*Федеральное государственное бюджетное образовательное учреждение высшего образования*

*«Владимирский государственный университет*

*имени Александра Григорьевича и Николая Григорьевича Столетовых»*

*Кафедра информационных систем и программной инженерии*

***ПОЯСНИТЕЛЬНАЯ ЗАПИСКА***

***к курсовому проекту по дисциплине   
"Технологии программирования"***

***на тему***

*Проектирование и разработка программной системы   
информационной системы «Кадастровое управление»*

*Выполнил: Шватов М.*

*Принял: доц. кафедры ИСПИ*

*Верешинин В.В.*

*Владимир, 2017*

**Аннотация**

Созданный прототип программной системы представляет собой web-приложение, по теме «Кадастровое управление» будет выглядеть как сайт, размещенный в интернете.

Данный курсовой проект создан с целью автоматизации однотипной и рутинной работы. Система представляет собой централизованное хранилище данных, с удобным интерфейсом доступа к ним. Со стороны пользователя система должна предоставлять информацию о недвижимости, земли, а также предоставлять возможность создавать заявки на регистрацию земли. Cо стороны сотрудников, необходимо упростить их работу, сделать ее удобнее, а также реализовать возможность добавления новых земель, одобрять заявки на регистрацию земли, отклонять, отправлять на доработку.

Created a prototype of a software system based on the theme "Сadastral Administration" is a web-application that will look like a site hosted on the Internet.

This course project is designed to automate a single-type and routine work. The system is a centralized data store, with a convenient interface to access them. On the part of the user, the system must provide information on real estate, land, and also provide the opportunity to create applications for land registration. From the side of the staff, it is necessary to simplify their work, make it more convenient, and also realize the possibility of adding new land, approve applications for land registration, reject, send for revision.

Курсовой проект представлен на 54 страницах, рисунков – 18, использованных источников – 7, приложений – 2.

**Содержание**

Оглавление

[ВВЕДЕНИЕ 3](#_Toc501072621)

[1 Описание предметной области 4](#_Toc501072622)

[1.1. Постановка задачи 4](#_Toc501072623)

[1.2. Словарь предметной области 4](#_Toc501072624)

[1.3. Перечень ролей, предусмотренных в системе 5](#_Toc501072625)

[2. проектирование системы 6](#_Toc501072626)

[2.2. Спецификация прецедентов 6](#_Toc501072627)

[2.3. Диаграмма классов 9](#_Toc501072628)

[2.4. Диаграмма состояний 12](#_Toc501072629)

[2.5. Диаграмма последовательностей 13](#_Toc501072630)

[3. РЕАЛИЗАЦИЯ СИСТЕМЫ 15](#_Toc501072631)

[3.1 Архитектурный шаблон MVC. 15](#_Toc501072632)

[3.2. База данных. 16](#_Toc501072633)

[3.2.1. Описание взаимоотношений между сущностями 17](#_Toc501072634)

[3.3. Авторизация 17](#_Toc501072635)

[4. ОБЗОР РЕАЛИЗОВАННОЙ СИСТЕМЫ 19](#_Toc501072636)

[заключение 26](#_Toc501072637)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc501072638)

[ПРИЛОЖЕНИЕ А Исходный скрипт базы данных 28](#_Toc501072639)

[Приложение б. Исходный код программной системы 35](#_Toc501072640)

# ВВЕДЕНИЕ

На современном этапе развития производительных сил и производственных отношений информация стала товаром со всеми присущими ей свойствами. Сегодня существуют информационная промышленность, национальные информационные ресурсы, происходит переход от индустриальной экономики к экономике, основанной на информации. Особенно это актуально для транспорта, как отрасли народного хозяйства. Основной целью курсового проекта является разработка и реализация программной системы по предметной области “Кадастровое управление”.

Данный курсовой проект создан с целью автоматизации однотипной и рутинной работы. Система представляет собой централизованное хранилище данных, с удобным интерфейсом доступа к ним. Со стороны пользователя система должна предоставлять информацию о недвижимости, земли, а также предоставлять возможность создавать заявки на регистрацию земли. Cо стороны сотрудников, необходимо упростить их работу, сделать ее удобнее, а также реализовать возможность добавления новых земель, одобрять заявки на регистрацию земли, отклонять, отправлять на доработку.

# Описание предметной области

* 1. Постановка задачи

Необходимо решить следующие задачи:

* изучение особенностей конкретной предметной области, относящихся к теме курсового проекта;
* анализ возможных подходов и методов решения с обоснованием выбранного подхода;
* разработка моделей программной системы;
* реализация функционала программной системы.
* реализация программной системы с сохранением результатов работы в репозиторий.

Исходные данные:

* методология проектирования и разработки RUP;
* язык моделирования UML;
* платформа разработки ASP.NET;
* объектно-ориентированная технология доступа к данным – ADO.NET Entity Framework;
* язык интегрированных запросов Linq;
* функции системы: Ввод данных в систему о кадастровых объектах, работа со справочниками, работа с типами учета объектов в кадастровом управлении, заявки в системе: подача заявок на регистрацию объекта, рассмотрение заявки, отклонение заявки, отправка на доработку.
  1. Словарь предметной области

1. Пользователь – зарегистрированное лицо, пользующийся системой; характеристики: фамилия, имя, отчество, адрес, телефон, электронная почта; действия: регистрация, вход в систему;

2. Запрос – заявка на запрашиваемые данные, результатом получения данных будет в соответствии с выбранными фильтрами (поиск по адресу, по имени владельца, по фамилии владельца, стоимости земли или площади):

3. Администратор – должностное лицо, имеющий привилегированные права по использованию системы;

4. Модератор – должностное лицо, модерирущий запросы на регистрацию земель от пользователей, в результате модерации могут быть два состояния:

4.1. Запрос на регистрацию отклонен, в виде недостаточности предоставленной информации;

4.2. Запрос на регистрацию одобрен, в результате которого уполномоченное лицо приезжает на место регистрации земли и идет оформление документов с последующим внесением соответствующей информации в централизованное хранилище данных системы.

* 1. Перечень ролей, предусмотренных в системе
  2. Клиент — зарегистрированный пользователь с ограниченными возможностями в системе, для которого доступны операции запроса соответствующих данных, создания заявки;
  3. Модератор — зарегистрированный пользователь с расширенными правами, предусмотренными в соответствии с ролевой моделью системы, в качестве основных функций, которые доступны модератору — управления заявками, запросами с последующим отклонением/одобрением и др.;
  4. Администратор — зарегистрированный пользователь с самыми привилегированными правами доступа в системе. Администратору доступны все функции, которые предусмотрены в системе, включая управление над пользователями — создание, редактирование страниц доступа, смена роли, блокировка пользователя за нарушения в системе.

# проектирование системы

Диаграммой прецедентов называется диаграмма, на которой показана совокупность прецедентов и актеров, а также отношения между ними. Диаграмма прецедентов представлена на рисунке 1.

