МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Самарский национальный исследовательский
университет имени академика С.П. Королева»
(Самарский университет)

Институт информатики, математики и электроники
Факультет Информатики
Кафедра информационных систем и технологий

КУРСОВОЙ ПРОЕКТ ПО ДИСЦИПЛИНЕ
«Разработка WEB-приложений»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

«Автоматизированная информационная система учета продаж автомобилей»

Студент _____ В.А.Виханов
(подпись)

Руководитель работы _____ И.А. Лёзин
(подпись)

САМАРА 2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»
(Самарский университет)

Институт информатики, математики и электроники

Факультет Информатики

Кафедра информационных систем и технологий

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

Студенту ***Виханову Владимиру Алексеевичу*** группы 6404-090301D

Тема проекта: ***«Автоматизированная информационная система учета продаж автомобилей»***

Планируемые результаты освоения образовательной программы (компетенции)	Планируемые результаты практики	Содержание задания
ПК-1 – способность разрабатывать модели компонентов информационных систем, включая модели баз данных и модели интерфейсов "человек - электронно-вычислительная машина"	**знать**: теоретические основы проектирования и создания моделей баз данных и программного обеспечения при разработке web-приложений в рамках технологии Java Enterprise Edition (EE); **уметь**: создавать и проверять работоспособность моделей баз данных и программного обеспечения при разработке web-приложений в рамках технологии Java EE; **владеть**: современными программными продуктами для создания моделей баз данных и программного обеспечения при разработке web-приложений в рамках технологии Java Enterprise Edition.	1. Разработка логической модели базы данных автоматизированной информационной системы учета продаж автомобилей. 2. Разработка автоматизированной информационной системы учета продаж автомобилей. 3. Отладка и тестирование разработанной автоматизированной информационной системы учета продаж автомобилей. 4. Проведение тестирования и анализ результатов.

Дата выдачи задания 21 февраля 2020 г.
Срок представления на кафедру пояснительной записки 6 марта 2020 г.

Руководитель курсового проекта
доцент каф. ИСТ, к.т.н., доцент _____ И.А. Лёзин
(подпись)

Задание принял к исполнению
студент группы № _6404-090301D___ _____ В.А.Виханов
(подпись)

РЕФЕРАТ

Пояснительная записка к курсовому проекту: 19 с., 17 рисунков, 0 таблиц, 14 источников, 3 приложения.

ИНФОРМАЦИОННАЯ СИСТЕМА, WEB-ИНТЕРФЕЙС, ВЕДЕНИЕ УЧЁТА ПРОДАЖ АВТОМОБИЛЕЙ, JAVA EE-ТЕХНОЛОГИИ

Цель работы – разработать автоматизированную информационную систему учета продаж автомобилей на основе JavaEE-технологий.

Разработана автоматизированная информационная система учета продаж автомобилей на основе JavaEE-технологий. Для хранения данных система использует базу данных на сервере PostgreSQL.

Система обеспечивает ведение автосалона, заказов на автомобили, их производителей, а также клиентов и сотрудников салона: просмотр и редактирование списков имеющихся автомобилей и их производителей; просмотр информации о заказах, включая модель, ФИО клиента и продавца, дату и тип платежа. Вся работа с системой производится через Web-интерфейс, представленный в виде JSP-страниц.

Система реализована средствами разработки и отладки IntelliJ IDEA на языке программирования Java. Выполнена отладка и проверка системы на работоспособность.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В рамках курсового проекта необходимо реализовать информационную систему учета продаж автомобилей.

Учёт продаж автомобилей состоит в занесении в базу данных сведений о новом авто, включая название модели, цвет, тип трансмиссии, тип кузова, цену и производителя.

При добавлении нового производителя необходимо указать его название, адрес и телефон.

При добавлении нового клиента необходимо указать его ФИО, паспортные данные и телефон.

При добавлении нового сотрудника необходимо указать его ФИО, должность и телефон.

При добавлении нового заказа необходимо указать дату и тип платежа, а также выбрать клиента, сотрудника и автомобиль.

Традиционный метод ведения всего учёта на бумаге не обладает достаточным удобством и простотой, к его недостаткам можно отнести возможность потери или порчи носителя, ошибки в записи.

По завершении процессов разработки и реализации системы необходимо получить автоматизированную информационную систему, лишённую вышеперечисленных недостатков, позволяющую пользователю вести автомобили, заказы на них, производителей автомобилей, а также клиентов и сотрудников салона.

1 СТРУКТУРА БАЗЫ ДАННЫХ

1.1 Логическая структура базы данных

Логическая модель базы данных – графическое представление структуры информации, необходимой для решения задач и описания предметной области.

Основным средством разработки логической модели базы данных в настоящий момент являются различные варианты ER-диаграмм (Entity-Relationship, диаграммы сущность-связь) [1].

ER-диаграммы используют модель типа «сущность-связь» (ER-модель), основными понятиями которой являются сущность, экземпляр сущности, атрибут и связь.

В этой модели реальный или мыслимый объект представлен сущностью. Сущности характеризуются атрибутами (свойствами) и могут содержать множество экземпляров. Связь отражает отношение объектов предметной области и может быть установлена между сущностями [2].

На рисунке 1 представлена логическая модель базы данных системы.

