find))**

**return toSearch[i];**

**}**

**return "";**

**}**

**public static String cityIdToName(int id) throws SQLException {**

**final String sql = "select city from cities where id\_city='" + id + "'";**

**ResultSet officeCity = TableQuery.execute(sql);**

**return officeCity.getString(1);**

**}**

**public static boolean phoneExists(String phone) throws SQLException {**

**ResultSet record = null;**

**for(int i = 0; i < TableQuery.users.length; ++i) {**

**record = TableQuery.getRecordFromTable(**

**"phone", phone, users[i]);**

**if(record != null) return true;**

**}**

**return false;**

**}**

**public static int getCompanyId(String companyName) throws SQLException {**

**String sql = "select \* from companies where company='" + companyName + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return 0;**

**return rs.getInt("id\_company");**

**}**

**public static LinkedList<Company> allCompanies() throws SQLException {**

**LinkedList<Company> companies = new LinkedList<>();**

**String sql = "select \* from companies";**

**ResultSet rs = TableQuery.execute(sql);**

**do companies.add(new Company(rs.getInt("id\_company"), rs.getString("company")));**

**while(rs.next());**

**for(int i = 0; i < companies.size(); ++i) {**

**sql = "select \* from office where id\_company='" + companies.get(i).getId() + "'";**

**rs = TableQuery.execute(sql);**

**if(rs == null) continue;**

**do companies.get(i).offices.add(**

**new Office(rs.getInt("id\_office"), rs.getInt("id\_company"),**

**rs.getInt("id\_city"), rs.getString("address")));**

**while(rs.next());**

**}**

**return companies;**

**}**

**public static int getCategoryId(String category) throws SQLException {**

**String sql = "select id\_category from categories where category='" + category + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return 0;**

**return rs.getInt("id\_category");**

**}**

**public static int getOfficeId(String fullAddress) throws SQLException {**

**String sql = "select \* from office";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return 0;**

**Company c = new Company();**

**c.setOffices(rs);**

**for(int i = 0; i < c.offices.size(); ++i) {**

**Office o = c.offices.get(i);**

**if(o.getFullAddress().equals(fullAddress))**

**return o.getId\_office();**

**}**

**return 0;**

**}**

**public static int getCustomerId(String phone) throws SQLException {**

**String sql = "select id\_customer from customers where phone='" + phone + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return 0;**

**return rs.getInt("id\_customer");**

**}**

**public static String categoryIdToString(int id\_category) throws SQLException {**

**String sql = "select category from categories where id\_category='"+id\_category + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return null;**

**return rs.getString("category");**

**}**

**public static String getOffice(int id\_office) throws SQLException {**

**String sql = "select \* from office where id\_office='" + id\_office + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return null;**

**return TableQuery.cityIdToName(rs.getInt("id\_city")) + " " + rs.getString("address");**

**}**

**public static String getCustomer(int id\_customer) throws SQLException {**

**String sql = "select name from customers where id\_customer='" + id\_customer + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return null;**

**return rs.getString("name");**

**}**

**public static String getCourier(int id\_courier) throws SQLException {**

**String sql = "select name from couriers where id\_courier='" + id\_courier + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return null;**

**return rs.getString("name");**

**}**

**public static String getStatus(int id\_status) throws SQLException {**

**String sql = "select status\_name from statuses where id\_status='" + id\_status +"'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return null;**

**return rs.getString("status\_name");**

**}**

**public static void checkOrderUpdate(int id\_customer) throws SQLException {**

**String sql = "select id\_order, customer\_delivery from orders where id\_customer\_recipient='" + id\_customer + "' and id\_status='" + Property.getStatus(0) + "' and customer\_delivery <= CURRENT\_DATE()";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return;**