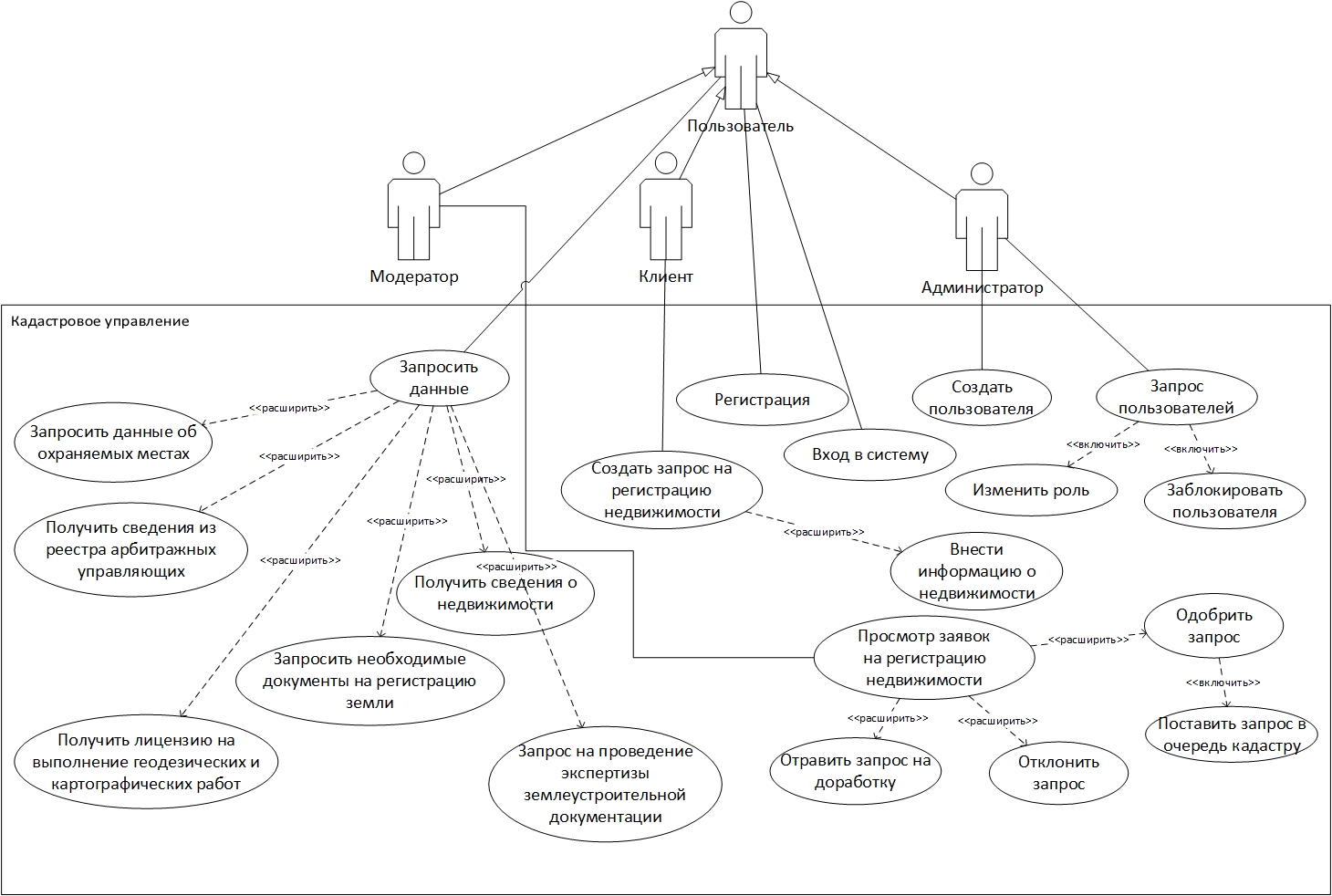
****

Рисунок 1. Диаграмма прецедентов

2.2. Спецификация прецедентов

* + 1. Название прецедента: **«Запросить данные об охраняемых местах»**

**Предусловие:** Пользователь вошел в систему.

**Действующее лицо:** Авторизованный пользователь.

**Основной поток:**

1. Пользователь запросил страницу для запроса данных;
2. Система обработала запрос пользователя и вернула страницу для создания запроса;
3. Пользователь заполняет все необходимые фильтры и нажимает на кнопку «Поиск»;
4. Система обрабатывает запрос пользователя и фильтрует выборку на основе фильтров;

**Альтернативный поток:**

1. Ошибка при обработке запроса, введены некорректные данные.

Вывод сообщения об ошибке.

**Постусловие:** Зарегистрированный пользователь получил запрошенные данные.

* + 1. Название прецедента: **«Создать запрос на регистрацию недвижимости»**

**Предусловие:** Пользователь вошел в систему.

**Действующее лицо:** Авторизованный пользователь — клиент.

**Основной поток:**

1. Пользователь запросил страницу для создания запроса на регистрацию недвижимости;
2. Система обработала запрос пользователя и вернула страницу для создания запроса на регистрацию недвижимости;
3. Пользователь заполняет все необходимые данные и нажимает на кнопку «Создать запрос»;
4. Система обрабатывает запрос пользователя и сохраняет заявку со статусом «Pending» (ожидание);

**Альтернативный поток:**

1. Пользователь не заполнил все обязательные данные. Система вернула соответствующее сообщение.
2. Пользователь ввел некорректные данные. Система вернула соответствующее сообщение.

**Постусловие:** Зарегистрированный пользователь успешно создал заявку на регистрацию недвижимости.

* + 1. Название прецедента: **«Запрос на проведение экспертизы землеустроительной документации»**

**Предусловие:** Пользователь вошел в систему.

**Действующее лицо:** Авторизованный пользователь — клиент.

**Основной поток:**

1. Пользователь запросил страницу для создания запроса на проведение экспертизы землеустроительной документации;
2. Система обработала запрос пользователя и вернула страницу для создания запроса на проведение экспертизы землеустроительной документации;
3. Пользователь заполняет все необходимые данные и нажимает на кнопку «Создать запрос на экспертизу»;
4. Система обрабатывает запрос пользователя и сохраняет заявку со статусом «Pending» (ожидание);

**Альтернативный поток:**

1. Пользователь не заполнил все обязательные данные. Система вернула соответствующее сообщение.
2. Пользователь ввел некорректные данные. Система вернула соответствующее сообщение.

**Постусловие:** Зарегистрированный пользователь успешно создал заявку на проведение экспертизы землеустроительной документации.

* + 1. Название прецедента: **«Запрос на проведение экспертизы землеустроительной документации»**

**Предусловие:** Пользователь вошел в систему.

**Действующее лицо:** Авторизованный пользователь — администратор.

**Основной поток:**

1. Запрос списка зарегистрированных пользователей;
2. Система обработала запрос пользователя и вернула страницу со списком зарегистрированных пользователей;
3. Пользователь фильтрует список пользователей в соответствии с количеством нарушений в системе;
4. Система обрабатывает запрос пользователя и возвращает отфильтрованный список пользователей в соответствии с выбранными фильтрами;
5. Пользователь нажимает на кнопку «Заблокировать» для выбранного пользователя.

**Альтернативный поток:**

1. Ошибка при запросе пользователей на основе выбранных фильтров.

**Постусловие:** Зарегистрированный пользователь успешно заблокировал пользователя.

* 1. Диаграмма классов

Модель классов описывает статическую структуру системы: объекты и отношения между ними, атрибуты и операции для каждого класса объектов. Диаграмма классов представлена на рисунке 2.



Рисунок 2. Диаграмма классов

**Описание сущностей системы:**

* 1. **Cadastr —** более абстрактное название объектов, подлежащих к налогообложению, который включает в себя идентификационный номер кадастра и тип. В ней содержится метод получения всех кадастров.
  2. **Immovable —** тип кадастра, представляющий собой налогооблагаемую недвижимость, которую описывают соответствующие атрибуты. Данная сущностью включает в себя методы получения сведения о недвижимого имущества, изменении данных о недвижимом имуществе, создании заявки на получении лицензии и др.,
  3. **ImmovableType —** тип недвижимого имущества, представляющий собой описание объекта, исходя из которых можно определить, к какому виду или типу относится данное недвижимое имущество (офис, частный дом, торговое помещение и др.). Содержит в себе методы создания, редактирования и удаления типа недвижимого имущества.
  4. **Land —** тип кадастра, представляющий собой земельный участок, предназначенный для строительных целей либо других, описывающаяся соответствующими атрибутами, и, включающая в себе логику создания, редактирования данных о земле, получении информации о хозяине, стоимости и т.д.
  5. **LandType —** представляющий собой сущность, представляющий собой описание объекта, из которых можно определить, к какому виду или типу относится данный земельный участок. Содержит в себе методы создания, редактирования и удаления земельного участка.
  6. **LicenseRequest —** сущность, представляющая собой описание заявки на получение лицензии на земельный участок. Содержим в себе бизнес-функции, предназначенные для работы с заявками, включающая одобрение, изменение, сохранение.
  7. **Owner —** Владелец земельного участка либо недвижимого имущества.
  8. **User —** пользователь, зарегистрированный в системе, включающий в себя атрибуты пользователя и методы для манипуляции над пользователями.
  9. **Role —** сущность, предназначенная для делегирования пользователям, соответствующих функций, с целью разграничить доступ к функционалу системы. Содержим в себе методы создания, редактирования роли.
  10. **Claim —** сущность, содержащаяся в себе ссылки о каждом пользователе системы и предназначенная для принятия решения об авторизации вызова какой-то функции или метода, отображение информации о текущем пользователе и тд.
  11. Диаграмма состояний

****Диаграмма состояний описывает последовательности состояний, вызываемые последовательностями событий. Диаграмма состояний представлена на рисунке 3.

