МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ Федеральное государственное автономное образовательное

учреждение высшего образования «Южно-Уральский государственный университет» (национальный исследовательский университет) Высшая школа экономики и управления

Кафедра «Информационные технологии в экономике»

|  |  |  |  |
| --- | --- | --- | --- |
| РАБОТА ПРОВЕРЕНА | | ДОПУСТИТЬ К ЗАЩИТЕ | |
| Рецензент, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | Заведующий кафедрой, д.т.н., с.н.с. | |
| /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ | |  | / Б.М. Суховилов / |
|  |  |  |  |
| «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. | | «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. | |

Разработка сайта аренды вещей

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ – 09.03.03.2020.119. ВКР

Руководитель, к.т.н., доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ В.А. Конов /

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Автор

студент группы ЭиУ–462

/ А.Р. Губайдуллин /

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Нормоконтролер, ст. преподаватель

/ Е.А. Конова /

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020

Челябинск 2020

**АННОТАЦИЯ**

Губайдуллин А.Р. Разработка Web-

приложение сайта аренды

вещей. – Челябинск: ЮУрГУ,

ЭиУ-462, 58 стр., 23 ил., 8 табл., биб-

лиогр. список – 6 наим.

Целью дипломного проекта является разработка Web-приложения, автоматизирующего работу сайта аренды вещей.

В ходе работы поставлены задачи: Удобство эксплуатации, отвечать всем современным тенденциям и условиям рынка, предоставить возможности управления как со стороны арендатора, так и со стороны арендодателя.

Разработано приложение, которое позволяет арендодателю создавать новые объявления, управлять ими, подтверждать движение предмета аренды, выводить средства на счет. Со стороны арендатора приложение позволяет контролировать все взятые в аренду вещи, отмечать их возврат.

В сравнении с другими Web-приложениями, с схожей направленностью, преимущество заключается в большей функциональности, удобстве использования, доступности для внедрения.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc43297797)

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc43297798)

[1.1 Анализ деятельности сайта аренды вещей 5](#_Toc43297799)

[1.2 Анализ существующих систем 6](#_Toc43297800)

[1.3 Сравнительная таблица 8](#_Toc43297801)

[Вывод по разделу один 9](#_Toc43297802)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 11](#_Toc43297803)

[2.1 Разработка структуры ИС 11](#_Toc43297804)

[2.2 Задачи Web-приложения 14](#_Toc43297805)

[2.4 Выбор программных средств 15](#_Toc43297806)

[2.5 Проектирование и применение базы данных 16](#_Toc43297807)

[2.6 Описание структуры таблиц базы данных предметной области 17](#_Toc43297808)

[Вывод по разделу два 21](#_Toc43297809)

[3 ОПИСАНИЕ ПРОГРАММНОГО ПРОДУКТА 22](#_Toc43297810)

[3.1 Общий вид приложения 22](#_Toc43297811)

[3.2 Публикация сайта 46](#_Toc43297812)

[Выводы по разделу три 47](#_Toc43297813)

[ЗАКЛЮЧЕНИЕ 48](#_Toc43297814)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 49](#_Toc43297815)

[ПРИЛОЖЕНИЯ 50](#_Toc43297816)

[ПРИЛОЖЕНИЕ А 50](#_Toc43297817)

[ПРИЛОЖЕНИЕ Б 52](#_Toc43297818)

# ВВЕДЕНИЕ

Сейчас аренда вещей распространена в основном в крупных городах — Москва, Санкт-Петербург, Новосибирск, Екатеринбург, Нижний Новгород, Ростов, Казань. Самый хороший вариант сервиса, по предоставлении гаранта между арендодателем и арендатором, находится в Лондоне, и называется fatlama, но имеет свои недостатки в виде автоматизации и обратной связи с клиентами, а также в расположении для российского рынка, поэтому разработка оптимального приложения для стран СНГ может оказаться востребованным.

Актуальность дипломного проекта в том, что данная отрасль только начинает развиваться и имеется большой потенциал для реализации новых идей.

Цель дипломного проекта – разработка приложения, автоматизирующего работу сайта аренды вещей.

Для достижения поставленной цели решены следующие задачи:

* выполнен анализ предметной области;
* произведен анализ рынка;
* разработаны диаграммы и модели
* разработана структура и сама база данных;
* разработан Web–интерфейс;
* разработан функционал Web-приложения
* произведены тесты приложения

Объектом работы является сайт аренды вещей.

Данный дипломный проект представляет интерес не только для пользователей, но и положительно влияет на экономическую составляющую сервиса.

# 1 ПОСТАНОВКА ЗАДАЧИ

## 1.1 Анализ деятельности сайта аренды вещей

Основными пользователями будут – пользователи, администраторы, модераторы сайта.

Администратор сайта выполняет задачи такие как:

* Назначение модераторов;
* Получение анализа работы сервиса;
* Управления пользователями.

Пользователь-это человек, который пользуется услугами:

* Создание объявления аренды;
* Возможность брать в аренду;
* Получение списка товаров;
* Возможность пополнения счета;
* Возможность снятия денег со счета.

Модератор-человек, следящий за стабильностью работы сайта:

* Просмотр списка пользователей;
* Верификация пользователей;

## 1.2 Анализ существующих систем

Fatllama.com, большая часть пользователей сервиса — физические лица, которые хотят заработать на «пылящихся в чулане вещах». Особым успехом Fat Lama пользуется у сотрудников творческой индустрии: их работа требует дорогостоящего оборудования, а аренда позволяет снизить затраты. Доверие обеспечивает страховка: все сделки покрываются страховщиком XL Catlin, входящим в американский холдинг XL Group. Прежде чем пользоваться платформой, клиент должен заполнить форму регистрации и приложить фото с паспортом или иным удостоверением личности в руках. Всего сервис собирает до 40 пользовательских параметров: имя, адрес, электронная почта, аккаунты в соцсетях и др. По данным Fat Lama, этой информации достаточно, чтобы в 90% случаев оценить риски корректно. В оставшихся 10% у пользователя запрашивают дополнительные данные. Арендатор платит деньги на сайте, деньги замораживаются, после успешной сделки деньги отправляются на счет арендодателя. Нет мобильного приложения.

В основном проект распространен на территории Великобритании. Присутствует поиск по карте в реальном времени. Прибыль компании приносит процент с каждой успешной сделки.

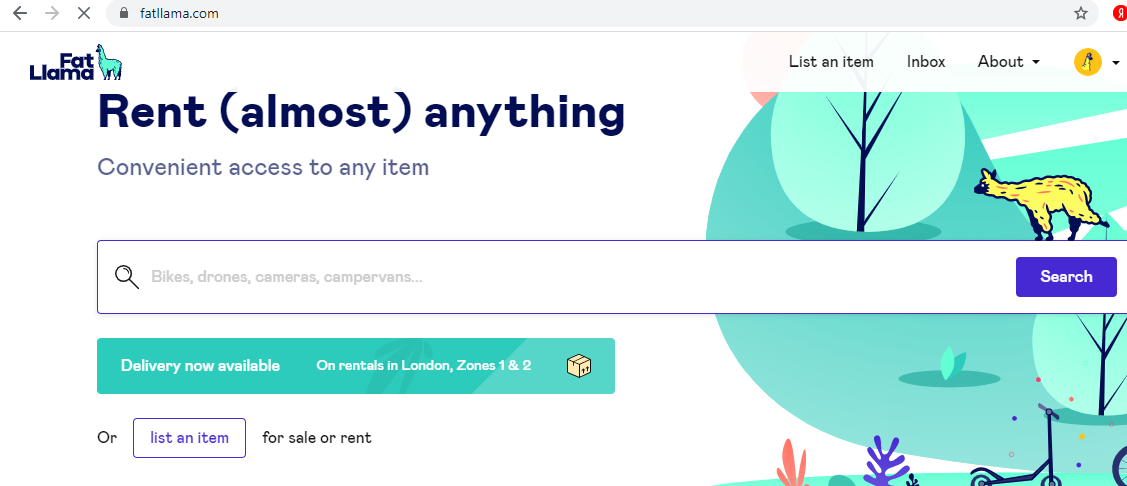


Рисунок 1.1 – Главная страница сайта fatllama.com

https://otr24.com OneTwoRent в основном рассчитан на юридические лица, но также есть возможность и физических. Существует мобильное приложение, которое работает неисправно. Как происходит аренда: Арендатор находит нужный объект на портале, связывается с владельцем, договаривается о передаче объекта в аренду, способе оплаты, об оставленном залоге. Получает вещь и по истечению срока аренды ее возвращает. При этом подписывается договор приема-передачи. Нет рейтинговой системы. Нет поиска по карте. Прибыль приносят внутренние платные функции. Работает на территории РФ.

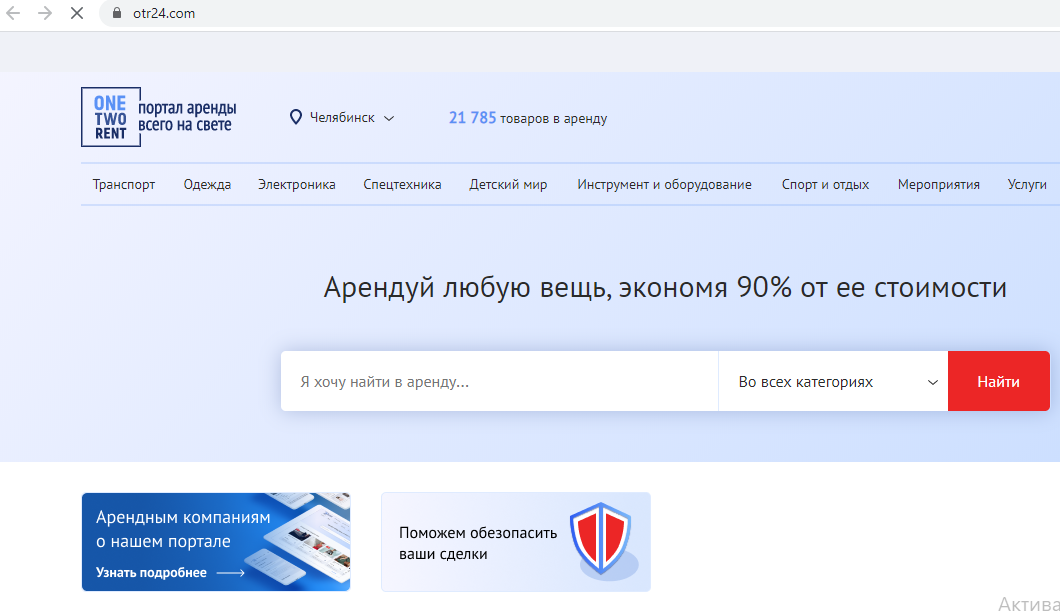


Рисунок 1.2 – Главная страница сайта otr24.com

https://next2u.ru/- большая часть пользователей сервиса-юридические лица, основные товары-одежда. Заполнение всех бумаг происходит на сайте.

Нет мобильного приложения. Прибыль предприятия приходит с каждой успешной сделки. Безопасность сделки лежит на сервисе, он отвечает за подписание всех бумаг. Нет поиска по карте. Работает на территории РФ. Существует рейтинговая система. Есть отзывы.

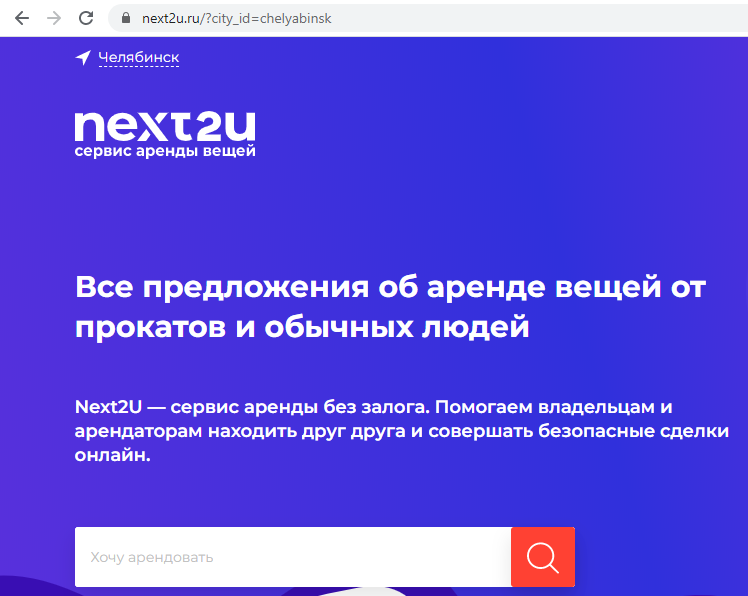


Рисунок 1.3 – Главная страница сайта next2u.ru

## 1.3 Сравнительная таблица

Сравнительный анализ приведен в таблице 1.1.

