

Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Модели данных и системы управления базами данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

«Интернет площадка услуг по аренде легковых автомобилей (у
различных владельцев)»

Студент гр. 753501
Харлап Н. Ю.
Руководитель
Удовин И. А.

Минск 2020

Содержание

Введение.....	3
1. Анализ предметной области	4
1.1. Краткие теоретические сведения.....	4
1.2. Технологии и средства разработки	5
2. Моделирование предметной области.....	9
2.1. Разработка модели данных.....	10
2.2. Разработка логической модели данных	12
3. Проектирование приложения	17
3.1. Диаграммы классов	17
3.2. Диаграмма состояний	18
3.3. Диаграмма активности	19
3.4. Диаграмма компонентов.....	20
4. Методика использования программного средства	21
Заключение.....	29
Приложение 1. Исходные файлы	31

Введение

Интернет представляет собой идеальный источник для получения информации, а также является великолепным инструментом для коммуникации и построения собственного бизнеса. Другими словами, интернет предоставляет для человека большие возможности, которые при правильном использовании могут пойти ему на пользу. Допустим вам необходимо арендовать автомобиль на какое-то время или вы хотите сдать свой автомобиль в аренду, например, из-за того, что вы уезжаете в другую страну на некоторое время. И данное программное средство может помочь вам в этом. Тема моей работы: «Интернет площадка услуг по аренде легковых автомобилей (у различных владельцев)», своего рода интернет-площадка, на которой пользователи просматривать доступные автомобили, брать в аренду или сдавать свой автомобиль в аренду.

1. Анализ предметной области

1.1. Краткие теоретические сведения

База данных (БД) — это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов. Базы данных активно используются для динамических сайтов со значительными объемами данных — часто это интернет-магазины, порталы, корпоративные сайты. В контексте баз данных стоит рассмотреть понятие СУБД.

Система управления базами данных (СУБД) — это комплекс программных средств, необходимых для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации. Наиболее распространенными СУБД являются MySQL, PostgreSQL, Oracle, Microsoft SQL Server [1].

Реляционная база данных — это набор данных с предопределенными связями между ними. Эти данные организованы в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке — значение атрибута. Каждая строка таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности. Каждая строка в таблице может быть помечена уникальным идентификатором, называемым первичным ключом, а строки из нескольких таблиц могут быть связаны с помощью внешних ключей. К этим данным можно получить доступ многими способами, и при этом реорганизовывать таблицы БД не требуется [2].

Аренда — имущественный наем; договор, по которому собственник (арендодатель) передает арендатору в срочное владение и пользование имущество за соответствующую арендную плату. Различают краткосрочную аренду (рентинг), среднесрочную аренду (хайринг) и долгосрочную аренду (лизинг) [3].

Арендатор — физическое или юридическое лицо, взявшее на определенных условиях во временное пользование не принадлежащие ему средства производства с целью получения дохода.

Арендодатель — дилер или другая организация, которая сдает средства производства в аренду клиенту.

Арендная плата — денежная оплата права пользования арендуемым имуществом.

1.2. Технологии и средства разработки

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight[4](см. рис. 1.1.).



Рисунок 1.1. Иконка Visual Studio 2019

C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML (см. рис. 1.2)[5].



Рисунок 1.2. Иконка C#

HTML5 (HyperText Markup Language) — язык для структурирования и представления содержимого всемирной паутины. Это пятая версия HTML. Хотя стандарт был завершён (рекомендованная версия к использованию) только в 2014 году (предыдущая, четвёртая, версия опубликована в 1999 году), уже с 2013 года браузерами оперативно осуществлялась поддержка, а разработчиками — использование рабочего стандарта (англ. HTML Living

Standard). Цель разработки HTML5 — улучшение уровня поддержки мультимедиа-технологий с одновременным сохранением обратной совместимости, удобочитаемости кода для человека и простоты анализа для парсеров (см. рис. 1.3) [6].



Рисунок 1.3. Иконка HTML5

CSS3 (Cascading Style Sheets) — формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML). Также может применяться к любым XML-документам, например, к SVG или XUL. CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS) (см. рис. 1.4) [7].



Рисунок 1.4. Иконка CSS3

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией стандарта ECMAScript (стандарт ECMA-262). JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. Основные архитектурные черты: динамическая типизация, слабая типизация,

автоматическое управление памятью, прототипное программирование, функции как объекты первого класса. (см. рис. 1.5) [8].



Рисунок 1.5. Иконка JavaScript

MySQL Workbench — инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL. Является преемником DBDesigner 4 от FabForce. Возможности программы (см. рис. 1.6):

- Позволяет наглядно представить модель базы данных в графическом виде;
- Наглядный и функциональный механизм установки связей между таблицами, в том числе «многие ко многим» с созданием таблицы связей;
- Reverse Engineering — восстановление структуры таблиц из уже существующей на сервере БД (связи восстанавливаются в InnoDB, при использовании MyISAM — связи необходимо устанавливать вручную);
- Удобный редактор SQL запросов, позволяющий сразу же отправлять их серверу и получать ответ в виде таблицы;
- Возможность редактирования данных в таблице в визуальном режиме [9].



Рисунок 1.6. Иконка MySQL Workbench

Microsoft Azure – облачная платформа компании Microsoft.

Предоставляет возможность разработки, выполнения приложений и хранения данных на серверах, расположенных в распределённых дата-центрах. Облако Azure было анонсировано в октябре 2008 года под кодовым названием “Project Red Dog”. Релиз состоялся 1 февраля 2010 года под названием “Windows Azure”. В 2014 году платформа была переименована в Microsoft Azure. Microsoft Azure реализует облачные модели платформы как сервиса (PaaS) и инфраструктуры как сервиса (IaaS). Возможно использование как сторонних, так и сервисов Microsoft в качестве модели ПО как сервиса (SaaS). Работоспособность платформы Microsoft Azure обеспечивает глобальная сеть распределённых дата-центров Microsoft. Помимо базовых функций операционных систем, Microsoft Azure имеет и дополнительные: выделение ресурсов по требованию для масштабирования, автоматическую синхронную репликацию данных для повышения отказоустойчивости, обработку отказов инфраструктуры для обеспечения постоянной доступности и другое(см. рис. 1.7)[9].



Рисунок 1.7. Иконка Azure

2. Моделирование предметной области

Функциональную модель предметной области представим в виде диаграммы вариантов использования в нотации UML, представляющей систему в виде набора варианта использования и актеров, взаимодействующих с ними.

В рамках предметной области можно выделить четырёх актеров:

- Администратор;
- Пользователь;
- Менеджер;
- Поставщик;

Диаграмма вариантов использования подсистемы Web-сайт приведена на рисунке 2.1.

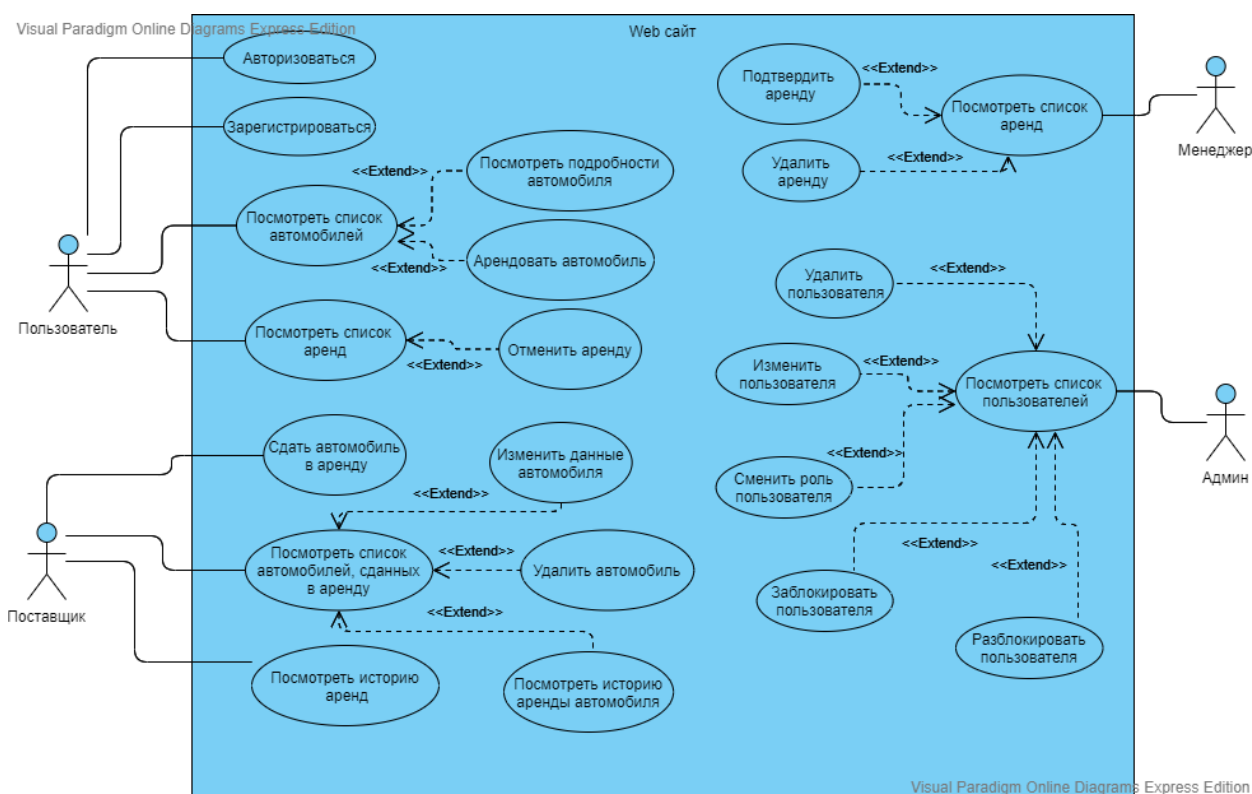


Рисунок **Ошибка! Текст указанного стиля в документе отсутствует..1** –
Диаграмма вариантов использования