**do {**

**Update.changeOrderStatus(rs.getInt("id\_order"), Property.getStatus(2));**

**}while(rs.next());**

**}**

**}**

**class Launch**

**package bg.tu\_varna.sit.group17.application;**

**import java.io.IOException;**

**import java.sql.SQLException;**

**import org.apache.log4j.Logger;**

**import org.apache.log4j.PropertyConfigurator;**

**import javafx.application.Application;**

**import javafx.application.HostServices;**

**import javafx.fxml.FXMLLoader;**

**import javafx.scene.Parent;**

**import javafx.scene.Scene;**

**import javafx.scene.control.Alert;**

**import javafx.scene.control.Alert.AlertType;**

**import javafx.scene.image.Image;**

**import javafx.stage.Stage;**

**public class Launch extends Application {**

**private Scene loginScene, registerScene, homeScene, pratkaScene, firmaScene;**

**private Stage stage;**

**private Parent login, register, home, pratka, firma;**

**private HostServices hostServices = getHostServices();**

**public static Launch launch;**

**static final Logger logger = Logger.getLogger(Launch.class.getName());**

**@Override**

**public void start(Stage stage) {**

**stage.setResizable(false);**

**PropertyConfigurator.configure(getClass().getResource("log4j.properties.txt"));**

**logger.info("Starting application");**

**this.stage = stage;**

**this.stage.getIcons().add(new Image(getClass().getResourceAsStream("../img/logoAvatar.png")));**

**Launch.launch = this;**

**try {**

**Property.initAll();**

**loginForm();**

**}catch(Exception e) {**

**e.printStackTrace();**

**}**

**}**

**public void openLink(String link) {**

**hostServices.showDocument(link);**

**}**

**public void registerForm() throws SQLException, IOException {**

**this.stage.hide();**

**this.register = FXMLLoader.load(getClass().getResource("../fxml/register.fxml"));**

**this.registerScene = new Scene(this.register);**

**this.registerScene.getStylesheets().add(getClass().getResource("../css/login-register.css").toExternalForm());**

**this.stage.setScene(this.registerScene);**

**this.stage.show();**

**}**

**public void loginForm() throws SQLException, IOException {**

**this.stage.hide();**

**this.login = FXMLLoader.load(getClass().getResource("../fxml/login.fxml"));**

**this.loginScene = new Scene(this.login);**

**this.loginScene.getStylesheets().add(getClass().getResource("../css/login-register.css").toExternalForm());**

**this.stage.setScene(this.loginScene);**

**this.stage.show();**

**}**

**public void homeFormAdmin() throws SQLException, IOException {**

**this.stage.hide();**

**this.home = FXMLLoader.load(getClass().getResource("../fxml/home.fxml"));**

**this.home.getStylesheets().add(getClass().getResource("../css/home.css").toExternalForm());**

**this.home.getStylesheets().add(getClass().getResource("../css/hide-pratka.css").toExternalForm());**

**this.homeScene = new Scene(this.home);**

**this.stage.setScene(this.homeScene);**

**this.stage.show();**

**}**

**public void homeFormCourier() throws SQLException, IOException {**

**this.stage.hide();**

**this.home = FXMLLoader.load(getClass().getResource("../fxml/home.fxml"));**

**this.home.getStylesheets().add(getClass().getResource("../css/home.css").toExternalForm());**

**this.home.getStylesheets().add(getClass().getResource("../css/hide-firma.css").toExternalForm());**

**this.homeScene = new Scene(this.home);**

**this.stage.setScene(this.homeScene);**

**this.stage.show();**

**}**

**public void homeFormCustomer() throws SQLException, IOException {**

**this.stage.hide();**

**this.home = FXMLLoader.load(getClass().getResource("../fxml/home.fxml"));**

**this.home.getStylesheets().add(getClass().getResource("../css/home.css").toExternalForm());**

**this.home.getStylesheets().add(getClass().getResource("../css/hide-pratka.css").toExternalForm());**

**this.home.getStylesheets().add(getClass().getResource("../css/hide-firma.css").toExternalForm());**

**this.homeScene = new Scene(this.home);**

**this.stage.setScene(this.homeScene);**

**this.stage.show();**

**}**

**public void pratkaForm() throws SQLException, IOException {**

**this.stage.hide();**

**this.pratka = FXMLLoader.load(getClass().getResource("../fxml/pratkaRegister.fxml"));**

**this.pratkaScene = new Scene(this.pratka);**

**this.pratka.getStylesheets().add(getClass().getResource("../css/home.css").toExternalForm());**

**this.pratka.getStylesheets().add(getClass().getResource("../css/pratkaRegister.css").toExternalForm());**

**this.stage.setScene(this.pratkaScene);**

**this.stage.show();**

**}**

**public void firmaForm() throws SQLException, IOException {**

**this.stage.hide();**

**this.firma = FXMLLoader.load(getClass().getResource("../fxml/firma.fxml"));**

**this.firmaScene = new Scene(this.firma);**

**this.firma.getStylesheets().add(getClass().getResource("../css/home.css").toExternalForm());**

**this.firma.getStylesheets().add(getClass().getResource("../css/hide-pratka.css").toExternalForm());**

**this.stage.setScene(this.firmaScene);**

**this.stage.show();**

**}**

**public static void alert(String error, String content) {**

**Alert alert = new Alert(AlertType.INFORMATION);**

**alert.setTitle("Съобщение");**

**alert.setHeaderText(error);**

**alert.setContentText(content);**

**alert.showAndWait();**

**}**

**public static void alert(String error) {**

**Alert alert = new Alert(AlertType.INFORMATION);**

**alert.setTitle("Съобщение");**

**alert.setHeaderText(error);**

**alert.showAndWait();**

**}**

**public static void main(String[] args) {**

**launch(args);**

**}**

**}**

**class Property**

**package bg.tu\_varna.sit.group17.application;**

**import java.sql.Connection;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.sql.Statement;**

**import java.util.HashMap;**

**import java.util.LinkedList;**

**import bg.tu\_varna.sit.group17.database.Create;**

**import bg.tu\_varna.sit.group17.database.TableQuery;**

**import javafx.scene.image.Image;**

**public final class Property {**

**public static HashMap<String, Integer> citiesMap;**

**public static HashMap<String, Integer> companiesMap;**

**private static final int[] statusesId = {1, 2, 3, 4, 5};**

**//1 неполучен 2 отказана 3 получена 4 взета 5 отказана за постоянно**

**public static byte user = 0;//1 customer 2 courier 3 admin**

**private static byte avatarIndex = 0;**

**public static Image[] avatars = {**

**new Image(Property.class.getResourceAsStream("../img/defalut avatar.png")),**

**new Image(Property.class.getResourceAsStream("../img/ang.png")),**

**new Image(Property.class.getResourceAsStream("../img/avatar.png"))**

**};**

**public static LinkedList<String> alertNotificationList = new LinkedList<>();**

**public static LinkedList<Integer> ordersIdNotification = new LinkedList<>();**

**public static boolean delivered = false;**

**public static final String izv = "-fx-background-color: black;"**

**+ "-fx-background-image: url("+Property.class.getResource("../img/izv.png").toExternalForm()+");"**

**+ " -fx-background-position: center, center;"**

**+ " -fx-background-repeat: no-repeat;"**

**+ " -fx-background-size: 70% 90%;"**

**+ " -fx-border-radius: 50;";**

**public static final String izv2 = "-fx-background-color: black;"**

**+ "-fx-background-image: url("+Property.class.getResource("../img/izv2.png").toExternalForm()+");"**

**+ " -fx-background-position: center, center;"**

**+ " -fx-background-repeat: no-repeat;"**

**+ " -fx-background-size: 70% 90%;"**

**+ " -fx-border-radius: 50;";**

**public static int getStatus(int status) throws IllegalArgumentException {**

**if(status < 0 || status > statusesId.length-1)**

**throw new IllegalArgumentException("Wrong status");**

**return statusesId[status];**

**}**

**public static void initAll() throws SQLException {**

**initCityMap();**

**initCompaniesMap();**

**}**

**public static void resetAvatar() {**

**avatarIndex = 0;**

**}**

**public static Image nextAvatar() {**

**if(avatarIndex >= avatars.length-1) avatarIndex = 0;**

**else avatarIndex++;**

**return avatars[avatarIndex];**

**}**

**public static Image getAvatar() {**

**return avatars[avatarIndex];**

**}**

**public static void initCityMap() throws SQLException {**

**ResultSet rs = TableQuery.execute("select \* from cities");**

**Property.citiesMap = new HashMap<String, Integer>();**

**if(rs == null) return;**

**do {**

**Property.citiesMap.put(rs.getString("city"),**

**Integer.parseInt(rs.getString("id\_city")));**

**}**

**while(rs.next());**

**}**

**public static void initCompaniesMap() throws SQLException {**

**Connection con = Create.getConnection();**

**Statement stmt = con.createStatement();**

**ResultSet rs = stmt.executeQuery("select \* from companies");**

**companiesMap = new HashMap<String, Integer>();**

**while(rs.next())**

**companiesMap.put(rs.getString(2),**

**Integer.parseInt(rs.getString(1)));**

**}**

**}**

**FirmaController**

**package bg.tu\_varna.sit.group17.controllers;**

**import java.io.IOException;**

**import java.net.URL;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.util.ResourceBundle;**

**import bg.tu\_varna.sit.group17.application.Launch;**

**import bg.tu\_varna.sit.group17.application.Logger;**

**import bg.tu\_varna.sit.group17.application.Property;**

**import bg.tu\_varna.sit.group17.database.Add;**

**import bg.tu\_varna.sit.group17.database.TableQuery;**

**import bg.tu\_varna.sit.group17.database.Update;**

**import bg.tu\_varna.sit.group17.database.property.Company;**

**import bg.tu\_varna.sit.group17.database.property.Office;**

**import bg.tu\_varna.sit.group17.database.users.Admin;**

**import bg.tu\_varna.sit.group17.database.users.Courier;**

**import bg.tu\_varna.sit.group17.database.users.Delete;**

**import bg.tu\_varna.sit.group17.validation.Valid;**

**import javafx.beans.value.ChangeListener;**

**import javafx.beans.value.ObservableValue;**

**import javafx.fxml.FXML;**

**import javafx.scene.control.ComboBox;**

**import javafx.scene.control.TextField;**

**import javafx.scene.image.ImageView;**

**public class FirmaController {**

**public Company company;**

**public Office office;**

**public Courier courier;**

**public static Admin admin;**

**private final Logger logger = new Logger(FirmaController.class.getName());**

**@FXML**

**public ComboBox<String> companies, offices, city, couriers;**

**@FXML**

**public TextField firmaName, address, courierTextField, phone, password,**

**plik, kolet, paket, tovar;**

**@FXML**

**public ImageView avatar;**

**@FXML**

**private ResourceBundle resources;**

**@FXML**

**private URL location;**

**@FXML**

**void initialize() throws SQLException {**

**logger.info("In firma form");**

**logger.info("Logged admin: " + admin);**

**Property.initAll();**

**this.avatar.setImage(Property.getAvatar());**

**if(companies.getItems().size() == 0) {**

**companies.getItems().addAll(Property.companiesMap.keySet());**

**companies.valueProperty().addListener(firmaListener());**

**}**

**if(this.city.getItems().size() == 0) {**

**this.city.getItems().addAll(Property.citiesMap.keySet());**

**this.city.getSelectionModel().select(0);**

**}**

**offices.valueProperty().addListener(officesListener());**

**couriers.valueProperty().addListener(couriersListener());**

**}**

**@FXML**

**void queries() throws SQLException, IOException {**

**Launch.launch.homeFormAdmin();**

**}**

**@FXML**

**void registerOrder() throws SQLException, IOException {**

**Launch.launch.pratkaForm();**

**}**

**@FXML**

**private void changeAvatar() {**

**this.avatar.setImage(Property.nextAvatar());**

**}**

**@FXML**

**private void logOut() throws SQLException, IOException {**

**Launch.launch.loginForm();**

**}**

**@FXML**

**private void notificationBellClick() {**

**Launch.alert("Нямате известия");**

**}**

**@FXML**

**private void addFirma() throws SQLException {**

**logger.info("Clicked add firma");**

**String name = firmaName.getText();**

**if(name.equals("")) {**

**Launch.alert("Полето име на фирма е празно");**

**return;**

**}**

**if(TableQuery.getCompanyId(name) > 0) {**

**Launch.alert("Не може да се добави фирмата. Вече съществува фирма с такова име");**

**return;**

**}**

**double plik = -1, kolet = -1, paket = -1, tovar = -1;**

**try {**

**plik = Double.parseDouble(this.plik.getText());**

**kolet = Double.parseDouble(this.kolet.getText());**

**paket = Double.parseDouble(this.paket.getText());**

**tovar = Double.parseDouble(this.tovar.getText());**

**if(!Valid.price(plik) || !Valid.price(kolet) ||**

**!Valid.price(paket) || !Valid.price(tovar))**

**throw new NullPointerException();**

**}catch(NullPointerException e) {**

**Launch.alert("Невалидна цена");**

**return;**

**}**

**Add.companyPrice(name, plik, kolet, paket, tovar);**

**Launch.alert("Успешно добавена фирма");**

**logger.info("Successful added company");**

**companies.getItems().clear();**

**Property.initCompaniesMap();**

**companies.getItems().addAll(Property.companiesMap.keySet());**

**}**

**@FXML**

**private void changeFirma() throws SQLException {**

**logger.info("Clicked change firma");**

**String name = firmaName.getText();//companies.getSelectionModel().getSelectedItem();**

**if(name.equals("") || name.length() < 3) {**

**Launch.alert("Невалидно име на фирма");**

**return;**

**}**

**int id\_company = TableQuery.getCompanyId(companies.getSelectionModel().getSelectedItem());**

**if(id\_company < 1){**

**Launch.alert("Фирмата не може да се промени, защото не съществува");**

**return;**

**}**

**double plik = -1, kolet = -1, paket = -1, tovar = -1;**

**try {**

**plik = Double.parseDouble(this.plik.getText());**

**kolet = Double.parseDouble(this.kolet.getText());**

**paket = Double.parseDouble(this.paket.getText());**

**tovar = Double.parseDouble(this.tovar.getText());**

**if(!Valid.price(plik) || !Valid.price(kolet) ||**

**!Valid.price(paket) || !Valid.price(tovar))**

**throw new NullPointerException();**

**}catch(NullPointerException e) {**

**Launch.alert("Невалидна цена");**

**return;**

**}**

**Update.companyCategory(id\_company, name, plik, kolet, paket, tovar);**

**Launch.alert("Фирмата е успешно редактирана");**

**logger.info("Successful changed company");**

**companies.getItems().clear();**

**Property.initCompaniesMap();**

**companies.getItems().addAll(Property.companiesMap.keySet());**

**companies.getSelectionModel().selectFirst();**

**}**

**@FXML**

**private void deleteFirma() throws SQLException {**

**logger.info("Clicked delete firma");**

**if(this.company == null) return;**

**Delete.companyCategory(this.company);**

**this.company = null;**

**Launch.alert("Фирмата е успешно изтрита");**

**logger.info("Successful deleted company");**

**if(couriers.getItems() == null) return;**

**couriers.getItems().clear();**

**if(offices.getItems() == null) return;**

**offices.getItems().clear();**

**if(companies.getItems() == null) return;**

**companies.getItems().clear();**

**Property.initCompaniesMap();**

**companies.getItems().addAll(Property.companiesMap.keySet());**

**}**

**@FXML**

**public void addOffice() throws SQLException {**

**logger.info("Clicked add office");**

**if(company == null) {**

**Launch.alert("Фирмата е празна");**

**return;**

**}**

**String cityName = this.city.getPromptText(),**

**address = this.address.getText();**

**if(cityName == null || address == null || cityName.isBlank() || address.isBlank()) {**

**Launch.alert("Не може града и/или адреса да са празни");**

**return;**

**}**

**int cityIndex = Property.citiesMap.get(cityName);**

**String sql = "select \* from office where id\_company='" +**

**this.company.getId() +**

**"' and id\_city='" + cityIndex + "'and address='" + address + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs != null) {**

