Министерство образования и науки РФ
Санкт-Петербургский политехнический университет
Петра Великого

Институт компьютерных наук и технологий

Высшая школа «Киберфизических систем и управления»

УТВЕРЖДАЮ

_____

_____

_____

«____» _____ _____ г.

ИТОГОВЫЙ ПРОЕКТ

по дисциплине «Системный подход в разработке ПО»

Приложение «Автосалон»

Выполнил:

студент гр. 3530902/80201 _____Кулешов И.Н.

подпись, дата

Проверил

к.т.н. _____ Нестеров С.А.

подпись, дата

Санкт-Петербург 2021

Оглавление

Постановка задачи

Разработать и реализовать базу данных для заявленной предметной области:

1. Описать предметную область, выделить правила и ограничения.
2. Построить логическую модель, определить атрибуты, идентификаторы, связи.
3. Перейти к реляционной модели. Уточнить типы данных, разрешить связи многие-ко-многим, описать ограничения.
4. Создать базу данных в MySQL. При необходимости, добавить ограничения, не учтенные в модели. Внести тестовые данные, проверить правильность работы ограничений.
5. Создать клиентское приложение.
6. Описывать и реализовать план тестирования приложения.

Описание предметной области

Автосалон “МОРО”.

В базе данных хранится информация о сотрудниках (Номер, фамилия, имя, телефон, почта, должность), клиентах (Номер, фамилия, имя, отчество, телефон, почта), информация об автомобиле (Номер, марка, серия, количество в наличии), категория автомобиля (Стандарт, комфорт, премиум), информация о доп. услугах (Шины, камера заднего вида, цвет, салон) и информация о продажах (Общая сумма заказа, номер сотрудника, номер клиента, номер автомобиля, стоимость автомобиля, категория автомобиля, стоимость доп. услуги, название доп. услуги).

Клиент:

1. Заполняет свои личные данные;
2. Выбирает категорию автомобиль;
3. В зависимости от категории выбирает автомобиль;
4. Выбирает доп. услугу;
5. Подтверждает покупку или отклоняет;
6. При подтверждении может посмотреть информацию о заказе;
7. При отказе выходит из приложения.

Администратор может посмотреть историю продаж, информацию о сотрудниках.

Также администратор может очистить историю продаж.

1 USE-CASE диаграмма

Для наглядной работы программы каждого пользователя сделана UML-диаграмма вариантов использования use-case (см. рис. 1).

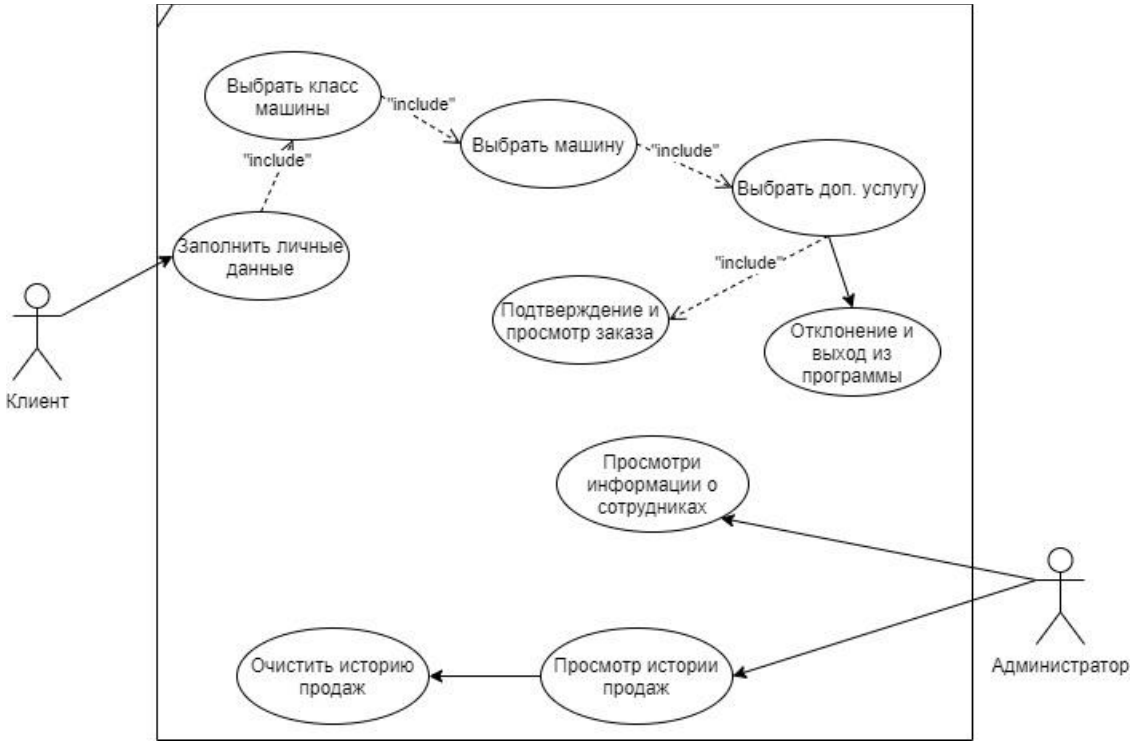


Рисунок 1-UML-Диаграмма

2 Логическая модель

На рисунке 2 представлены сущности и связи между ними.

Выполнено в программе OracleDataModeler.

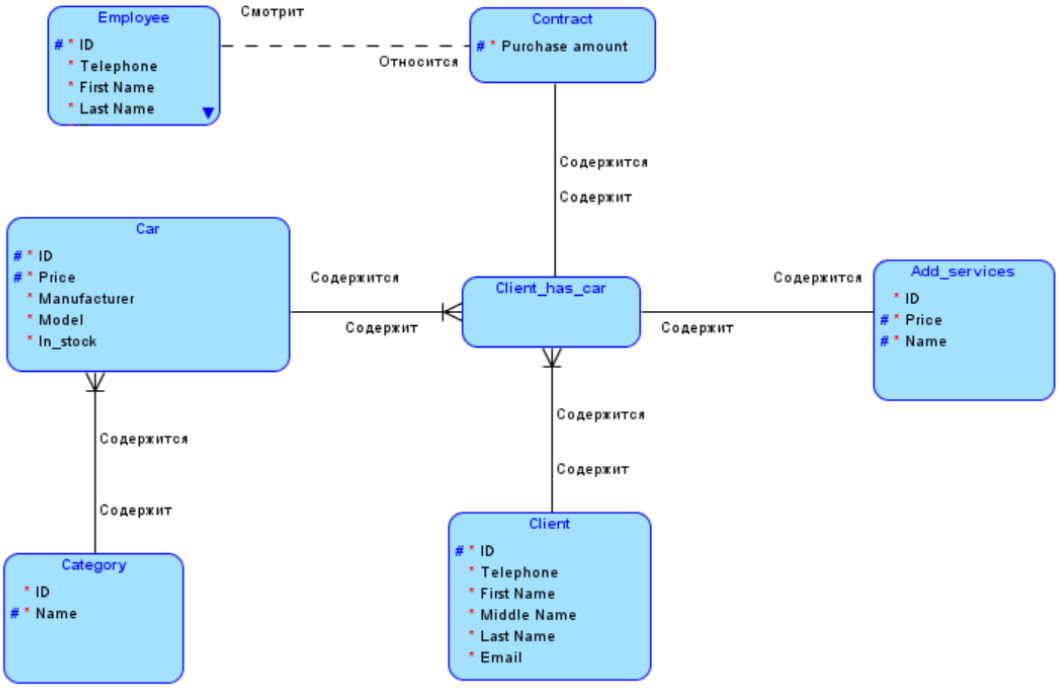


Рисунок 2-Логическая модель

3 Реляционная модель

На реляционной модели, представленной на рисунке 3, видно тип каждого атрибута, а также первичные и внешние ключи.

Реляционная модель построена в программе OracleDataModeler.

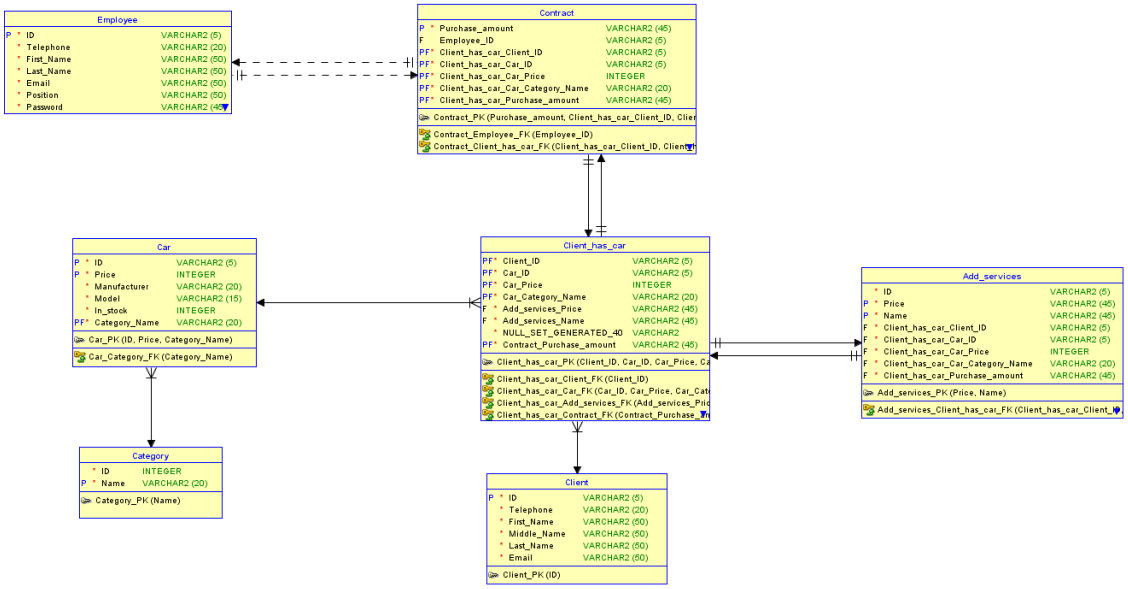


Рисунок 3-Логическая модель

4 Соответствие нормальным формам

Первая нормальная форма:

Все таблицы находятся в первой нормальной форме, так как содержат только скалярные значения атрибутов и ни один из ключевых атрибутов не содержит значение NULL.

Ключевой атрибут – атрибут, входящий в любой из потенциальных ключей.

Вторая нормальная форма:

Все таблицы находятся во второй нормальной форме, так как они удовлетворяют определению первой нормальной формы и все его атрибуты, не входящие в первичный ключ, неприводимо зависят от него.

Функциональная зависимость называется неприводимой, если выполняются следующие условия:

- правая (зависимая) часть является одноэлементным множеством, то есть справа находится только один атрибут;

- левая часть (детерминант) является неприводимой, то есть ни один атрибут из детерминанта не может быть опущен без потери ФЗ.

Третья нормальная форма:

Все таблицы находятся в третьей нормальной форме, так как они удовлетворяют определению второй нормальной формы и ни один из его неключевых атрибутов не зависит функционально от любого другого неключевого атрибута.

Нормальная форма Бойса-Кодда:

Таблица находится в нормальной форме Бойса-Кодда, тогда и только тогда, когда все детерминанты нетривиальных и неприводимых ФЗ являются потенциальными ключами. Нормальная форма Бойса-Кодда – это частный случай третьей нормальной формы для отношений, имеющих один потенциальный ключ.

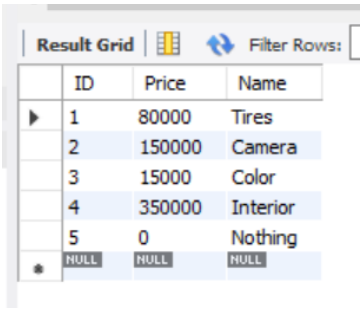
Все таблицы находятся в нормальной форме Бойса-Кодда, так как каждая нетривиальная и неприводимая слева функциональная зависимость обладает потенциальным ключом в качестве детерминанта.

В моей базе данных представлены таблицы, для которых соответсвие нормальной форме Бойса-Кодда вытекает из соответствия таблиц третьей нормальной форме.

5 Создание базы данных

Для разработки баз данных был использован MySQL WorkBench.

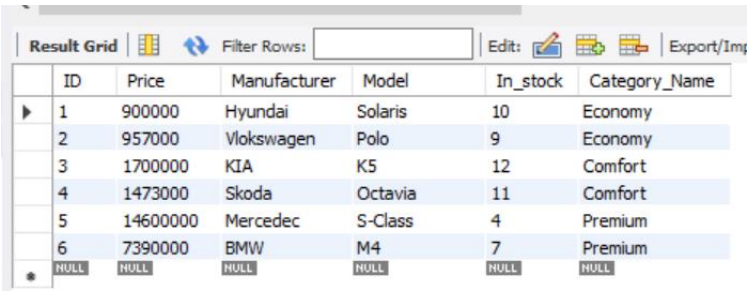
Скрипт приведен в приложении 1.



Result Grid | Filter Rows:

ID	Price	Name
1	80000	Tires
2	150000	Camera
3	15000	Color
4	350000	Interior
5	0	Nothing
NULL	NULL	NULL

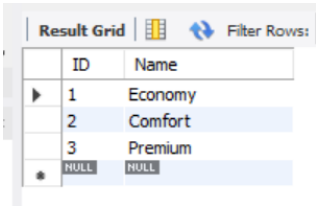
Рисунок 4-Таблица add_services



Result Grid | Filter Rows: | Edit: | Export/Im

ID	Price	Manufacturer	Model	In_stock	Category_Name
1	900000	Hyundai	Solaris	10	Economy
2	957000	Vlokswagen	Polo	9	Economy
3	1700000	KIA	K5	12	Comfort
4	1473000	Skoda	Octavia	11	Comfort
5	14600000	Mercedec	S-Class	4	Premium
6	7390000	BMW	M4	7	Premium
NULL	NULL	NULL	NULL	NULL	NULL

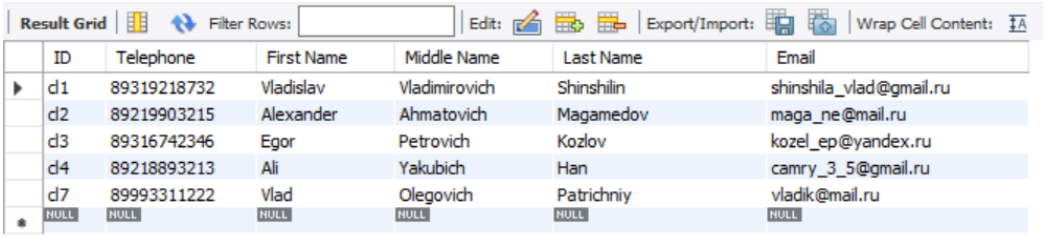
Рисунок 5-Таблица car



Result Grid | Filter Rows:

ID	Name
1	Economy
2	Comfort
3	Premium
NULL	NULL

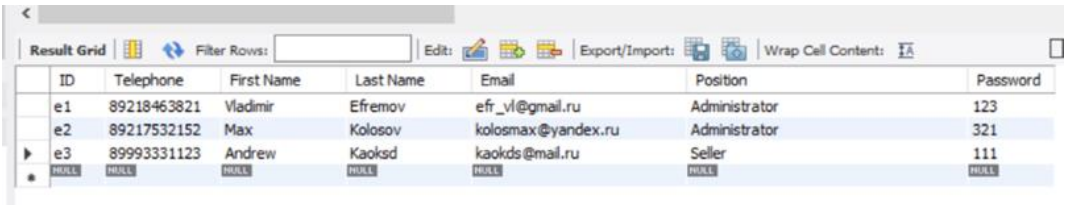
Рисунок 6-Таблица category



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ID	Telephone	First Name	Middle Name	Last Name	Email
cl1	89319218732	Vladislav	Vladimirovich	Shinshilin	shinshila_vlad@gmail.ru
cl2	89219903215	Alexander	Ahmatovich	Magamedov	maga_ne@mail.ru
cl3	89316742346	Egor	Petrovich	Kozlov	kozel_ep@yandex.ru
cl4	89218893213	Ali	Yakubich	Han	camry_3_5@gmail.ru
cl7	89993311222	Vlad	Olegovich	Patrichniy	vladik@mail.ru
NULL	NULL	NULL	NULL	NULL	NULL

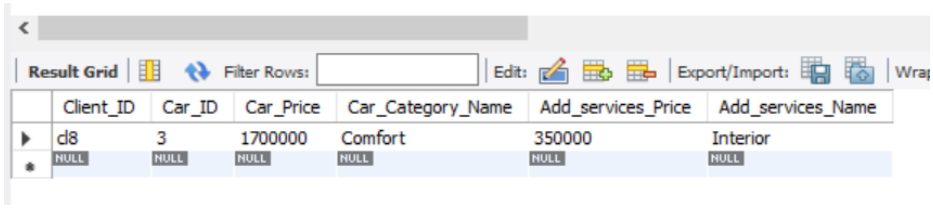
Рисунок 7-Таблица client



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

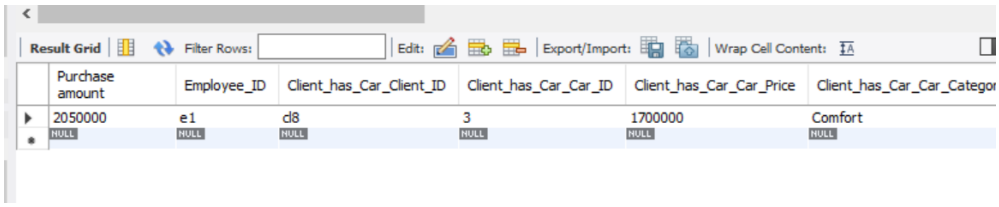
ID	Telephone	First Name	Last Name	Email	Position	Password
e1	89218463821	Vladimir	Efremov	efr_vl@gmail.ru	Administrator	123
e2	89217532152	Max	Kolosov	kolosmax@yandex.ru	Administrator	321
e3	89993331123	Andrew	Kaoksd	kaokds@mail.ru	Seller	111
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 8-Таблица employee