Согласно приведенной диаграмме пользователь применяет следующие варианты использования:

- посмотреть список автомобилей;
- зарегистрироваться;
- ввести паспортные данные;

- арендовать автомобиль;
- отменить аренду автомобиля;
- посмотреть список своих аренд;
- поиск автомобиля по критерию;

Поставщик применяет следующие варианты использования:

- зарегистрироваться;
- ввести юридическую информацию;
- сдать автомобиль в аренду;
- изменить условия аренды автомобиля;
- удалить автомобиль из аренды;
- посмотреть историю аренды автомобиля;

Менеджер применяет варианты использования: посмотреть список всех аренд автомобилей, их отмена и удаление.

Администратор применяет варианты использования, доступные менеджеру, а также дополнительные варианты использования – изменить, удалить, сменить роль, заблокировать пользователя.

Также на диаграмме приведен вспомогательный вариант использования «Авторизоваться», реализующий защищенный доступ к данным системы на основании парольного доступа.

2.1. Разработка модели данных

Разработка модели данных включает разработку классов, обеспечивающих взаимодействие с базой данных (рисунок 3.1).

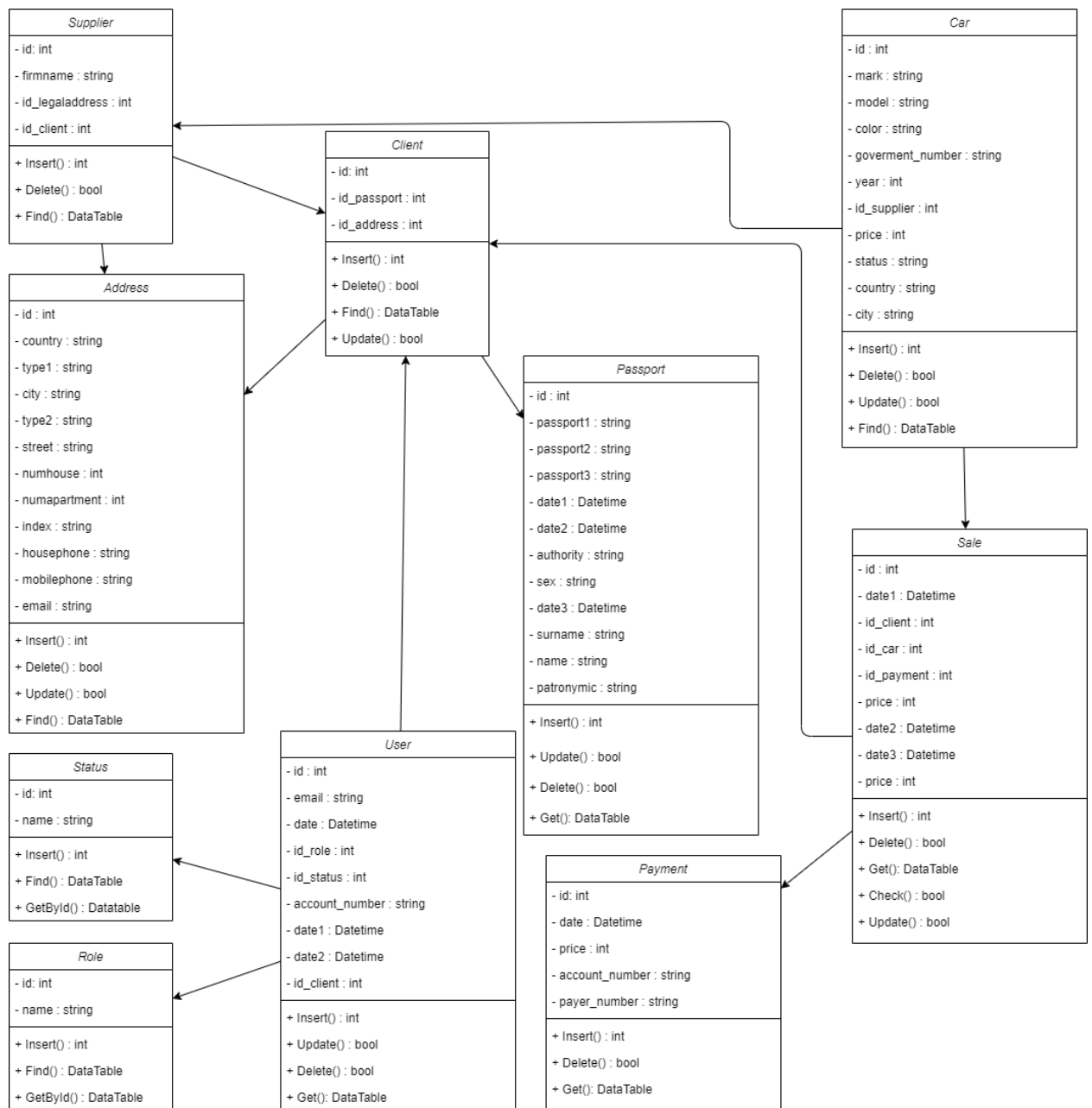


Рисунок 2.2. – Модель данных

Модель данных включает классы, соответствующие сущностям базы данных.

Рассмотрим типичные методы классов:

Get() – возвращает таблицу данных;

Insert() – функция вставки данных в таблицу БД;

Update() – функция обновления данных в таблице БД;

Delete() – функция удаления данных из БД;

Find() – функция поиска по критерию данных в БД;

2.2. Разработка логической модели данных

Проведенный анализ предметной области позволил выявить следующий набор сущностей:

- адреса (address);
- автомобили (car);
- клиенты (client);
- паспортные данные (passport);
- оплаты (payments);
- роли (role);
- заказы (sale);
- статусы (status);
- поставщики (supplier);
- пользователи (users);

Схема данных в нотации IDEF1.X – логическая ER-модель, приведенная на рисунке 2.1, показывает сущности, их атрибуты, связи между сущностями, первичные и внешние ключи.