Таблица 1.1 – сравнительный анализ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | https://fatllama.com/ | https://otr24.com | https://next2u.ru/ | Мой сайт |
| Поиск по карте | + | - | - | + |
| Кто выступает гарантом сделки | Площадка, а также страховая компания | Арендодатели (залог) | Площадка, Арендодатели (залог) | Площадка, в перспективе страховая компания |
| Основная направленность аренды | Всенаправленная площадка | Всенаправленная площадка | Платья, одежда | Всенаправленная площадка |
| Страна использования | Великобритания США | Россия | Россия | Россия |

Продолжение таблицы 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | https://fatllama.com/ | https://otr24.com | https://next2u.ru/ | Мой сайт |
| Типы арендодателей | Физические лица и юридические | В основном Юридические лица | В основном Юридические лица | Физические лица и юридические |
| Мобильное приложение | - | + | - | +, в перспективе |
| Регистрация | До 40 параметров, включая фотографию с паспортом | Минимальное количество параметров, можно регистрироваться через социальные сети, для юридических лиц отдельная регистрация | Минимальное количество параметров, можно регистрироваться через социальные сети, для юридических лиц отдельная регистрация | Много параметров включая фотографию с паспортом |
| Рейтинг | + | - | + | + |
| Прибыль | Процент от каждой сделки | Платные функции | Процент от каждой сделки | Процент от каждой сделки |

В ходе поиска и анализа сайтов по аренде было выявлено, что большинство систем имеют схожий функционал, однако у каждой системы есть свои изъяны.

В результате сравнительного анализа выявлено:

* Отсутствие современных технологий;
* Недостаточный, или плохой сервис;
* Имеется потенциал развития данной области;

# Вывод по разделу один

Цель первого раздела была полностью выполнена:

* Был произведен анализ деятельности сайта по аренде вещей;
* Выполнен анализ существующих систем;
* Изучена предметная область.

# 2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

## 2.1 Разработка структуры ИС

UML (унифицированный язык моделирования) – язык графического описания для объектного моделирования.

Диаграмма прецедентов представлена на рисунке 2.1.

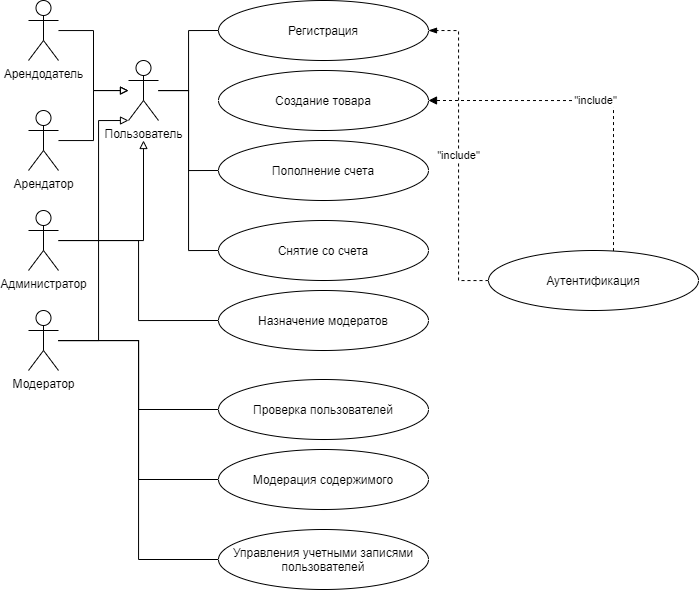


Рисунок 2.1-Диграмма прецедентов

Диаграмма жизнедеятельности представлена на рисунке 2.2.

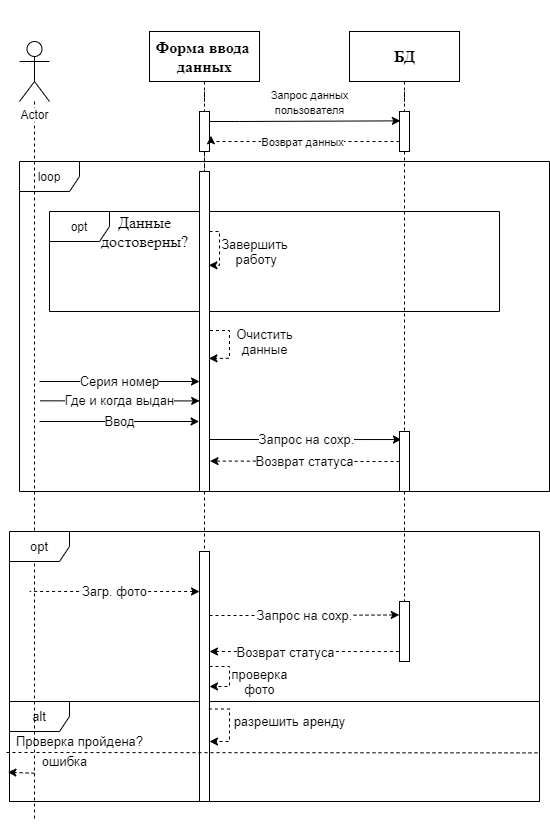


Рисунок 2.2-Диграмма жизнедеятельности

Диаграмма состояний (форма регистрации товара) представлена на рисунке 2.3.

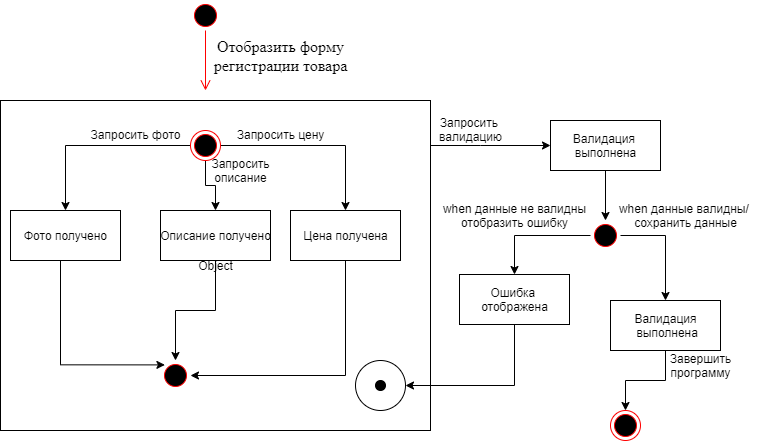


Рисунок 2.3-Диграмма состояний

Диаграмма компонентов представлена на рисунке 2.4.

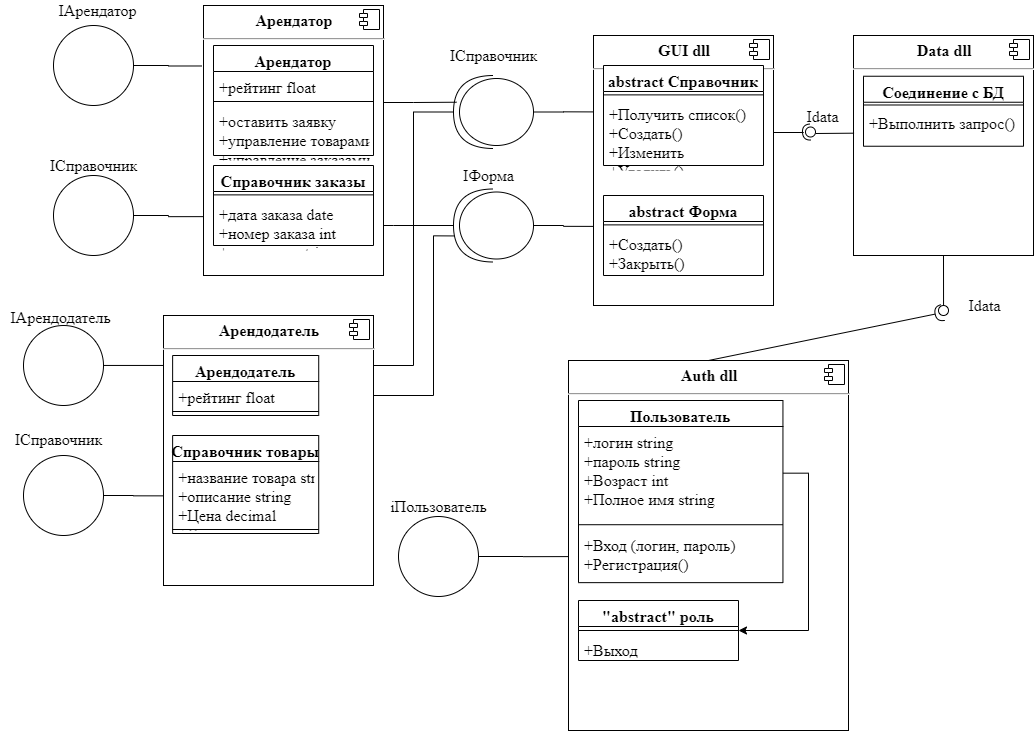


Рисунок 2.4-Диграмма компонентов

## 2.2 Задачи Web-приложения

Данное приложение позволяет решить ряд определенных задач:

* объединение работы арендатора, арендодателя, модератора, администратора в одной системе;
* автоматизация сервиса для арендаторов и арендодателей;
* возможность работы с системой платежей;
* создание индивидуального кабинета с необходимыми возможностями;
* удобная, понятная, и простая в использовании система.

## 2.4 Выбор программных средств

Для разработки данного дипломного проекта необходимы различные инструментальные средства разработки. Правильный выбор таких инструментов способен не только ускорить разработку дипломного проекта, но и сделать решение более гибким и легким.

Сервер:

1. C#
2. Microsoft .NET Framework
3. Microsoft SQL Server
4. Шаблон MVC
5. Razor

Основной язык, используемый в Web-приложении, C# - является объектно-ориентированным. Также C# хорошо сочетается с другими программными продуктами, используемыми в работе.

Microsoft .NET Framework – представляет собой программную платформу. ASP.NET – представляет собой web-платформу для создания серверных web-приложений.

Microsoft SQL Server – представляет собой систему создания и управления реляционными базами данных.

Даная СУБД выбрана так, как она отвечает всем современным стандартам, имеет возможность кроссплатформенности, а также легко работает с другими продуктами Microsoft.

MVC (model - view - controller) предполагает разделение приложения на три компонента:

Контроллер (controller) представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления.

Представление (view) - это собственно визуальная часть или пользовательский интерфейс приложения. Как правило, html-страница, которую пользователь видит, зайдя на сайт.

Модель (model) представляет класс, описывающий логику используемых данных.

Razor— это синтаксис разметки для внедрения серверного кода на веб-страницы. RazorСинтаксис состоит из Razor разметки, C# и HTML. Файлы, содержащие Razor обычно, имеют расширение . cshtml

Клиент:

HTML (язык разметки документов), CSS (язык описания внешнего вида документов), JavaScript (язык программирования с помощью которого веб-страницам придается интерактивность). JQuery (библиотека JavaScript, которая основана на взаимодействии языка HTML и JavaScript)

## 2.5 Проектирование и применение базы данных

База данных

Схема базы данных

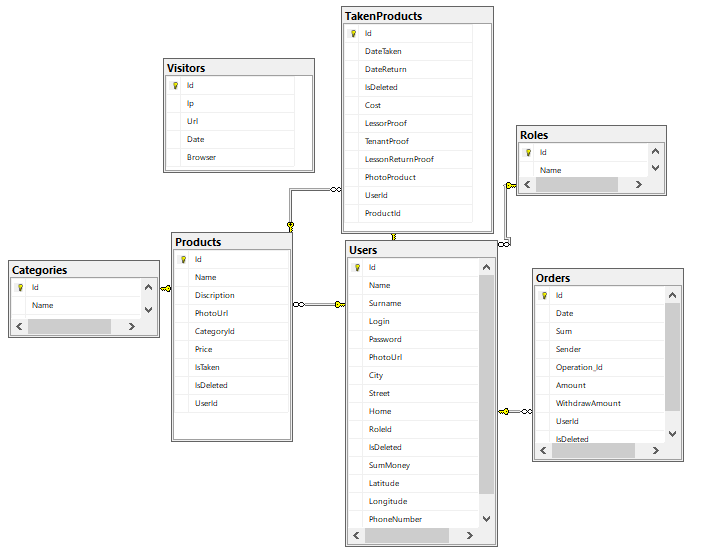


Рисунок 2.5-Схема базы данных

Все модели сервиса представлены в приложении А.