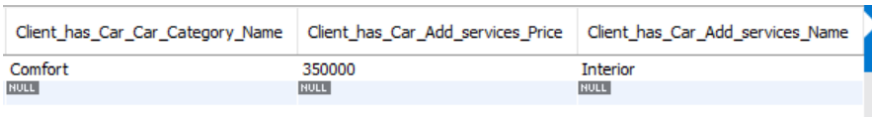
Client_ID	Car_ID	Car_Price	Car_Category_Name	Add_services_Price	Add_services_Name
cl8	3	1700000	Comfort	350000	Interior
NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 9-Таблица client_has_car



Purchase amount	Employee_ID	Client_has_Car_Client_ID	Client_has_Car_Car_ID	Client_has_Car_Car_Price	Client_has_Car_Car_Categor
2050000	e1	cl8	3	1700000	Comfort
NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 10-Таблица contract



Client_has_Car_Car_Category_Name	Client_has_Car_Add_services_Price	Client_has_Car_Add_services_Name
Comfort	350000	Interior
NULL	NULL	NULL

Рисунок 11-Таблица contract продолжение

6 UML-диаграмма

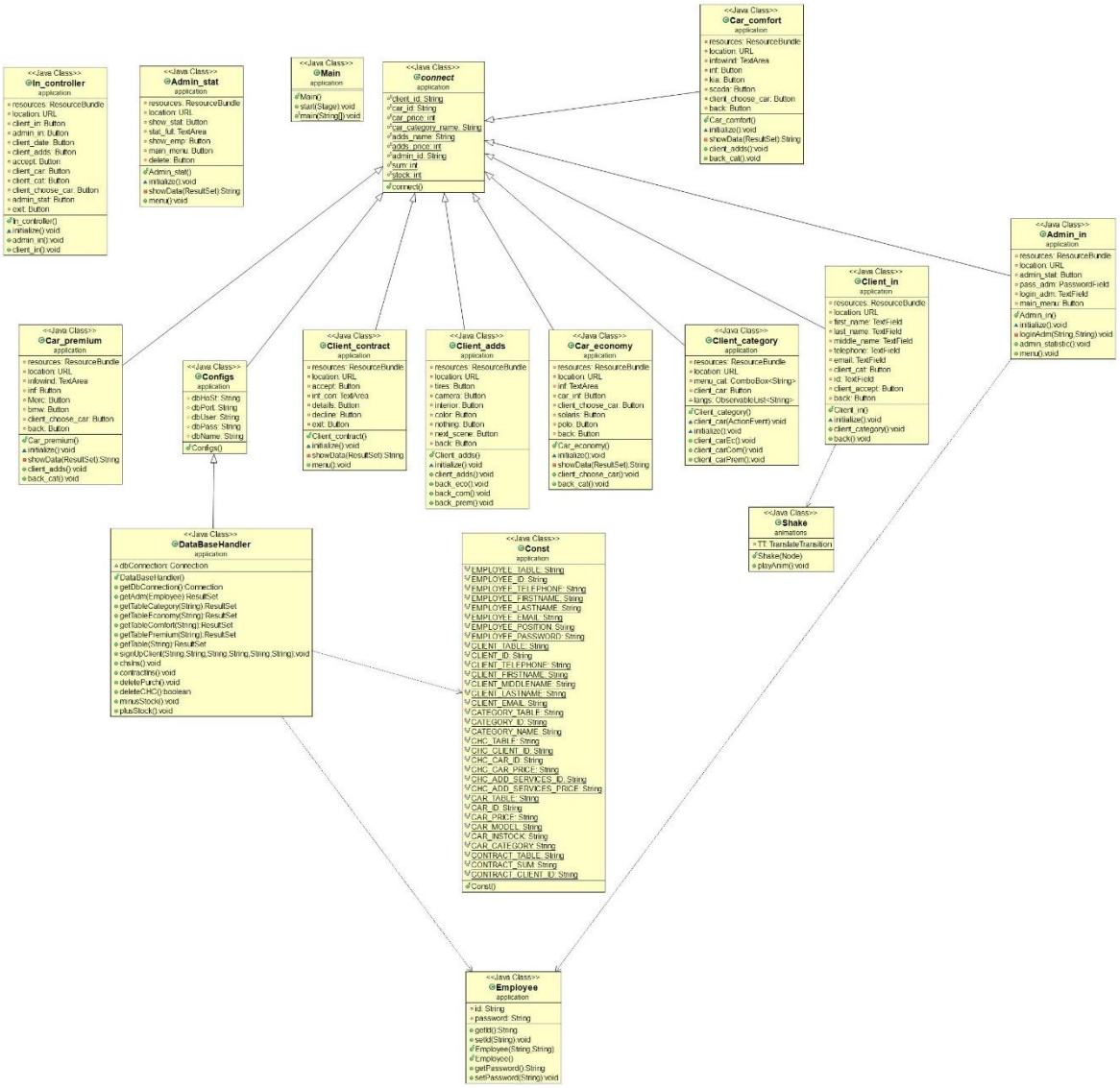


Рисунок 12-UML-диаграмма

На диаграмме отражены 17 используемых классов, в каждом из которых отображены объявленные переменные и методы.

Также на данной диаграмме видны связи между классами.

7 Клиентское приложение

При запуске пользователь видит данное окно:

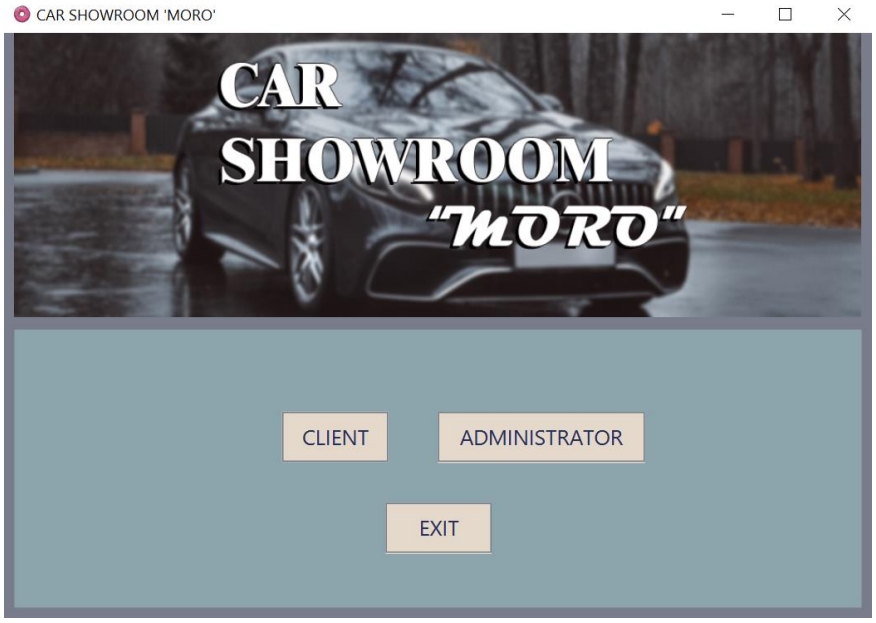


Рисунок 13-Начальное окно

Пользователь должен выбрать, клиент он или администратор, либо просто выйти из приложения.

7.1 Клиент

При нажатии на кнопку CLIENT в начальном окне, пользователь увидит следующее окно:

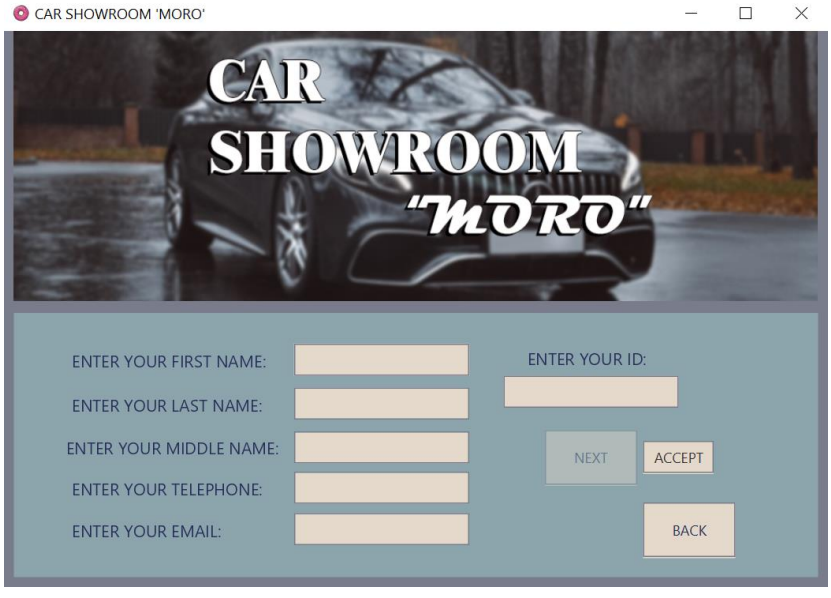


Рисунок 14-Окно регистрации клиента

Далее клиент вводит свои личные данные, такие как:

- Имя;
- Фамилия;
- Отчество;
- Телефон;
- Почта
- Клиентский айди.

Поля с вводом данных (FIRST NAME, LAST NAME, MIDDLE NAME) должны содержать не менее одной английской буквы.

Поле с вводом данных TELEPHONE должно содержать 11 цифр, начинаться телефонный номер должен с 8 или 7.

Поле с вводом данных EMAIL должно соответствовать шаблону asd@asd.asd, где “asd” любые цифры или буквы английского языка.

Поле с вводом данных ID должно соответствовать шаблону cl*, где *- любая цифра.

При неверном вводе одного из полей, пользователь будет видеть анимацию тряски, которая говорит о том, что введено что-то неверно.

Также если оставить все поля пустыми, то также будет видна анимация, которая показывает, что введено что-то не так.

Далее нажимает на кнопку ACCEPT, тогда кнопка NEXT становится видна.

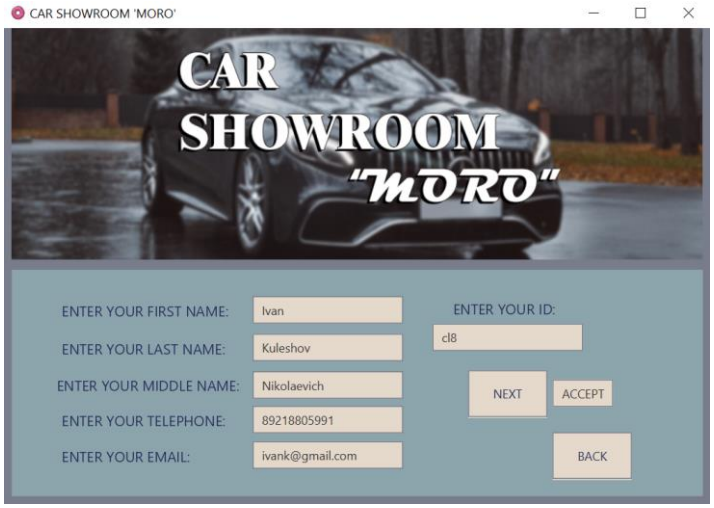


Рисунок 15-Кнопка NEXT становится активной

После нажатия на кнопку NEXT клиент видит перед собой окно с выбором категории:

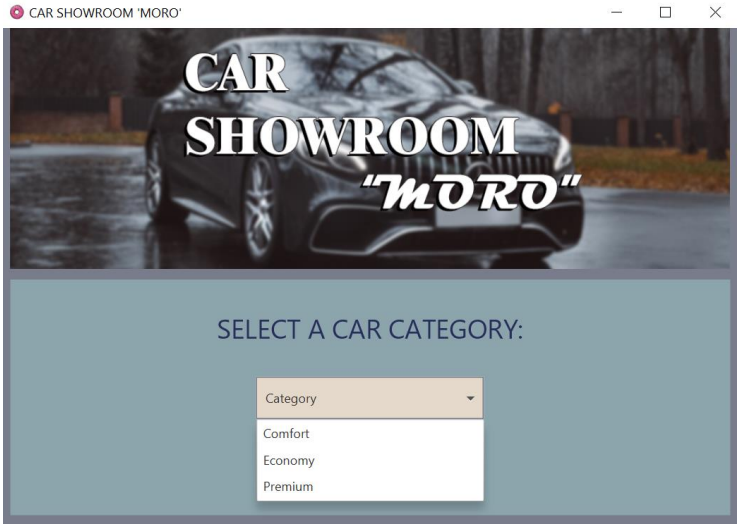


Рисунок 16-Окно с выбором категории

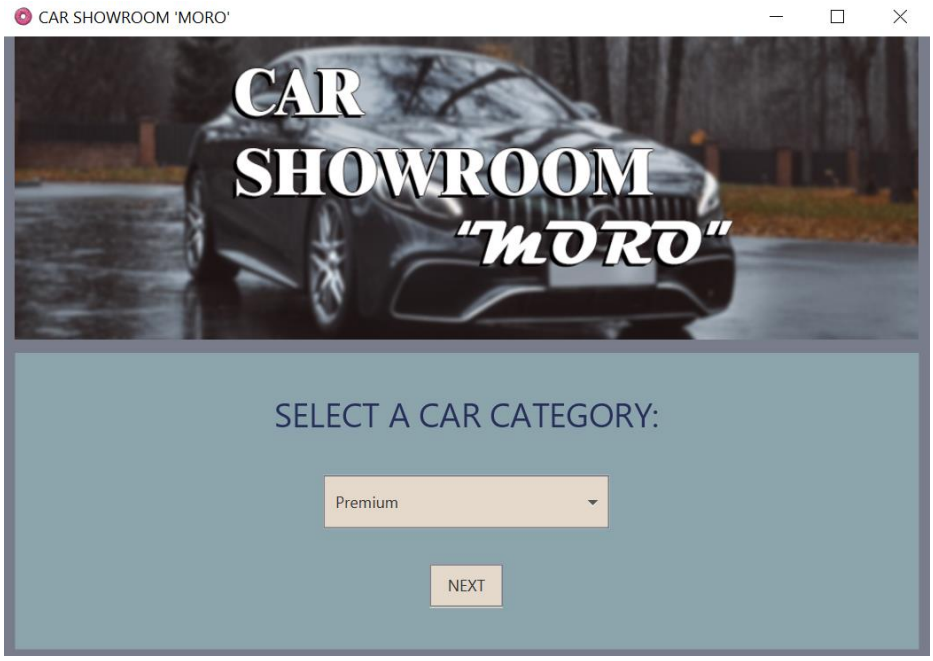


Рисунок 17-Выбираем, например premium

После нажатия на кнопку NEXT клиент видит окно с выбором автомобиля:

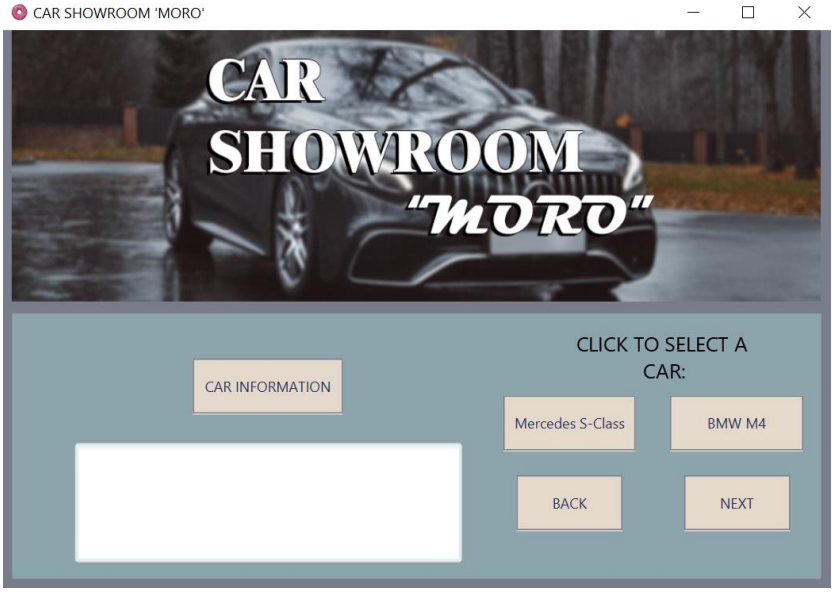


Рисунок 18-Окно с выбором автомобиля

Нажатие на кнопку CAR INFORMATION показывает информацию об автомобилях, которые входят в эту категорию.

Также выбрав автомобиль, вторая кнопка с другим автомобилем становится неактивной.