**Launch.alert("Вече съществува такъв офис.");**

**return;**

**}**

**Add.office(this.company.getId(), cityIndex, address);**

**Launch.alert("Успешно добавен офис");**

**logger.info("Successful added office");**

**offices.getSelectionModel().selectFirst();**

**initialize();**

**}**

**@FXML**

**private void changeOffice() throws SQLException {**

**logger.info("Clicked change office");**

**if(this.office == null || this.offices.getPromptText().isBlank() ||**

**this.companies.getPromptText().isBlank()) {**

**Launch.alert("Фирмата и офиса не може да са празни");**

**return;**

**}**

**String cityName = this.city.getSelectionModel().getSelectedItem();**

**if(cityName.isBlank()) {**

**Launch.alert("Града не е избран");**

**return;**

**}**

**int cityIndex = Property.citiesMap.get(cityName);**

**String address = this.address.getText();**

**if(cityIndex == 0 || address.equals("")) {**

**Launch.alert("Не може града и/или адреса да са празни");**

**return;**

**}**

**String sql = "select \* from office where id\_company='" +**

**this.company.getId() +**

**"' and id\_office='" + this.office.getId\_office() + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) {**

**Launch.alert("Офиса не може да се промени. Не съществува такъв");**

**return;**

**}**

**Update.office(this.office.getId\_office(), cityIndex, address);**

**Launch.alert("Успешно редактиран офис");**

**logger.info("Successful changef office");**

**initialize();**

**}**

**@FXML**

**private void deleteOffice() throws SQLException {**

**logger.info("Clicked delete office");**

**if(this.office == null || this.offices.getPromptText().isBlank() ||**

**this.companies.getPromptText().isBlank()) {**

**Launch.alert("Фирмата и офиса не може да са празни");**

**return;**

**}**

**Delete.office(this.office);**

**this.office = null;**

**Launch.alert("Успешно изтрит офис");**

**logger.info("Successful deleted office");**

**if(couriers.getItems() == null) return;**

**couriers.getItems().clear();**

**if(offices.getItems() == null) return;**

**offices.getItems().clear();**

**initialize();**

**}**

**@FXML**

**private void addCourier() throws SQLException {**

**logger.info("Clicked add courier");**

**if(this.company == null || this.office == null || this.office.getId\_office() == 0) {**

**Launch.alert("Фирмата или офиса са празни");**

**return;**

**}**

**String name = this.courierTextField.getText(),**

**phone = this.phone.getText(),**

**password = this.password.getText();**

**if(name.equals("") || phone.equals("") || password.equals("")) {**

**Launch.alert("Полетата не може да са празни");**

**return;**

**}**

**String err = Valid.user(name, phone, password, password);**

**if(!err.equals("")) {**

**Launch.alert(err);**

**return;**

**}**

**String sql = "select \* from couriers where id\_office='" +**

**this.office.getId\_office() +**

**"' and name='" + name + "' and phone='" + phone +**

**"' and password='" + password + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs != null) {**

**Launch.alert("Вече съществува такъв куриер");**

**return;**

**}**

**Add.courier(name, phone, password, this.office.getId\_office());**

**rs = TableQuery.execute("select \* from couriers where phone='" + phone + "'");**

**this.courier = Courier.create(rs);**

**office.couriers.add(this.courier);**

**Launch.alert("Успешно добавен куриер");**

**logger.info("Successful added courier");**

**couriers.getSelectionModel().selectFirst();**

**rs = TableQuery.execute("select \* from couriers where phone='" + phone + "'");**

**if(rs != null)**

**this.courier = Courier.create(rs);**

**initialize();**

**}**

**@FXML**

**private void changeCourier() throws SQLException {**

**logger.info("Clicked change courier");**

**if(this.company == null || this.office == null ||**

**this.courier == null) {**

**Launch.alert("Фирмата офиса и куриера не може да са празни");**

**return;**

**}**

**String name = this.courierTextField.getText(),**

**phone = this.phone.getText(),**

**password = this.password.getText();**

**if(name.equals("") || phone.equals("") ||**

**password.equals("") || this.courier.getId() == 0) {**

**Launch.alert("Полетата не може да са празни");**

**return;**

**};**

**String err = Valid.user(name, phone, password, password);**

**if(!err.equals("") && !err.equals("Вече съществува такъв телефон в базата данни")) {**

**Launch.alert(err);**

**return;**

**}**

**Update.courier(this.courier.getId(), name, phone, password);**

**Launch.alert("Успешно редактиран куриер");**

**logger.info("Successful changed courier");**

**initialize();**

**}**

**@FXML**

**private void deleteCourier() throws SQLException {**

**logger.info("Clicked delete courier");**

**if(this.company == null || this.office == null || this.courier == null ||**

**this.offices.getPromptText().isBlank() ||**

**this.companies.getPromptText().isBlank() ||**

**this.couriers.getPromptText().isBlank()) {**

**Launch.alert("Фирмата офиса и куриера не може да са празни");**

**return;**

**}**

**Delete.courier(this.courier);**

**this.courier = null;**

**Launch.alert("Успешно изтрит куриер");**

**logger.info("Successful deleted courier");**

**if(couriers.getItems() == null) return;**

**couriers.getItems().clear();**

**initialize();**

**}**

**private ChangeListener<String> firmaListener() {**

**return new ChangeListener<String>() {**

**@Override**

**public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {**

**logger.info("Changed firma combobox");**

**offices.getItems().clear();**

**city.setPromptText(city.getSelectionModel().getSelectedItem().toString());**

**address.setText("");**

**couriers.getItems().clear();**

**courierTextField.setText("");**

**phone.setText("");**

**password.setText("");**

**company = new Company();**

**if(newValue == null) return;**

**company.setId(Property.companiesMap.get(newValue));**

**company.setName(newValue);**

**firmaName.setText(company.getName());**

**try {**

**String sql = "select \* from price\_list where id\_company='" + company.getId() + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) {**

**Launch.alert("firmata nqma ceni");**

**return;**

**}**

**do {**

**switch(rs.getInt("id\_category")) {**

**case 1: plik.setText(Double.toString(rs.getDouble("price")));**

**case 2: kolet.setText(Double.toString(rs.getDouble("price")));**

**case 3: paket.setText(Double.toString(rs.getDouble("price")));**

**case 4: tovar.setText(Double.toString(rs.getDouble("price")));**

**}**

**}while(rs.next());**

**sql = "select \* from office where id\_company='" + company.getId() + "'";**

**rs = TableQuery.execute(sql);**

**if(rs == null) {**

**offices.setPromptText("Няма офис");**

**couriers.setPromptText("Няма куриер");**

**Launch.alert("Фирмата още няма офиси");**

**return;**

**}**

**company.setOffices(rs);**

**if(company.offices.size() == 0) {**

**offices.setPromptText("Няма офис");**

**couriers.setPromptText("Няма куриер");**

**return;**

**}**

**for(int i = 0; i < company.offices.size(); ++i) {**

**Office office = company.offices.get(i);**

**offices.getItems().add(**

**TableQuery.cityIdToName(office.getId\_city()) +**

**" " + office.getAddress());**

**}**

**office = company.offices.get(0);**

**offices.getSelectionModel().selectFirst();**

**city.setPromptText(office.getCity());**

**address.setText(office.getAddress());**

**} catch(SQLException e) {**

**e.printStackTrace();**

**Launch.alert("Не се намира офиса");**

**}**

**}**

**};**

**}**

**private ChangeListener<String> officesListener() {**

**return new ChangeListener<String>() {**

**@Override**

**public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {**

**logger.info("Changed office combobox");**

**couriers.getItems().clear();**

**courierTextField.setText("");**

**phone.setText("");**

**password.setText("");**

**office = new Office();**

**try {**

**if(company != null)**

**for(int i = 0; i < company.offices.size(); ++i) {**

**String officeName =**

**TableQuery.cityIdToName(company.offices.get(i).getId\_city()) +**

**" " + company.offices.get(i).getAddress();**

**if(officeName.equals(newValue)) {**

**office = company.offices.get(i);**

**break;**

**}**

**}**

**if(office.getId\_office() == 0) {**

**offices.setPromptText("Няма офис");**

**return;**

**}**

**city.setPromptText(TableQuery.cityIdToName(office.getId\_city()));**

**address.setText(office.getAddress());**

**String sql = "select \* from couriers where id\_office='" + office.getId\_office() + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) {**

**couriers.setPromptText("Няма куриери");**

**return;**

**}**

**office.setCouriers(rs);**

**if(office.couriers.size() == 0) {**

**couriers.setPromptText("Няма куриери");**

**return;**

**}**

**for(int i = 0; i < office.couriers.size(); ++i) {**

**Courier courier = office.couriers.get(i);**

**couriers.getItems().add(courier.getName());**

**}**

**courier = office.couriers.get(0);**

**couriers.getSelectionModel().selectFirst();**

**} catch(SQLException e) {**

**e.printStackTrace();**

**Launch.alert("Не са намират офисите");**

**}**

**}**

**};**

**}**

**private ChangeListener<String> couriersListener() {**

**return new ChangeListener<String>() {**

**@Override**

**public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {**

**logger.info("Changed courier combobox");**

**courierTextField.setText("");**

**phone.setText("");**

**password.setText("");**

**int selectedIndex = couriers.getSelectionModel().getSelectedIndex();**

**if(selectedIndex < 0 || selectedIndex > office.couriers.size()-1) return;**

**courier = office.couriers.get(selectedIndex);**

**courierTextField.setText(courier.getName());**

**phone.setText(courier.getPhone());**

**password.setText(courier.getPassword());**

**}**

**};**

**}**

**}**

**class HomeController**

**package bg.tu\_varna.sit.group17.controllers;**

**import java.io.IOException;**

**import java.net.URL;**

**import java.sql.Date;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.time.LocalDate;**

**import java.util.Arrays;**

**import java.util.LinkedList;**

**import java.util.List;**

**import java.util.ResourceBundle;**

**import bg.tu\_varna.sit.group17.application.Launch;**

**import bg.tu\_varna.sit.group17.application.Logger;**

**import bg.tu\_varna.sit.group17.application.Property;**

**import bg.tu\_varna.sit.group17.database.TableQuery;**

**import bg.tu\_varna.sit.group17.database.Update;**

**import bg.tu\_varna.sit.group17.database.property.Order;**

**import bg.tu\_varna.sit.group17.database.queries.Query;**

**import bg.tu\_varna.sit.group17.database.queries.Query2;**

**import bg.tu\_varna.sit.group17.database.queries.Query3;**

**import bg.tu\_varna.sit.group17.database.queries.Query4;**

**import bg.tu\_varna.sit.group17.database.users.Customer;**

**import bg.tu\_varna.sit.group17.validation.Valid;**

**import javafx.collections.FXCollections;**

**import javafx.collections.ObservableList;**

**import javafx.fxml.FXML;**

**import javafx.scene.control.Button;**

**import javafx.scene.control.ComboBox;**

**import javafx.scene.control.DatePicker;**

**import javafx.scene.control.Label;**

**import javafx.scene.control.TableColumn;**

**import javafx.scene.control.TableView;**

**import javafx.scene.control.TextField;**

**import javafx.scene.control.cell.PropertyValueFactory;**

**import javafx.scene.image.ImageView;**

**public final class HomeController {**

**private String[] quеryNames = {**

**"Пратки", "Статус на пратка", "Статистика на фирма",**

**"Куриери", "Клиенти"**

**};**

**public static Customer customer;**

**private final Logger logger = new Logger(HomeController.class.getName());**

**private LinkedList<Query2> orders = new LinkedList<>();**

**@FXML**

**private ComboBox<String> functions;**

**@FXML**

**private DatePicker dateFrom, dateTo;**

**@FXML**

**private TextField phone, IdOrder;**

**@FXML**

**private TableView<Query> table;**

**@FXML**

**public ImageView avatar;**

**@FXML**

**private Button notificationBell, cancelOrderButton;**

**@FXML**

**private ResourceBundle resources;**

**@FXML**

**private URL location;**

**@FXML**

**void initialize() throws SQLException {**

**logger.info("In home form");**

**Property.alertNotificationList = new LinkedList<>();**

**this.avatar.setImage(Property.getAvatar());**

**if(Property.user == 1) {**

**logger.info("Logged customer: " + customer);**

**logger.info("Checking for updating order status");**

**TableQuery.checkOrderUpdate(HomeController.customer.getId());**

**this.phone.setText(HomeController.customer.getPhone());**

**this.phone.setDisable(true);**

**this.quеryNames = new String[] {"Статус на пратка"};**

**this.functions.setDisable(true);**

**}**

**else if(Property.user == 2){**

**cancelOrderButton.setText("Статус: Приета пратка");**

**} else if(Property.user == 3) {**

**this.phone.setDisable(true);**

**this.IdOrder.setDisable(true);**

**this.cancelOrderButton.setDisable(true);**

**}**

**if(Property.user !=3 ) {**

**Property.initAll();**

**notificationCheck();**

**}**

**functions.getItems().addAll(Arrays.asList(quеryNames));**

**functions.getSelectionModel().selectFirst();;**

**this.table.setEditable(true);**

**this.table.setPlaceholder(new Label("Няма данни"));**

**}**

**@FXML**

**private void changeAvatar() {**

**this.avatar.setImage(Property.nextAvatar());**

**}**

**@FXML**

**private void logOut() throws SQLException, IOException {**

**Launch.launch.loginForm();**

**}**

**@FXML**

**private void notificationBellClick() throws IllegalArgumentException, SQLException {**

**if(Property.delivered) {**

**if(Property.user == 1) {**

**this.notificationBell.setStyle(Property.izv);**

**Property.delivered = false;**

**String offices = "";**

**for(String s : Property.alertNotificationList) offices += s + " ";**

**Launch.alert("Имате пратка в офис " + offices);**

**return;**

**} if(Property.user == 2) {**

**this.notificationBell.setStyle(Property.izv);**

**Property.delivered = false;**

**String customers = "";**

**for(String s : Property.alertNotificationList) customers += s + " ";**

**Launch.alert("Отказани пратки от клиенти: " + customers);**

**for(int i : Property.ordersIdNotification)**

**Update.changeOrderStatus(i, Property.getStatus(4));**

**return;**

**}**

**}**

**Launch.alert("Нямате известия");**

**}**

**private void notificationCheck() throws SQLException {**

**if(Property.user == 2) {**

**courierNotification();**

**return;**

**}**

**String sql = "select id\_office\_recipient from orders where id\_customer\_recipient='" + HomeController.customer.getId() + "' and id\_status='" + Property.getStatus(2) + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return;**