На данном рисунке видно, что база нормализована и состоит из 6 взаимосвязанных таблиц.

## 2.6 Описание структуры таблиц базы данных предметной области

Рассмотрим структуру и тип данных каждой таблицы, созданных исходя из построенной модели.

Таблица «Users» содержит информацию о пользователях. Состав таблицы приведен в таблице 2.1.

Таблица 2.1 - Users

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название поля | Тип поля | Назначение |
| 1. | Id | int | Идентификатор таблицы |
| 2. | Name | nvarchar(50) | Имя пользователя |
| 3. | Surnname | nvarchar(100) | Фамилия пользователя |
| 4. | Login | nvarchar(500) | Логин пользователя |
| 5. | Password | nvarchar(200) | Пароль пользователя |
| 6. | PhotoUrl | nvarchar(MAX) | Фото пользователя |
| 7. | City | nvarchar(200) | Город |
| 8. | Street | nvarchar(200) | Улица |
| 9. | Home | nvarchar(50) | Дом |
| 10. | RoleId | int | Роль в системе |
| 11. | IsDeleted | Bit | Пометка удаления |
| 12. | SumMoney | Money | Баланс пользователя |
| 13. | Latitude | nvarchar(50) | Широта |
| 14. | Longitude | nvarchar(50) | Долгота |
| 15. | PhoneNumber | nvarchar(50) | Номер телефона |

Таблица «Roles» содержит информацию о ролях. Состав таблицы приведен в таблице 2.2.

Таблица 2.2 - Roles

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название поля | Тип поля | Назначение |
| 1. | Id | int | Идентификатор таблицы |
| 2. | Name | nvarchar(100) | Название роли |

Таблица «Products» содержит информацию о товарах. Состав таблицы приведен в таблице 2.3.

Таблица 2.3 - Products

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название поля | Тип поля | Назначение |
| 1. | Id | int | Идентификатор таблицы |
| 2. | Name | nvarchar(50) | Название товара |
| 3. | Discription | nvarchar(MAX) | Описание |
| 4. | PhotoUrl | nvarchar(MAX) | Фото |
| 5. | CategoryId | int | Id категории |
| 6. | Price | money | Стоимость за суxтки |
| 7. | IsTaken | bit | Пометка взятия |
| 8. | IsDeleted | bit | Пометка удаления |
| 9. | UserId | int | Id пользователя |

Таблица «Categories» содержит информацию о категориях. Состав таблицы приведен в таблице 2.4.

Таблица 2.4 - Categories

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название поля | Тип поля | Назначение |
| 1. | Id | int | Идентификатор таблицы |
| 2. | Name | nvarchar(50) | Название категории |
| 3. | IsDeleted | bit | Пометка удаления |

Таблица «Orders» содержит информацию о пополнении счета, снятии со счета. Состав таблицы приведен в таблице 2.5.

Таблица 2.5 - Orders

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название поля | Тип поля | Назначение |
| 1. | Id | int | Идентификатор таблицы |
| 2. | Date | datetime | Дата |
| 3. | Sum | money | Сумма оптарвки |
| 4. | Sender | nvarchar(MAX) | Отправитель |
| 5. | Operation\_Id | nvarchar(MAX) | Id операции |
| 6. | Amount | money | Сколько оправил |
| 7. | WithdrawAmount | money | Сколько отправил без учета комисии |
| 8. | UserId | int | Id пользователя |
| 9. | IsDeleted | bit | Пометка удаления |

Таблица «Visitors» содержит информацию о пользователях посещавших сайт. Состав таблицы приведен в таблице 2.6.

Таблица 2.6 - Visitors

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название поля | Тип поля | Назначение |
| 1. | Id | int | Идентификатор таблицы |
| 2. | Ip | nvarchar(50) | Ip |
| 3. | Url | nvarchar(500) | Url |
| 4. | Date | datetime | Дата |
| 5. | Browser | nvarchar(500) | Браузер |

Таблица «TakenProducts» содержит информацию о взятых в аренду товаров посещавших сайт. Состав таблицы приведен в таблице 2.7.

Таблица 2.7 - TakenProducts

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название поля | Тип поля | Назначение |
| 1. | Id | int | Идентификатор таблицы |
| 2. | DateTaken | date | Дата взятия товара |
| 3. | DateReturn | date | Дата возвращения |
| 4. | IsDeleted | bit | Пометка удаления |
| 5. | Cost | money | Суммарная стоимость |
| 6. | LessorProof | bit | Подтверждение арендодателя |
| 7. | TetantProof | bit | Подтверждение арендатора |
| 8. | LessorReturnProof | bit | Подтверждение арендодателя  о возврате |
| 9. | PhotoProduct | nvarchar(MAX) | Фото товара, по итогу |
| 10. | UserId | int | Id пользователя |
| 11. | ProductId | int | Id товара |

# Вывод по разделу два

* разработана структура информационной системы
* поставлены задачи приложения;
* выбраны программные средства;
* разработана база данных
* выполнена структура приложения

# 3 ОПИСАНИЕ ПРОГРАММНОГО ПРОДУКТА

## 3.1 Общий вид приложения

С приложением могут взаимодействовать только зарегистрированные пользователи.

Незарегистрированные пользователи могут зарегистрироваться на форме рисунок 3.1

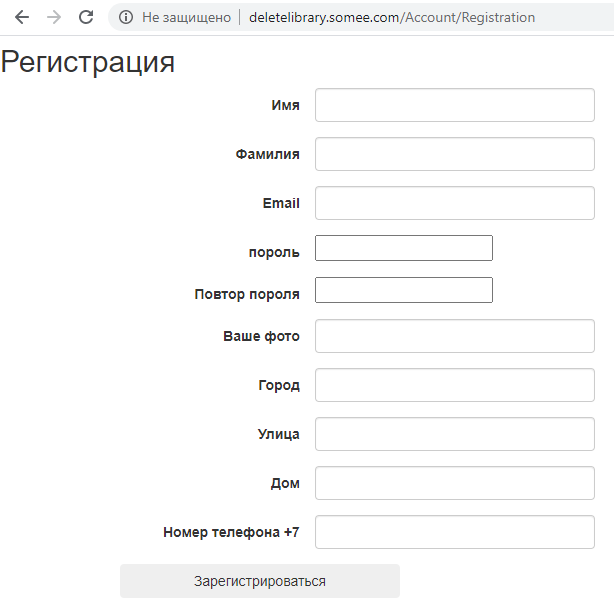


Рисунок 3.1-Форма регистрации

Зарегистрированные же пользователи, могут пройти аутентификацию на форме login рисунок 3.2

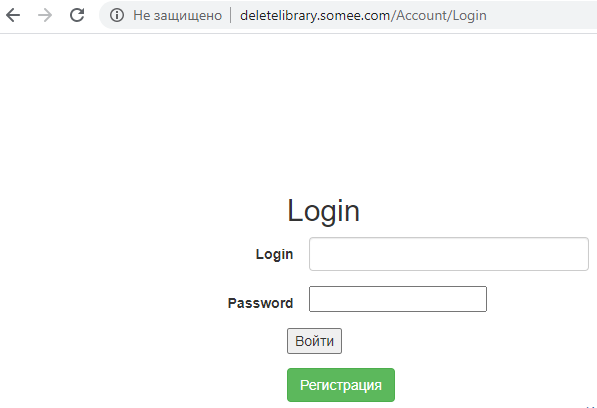


Рисунок 3.2-Форма аутентификации

Код отображения страница регистрации представлен в листинге 1, а код отображения формы аутентификации в листинге 2.

Листинг 1-Код страницы регистрации

@model Rent.Models.Tag.RegisterTag

@{

ViewBag.Title = "Регистрация";

Layout = null;

}

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

<h2>Регистрация</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal" style="margin-left: 120px">

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })

</div>

<div class="form-group">

@Html.LabelFor(model => model.Surname, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.EditorFor(model => model.Surname, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Surname, "", new { @class = "text-danger" })

</div>

<div class="form-group">

@Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })

</div>

<div class="form-group">

@Html.LabelFor(model => model.Password, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.PasswordFor(model => model.Password, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Password, "", new { @class = "text-danger" })

</div>

<div class="form-group">

@Html.LabelFor(model => model.ConfirmPassword, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.PasswordFor(model => model.ConfirmPassword, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.ConfirmPassword, "", new { @class = "text-danger" })

</div>

<div class="form-group">

@Html.LabelFor(model => model.PhotoUrl, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.EditorFor(model => model.PhotoUrl, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.PhotoUrl, "", new { @class = "text-danger" })

</div>

<div class="form-group">

@Html.LabelFor(model => model.City, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.EditorFor(model => model.City, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.City, "", new { @class = "text-danger" })

</div>

<div class="form-group">

@Html.LabelFor(model => model.Street, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.EditorFor(model => model.Street, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Street, "", new { @class = "text-danger" })

</div>

<div class="form-group">

@Html.LabelFor(model => model.Home, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.EditorFor(model => model.Home, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Home, "", new { @class = "text-danger" })

</div>

<div class="form-group">

@Html.LabelFor(model => model.PhoneNumber, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.EditorFor(model => model.PhoneNumber, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.PhoneNumber, "", new { @class = "text-danger" })

</div>

<input type="submit" value="Зарегистрироваться" class="btn btn-block" />

</div>

}

<div>

@Html.ActionLink("Вернуться на форму входа", "Login")

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/bundles/bootstrap")

Листинг 2- код формы аутентификации

@model Rent.Models.Tag.LoginTag

@{

ViewBag.Title = "Аутентификация";

Layout = null;

}

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

<div class="form-horizontal" style="margin-left: 120px">

<h2 style="margin-left: 176px;">Login</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-group">

@Html.LabelFor(model => model.Login, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.EditorFor(model => model.Login, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Login, "", new { @class = "text-danger" })

</div>

<div class="form-group">

@Html.LabelFor(model => model.Password, htmlAttributes: new { @class = "control-label col-md-2" })

@Html.PasswordFor(model => model.Password, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Password, "", new { @class = "text-danger" })

</div>

<input style="margin-left: 176px;" type="submit" value="Войти" />

}

<a href="Registration" style="margin-left: 176px;" class="btn btn-success">Регистрация</a>

</div>

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/bundles/bootstrap")

Контролер для действия входа и регистрации представлен в листинге 3.

Реализация интерфейса IRentService представлена в приложении Б.

Листинг 3 – Account контроллер

public class AccountController : Controller

{

public IRentService \_rentService;

public AccountController(IRentService rentService)

{

\_rentService = rentService;

}

// GET: Account

public ActionResult Login()

{

if (User.Identity.IsAuthenticated) return RedirectToAction("Browse\_item", "Rent");

LoginTag loginModel = new LoginTag();

return View();

}

[HttpPost]

public ActionResult Login(LoginTag loginModel)

{

if (ModelState.IsValid)

{

if (\_rentService.VerifyUserByLoginAndPassword(loginModel))

{

FormsAuthentication.SignOut();

FormsAuthentication.SetAuthCookie(loginModel.Login, true);

return RedirectToAction("Index", "Home");

}

}

return View();

}

public ActionResult Logout()

{

FormsAuthentication.SignOut();

return RedirectToAction("Login");

}

public ActionResult Registration()

{

if (User.Identity.IsAuthenticated) return RedirectToAction("Browse\_item", "Rent");

var registerTag = new RegisterTag();

return View(registerTag);

}

[HttpPost]

public ActionResult Registration(RegisterTag registerTag)

{

if (ModelState.IsValid)

{

var geocode = registerTag.City + ", " + registerTag.Street + ", " + registerTag.Home;

\_rentService.AddLongAndLatiByAddress(geocode, registerTag);

if (!\_rentService.RegistrationCreateUser(registerTag))

{

ModelState.AddModelError("Email", "Пользователь с таким email уже существует");

return View(registerTag);

}

FormsAuthentication.SignOut();

FormsAuthentication.SetAuthCookie(registerTag.Email, true);

return RedirectToAction("Browse\_item", "Rent");

}

else

{

return View(registerTag);

}

}

}

После аутентификации пользователь попадает на главную страницу, где изображена карта с объявлениями Рисунок 3.3.