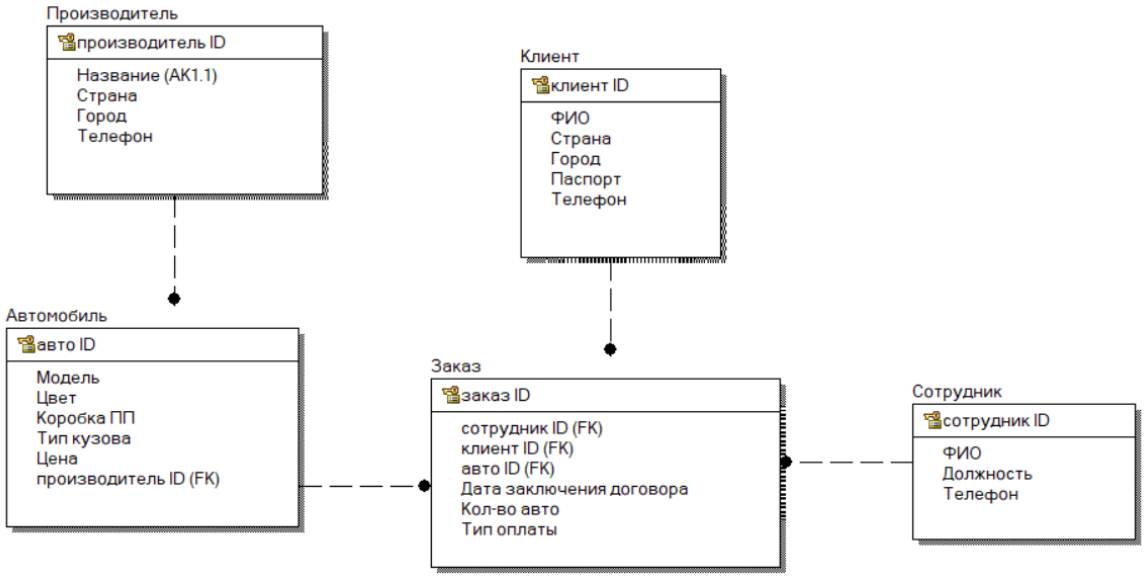

Рисунок 1 – Логическая модель базы данных

Логическая модель базы данных системы включает в себя пять сущностей: «Производитель», «Клиент», «Автомобиль», «Заказ», «Сотрудник».

1.2 Физическая структура базы данных

Физическая модель базы данных оперирует категориями, касающимися организации внешней памяти и структур хранения, используемых в данной операционной среде. Физическая модель базы данных описывает реализацию объектов логической модели на уровне объектов конкретной базы данных [2].

При переходе от логической модели базы данных (диаграммы сущностных классов) к физической (схеме таблиц) и наоборот существует соответствие между элементами модели.

На рисунке 2 приведена физическая модель базы данных, соответствующая логической модели, представленной в пункте 1.1.

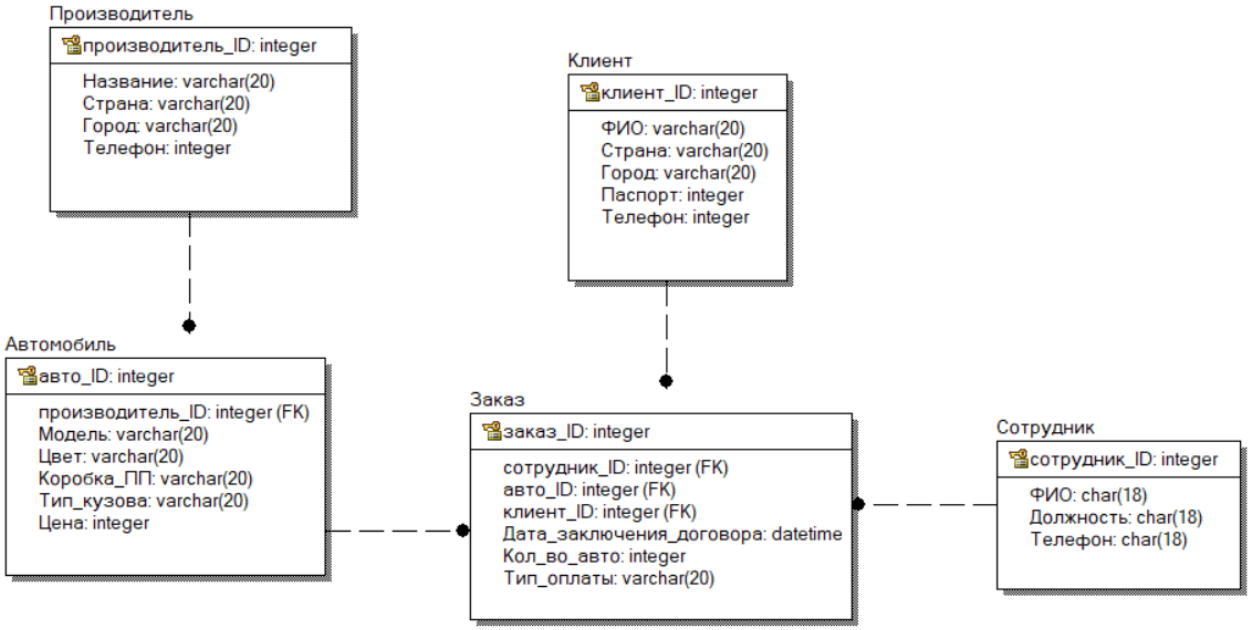

Рисунок 2 – Физическая модель базы данных

2 АРХИТЕКТУРА ПРИЛОЖЕНИЯ

2.1 Архитектурная модель

Архитектура программного обеспечения определяет совокупность важнейших решений об организации программной системы. Архитектура включает: выбор структурных элементов, с помощью которых составлена система; соединение выбранных элементов структуры и поведения во всё более крупные системы; архитектурный стиль, который направляет всю организацию [3].

Для реализации системы, соответствующей поставленному заданию, была выбрана трёхзвенная модель архитектуры. Такая архитектурная модель предполагает наличие в системе трёх компонентов: клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений) [4]. На рисунке 3 представлена диаграмма компонентов трёхзвенной архитектуры.