**Property.delivered = true;**

**do Property.alertNotificationList.add(TableQuery.getOffice(rs.getInt("id\_office\_recipient")));**

**while(rs.next());**

**notification();**

**}**

**private void courierNotification() throws SQLException {**

**String sql = "select id\_order, id\_customer\_recipient from orders where id\_courier='" + PratkaRegisterController.courier.getId() + "' and id\_status='" + Property.getStatus(1) + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return;**

**Property.delivered = true;**

**do {**

**Property.alertNotificationList.add(TableQuery.getCustomer(rs.getInt("id\_customer\_recipient")));**

**Property.ordersIdNotification.add(rs.getInt("id\_order"));**

**}while(rs.next());**

**notification();**

**}**

**private void notification() {**

**this.notificationBell.setStyle(Property.izv2);**

**}**

**@SuppressWarnings("unused")**

**private void noNotification() {**

**this.notificationBell.setStyle(Property.izv);**

**}**

**@FXML**

**private void registerPratka() throws SQLException, IOException {**

**Launch.launch.pratkaForm();**

**}**

**@FXML**

**private void firma() throws SQLException, IOException {**

**Launch.launch.firmaForm();**

**}**

**@FXML**

**private void cancelOrder() throws SQLException {**

**logger.info("Clicked cancel order");**

**if(Property.user != 1) {**

**int id\_order = 0;**

**try {**

**id\_order = Integer.parseInt(this.IdOrder.getText());**

**}catch(NullPointerException e) {**

**Launch.alert("Невалидно ID");**

**return;**

**}**

**if(id\_order == 0) {**

**Launch.alert("Невалидно ID");**

**return;**

**}**

**String phone = this.phone.getText();**

**TableQuery customerQuery = new TableQuery("customers");**

**if(!Valid.phoneNumber(phone) || !customerQuery.contains("phone", phone)) {**

**Launch.alert("Невалиден телефон на клиент");**

**return;**

**}**

**int id\_customer = TableQuery.getCustomerId(phone);**

**String sql = "select \* from orders where id\_customer\_recipient='" + id\_customer + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) {**

**Launch.alert("Клиента няма поръчки");**

**return;**

**}**

**int id\_status = rs.getInt("id\_status");**

**if(id\_status == Property.getStatus(1)) {**

**Launch.alert("Пратката вече е отказана");**

**return;**

**}**

**if(id\_status == Property.getStatus(0)) {**

**Launch.alert("Пратката пратката още не е получена");**

**return;**

**}**

**sql = "select \* from orders where id\_order='" + id\_order + "' and id\_customer\_recipient='" + id\_customer + "'";**

**rs = TableQuery.execute(sql);**

**if(rs == null) {**

**Launch.alert("Клиента няма такава пратка с това ID");**

**return;**

**}**

**Update.changeOrderStatus(id\_order, Property.getStatus(3));**

**Launch.alert("Променен статус на пратката: взета от куриер");**

**logger.info("Order accepted from courier");**

**filter();**

**return;**

**}**

**if(this.orders.size() == 0) {**

**Launch.alert("Няма пратки");**

**return;**

**}**

**int id\_order = 0;**

**try {**

**id\_order = Integer.parseInt(this.IdOrder.getText());**

**}catch(NullPointerException e) {**

**Launch.alert("Невалидно ID");**

**return;**

**}**

**if(id\_order == 0) {**

**Launch.alert("Невалидно ID");**

**return;**

**}**

**String sql = "select id\_status from orders where id\_order='" + id\_order + "' and id\_customer\_recipient='" + customer.getId() + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) {**

**Launch.alert("Невалидно ID");**

**return;**

**}**

**int id\_status = rs.getInt("id\_status");**

**if(id\_status == Property.getStatus(1)) {**

**Launch.alert("Пратката вече е отказана");**

**return;**

**}**

**if(id\_status == Property.getStatus(2)) {**

**Launch.alert("Пратката не може да бъде отказана, вече е получена");**

**return;**

**}**

**Update.changeOrderStatus(id\_order, Property.getStatus(1));**

**Launch.alert("Пратката е отказана");**

**logger.info("Order cancelled");**

**filter();**

**}**

**@FXML**

**private void filter() throws SQLException {**

**logger.info("Clicked filter");**

**this.table.getColumns().clear();**

**if(Property.user == 1) {**

**query2();**

**return;**

**}**

**int queryIndex = functions.getSelectionModel().getSelectedIndex();**

**logger.info("Query index: " + queryIndex);**

**switch(queryIndex) {**

**case 0: query1(); break;**

**case 1: query2(); break;**

**case 2: query3(); break;**

**case 3: query4(); break;**

**case 4: query5(); break;**

**}**

**}**

**private void query1() throws SQLException {**

**logger.info("In query1");**

**LocalDate from = this.dateFrom.getValue(),**

**to = this.dateTo.getValue();**

**if(from == null || to == null) {**

**Launch.alert("Полетата за дати не може да са празни");**

**return;**

**}**

**if(Date.valueOf(from).after(Date.valueOf(to))) {**

**Launch.alert("Началната дата не може да е преди крайната дата");**

**return;**

**}**

**TableColumn<Query, String> idColumn = new TableColumn<Query, String>("ID Поръчка"),**

**category = new TableColumn<Query, String>("Категория"),**

**officeSender = new TableColumn<Query, String>("Изпращащ офис"),**

**officeRecipient = new TableColumn<Query, String>("Получаващ офис"),**

**customerSender = new TableColumn<Query, String>("Клиент изпращач"),**

**customerRecipient = new TableColumn<Query, String>("Клиент получател"),**

**courierC = new TableColumn<Query, String>("Обслужващ куриер"),**

**statusC = new TableColumn<Query, String>("Статус"),**

**fragileC = new TableColumn<Query, String>("Чупливо"),**

**paidC = new TableColumn<Query, String>("Наложен платеж"),**

**cashDelivery = new TableColumn<Query, String>("Цена на доставка"),**

**deliveryToAddress = new TableColumn<Query, String>("Досавяне до адрес"),**

**addressC = new TableColumn<Query, String>("Адрес на доставяне"),**

**acceptanceSender = new TableColumn<Query, String>("Дата на изпращане"),**

**customerDelivery = new TableColumn<Query, String>("Дата на получаване");**

**table.getColumns().addAll(List.of(idColumn, category, officeSender, officeRecipient,**

**customerSender, customerRecipient, courierC, statusC, fragileC, paidC,**

**cashDelivery, deliveryToAddress, addressC, acceptanceSender, customerDelivery));**

**final ObservableList<Query> data = FXCollections.observableArrayList();**

**String sql = "select \* from orders";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return;**

**do {**

**int id\_order = rs.getInt("id\_order");**

**String categoryName = TableQuery.categoryIdToString(rs.getInt("id\_category"));**

**String office\_sender = TableQuery.getOffice(rs.getInt("id\_office\_sender")),**

**office\_recipient = TableQuery.getOffice(rs.getInt("id\_office\_recipient"));**

**String customer\_sender = TableQuery.getCustomer(rs.getInt("id\_customer\_sender")),**

**customer\_recipient = TableQuery.getCustomer(rs.getInt("id\_customer\_recipient")),**

**courier = TableQuery.getCourier(rs.getInt("id\_courier")),**

**status = TableQuery.getStatus(rs.getInt("id\_status"));**

**boolean fragile = rs.getBoolean("fragile"),**

**paid = rs.getBoolean("paid"),**

**delivery\_to\_address = rs.getBoolean("delivery\_to\_address");**

**String address = rs.getString("address");**

**Date acceptance\_by\_sender = rs.getDate("acceptance\_by\_sender"),**

**customer\_delivery = rs.getDate("customer\_delivery");**

**if(acceptance\_by\_sender.before(Date.valueOf(dateFrom.getValue())) ||**

**customer\_delivery.after(Date.valueOf(dateTo.getValue())))**

**continue;**

**double payPrice = rs.getDouble("cash\_on\_delivery");**

**data.add(new Order(id\_order, categoryName, office\_sender, office\_recipient,**

**customer\_sender, customer\_recipient, courier, status, address,**

**fragile, paid, delivery\_to\_address,payPrice, acceptance\_by\_sender, customer\_delivery));**

**} while(rs.next());**

**idColumn.setCellValueFactory(new PropertyValueFactory<>("id\_order"));**

**category.setCellValueFactory(new PropertyValueFactory<>("category"));**

**officeSender.setCellValueFactory(new PropertyValueFactory<>("office\_sender"));**

**officeRecipient.setCellValueFactory(new PropertyValueFactory<>("office\_recipient"));**

**customerSender.setCellValueFactory(new PropertyValueFactory<>("customer\_sender"));**

**customerRecipient.setCellValueFactory(new PropertyValueFactory<>("customer\_recipient"));**

**courierC.setCellValueFactory(new PropertyValueFactory<>("courier"));**

**statusC.setCellValueFactory(new PropertyValueFactory<>("status"));**

**fragileC.setCellValueFactory(new PropertyValueFactory<>("fragile"));**

**paidC.setCellValueFactory(new PropertyValueFactory<>("paid"));**

**cashDelivery.setCellValueFactory(new PropertyValueFactory<>("cash\_on\_delivery"));**

**deliveryToAddress.setCellValueFactory(new PropertyValueFactory<>("delivery\_to\_address"));**

**addressC.setCellValueFactory(new PropertyValueFactory<>("address"));**

**acceptanceSender.setCellValueFactory(new PropertyValueFactory<>("acceptance\_by\_sender"));**

**customerDelivery.setCellValueFactory(new PropertyValueFactory<>("customer\_delivery"));**

**this.table.setItems(data);**

**}**

**private void query2() throws SQLException {**

**logger.info("In query2");**

**this.orders.clear();**

**String phone = this.phone.getText();**

**if(!Valid.phoneNumber(phone)) {**

**Launch.alert("Въведете правилен телефонен номер на клиент", "Телефона трябва да съдържа 12 цифри");**

**return;**

**}**

**LocalDate from = this.dateFrom.getValue(),**

**to = this.dateTo.getValue();**

**if(from == null || to == null) {**

**Launch.alert("Полетата за дати не може да са празни");**

**return;**

**}**

**if(Date.valueOf(from).after(Date.valueOf(to))) {**

**Launch.alert("Началната дата не може да е преди крайната дата");**

**return;**

**}**

**TableColumn<Query, String> numOrder = new TableColumn<Query, String>("№ пратка"),**

**nameC = new TableColumn<Query, String>("Име на клиент"),**

**phoneC = new TableColumn<Query, String>("Телефон"),**

**statusC = new TableColumn<Query, String>("Статус"),**

**fromC = new TableColumn<Query, String>("Дата на приемане"),**

**toC = new TableColumn<Query, String>("Дата на пристигане");**

**this.table.getColumns().addAll(List.of(numOrder, nameC, phoneC, statusC, fromC, toC));**

**final ObservableList<Query> data = FXCollections.observableArrayList();**

**String sql = "select id\_customer, name, phone from customers where phone='" + phone + "'";**

**ResultSet customerSet = TableQuery.execute(sql);**

**if(customerSet == null) {**

**Launch.alert("Няма клиенти");**

**return;**

**}**

**do {**

**int customerId = TableQuery.getCustomerId(this.phone.getText());**

**sql = "select id\_order, id\_status, acceptance\_by\_sender, customer\_delivery from orders where " + /\*"id\_customer\_sender='" + customerId + "' or " + \*/"id\_customer\_recipient='" + customerId + "'";**