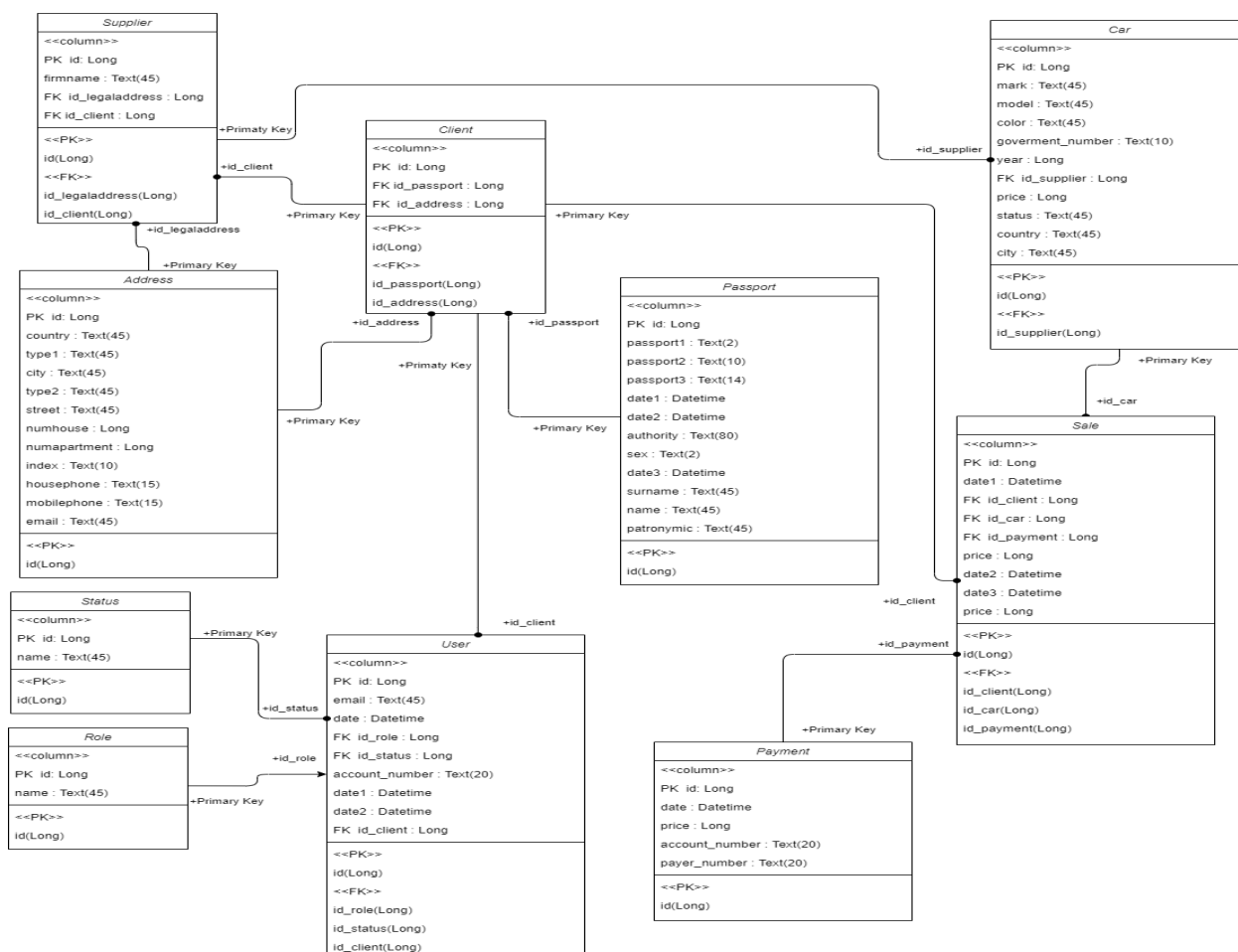


Рисунок 2.3 – Схема логической модели данных

Между сущностями установлены следующие отношения:

- адрес относится к одному клиенту, одному клиенту соответствует один адрес (1 : 1);
- адрес относится к одному поставщику, одному поставщику соответствует один адрес (1 : 1);
- автомобиль относится к одному поставщику, одному поставщику могут соответствовать много автомобилей (1 : M);
- автомобиль относится к одному заказу, одному заказу соответствует один автомобиль (1 : 1);
- паспорт относится к одному клиенту, одному клиенту соответствует один паспорт (1 : 1);
- оплата относится к одному заказу, одному заказу соответствует много оплат (1 : M);
- пользователь относится к одному клиенту, одному клиенту соответствует один пользователь (1 : 1);

– заказ относится к одному клиенту, одному клиенту могут соответствовать много заказов (1 : M);

– пользователь относится к одному статусу, одному статусу могут соответствовать много пользователей (1 : M);

– пользователь относится к одной роли, одной роли могут соответствовать много пользователей (1 : M);

– поставщик относится к одному клиенту, одному клиенту соответствует один поставщик (1 : 1).

Рассмотрим атрибуты сущностей.

Сущность «Адрес» (address) включает следующие атрибуты:

– код адреса (id) – уникальный идентификатор адреса, первичный ключ;

– страна (country) – название страны

– тип населённого пункта (type1) – город, посёлок, деревня и т.д.

– город (city) – название города

– тип улицы (type2) – улица, проспект, проезд и т.д.

– улица (street) – название улицы

– номер дома (numhouse) – номер дома адреса

– номер квартиры (numapartment) – номер квартиры адреса

– индекс (index) – индекс адреса

– телефон (housephone) – телефон

– мобильный телефон (mobilephone) – мобильный телефон

– почта (email) – почта

Сущность «Автомобиль» (car) включает следующие атрибуты:

– код автомобиля (id) – уникальный идентификатор автомобиля, первичный ключ;

– марка (mark) – марка автомобиля;

– модель (model) – модель автомобиля;

– госномер(government_number) – государственный номер автомобиля;

– год (year) – год выпуска автомобиля;

– код поставщика (id_supplier) – ссылка на идентификатор поставщика, внешний ключ;

– цена (price) – стоимость автомобиля за час;

– статус (status) – статус автомобиля;

– страна (country) - страна расположения автомобиля;

– город (city) - город расположения автомобиля;

Сущность «Клиент» (client) включает следующие атрибуты:

- код клиента (id) – уникальный идентификатор клиента, первичный ключ;
- код паспорта (id_passport) – ссылка на идентификатор паспорта, внешний ключ;
- код адреса (id_address) – ссылка на идентификатор адреса, внешний ключ.

Сущность «Паспорт» (passport) включает следующие атрибуты:

- код паспорта (id) – уникальный идентификатор паспорта, первичный ключ;
- серия паспорта (passport1) – серия паспорта пользователя;
- номер паспорта (passport2) – номер паспорта;
- личный номер (passport3) – личный номер паспорта;
- дата1 (date1) – дата выдачи паспорта;
- дата2 (date2) – срок действия паспорта;
- орган (authority) – орган, который выдал паспорт;
- пол (sex) – пол(м/ж);
- дата3 (date3) – дата рождения;
- фамилия (surname) – фамилия;
- имя (name) – имя;
- отчество (patronymic) – отчество;

Сущность «Оплата» (payment) включает следующие атрибуты:

- код оплаты (id) – уникальный идентификатор оплаты, первичный ключ;
- дата (date) – дата оплаты;
- цена (price) – стоимость оплаты;
- номер (account_number) – номер оплаты;
- номер плательщика (payer_number) – номер плательщика;

Сущность «Роль» (role) включает следующие атрибуты:

- код роли (id) – уникальный идентификатор роли, первичный ключ;
- название (name) – название роли;

Сущность «Продажа» (sale) включает следующие атрибуты:

- код актива (id) – уникальный идентификатор аренды, первичный ключ;
- дата1 (date1) – дата совершения сделки;
- код клиента (id_client) – ссылка на идентификатор клиента, внешний ключ;

- код автомобиля (id_car) – ссылка на идентификатор автомобиля, внешний ключ.

- код оплаты (id_payment) – ссылка на идентификатор оплаты, внешний ключ.

- дата2 (date2) – дата начала аренды;

- дата3 (date3) – дата окончания аренды;

- цена (price) – стоимость аренды;

Сущность «Статус» (status) включает следующие атрибуты:

- код статуса (id) – уникальный идентификатор статуса, первичный ключ;

- название (name) – название статуса;

Сущность «Поставщик» (supplier) включает следующие атрибуты:

- код поставщика (id) – уникальный идентификатор поставщика, первичный ключ;

- название (firmname) – название фирмы;

- код адреса (id_legaladdress) – ссылка на идентификатор адреса, внешний ключ;

- УНН (unn) – УНН поставщика.

- код клиента (id_client) – ссылка на идентификатор клиента, внешний ключ.

Сущность «Пользователи» (user) включает следующие атрибуты:

- код пользователя (id) – уникальный идентификатор пользователя, первичный ключ;

- почта (email) – адрес электронной почты пользователя;

- пароль (password) – пароль пользователя для входа в систему.

- дата регистрации (dateofregistration) – дата регистрации пользователя;

- код роли (id_role) – ссылка на идентификатор роли, внешний ключ.

- код статуса (id_status) – ссылка на идентификатор статуса, внешний ключ.

- дата1 (date_beginblock) – дата начала блокировки;

- дата2 (date_endblock) – дата окончания блокировки;

- код клиента (id_client) – ссылка на идентификатор клиента, внешний ключ.

В силу большого количества связей между сущностями будем использовать реляционную базу данных.

3. Проектирование приложения

3.1. Диаграммы классов

Разработаем диаграммы классов приложения с разделением на контроллеры.

Диаграмма классов приложения приведена на рисунке 3.1.