Рисунок 3 – Диаграмма компонентов трёхзвенной архитектуры

2.2 Выбор языка программирования и среды разработки

Среда разработки IntelliJ IDEA представляет мощные и удобные средства для быстрой и простой разработки настольных, мобильных и веб-приложений, написанных на Java, JavaScript, HTML5 и др.

Язык программирования Java является сильно типизированным объектно-ориентированным языком программирования. Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре с помощью виртуальной Java-машины [5]. Исходя из характера поставленной задачи и технических требований, была выбрана среда разработки IntelliJ IDEA и язык программирования Java.

2.3 Выбор системы управления базами данных

Система управления базами данных (СУБД) – специализированная программа (чаще комплекс программ), предназначенная для организации и ведения базы данных. Для создания и управления информационной системой СУБД необходима в той же степени, как для разработки программы на алгоритмическом языке необходим транслятор [6].

PostgreSQL – это свободно распространяемая объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных. PostgreSQL поддерживается на всех современных Unix-системах (34 платформы), включая наиболее распространённые, такие как Linux, FreeBSD, NetBSD, OpenBSD, SunOS, Solaris, DUX, а также под Mac OS X и Microsoft Windows [7].

2.4 Выбор способа работы с базой данных

В ходе работы приложения система должна обеспечить взаимодействие с базой данных.

API JDBC (Java Database Connectivity) является отраслевым стандартом для независимой от базы данных связи между языком программирования Java и широким спектром баз данных. Преимуществом JDBC как средства взаимодействия с базами данных является возможность организации связи с базами данных SQL и другими табличными источниками данных, такими как электронные таблицы или плоские файлы. JDBC API предоставляет API уровня вызовов для доступа к базе данных на основе SQL. Драйвер с технологией JDBC позволяет подключать все корпоративные данные даже в разнородной среде [8].

2.5 Выбор технологий JavaEE

Так как было принято выбрать трёхзвенную архитектуру приложения, необходимо определить технологии, которые будут использоваться на уровнях клиента и сервера приложений.

Для реализации клиентского уровня оптимально использовать технологию JSP (Java Server Pages). Страница JSP является текстовым документом, который содержит текст двух типов: статические исходные данные, которые могут быть оформлены в одном из текстовых форматов (HTML, XML), и JSP-элементы, которые конструируют динамическое содержимое [9]. Преимуществом страниц JSP является то, что они позволяют упростить процесс создания Web-страниц с динамическим содержимым как для программиста, так и для дизайнера.

Для реализации взаимодействия на уровне сервера приложений выбрана технология сервлетов (Servlets). Сервлет позволяет разработчику реализовать сложную обработку запросов клиента на сервере. При использовании комбинации технологий «сервлет + JSP» сервлет позволяет отделить реализацию бизнес-логики от представления на стороне клиента.

К преимуществам сервлетов можно отнести высокую скорость работы, переносимость, удобство кодирования [10].

2.6 Выбор сервера приложений

Apache Tomcat 9.0.31 является контейнером сервлетов с открытым исходным кодом. Он реализует спецификацию сервлетов, спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Tomcat позволяет запускать веб-приложения, содержит ряд программ для самоконфигурирования. Преимуществом Tomcat можно считать то, что он используется в качестве самостоятельного веб-сервера [11].

2.7 Варианты использования, структуры данных и классы

Объект – структура данных, содержащая описание свойств внешнего объекта программирования [12]. Класс – это структурный тип данных, который включает описание полей данных, а также процедур и функций, работающих с этими полями данных [13]. Классы в объектно-ориентированном программировании используются для реализации всех необходимых функций системы.

Диаграмма вариантов использования позволяет описать все необходимые взаимодействия пользователей с системой (рисунок 4).

Подобное описание используется как основа для последующего описания диаграммы классов, необходимых для реализации функций системы.

Диаграммы классов – это наиболее часто используемый тип диаграмм, которые создаются при моделировании объектно-ориентированных систем, они показывают набор классов, интерфейсов и кооперацией, а также их связи [12].

С учётом выбранного языка программирования и архитектурной модели были созданы классы, реализующие основные функции системы. Диаграмма классов системы представлена на рисунке 5.