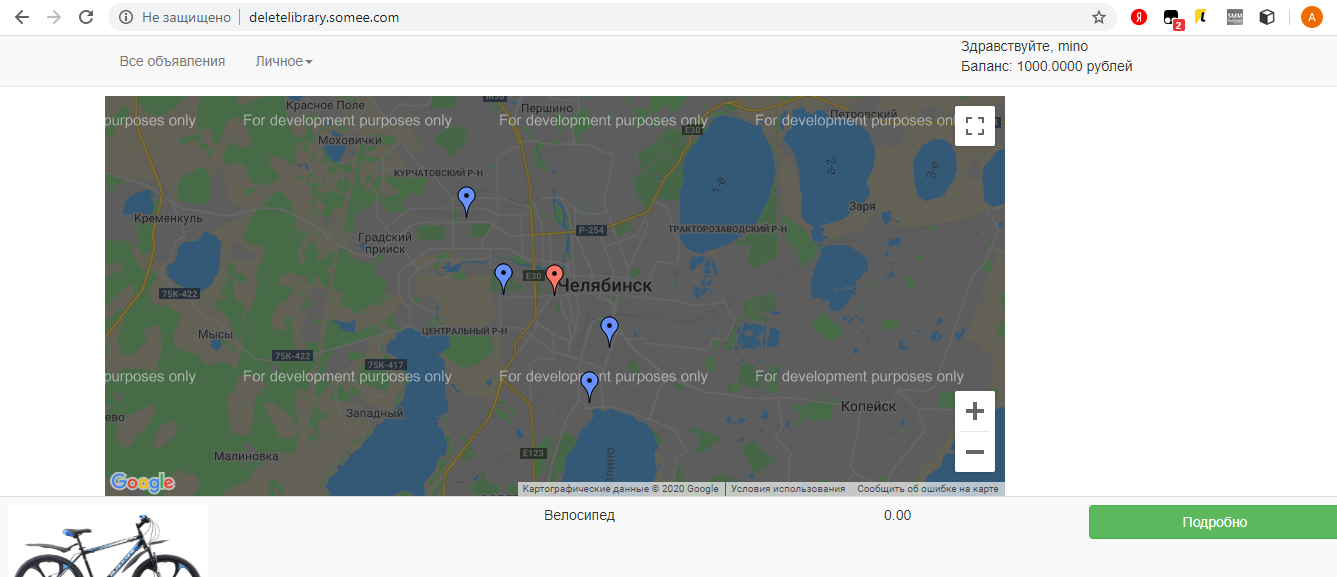


Рисунок 3.3-Страница объявлений

Код работы по просмотру страницы объявлений показан в листинге 4

Листинг 4-просмотр списка объявлений

@model Rent.Models.Product[]

@{

ViewBag.Title = "BrowseItems";

}

<style>

.stationInfo {

height: 150px;

width: 250px;

}

</style>

<div id="map" style="width: 900px; height: 400px;"></div>

@\*<div id="canvas" style="width: 100px;height: 100px ">\*@

<!--Устанавливаем секцию scripts, которая затем будет рендериться на мастер-странице -->

@section scripts {

<script type="text/javascript">

$(document).ready(function () {

GetMap();

});

// Функция загрузки

function GetMap() {

google.maps.visualRefresh = true;

// установка основных координат

var Chel = new google.maps.LatLng(55.159897, 61.402554);

// Установка общих параметров отображения карты, как масштаб, центральная точка и тип карты

var mapOptions = {

zoom: 11,

center: Chel,

mapTypeId: google.maps.MapTypeId.G\_NORMAL\_MAP

};

// Встраиваем гугл-карты в элемент на странице и получаем объект карты

var map = new google.maps.Map(document.getElementById("map"), mapOptions);

// Настраиваем красный маркер, который будет использоваться для центральной точки

var myLatlng = new google.maps.LatLng(55.159897, 61.402554);

var marker = new google.maps.Marker({

position: myLatlng,

map: map,

title: 'Челябинск'

});

// Берем для маркера иконку с сайта google

marker.setIcon('http://maps.google.com/mapfiles/ms/icons/red-dot.png')

// Получаем данные

$.getJSON('@Url.Action("GetData","Home")', function (data) {

// Проходим по всем данным и устанавливаем для них маркеры

$.each(data,

function(i, item) {

var marker = new google.maps.Marker({

'position': new google.maps.LatLng(item.User.Longitude, item.User.Latitude /\*,\*/),

'map': map,

'title': item.Name

});

// Берем для этих маркеров синие иконки с сайта google

marker.setIcon('http://maps.google.com/mapfiles/ms/icons/blue-dot.png')

// Для каждого объекта добавляем доп. информацию, выводимую в отдельном окне

var infowindow = new google.maps.InfoWindow({

content: "<img class='img-map' src=" +

item.PhotoUrl +

"/><div class='stationInfo'><a href=/rent/detailproduct/"+ item.Id +">"+item.Name +"</a><div><h4>Описание: " + item.Discription+"</h4></div><div><h4>Цена: " + item.Price +

" рублей в сутки</h4></div></div>"

});

// обработчик нажатия на маркер объекта

google.maps.event.addListener(marker,

'click',

function() {

infowindow.open(map, marker);

});

});

});

}

</script>

}

</div>

<table class="table table-striped table-hover">

@foreach (var item in Model)

{

<tr>

<td>

<img class="browse-item-img" src="@item.PhotoUrl" />

</td>

<td>

@Html.DisplayFor(modelItem => item.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Price)

</td>

<td>

<td>@Html.ActionLink("Подробно", "DetailProduct", "Rent", new { Id = item.Id }, new { @class = "btn btn-block btn-success" })</td>

</td>

</tr>

}

</table>

Далее пользователь может нажать на интересующее его объявление, и перейти в страницу запроса товара рисунок 3.3 и 3.4.

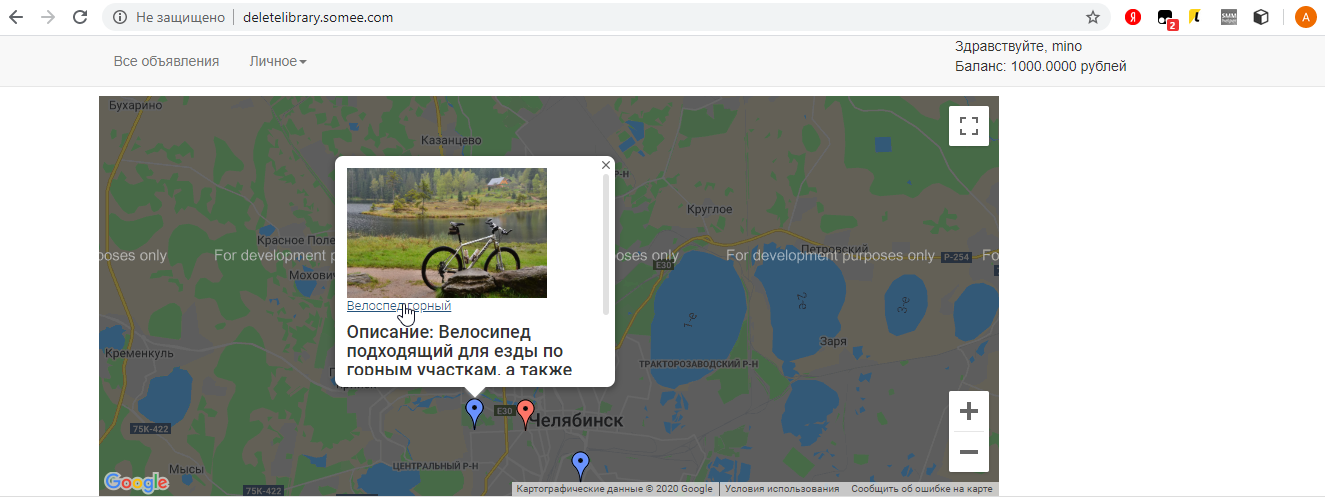


Рисунок 3.3-Выбор объявления

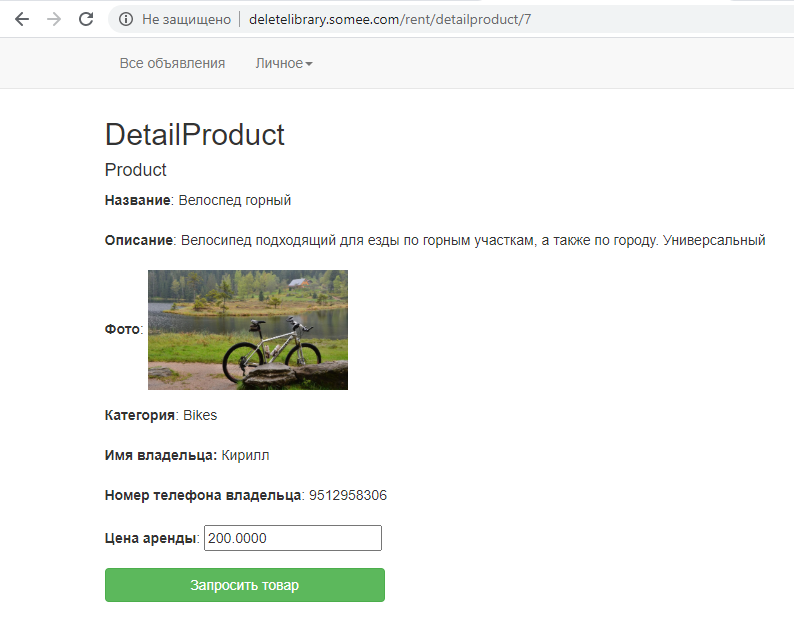


Рисунок 3.4-страница подробного описания

После запрос товара, нас редеректит на страницу всех запрошенных товаров рисунок 3.5

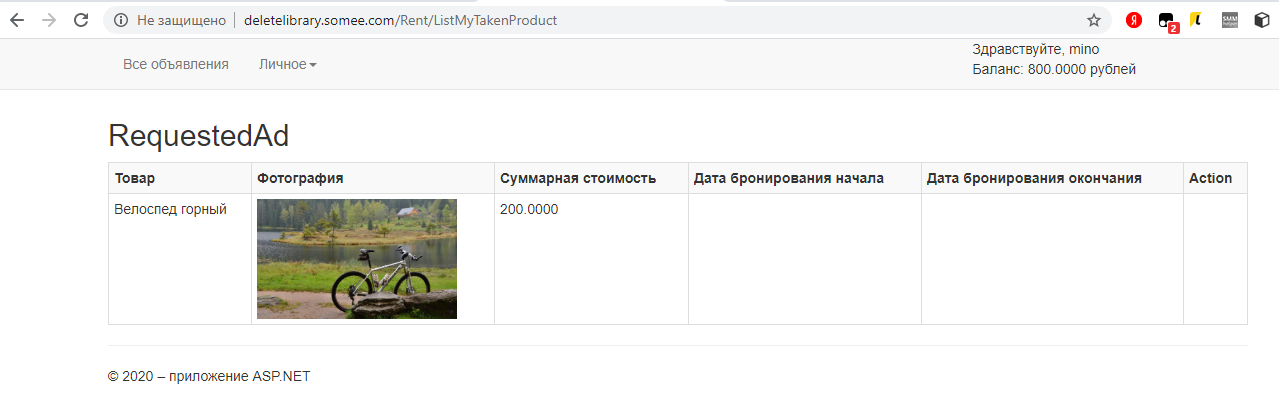


Рисунок 3.5-список запрошенных товаров

После того нужно арендодатель и арендатор связываются, арендодатель передает товар и отмечает у себя в списке активных объявлений, что он отдал товар рисунок 3.6.