**ResultSet orderSet = TableQuery.execute(sql);**

**if(orderSet == null) {**

**Launch.alert("Клиента с този номер няма поръчки");**

**return;**

**}**

**do {**

**String status = TableQuery.getStatus(orderSet.getInt("id\_status"));**

**Query2 query = new Query2(orderSet.getInt("id\_order"), customerSet.getString("name"), customerSet.getString("phone"), status, orderSet.getDate("acceptance\_by\_sender"), orderSet.getDate("customer\_delivery"));**

**data.add(query);**

**this.orders.add(query);**

**}while(orderSet.next());**

**}while(customerSet.next());**

**numOrder.setCellValueFactory(new PropertyValueFactory<>("number"));**

**nameC.setCellValueFactory(new PropertyValueFactory<>("name"));**

**phoneC.setCellValueFactory(new PropertyValueFactory<>("phone"));**

**statusC.setCellValueFactory(new PropertyValueFactory<>("status"));**

**fromC.setCellValueFactory(new PropertyValueFactory<>("from"));**

**toC.setCellValueFactory(new PropertyValueFactory<>("to"));**

**this.table.setItems(data);**

**}**

**private void query3() throws SQLException {**

**logger.info("In query3");**

**TableColumn<Query, String>**

**companiesC = new TableColumn<>("Брой фирми"),**

**ordersC = new TableColumn<>("Брой поръчки"),**

**officesC = new TableColumn<>("Брой офиси"),**

**couriersC = new TableColumn<>("Брой куриери"),**

**adminsC = new TableColumn<>("Брой администратори");**

**table.getColumns().addAll(List.of(companiesC, ordersC, officesC, couriersC, adminsC));**

**final ObservableList<Query> data = FXCollections.observableArrayList();**

**String sql = "select count(\*) from companies";**

**ResultSet rs = TableQuery.execute(sql);**

**int companies = rs.getInt(1);**

**sql = "select count(\*) from orders";**

**rs = TableQuery.execute(sql);**

**int orders = rs.getInt(1);**

**sql = "select count(\*) from office";**

**rs = TableQuery.execute(sql);**

**int offices = rs.getInt(1);**

**sql = "select count(\*) from couriers";**

**rs = TableQuery.execute(sql);**

**int couriers = rs.getInt(1);**

**sql = "select count(\*) from admins";**

**rs = TableQuery.execute(sql);**

**int admins = rs.getInt(1);**

**data.add(new Query3(companies, orders, offices, couriers, admins));**

**companiesC.setCellValueFactory(new PropertyValueFactory<>("company"));**

**ordersC.setCellValueFactory(new PropertyValueFactory<>("orders"));**

**officesC.setCellValueFactory(new PropertyValueFactory<>("offices"));**

**couriersC.setCellValueFactory(new PropertyValueFactory<>("couriers"));**

**adminsC.setCellValueFactory(new PropertyValueFactory<>("admins"));**

**this.table.setItems(data);**

**}**

**private void query4() throws SQLException {**

**logger.info("In query4");**

**TableColumn<Query, String>**

**idC = new TableColumn<>("ID куриер"),**

**nameC = new TableColumn<>("Име на куриер"),**

**officeC = new TableColumn<>("Офис"),**

**ordersC = new TableColumn<>("Брой регистрирани поръчки");**

**table.getColumns().addAll(List.of(idC, nameC, officeC, ordersC));**

**final ObservableList<Query> data = FXCollections.observableArrayList();**

**String sql = "select id\_courier, name, id\_office from couriers";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) {**

**Launch.alert("Няма куриери");**

**return;**

**}**

**do {**

**int id\_courier = rs.getInt("id\_courier");**

**String office = TableQuery.getOffice(rs.getInt("id\_office"));**

**sql = "select count(\*) from orders where id\_courier='" + id\_courier + "'";**

**ResultSet orders = TableQuery.execute(sql);**

**data.add(new Query4(id\_courier, orders.getInt(1), rs.getString("name"), office));**

**}while(rs.next());**

**idC.setCellValueFactory(new PropertyValueFactory<>("id"));**

**nameC.setCellValueFactory(new PropertyValueFactory<>("name"));**

**officeC.setCellValueFactory(new PropertyValueFactory<>("office"));**

**ordersC.setCellValueFactory(new PropertyValueFactory<>("orders"));**

**this.table.setItems(data);**

**}**

**private void query5() throws SQLException {**

**logger.info("In query5");**

**TableColumn<Query, String>**

**idC = new TableColumn<>("ID на клиент"),**

**nameC = new TableColumn<>("Име"),**

**phoneC = new TableColumn<>("Телефон"),**

**cityC = new TableColumn<>("Град"),**

**addressC = new TableColumn<>("Адрес");**

**table.getColumns().addAll(List.of(idC, nameC, phoneC, cityC, addressC));**

**final ObservableList<Query> data = FXCollections.observableArrayList();**

**String sql = "select \* from customers";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) {**

**Launch.alert("Няма клиенти");**

**return;**

**}**

**do {**

**Customer c = new Customer(rs.getInt("id\_customer"), rs.getString("name"), rs.getString("phone"), rs.getString("password"), rs.getInt("id\_city"), rs.getString("address"));**

**c.setCity\_name(TableQuery.cityIdToName(c.getId\_city()));**

**data.add(c);**

**} while(rs.next());**

**idC.setCellValueFactory(new PropertyValueFactory<>("id"));**

**nameC.setCellValueFactory(new PropertyValueFactory<>("name"));**

**phoneC.setCellValueFactory(new PropertyValueFactory<>("phone"));**

**cityC.setCellValueFactory(new PropertyValueFactory<>("city\_name"));**

**addressC.setCellValueFactory(new PropertyValueFactory<>("address"));**

**this.table.setItems(data);**

**}**

**}**

**class LoginController**

**package bg.tu\_varna.sit.group17.controllers;**

**import java.io.IOException;**

**import java.net.URL;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.util.ResourceBundle;**

**import bg.tu\_varna.sit.group17.application.Launch;**

**import bg.tu\_varna.sit.group17.application.Logger;**

**import bg.tu\_varna.sit.group17.application.Property;**

**import bg.tu\_varna.sit.group17.database.TableQuery;**

**import bg.tu\_varna.sit.group17.database.users.Admin;**

**import bg.tu\_varna.sit.group17.database.users.Courier;**

**import bg.tu\_varna.sit.group17.database.users.Customer;**

**import bg.tu\_varna.sit.group17.validation.Valid;**

**import javafx.event.ActionEvent;**

**import javafx.fxml.FXML;**

**import javafx.scene.control.Hyperlink;**

**import javafx.scene.control.TextField;**

**public final class LoginController {**

**private final String err = "Невалидни телефон и/или парола";**

**private final String[] users =**

**{"customers", "couriers", "admins"};**

**private final String ginko = "https://medpedia.framar.bg/%D0%B0%D0%BB%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0-%D0%BC%D0%B5%D0%B4%D0%B8%D1%86%D0%B8%D0%BD%D0%B0/%D0%BA%D0%B0%D0%BA-%D0%B4%D0%B0-%D0%BF%D0%BE%D0%B4%D1%81%D0%B8%D0%BB%D0%B8%D0%BC-%D0%BF%D0%B0%D0%BC%D0%B5%D1%82%D1%82%D0%B0-%D1%81%D0%B8";**