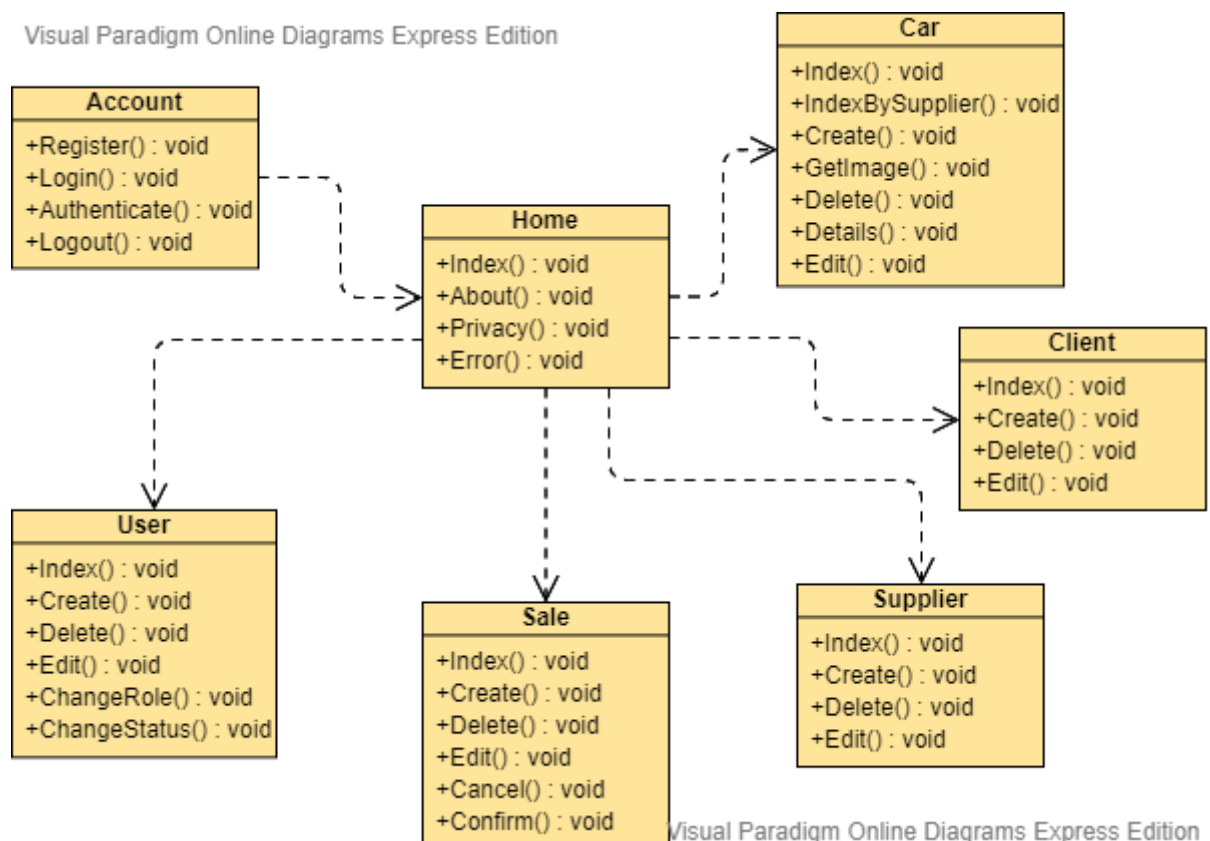


Рисунок 3.2. – Диаграмма классов

На диаграмме, кроме классов модели данных, представлены следующие классы:

Account – класс для работы с аккаунтом;

Home – класс для главной страницы приложения;

Car – класс для страницы автомобилей и работы с ними;

Client – класс для страницы клиентов и работы с ними;

Supplier – класс для страницы поставщиков и работы с ними;

Sale – класс для страницы аренд и работы с ними;

User – класс для страницы пользователей и работы с ними;

3.2. Диаграмма состояний

Далее разработаем диаграмму состояний приложения (рисунок 3.2).

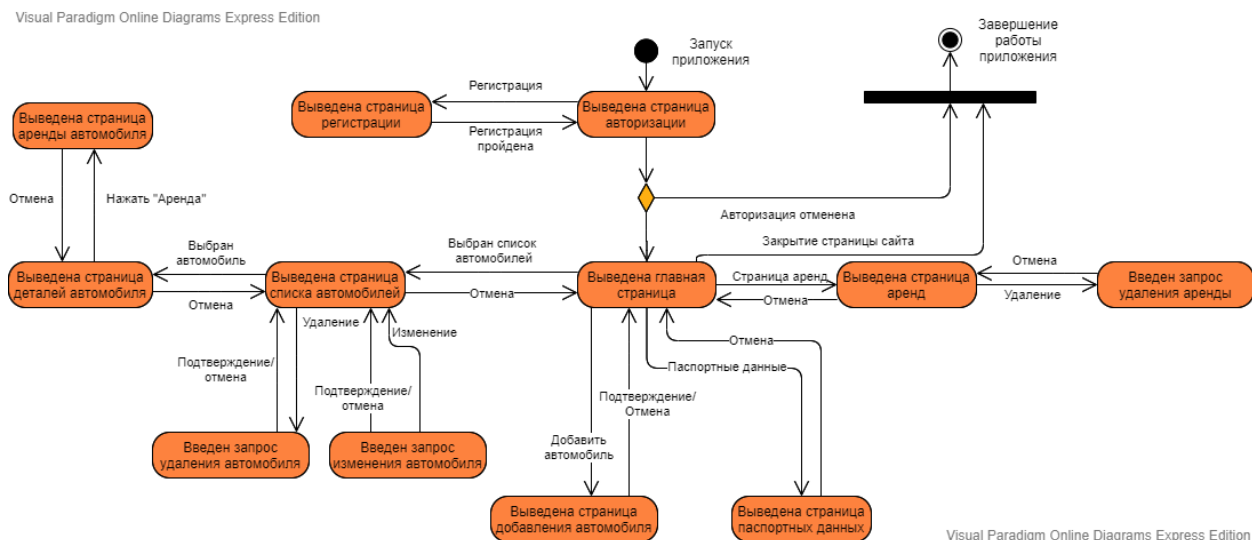


Рисунок 3.3 – Диаграмма состояний приложения

После запуска приложения система принимает состояние «Выведена страница авторизации».

Пользователь может выбрать “Регистрация”, после чего приложение переходит в состояние “Выведена форма регистрации”.

В случае прохождения авторизации переходим в состояние «Выведена главная страница». В случае отмены авторизации система завершает работу.

Дальнейшие состояния зависят от функции или операции, выбранной пользователем.

Если пользователь выбирает операцию посмотреть список автомобилей система переходит в состояние «Выведена страница списка автомобилей» с возвратом при отмене. Далее пользователь может нажать на “Подробности” и система перейдёт в состояние “Выведена страница деталей автомобиля”. На этой странице пользователь может нажать “Аренда”, после чего система перейдет в состояние “Выведена страница аренды автомобиля” или “Отмена” для возврата к состоянию “Выведена страница деталей автомобиля”.

Если пользователь на главной странице выбирает операцию получить список аренд система переходит в состояние «Выведена страница аренд» с возвратом при отмене. На этой странице пользователь может нажать на “Удалить”, после чего система перейдет в состояние “Введен запрос удаления аренды”.

Если пользователь на главной странице выбирает операцию добавить автомобиль система переходит в состояние «Выведена страница добавления автомобиля» с возвратом при отмене или подтверждении. На этой странице пользователь может нажать на “Сохранить”, после чего система перейдет в состояние “Выведена главная страница”.

3.3. Диаграмма активности

Разработаем диаграмму активности для варианта использования «Добавить автомобиль» (рисунок 3.3).