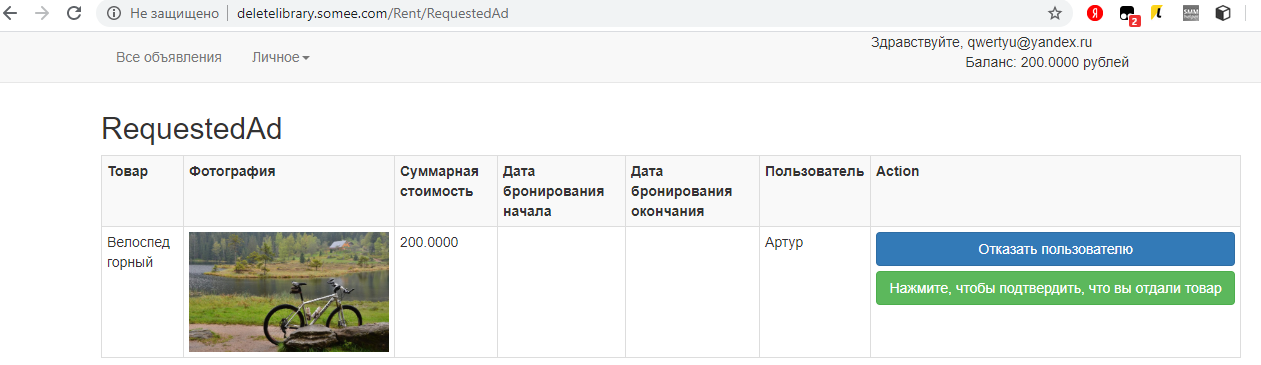


Рисунок 3.6-Страница активных объявлений

После арендатор подтверждает получение товара рисунок 3.7.

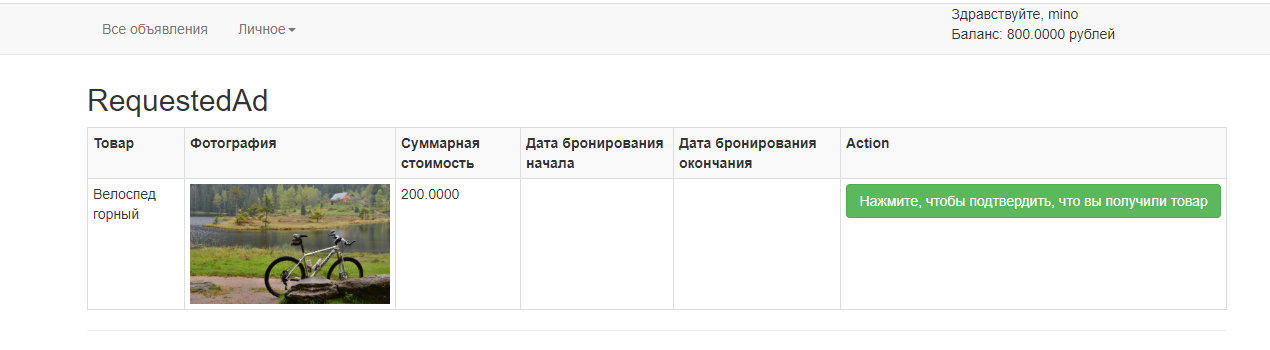


Рисунок 3.7-Страница взятых товаров

Затем происходит возврат товара, арендодатель это подтверждает на странице, сделка завершается, замороженная сумма денег отправляется арендодателю рисунок 3.8.

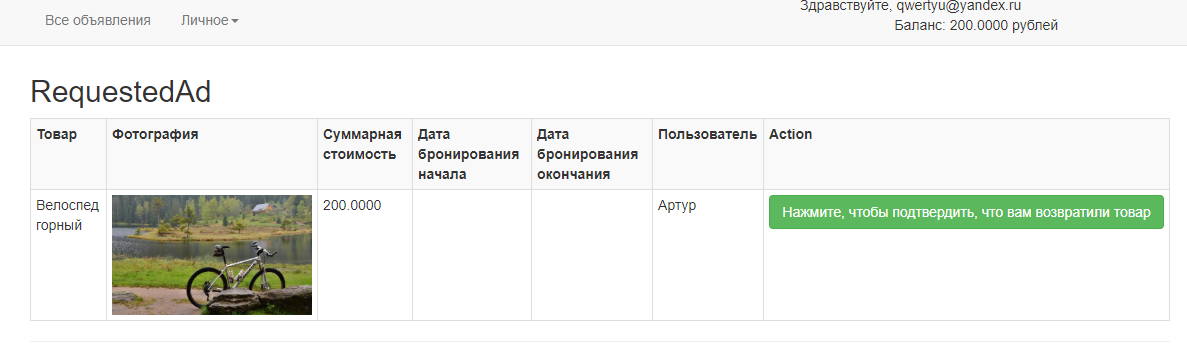


Рисунок 3.8-активных объявлений

Также пользователи могут пополнять свой счет, доступно два способа: яндекс.деньги, или банковской картой рисунок 3.9.

Код работы по просмотру списка активных объявлений представлен в листинге 2, а код работы по запросу на аренды объявления в листинге 3.

Листинг 5-Список активных объявлений

@model Rent.Models.TakenProduct[]

@{

ViewBag.Title = "RequestedAd";

}

<h2>RequestedAd</h2>

<table class="table table-bordered table-hover table-striped table-condensed">

<th>Товар</th>

<th>Фотография</th>

<th>Суммарная стоимость</th>

<th>Дата бронирования начала</th>

<th>Дата бронирования окончания</th>

<th>Пользователь</th>

<th>Action</th>

@foreach (var item in Model)

{

<tr>

<td>@item.Product.Name</td>

<td><img src="@item.Product.PhotoUrl" style="width: 200px; height: 120px;" /></td>

<td>@item.Cost</td>

<td>@item.DateTaken</td>

<td>@item.DateReturn</td>

<td>@item.User.Name</td>

<th>

@{

if (!item.LessorProof)

{

@Html.ActionLink("Отказать пользователю", "Refusal", "Rent", new { idTakenProduct = item.Id }, new { @class = "btn btn-block btn-primary" })

@Html.ActionLink("Нажмите, чтобы подтвердить, что вы отдали товар", "ChekedLessorProof", "Rent", new { idTakenProduct = item.Id }, new { @class = "btn btn-block btn-success" })

}

else if (!item.LessonReturnProof)

{

@Html.ActionLink("Нажмите, чтобы подтвердить, что вам возвратили товар", "ChekedLessorReturnProof", "Rent", new { idTakenProduct = item.Id }, new { @class = "btn btn-block btn-success" })

//Диспут @Html.ActionLink("Нажмите, чтобы подтвердить, что вам возвратили товар", "ChekedLessorReturnProof", "Rent", new { Id = item.Id }, new { @class = "btn btn-block btn-success" })

}

}

</th>

</tr>

}

</table>

Листинг 6-Форма запроса товара

@model Rent.Models.Product

@{

ViewBag.Title = "DetailProduct";

}

<h2>DetailProduct</h2>

<div>

<h4>Product</h4>

<div class="form-group">

<label>Название</label>:

@Html.DisplayFor(model => model.Name)

</div>

<div class="form-group">

<label>Описание</label>:

@Html.DisplayFor(model => model.Discription)

</div>

<div class="form-group">

<label>Фото</label>:

<img src=@Model.PhotoUrl style="width: 200px; height: 120px;" />

</div>

<div class="form-group">

<label>Категория</label>:

@Html.DisplayFor(model => model.Category.Name)

</div>

<div class="form-group">

<label>Имя владельца: </label>

@Html.DisplayFor(model => model.User.Name)

</div>

<div class="form-group">

<label>Номер телефона владельца</label>:

@Html.DisplayFor(model => model.User.PhoneNumber)

</div>

@using (Html.BeginForm("RequestToRent", "Rent", FormMethod.Post))

{

<div class="form-group">

<label>Цена аренды</label>:

<input type="text" name="Price" value="@Model.Price" readonly />

</div>

@Html.HiddenFor(m => m.Id)

<input type="submit" value="Запросить товар" class="btn btn-block btn-success" />

}

</div>

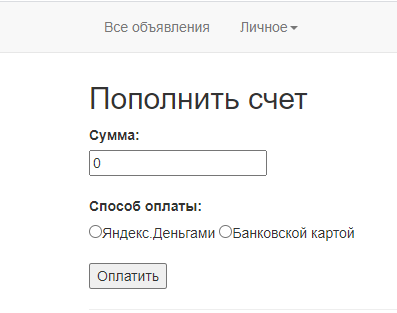


Рисунок 3.9-страница пополнения счета

Код работы по просмотру страницы оплаты показан в листинге 4, а код работы по функциональный части представлен в листинге 5. Реализация сервиса IRentService представлена в приложении А.

Листинг 7 – Страница оплаты

<h2>Пополнить счет</h2>

<div>

<form method="POST" action="https://money.yandex.ru/quickpay/confirm.xml">

<input name="label" value="@Model.OrderId" type="hidden">

<input name="receiver" value="410011674560436" type="hidden">

<input name="quickpay-form" value="shop" type="hidden">

<input type="hidden" name="targets" value="Оплата заказа @Model.OrderId">

<label for="sum">Сумма: </label><br />

<input name="sum" value="@Model.Sum" maxlength="10" data-type="number" type="text"><br /><br />

<label for="sum">Способ оплаты: </label><br />

<input type="radio" name="paymentType" value="PC">Яндекс.Деньгами

<input type="radio" name="paymentType" value="AC">Банковской картой<br /><br />

<input type="submit" name="submit-button" value="Оплатить">

<input name="successURL" value="http://deleteLibrary.somee.com/Pay/Paid" type="hidden">

<input name="quickpay-back-url" value="http://deleteLibrary.somee.com/" type="hidden">

</form>

</div>

Листинг 8-код работы оплаты

[Authorize]

public class PayController : Controller

{

public IRentService \_rentService;

public PayController(IRentService rentService)

{

\_rentService = rentService;

}

// GET: Pay

public ActionResult Index()

{

var userId = \_rentService.GetUserByLogin(User.Identity.Name).Id;

Order order = new Order

{

UserId = userId

};

\_rentService.CreateOrder(order);

var newOrder = \_rentService.LastOrderByUserId(userId);

OrderModel orderModel = new OrderModel { OrderId = newOrder.Id.ToString() };

return View(orderModel);

}

[HttpGet]

public string Paid()

{

return "<p>заказ оплачен</p>";

}

[HttpPost]

public void Paid(string notification\_type, string operation\_id, string label, string datetime,

decimal amount, decimal withdraw\_amount, string sender, string sha1\_hash, string currency, bool codepro)

{

string key = "\*\*\*"; // секретный код

// проверяем хэш

string paramString = String.Format("{0}&{1}&{2}&{3}&{4}&{5}&{6}&{7}&{8}",

notification\_type, operation\_id, amount, currency, datetime, sender,

codepro.ToString().ToLower(), key, label);

string paramStringHash1 = GetHash(paramString);

// создаем класс для сравнения строк

StringComparer comparer = StringComparer.OrdinalIgnoreCase;

// если хэши идентичны, добавляем данные о заказе в бд

if (0 == comparer.Compare(paramStringHash1, sha1\_hash))

{

var order=\_rentService.GetOrderById(Convert.ToInt32(label));

order.Operation\_Id = operation\_id;

order.Date = DateTime.Now;

order.Amount = amount;

order.WithdrawAmount = withdraw\_amount;

order.Sender = sender;

order.IsDeleted = true;

\_rentService.UpdateOrder(order);

\_rentService.PayMoneyForUserByLogin(amount, order.User.Login);

}

}

public string GetHash(string val)

{

SHA1 sha = new SHA1CryptoServiceProvider();

byte[] data = sha.ComputeHash(Encoding.Default.GetBytes(val));

StringBuilder sBuilder = new StringBuilder();

for (int i = 0; i < data.Length; i++)

{

sBuilder.Append(data[i].ToString("x2"));

}

return sBuilder.ToString();

}

}

Также пользователи могут создавать новые объявления рисунок 3.10

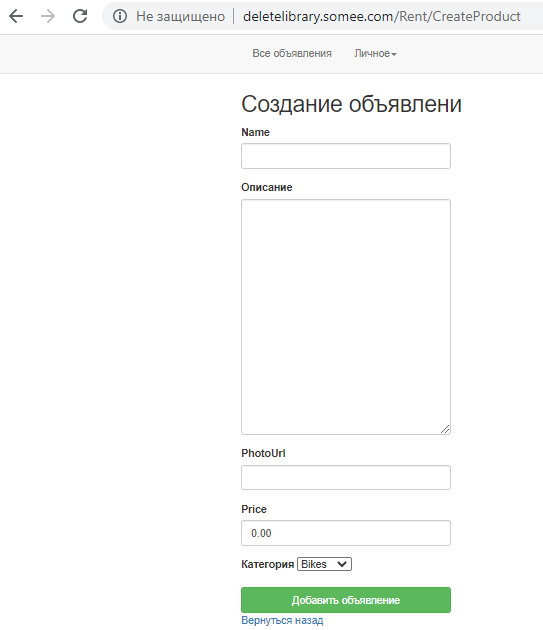


Рисунок 3.10-форма создания объявления

Код страницы создания объявлений представлен в листинге 9.