**private final Logger logger = new Logger(LoginController.class.getName());**

**@FXML**

**private TextField phone, pass;**

**@FXML**

**private Hyperlink forgot;**

**@FXML**

**private ResourceBundle resources;**

**@FXML**

**private URL location;**

**@FXML**

**void initialize() {**

**logger.info("In login form");**

**Property.resetAvatar();**

**Property.delivered = false;**

**Property.alertNotificationList.clear();**

**Property.ordersIdNotification.clear();**

**}**

**@FXML**

**void registerFx(ActionEvent event) throws SQLException, IOException {**

**String phoneNumber = this.phone.getText(),**

**password = this.pass.getText();**

**if(Valid.phoneNumber(phoneNumber) && Valid.password(password)) {**

**ResultSet record = null;**

**record = TableQuery.getRecordFromTable(**

**"phone", phoneNumber, "couriers", "password", password);**

**if(record != null) {**

**RegisterController.courier = Courier.create(record);**

**Launch.launch.registerForm();**

**return;**

**}**

**}**

**Launch.alert("Поискайте от куриер да ви регистрира");**

**}**

**@FXML**

**void loginFx(ActionEvent event) throws SQLException, IOException {**

**logger.info("Clicked login");**

**String phoneNumber = this.phone.getText(),**

**password = this.pass.getText();**

**if(!Valid.phoneNumber(phoneNumber) || !Valid.password(password)) {**

**Launch.alert(err);**

**return;**

**}**

**ResultSet record = null;**

**String recordTable = null;**

**for(int i = 0; i < users.length; ++i) {**

**record = TableQuery.getRecordFromTable(**

**"phone", phoneNumber, users[i], "password", password);**

**if(record != null) {**

**recordTable = users[i];**

**break;**

**}**

**}**

**if(recordTable == null) {**

**Launch.alert(err);**

**return;**

**}**

**switch(recordTable) {**

**case "customers":**

**HomeController.customer = Customer.create(record);**

**Property.user = 1;**

**Launch.launch.homeFormCustomer();**

**break;**

**case "couriers":**

**PratkaRegisterController.courier = Courier.create(record);**

**Property.user = 2;**

**Launch.launch.pratkaForm();**

**break;**

**case "admins":**

**FirmaController.admin = Admin.create(record);**

**Property.user = 3;**

**Launch.launch.firmaForm();**

**break;**

**}**

**}**

**@FXML**

**private void forgotPassword() {**

**Launch.launch.openLink(this.ginko);**

**}**

**}**

**PratkaRegisterController**

**package bg.tu\_varna.sit.group17.controllers;**

**import java.io.IOException;**

**import java.net.URL;**

**import java.sql.Date;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.time.LocalDate;**

**import java.util.LinkedList;**

**import java.util.ResourceBundle;**

**import bg.tu\_varna.sit.group17.application.Launch;**

**import bg.tu\_varna.sit.group17.application.Logger;**

**import bg.tu\_varna.sit.group17.application.Property;**

**import bg.tu\_varna.sit.group17.database.Add;**

**import bg.tu\_varna.sit.group17.database.TableQuery;**

**import bg.tu\_varna.sit.group17.database.Update;**

**import bg.tu\_varna.sit.group17.database.property.Company;**

**import bg.tu\_varna.sit.group17.database.users.Courier;**

**import bg.tu\_varna.sit.group17.validation.Valid;**

**import javafx.beans.value.ChangeListener;**

**import javafx.beans.value.ObservableValue;**

**import javafx.fxml.FXML;**

**import javafx.scene.control.Button;**

**import javafx.scene.control.CheckBox;**

**import javafx.scene.control.ComboBox;**

**import javafx.scene.control.DatePicker;**

**import javafx.scene.control.TextField;**

**import javafx.scene.image.ImageView;**

**public class PratkaRegisterController {**

**public LinkedList<Company> companies;**

**public Company company;**

**public static Courier courier;**

**private final Logger logger = new Logger(PratkaRegisterController.class.getName());**

**@FXML**

**private ComboBox<String> category, officeSender,**

**officeReceiver, companySender;**

**@FXML**

**private CheckBox fragile, sendToAddress, isPaid;**

**@FXML**

**private TextField phoneSender, phoneReceiver, sendPrice, address;**

**@FXML**

**public ImageView avatar;**

**@FXML**

**private DatePicker receiveDate, clientReceiveDate;**

**@FXML**

**private Button notificationBell;**

**@FXML**

**private ResourceBundle resources;**

**@FXML**

**private URL location;**

**@FXML**

**private void queries() throws SQLException, IOException {**

**Launch.launch.homeFormCourier();**

**}**

**@FXML**

**void initialize() throws SQLException {**

**logger.info("In pratka register form");**

**logger.info("Logged courier: " + courier);**

**this.avatar.setImage(Property.getAvatar());**

**courierNotification();**

**companies = TableQuery.allCompanies();**

**if(companySender.getItems().size() == 0) {**

**for(Company c : companies)**

**if(c.offices.size() != 0)**

**companySender.getItems().add(c.getName());**

**companySender.valueProperty().addListener(firmaListener());**

**}**

**String sql = "select category from categories";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) {**

**Launch.alert("Не е избрана категория");**

**return;**

**}**

**this.category.valueProperty().addListener(categoryListener());**

**do this.category.getItems().add(rs.getString("category"));**

**while(rs.next());**

**}**

**@FXML**

**private void changeAvatar() {**

**this.avatar.setImage(Property.nextAvatar());**

**}**

**@FXML**

**private void logOut() throws SQLException, IOException {**

**Launch.launch.loginForm();**

**}**

**@FXML**

**private void notificationBellClick() throws SQLException {**

**if(Property.delivered) {**

**this.notificationBell.setStyle(Property.izv);**

**Property.delivered = false;**

**String customers = "";**

**for(String s : Property.alertNotificationList) customers += s + " ";**

**Launch.alert("Отказани пратки от клиенти: " + customers);**

**for(int i : Property.ordersIdNotification)**

**Update.changeOrderStatus(i, Property.getStatus(4));**

**return;**

**}**

**Launch.alert("Нямате известия");**

**}**

**private void courierNotification() throws SQLException {**

**String sql = "select id\_order, id\_customer\_recipient from orders where id\_courier='" + courier.getId() + "' and id\_status='" + Property.getStatus(1) + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) return;**

**Property.delivered = true;**

**do {**

**Property.alertNotificationList.add(TableQuery.getCustomer(rs.getInt("id\_customer\_recipient")));**

**Property.ordersIdNotification.add(rs.getInt("id\_order"));**

**}while(rs.next());**

**notification();**

**}**

**private void notification() {**

**this.notificationBell.setStyle(Property.izv2);**

**}**

**@FXML**

**void registerOrder() throws SQLException {**

**logger.info("Clicked register order");**

**String phoneSender = this.phoneSender.getText(),**

**phoneReceiver = this.phoneReceiver.getText(),**

**address = this.address.getText();**

**LocalDate receiveDate = this.receiveDate.getValue(),**

**clientLocalDate = this.clientReceiveDate.getValue();**

**if(receiveDate == null || clientLocalDate == null) {**

**Launch.alert("Полетата за дати не може да са празни");**

**return;**

**}**

**boolean fragile = this.fragile.isSelected(),**

**sendToAddress = this.sendToAddress.isSelected(),**

**isPaid = this.isPaid.isSelected();**

**double price = 0;**

**if(this.sendPrice == null) {**

**Launch.alert("Полето за цена не може да е празно");**

**return;**

**}**

**try {**

**price = Double.parseDouble(this.sendPrice.getText());**

**}catch(Exception e) {return;}**

**if(!Valid.order(phoneSender, phoneReceiver,**

**this.receiveDate.getValue().toString(), this.clientReceiveDate.getValue().toString()))**

**{**

**Launch.alert("Невалиден телефон на клиент");**

**return;**

**}**

**if(!sendToAddress && officeSender.getSelectionModel().getSelectedItem().equals(officeReceiver.getSelectionModel().getSelectedItem())) {**

**Launch.alert("Не може да се достави до същия офис");**

**return;**

**}**

**if(sendToAddress && address.equals("")) {**

**Launch.alert("Полето адрес не може да е празно");**

**return;**

**}**

**if(Date.valueOf(receiveDate).after(Date.valueOf(clientLocalDate))) {**

**Launch.alert("Не може да се изпрати, преди дата на получаване");**

**return;**

**}**

**int id\_category = TableQuery.getCategoryId(category.getSelectionModel().getSelectedItem());**

**int id\_office\_sender = TableQuery.getOfficeId(officeSender.getSelectionModel().getSelectedItem()),**

**id\_office\_recipient = TableQuery.getOfficeId(officeReceiver.getSelectionModel().getSelectedItem()),**

**id\_customer\_sender = TableQuery.getCustomerId(phoneSender),**

**id\_customer\_recipient = TableQuery.getCustomerId(phoneReceiver),**

**id\_courier = PratkaRegisterController.courier.getId();**

**final int id\_status = 1;**

**if(id\_category == 0) {**

**System.out.println("Can't find ID");**

**return;**

**}**

**Add.order(id\_category, id\_office\_sender,**

**id\_office\_recipient, id\_customer\_sender,**

**id\_customer\_recipient, id\_courier, id\_status,**

**fragile, isPaid, price, sendToAddress, address,**

**Date.valueOf(receiveDate), Date.valueOf(clientLocalDate));**

**Launch.alert("Успешно добавена поръчка");**

**logger.info("Successful added order");**

**}**

**private ChangeListener<String> firmaListener() {**

**return new ChangeListener<String>() {**

**@Override**

**public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {**

**logger.info("Changed firma combobox");**

**officeSender.getItems().clear();**

**officeReceiver.getItems().clear();**

**company = new Company();**

**if(newValue == null) return;**

**company.setId(Property.companiesMap.get(newValue));**

**company.setName(newValue);**

**try {**

**for(int i = 0; i < companies.size(); ++i) {**

**for(int j = 0; j < companies.get(i).offices.size(); ++j)**

**officeSender.getItems().add(**

**companies.get(i).offices.get(j).getFullAddress());**

**}**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**officeReceiver.getItems().addAll(officeSender.getItems());**

**}**

**};**

**}**

**private ChangeListener<String> categoryListener() {**

**return new ChangeListener<String>() {**

**@Override**

**public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {**

**logger.info("Changed category combobox");**

**int id\_category = category.getSelectionModel().getSelectedIndex()+1;**

**try {**

**String sql = "select price from price\_list where id\_company='" + company.getId() + "' and id\_category='" + id\_category + "'";**

**ResultSet rs = TableQuery.execute(sql);**

**if(rs == null) {**

**Launch.alert("Фирмата няма цени");**

**return;**

**}**

**sendPrice.setText(Double.toString(rs.getDouble("price")));**

**} catch(SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**};**

**}**

**}**

**class RegisterController**

**package bg.tu\_varna.sit.group17.controllers;**

**import java.io.IOException;**

**import java.net.URL;**

**import java.sql.SQLException;**

**import java.util.ResourceBundle;**

**import bg.tu\_varna.sit.group17.application.Launch;**

**import bg.tu\_varna.sit.group17.application.Logger;**

**import bg.tu\_varna.sit.group17.application.Property;**

**import bg.tu\_varna.sit.group17.database.Add;**

**import bg.tu\_varna.sit.group17.database.users.Courier;**

**import bg.tu\_varna.sit.group17.validation.Valid;**

**import javafx.event.ActionEvent;**

**import javafx.fxml.FXML;**

**import javafx.fxml.Initializable;**

**import javafx.scene.control.ComboBox;**

**import javafx.scene.control.PasswordField;**

**import javafx.scene.control.TextField;**

**public final class RegisterController implements Initializable {**

**private final Logger logger = new Logger(RegisterController.class.getName());**

**@FXML**

**private TextField name, phone, address;**

**@FXML**

**private ComboBox<String> cBox0;**

**@FXML**

**private PasswordField password, repeatPassword;**

**public static Courier courier;**

**@Override**

**public void initialize(URL arg0, ResourceBundle arg1) {**

**logger.info("In register form");**

**cBox0.getItems().addAll(Property.citiesMap.keySet());**

**}**

**@FXML**

**public void register(ActionEvent e) throws IOException, SQLException {**

**logger.info("Clicked register");**

**String username = this.name.getText(),**

**phoneNumber = this.phone.getText(),**

**city = this.cBox0.getValue(),**

**location = this.address.getText(),**

**password = this.password.getText(),**

**repeatPassword = this.repeatPassword.getText();**

**String err = "";**

**err = Valid.user(username, phoneNumber, password, repeatPassword);**

**if(!err.equals("")) {**

**Launch.alert(err);**

**return;**

**}**

**if(city.equals("град")) {**

**city = "";**

**}**

**Add.customer(username, phoneNumber, Property.citiesMap.get(city), location, password);**

**Launch.alert("Успешна регистрация.");**

**logger.info("Successful register");**

**}**

**@FXML**

**public void login(ActionEvent e) throws SQLException, IOException {**

**Launch.launch.loginForm();**

**}**

**@FXML**

**private ResourceBundle resources;**

**@FXML**

**private URL location;**

**}**

**package bg.tu\_varna.sit.group17.database.property;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.util.LinkedList;**

**public class Company {**

**private int id;**

**private String name;**

**public LinkedList<Office> offices;**

**public Company() {**

**id = 0;**

**name = "";**

**offices = new LinkedList<>();**

**}**

**public Company(int id, String companyName) {**

**this.id = id;**

**this.name = companyName;**

**offices = new LinkedList<Office>();**

**}**

**public int getId() {**

**return id;**

**}**

**public void setId(int id) {**

**this.id = id;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public void setOffices(ResultSet rs) throws SQLException {**

**this.offices.clear();**

**Office office;**

**do {**

**office = new Office(rs.getInt("id\_office"), this.id, rs.getInt("id\_city"), rs.getString("address"));**

**//if(office.getId\_office() == 0) return;**

**offices.add(office);**

**}while(rs.next());**

**}**

**}**

**package bg.tu\_varna.sit.group17.database.property;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.util.LinkedList;**

**import bg.tu\_varna.sit.group17.database.TableQuery;**

**import bg.tu\_varna.sit.group17.database.users.Courier;**

**public class Office {**

**private int id\_office, id\_company, id\_city;**

**private String address;**

**public LinkedList<Courier> couriers;**

**public Office() {**

**this.id\_office = 0;**

**this.id\_company = 0;**

**this.id\_city = 0;**

**this.address = "";**

**this.couriers = new LinkedList<Courier>();**

**}**

**public Office(int id\_office, int id\_company, int id\_city, String address) {**

**this.id\_office = id\_office;**

**this.id\_company = id\_company;**

**this.id\_city = id\_city;**

**this.address = address;**

**this.couriers = new LinkedList<Courier>();**

**}**

**public int getId\_office() {**

**return id\_office;**

**}**

**public void setId\_office(int id\_office) {**

**this.id\_office = id\_office;**

**}**

**public int getId\_company() {**

**return id\_company;**

**}**

**public void setId\_company(int id\_company) {**

**this.id\_company = id\_company;**

**}**

**public int getId\_city() {**

**return id\_city;**

**}**

**public void setId\_city(int id\_city) {**

**this.id\_city = id\_city;**

**}**

**public String getAddress() {**

**return address;**

**}**

**public void setAddress(String address) {**

**this.address = address;**

**}**

**public String getCity() throws SQLException {**

**return TableQuery.cityIdToName(this.id\_city);**

**}**

**public String getFullAddress() throws SQLException {**

**return this.getCity() + " " + this.address;**

**}**

**public void setCouriers(ResultSet rs) throws SQLException {**

**this.couriers.clear();**

**Courier courier;**

**do {**

**courier = new Courier(rs.getInt("id\_courier"), rs.getString("name"), rs.getString("phone"), rs.getString("password"), this.id\_office);**