Рисунок 3. Диаграмма состояний объекта «Заявка»

В диаграмме состояний объекта «Заявка» демонстрируется жизненный цикл объекта, начиная от создания и подтверждения либо отклонения. В данной диаграмме у заявки имеются несколько состояний, переход из одного состояния объекта в другое порождают соответствующие последовательности. На первом этапе ЖЦ цикла заявки на получение лицензии создается. Если данная заявка удовлетворяет требованиям, заложенные в системе и удовлетворяет требованиям кадастрового агента, то она подтверждается и жизненный цикл объекта заканчивается, если заявка не удовлетворяет требованиям кадастрового агента, то она отправляется на доработку до тех пор, пока не будет подтверждена либо отклонена, в противном заявка откланяется, если она не удовлетворяет требования системы.

* 1. Диаграмма последовательностей

Диаграмма последовательности показывает участников взаимодействия и последовательность сообщений, которыми они обмениваются. Она описывает взаимодействие системы с действующими лицами в процессе полного или частичного выполнения варианта использования. Диаграмма последовательностей прецедента «Создать заявку на получение лицензии» представлена на рисунке 4.



Рисунок 4. Диаграмма последовательностей «Создать заявку на получение лицензии»

1. РЕАЛИЗАЦИЯ СИСТЕМЫ
   1. Архитектурный шаблон MVC.

Model – View – Controller (модель – представление– контроллер) – это конструкционный шаблон, который описывает способ построения структуры приложения, сферы ответственности и взаимодействия каждой из частей в данной структуре.

Модель – содержит или представляет данные, с которыми работают пользователи. Они могут быть простыми моделями представлений, которые только представляют данные, передаваемые между представлениями и контроллерами; или же они могут быть моделями предметной области, которые содержат бизнес данные, а также операции, преобразования и правила для манипулирования этими данными.

Представление – применяется для визуализации некоторой части модели в виде пользовательского интерфейса.

Контроллер – обрабатывает поступающие запросы, выполняет операции с моделью и выбирает представления для визуализации пользователю.

* 1. База данных.

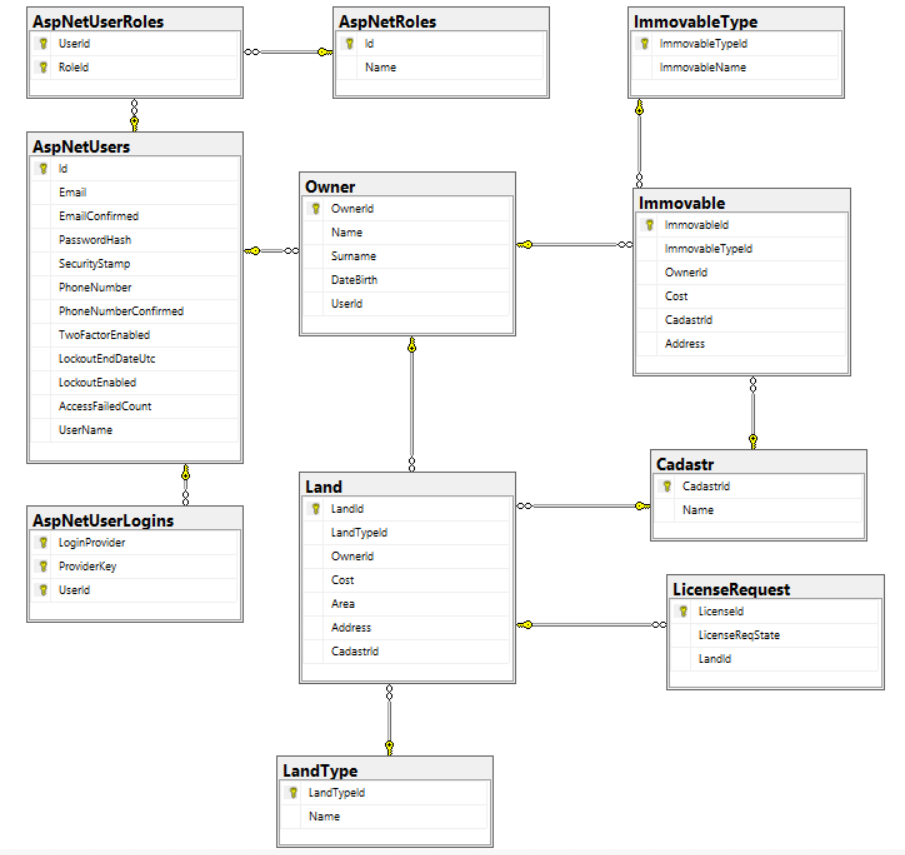
В качестве СУБД выбор пал на Microsoft SQL Express 2017, так как она бесплатная и есть опыт работы с ней. Схема базы данных представлена на рисунке 5.

Рисунок 5. Схема базы данных

3.2.1. Описание взаимоотношений между сущностями

* 1. **LandType – Land –** кратность один-ко-многим – У одного типа земли может быть несколько земель, то есть одного и того же типа может быть много земель.
  2. **ImmovableType – Immovable –** кратность один-ко-многим – У одного и того же типа недвижимости может существовать несколько недвижимостей.
  3. **Cadastr – Immovable –** кратность один-ко-многим – У одного объекта «кадастр» может существовать много недвижимых объектов.
  4. **Cadastr – Land –** кратность один-ко-многим – У одного объекта «кадастр» может существовать много земельных участков.
  5. **Owner – Land, Owner – Immovable –** кратность один-ко-многим – У одного владельца может быть несколько земельных участков и недвижимых объектов.
  6. Авторизация

ASP.NET Identity - это система, которая позволяет добавлять в приложение функционал для логирования. Пользователь может создать аккаунт и залогироваться, используя имя пользователя или пароль, либо он может использовать такие провайдеры, как Facebook, Google, Microsoft Account, Twitter и т.д.

По умолчанию при регистрации первого пользователя создается следующий набор таблиц:

1. **AspNetRoles**: содержит определения ролей.
2. **AspNetUserClaims**: таблица, хранящая набор клеймов (claim). Claim представляет иную модель авторизации по сравнению с ролями. Иными словами, claim содержит некоторую информацию о пользователе, например, адрес электронной почты, логин, возраст и т.д. И эта информация позволяет идентифицировать пользователя и наделить его соответствующими правами доступа.
3. **AspNetUserLogins:** таблица логинов пользователя.
4. **AspNetUserRoles:** таблица, устанавливающая для пользователей определенные роли.
5. **AspNetUsers:** таблица пользователей.
6. ОБЗОР РЕАЛИЗОВАННОЙ СИСТЕМЫ
   1. **Неавторизованный пользователь.** Неавторизованному пользователю доступна только главная страница системы. Любое нажатие на другую кнопку, пользователь автоматически переходит на страницу авторизации. Главная страница системы представлена на рисунке 5.