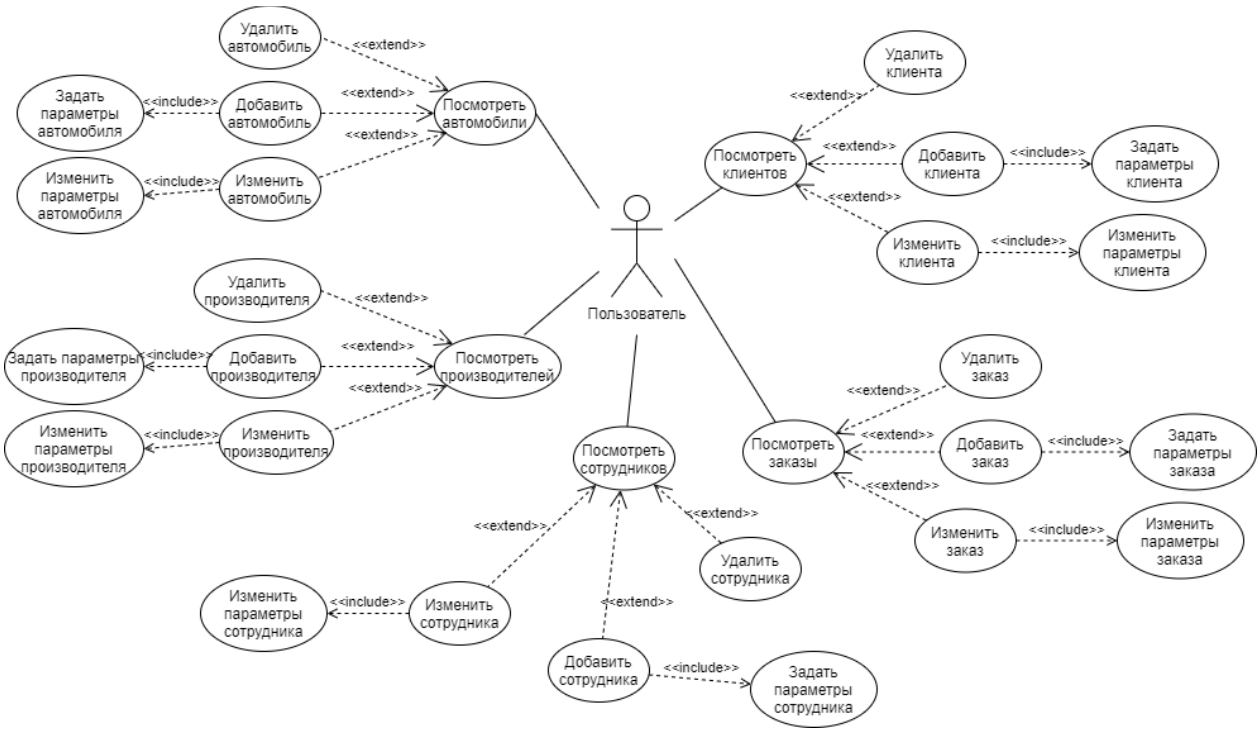

Рисунок 4 – Диаграмма вариантов использовани

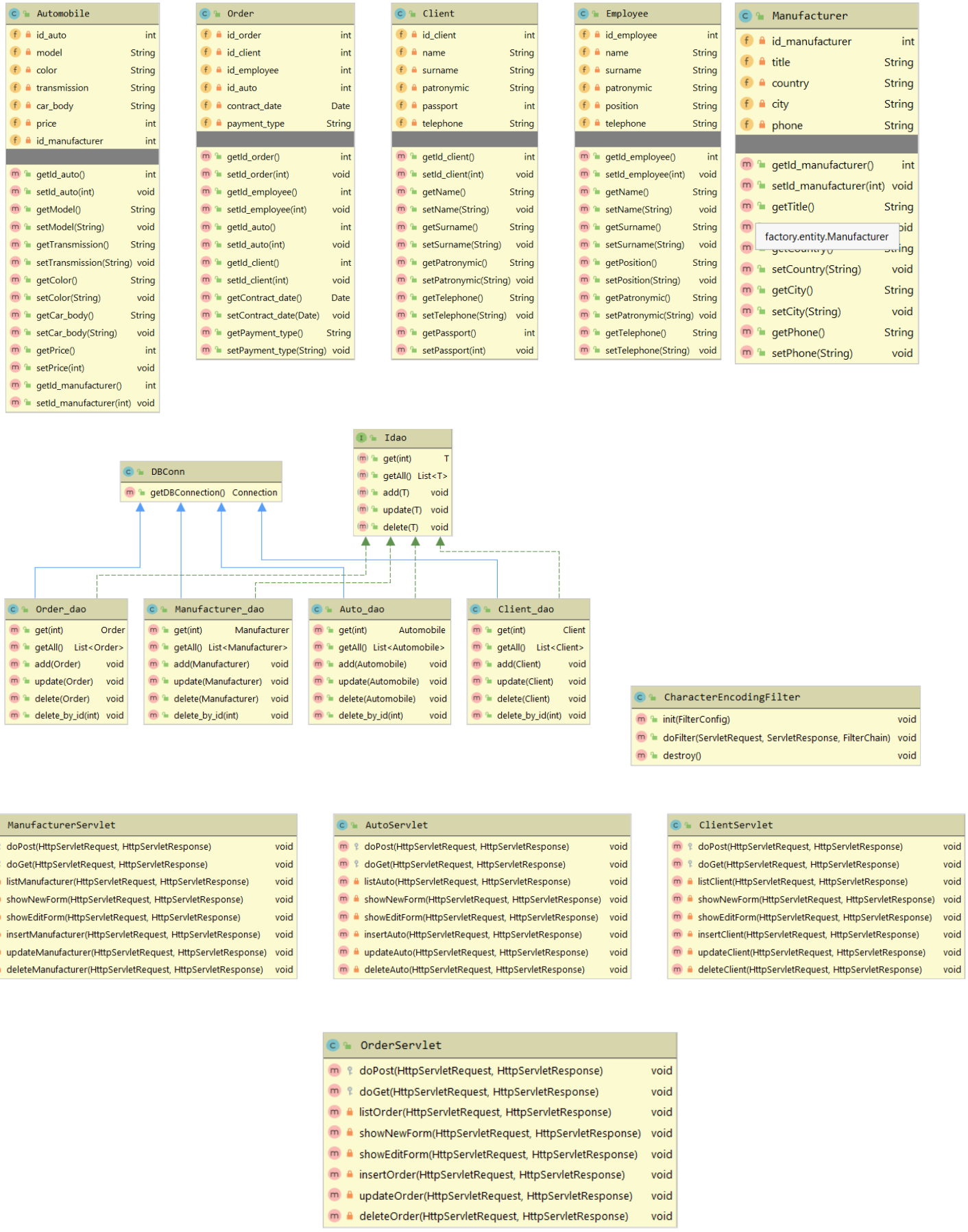

Рисунок 5 – Диаграмма классов

3 ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

Пользовательский интерфейс – это набор программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Основу такого взаимодействия составляют диалоги [14]. Под диалогом в данном случае понимают регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи. Каждый диалог состоит из отдельных процессов ввода/вывода, которые физически обеспечивают связь пользователя и компьютера. Обмен информацией осуществляется передачей сообщения.