**//if(office.getId\_office() == 0) return;**

**couriers.add(courier);**

**}while(rs.next());**

**}**

**}**

**package bg.tu\_varna.sit.group17.database.property;**

**import java.sql.Date;**

**import bg.tu\_varna.sit.group17.database.queries.Query;**

**public class Order implements Query {**

**private int id\_order;**

**private String category, office\_sender, office\_recipient, customer\_sender, customer\_recipient, courier, status,**

**address;**

**private boolean fragile, paid, delivery\_to\_address;**

**private double cash\_on\_delivery;**

**private Date acceptance\_by\_sender, customer\_delivery;**

**public Order(int id\_order, String category, String office\_sender, String office\_recipient, String customer\_sender,**

**String customer\_recipient, String courier, String status, String address, boolean fragile, boolean paid,**

**boolean delivery\_to\_address, double cash\_on\_delivery, Date acceptance\_by\_sender, Date customer\_delivery) {**

**super();**

**this.id\_order = id\_order;**

**this.category = category;**

**this.office\_sender = office\_sender;**

**this.office\_recipient = office\_recipient;**

**this.customer\_sender = customer\_sender;**

**this.customer\_recipient = customer\_recipient;**

**this.courier = courier;**

**this.status = status;**

**this.address = address;**

**this.fragile = fragile;**

**this.paid = paid;**

**this.delivery\_to\_address = delivery\_to\_address;**

**this.cash\_on\_delivery = cash\_on\_delivery;**

**this.acceptance\_by\_sender = acceptance\_by\_sender;**

**this.customer\_delivery = customer\_delivery;**

**}**

**public int getId\_order() {**

**return id\_order;**

**}**

**public void setId\_order(int id\_order) {**

**this.id\_order = id\_order;**

**}**

**public String getCategory() {**

**return category;**

**}**

**public void setCategory(String category) {**

**this.category = category;**

**}**

**public String getOffice\_sender() {**

**return office\_sender;**

**}**

**public void setOffice\_sender(String office\_sender) {**

**this.office\_sender = office\_sender;**

**}**

**public String getOffice\_recipient() {**

**return office\_recipient;**

**}**

**public void setOffice\_recipient(String office\_recipient) {**

**this.office\_recipient = office\_recipient;**

**}**

**public String getCustomer\_sender() {**

**return customer\_sender;**

**}**

**public void setCustomer\_sender(String customer\_sender) {**

**this.customer\_sender = customer\_sender;**

**}**

**public String getCustomer\_recipient() {**

**return customer\_recipient;**

**}**

**public void setCustomer\_recipient(String customer\_recipient) {**

**this.customer\_recipient = customer\_recipient;**

**}**

**public String getCourier() {**

**return courier;**

**}**

**public void setCourier(String courier) {**

**this.courier = courier;**

**}**

**public String getStatus() {**

**return status;**

**}**

**public void setStatus(String status) {**

**this.status = status;**

**}**

**public String getAddress() {**

**return address;**

**}**

**public void setAddress(String address) {**

**this.address = address;**

**}**

**public boolean isFragile() {**

**return fragile;**

**}**

**public void setFragile(boolean fragile) {**

**this.fragile = fragile;**

**}**

**public boolean isPaid() {**

**return paid;**

**}**

**public void setPaid(boolean paid) {**

**this.paid = paid;**

**}**

**public boolean isDelivery\_to\_address() {**

**return delivery\_to\_address;**

**}**

**public void setDelivery\_to\_address(boolean delivery\_to\_address) {**

**this.delivery\_to\_address = delivery\_to\_address;**

**}**

**public Date getAcceptance\_by\_sender() {**

**return acceptance\_by\_sender;**

**}**

**public void setAcceptance\_by\_sender(Date acceptance\_by\_sender) {**

**this.acceptance\_by\_sender = acceptance\_by\_sender;**

**}**

**public Date getCustomer\_delivery() {**

**return customer\_delivery;**

**}**

**public void setCustomer\_delivery(Date customer\_delivery) {**

**this.customer\_delivery = customer\_delivery;**

**}**

**public double getCash\_on\_delivery() {**

**return cash\_on\_delivery;**

**}**

**public void setCash\_on\_delivery(double cash\_on\_delivery) {**

**this.cash\_on\_delivery = cash\_on\_delivery;**

**}**

**}**

**package bg.tu\_varna.sit.group17.database.queries;**

**public interface Query {**

**}**

**package bg.tu\_varna.sit.group17.database.queries;**

**import java.sql.Date;**

**public class Query2 implements Query {**

**private int number;**

**private String name, phone, status;**

**private Date from, to;**

**public Query2(int number, String name, String phone, String status, Date from, Date to) {**

**super();**

**this.number = number;**

**this.name = name;**

**this.phone = phone;**

**this.status = status;**

**this.from = from;**

**this.to = to;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public String getPhone() {**

**return phone;**

**}**

**public void setPhone(String phone) {**

**this.phone = phone;**

**}**

**public String getStatus() {**

**return status;**

**}**

**public void setStatus(String status) {**

**this.status = status;**

**}**

**public Date getFrom() {**

**return from;**

**}**

**public void setFrom(Date from) {**

**this.from = from;**

**}**

**public Date getTo() {**

**return to;**

**}**

**public void setTo(Date to) {**

**this.to = to;**

**}**

**public int getNumber() {**

**return number;**

**}**

**public void setNumber(int number) {**

**this.number = number;**

**}**

**}**

**package bg.tu\_varna.sit.group17.database.queries;**

**public class Query3 implements Query {**

**private int company, orders, offices, couriers, admins;**

**public Query3(int company, int orders, int offices, int couriers, int admins) {**

**super();**

**this.company = company;**

**this.orders = orders;**

**this.offices = offices;**

**this.couriers = couriers;**

**this.admins = admins;**

**}**

**public int getOrders() {**

**return orders;**

**}**

**public void setOrders(int orders) {**

**this.orders = orders;**

**}**

**public int getOffices() {**

**return offices;**

**}**

**public void setOffices(int offices) {**

**this.offices = offices;**

**}**

**public int getCouriers() {**

**return couriers;**

**}**

**public void setCouriers(int couriers) {**

**this.couriers = couriers;**

**}**

**public int getAdmins() {**

**return admins;**

**}**

**public void setAdmins(int admins) {**

**this.admins = admins;**

**}**

**public int getCompany() {**

**return company;**

**}**

**public void setCompany(int company) {**

**this.company = company;**

**}**

**}**

**package bg.tu\_varna.sit.group17.database.queries;**

**public class Query4 implements Query {**

**private int id, orders;**

**private String name, office;**

**public Query4(int id, int orders, String name, String office) {**

**super();**

**this.id = id;**

**this.orders = orders;**

**this.name = name;**

**this.office = office;**

**}**

**public int getId() {**

**return id;**

**}**

**public void setId(int id) {**

**this.id = id;**

**}**

**public int getOrders() {**

**return orders;**

**}**

**public void setOrders(int orders) {**

**this.orders = orders;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public String getOffice() {**

**return office;**

**}**

**public void setOffice(String office) {**

**this.office = office;**

**}**

**}**

**package bg.tu\_varna.sit.group17.database.users;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**public class Admin extends Consumer {**

**public Admin() {**

**super();**

**}**

**public Admin(int id, String name, String phone, String password) {**

**super(id, name, phone, password);**

**}**

**public static Admin create(ResultSet rs) throws SQLException {**

**Consumer c = Consumer.create(rs);**

**return new Admin(c.getId(), c.getName(), c.getPhone(), c.getPassword());**

**}**

**}**

**package bg.tu\_varna.sit.group17.database.users;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import bg.tu\_varna.sit.group17.database.queries.Query;**

**public class Consumer implements Query {**

**protected int id;**

**protected String name;**

**protected String phone;**

**protected String password;**

**public Consumer() {**

**this.id = 0;**

**this.name = "";**

**this.phone = "";**

**this.password = "";**

**}**

**public Consumer(int id, String name, String phone, String password) {**

**this.id = id;**

**this.name = name;**

**this.phone = phone;**

**this.password = password;**

**}**

**public String toString() {**

**return "id: " + id + " name: " + name + " phone: " + phone;**

**}**

**public static Consumer create(ResultSet rs) throws SQLException {**

**int id = 0;**

**String name = null, phone = null, password = null;**

**id = rs.getInt(1);**

**name = rs.getString(2);**

**phone = rs.getString(3);**

**password = rs.getString(4);**

**return new Consumer(id, name, phone, password);**

**}**

**public int getId() {**

**return id;**

**}**

**public void setId(int id) {**

**this.id = id;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public String getPhone() {**

**return phone;**

**}**

**public void setPhone(String phone) {**

**this.phone = phone;**

**}**

**public String getPassword() {**

**return password;**

**}**

**public void setPassword(String password) {**

**this.password = password;**

**}**

**}**

**package bg.tu\_varna.sit.group17.database.users;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**public class Courier extends Consumer {**

**private int id\_office;**

**public Courier() {**

**super();**

**this.id\_office = 0;**

**}**

**public Courier(int id, String name, String phone, String password, int id\_office) {**

**super(id, name, phone, password);**

**this.setId\_office(id\_office);**

**}**

**public String toString() {**

**return "id: " + id + " name: " + name + " phone: " + phone;**

**}**

**public static Courier create(ResultSet rs) throws SQLException {**

**Consumer c = Consumer.create(rs);**

**int id\_office = 0;**

**try {**

**id\_office = rs.getInt(5);**

**} catch(SQLException e) {**

**e.printStackTrace();**

**}**

**return new Courier(c.getId(), c.getName(), c.getPhone(), c.getPassword(), id\_office);**

**}**

**public int getId\_office() {**

**return id\_office;**

**}**

**public void setId\_office(int id\_office) {**

**this.id\_office = id\_office;**

**}**

**}**

**package bg.tu\_varna.sit.group17.database.users;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**public class Customer extends Consumer {**