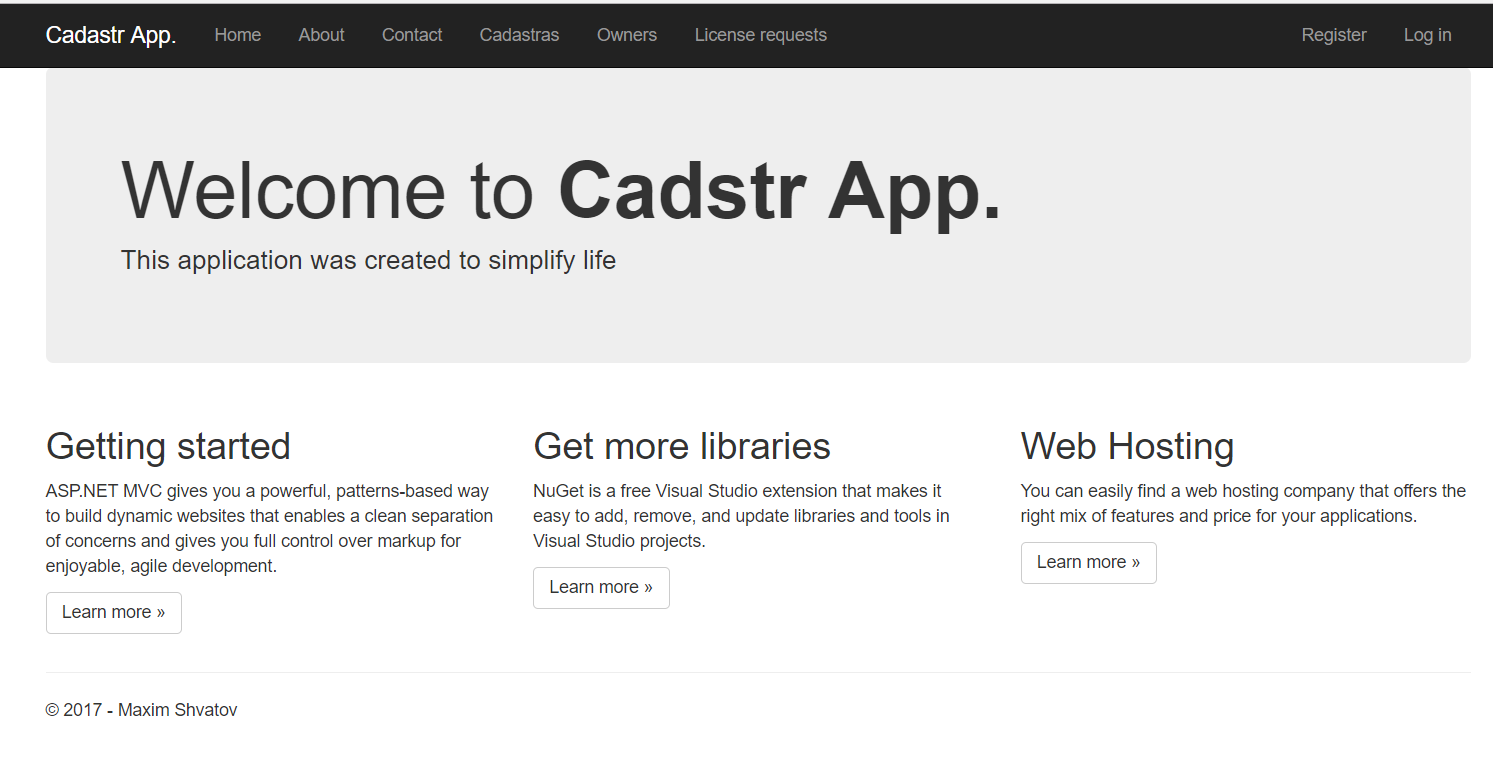


Рисунок 5. Главная страница.

* 1. **Авторизованный пользователь. Роль Client.** Пользователю с ролью Client доступен минимальный набор функций в системе. Страница с кадастрами представлена на рисунке 6.

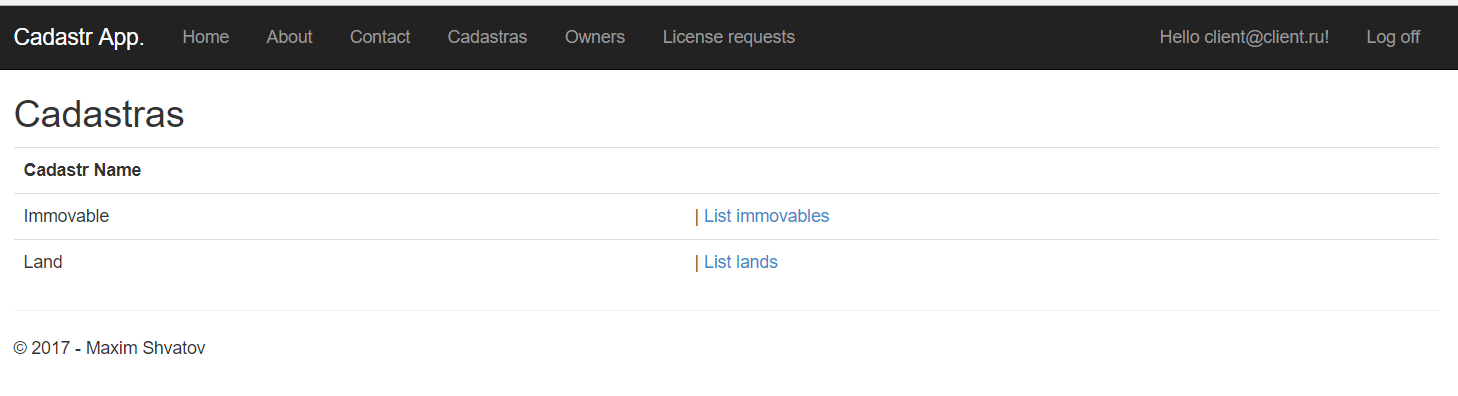


Рисунок 6. Страница с кадастрами

* 1. **Авторизованный пользователь. Роль Client.** Страница с недвижимостью представлена на рисунке 7.

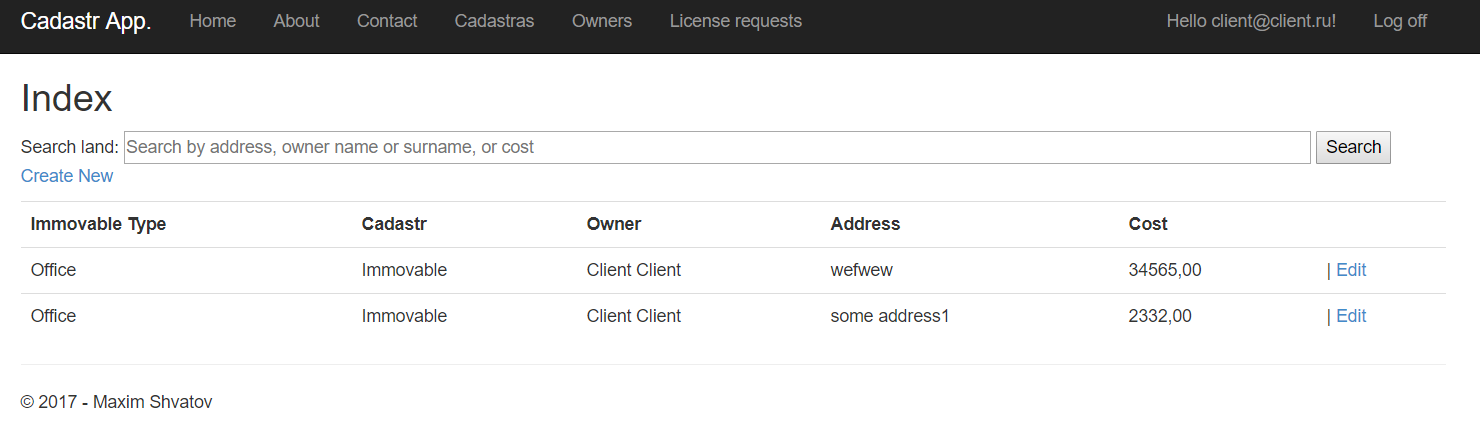


Рисунок 7. Страница с недвижимостью текущего пользователя

* 1. **Авторизованный пользователь. Роль Client.** Страница редактирования данных о недвижимости текущего клиента представлена на рисунке 8.