При запуске приложения открывается главная страница (рисунок 6).

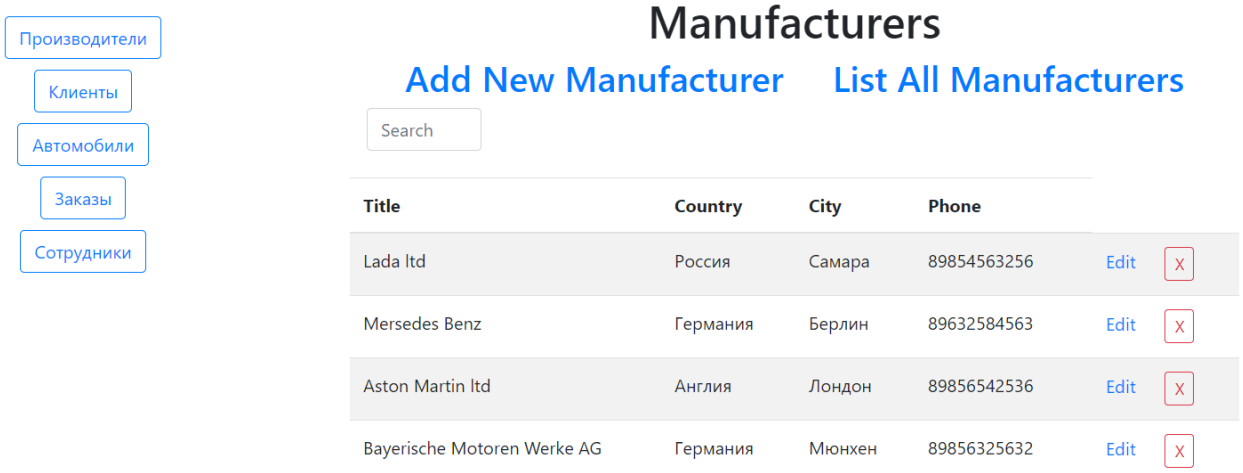

Рисунок 6 – Главная страница

При нажатии на название раздела слева открывается страница с отображением всех элементов, соответствующих выбранному разделу (рисунок 7). На этой странице можно добавить или удалить элементы, а также изменить информацию о существующих элементах.

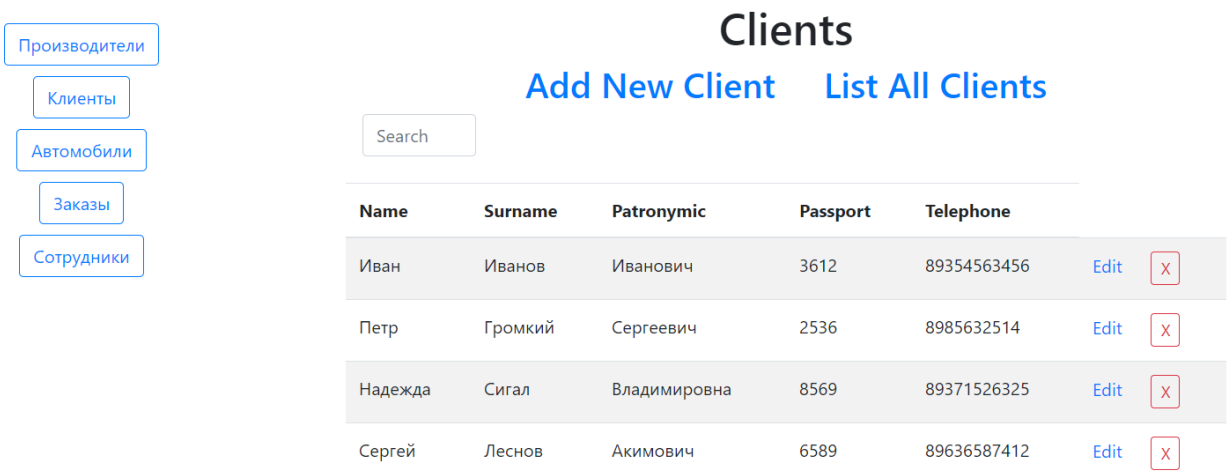

Рисунок 7 – Страница представления раздела «Клиенты»

Для добавления нового элемента необходимо на странице представления нажать кнопку «Add new client» и перейти к форме заполнения данных (рисунок 8). Новый элемент отобразится в соответствующем разделе.

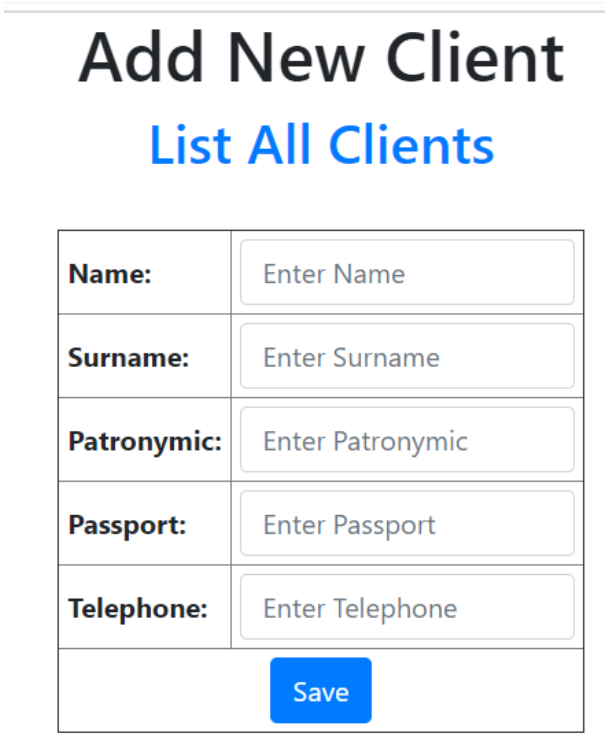

Рисунок 8 – Добавление нового элемента раздела «Клиенты»

Чтобы изменить информацию, необходимо нажать на кнопку «Edit» у необходимой записи. После этого отобразится форма, в которой необходимо

внести изменения в соответствующие поля и нажать кнопку «Изменить» (рисунок 9).