Листинг 9-страница создания объявления.

@model Rent.Models.Product

@{

ViewBag.Title = "Создание объявления";

}

<h2>Создание объявлени</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div>

<div class="form-group">

@Html.LabelFor(m => m.Name)

@Html.EditorFor(m => m.Name, new { HtmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(m => m.Name)

</div>

<div class="form-group">

@Html.LabelFor(m => m.Discription)

@Html.TextAreaFor(m => m.Discription, new { @class = "form-control", @rows = "15" })

@Html.ValidationMessageFor(m => m.Discription)

</div>

<div class="form-group">

@Html.LabelFor(m => m.PhotoUrl)

@Html.EditorFor(m => m.PhotoUrl, new { HtmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(m => m.PhotoUrl)

</div>

<div class="form-group">

@Html.LabelFor(m => m.Price)

@Html.EditorFor(m => m.Price, new { HtmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(m => m.Price)

</div>

<div class="form-group">

@Html.LabelFor(m => m.CategoryId)

@Html.DropDownList("CategoryId", ViewBag.Category as SelectList)

@Html.ValidationMessageFor(m => m.CategoryId)

</div>

<div>

<input class="btn btn-block btn-success" type="submit" value="Добавить объявление" />

</div>

</div>

}

<div>

@Html.ActionLink("Вернуться назад", "MyAd")

</div>

Код работы всех пользовательских страниц показан в листинге 10.

Листинг 10-Код логики пользовательских страниц

[Authorize]

public class RentController : Controller

{

public IRentService \_rentService;

public RentController(IRentService rentService)

{

\_rentService = rentService;

}

public ActionResult \_Map()

{

return PartialView();

}

// GET: Rent

public ActionResult Browse\_item()

{

var products = \_rentService.GetAllProducts();

return View(products);

}

public ActionResult DetailProduct(int id)

{

var product = \_rentService.GetProductById(id);

return View(product);

}

public ActionResult CreateProduct()

{

SelectList category = new SelectList(\_rentService.GetAllGategories(), "Id", "Name");

ViewBag.Category = category;

var product = new Product();

return View(product);

}

[HttpPost]

public ActionResult CreateProduct(Product product)

{

product.UserId = \_rentService.GetUserByLogin(User.Identity.Name).Id;

\_rentService.CreateProduct(product);

return RedirectToAction("MyAd");

}

[HttpPost]

public ActionResult RequestToRent(int id,decimal Price=0)

{

var userMoney = \_rentService.GetUserByLogin(User.Identity.Name).SumMoney;

if(userMoney<Price) return HttpNotFound();

if (!\_rentService.CheckedIsTakenProductByProductId(id)) return HttpNotFound();

//Если нельзя снять то выдаем ошибку, а не 404

if (!\_rentService.WriteOffMoneyFromUserByLogin(Price, User.Identity.Name)) return HttpNotFound();

var takenProduct = new TakenProduct

{

ProductId = id,

UserId = \_rentService.GetUserByLogin(User.Identity.Name).Id,

Cost = Price,

};

\_rentService.CreateTakenProduct(takenProduct);

return RedirectToAction("RequestedAd");

}

//мои объявления

public ActionResult MyAd()

{

var products=\_rentService.GetProductByUserId(\_rentService.GetUserByLogin(User.Identity.Name).Id);

return View(products);

}

//запрошенные на аренду

public ActionResult RequestedAd()

{

var takenProducts =

\_rentService.GetAllTakenProductsByUserId(\_rentService.GetUserByLogin(User.Identity.Name).Id);

return View(takenProducts);

}

public ActionResult ChekedLessorProof(int idTakenProduct)

{

var userId = \_rentService.GetUserByLogin(User.Identity.Name).Id;

\_rentService.ChekedLessorProof(idTakenProduct, userId);

return RedirectToAction("RequestedAd");

}

public ActionResult ChekedLessorReturnProof(int idTakenProduct)

{

var userId = \_rentService.GetUserByLogin(User.Identity.Name).Id;

//помечаем что арендодатель подтвердил возврат товара

if(!\_rentService.ChekedLessorReturnProof(idTakenProduct, userId)) return RedirectToAction("Browse\_item");

var takenProduct = \_rentService.GetTakenProductById(idTakenProduct);

var product = takenProduct.Product;

\_rentService.PayMoneyForUserByLogin(takenProduct.Cost,User.Identity.Name);

//убираем пометку взятия

\_rentService.CheckedNoIsTakenProductByProductId(takenProduct.ProductId);

//помечаем на удаление запись взятия товаря

\_rentService.DeleteTakenProductById(idTakenProduct,userId);

return RedirectToAction("RequestedAd");

}

public ActionResult ListMyTakenProduct()

{

var userId = \_rentService.GetUserByLogin(User.Identity.Name).Id;

var takenProducts = \_rentService.GetAllListMyTakenProduct(userId);

return View(takenProducts);

}

public ActionResult CheckedTenantProof(int idTakenProduct)

{

var userId = \_rentService.GetUserByLogin(User.Identity.Name).Id;

\_rentService.CheckedTenantProof(idTakenProduct, userId);

return RedirectToAction("ListMyTakenProduct");

}

public ActionResult Refusal(int idTakenProduct)

{

var takenProduct = \_rentService.GetTakenProductById(idTakenProduct);

if (User.Identity.Name != takenProduct.Product.User.Login) return HttpNotFound();

\_rentService.CheckedNoIsTakenProductByProductId(takenProduct.ProductId);

var userId = \_rentService.GetUserByLogin(User.Identity.Name).Id;

\_rentService.DeleteTakenProductById(idTakenProduct, userId);

return RedirectToAction("ListMyTakenProduct");

}

public ActionResult DeleteProduct(int id)

{

\_rentService.DeleteProductById(id);

return RedirectToAction("MyAd");

}

public ActionResult UpdateProduct(int id)

{

SelectList category = new SelectList(\_rentService.GetAllGategories(), "Id", "Name");

ViewBag.Category = category;

var product =\_rentService.GetProductById(id);

return View(product);

}

[HttpPost]

public ActionResult UpdateProduct(Product product)

{

\_rentService.UpdateProduct(product);

return RedirectToAction("MyAd");

}

}

Так как в системе помимо пользователей присутствуют еще и администраторы, и модераторы, для них созданы отдельные функции.

Для авторизации используется провайдер ролей, он сопоставляет пользователей с их ролями, код провайдера представлен в листинге 11.

Листинг 11-провайдер ролей

public class CustomRoleProvider : RoleProvider

{

public override bool IsUserInRole(string username, string roleName)

{

using (var db = new RentContext())

{

var user = db.Users.FirstOrDefault(m => m.Login == username);

var role = db.Roles.FirstOrDefault(m => m.Name.ToUpper() == roleName.ToUpper());

if (user != null && role != null)

{

if (user.RoleId == role.Id)

{

return true;

}

}

}

return false;

}

public override string[] GetRolesForUser(string username)

{

string[] roles = new string[] { };

using (var db = new RentContext())

{

User user = db.Users.FirstOrDefault(m => m.Login == username);

if (user != null)

{

Role role = db.Roles.Find(user.RoleId);

if (role != null)

{

roles = new string[] {role.Name};

}

}

}

return roles;

}

}

Для того чтобы функциональность была доступна только администраторам и пользователям на мастер странице создан dropdown, который доступен только этим ролям рисунок 3.11.

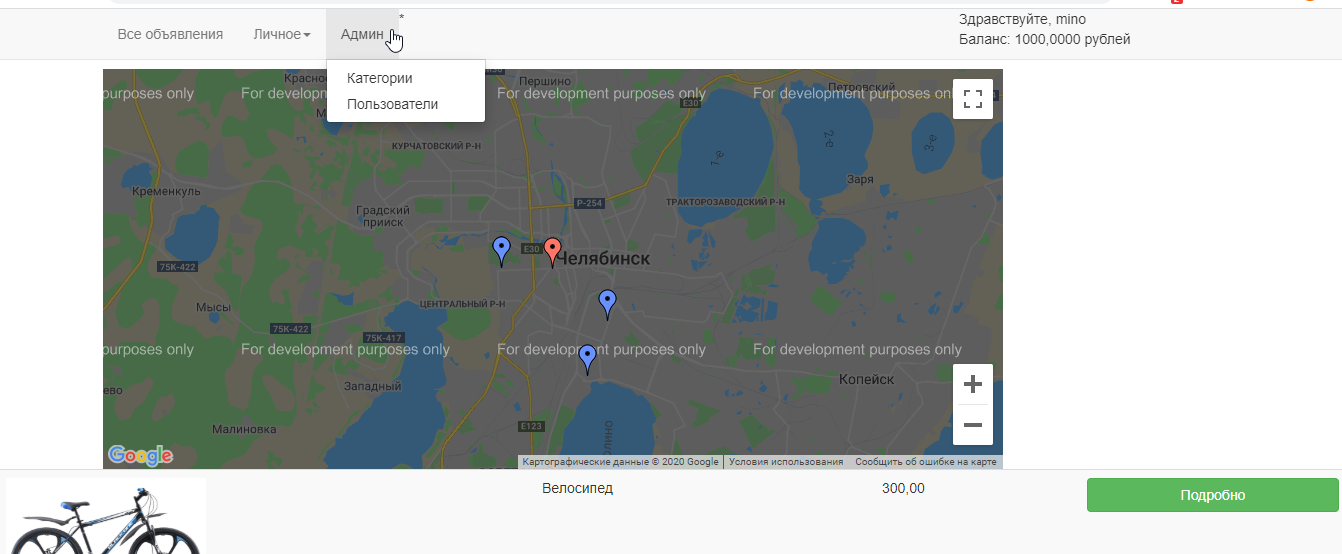


Рисунок 3.11-вкладка Admin

Код мастер-страницы представлен в листинге 12, а код хелпера, позволяющий отследить роль пользователя в листинге 13

Листинг 12-код мастер-страницы

@using Rent.ExtensionsForHelpers

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<script src="http://maps.google.com/maps/api/js?sensor=true" type="text/javascript"></script>

<title>@ViewBag.Title – приложение ASP.NET</title>

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

</head>

<body>

<nav class="navbar navbar-default navbar-fixed-top">

<div class="container">

<ul class="nav navbar-nav">

<li>@Html.ActionLink("Все объявления", "Browse\_item", "Rent")

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false">Личное<span class="caret"></span></a>

<ul class="dropdown-menu">

<li>@Html.ActionLink("Взятые мной в аренду", "ListMyTakenProduct", "Rent")</li>

<li>@Html.ActionLink("Мои объявления", "MyAd", "Rent")</li>

<li>@Html.ActionLink("Активные", "RequestedAd", "Rent")</li>

<li>@Html.ActionLink("Пополнить счет", "Index", "Pay")</li>

<li role="separator" class="divider"></li>

<li>@Html.ActionLink("Выйти", "Logout", "Account")</li>

<li role="separator" class="divider"></li>

</ul>

</li>

\*<li class="dropdown">

@Html.ActionLink("Любой текст", "ListMyTakenProduct", "Rent").Action().If(User.IsInRole("Admin"))