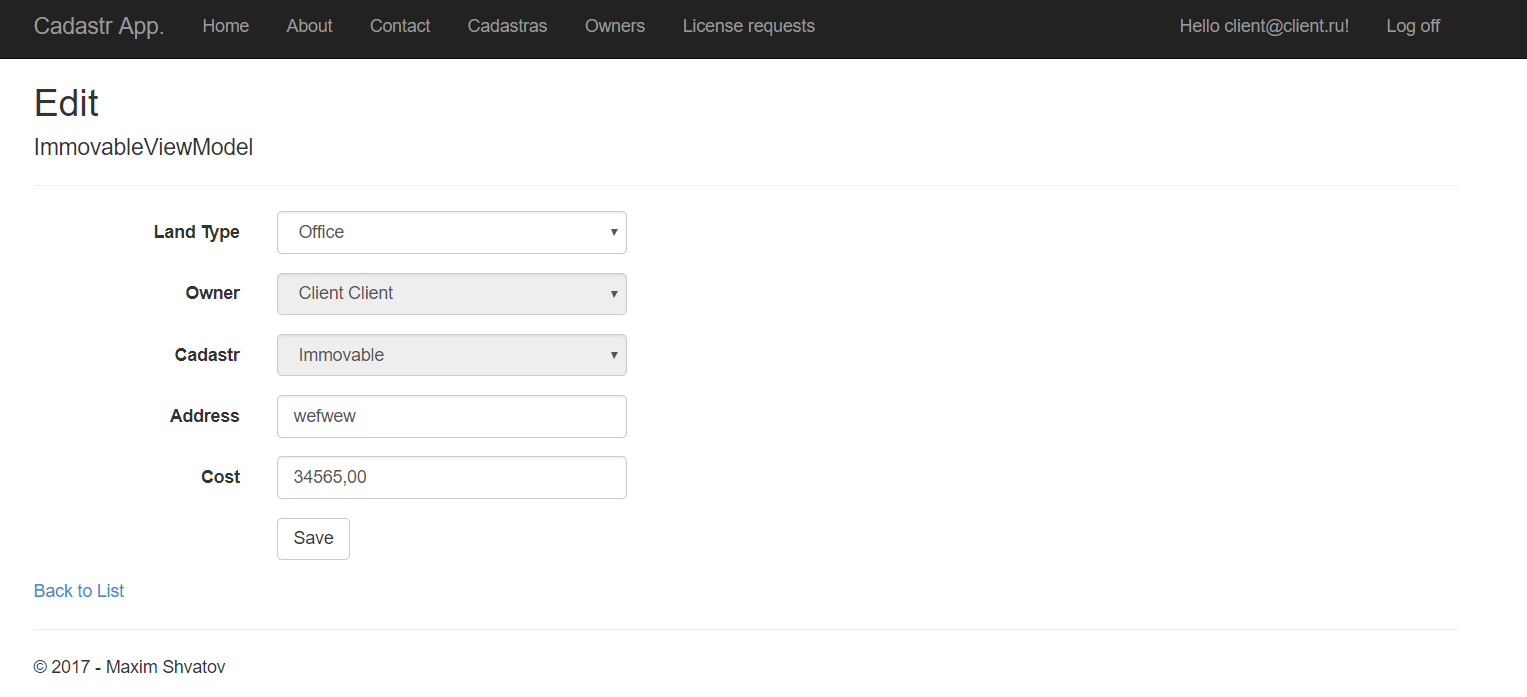


Рисунок 8. Страница редактирования недвижимости

* 1. **Авторизованный пользователь. Роль Client.** Результаты поиска информации по стоимости недвижимости представлена на рисунке 9.

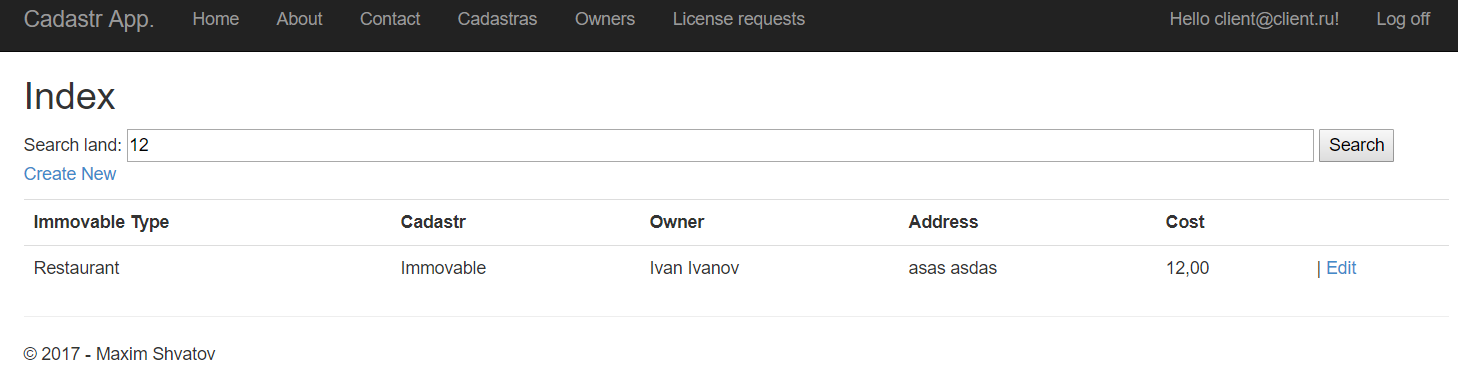


Рисунок 9. Поиск по стоимости

* 1. **Авторизованный пользователь. Роль Client.** Страницасозданияновой недвижимости для текущего пользователя представлена на рисунке 10.

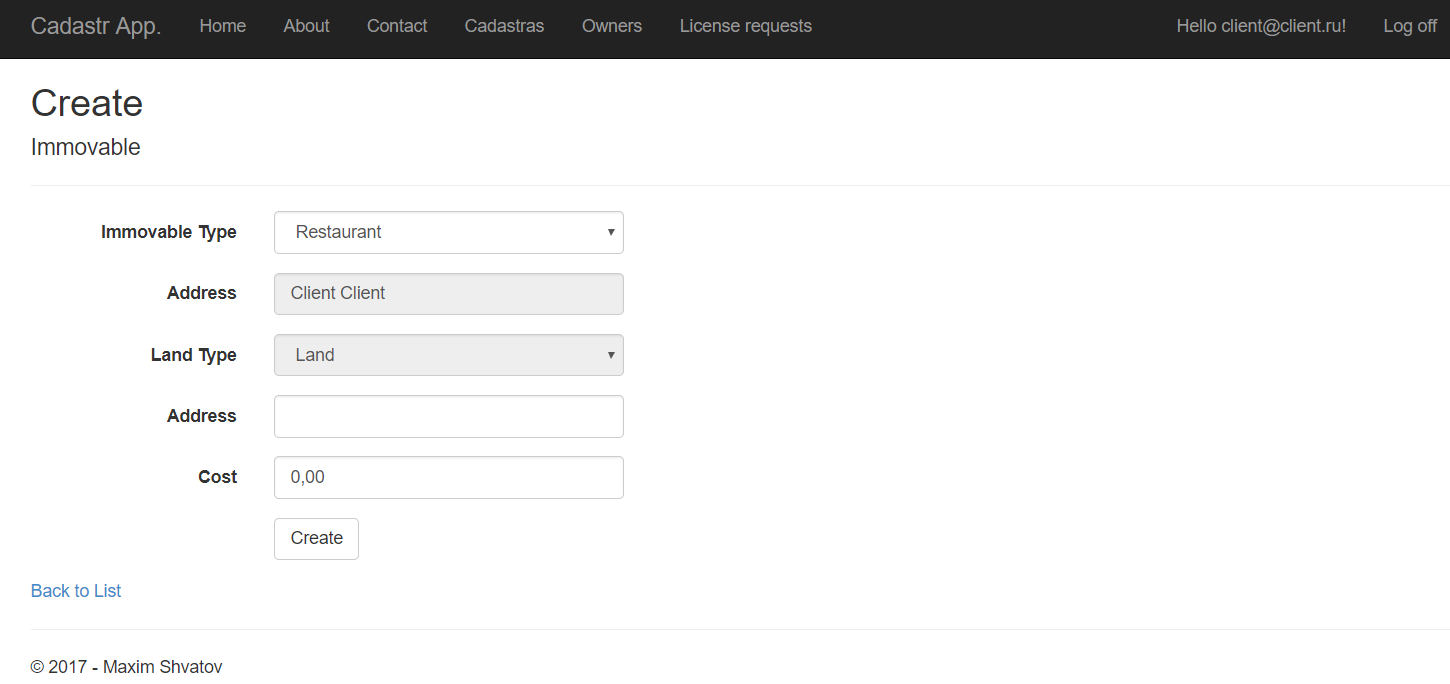


Рисунок 10. Страница создания новой недвижимости

* 1. **Авторизованный пользователь. Роль Client.** Страница с землей текущего клиента представлена на рисунке 11.