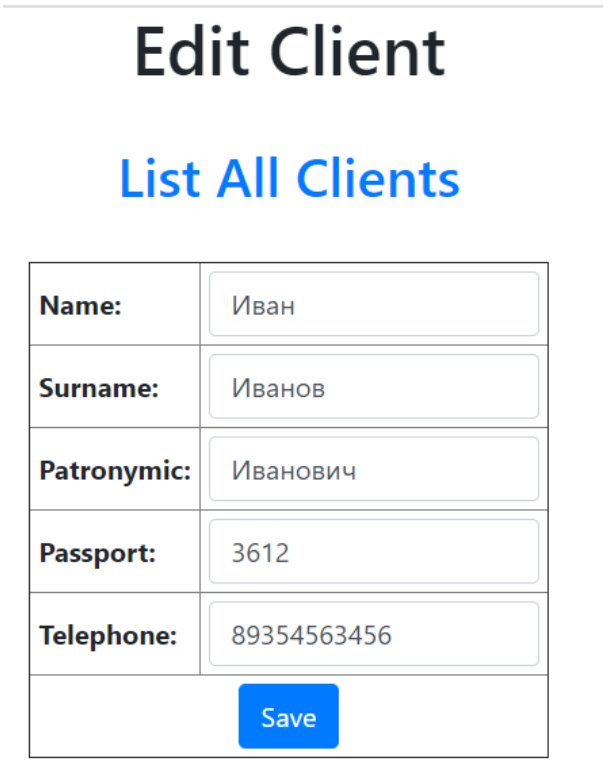

Рисунок 9 – Изменение данных об экспонате в форме редактирования

При нажатии на кнопку «Save» все введённые изменения сохраняются в БД и отображаются в соответствующем разделе меню.

Для удаления элемента из БД необходимо необходимо нажать на кнопку «X» у необходимой записи. После этого будет произведено удаление выбранного элемента из БД.

Чтобы визуально отсортировать любую из таблиц, необходимо нажать на название сортируемого столбца в шапке таблицы, и появится соответсвующий указатель сортировки рядом с названием столбца (Рисунок 10).

Для поиска необходимой информации в таблице можно воспользоваться полем ввода «Search» (Рисунок 11). После ввода ключевых слов в таблице отобразятся соответствующие критерию поиска записи.