<ul class="dropdown-menu">

<li>@Html.ActionLink("Категории", "ListCategories", "Admin")</li>

<li>@Html.ActionLink("Пользователи", "ListUsers", "Admin")</li>

</ul>

</li>

</ul>

<ul class="nav navbar-nav navbar-right">

<li>Здравствуйте, @User.Identity.Name</li>

<li style="padding-top: 20px; right: 127px;">@Html.Name(User.Identity.Name).Money() рублей</li>

</ul>

</div>

</nav>

<div style="margin-top: -80px;" class="container body-content">

@RenderBody()

<hr />

<footer>

<p>&copy; @DateTime.Now.Year – приложение ASP.NET</p>

</footer>

</div>

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/bundles/bootstrap")

@RenderSection("scripts", required: false)

</body>

</html>

Листинг 13 – код хелпера

public static class AdminHelper

{

public static MvcHtmlString If(this MvcHtmlString value, bool evaluation)

{

return evaluation ? value : MvcHtmlString.Empty;

}

public static MvcHtmlString Money(this MvcHtmlString value)

{

IRentService \_rentService=new EntityRentService();

var username=\_rentService.GetUserByLogin(value.ToString());

var mvcHtmlString = new MvcHtmlString("Баланс: "+username.SumMoney.ToString());

return mvcHtmlString;

}

public static MvcHtmlString Action(this MvcHtmlString value)

{

TagBuilder a = new TagBuilder("a");

a.Attributes.Add("href","#");

a.Attributes.Add("class","dropdown-toggle");

a.Attributes.Add("data-toggle","dropdown");

a.Attributes.Add("role","button");

a.Attributes.Add("aria-haspopup","true");

a.Attributes.Add("aria-expanded","false");

a.SetInnerText("Админ");

var mvcHtmlString = new MvcHtmlString(a.ToString());

return mvcHtmlString;

}

}

Администраторы могут добавлять, изменять и удалять категории рисунки 3.12, 3.13, 3.14

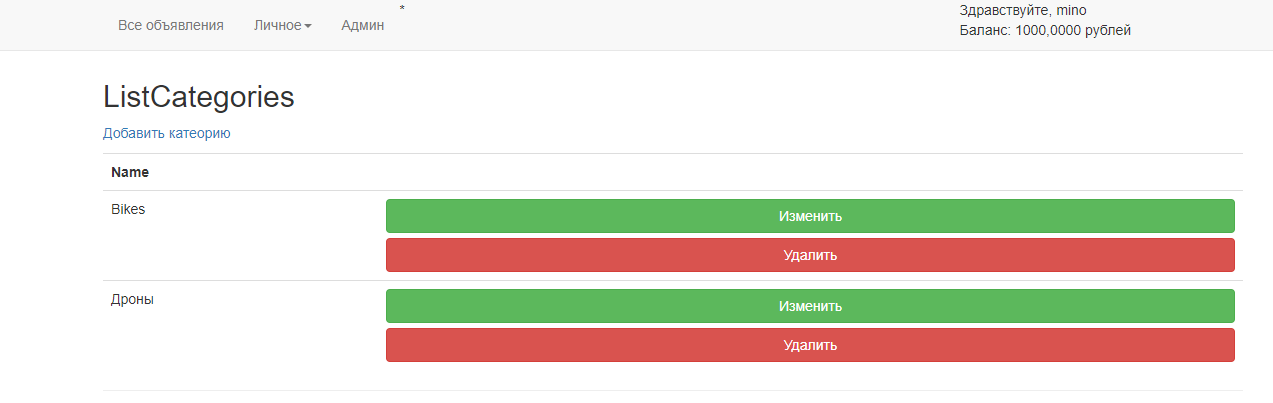


Рисунок 3.12 список категорий

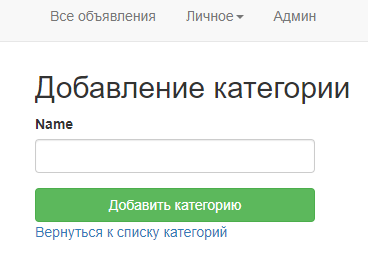


Рисунок 3.13 добавление категорий

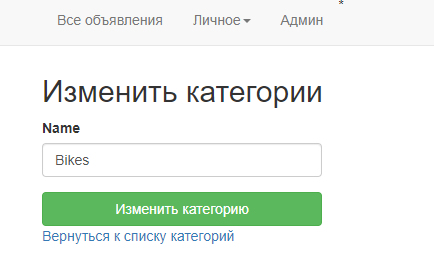


Рисунок 3.13-изменение категории

Код представления на вывод категорий приведен в листинге 14, код представления об добавления категории в листинге 15, код изменения в листинге 16.

Листинг 14-код страницы категорий

@model IEnumerable<Rent.Models.Category>

@{

ViewBag.Title = "ListCategories";

}

<h2>ListCategories</h2>

<p>

@Html.ActionLink("Добавить катеорию", "CreateCategory")

</p>

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.Name)

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Name)

</td>

<td>

@Html.ActionLink("Изменить", "UpdateCategory", new { id=item.Id }, new { @class = "col-md-3 btn btn-block btn-success" })

@Html.ActionLink("Удалить", "DeleteCategory", new { id=item.Id }, new { @class = "col-md-3 btn btn-block btn-danger" })

</td>

</tr>

}

</table>

Листинг 15-код страницы добавления категорий

@model Rent.Models.Category

@{

ViewBag.Title = "Добавление категории";

}

<h2>Добавление категории</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div>

<div class="form-group">

@Html.LabelFor(m => m.Name)

@Html.EditorFor(m => m.Name, new { HtmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(m => m.Name)

</div>

<div>

<input class="btn btn-block btn-success" type="submit" value="Добавить категорию" />

</div>

</div>

}

<div>

@Html.ActionLink("Вернуться к списку категорий", "ListCategories", new { @class = "btn btn-block" })

</div>

Листинг 16-код страницы изменения категорий

@model Rent.Models.Category

@{

ViewBag.Title = "Изменить категорию";

}

<h2>Изменить категории</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

@Html.HiddenFor(m => m.Id)

<div>

<div class="form-group">

@Html.LabelFor(m => m.Name)

@Html.EditorFor(m => m.Name, new { HtmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(m => m.Name)

</div>

<div>

<input class="btn btn-block btn-success" type="submit" value="Изменить категорию" />

</div>

</div>

}

<div>

@Html.ActionLink("Вернуться к списку категорий", "ListCategories", new { @class = "btn btn-block" })

</div>

## 3.2 Публикация сайта

Иногда по тем или иным причинам сложно опубликовать или поддерживать сайт на веб-сервере IIS на своем ПК: например, если в наличии нету такого компьютера, который бы можно было использовать в качестве сервера, недостаточная пропускная способность сети, ненужные хлопоты при возможной настройке и конфигурации IIS и т.д. Поэтому часто веб-приложение размещают на каких-то внешних ресурсах, специально предназначенных для хостинга веб-сайтов.

Я буду использовать хостинг [somee.com](https://somee.com/) с поддержкой asp.net и ms sql server, который в том числе предлагает и бесплатный хостинг, правда, с ограничениями по сравнению с платными услугами возможностями. Панель управления представлена на рисунке 3.14.

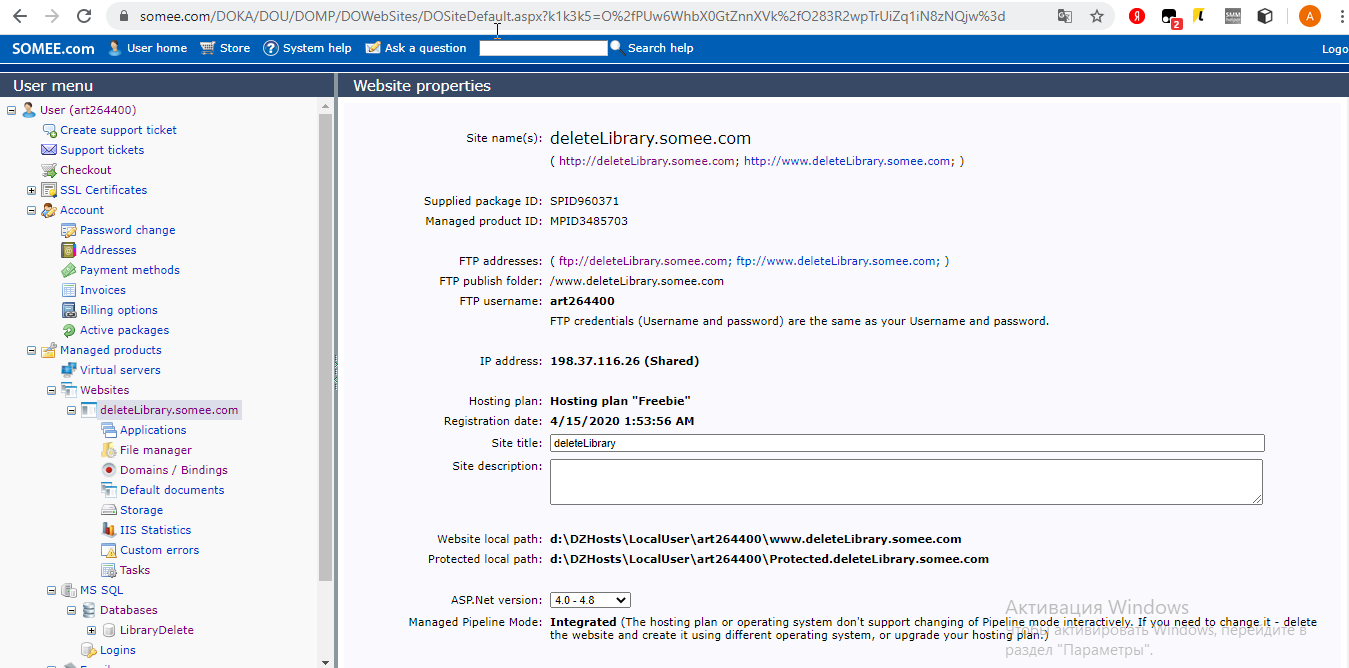


Рисунок 3.14-somme.com управление сайтом

После публикации сайта, он доступен по домену: http://deletelibrary.somee.com/

# Выводы по разделу три

В данном разделе были выполнены следующие задачи:

* представлен общий вид приложения;
* представлены возможности работы пользователя;
* представлены возможности работы администратора и модератора;
* представлен работающий сайт на хостинге

# ЗАКЛЮЧЕНИЕ

Целью данной работы было закрепление теоретических знаний, на практике, с помощью разработки Web - приложения средствами ASP.NET. Подводя итог проведенной работе, можно отметить, что приложение соответствует всем требованиям задания.

На основе данной разработки были усовершенствованы навыки и расширены знания в области баз данных, программирования, а также приобретены знания создания приложений.