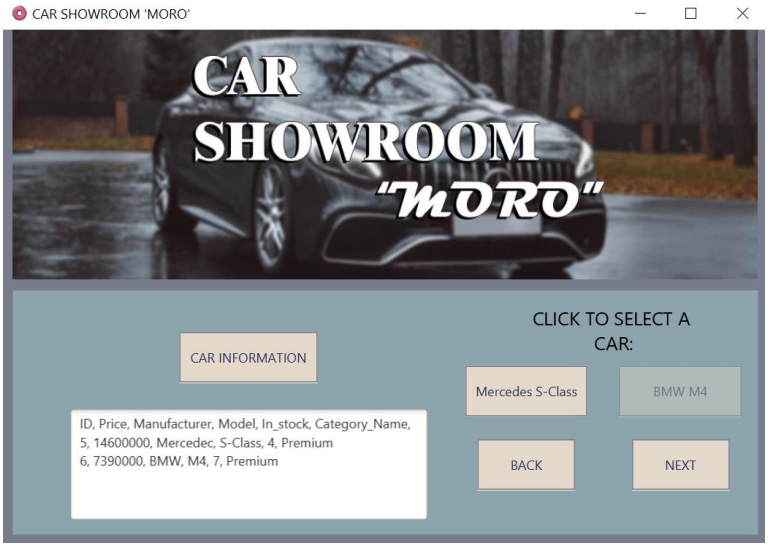


Рисунок 19-Окно при нажатии кнопок CAR INFORMATION и выбранным автомобилем

Далее клиент нажимает на кнопку NEXT и ему выходит окно с выбором дополнительных услуг:

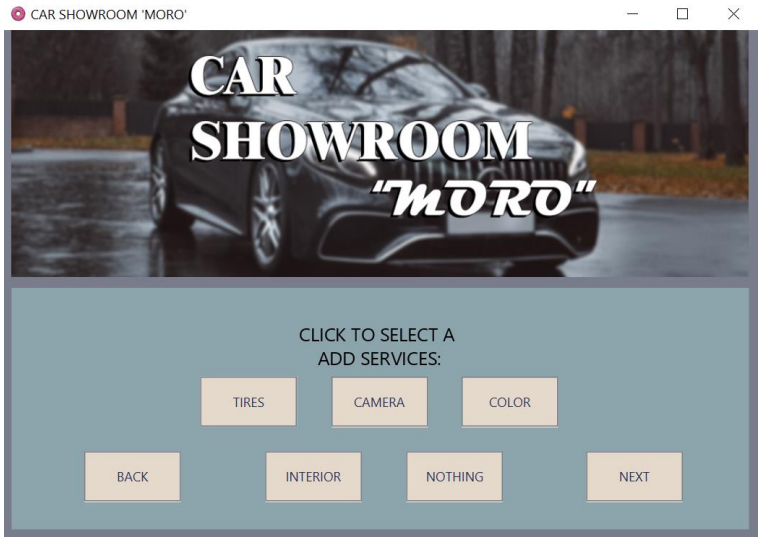


Рисунок 20-Окно с выбором дополнительных услуг

Когда пользователь выбирает какую-то из доп. услуг, то остальные кнопки становятся неактивные:

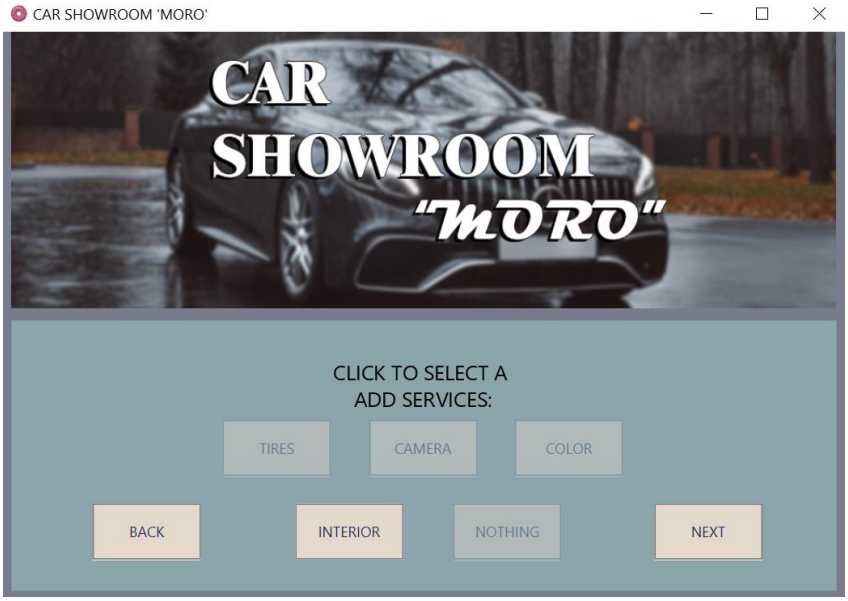


Рисунок 21-Пример окна с выбранной доп. услугой

При нажатии на кнопку NEXT клиент видит окно с подтверждением или отклонением своей покупки:

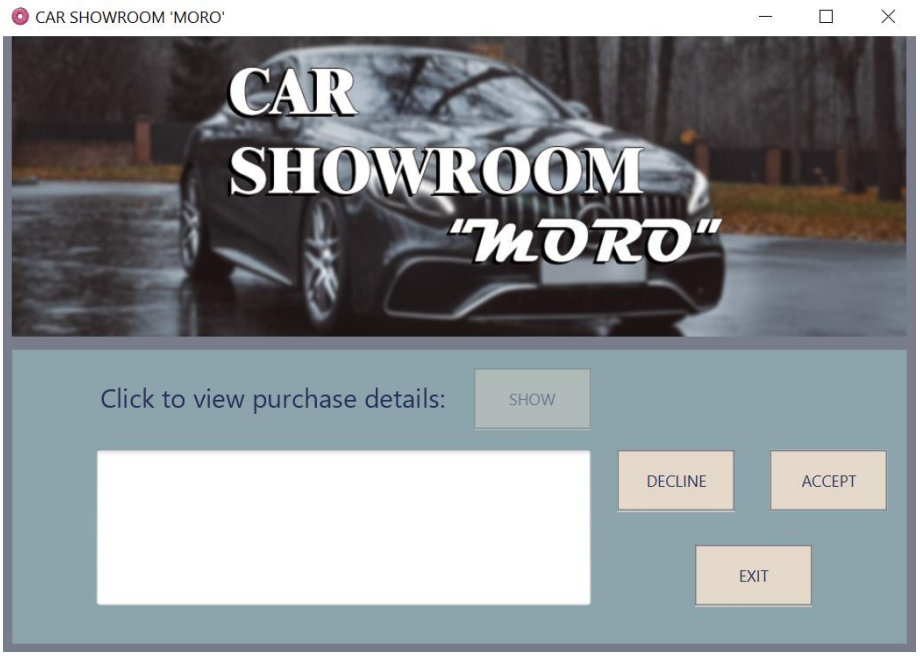


Рисунок 22-Окно с подтверждением или отклонением своей покупки

Если клиент нажимает на кнопку ACCEPT, то ему становится доступна кнопка SHOW, после нажатия на которую он может посмотреть детали своей покупки:

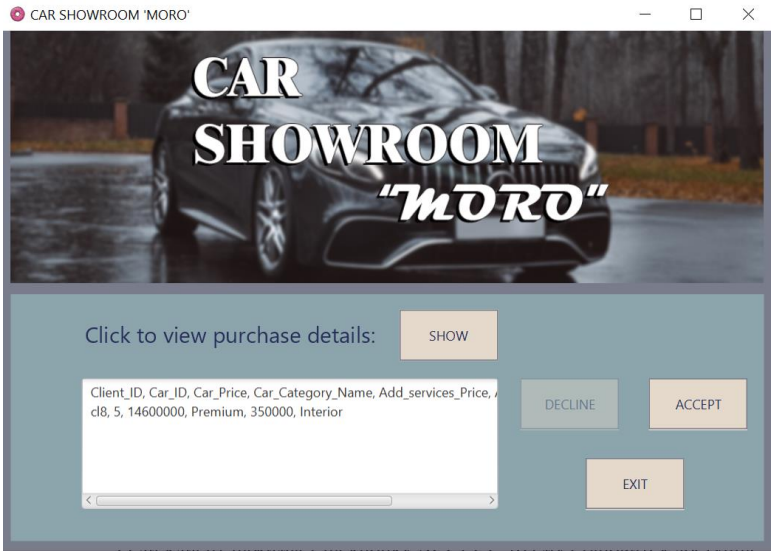


Рисунок 23-Окно с подтвержденным заказом

Далее клиент нажимает на кнопку EXIT, которая перебрасывает его в главное меню. Там он уже может выйти из программы.

7.2 Администратор

После нажатия кнопки ADMINISTRATOR в главном меню, пользователя перебрасывает на окно с входом:

Если оставить все поля пустыми или вводить неверные данные, то будет анимация тряски блоков, которая показывает пользователю, что введено что-то не так.

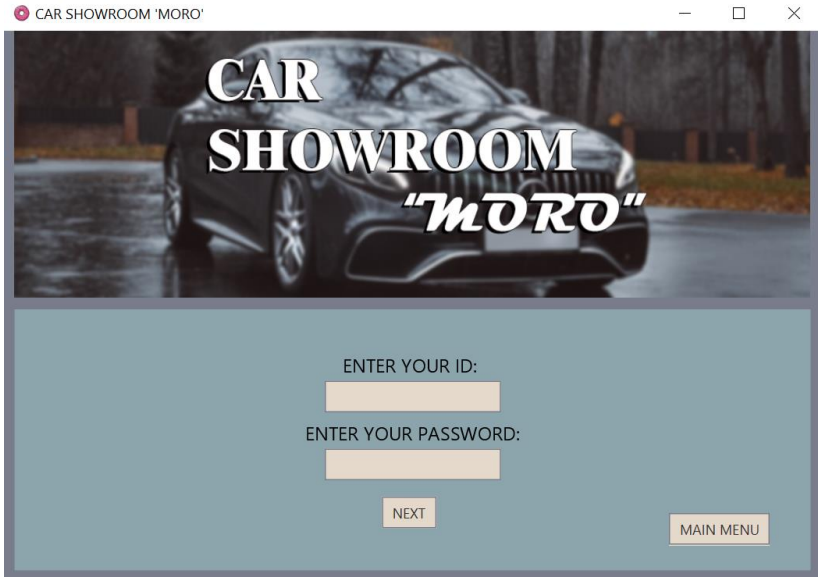


Рисунок 24-Окно входа администратора

Далее сотрудник вводит свой айди и пароль.

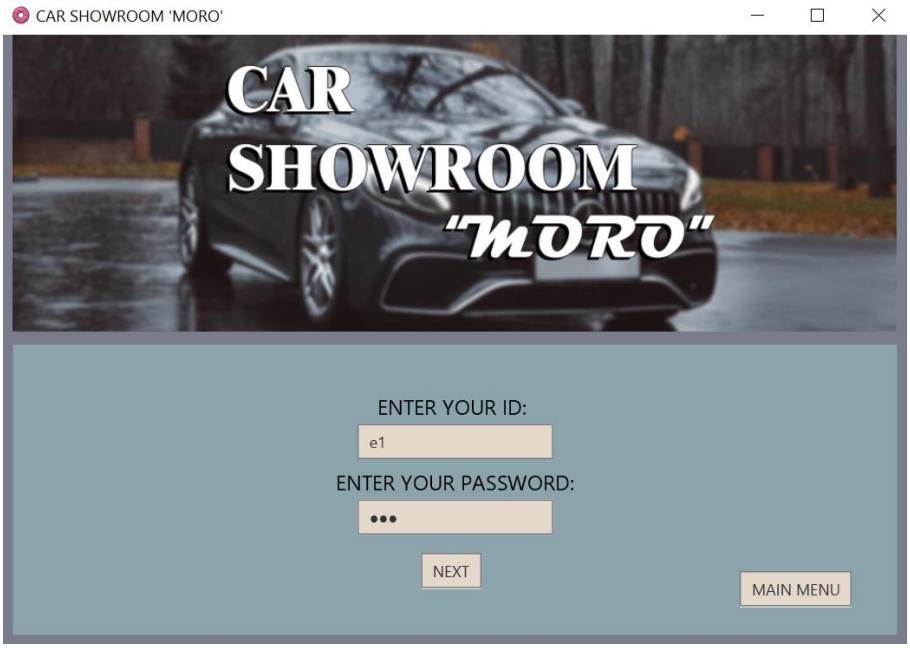


Рисунок 25-Окно с введенными данными

После нажатия на кнопку NEXT, сотрудник видит следующее окно:

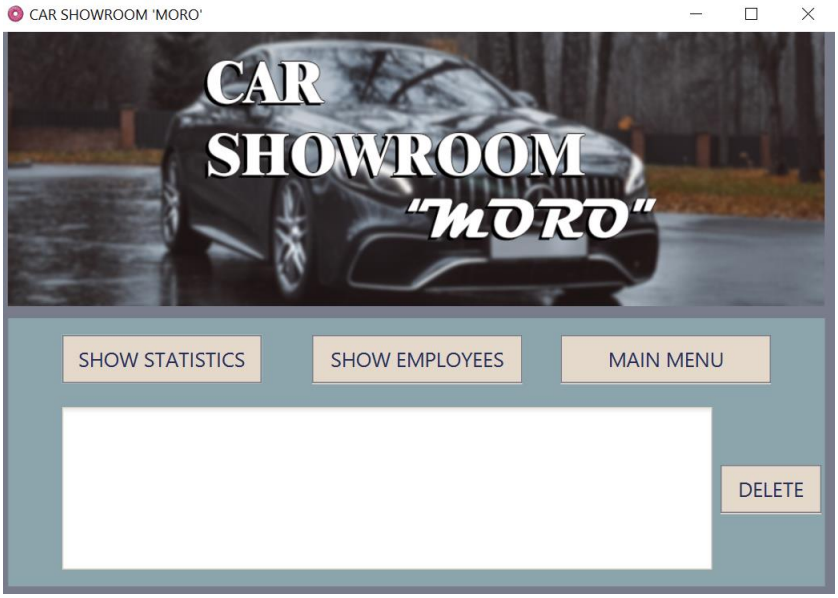


Рисунок 26-Окно администратора

При нажатии на кнопку SHOW STATISTICS сотрудник может посмотреть статистику продаж.

При нажатии на кнопку SHOW EMPLOYEES сотрудник может посмотреть информацию обо всех сотрудниках автосалона.

Нажатие на кнопку DELETE очищает полностью сстатистику продаж автосалона.

Кнопка MAIN MENU переносит сотрудника в главное меню.