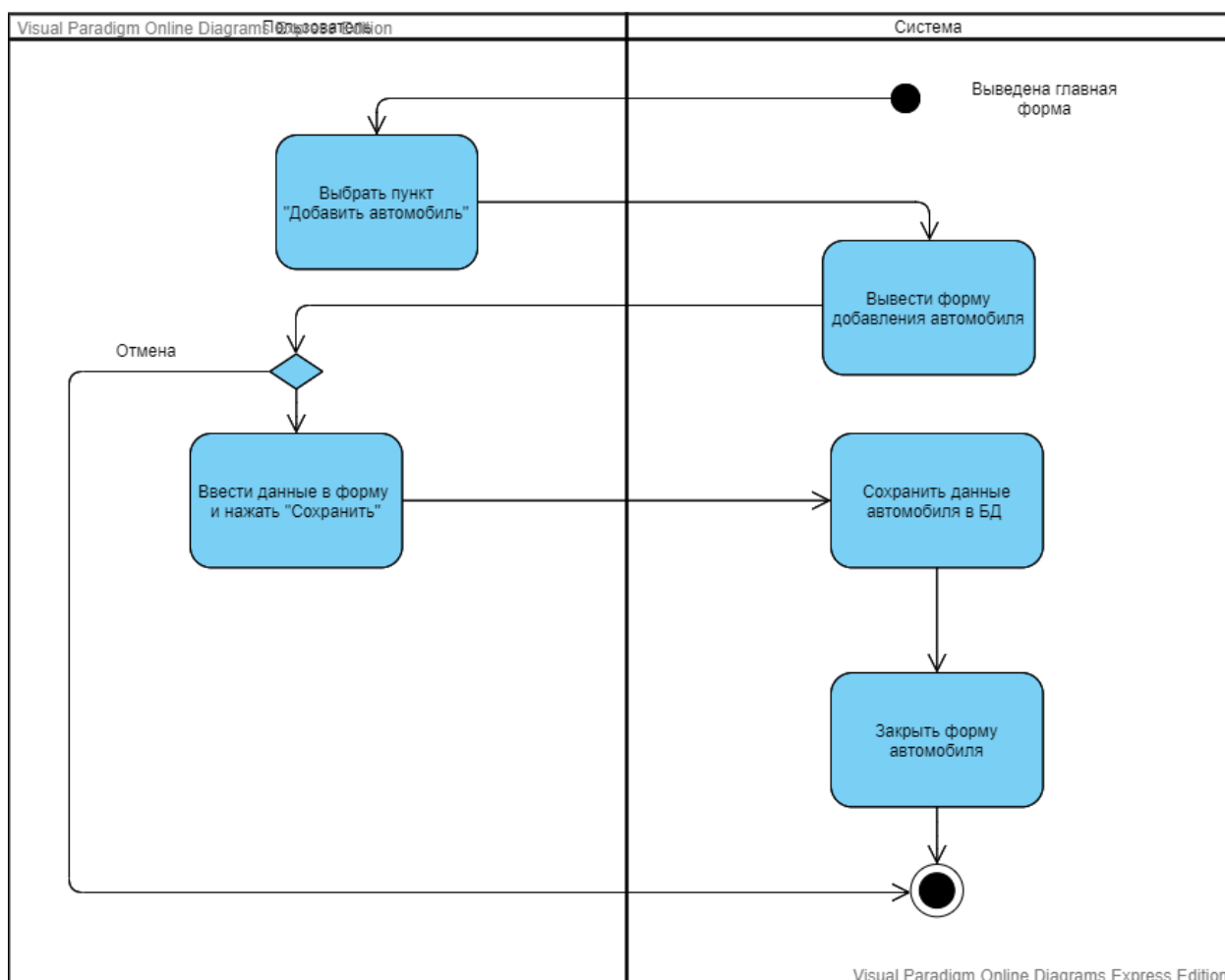


Рисунок 3.4 – Диаграмма активности варианта использования «Добавить автомобиль»

Активность предполагает состояние «Выведена главная форма» и включает следующие действия:

- пользователь нажимает на кнопку «Добавить автомобиль»;

- система выводит форму добавления автомобиля;
- пользователь вводит данные автомобиля (если пользователь отменяет операцию, активность завершается);
- система сохраняет данные автомобиля в БД;
- система закрывает форму добавления автомобиля.

3.4. Диаграмма компонентов

Разработаем диаграмму компонентов системы (рисунок 3.4) этапа разработки.

Visual Paradigm Online Diagrams Express Edition

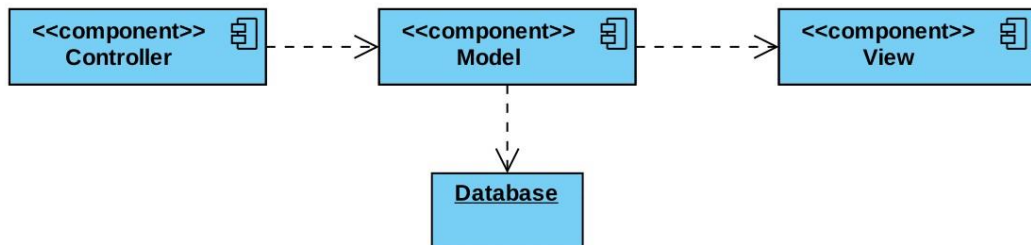


Рисунок 3.5 – Диаграмма компонентов

На этапе разработки система включает следующие компоненты:

Controller – контроллер обеспечивает «связь» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот.

Model - модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность.

View - Представление отвечает за получение необходимых данных из модели и отправляет их пользователю.

Database – компонент базы данных.

4. Методика использования программного средства

Программное средство размещено по ссылке <https://labs2020111132228.azurewebsites.net/Car>. При выполнении входа на сайт, пользователя перенаправит на страницу, содержащую список доступных автомобилей, форму для поиска и кнопки для регистрации/авторизации(см. рис. 4.1).

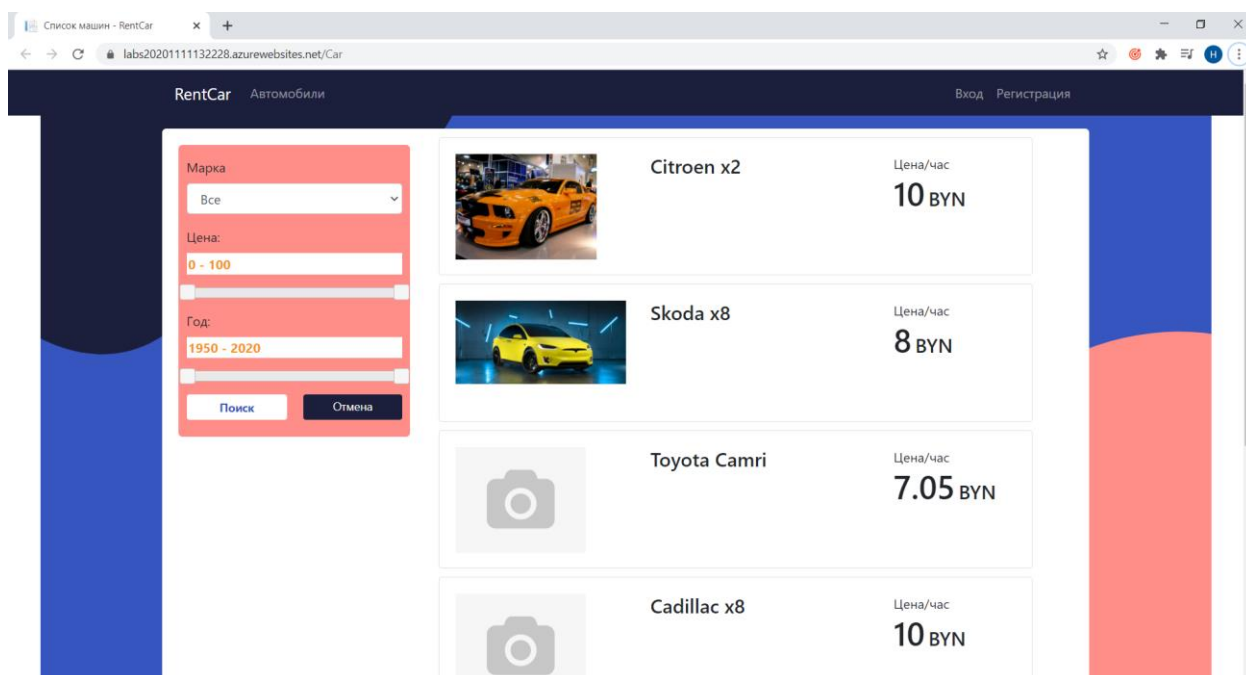


Рисунок 4.1. Главная страница сайта

4.1. Авторизация/Регистрация

Для аренды автомобиля пользователь должен быть зарегистрирован на сайте и выполнить вход в аккаунт. Страница авторизации(см. рис. 4.2) и страница регистрации(см. Рис. 4.3). Для регистрации пользователь должен ввести свою почту и придумать пароль. Также пользователь может зарегистрироваться как поставщик, чтобы иметь возможность сдавать автомобили в аренду.

Вход на сайт

Email

Пароль

Войти

Рисунок 4.2. Вход на сайт

Регистрация

Email

Пароль

Подтверждение пароля

Поставщик ☐

Зарегистрироваться

Рисунок 4.3. Регистрация на сайте

4.2. Аренда автомобиля

Для взятия автомобиля в аренду пользователь должен войти в аккаунт, как обычный пользователь. На главной странице пользователь может просматривать список всех автомобилей, искать автомобили по фильтру. Чтобы взять автомобиль в аренду пользователь должен нажать на фотографию понравившегося автомобиля, появится страница с подробностями об автомобиле и кнопкой для аренды(см. рис. 4.4.).