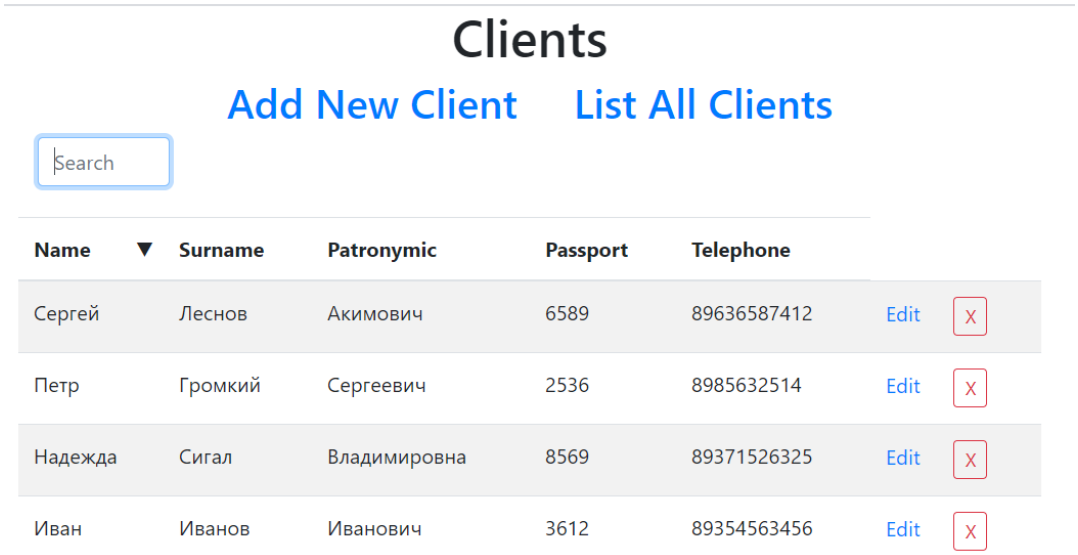

Рисунок 10 – Сортировка записей в таблице по имени

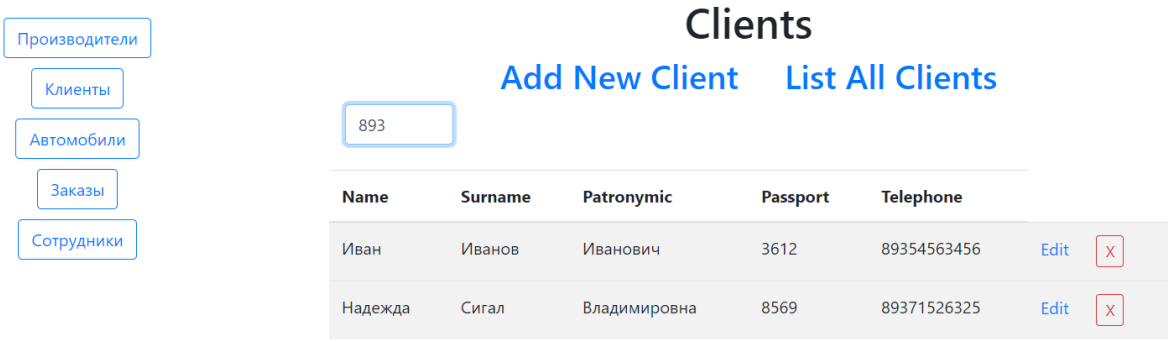

Рисунок 11 – Поиск записей в таблице по ключевым словам

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была разработана система «Автоматизированная информационная система автосалона», позволяющая вести учёт проданных автомобилей, их производителей, клиентов и сотрудников. Были спроектирована логическая модель базы данных, описаны используемые технологии и интерфейс пользователя.

Выполнено тестирование и отладка программы.

Реализованная система позволяет пользователю в простой и удобной форме вести учёт продаж автомобилей, осуществлять добавление, изменение и удаление информации о автомобилях, производителях, клиентов и сотрудников.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Мирошниченко, Г.А. Реляционные базы данных. Практические приемы оптимальных решений [Текст]/ Г.А. Мирошниченко. – Санкт-Петербург: БХВ-Петербург, 2005. – 400 с.: ил.

[2] Чигарина, Е.И. Базы данных [Текст]/ Е.И. Чигарина. – Самара: СГАУ, 2015. – 208 с

[3] Архитектура программного обеспечения [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Архитектура_программного_обеспечения (дата обращения: 27.02.2020 г.).

[4] Трёхуровневая архитектура [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Трёхуровневая_архитектура (дата обращения: 27.02.2020 г.).

[5] Java [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Java (дата обращения: 28.02.2020 г.).

[6] СУБД [Электронный ресурс] URL: https://dic.academic.ru/dic.nsf/ruwiki/1133383 (дата обращения: 28.02.2020 г.).

[7] Что такое PostgreSQL [Электронный ресурс] URL: http://www.sai.msu.su/~megera/postgres/talks/what_is_postgresql.html (дата обращения: 27.02.2020 г.).

[8] JDBC [Электронный ресурс] URL: https://www.oracle.com/technetwork/java/javase/jdbc/index.html (дата обращения: 27.02.2020 г.).

[9] Java Servlet Pages (JSP). Tomcat [Электронный ресурс]. – http://alfalavista.ru/idxfldr/2013-06-18-22-25-47/306-java-server-pages-jsp-tomcat.html (дата обращения: 29.02.2020 г.).

[10] Web-технологии, Java Servlet [Электронный ресурс]. – http://www.hardline.ru/4/86/3282/ (дата обращения: 29.02.2020 г.).

[11] Apache Tomcat [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Apache_Tomcat (дата обращения: 28.02.2020 г.).

[12] Объекты и классы [Электронный ресурс] URL: http://ermak.cs.nstu.ru/cprog/html/101.html (дата обращения: 27.02.2020 г.)

[13] Бабушкина И.А., Окулов С.М. Практикум по объектно-ориентированному программированию. [Электронный ресурс] URL: http://files.lbz.ru/pdf/cC2542-9-ch.pdf (дата обращения: 28.02.2020 г.).

[14] Пользовательский интерфейс [Электронный ресурс] URL: http://mirznanii.com/a/113961/polzovatelskiy-interfeys.html (дата обращения: 01.03.2020 г.).

ПРИЛОЖЕНИЕ А

SQL-скрипты

```
CREATE TABLE Manufacturers
          (Id_manufacturer serial PRIMARY KEY,
           Title varchar (50) NOT NULL,
           Country varchar(50) not NULL,
           City varchar(50) not NULL,
           Phone varchar(12) not NULL);


CREATE TABLE Clients
          (Id_client serial PRIMARY KEY,
           Name varchar(20) not NULL,
           Surname varchar(20) not NULL,
           Patronymic varchar(20)  NULL,
           Passport int not null,
           Telephone varchar(12) not NULL);



CREATE TABLE Employees
          (Id_employee serial PRIMARY KEY,
           Name varchar(20) not NULL,
           Surname varchar(20) not NULL,
           Patronymic varchar(20)  NULL,
           Position varchar(20) not null,
           Telephone varchar(12) not NULL);


CREATE TABLE Automobiles
       (Id_auto serial PRIMARY KEY,
```

```
            Model varchar(50) not NULL,

            Color varchar(50) not NULL,

            Transmission varchar(15) not NULL,

            Car_body varchar(20) not NULL,

            Price int not null,

            Id_manufacturer int NULL REFERENCES Manufacturers
(Id_manufacturer) on delete set null);




CREATE TABLE Orders

            (Id_order serial PRIMARY KEY,

             Id_client int NULL REFERENCES Clients  (Id_client) on
delete set null,

             Id_auto int NULL REFERENCES Automobiles  (Id_auto) on
delete set null,

             Id_employee int NULL REFERENCES Employees  (Id_employee) on
delete set null,

             Contract_date date NOT NULL,

             Payment_type varchar(10) not NULL);
```