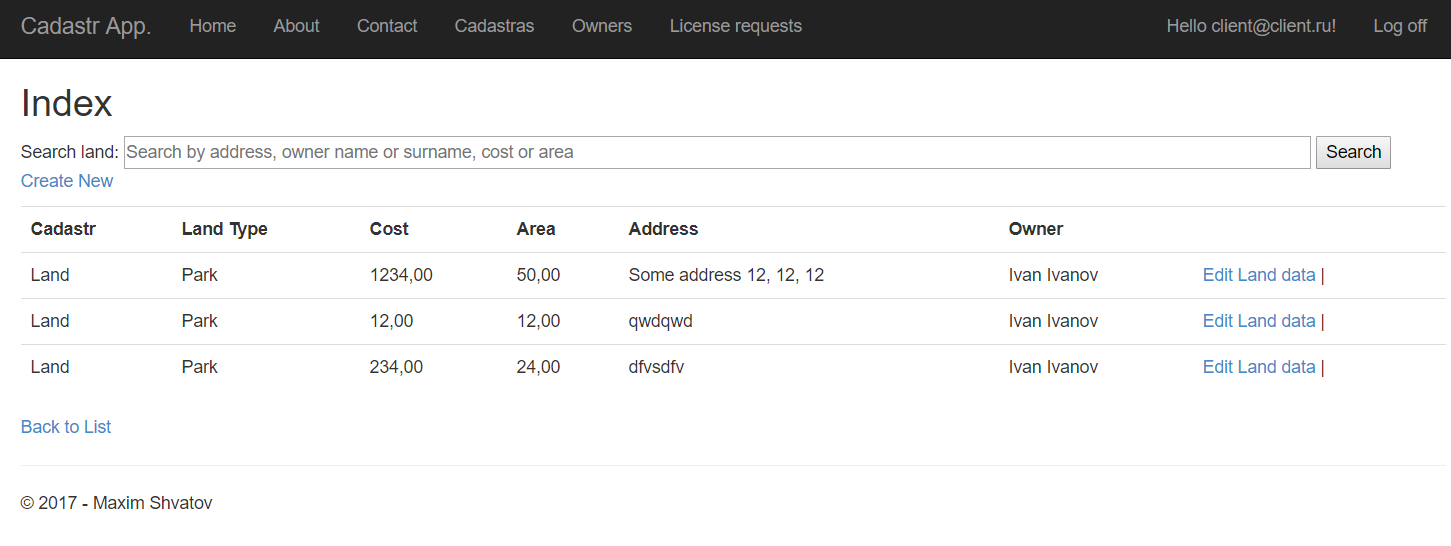


Рисунок 11. Страница с землей текущего клиента

* 1. **Авторизованный пользователь. Роль Client.** Страница редактирования земли для текущего пользователя представлена на рисунке 12.

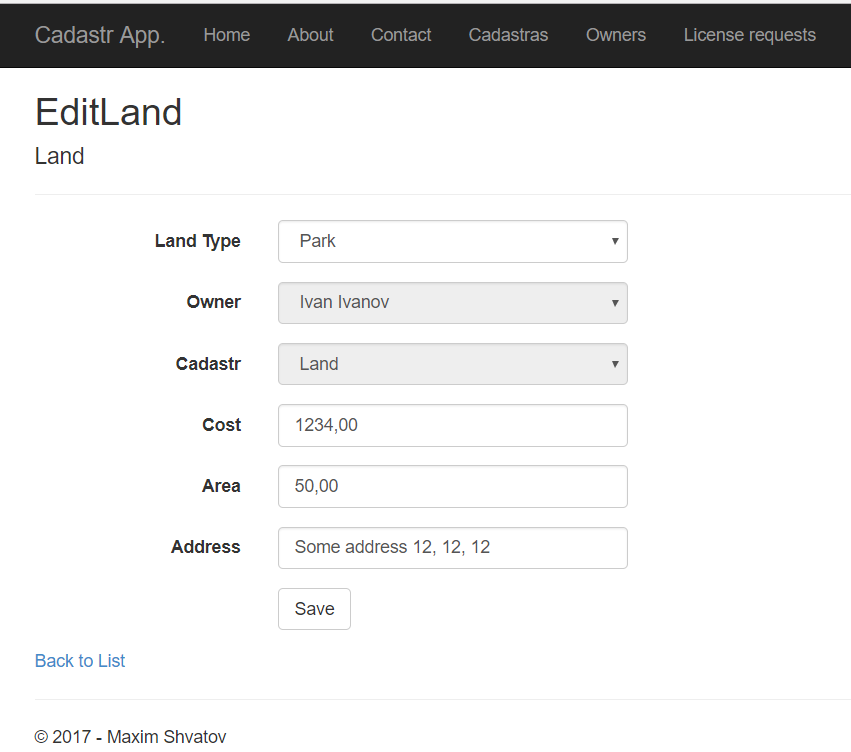


Рисунок 12. Страница редактирования земли

* 1. **Авторизованный пользователь. Роль Client.** Страница с заявками на регистрацию земли представлена на рисунке 13.

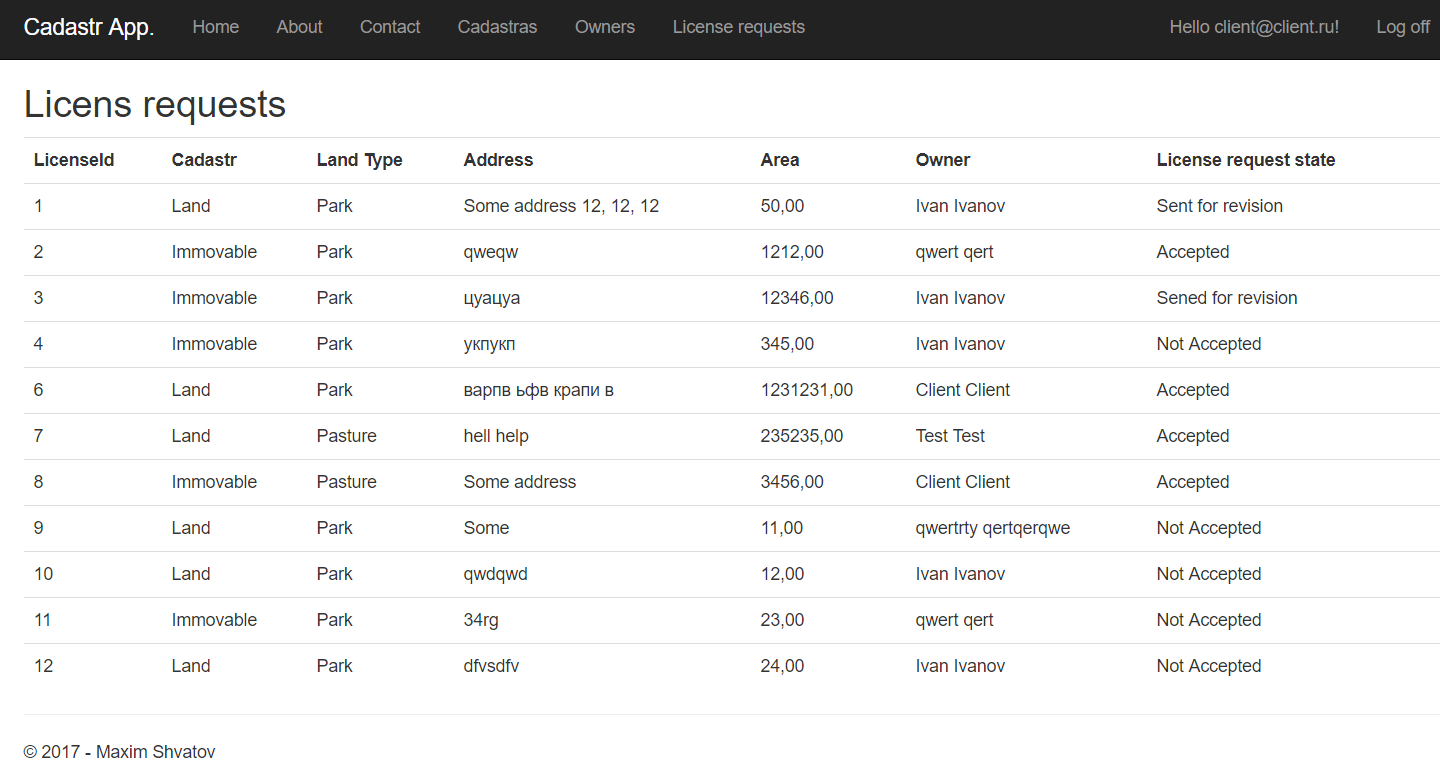


Рисунок 13. Страница с заявками

* 1. **Авторизованный пользователь. Роль Administration, Moderator.** Страница с землей представлена на рисунке 14.