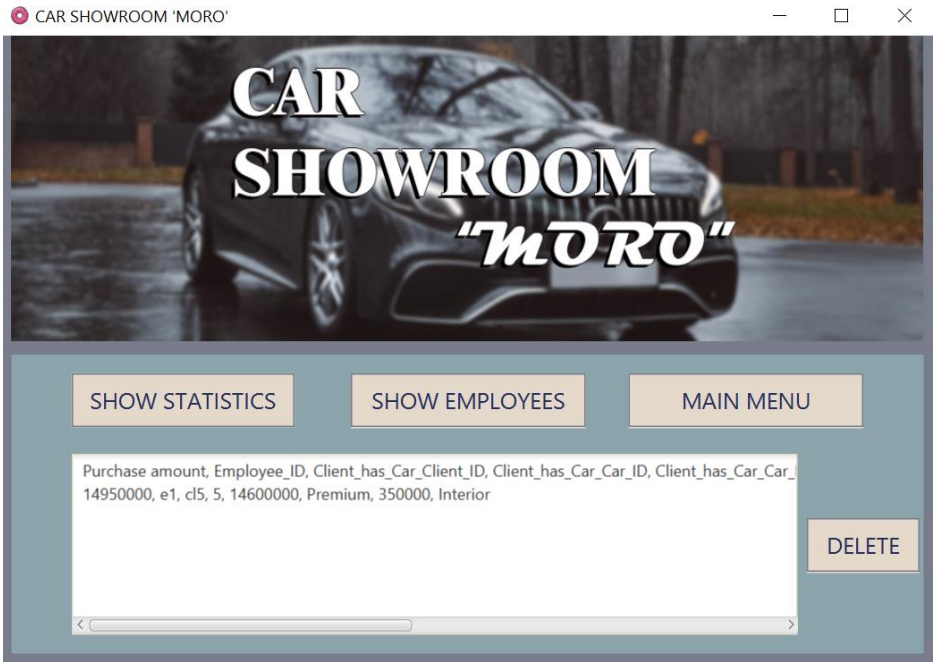


Рисунок 27-После нажатия на кнопку SHOW STATISTICS

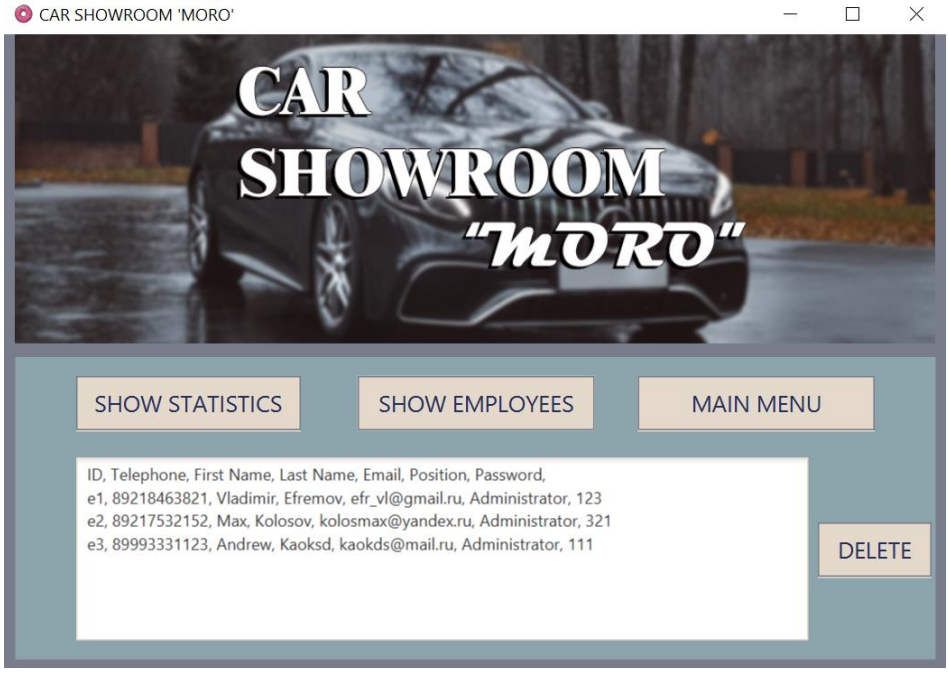


Рисунок 28-После нажатия на кнопку SHOW EMPLOYEES

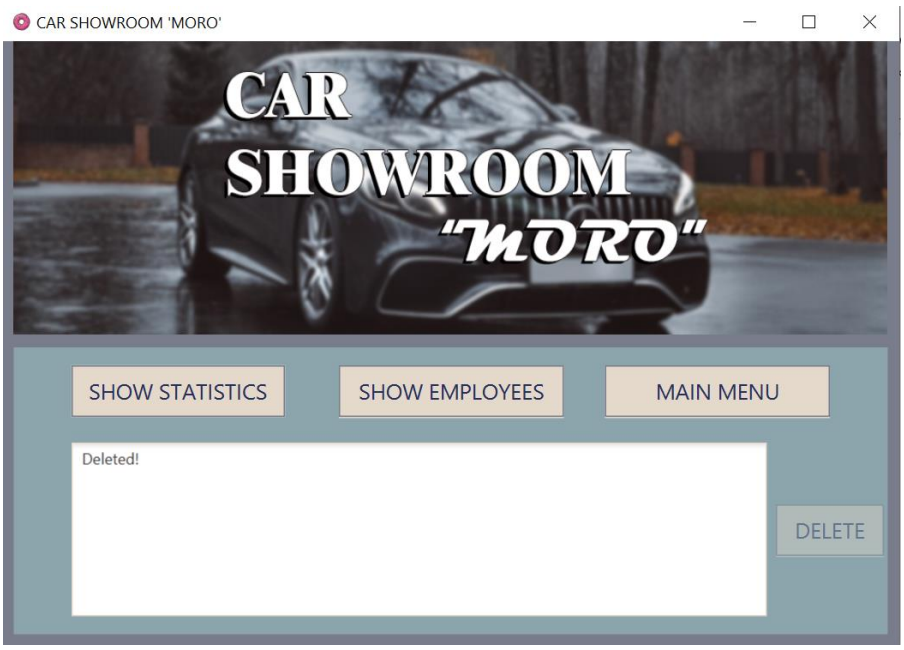


Рисунок 29-После нажатия на кнопку DELETE

Далее кнопка DELETE становится неактивной, т. к. и так уже очистили полностью статистику продаж и удалять там больше нечего.

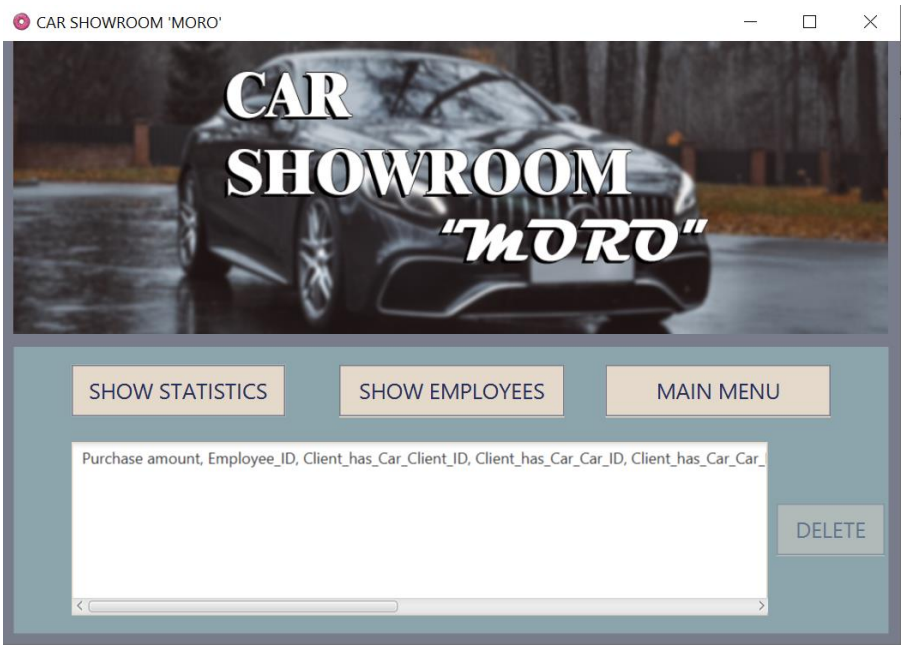


Рисунок 30-Статистика продаж очистилась

8 Тестирование

В данном пункте будет проведено тестирование программы:

1. Проверка реакции приложения на ввод некорректных данных клиентом;
2. Проверка реакции приложения на ввод некорректных данных администратором.

При некорректном вводе данных пользователь видит анимацию тряски блоков. Показать на скриншотах этого невозможно.