**private int id\_city;**

**private String address, city\_name = "";**

**public Customer() {**

**super();**

**this.id\_city = 0;**

**this.address = "";**

**}**

**public Customer(int id, String name, String phone, String password, int id\_city, String address) {**

**super(id, name, phone, password);**

**this.id\_city = id\_city;**

**this.address = address;**

**}**

**public static Customer create(ResultSet rs) throws SQLException {**

**Consumer c = Consumer.create(rs);**

**int id\_city = 0;**

**String address = null;**

**try {**

**id\_city = rs.getInt(5);**

**address = rs.getString(6);**

**} catch(SQLException e) {**

**e.printStackTrace();**

**}**

**return new Customer(c.getId(), c.getName(), c.getPhone(), c.getPassword(), id\_city, address);**

**}**

**public int getId\_city() {**

**return id\_city;**

**}**

**public void setId\_city(int id\_city) {**

**this.id\_city = id\_city;**

**}**

**public String getAddress() {**

**return address;**

**}**

**public void setAddress(String address) {**

**this.address = address;**

**}**

**public String getCity\_name() {**

**return city\_name;**

**}**

**public void setCity\_name(String city\_name) {**

**this.city\_name = city\_name;**

**}**

**}**

**package bg.tu\_varna.sit.group17.database.users;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.SQLException;**

**import bg.tu\_varna.sit.group17.database.Create;**

**import bg.tu\_varna.sit.group17.database.property.Company;**

**import bg.tu\_varna.sit.group17.database.property.Office;**

**public class Delete {**

**public static void company(Company company) throws SQLException {**

**if(company == null) {**

**return;**

**}**

**Connection conn = Create.getConnection();**

**if(company.offices.size() != 0)**

**for(int i = 0; i < company.offices.size(); ++i)**

**Delete.office(company.offices.get(i));**

**final String sql = "DELETE FROM companies WHERE id\_company=?";**

**PreparedStatement st = conn.prepareStatement(sql);**

**st.setInt(1, company.getId());**

**st.execute();**

**st.close();**

**}**

**public static void companyCategory(Company company) throws SQLException {**

**if(company == null) {**

**return;**

**}**

**Connection conn = Create.getConnection();**

**if(company.offices.size() != 0)**

**for(int i = 0; i < company.offices.size(); ++i)**

**Delete.office(company.offices.get(i));**

**String sql = "delete from price\_list where id\_company=?";**

**PreparedStatement ps = conn.prepareStatement(sql);**

**ps.setInt(1, company.getId());**

**ps.execute();**

**sql = "DELETE FROM companies WHERE id\_company=?";**

**ps = conn.prepareStatement(sql);**

**ps.setInt(1, company.getId());**

**ps.execute();**

**ps.close();**

**}**

**public static void office(Office office) throws SQLException {**

**if(office == null) {**

**return;**

**}**

**Connection conn = Create.getConnection();**

**if(office.couriers.size() != 0)**

**for(int i = 0; i < office.couriers.size(); ++i)**

**Delete.courier(office.couriers.get(i));**

**final String sql = "DELETE FROM office WHERE id\_office=?";**

**PreparedStatement st = conn.prepareStatement(sql);**

**st.setInt(1, office.getId\_office());**

**st.execute();**

**st.close();**

**}**

**public static void courier(Courier courier) throws SQLException {**

**if(courier == null) {**

**return;**

**}**

**Connection conn = Create.getConnection();**

**final String sql = "DELETE FROM couriers WHERE id\_courier=?";**

**PreparedStatement st = conn.prepareStatement(sql);**

**st.setInt(1, courier.getId());**

**st.execute();**

**st.close();**

**}**

**}**

**package bg.tu\_varna.sit.group17.validation;**

**import java.sql.SQLException;**

**import java.time.LocalDate;**

**import java.time.format.DateTimeParseException;**

**import java.util.regex.Pattern;**

**import bg.tu\_varna.sit.group17.application.Launch;**

**import bg.tu\_varna.sit.group17.database.TableQuery;**

**public final class Valid {**

**public static boolean username(String username) {**

**if(username == null || username.equals("")) return false;**

**if(username.length() > 45) return false;**

**final String regex = "1234567890!@№%€§\*()-–=+`~\_#^&[]{};:\'\"$\\/,<.>|?";**

**for(char ch : regex.toCharArray())**

**if(username.contains(Character.toString(ch)))**

**return false;**

**return true;**

**}**

**public static boolean phoneNumber(String phone) {**

**if(phone == null || phone.isBlank() || phone.length() != 12)**

**return false;**

**final String regex = "[0-9]+";**

**final Pattern p = Pattern.compile(regex);**

**return p.matcher(phone).matches();**

**}**

**public static boolean password(String password) {**

**if(password == null) return false;**

**return password.length() > 4 && password.length() < 31;**

**}**

**public static String user(String name, String phone, String password, String repeatPassword) throws SQLException {**

**if(!Valid.username(name))**

**return "Името не трябва да съдържа символи и да е твърде кратко";**

**if(!Valid.phoneNumber(phone))**

**return "Телефона трябва да съдържа 12 цифри";**

**if(!Valid.password(password))**

**return "Паролата трябва да е между 5 и 30 символа";**

**if(!password.equals(repeatPassword))**

**return "Паролите не съвпадат";**

**if(TableQuery.phoneExists(phone))**

**return "Вече съществува такъв телефон в базата данни";**

**return "";**

**}**

**public static boolean date(String date) {**

**try {**

**LocalDate.parse(date);**

**}catch(DateTimeParseException e) {return false;}**

**return true;**

**}**

**public static boolean order(String phoneSender, String phoneReceiver, String date1, String date2) {**

**if(phoneSender == null || phoneReceiver == null || date1 == null || date2 == null) return false;**

**if(!Valid.date(date1) || !Valid.date(date2)) {**

**Launch.alert("Невалидни дати");**

**return false;**

**}**

**if(phoneSender.equals("") || phoneReceiver.equals("") || date1.equals("") || date2.equals("")) {**

**Launch.alert("Всички полета трябва да са попълнени");**

**return false;**

**}**

**if(phoneSender.equals(phoneReceiver)) {**

**Launch.alert("Телефоните не може да са еднакви");**

**return false;**

**}**

**if(!Valid.phoneNumber(phoneSender) || ! Valid.phoneNumber(phoneReceiver)) {**

**Launch.alert("Невалиден телефонен номер");**

**return false;**

**}**

**return true;**

**}**

**public static boolean price(double price) {**

**if(price < 0) return false;**

**return true;**

**}**

**}**

**class Logger**

**package bg.tu\_varna.sit.group17.application;**

**public class Logger {**

**private String className;**

**public Logger(String className) {**

**super();**

**this.className = className;**

**}**

**public String getClassName() {**

**return className;**

**}**

**public void setClassName(String className) {**

**this.className = className;**

**}**

**public void info(String info) {**

**org.apache.log4j.Logger.getLogger(className).info(info);**

**}**

**}**

**log4j.properties**

**# Root logger option**

**log = ./Logs**

**log4j.rootLogger= DEBUG, file**

**#, stdout**

**# configuration to print into file**

**log4j.appender.file=org.apache.log4j.RollingFileAppender**

**log4j.appender.file.File=${log}/logfile.log**

**log4j.appender.file.MaxFileSize=12MB**

**log4j.appender.file.MaxBackupIndex=10**

**log4j.appender.file.layout=org.apache.log4j.PatternLayout**

**log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n**

**# configuration to print on console**

**#log4j.appender.stdout=org.apache.log4j.ConsoleAppender**

**#log4j.appender.stdout.Target=System.out**

**#log4j.appender.stdout.layout=org.apache.log4j.PatternLayout**

**#log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n**

**class ValidationTest**

**package bg.tu\_varna.sit.group17.test;**

**import static org.junit.jupiter.api.Assertions.assertEquals;**

**import java.sql.SQLException;**

**import org.junit.jupiter.api.Test;**

**import bg.tu\_varna.sit.group17.validation.Valid;**

**class ValidationTest {**

**@Test**

**void usernameLength() {**

**assertEquals(false, Valid.username("Uuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu"));**

**assertEquals(false, Valid.username("Uuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu"));**

**assertEquals(true, Valid.username("Uuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu"));**

**assertEquals(true, Valid.username("Uuuuuuuuuuuuuuuuu"));**

**}**

**@Test**

**void usernameEmpty() {**

**assertEquals(false, Valid.username(""));**

**assertEquals(false, Valid.username(null));**

**assertEquals(false, Valid.username(new String()));**

**}**

**@Test**

**void usernameInvalidChars() {**

**assertEquals(false, Valid.username("Susd^f"));**

**assertEquals(false, Valid.username("Susdf&"));**

**assertEquals(false, Valid.username("$Susdf"));**

**assertEquals(false, Valid.username("\*&@"));**

**assertEquals(false, Valid.username("A1awq"));**

**assertEquals(false, Valid.username("Sus3df"));**

**assertEquals(false, Valid.username("Gwdio5"));**

**}**

**@Test**

**void phoneNumberEmpty() {**

**assertEquals(false, Valid.phoneNumber(""));**

**assertEquals(false, Valid.phoneNumber(null));**

**assertEquals(false, Valid.phoneNumber(new String()));**

**}**

**@Test**

**void phoneNumberLength() {**

**assertEquals(true, Valid.phoneNumber("123452234532"));**

**assertEquals(false, Valid.phoneNumber("1234522345562"));**

**assertEquals(false, Valid.phoneNumber("12345"));**

**assertEquals(false, Valid.phoneNumber("12345232132342424"));**

**}**

**@Test**

**void phoneNumberInvalidChars() {**

**assertEquals(false, Valid.phoneNumber("\*12345234532"));**

**assertEquals(false, Valid.phoneNumber("12345234532("));**

**assertEquals(false, Valid.phoneNumber("a12345234532"));**

**assertEquals(false, Valid.phoneNumber("123452^34532"));**

**}**

**@Test**

**void passwordEmpty() {**

**assertEquals(false, Valid.password(""));**

**assertEquals(false, Valid.password(null));**

**assertEquals(false, Valid.password(new String()));**

**}**

**@Test**

**void passwordLength() {**

**assertEquals(false, Valid.password("\*1234523453sdfsw453223452345322345234532"));**

**assertEquals(true, Valid.password("234523wer4532(2453223r45234532"));**

**assertEquals(true, Valid.password("a12345234532"));**

**assertEquals(true, Valid.password("qwert"));**

**}**

**@Test**

**void user() throws SQLException {**

**String s = "", username = "Името не трябва да съдържа символи и да е твърде кратко",**

**phone = "Телефона трябва да съдържа 12 цифри",**

**passBig = "Паролата трябва да е между 5 и 30 символа",**

**repeatPass = "Паролите не съвпадат";**

**s = Valid.user("Ivan Ivanov", "098378291234", "12342", "12342");**

**assertEquals("", s);**

**s = Valid.user("Iv%an Ivanov", "098378291234", "1234", "1234");**

**assertEquals(username, s);**

**s = Valid.user("Ivan Ivanov", "09837u291234", "1234", "1234");**

**assertEquals(phone, s);**

**s = Valid.user("Ivan7Ivanov", "098378291234", "1234", "1234");**

**assertEquals(username, s);**

**s = Valid.user("Ivan Ivanov", "098378291234", "12234", "1234");**

**assertEquals(repeatPass, s);**

**s = Valid.user("Ivan7Ivanov", "098378291234", "12y34", "12y34");**

**assertEquals(username, s);**

**s = Valid.user("Ivan7Ivanov", "098378291234", "1234", "1234r");**

**assertEquals(username, s);**

**s = Valid.user(null, "098378291234", "1234", "1234r");**

**assertEquals(username, s);**

**s = Valid.user("Ivan Ivanov", null, "1234", "1234r");**

**assertEquals(phone, s);**

**s = Valid.user("Ivan Ivanov", "098378291234", null, "1234r");**

**assertEquals(passBig, s);**

**s = Valid.user("Ivan Ivanov", "098378291234", "12344", null);**

**assertEquals(repeatPass, s);**

**}**

**@Test**

**void date() {**

**assertEquals(true, Valid.date("2021-12-27"));**

**assertEquals(false, Valid.date("202112-27"));**

**assertEquals(false, Valid.date("20621-12-27"));**

**assertEquals(false, Valid.date("2021-12-78"));**

**assertEquals(false, Valid.date("2021-13-18"));**

**assertEquals(false, Valid.date("021-12-78"));**

**}**

**@Test**

**void order() {**

**assertEquals(true, Valid.order("098937281232", "239048273214", "2021-12-21", "2021-12-27"));**

**assertEquals(false, Valid.order("098937281232", "239048273214", "2021--21", "202112-27"));**

**assertEquals(false, Valid.order("098937281232", "239048273214", "20211221", "2021-12-27"));**

**assertEquals(false, Valid.order("098937281232", "239048273214", "2021-12-21", "2021-12-78"));**

**}**

**@Test**

**void price() {**

**assertEquals(true , Valid.price(1234));**

**assertEquals(true , Valid.price(345.2));**

**assertEquals(true , Valid.price(0));**

**assertEquals(false , Valid.price(-1));**

**assertEquals(false , Valid.price(-144));**

**}**

**}**