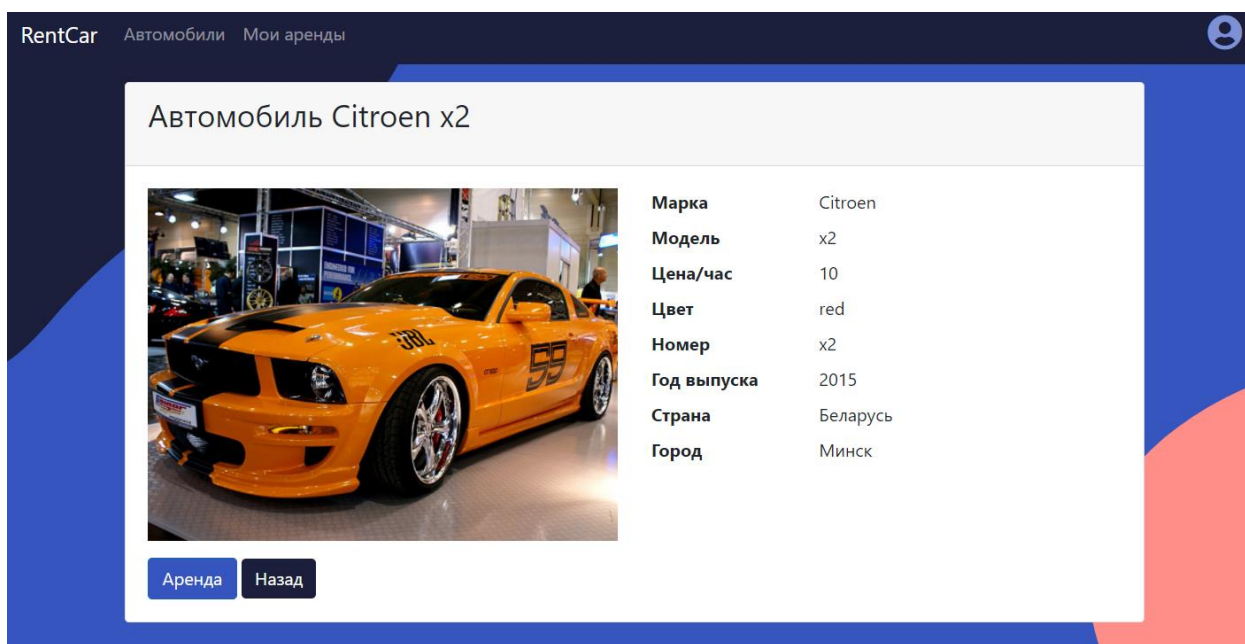


Рисунок 4.4. Подробности об автомобиле

Если пользователь не заполнил свои личные данные, то при нажатии на кнопку аренды его перенаправит на страницу для их заполнения.(см. рис. 4.5.)

Паспортные данные

Серия паспорта Номер паспорта

Личный номер паспорта

Дата выдачи паспорта Срок действия паспорта

Орган, который выдал паспорт

Дата рождения Пол

Фамилия

Имя

Рисунок 4.5. Заполнение личной информации

Если пользователь ранее заполнил свои личные данные, то при нажатии на кнопку аренды его перенаправит на страницу аренды данного автомобиля(см. рис. 4.6.).

Оформление аренды

Стоимость аренды/час = 10

Начало аренды Окончание аренды

Сумма к оплате

Рисунок 4.6. Оформление аренды

Пользователь может посмотреть список своих аренд. Для этого на главной странице ему надо выбрать вкладку “Мои аренды”. Появится список аренд пользователя.(см. рис. 4.7.) На этой странице пользователь может изменить, отменить или оплатить аренду.

Автомобиль	Начало аренды	Конец аренды	Цена	Статус	
42	17-11-2020 21:07	27-11-2020 21:07	7200	Обрабатывается	Изменить Оплатить Отменить
41	17-11-2020 21:19	27-11-2020 02:19	22100	Обрабатывается	Изменить Оплатить Отменить
44	21-01-2021 21:21	28-02-2021 21:21	10	Обрабатывается	Изменить Оплатить Отменить

Назад

Рисунок 4.7. Список аренд пользователя

При нажатии на оплату аренды, пользователя перенаправит на страницу выбора способа оплаты(см. рис. 4.8.).

Оплата аренды автомобиля Skoda x8 в период с 17.11.2020 21:07 по 27.11.2020 21:07

Сумма:

7200

Способ оплаты: ☐ Яндекс.Деньгами ☐ Банковской картой

Оплатить Назад

Рисунок 4.8. Выбор способа оплаты

При нажатии на кнопку “Оплатить”, пользователя перенаправит на сайт uomoney.ru для оплаты аренды.

Если пользователь зарегистрировался как поставщик, то он может добавить автомобиль для аренды. Для этого ему необходимо на главной странице сайта нажать на “Добавить автомобиль”, появится страница для сдачи автомобиля в аренду(см. рис. 4.9.).

Добавление автомобиля

Марка: Модель:

Цвет: Год:

Гос. Номер:

Цена/час BYN:

Страна: Город:

Фото: Файл не выбран

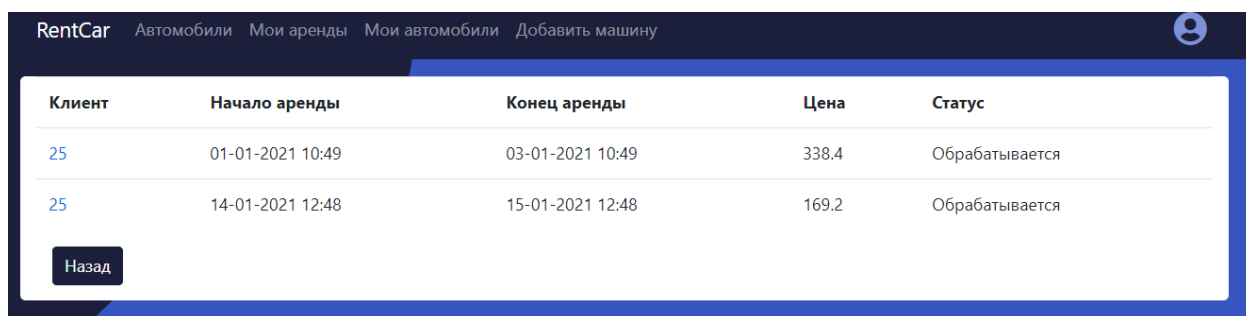
Рисунок 4.9. Добавление автомобиля в аренду

Также поставщик может просматривать список своих автомобилей, для это ему на главной странице надо нажать на “Мои автомобили”, появится страница со списком его автомобилей(см. рис. 4.10.). На этой странице поставщик может посмотреть список своих автомобилей, их статус, историю аренды каждого автомобиля и изменить условия аренды.

Фото	Марка	Модель	Цена/час	Статус	
	Citroen	x2	10	Свободен	<input type="button" value="Изменить"/> <input type="button" value="История"/> <input type="button" value="Удалить"/>
	Skoda	x8	8	Свободен	<input type="button" value="Изменить"/> <input type="button" value="История"/> <input type="button" value="Удалить"/>
	Toyota	Camri	7.05	Свободен	<input type="button" value="Изменить"/> <input type="button" value="История"/> <input type="button" value="Удалить"/>
	Cadillac	x8	10	Свободен	<input type="button" value="Изменить"/> <input type="button" value="История"/> <input type="button" value="Удалить"/>

Рисунок 4.10. Список автомобилей поставщика

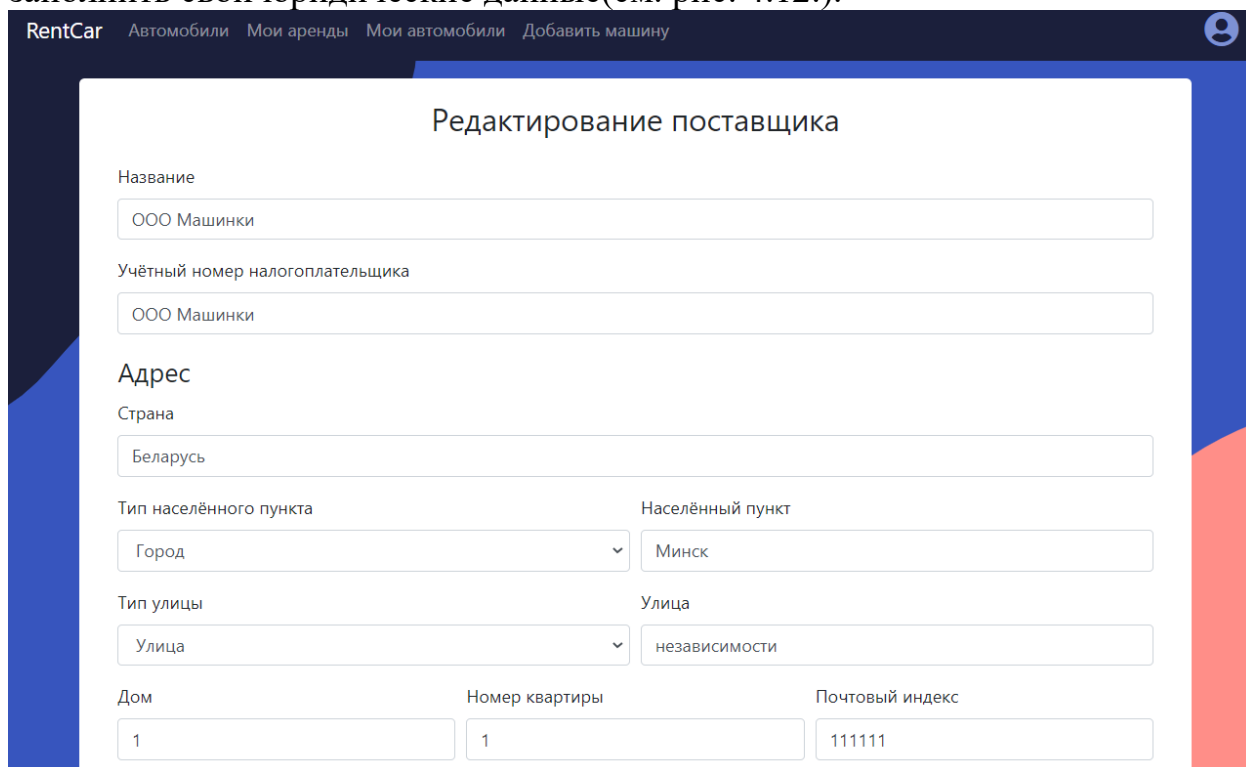
При нажатии на кнопку истории автомобиля появится страница со списком аренд его автомобиля. На этой странице поставщик может посмотреть данные клиента, сроки аренды и статус аренды (см. рис. 4.11.).