8.1 Клиент

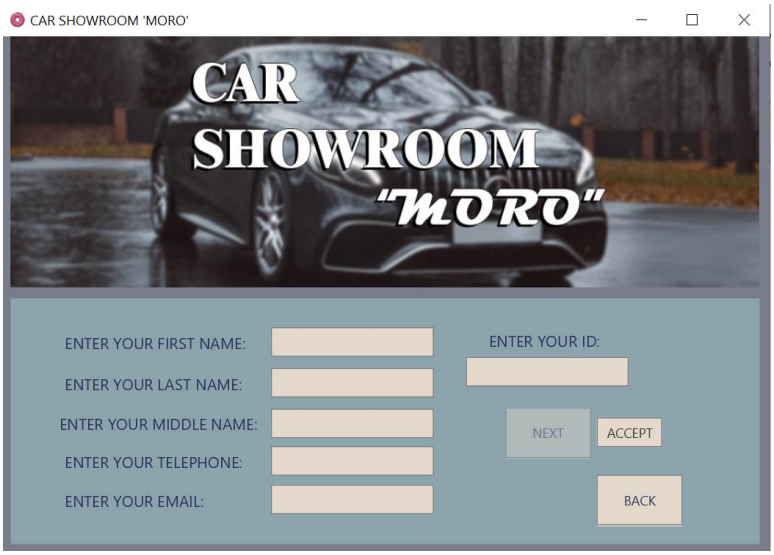


Рисунок 31-При пустом вводе данных

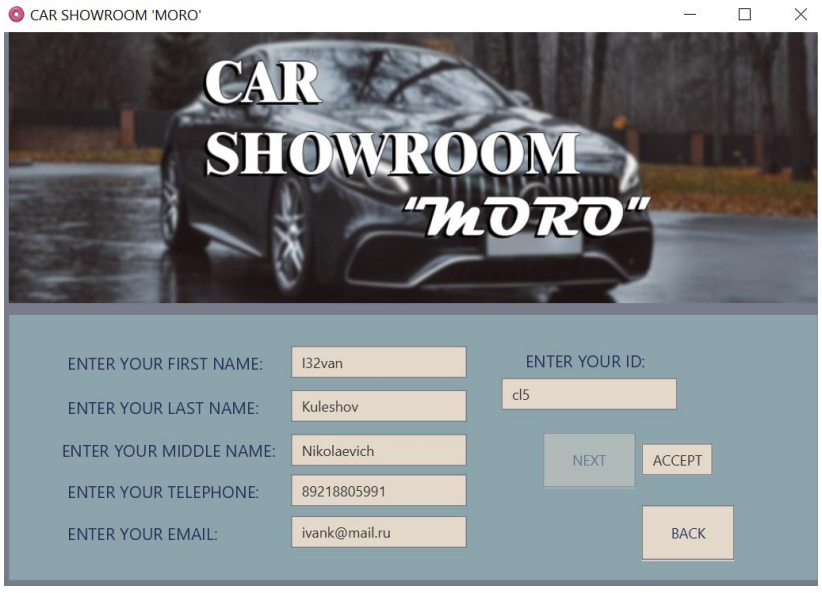


Рисунок 32-При некорректном вводе фамилии

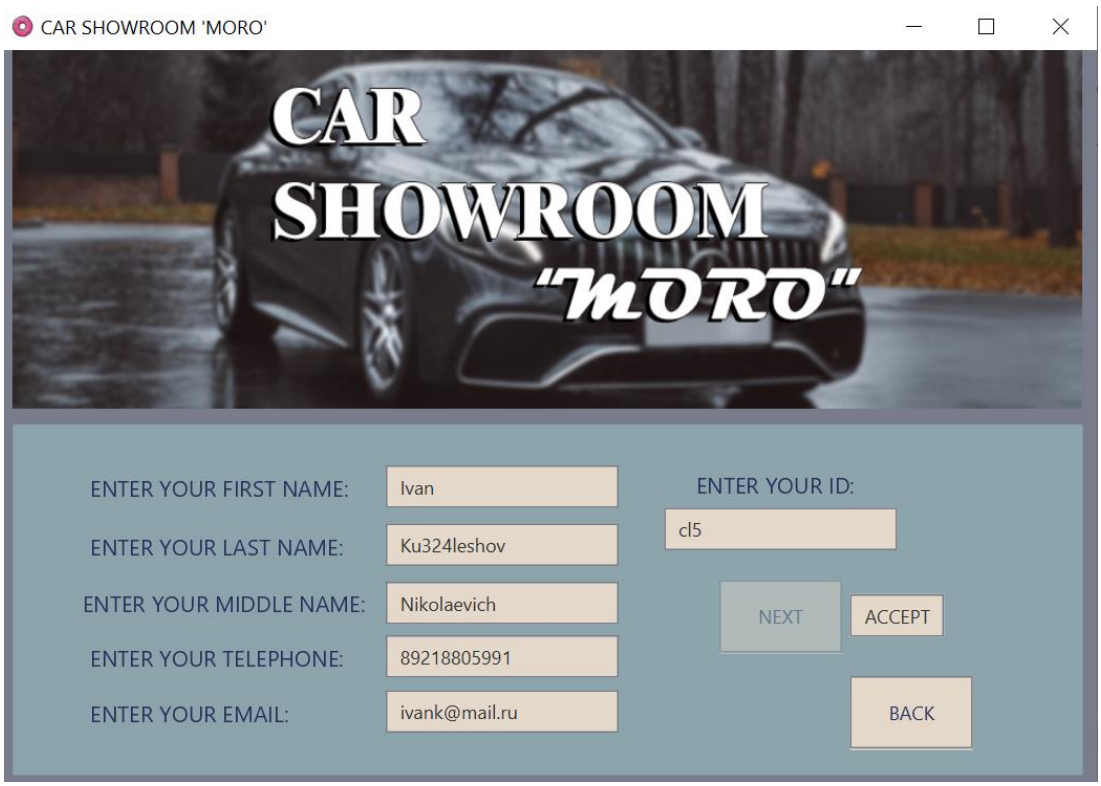


Рисунок 33-При некорректном вводе фамилии

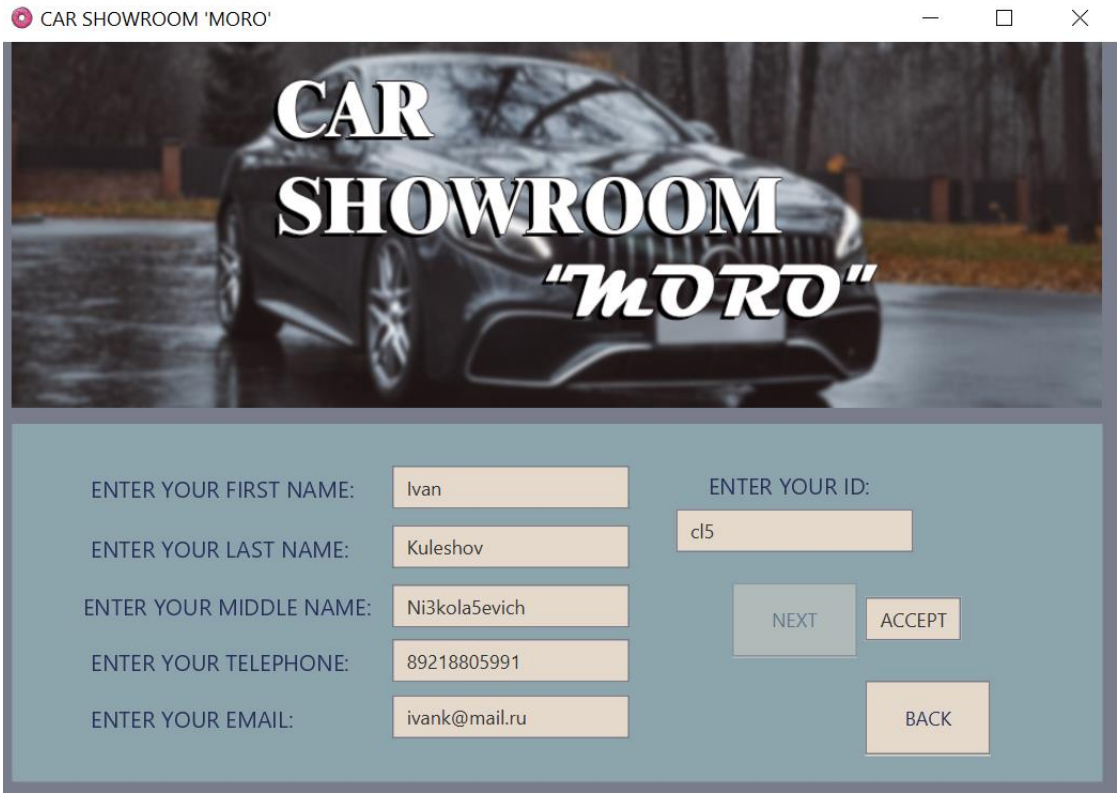


Рисунок 34-При некорректном вводе отчества

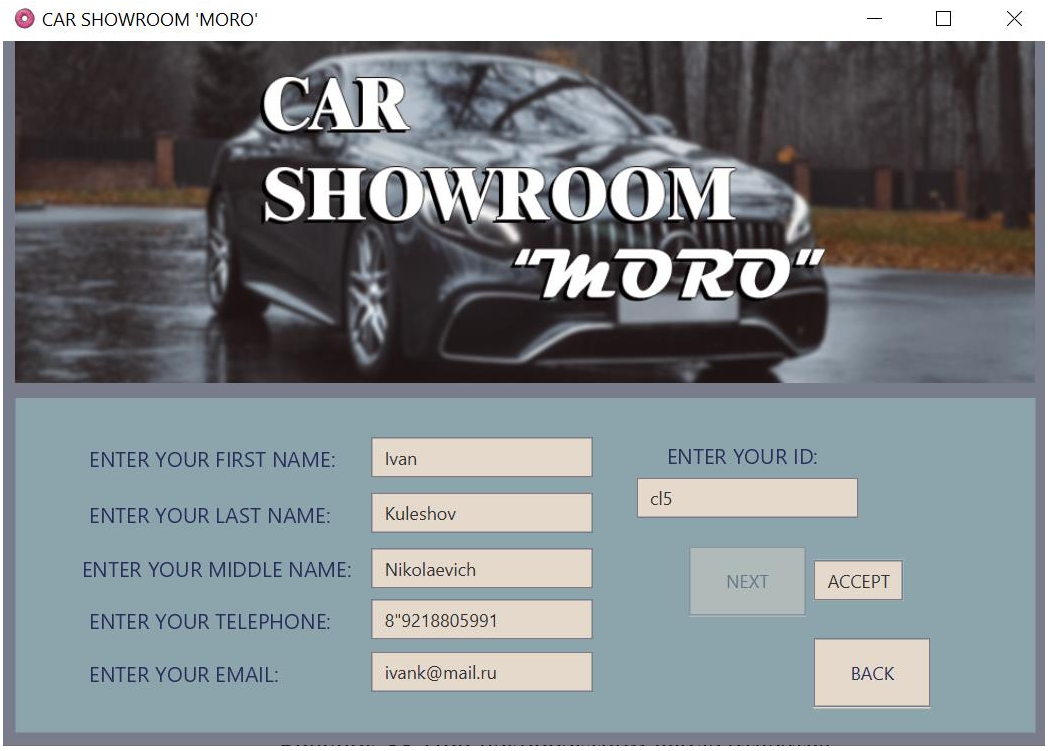


Рисунок 35-При некорректном вводе номера

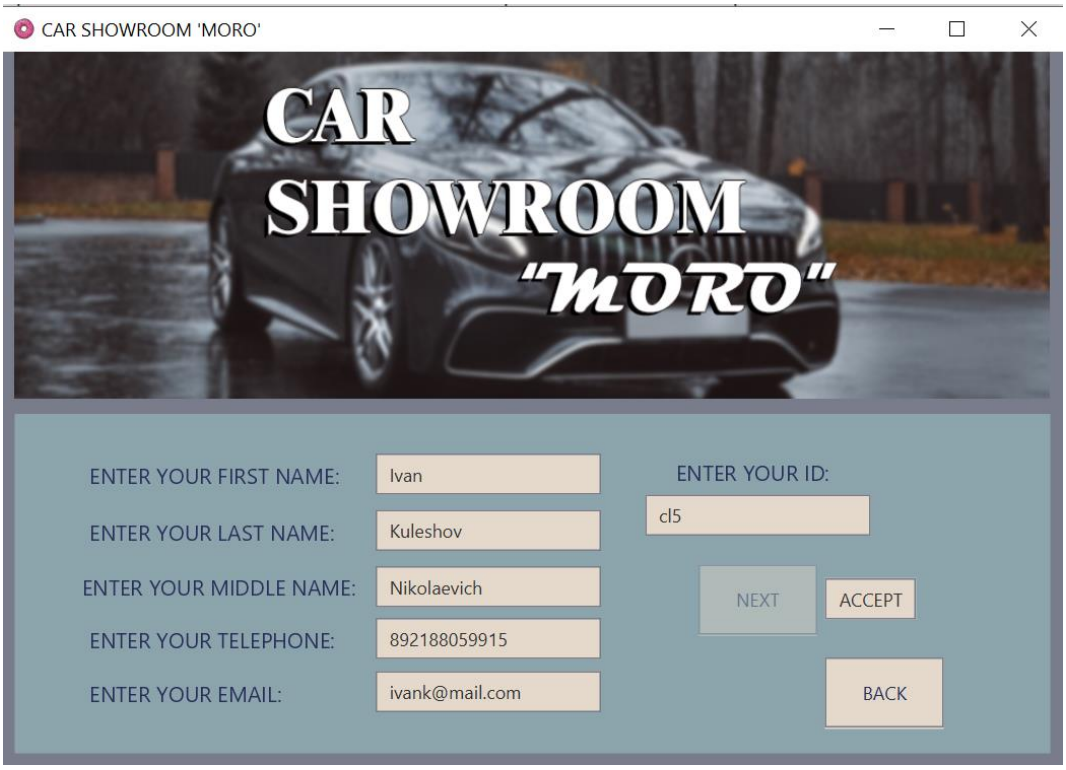


Рисунок 36-При вводе номера с 12 цифрами

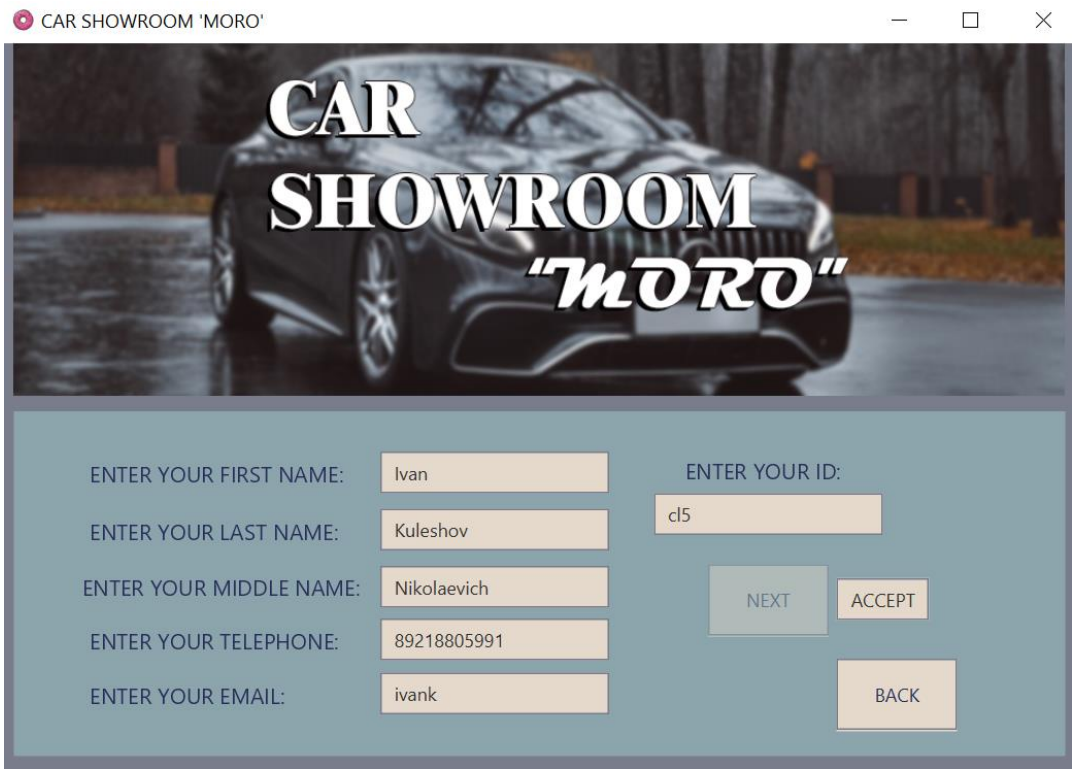


Рисунок 37-Некорректный ввод почты

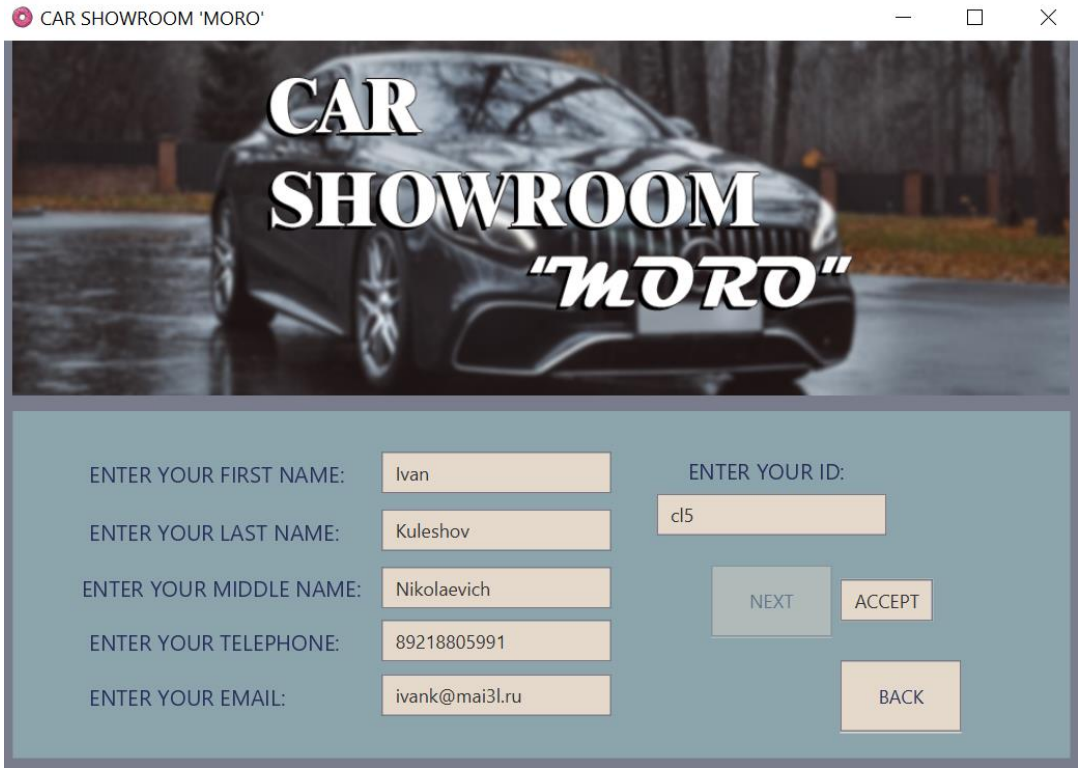


Рисунок 38-Некорректный ввод почты

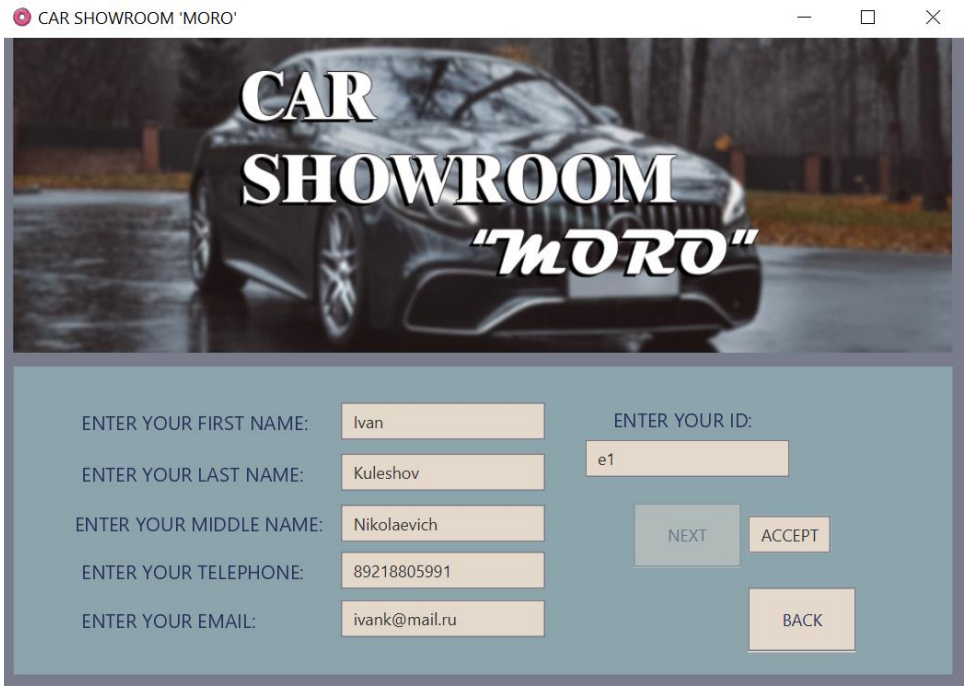


Рисунок 39-Некорректный ввод айди

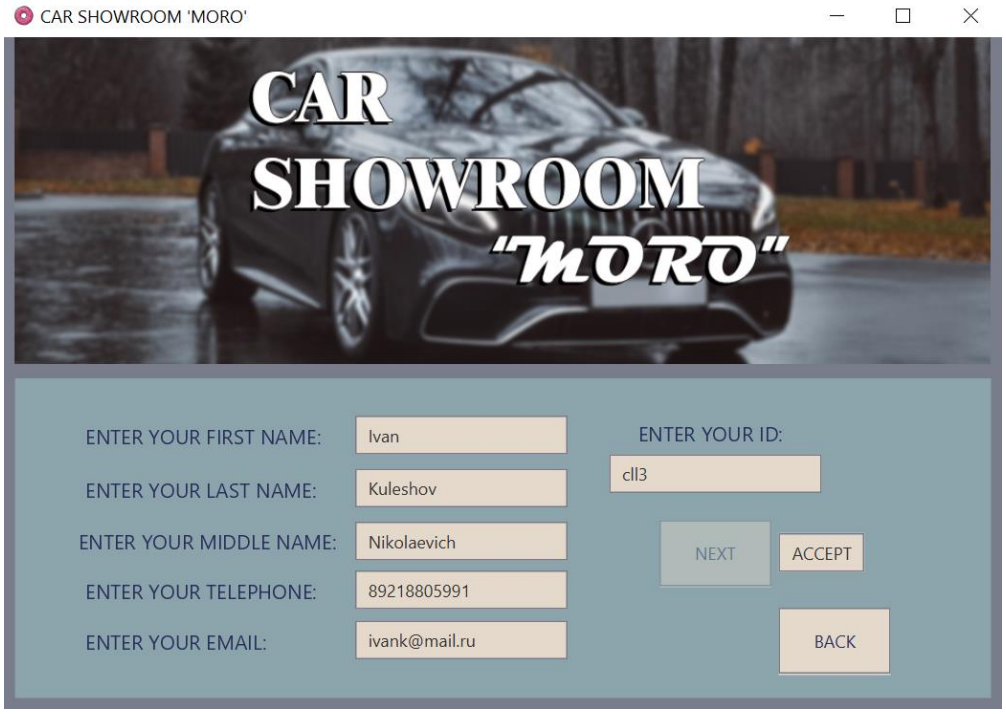


Рисунок 40-Некорректный ввод айди

При каждом некорректном вводе пользователь видит анимацию тряски блоков.

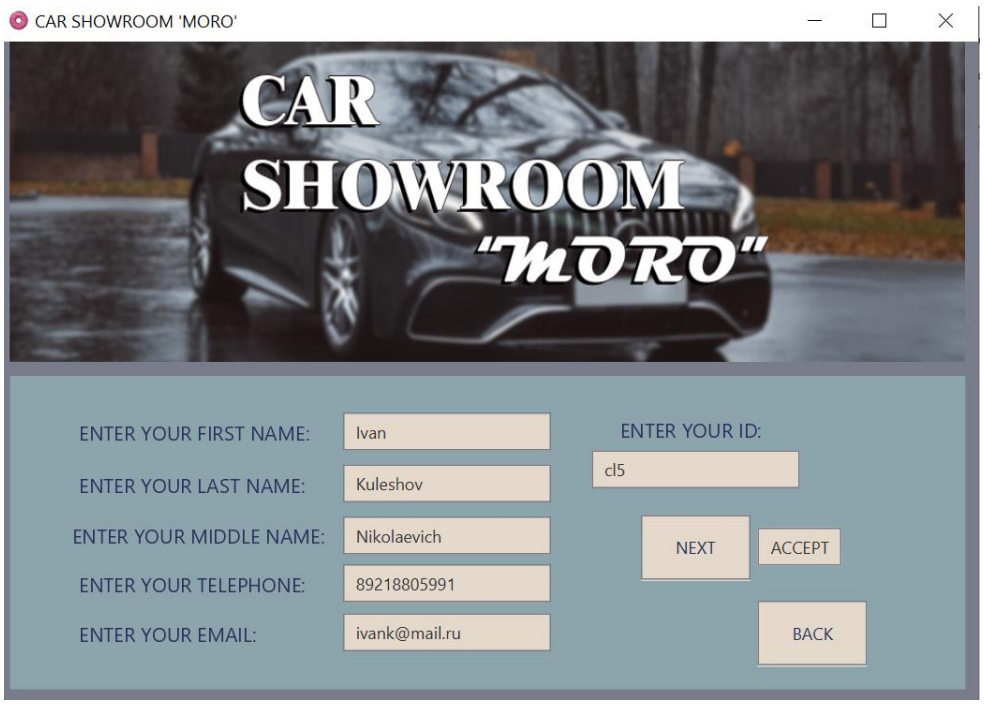


Рисунок 41-При корректном вводе данных пользователю становится доступна кнопка NEXT