Цель работы достигнута. В полной мере решены поставленные задачи:

* Создать удобное и интуитивно понятное Web приложение;
* Реализовать систему отображения объявлений;
* Реализовать работу основого функционала сайта аренды;
* Объединение работы арендатора, арендодателя, администратора, модератора в однойсистеме.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дино Эспозито, Modern Web Development: Understanding Domains, Technologies, and User Experienceк: / Дино Эспозито. – New York: Изд-во Вильямс, 2017. – 464 с.
2. Это онлайн курсы для обучения программированию. - https://www.freecodecamp.org
3. Интерактивная онлайн-платформа по обучению 12 языкам программирования. - https://www.codecademy.com
4. Библиотека официальной технической документации для разработчиков под ОС Microsoft Windows. - https://msdn.microsoft.com/ru-ru/dn308572.aspx
5. Крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. - https://github.com
6. Русскоязычный веб-сайт в формате коллективного блога с элементами новостного сайта. - <https://habr.com>

# ПРИЛОЖЕНИЯ

# ПРИЛОЖЕНИЕ А

Все модели, использующиеся в программе

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

public bool IsDeleted { get; set; }

}

public class Order

{

public int Id { get; set; } // id заказа

public DateTime? Date { get; set; } // дата

public decimal Sum { get; set; } // сумма заказа

public string Sender { get; set; } // отправитель - кошелек в ЯД

public string Operation\_Id { get; set; } // id операции в ЯД

public decimal? Amount { get; set; } // сумма, которую заплатали с учетом комиссии

public decimal? WithdrawAmount { get; set; } // сумма, которую заплатали без учета комиссии

public int? UserId { get; set; } // id пользователя в системе, который сделал заказ

public User User { get; set; }

public bool IsDeleted { get; set; }

}

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

[Display(Name = "Описание")]

public string Discription { get; set; }

public string PhotoUrl { get; set; }

public decimal Price { get; set; }

public bool IsTaken { get; set; }

public bool IsDeleted { get; set; }

[Display(Name = "Категория")]

public int CategoryId { get; set; }

public Category Category { get; set; }

public int UserId { get; set; }

public User User { get; set; }

}

public class Role

{

public int Id { get; set; }

public string Name { get; set; }

}

public class TakenProduct

{

public int Id { get; set; }

public DateTime? DateTaken { get; set; }

public DateTime? DateReturn { get; set; }

public bool IsDeleted { get; set; }

public bool LessorProof { get; set; }

public bool TenantProof { get; set; }

public bool LessonReturnProof { get; set; }

public decimal Cost { get; set; }

public string PhotoProduct { get; set; }

Окончание приложения А

public int UserId { get; set; }

public User User { get; set; }

public int ProductId { get; set; }

public Product Product { get; set; }

}

public class User

{

public int Id { get; set; }

[Display(Name = "Имя")]

public string Name { get; set; }

[Display(Name = "Логин")]

public string Login { get; set; }

[Display(Name = "Фамилия")]

public string Surname { get; set; }

[Display(Name = "Пароль")]

public string Password { get; set; }

[Display(Name = "Фото")]

public string PhotoUrl { get; set; }

[Display(Name = "Город")]

public string City { get; set; }

[Display(Name = "Улица")]

public string Street { get; set; }

[Display(Name = "Дом")]

public string Home { get; set; }

public bool IsDeleted { get; set; }

[Display(Name = "Баланс")]

public decimal SumMoney { get; set; }

public int RoleId { get; set; }

[Display(Name = "Роль")]

public Role Role { get; set; }

[Display(Name = "Широта")]

public string Latitude { get; set; }

[Display(Name = "Долгота")]

public string Longitude { get; set; }

[Display(Name = "Номер телефона")]

public string PhoneNumber { get; set; }

}

# ПРИЛОЖЕНИЕ Б

Реализация сервиса IRentService

public class EntityRentService : IRentService

{

public User[] GetAllUsers()

{

using (var db = new RentContext())

{

return db.Users.Where(m => m.IsDeleted == false).ToArray();

}

}

public User GetUserById(int id)

{

using (var db = new RentContext())

{

return db.Users.FirstOrDefault(m => m.Id == id);

}

}

public User GetUserByLogin(string login)

{

using (var db = new RentContext())

{

return db.Users.FirstOrDefault(m => m.Login == login);

}

}

public bool RemoveUserById(int id)

{

using (var db = new RentContext())

{

var user = GetUserById(id);

if (user == null) return false;

user.IsDeleted = true;

db.Users.AddOrUpdate(user);

db.SaveChanges();

return true;

}

}

public bool updateUser(User updateUser)

{

using (var db = new RentContext())

{

db.Users.AddOrUpdate(updateUser);

db.SaveChanges();

return true;

}

}

public bool VerifyUserByLoginAndPassword(LoginTag loginModel)

{

using (var db = new RentContext())

{

var user = db.Users.FirstOrDefault(m => m.Login == loginModel.Login && m.Password == loginModel.Password);

if (user != null)

{

return true;

Продолжение приложения Б

}

return false;

}

}

public Product[] GetAllProducts()

{

using (var db = new RentContext())

{

return db.Products

.Where(m => m.IsDeleted == false && m.IsTaken == false)

.Include(m => m.Category)

.Include(m => m.User)

.ToArray();

}

}

public Product GetProductById(int id)

{

using (var db = new RentContext())

{

var product = db.Products.Include(m => m.Category).Include(m => m.User).FirstOrDefault(m =>

m.IsDeleted == false && m.IsTaken == false && m.Id == id);

return product;

}

}

//списываем деньги у пользователя по логину

public bool WriteOffMoneyFromUserByLogin(decimal money, string login)

{

using (var db = new RentContext())

{

var user = db.Users.FirstOrDefault(m => m.Login == login);

if (user == null) return false;

if (user.SumMoney < money) return false;

user.SumMoney -= money;

db.Users.AddOrUpdate(user);

db.SaveChanges();

return true;

}

}

public bool CheckedIsTakenProductByProductId(int id)

{

using (var db = new RentContext())

{

var product = db.Products.Include(m => m.User).FirstOrDefault(m => m.Id == id);

if (product == null) return false;

product.IsTaken = true;

db.Products.AddOrUpdate(product);

db.SaveChanges();

return true;

}

}

public bool CreateProduct(Product product)

{

using (var db=new RentContext())

{

db.Products.Add(product);

db.SaveChanges();

Продолжение приложения Б

return true;

}

}

public TakenProduct[] GetAllTakenProductsByUserId(int userId)

{

using (var db = new RentContext())

{

return db.TakenProducts.Include(m => m.User).Include(m => m.Product).Where(m => m.IsDeleted == false && m.Product.UserId == userId).ToArray();

}

}

public TakenProduct[] GetAllListMyTakenProduct(int userId)

{

using (var db = new RentContext())

{

return db.TakenProducts.Include(m => m.User).Include(m => m.Product).Where(m => m.IsDeleted == false && m.UserId == userId).ToArray();

}

}

public Product[] GetProductByUserId(int id)

{

using (var db = new RentContext())

{

return db.Products.Include(m => m.User).Include(m => m.Category).Where(m => m.UserId == id && m.IsDeleted==false).ToArray();

}

}

public bool CreateTakenProduct(TakenProduct takenProduct)

{

using (var db = new RentContext())

{

db.TakenProducts.AddOrUpdate(takenProduct);

db.SaveChanges();

return true;

}

}

public bool ChekedLessorProof(int idProof, int UserId)

{

using (var db = new RentContext())

{

var takenProduct = db.TakenProducts.Include(m => m.Product).FirstOrDefault(m => m.Id == idProof && m.Product.UserId == UserId && m.IsDeleted == false);

if (takenProduct == null) return false;

takenProduct.LessorProof = true;

db.TakenProducts.AddOrUpdate(takenProduct);

db.SaveChanges();

return true;

}

}

public bool ChekedLessorReturnProof(int idProof, int UserId)

{

using (var db = new RentContext())

{

var takenProduct = db.TakenProducts.Include(m => m.Product).FirstOrDefault(m => m.Id == idProof && m.Product.UserId == UserId && m.IsDeleted == false);

if (takenProduct == null) return false;

takenProduct.LessonReturnProof = true;

db.TakenProducts.AddOrUpdate(takenProduct);

db.SaveChanges();

return true;

}

}

public bool CheckedTenantProof(int idProof, int UserId)

{

using (var db = new RentContext())

{

var takenProduct = db.TakenProducts.FirstOrDefault(m => m.Id == idProof && m.UserId == UserId && m.IsDeleted == false);

if (takenProduct == null) return false;

takenProduct.TenantProof = true;

db.TakenProducts.AddOrUpdate(takenProduct);

db.SaveChanges();

return true;

}

}

public void DeleteTakenProductById(int id, int userId)

{

using (var db = new RentContext())

{

var takenProduct = db.TakenProducts.FirstOrDefault(m => m.Id == id && m.Product.UserId == userId);

takenProduct.IsDeleted = true;

db.TakenProducts.AddOrUpdate(takenProduct);

db.SaveChanges();

}

}

public Category[] GetAllGategories()

{

using (var db = new RentContext())

{

return db.Categories.Where(m => m.IsDeleted == false).ToArray();

}

}

public bool CheckedNoIsTakenProductByProductId(int id)

{

using (var db = new RentContext())

{

var product = db.Products.Include(m => m.User).FirstOrDefault(m => m.Id == id);

if (product == null) return false;

product.IsTaken = false;

db.Products.AddOrUpdate(product);

db.SaveChanges();

return true;

}

}

public bool RegistrationCreateUser(RegisterTag registerTag)

{

var user = new User()

{

Login = registerTag.Email,

Password = registerTag.Password,

Name = registerTag.Name,

Surname = registerTag.Surname,

PhotoUrl = registerTag.PhotoUrl,

City = registerTag.City,

Продолжение приложения Б

Street = registerTag.Street,

Home = registerTag.Home,

Longitude = registerTag.Longitude,

Latitude = registerTag.Latitude,

PhoneNumber = registerTag.PhoneNumber,

IsDeleted = false,

RoleId = 2,

SumMoney = 0

};

using (var db = new RentContext())

{

var userIsLive = db.Users.FirstOrDefault(m => m.Login == registerTag.Email);

if (userIsLive != null) return false;

db.Users.AddOrUpdate(user);

db.SaveChanges();

return true;

}

}

public bool PayMoneyForUserByLogin(decimal money, string login)

{

using (var db = new RentContext())

{

var user=db.Users.FirstOrDefault(m => m.Login == login);

if (user == null) return false;

user.SumMoney += money;

db.Users.AddOrUpdate(user);

db.SaveChanges();

return true;

}

}

public TakenProduct GetTakenProductById(int takenProductId)

{

using (var db = new RentContext())

{

return db.TakenProducts.FirstOrDefault(m => m.Id == takenProductId);

}

}

public RegisterTag AddLongAndLatiByAddress(string address,RegisterTag registerTag)

{

try

{

var client = new RestClient("https://geocode-maps.yandex.ru/1.x/");

var apikey = "\*\*\*";

var geocode = address;

var format = "json";

var request = new RestRequest("?apikey=" + apikey + "&geocode=" + geocode + "&format=" + format, Method.GET);

var content = client.Execute(request).Content;

MainClass res = JsonConvert.DeserializeObject<MainClass>(content);

String pos = res.response.GeoObjectCollection.featureMember[0].GeoObject.Point.pos;

String[] words =pos.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

registerTag.Latitude = words[0];

registerTag.Longitude = words[1];

return registerTag;

}

catch (Exception e)

{

Продолжение приложения Б

return registerTag;

}

}

public bool DeleteUserById(int id)

{

using (var db = new RentContext())

{

var user= db.Users.FirstOrDefault(m => m.Id == id && m.IsDeleted==false);

if (user == null) return false;

user.IsDeleted = true;

db.Users.AddOrUpdate(user);

db.SaveChanges();

return true;

}

}

public bool CreateCategory(Category category)

{

using (var db = new RentContext())

{

db.Categories.AddOrUpdate(category);

db.SaveChanges();

return true;

}

}

public bool DeleteCategoryById(int id)

{

using (var db = new RentContext())

{

var category = db.Categories.FirstOrDefault(m => m.Id == id && m.IsDeleted == false);

if (category == null) return false;

category.IsDeleted = true;

db.Categories.AddOrUpdate(category);

db.SaveChanges();

return true;

}

}

public bool UpdateCategory(Category category)

{

using (var db = new RentContext())

{

db.Categories.AddOrUpdate(category);

db.SaveChanges();

return true;

}

}

public Category GetCategoryById(int id)

{

using (var db = new RentContext())

{

return db.Categories.FirstOrDefault(m => m.Id == id);

}

}

public bool DeleteProductById(int id)

{

Окончание приложения Б

using (var db = new RentContext())

{

var product = db.Products.FirstOrDefault(m => m.Id == id && m.IsDeleted==false);

if (product == null) return false;

product.IsDeleted = true;

db.Products.AddOrUpdate(product);

db.SaveChanges();

return true;

}

}

public bool UpdateProduct(Product product)

{

using (var db = new RentContext())

{

db.Products.AddOrUpdate(product);

db.SaveChanges();

return true;

}

}

public bool CreateOrder(Order order)

{

using (var db = new RentContext())

{

db.Orders.Add(order);

db.SaveChanges();

return true;

}

}

public Order LastOrderByUserId(int id)

{

using (var db = new RentContext())

{

return db.Orders.Include(m => m.User).FirstOrDefault(m => m.UserId == id && m.IsDeleted==false);

}

}

public Order GetOrderById(int id)

{

using (var db = new RentContext())

{

return db.Orders.Include(m=>m.User).FirstOrDefault(m => m.Id == id);

}

}

public bool UpdateOrder(Order order)

{

using (var db = new RentContext())

{

db.Orders.AddOrUpdate(order);

db.SaveChanges();

return true;

}

}

}