Клиент	Начало аренды	Конец аренды	Цена	Статус
25	01-01-2021 10:49	03-01-2021 10:49	338.4	Обрабатывается
25	14-01-2021 12:48	15-01-2021 12:48	169.2	Обрабатывается

Рисунок 4.11. Список аренд автомобиля

Чтобы поставщик имел возможность сдать автомобиль в аренду, он должен заполнить свои юридические данные(см. рис. 4.12.).



Редактирование поставщика

Название
ООО Машинки

Учётный номер налогоплательщика
ООО Машинки

Адрес
Страна
Беларусь

Тип населённого пункта
Город

Населённый пункт
Минск

Тип улицы
Улица

Улица
независимости

Дом
1

Номер квартиры
1

Почтовый индекс
111111

Рисунок 4.12. Юридическая информация поставщика

Если пользователь имеет роль менеджера, то он может просматривать список всех аренд, удалять их или подтверждать.(см. рис. 4.13.)

RentCar		Автомобили	Заказы				
ID Клиента	ID машины	Начало аренды	Конец аренды	Цена	Статус		
22	41	19-11-2020 20:47	21-11-2020 20:47	4800	Отменён	<button>Удалить</button>	
22	41	19-11-2020 20:48	21-11-2020 20:48	4800	Отменён	<button>Удалить</button>	
22	41	19-11-2020 20:49	20-11-2020 20:49	2400	Отменён	<button>Удалить</button>	
22	41	20-11-2020 20:50	21-11-2020 20:50	2400	Отменён	<button>Удалить</button>	
20	42	17-11-2020 21:07	27-11-2020 21:07	7200	Обрабатывается		

Рисунок 4.13. Список всех аренд для менеджера

Заключение

В рамках данной курсовой работы я реализовал программное средство на тему «Интернет площадка услуг по аренде легковых автомобилей (у различных владельцев)» и разместил его на Azure. В программном средстве я использовал систему управления базы данных MySQL, сервер которой также разместил на Azure. Выбор системы управления базой данных упал на MySQL, потому что для проекта требовалась реляционная база данных, простая в использовании, с обширным функционалом и высокой скоростью.

Список использованной литературы

1. Базы данных – Что это такое? [Электронный ресурс]. - Электронные данные. - Режим доступа: (<https://hostiq.ua/wiki/database/>);
2. What is a Relational Database? [Электронный ресурс]. - Электронные данные. - Режим доступа: (<https://aws.amazon.com/relational-database/>);
3. Аренда [Электронный ресурс]. - Электронные данные. - Режим доступа: (<https://ru.wikipedia.org/wiki/Аренда>);
4. Microsoft Visual Studio [Электронный ресурс]. - Электронные данные. - Режим доступа: (https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio);
5. Microsoft Visual Studio [Электронный ресурс]. - Электронные данные. - Режим доступа: (https://ru.wikipedia.org/wiki/C_Sharp);
6. HTML5 [Электронный ресурс]. - Электронные данные. - Режим доступа: (<https://ru.wikipedia.org/wiki/HTML5>);
7. CSS [Электронный ресурс]. - Электронные данные. - Режим доступа: (<https://ru.wikipedia.org/wiki/CSS>);
8. JavaScript [Электронный ресурс]. - Электронные данные. - Режим доступа: (<https://ru.wikipedia.org/wiki/JavaScript>);
9. MySQL Workbench [Электронный ресурс]. - Электронные данные. - Режим доступа: (https://ru.wikipedia.org/wiki/MySQL_Workbench);
10. Microsoft Azure [Электронный ресурс]. - Электронные данные. - Режим доступа: (https://ru.wikipedia.org/wiki/Microsoft_Azure);

Приложение 1. Исходные файлы

AccountController.cs

```
using Labs.Models;
using Labs.ViewModels;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace Labs.Controllers
{
    public class AccountController : Controller
    {
        private readonly UserContext _context;
        public string ConnectionString { get; set; }
        public AccountController(UserContext context)
        {
            _context = context;
        }
        [HttpGet]
        public IActionResult Register()
        {
            return View();
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Register(RegisterViewModel model)
        {
            if (ModelState.IsValid)
            {
                User user = _context.FindUser(model.Email);
                if (user == null)
                {
                    // добавляем пользователя в бд
                    user = new User { Email = model.Email, Password =
UserContext.HashPassword(model.Password) };
                    if (model.issupplier)
                    {
                        Role userRole = _context.FindRole("supplier");
                        if (userRole != null)
                        {
                            user.id_role = userRole.Id;
                        }
                    }
                    else
                    {
                        Role userRole = _context.FindRole("user");
                        if (userRole != null)
                        {
                            user.id_role = userRole.Id;
                        }
                    }

                    Status status = _context.FindStatus("notblock");
                    if (status != null)
                    {

```

```

        user.id_status = status.Id;
    }

    user.dateofregistration = DateTime.Now;
    user.id_client = null;
    user.dateofbeginblock = new DateTime();
    user.dateofendblock = new DateTime();
    _context.AddNewUser(user);

    await Authenticate(user); // аутентификация

    return RedirectToAction("Login");
}
}
else ModelState.AddModelError("", "Некорректные логин и(или) пароль");
return View(model);
}
[HttpGet]
public IActionResult Login()
{
    return View();
}
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Login(LoginViewModel model)
{
    if (ModelState.IsValid)
    {
        (string status, string hash) = _context.Login(model.Email);
        if (hash != null)
        {
            bool result = UserContext.VerifyHashedPassword(hash, model.Password);
            if (result == true)
            {
                if (status != null)
                {
                    if (status == "block")
                    {
                        ModelState.AddModelError("", "Пользователь
заблокирован");
                    }
                    else
                    {
                        await Authenticate(_context.FindUser(model.Email)); //
аутентификация

                        return RedirectToAction("Index", "Home");
                    }
                }
                else ModelState.AddModelError("", "Некорректные логин и(или)
пароль");
            }
            else ModelState.AddModelError("", "Неверный пароль");
        }
        else ModelState.AddModelError("", "Нет такого пользователя");
    }
    return View(model);
}
private async Task Authenticate(User user)
{
    // создаем claims
    List<Claim> claims = new List<Claim>
    {
        new Claim(ClaimsIdentity.DefaultNameClaimType, user.Email),
    }
}

```



```

        new Claim(ClaimsIdentity.DefaultRoleClaimType,
_context.FindRole(user.id_role).Name)
    };
    // создаем объект ClaimsIdentity
    ClaimsIdentity id = new ClaimsIdentity(claims, "ApplicationCookie",
ClaimsIdentity.DefaultNameClaimType, ClaimsIdentity.DefaultRoleClaimType);
    // установка аутентификационных куки
    await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new
ClaimsPrincipal(id));

    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Logout()
    {
        // удаляем аутентификационные куки
        await
HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
        return RedirectToAction("Index", "Home");
    }
}
}

```

CarController.cs

```

using Labs.Models;
using Labs.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Formatters;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore.Migrations;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;

namespace Labs.Controllers
{
    public class CarController : Controller
    {
        private UserContext _context;
        public CarController(UserContext context)
        {
            _context = context;
        }

        public IActionResult Index(string mark, string color, int? year)
        {
            return View(_context.GetAllCars(mark, color, year));
        }

        public IActionResult IndexBySupplier()
        {
            int id_cl = _context.FindUser(User.Identity.Name).id_client.Value;
            int id_supp = _context.FindSupplierByClient(id_cl).Id;