8.2 Администратор

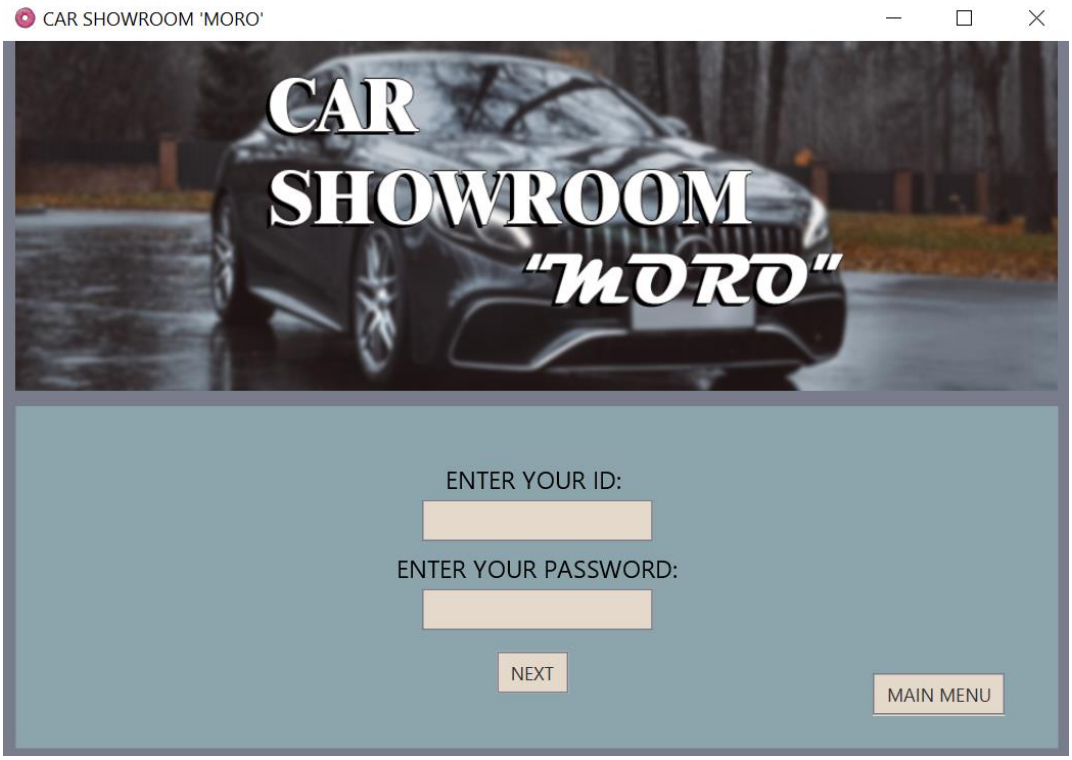


Рисунок 42-Пустые поля в окне входа администратора

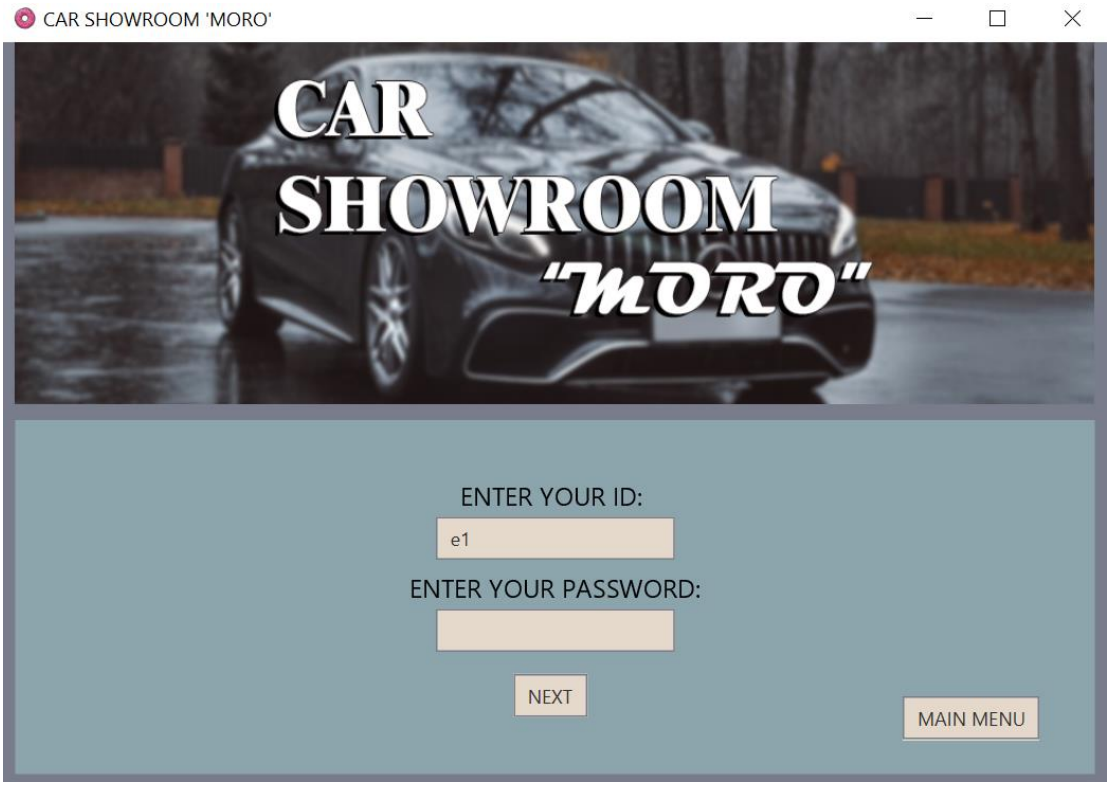


Рисунок 43-Проверка на вход администратора

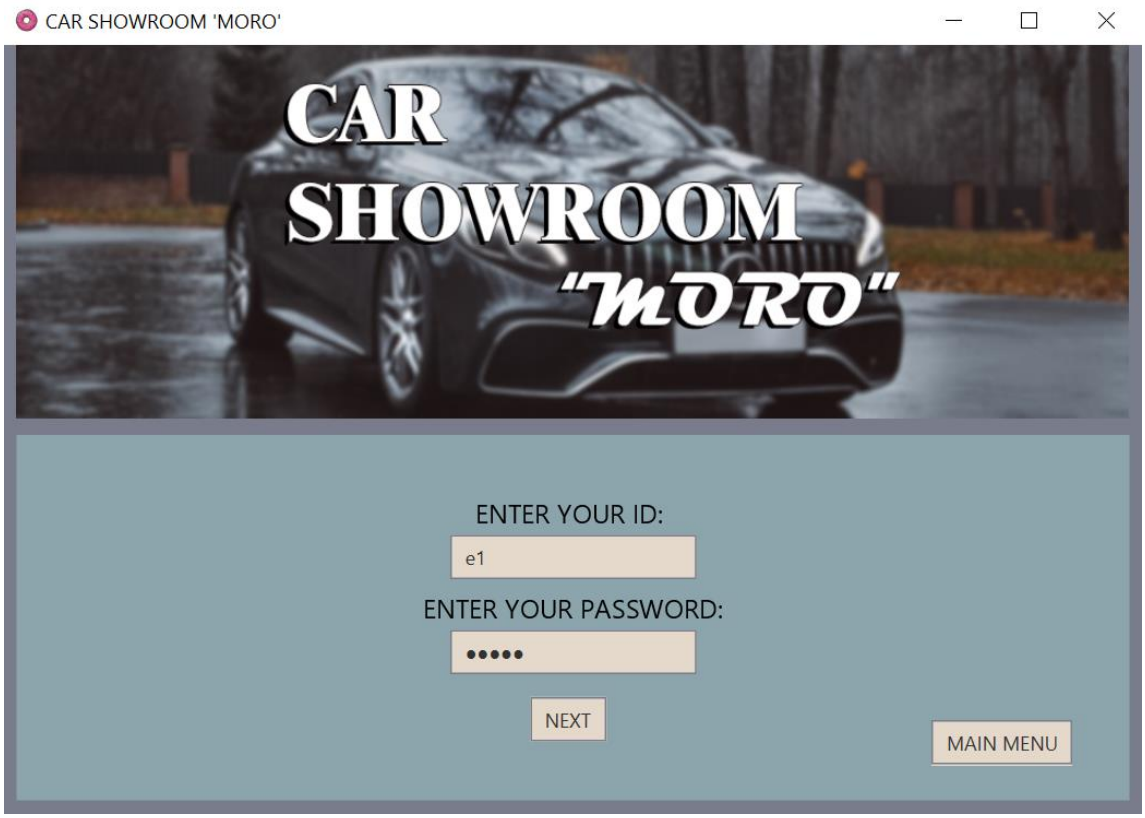


Рисунок 44- Проверка на вход администратора

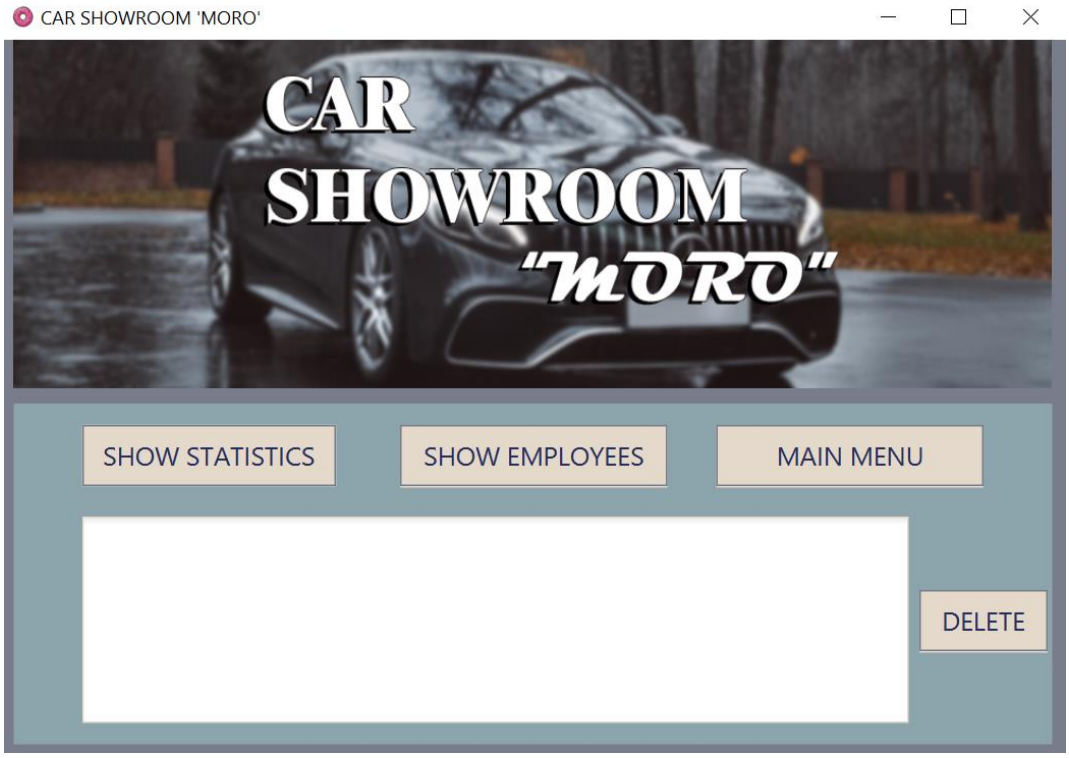


Рисунок 45-При верном вводе данных переходит в окно для администратора

Все тесты были пройдены успешно.

Заключение

В результате выполнения данной работы была успешно описана предметная область, выделены правила и ограничения.

После построения логической модели, в которой я определил основные сущности, атрибуты, первичные и уникальные ключи, связи между сущностями, был выполнен переход к реляционной модели, в которой видно используемые типы данных, разрешил связи многие-ко-многим, описал ограничения.

В программы MySQL WorkBench была создана база данных, для работы с которой создано клиентское приложение с использованием IDE Eclipse и JavaFX. Для построения пользовательского интерфейса использовался плагин Scene Builder.

После написания программы я протестировал ее, тесты приложены в пункте 6. Все тесты были пройдены успешно.

Приложения

Приложение 1

```
-- MySQL dump 10.13  Distrib 8.0.24, for Win64 (x86_64)
--
-- Host: localhost    Database: project_schema
-- ------------------------------------------------------
-- Server version    8.0.24

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
```

```
-- Table structure for table `add_services`
--

DROP TABLE IF EXISTS `add_services`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `add_services` (
  `ID` varchar(5) NOT NULL,
  `Price` varchar(45) NOT NULL,
  `Name` varchar(45) NOT NULL,
  PRIMARY KEY (`Price`,`Name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `add_services`
--

LOCK TABLES `add_services` WRITE;
/*!40000 ALTER TABLE `add_services` DISABLE KEYS */;
INSERT INTO `add_services` VALUES
('5','0','Nothing'),('3','15000','Color'),('2','150000','Camera'),('4','350000','Interior'),('
1','80000','Tires');
/*!40000 ALTER TABLE `add_services` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `car`
--
```

```
DROP TABLE IF EXISTS `car`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `car` (
  `ID` varchar(5) NOT NULL,
  `Price` int NOT NULL,
  `Manufacturer` varchar(20) NOT NULL,
  `Model` varchar(15) NOT NULL,
  `In_stock` int NOT NULL,
  `Category_Name` varchar(20) NOT NULL,
  PRIMARY KEY (`ID`,`Price`,`Category_Name`),
  UNIQUE KEY `Price_UNIQUE` (`Price`),
  KEY `fk_Car_Category_idx` (`Category_Name`),
  CONSTRAINT `fk_Car_Category` FOREIGN KEY (`Category_Name`) REFERENCES `category` (`Name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;


--
-- Dumping data for table `car`
--


LOCK TABLES `car` WRITE;
/*!40000 ALTER TABLE `car` DISABLE KEYS */;
INSERT INTO `car` VALUES ('1',900000,'Hyundai','Solaris',10,'Economy'),('2',957000,'Vlokswagen','Polo',9,'Economy'),('3',1700000,'KIA','K5',10,'Comfort'),('4',1473000,'Skoda','Octavia',11,'Comfort'),('5',14600000,'Mercedec','S-Class',4,'Premium'),('6',7390000,'BMW','M4',7,'Premium');
/*!40000 ALTER TABLE `car` ENABLE KEYS */;
```

```
UNLOCK TABLES;

--
-- Table structure for table `category`
--

DROP TABLE IF EXISTS `category`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `category` (
  `ID` int NOT NULL,
  `Name` varchar(20) NOT NULL,
  PRIMARY KEY (`Name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `category`
--

LOCK TABLES `category` WRITE;
/*!40000 ALTER TABLE `category` DISABLE KEYS */;
INSERT INTO `category` VALUES
(2,'Comfort'),(1,'Economy'),(3,'Premium');
/*!40000 ALTER TABLE `category` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `client`
--
```

```
DROP TABLE IF EXISTS `client`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `client` (
  `ID` varchar(5) NOT NULL,
  `Telephone` varchar(20) NOT NULL,
  `First Name` varchar(50) NOT NULL,
  `Middle Name` varchar(50) NOT NULL,
  `Last Name` varchar(50) NOT NULL,
  `Email` varchar(50) NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE KEY `Telephone_UNIQUE` (`Telephone`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `client`
--

LOCK TABLES `client` WRITE;
/*!40000 ALTER TABLE `client` DISABLE KEYS */;
INSERT INTO `client` VALUES
('cl1','89319218732','Vladislav','Vladimirovich','Shinshilin','shinshila_vlad@gmail.
ru'),('cl2','89219903215','Alexander','Ahmatovich','Magamedov','maga_ne@mail.ru
'),('cl3','89316742346','Egor','Petrovich','Kozlov','kozel_ep@yandex.ru'),('cl4','8921
8893213','Ali','Yakubich','Han','camry_3_5@gmail.ru'),('cl5','89218805991','Ivan','
Nikolaevich','Kuleshov','ivank@mail.ru');
/*!40000 ALTER TABLE `client` ENABLE KEYS */;
UNLOCK TABLES;
```

```
--
-- Table structure for table `client_has_car`
--

DROP TABLE IF EXISTS `client_has_car`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `client_has_car` (
  `Client_ID` varchar(5) NOT NULL,
  `Car_ID` varchar(5) NOT NULL,
  `Car_Price` int NOT NULL,
  `Car_Category_Name` varchar(20) NOT NULL,
  `Add_services_Price` varchar(45) NOT NULL,
  `Add_services_Name` varchar(45) NOT NULL,
  PRIMARY KEY (`Client_ID`,`Car_ID`,`Car_Price`,`Car_Category_Name`,`Add_services_Price`,`Add_services_Name`),
  KEY `fk_Client_has_Car_Car1_idx` (`Car_ID`,`Car_Price`,`Car_Category_Name`),
  KEY `fk_Client_has_Car_Client1_idx` (`Client_ID`),
  KEY `fk_Client_has_Car_Add_services1_idx` (`Add_services_Price`,`Add_services_Name`),
  CONSTRAINT `fk_Client_has_Car_Add_services1` FOREIGN KEY (`Add_services_Price`, `Add_services_Name`) REFERENCES `add_services` (`Price`, `Name`),
  CONSTRAINT `fk_Client_has_Car_Car1` FOREIGN KEY (`Car_ID`, `Car_Price`, `Car_Category_Name`) REFERENCES `car` (`ID`, `Price`, `Category_Name`),
```

```
  CONSTRAINT `fk_Client_has_Car_Client1` FOREIGN KEY (`Client_ID`) REFERENCES `client` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `client_has_car`
--

LOCK TABLES `client_has_car` WRITE;
/*!40000 ALTER TABLE `client_has_car` DISABLE KEYS */;
/*!40000 ALTER TABLE `client_has_car` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `contract`
--

DROP TABLE IF EXISTS `contract`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `contract` (
  `Purchase amount` varchar(45) NOT NULL,
  `Employee_ID` varchar(5) NOT NULL,
  `Client_has_Car_Client_ID` varchar(5) NOT NULL,
  `Client_has_Car_Car_ID` varchar(5) NOT NULL,
  `Client_has_Car_Car_Price` int NOT NULL,
  `Client_has_Car_Car_Category_Name` varchar(20) NOT NULL,
  `Client_has_Car_Add_services_Price` varchar(45) NOT NULL,
  `Client_has_Car_Add_services_Name` varchar(45) NOT NULL,
```

```
  PRIMARY KEY (`Purchase amount`,`Client_has_Car_Client_ID`,`Client_has_Car_Car_ID`,`Client_has_Car_Car_Price`,`Client_has_Car_Car_Category_Name`,`Client_has_Car_Add_services_Price`,`Client_has_Car_Add_services_Name`),
  KEY `fk_Contract_Employee1_idx` (`Employee_ID`),
  KEY `fk_Contract_Client_has_Car1_idx` (`Client_has_Car_Client_ID`,`Client_has_Car_Car_ID`,`Client_has_Car_Car_Price`,`Client_has_Car_Car_Category_Name`,`Client_has_Car_Add_services_Price`,`Client_has_Car_Add_services_Name`),
  CONSTRAINT `fk_Contract_Client_has_Car1` FOREIGN KEY (`Client_has_Car_Client_ID`, `Client_has_Car_Car_ID`, `Client_has_Car_Car_Price`, `Client_has_Car_Car_Category_Name`, `Client_has_Car_Add_services_Price`, `Client_has_Car_Add_services_Name`) REFERENCES `client_has_car` (`Client_ID`, `Car_ID`, `Car_Price`, `Car_Category_Name`, `Add_services_Price`, `Add_services_Name`),
  CONSTRAINT `fk_Contract_Employee1` FOREIGN KEY (`Employee_ID`) REFERENCES `employee` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `contract`
--

LOCK TABLES `contract` WRITE;
/*!40000 ALTER TABLE `contract` DISABLE KEYS */;
/*!40000 ALTER TABLE `contract` ENABLE KEYS */;
UNLOCK TABLES;

--
```

```
-- Table structure for table `employee`
--

DROP TABLE IF EXISTS `employee`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `employee` (
  `ID` varchar(5) NOT NULL,
  `Telephone` varchar(20) NOT NULL,
  `First Name` varchar(50) NOT NULL,
  `Last Name` varchar(50) NOT NULL,
  `Email` varchar(50) NOT NULL,
  `Position` varchar(50) NOT NULL,
  `Password` varchar(45) NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE KEY `Telephone_UNIQUE` (`Telephone`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `employee`
--

LOCK TABLES `employee` WRITE;
/*!40000 ALTER TABLE `employee` DISABLE KEYS */;
INSERT INTO `employee` VALUES
('e1','89218463821','Vladimir','Efremov','efr_vl@gmail.ru','Administrator','123'),('e
2','89217532152','Max','Kolosov','kolosmax@yandex.ru','Administrator','321'),('e3'
,'89993331123','Andrew','Kaoksd','kaokds@mail.ru','Administrator','111');
/*!40000 ALTER TABLE `employee` ENABLE KEYS */;
```

```
UNLOCK TABLES;

--
-- Dumping events for database 'project_schema'
--


--
-- Dumping routines for database 'project_schema'
--
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET
FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET
CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET
CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET
COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2021-05-19 23:05:48
```