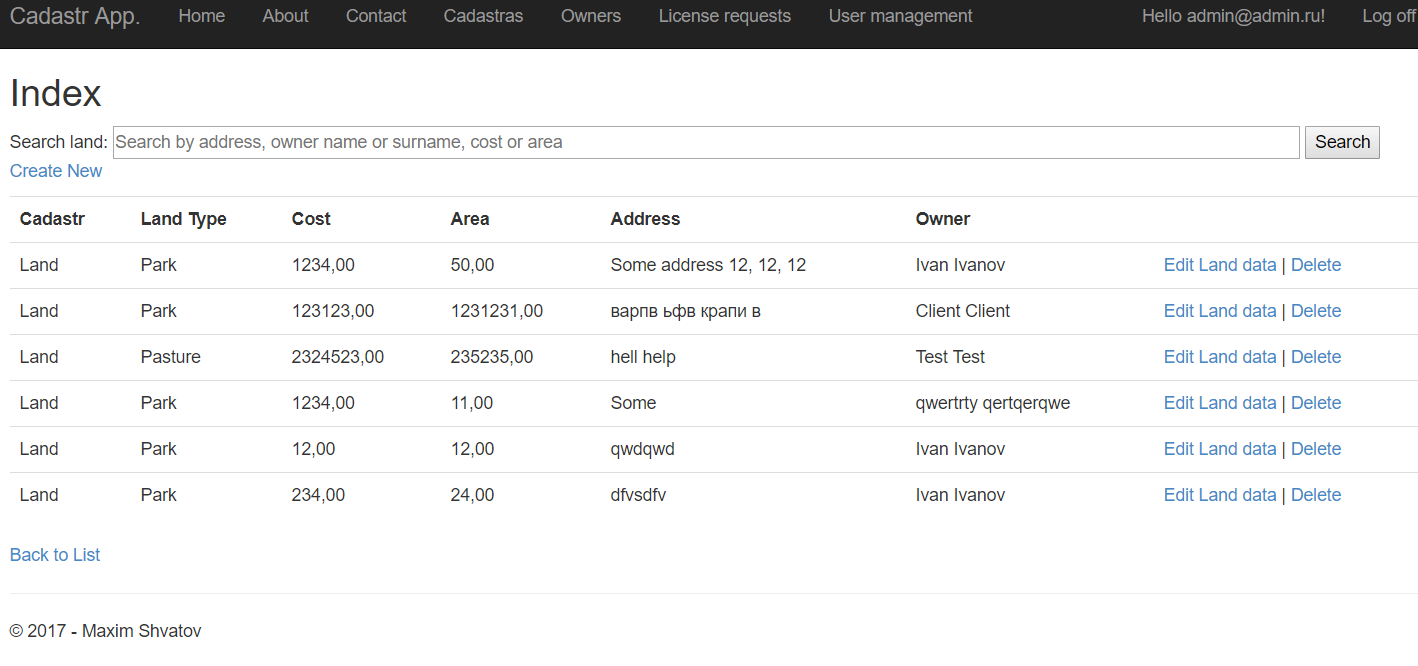


Рисунок 14. Страница с землей

* 1. **Авторизованный пользователь. Роль Administration, Moderator.** Страница редактирования земли представлена на рисунке 15.

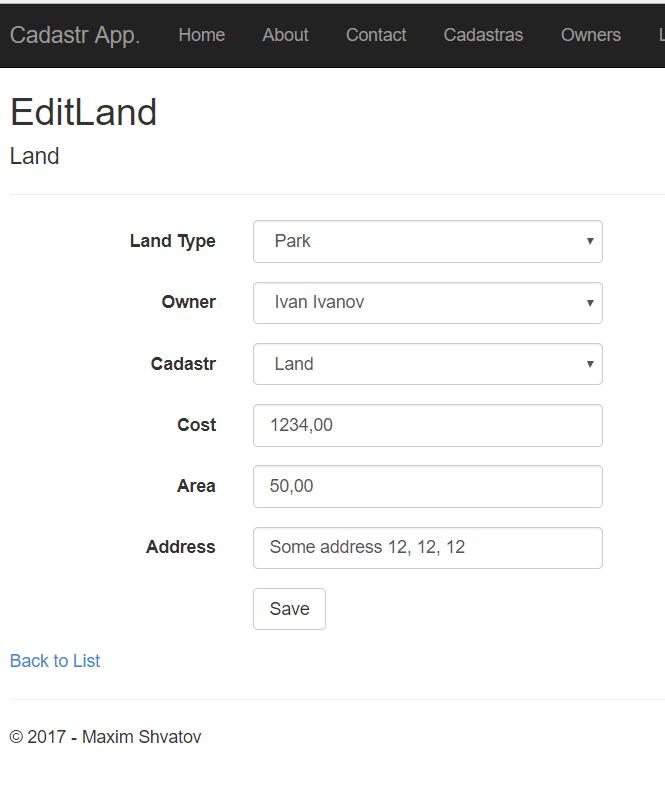


Рисунок 15. Страница редактирования земли

* 1. **Авторизованный пользователь. Роль Administration, Moderator.** Страница с заявками.