```

```

        return View(_context.GetAllCars(id_supp));
    }

    [Authorize]
    public IActionResult Create()
    {
        int id_cl = _context.FindUser(User.Identity.Name).id_client.Value;
        int id_supp = _context.FindSupplierByClient(id_cl).Id;
        if (id_cl == 0)
        {
            return RedirectToAction("Create", "Client");
        }
        else if (id_supp == 0)
        {
            return RedirectToAction("Create", "Supplier");
        }
        return View();
    }

    [HttpPost]
    public IActionResult Create(CreateCarViewModel modelvm)
    {
        if (ModelState.IsValid)
        {
            int id_cl = _context.FindUser(User.Identity.Name).id_client.Value;
            int id_supp = _context.FindSupplierByClient(id_cl).Id;
            Car car = new Car()
            {
                Mark = modelvm.Mark,
                Model = modelvm.Model,
                Color = modelvm.Color,
                Goverment_number = modelvm.Goverment_number,
                Year = modelvm.Year,
                id_supplier = id_supp,
                Price = modelvm.Price,
                status = "Свободен",
                country = modelvm.country,
                city = modelvm.city
            };

            if (modelvm.Image != null)
            {
                byte[] imageData = null;
                using (var binaryReader = new
BinaryReader(modelvm.Image.OpenReadStream()))
                {
                    imageData = binaryReader.ReadBytes((int)modelvm.Image.Length);
                }
                car.Image = Convert.ToBase64String(imageData); ;
                car.ImageMimeType = modelvm.Image.ContentType;
            }
            if (_context.AddCar(car)) return RedirectToAction("Index", "Home");
            else ModelState.AddModelError("", "Ошибка");
        }
        return View(modelvm);
    }

    public FileContentResult GetImage(int id)
    {
        Car car = _context.FindCar(id);
        if (car != null && car.Image != null && car.ImageMimeType != "")
        {
            byte[] bytes = Convert.FromBase64String(car.Image);

```

```

        return File(bytes, car.ImageMimeType);
    }
    else
    {
        string path = Path.Combine(Directory.GetCurrentDirectory(),
@"wwwroot\images");
        byte[] mas = System.IO.File.ReadAllBytes(path + @"\NoCar.png");
        string file_type = "image/png";
        string file_name = "NoCar.png";
        return File(mas, file_type, file_name);
    }
}

public FileContentResult GetImageFromString(string im,string type)
{
    if (im != null && type != null)
    {
        byte[] bytes = Convert.FromBase64String(im);
        return File(bytes, type);
    }
    else
    {
        string path = Path.Combine(Directory.GetCurrentDirectory(),
@"wwwroot\images");
        byte[] mas = System.IO.File.ReadAllBytes(path + @"\NoCar.png");
        string file_type = "image/png";
        string file_name = "NoCar.png";
        return File(mas, file_type, file_name);
    }
}

public IActionResult Delete(int id)
{
    Car car = _context.FindCar(id);

    if (car != null)
    {
        _context.DeleteCar(id);
        return RedirectToAction("Index", "Home");
    }
    return RedirectToAction("Index", "Home");
}

public ActionResult Details(int id)
{
    Car car = _context.FindCar(id);
    if (car != null)
    {
        return View(car);
    }
    return NotFound();
}

[Authorize]
public IActionResult Edit(int id_car)
{
    Car car = _context.FindCar(id_car);

    byte[] bytes = Convert.FromBase64String(car.Image);
    var stream = new MemoryStream(bytes);
    IFormFile file = new FormFile(stream, 0, bytes.Length, "name", "fileName");

    EditCarViewModel model = new EditCarViewModel

```

```

        {
            Id = car.Id,
            Mark = car.Mark,
            Model = car.Model,
            Color = car.Color,
            Government_number = car.Government_number,
            Year = car.Year,
            id_supplier = car.id_supplier,
            Price = car.Price,
            status = car.status,
            country = car.country,
            city = car.city,
            Image = file,
        };
        return View(model);
    }

    [HttpPost]
    public IActionResult Edit(EditCarViewModel modelvm)
    {
        if (ModelState.IsValid)
        {
            Car car = new Car()
            {
                Id = modelvm.Id,
                Mark = modelvm.Mark,
                Model = modelvm.Model,
                Color = modelvm.Color,
                Government_number = modelvm.Government_number,
                Year = modelvm.Year,
                id_supplier = modelvm.id_supplier,
                Price = modelvm.Price,
                status = modelvm.status,
                country = modelvm.country,
                city = modelvm.city
            };

            if (modelvm.Image != null)
            {
                byte[] imageData = null;
                using (var binaryReader = new
BinaryReader(modelvm.Image.OpenReadStream()))
                {
                    imageData = binaryReader.ReadBytes((int)modelvm.Image.Length);
                }
                car.Image = Convert.ToBase64String(imageData); ;
                car.ImageMimeType = modelvm.Image.ContentType;
                if (_context.UpdateCar(car)) return
RedirectToAction("IndexBySupplier", "Car");
                else ModelState.AddModelError("", "Ошибка");
            }
            else
            {
                if (_context.UpdateCarWithoutImage(car)) return
RedirectToAction("IndexBySupplier", "Car");
                else ModelState.AddModelError("", "Ошибка");
            }
        }
        return View(modelvm);
    }
}
}
}

```

Create.cshtml

```
@model Labs.ViewModels.CreateCarViewModel
```

```
@{ ViewBag.Title = "Добавление машины"; }
```

```
<div class="container sm">
    <div class="container">
        <form asp-action="Create" asp-controller="Car" enctype="multipart/form-data">
            <div class="text-center mb-4">
                <h1 class="h3 mb-3 font-weight-normal">Добавление автомобиля</h1>
            </div>
            <div asp-validation-summary="All" class="text-danger"></div>

            <div class="form-row">
                <div class="form-group col-md-6">
                    <label asp-for="Mark" class="control-label">Марка</label>
                    <select name="mark" asp-items="ViewBag.Marks" class="form-control"
asp-for="Mark"></select>
                </div>
                <div class="form-group col-md-6">
                    <label asp-for="Model" class="control-label">Модель</label>
                    <input type="text" asp-for="Model" class="form-control" required />
                </div>
            </div>

            <div class="form-row">
                <div class="form-group col-md-6">
                    <label asp-for="Color" class="control-label">Цвет</label>
                    <input type="text" asp-for="Color" class="form-control" required />
                </div>
                <div class="form-group col-md-6">
                    <label asp-for="Year" class="control-label">Год</label>
                    <input min="0" max="2020" type="number" asp-for="Year" class="form-
control" required />
                </div>
            </div>

            <div class="form-group">
                <label asp-for="Government_number" class="control-label">Гос.
Номер</label>
                <input type="text" asp-for="Government_number" class="form-control"
required />
            </div>

            <div class="form-group">
                <label asp-for="Price" class="control-label">Цена/час BYN</label>
                <input min="0" max="100" type="number" asp-for="Price" class="form-
control" step="0.01" required />
            </div>

            <div class="form-row">
                <div class="form-group col-md-6">
                    <label asp-for="country" class="control-label">Страна</label>
                    <input type="text" asp-for="country" class="form-control" required />
                </div>
                <div class="form-group col-md-6">
                    <label asp-for="city" class="control-label">Город</label>
                    <input type="text" asp-for="city" class="form-control" required />
                </div>
            </div>

            <div class="form-group">
                <label asp-for="Image" class="control-label">Фото</label>
            </div>
        </form>
    </div>
</div>
```

```

        <input class="form-control-file" type="file" id="imageinput" asp-
for="Image" name="Image" accept="image/*" />
    </div>
    <div class="form-group" id="preview"></div>

    <div class="form-group button-group">
        <input type="submit" value="Добавить" class="btn btn btn-primary" />
        <a asp-controller="Home" asp-action="Index" class="btn btn-primary-
dark">Отмена</a>
    </div>
</form>

</div>
</div>

```

```
@section scripts
```

```

{
    <script type="text/javascript">
        const inputElement = document.getElementById("imageinput");
        inputElement.addEventListener("change", function (e) {
            var file = inputElement.files[0];
            var preview = document.getElementById("preview");
            preview.innerHTML = "";
            var img = document.createElement("img");
            img.classList.add("imgcar1");
            img.file = file;
            preview.appendChild(img);
            var reader = new FileReader();
            reader.onload = (function (aImg) { return function (e) { aImg.src =
e.target.result; }; })(img);
            reader.readAsDataURL(file);
        }, false);

    </script>
}

```