Приложение 2

Класс Shake:

```
package animations;

import javafx.util.Duration;
import javafx.animation.TranslateTransition;
import javafx.scene.Node;

public class Shake {
	private TranslateTransition TT;

	public Shake(Node node) {
		TT = new TranslateTransition(Duration.millis(70), node);
		TT.setFromX(-10f);
		TT.setByX(10f);
		TT.setCycleCount(3);
		TT.setAutoReverse(true);
	}

	public void playAnim() {
		TT.playFromStart();
	}
}
```

Класс Admin_in:

```
package application;

import java.net.URL;
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import animations.Shake;

public class Admin_in extends connect {
        @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private Button admin_stat;

    @FXML
    private PasswordField pass_adm;

    @FXML
    private TextField login_adm;

    @FXML
```

```
        private Button main_menu;

        @FXML
        void initialize() {
            admin_stat.setOnAction(event ->{
              String loginText = login_adm.getText().trim();
              String loginPass = pass_adm.getText().trim();
              String pos = Const.EMPLOYEE_POSITION;
              if (!loginText.equals("") && !loginPass.equals(""))
                                try {
                                        loginAdm(loginText, loginPass);
                                        admin_id = loginText;
                                        DataBaseHandler dbHandler2 = new
DataBaseHandler();
                                dbHandler2.contractIns();
                                        //System.out.println("Good!!!");
                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        else {
                                Shake userLogAnim = new Shake(login_adm);
                                Shake userPassAnim = new Shake(pass_adm);
                                userLogAnim.playAnim();
                                userPassAnim.playAnim();
                                //System.out.println("Empty!");
                        }
            });

            main_menu.setOnAction(event ->{
                    try {
```

```
                menu();
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }

    private void loginAdm(String loginText, String loginPass) throws Exception {
        DataBaseHandler dbHandler = new DataBaseHandler();
        Employee employee = new Employee();
        employee.setId(loginText);
        employee.setPassword(loginPass);
        ResultSet resultSet= dbHandler.getAdm(employee);

        int counter = 0;

        try {
            while(resultSet.next()) {
                counter++;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        if (counter >=1) {
            admin_statistic();
            //System.out.println("Succes!");
        }
        else {
```

```
            }
        }

        public void admin_statistic () throws Exception {
            Parent root = FXMLLoader.load(getClass().getResource("Administrator_statistic.fxml"));
            Stage window=(Stage) admin_stat.getScene().getWindow();
            window.setScene(new Scene(root));
        }

        public void menu () throws Exception {
            Parent root = FXMLLoader.load(getClass().getResource("First_in.fxml"));
            Stage window=(Stage) main_menu.getScene().getWindow();
            window.setScene(new Scene(root));
        }
    }
```

Класс Admin_stat:

```
package application;

import java.net.URL;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.ResourceBundle;

import animations.Shake;
import javafx.event.EventHandler;
```

```
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.stage.Stage;

public class Admin_stat {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private Button show_stat;

    @FXML
    private TextArea stat_full;

    @FXML
    private Button show_emp;

    @FXML
    private Button main_menu;

    @FXML
    private Button delete;
```

```
@FXML
void initialize() {
    delete.setDisable(false);
    show_emp.setOnAction(event ->{
    try {
            DataBaseHandler dbh = new DataBaseHandler();
            ResultSet result = dbh.getTable("employee");
                    stat_full.setText(showData(result));
                } catch (SQLException e) {
                    e.printStackTrace();
                }
       });

        show_stat.setOnAction(event ->{
                try {
                    DataBaseHandler dbh_1 = new
DataBaseHandler();
            ResultSet result = dbh_1.getTable("contract");
                    stat_full.setText(showData(result));
                } catch (SQLException e) {
                    e.printStackTrace();
                }
        });

        main_menu.setOnAction(event ->{
            try {
                    menu();
                } catch (Exception e) {
                    // TODO Auto-generated catch block
```

```
                    e.printStackTrace();
                }
            });

             delete.setOnAction(event ->{
                    DataBaseHandler dbHandler=new DataBaseHandler();
                    DataBaseHandler dbHandler1=new DataBaseHandler();
                    DataBaseHandler dbHandler2=new DataBaseHandler();
                        dbHandler.deletePurch();
                    stat_full.setText("Deleted!");
                    dbHandler1.deleteCHC();
                    dbHandler2.plusStock();
                delete.setDisable(true);
            });
        }

        private String showData(ResultSet result) throws SQLException {
            ResultSetMetaData rsmd = result.getMetaData();
            int numberOfColumns = rsmd.getColumnCount();
            String text  = "";
            for (int i = 1; i < numberOfColumns + 1; i++) { //получаем
названия столбцов
                String columnName = rsmd.getColumnName(i);
                text = text + columnName + ", ";
            }
            text = text + "\n";
            while(result.next()) { //получаем данные из строк таблицы

                for(int i =0; i<numberOfColumns; i++) {
                    if(i+1 == numberOfColumns) {;
```

```
                    text = text + result.getString(i+1) + "\n";
                }
                else {
                    text += result.getString(i+1) + ", ";
                }
            }
        }
        return text;
    }

    public void menu () throws Exception {
        Parent root =
FXMLLoader.load(getClass().getResource("First_in.fxml"));
        Stage window=(Stage) main_menu.getScene().getWindow();
        window.setScene(new Scene(root));
    }
}
```

Класс Car_comfort:

```
package application;

import java.net.URL;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
```

```
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.stage.Stage;

public class Car_comfort extends connect {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private TextArea infowind;

    @FXML
    private Button inf;

    @FXML
    private Button kia;

    @FXML
    private Button scoda;

    @FXML
    private Button client_choose_car;

    @FXML
    private Button back;
```

```
@FXML
void initialize() {
    kia.setDisable(false);
    scoda.setDisable(false);
    inf.setOnAction(event ->{
    try {
            DataBaseHandler dbh = new DataBaseHandler();
            ResultSet result = dbh.getTableComfort("car");
            //ResultSet result = dbh.getTable("car");
            infowind.setText(showData(result));
                } catch (SQLException e) {
                    e.printStackTrace();
                }
   });

    kia.setOnAction(event ->{
            car_id="3";
            car_price=1700000;
            scoda.setDisable(true);
   });

    scoda.setOnAction(event ->{
            car_id="4";
            car_price=1473000;
            kia.setDisable(true);
   });

    client_choose_car.setOnAction(event ->{
            try {
                client_adds();
```

```
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
            });

             back.setOnAction(event ->{
                try {
                        back_cat();
                    } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
            });

        }

        private String showData(ResultSet result) throws SQLException {
            ResultSetMetaData rsmd = result.getMetaData();
            int numberOfColumns = rsmd.getColumnCount();
            String text  = "";
            for (int i = 1; i < numberOfColumns + 1; i++) { //получаем названия
столбцов
                String columnName = rsmd.getColumnName(i);
                text = text + columnName + ", ";
            }
            text = text + "\n";
            while(result.next()) { //получаем данные из строк таблицы

                for(int i =0; i<numberOfColumns; i++) {
                    if(i+1 == numberOfColumns) {;
```

```
                text = text + result.getString(i+1) + "\n";
            }
            else {
                text += result.getString(i+1) + ", ";
            }
          }
        }
        return text;
    }

    public void client_adds () throws Exception {
        Parent root =
FXMLLoader.load(getClass().getResource("Client_add_services.fxml"));
        Stage window=(Stage) client_choose_car.getScene().getWindow();
        window.setScene(new Scene(root));
    }

    public void back_cat () throws Exception {
        Parent root =
FXMLLoader.load(getClass().getResource("Client_category.fxml"));
        Stage window=(Stage) back.getScene().getWindow();
        window.setScene(new Scene(root));
    }
}
```

Класс Car_economy:

```
package application;

import java.net.URL;
import java.sql.ResultSet;
```

```
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

public class Car_economy extends connect {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private TextArea inf;

    @FXML
    private Button car_inf;

    @FXML
    private Button client_choose_car;
```

```
@FXML
private Button solaris;

@FXML
private Button polo;

@FXML
private Button back;

@FXML
void initialize() {
    polo.setDisable(false);
    solaris.setDisable(false);
    car_inf.setOnAction(event ->{
    try {
            DataBaseHandler dbh = new DataBaseHandler();
            ResultSet result = dbh.getTableEconomy("car");
            //ResultSet result = dbh.getTable("car");
            inf.setText(showData(result));
                } catch (SQLException e) {
                    e.printStackTrace();
                }
   });

    solaris.setOnAction(event ->{
            car_id="1";
            car_price=900000;
            polo.setDisable(true);
   });
```

```
        polo.setOnAction(event ->{
                car_id="2";
                car_price=957000;
                solaris.setDisable(true);
    });

        client_choose_car.setOnAction(event ->{
                try {
                        client_choose_car();
                } catch (Exception e) {
                        e.printStackTrace();
                }
    });

        back.setOnAction(event ->{
                try {
                        back_cat();
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
    });

}

private String showData(ResultSet result) throws SQLException {
    ResultSetMetaData rsmd = result.getMetaData();
    int numberOfColumns = rsmd.getColumnCount();
    String text  = "";
```

```
            for (int i = 1; i < numberOfColumns + 1; i++) { //получаем названия
столбцов
                String columnName = rsmd.getColumnName(i);
                text = text + columnName + ", ";
            }
            text = text + "\n";
            while(result.next()) { //получаем данные из строк таблицы

                for(int i =0; i<numberOfColumns; i++) {
                    if(i+1 == numberOfColumns) {;
                        text = text + result.getString(i+1) + "\n";
                    }
                    else {
                        text += result.getString(i+1) + ", ";
                    }
                }
            }
            return text;
        }

        public void client_choose_car () throws Exception {
            Parent                                root                                =
FXMLLoader.load(getClass().getResource("Client_add_services.fxml"));
            Stage window=(Stage) client_choose_car.getScene().getWindow();
            window.setScene(new Scene(root));
        }

        public void back_cat () throws Exception {
            Parent                                root                                =
FXMLLoader.load(getClass().getResource("Client_category.fxml"));
```

```
        Stage window=(Stage) back.getScene().getWindow();
        window.setScene(new Scene(root));
    }
}
```

Класс Car_premium:

```
package application;

import java.net.URL;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.stage.Stage;

public class Car_premium extends connect {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
```

```
private TextArea infowind;

@FXML
private Button inf;

@FXML
private Button Merc;

@FXML
private Button bmw;

@FXML
private Button client_choose_car;

@FXML
private Button back;

@FXML
void initialize() {
    Merc.setDisable(false);
    bmw.setDisable(false);
    inf.setOnAction(event ->{
    try {
            DataBaseHandler dbh = new DataBaseHandler();
            ResultSet result = dbh.getTablePremium("car");
            //ResultSet result = dbh.getTable("car");
            infowind.setText(showData(result));
                } catch (SQLException e) {
                    e.printStackTrace();
                }
```

```
});

 Merc.setOnAction(event ->{
        car_id="5";
        car_price=14600000;
        bmw.setDisable(true);
});

 bmw.setOnAction(event ->{
        car_id="6";
        car_price=7390000;
        Merc.setDisable(true);
});

 client_choose_car.setOnAction(event ->{
        try {
                client_adds();
                } catch (Exception e) {
                        e.printStackTrace();
                }
});

 back.setOnAction(event ->{
        try {
                        back_cat();
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
});
```

```
        }

        private String showData(ResultSet result) throws SQLException {
            ResultSetMetaData rsmd = result.getMetaData();
            int numberOfColumns = rsmd.getColumnCount();
            String text  = "";
            for (int i = 1; i < numberOfColumns + 1; i++) { //получаем названия
столбцов
                String columnName = rsmd.getColumnName(i);
                text = text + columnName + ", ";
            }
            text = text + "\n";
            while(result.next()) { //получаем данные из строк таблицы

                for(int i =0; i<numberOfColumns; i++) {
                    if(i+1 == numberOfColumns) {;
                        text = text + result.getString(i+1) + "\n";
                    }
                    else {
                        text += result.getString(i+1) + ", ";
                    }
                }
            }
            return text;
        }

        public void client_adds () throws Exception {
            Parent                                root                                =
FXMLLoader.load(getClass().getResource("Client_add_services.fxml"));
```

```
        Stage window=(Stage) client_choose_car.getScene().getWindow();
        window.setScene(new Scene(root));
    }

    public void back_cat () throws Exception {
        Parent root =
FXMLLoader.load(getClass().getResource("Client_category.fxml"));
        Stage window=(Stage) back.getScene().getWindow();
        window.setScene(new Scene(root));
    }
}
```

Класс Client_adds:

```
package application;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class Client_adds extends connect {

    @FXML
    private ResourceBundle resources;

    @FXML
```

```
private URL location;

@FXML
private Button tires;

@FXML
private Button camera;

@FXML
private Button interior;

@FXML
private Button color;

@FXML
private Button nothing;

@FXML
private Button next_scene;

@FXML
private Button back;

@FXML
void initialize() {
    tires.setDisable(false);
    camera.setDisable(false);
    interior.setDisable(false);
    color.setDisable(false);
    nothing.setDisable(false);
```

```
  tires.setOnAction(event ->{
        adds_name="Tires";
        adds_price=80000;
        camera.setDisable(true);
  interior.setDisable(true);
  color.setDisable(true);
  nothing.setDisable(true);
});

  camera.setOnAction(event ->{
        adds_name="Camera";
        adds_price=150000;
        tires.setDisable(true);
  interior.setDisable(true);
  color.setDisable(true);
  nothing.setDisable(true);
});

  interior.setOnAction(event ->{
        adds_name="Interior";
        adds_price=350000;
        tires.setDisable(true);
  camera.setDisable(true);
  color.setDisable(true);
  nothing.setDisable(true);
});

  color.setOnAction(event ->{
        adds_name="Color";
```

```
            adds_price=15000;
            tires.setDisable(true);
  interior.setDisable(true);
  camera.setDisable(true);
  nothing.setDisable(true);
});

  nothing.setOnAction(event ->{
            adds_name="Nothing";
            adds_price=0;
            tires.setDisable(true);
  interior.setDisable(true);
  color.setDisable(true);
  camera.setDisable(true);
});
  next_scene.setOnAction(event ->{
            try {
                        client_adds();
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
});

  back.setOnAction(event ->{
            try {
                  switch(car_category_name) {
                        case("Economy"):
                              back_eco();
                              break;
                        case("Comfort"):
```

```
                    back_com();
                        break;
                case("Premium"):
                    back_prem();
                    break;
                default:
                                break;
                }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            });
        }

        public void client_adds () throws Exception {
            Parent                          root                          =
FXMLLoader.load(getClass().getResource("Client_contract.fxml"));
            Stage window=(Stage) next_scene.getScene().getWindow();
            window.setScene(new Scene(root));
        }

        public void back_eco () throws Exception {
            Parent                          root                          =
FXMLLoader.load(getClass().getResource("Client_car(economy).fxml"));
            Stage window=(Stage) back.getScene().getWindow();
            window.setScene(new Scene(root));
        }

        public void back_com () throws Exception {
```

```
            Parent root =
FXMLLoader.load(getClass().getResource("Client_car(comfort).fxml"));
            Stage window=(Stage) back.getScene().getWindow();
            window.setScene(new Scene(root));
        }

        public void back_prem () throws Exception {
            Parent root =
FXMLLoader.load(getClass().getResource("Client_car(premium).fxml"));
            Stage window=(Stage) back.getScene().getWindow();
            window.setScene(new Scene(root));
        }
    }
```

Класс Client_category:

```
package application;

import java.net.URL;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.ResourceBundle;

import java.util.ArrayList;
import java.util.List;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
```

```
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.stage.Stage;

public class Client_category extends connect {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private ComboBox<String> menu_cat;

    @FXML
    private Button client_car;

    @FXML
    void client_car(ActionEvent event) {

    }

    ObservableList<String> langs = FXCollections.observableArrayList();

    @FXML
    void initialize() {
```

```
DataBaseHandler dbh = new DataBaseHandler();
ResultSet rs = dbh.getTableCategory(null);
try {
		while(rs.next()) {
			langs.add(rs.getString(1));
		}
		menu_cat.setItems(langs);
		menu_cat.setPromptText("Category");
	} catch (SQLException e) {
		e.printStackTrace();
	}

client_car.setOnAction(event ->{
		try {
			//System.out.println(menu_cat.getValue());
			switch (menu_cat.getValue()) {
				case("Economy"):
					client_carEc();
					car_category_name="Economy";
					//System.out.println("бомж");
					break;
				case("Comfort"):
					client_carCom();
					car_category_name="Comfort";
					//System.out.println("хорош");
					break;
				case("Premium"):
					client_carPrem();
					car_category_name="Premium";
					//System.out.println("владик");
```

```
                                        break;
                                default:
                                        break;
                                }
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                });
        }

        public void client_carEc () throws Exception {
            Parent                                  root                                =
FXMLLoader.load(getClass().getResource("Client_car(economy).fxml"));
            Stage window=(Stage) client_car.getScene().getWindow();
            window.setScene(new Scene(root));
        }

        public void client_carCom () throws Exception {
            Parent                                  root                                =
FXMLLoader.load(getClass().getResource("Client_car(comfort).fxml"));
            Stage window=(Stage) client_car.getScene().getWindow();
            window.setScene(new Scene(root));
        }

        public void client_carPrem () throws Exception {
            Parent                                  root                                =
FXMLLoader.load(getClass().getResource("Client_car(premium).fxml"));
            Stage window=(Stage) client_car.getScene().getWindow();
            window.setScene(new Scene(root));
        }
```