ПРИЛОЖЕНИЕ Б

Исходный код модулей

Модуль «Клиент» (“Client”):

```
package factory.entity;

public class Client {

    private int id_client;

    private String name;

    private String surname;

    private String patronymic;

    private int passport;

    private String telephone;


    public Client() { }


    public  Client(String name, String surname, String patronymic, int passport, String telephone){

        this.name =name;

        this.surname = surname;

        this.patronymic = patronymic;

        this.passport =passport;

        this.telephone = telephone;

    }

    public  Client(int id_client, String name, String surname, String patronymic, int passport, String telephone){

        this.id_client = id_client;

        this.name =name;

        this.surname = surname;

        this.patronymic = patronymic;
```

```
        this.passport =passport;
        this.telephone = telephone;
    }

    public int getId_client() {
        return id_client;
    }
    public void setId_client(int Id_client) {
        this.id_client = Id_client;
    }

    public String getName() {
        return name;
    }
    public void setName(String Name) {
        this.name = Name;
    }

    public String getSurname() {
        return surname;
    }
    public void setSurname(String Surname) {
        this.surname = Surname;
    }

    public String getPatronymic() {
        return patronymic;
    }
```

```
    public void setPatronymic(String Patronymic) {
        this.patronymic = Patronymic;
    }


    public String getTelephone() {
        return telephone;
    }
    public void setTelephone(String Telephone) {
        this.telephone = Telephone;
    }
    public int getPassport() {
        return passport;
    }

    public void setPassport(int Passport) {
        this.passport = Passport;
    }
}
```

ПРИЛОЖЕНИЕ В

JSP

clientList.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ page isELIgnored="false" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>

    <title>Auto Dealer</title>
    <style>
        #container {
            width: 800px; /* Ширина макета */
            margin: 0 auto; /* Выравнивание по центру */
        }

        th.sorted[data-order="1"],
        th.sorted[data-order="-1"] {
            position: relative;
        }

        th.sorted[data-order="1"]::after,
        th.sorted[data-order="-1"]::after {
            right: 8px;
            position: absolute;
        }

        th.sorted[data-order="-1"]::after {
            content: "▼"
        }
```

```
        th.sorted[data-order="1"]::after {
            content: "▲"
        }
        th {
            cursor: pointer;
        }
    </style>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap
.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.m
in.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
    <script>
        document.addEventListener('DOMContentLoaded', () => {

            const getSort = ({ target }) => {
                const order = (target.dataset.order = -
(target.dataset.order || -1));
                const index =
[...target.parentNode.cells].indexOf(target);
                const collator = new Intl.Collator(['en', 'ru'], {
numeric: true });
                const comparator = (index, order) => (a, b) => order *
collator.compare(
                    a.children[index].innerHTML,
                    b.children[index].innerHTML
                );

                for(const tBody of target.closest('table').tBodies)
```

```
tBody.append(...[...tBody.rows].sort(comparator(index, order)));

                for(const cell of target.parentNode.cells)
                    cell.classList.toggle('sorted', cell === target);
            };

            document.querySelectorAll('.table-striped
thead').forEach(tableTH => tableTH.addEventListener('click', () =>
getSort(event)));

        });
    </script>
    <script>function tableSearch() {
        var phrase = document.getElementById('search-text');
        var table = document.getElementById('info-table');
        var regPhrase = new RegExp(phrase.value, 'i');
        var flag = false;
        for (var i = 1; i < table.rows.length; i++) {
            flag = false;
            for (var j = table.rows[i].cells.length - 2; j >= 0; j--)
{
                flag =
regPhrase.test(table.rows[i].cells[j].innerHTML);
                if (flag) break;
            }
            if (flag) {
                table.rows[i].style.display = "";
            } else {
                table.rows[i].style.display = "none";
            }

        }
    }</script>
```

```
</head>
<body>
<div class="jumbtron">
    <div class="container-fluid">
        <div class="row">
            <div class="col-md-2" style="top: 20px">
                <center>
                    <form action="/">
                        <button class="btn btn-outline-primary"
>Производители</button>
                    </form>
                    <form action="/client" >
                        <button class="btn btn-outline-primary"
style="margin-top: 10px">Клиенты</button>
                    </form>
                    <form action="/auto">
                        <button class="btn btn-outline-primary"
style="margin-top: 10px">Автомобили</button>
                    </form>
                    <form action="/order">
                        <button class="btn btn-outline-primary"
style="margin-top: 10px">Заказы</button>
                    </form>
                    <form action="/employee">
                        <button class="btn btn-outline-primary"
style="margin-top: 10px">Сотрудники</button>
                    </form>
                </center>
            </div>

            <div class="col-md-8">
                <center>
                    <h1>Clients</h1>
                    <h2>
```

```
                        <a href="/client/new">Add New Client</a>
                           
                        <a href="/client/list">List All Clients</a>
                    </h2>
                </center>
                <div id="container">
                    <div class="col-2" align="left">
                        <input class="form-control" type="text"
placeholder="Search" id="search-text" onkeyup="tableSearch()">
                    </div>
                    <br>
                    <table class="table table-striped" id="info-
table">
                        <thead>
                        <tr>
                            <th>Name</th>
                            <th>Surname</th>
                            <th>Patronymic</th>
                            <th>Passport</th>
                            <th>Telephone</th>
                        </tr>
                        </thead>
                        <tbody>
                        <c:forEach var="client" items="${clientList}">
                            <tr>
                                <td><c:out value="${client.name}"
/></td>
                                <td><c:out value="${client.surname}"
/></td>
                                <td><c:out
value="${client.patronymic}" /></td>
                                <td><c:out value="${client.passport}"
/></td>
```

```
                                <td><c:out value="${client.telephone}"
/></td>
                                <td>
                                    <a
href="/client/edit?id_client=<c:out value='${client.id_client}'
/>">Edit</a>
                                        
                                    <a
href="/client/delete?id_client=<c:out value='${client.id_client}' />"
class="btn btn-sm btn-outline-danger">X</a>
                                </td>
                            </tr>
                        </c:forEach>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div></div>
</body>
</html>
```