}

Класс Client_contract:

```
package application;

import java.net.URL;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.ResourceBundle;

import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.stage.Stage;

public class Client_contract extends connect {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
```

```
private Button accept;

@FXML
private TextArea inf_con;

@FXML
private Button details;

@FXML
private Button decline;

@FXML
private Button exit;

@FXML
void initialize() {
    decline.setDisable(false);
    details.setDisable(true);
    DataBaseHandler dbHandler = new DataBaseHandler();
    details.setOnAction(event ->{
    ResultSet result = dbHandler.getTable("client_has_car");
    try {
                        inf_con.setText(showData(result));
                } catch (SQLException e) {
                        e.printStackTrace();
                }
    });

    decline.setOnAction(event ->{
            Stage stage = (Stage) decline.getScene().getWindow();
```

```
            stage.close();
        });
        DataBaseHandler dbHandler2=new DataBaseHandler();
        DataBaseHandler dbHandler1 = new DataBaseHandler();
        accept.setOnAction(event ->{
                sum=car_price+adds_price;
        dbHandler1.chsIns();
                dbHandler2.minusStock();
        decline.setDisable(true);
        details.setDisable(false);
        });

        exit.setOnAction(event ->{
                try {
                        menu();
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        });
    }

    private String showData(ResultSet result) throws SQLException {
        ResultSetMetaData rsmd = result.getMetaData();
        int numberOfColumns = rsmd.getColumnCount();
        String text  = "";
        for (int i = 1; i < numberOfColumns + 1; i++) { //получаем названия
столбцов
            String columnName = rsmd.getColumnName(i);
            text = text + columnName + ", ";
```

```
            }
            text = text + "\n";
            while(result.next()) { //получаем данные из строк таблицы

               for(int i =0; i<numberOfColumns; i++) {
                  if(i+1 == numberOfColumns) {;
                     text = text + result.getString(i+1) + "\n";
                  }
                  else {
                     text += result.getString(i+1) + ", ";
                  }
               }
            }
            return text;
         }

         public void menu () throws Exception {
            Parent root =
FXMLLoader.load(getClass().getResource("First_in.fxml"));
            Stage window=(Stage) exit.getScene().getWindow();
            window.setScene(new Scene(root));
         }
      }
```

Класс Client_in:

```
package application;

import java.net.URL;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
import java.util.ResourceBundle;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import animations.Shake;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

public class Client_in extends connect{

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private TextField first_name;

    @FXML
    private TextField last_name;

    @FXML
    private TextField middle_name;
```

```
    @FXML
    private TextField telephone;

    @FXML
    private TextField email;

    @FXML
    private Button client_cat;

    @FXML
    private TextField id;

    @FXML
    private Button client_accept;

    @FXML
    private Button back;

    @FXML
    void initialize() {
        //Паттерн на почту и телефонный номер
        Pattern emailCheck = Pattern.compile("[A-Za-z]+@{1}[A-Za-z]+.{1}[A-Za-z]+");
        Pattern telCheck = Pattern.compile("[0-9]{11}$");
        Pattern idCheck = Pattern.compile("c{1}l{1}[0-9]+");
        Pattern firCheck = Pattern.compile("[A-Za-z]+");
        Pattern midCheck = Pattern.compile("[A-Za-z]+");
        Pattern lstCheck = Pattern.compile("[A-Za-z]+");
        client_cat.setDisable(true);
       DataBaseHandler dbHandler = new DataBaseHandler();
```

```
            Shake clientId = new Shake(id);
                Shake tel = new Shake(telephone);
                Shake firName = new Shake(first_name);
                Shake midName = new Shake(middle_name);
                Shake lstName = new Shake(last_name);
                Shake mail = new Shake(email);
             client_accept.setOnAction(event ->{
                String idText = id.getText().trim();
                String telText = telephone.getText().trim();
                String firstName = first_name.getText().trim();
                String middleName = middle_name.getText().trim();
                String lastName = last_name.getText().trim();
                String emailText = email.getText().trim();
                Matcher telMatcher = telCheck.matcher(telText);
                Matcher emailMatcher = emailCheck.matcher(emailText);
                Matcher idMatcher = idCheck.matcher(idText);
                Matcher firMatcher = firCheck.matcher(firstName);
                Matcher midMatcher = midCheck.matcher(middleName);
                Matcher lstMatcher = lstCheck.matcher(lastName);
                if (!(idText.length()==0) && !(telText.length()==0) &&
!(firstName.length()==0)
                        && !(middleName.length()==0)
&&!(lastName.length()==0) && !(emailText.length()==0) &&
telMatcher.matches()
                        && emailMatcher.matches() &&
idMatcher.matches() && firMatcher.matches() && midMatcher.matches()
                        && lstMatcher.matches()) {
                    dbHandler.signUpClient(idText, telText, firstName,
middleName, lastName, emailText);
                    client_id = idText;
```

```
                client_cat.setDisable(false);
                }else {
                        clientId.playAnim();
                        tel.playAnim();
                        firName.playAnim();
                        midName.playAnim();
                        lstName.playAnim();
                        mail.playAnim();
                }
        });
        client_cat.setOnAction(event ->{
        try {
                        client_category();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        });

        back.setOnAction(event ->{
                try {
                        back();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        });
    }

    public void client_category () throws Exception {
        Parent root =
FXMLLoader.load(getClass().getResource("Client_category.fxml"));
```

```
            Stage window=(Stage) client_cat.getScene().getWindow();
            window.setScene(new Scene(root));
        }

        public void back () throws Exception {
            Parent root =
FXMLLoader.load(getClass().getResource("First_in.fxml"));
            Stage window=(Stage) back.getScene().getWindow();
            window.setScene(new Scene(root));
        }
    }
```

Класс Configs:

```
package application;

public class Configs extends connect {
    protected String dbHoSt = "localhost";
    protected String dbPort = "3306";
    protected String dbUser = "root";
    protected String dbPass = "Albus011415fazimo";
    protected String dbName = "project_schema";
}
```

Класс connect:

```
package application;

public abstract class connect {
    public static String client_id;
    public static String car_id;
    public static int car_price;
    public static String car_category_name;
    public static String adds_name;
    public static int adds_price;
    public static String admin_id;
    public static int sum;
```

```
	public static int stock;
}
```

Класс Const:

```
package application;

public class Const {
	public static final String EMPLOYEE_TABLE = "employee";
	public static final String EMPLOYEE_ID = "ID";
	public static final String EMPLOYEE_TELEPHONE = "Telephone";
	public static final String EMPLOYEE_FIRSTNAME = "First_Name";
	public static final String EMPLOYEE_LASTNAME = "Last_Name";
	public static final String EMPLOYEE_EMAIL = "Email";
	public static final String EMPLOYEE_POSITION = "Position";
	public static final String EMPLOYEE_PASSWORD = "Password";

	public static final String CLIENT_TABLE = "client";
	public static final String CLIENT_ID = "ID";
	public static final String CLIENT_TELEPHONE = "Telephone";
	public static final String CLIENT_FIRSTNAME = "First_Name";
	public static final String CLIENT_MIDDLENAME = "Middle_Name";
	public static final String CLIENT_LASTNAME = "Last_Name";
	public static final String CLIENT_EMAIL = "Email";

	public static final String CATEGORY_TABLE = "category";
	public static final String CATEGORY_ID = "ID";
	public static final String CATEGORY_NAME = "Name";

	public static final String CHC_TABLE = "client_has_car";
	public static final String CHC_CLIENT_ID = "Client_ID";
	public static final String CHC_CAR_ID = "Car_ID";
	public static final String CHC_CAR_PRICE = "Car_Price";
	public static final String CHC_ADD_SERVICES_ID = "Add_services_ID";
	public static final String CHC_ADD_SERVICES_PRICE =
"Add_services_Price";

	public static final String CAR_TABLE = "car";
	public static final String CAR_ID = "ID";
	public static final String CAR_PRICE = "Price";
	public static final String CAR_MODEL = "Model";
```

```
        public static final String CAR_INSTOCK = "In_stock";
        public static final String CAR_CATEGORY = "Category_Name";

        public static final String CONTRACT_TABLE = "contract";
        public static final String CONTRACT_SUM = "Purchase_amount";
        public static final String CONTRACT_CLIENT_ID =
"Client_has_Car_Client_ID";
}
```

Класс DataBaseHandler:

```
        package application;


        import java.sql.Connection;
        import java.sql.SQLException;
        import java.sql.DriverManager;
        import java.sql.PreparedStatement;
        import java.sql.ResultSet;


        public class DataBaseHandler extends Configs {
                Connection dbConnection;
                public Connection getDbConnection() throws
ClassNotFoundException, SQLException {
                        String connectionString = "jdbc:mysql://" +dbHoSt + ":" +
dbPort + "/" + dbName;
                        Class.forName("com.mysql.cj.jdbc.Driver");
                        dbConnection =
DriverManager.getConnection(connectionString, dbUser, dbPass);
                        return dbConnection;
                }
                //администратор вход
                public ResultSet getAdm(Employee employee) {
                        ResultSet resSet = null;
```

```
                String select = "SELECT * FROM " +
Const.EMPLOYEE_TABLE + " WHERE " +
                        Const.EMPLOYEE_ID + "=? AND " +
Const.EMPLOYEE_PASSWORD + "=?";
                PreparedStatement prSt;
                try {
                        prSt = getDbConnection().prepareStatement(select);
                        prSt.setString(1, employee.getId());
                        prSt.setString(2, employee.getPassword());

                        resSet = prSt.executeQuery();
                } catch (ClassNotFoundException e) {
                        e.printStackTrace();
                } catch (SQLException e) {
                        e.printStackTrace();
                }

                return resSet;
            }
            //категория машин
            public ResultSet getTableCategory(String name) {
          ResultSet resSet=null;
          String select = "SELECT " + Const.CATEGORY_NAME + " FROM
"+ Const.CATEGORY_TABLE +";";
          try {
              PreparedStatement prSt =
getDbConnection().prepareStatement(select);
              resSet=prSt.executeQuery();

          } catch (ClassNotFoundException e) {
```

```
            e.printStackTrace();

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return resSet;
    }

      public ResultSet getTableEconomy(String name) {
        ResultSet resSet=null;
        String select = "SELECT * FROM " + name + " WHERE "+
Const.CAR_CATEGORY + "= 'Economy'" + ";";
        try {
            PreparedStatement prSt =
getDbConnection().prepareStatement(select);
            resSet=prSt.executeQuery();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return resSet;
```

```
        }

          public ResultSet getTableComfort(String name) {
        ResultSet resSet=null;
        String select = "SELECT * FROM " + name + " WHERE "+
Const.CAR_CATEGORY + "= 'Comfort'" + ";";
        try {
          PreparedStatement prSt =
getDbConnection().prepareStatement(select);
          resSet=prSt.executeQuery();

        } catch (ClassNotFoundException e) {

          e.printStackTrace();

        } catch (SQLException e) {

          e.printStackTrace();

        }
        return resSet;
        }

          public ResultSet getTablePremium(String name) {
        ResultSet resSet=null;
        String select = "SELECT * FROM " + name + " WHERE "+
Const.CAR_CATEGORY + "= 'Premium'" + ";";
        try {
          PreparedStatement prSt =
getDbConnection().prepareStatement(select);
```

```
            resSet=prSt.executeQuery();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        } catch (SQLException e) {

            e.printStackTrace();

        }
        return resSet;
    }

      //получение таблицы
      public ResultSet getTable(String name) {
        ResultSet resSet=null;
        String select = "SELECT * FROM "+ name +";";
        try {
            PreparedStatement prSt =
getDbConnection().prepareStatement(select);
            resSet=prSt.executeQuery();

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        } catch (SQLException e) {

            e.printStackTrace();
```

```
            }
            return resSet;
        }
            public void signUpClient(String idText, String telText, String
firstName, String middleName, String lastName,
                    String emailText) {
                String insert = "INSERT INTO " + Const.CLIENT_TABLE + "
VALUES(?,?,?,?,?,?)" + ";";
                PreparedStatement prSt;
                try {
                    prSt = getDbConnection().prepareStatement(insert);
                    prSt.setString(1, idText);
                    prSt.setString(2, telText);
                    prSt.setString(3, firstName);
                    prSt.setString(4, middleName);
                    prSt.setString(5, lastName);
                    prSt.setString(6, emailText);

                    prSt.executeUpdate();
                } catch (ClassNotFoundException e) {
                    e.printStackTrace();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }

            public void chsIns() {
                String insert = "INSERT INTO " + Const.CHC_TABLE + "
VALUES(?,?,?,?,?,?)" + ";";
```

```
                PreparedStatement prSt;
                try {
                        prSt = getDbConnection().prepareStatement(insert);
                        prSt.setString(1, client_id);
                        prSt.setString(2, car_id);
                        prSt.setLong(3, car_price);
                        prSt.setString(4, car_category_name);
                        prSt.setLong(5, adds_price);
                        prSt.setString(6, adds_name);

                        prSt.executeUpdate();
                } catch (ClassNotFoundException e) {
                        e.printStackTrace();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }

        public void contractIns() {
                String insert = "INSERT INTO " +
Const.CONTRACT_TABLE + " VALUES(?,?,?,?,?,?,?,?)" + ";";
                PreparedStatement prSt;
                try {
                        prSt = getDbConnection().prepareStatement(insert);
                        prSt.setLong(1, sum);
                        prSt.setString(2, admin_id);
                        prSt.setString(3, client_id);
                        prSt.setString(4, car_id);
                        prSt.setLong(5, car_price);
                        prSt.setString(6, car_category_name);
```

```
                prSt.setLong(7, adds_price);
                prSt.setString(8, adds_name);

                prSt.executeUpdate();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        public void deletePurch() {
            String delete = "DELETE FROM "+ Const.CONTRACT_TABLE +";";
            PreparedStatement prSt;
            try {
                prSt = getDbConnection().prepareStatement(delete);
                prSt.executeUpdate();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            } catch (SQLException e) {
                e.printStackTrace();
            }

        }

        public boolean deleteCHC() {
            String delete = "DELETE FROM "+ Const.CHC_TABLE +";";
            PreparedStatement prSt;
            try {
```

```
                prSt = getDbConnection().prepareStatement(delete);
                prSt.executeUpdate();
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return false;
        }

        public void minusStock() {
            String insert = "UPDATE "+Const.CAR_TABLE + " SET " +
Const.CAR_INSTOCK + " = " + Const.CAR_INSTOCK + "-1 " + " WHERE " +
Const.CAR_ID + " = " + car_id;
            PreparedStatement prSt;
            try {
                prSt = getDbConnection().prepareStatement(insert);
                prSt.executeUpdate();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        public void plusStock() {
```

```
                String insert = "UPDATE "+Const.CAR_TABLE + " SET " +
Const.CAR_INSTOCK + " = " + Const.CAR_INSTOCK + "+1 " + " WHERE " +
Const.CAR_ID + " = " + car_id;
                PreparedStatement prSt;
                try {
                        prSt = getDbConnection().prepareStatement(insert);
                        prSt.executeUpdate();
                } catch (ClassNotFoundException e) {
                        e.printStackTrace();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }

}
```

Класс Employee:

```
package application;

public class Employee {
        public String getId() {
                return id;
        }
        public void setId(String id) {
                this.id = id;
        }

        public Employee(String id, String password) {
                super();
                this.id = id;
                this.password = password;
        }

        public Employee() {
                // TODO Auto-generated constructor stub
        }
```

```
	private String id;
	private String password;

	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
}
```

Класс In_controller:

```
package application;


import java.net.URL;
import java.util.ResourceBundle;


import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;


public class In_controller {


   @FXML
   private ResourceBundle resources;


   @FXML
```

```
private URL location;

@FXML
private Button client_in;

@FXML
private Button admin_in;

@FXML
private Button client_date;

@FXML
private Button client_adds;

@FXML
private Button accept;

@FXML
private Button client_car;

@FXML
private Button client_cat;

@FXML
private Button client_choose_car;

@FXML
private Button admin_stat;

@FXML
```

```
    private Button exit;

    @FXML
    void initialize() {
        admin_in.setOnAction(event -> {
            try {
                    admin_in();
                } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
        });

        exit.setOnAction(event ->{
            Stage stage = (Stage) exit.getScene().getWindow();
          stage.close();
        });
    }

    public void admin_in () throws Exception {
        Parent root =
FXMLLoader.load(getClass().getResource("Administrator_in.fxml"));
        Stage window=(Stage) admin_in.getScene().getWindow();
        window.setScene(new Scene(root));
    }

    public void client_in () throws Exception {
        Parent root =
FXMLLoader.load(getClass().getResource("Client_in.fxml"));
        Stage window=(Stage) client_in.getScene().getWindow();
```

```
            window.setScene(new Scene(root));
        }

    }
```

Класс Main:

```
package application;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.fxml.FXMLLoader;


public class Main extends Application {
        @Override
        public void start(Stage primaryStage) throws Exception {
                Parent root =
FXMLLoader.load(getClass().getResource("First_in.fxml"));
                primaryStage.setScene(new Scene(root));
                primaryStage.getIcons().add(new
Image(getClass().getResourceAsStream("test.png")));
                primaryStage.setTitle("CAR SHOWROOM 'MORO'");
                primaryStage.show();
        }

        public static void main(String[] args) {
            launch(args);
```

```
        }
    }
```