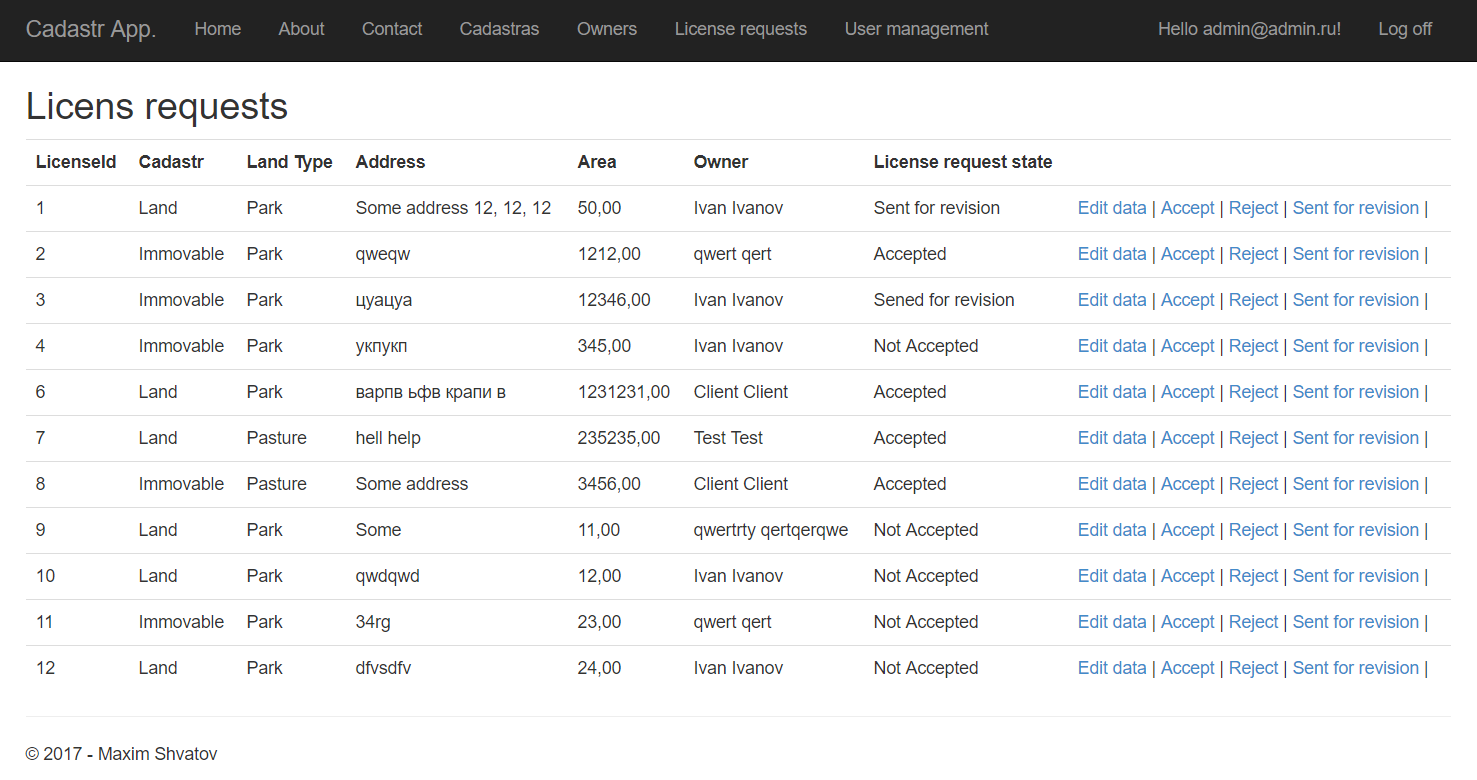


Рисунок 16. Страница с заявками

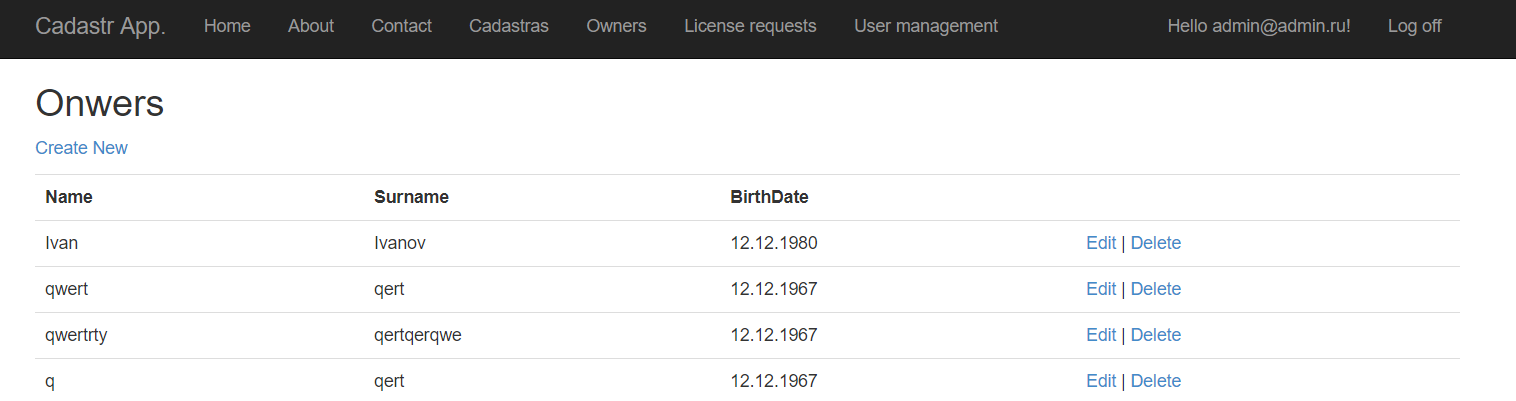
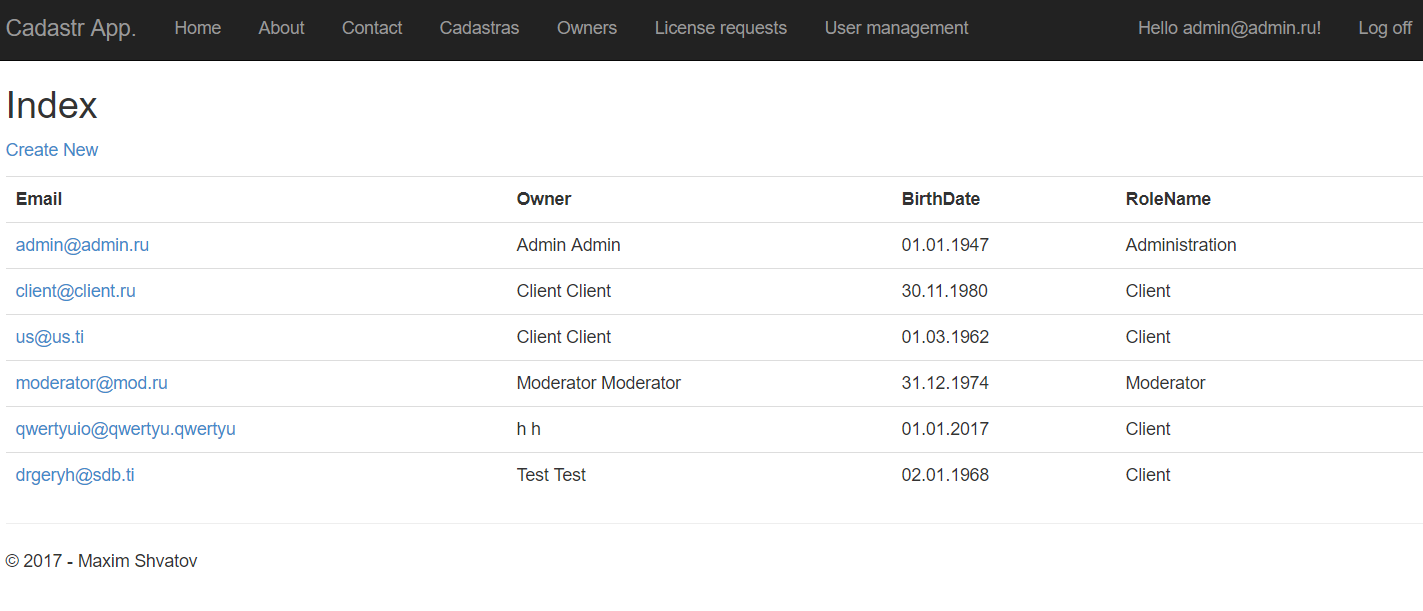
* 1. **Авторизованный пользователь. Роль Administration, Moderator.** Страница с владельцами земельных участков и недвижимого имущества представлена на рисунке 17.

Рисунок 17. Страница с владельцами

* 1. **Авторизованный пользователь. Роль Administration.** Страница с пользователями системы.

Рисунок 18. Страница с пользователями системы 

# заключение

В результате выполнения курсового проекта была разработана и реализована программная система «Кадастровое управление».

Был произведен анализ предметной области, смоделирована структура объектов и спроектирована структура программной системы согласно нотации UML. Приложение реализовано в программной среде разработки Microsoft Visual Studio 2017. Структура приложения полностью соответствует шаблону MVC. Разработка данного приложения сопровождалась отображением в репозитории.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Липаев В.В. Проектирование программных систем. М.:Высш.шк, 1990
2. Буч Г. Объекто-ориентированное проектирование / пер. с анг. Конкорд, 1996г.
3. Майерс Г. Надежность программного обеспечения. М.: Мир, 1980
4. Коленов Г.Н. CASE структурный системный анализ (автоматизация и применения). – М.: «ЛОРИ», 1996
5. Шмуллер Дж. Освой самостоятельно UML. 2-е изд. М.: изд-дом «Вильямс», 2002
6. Колберстон Р., Браун К., Кобб Г. Быстрое тестирование М.: изд.дом «Вильямс», 2002
7. Кармайкл Э., Хейвуд Д. Быстрая и качественная разработка программного обеспечения. М.: изд. дом «Вильямс, 2003»

ПРИЛОЖЕНИЕ А Исходный скрипт базы данных

USE [master]

GO

CREATE DATABASE [CadastraDB]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'CadastraDB', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\CadastraDB.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

LOG ON

( NAME = N'CadastraDB\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\CadastraDB\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

GO

/\*\*\*\*\*\* Object: Table [dbo].[AspNetRoles] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetRoles](

[Id] [nvarchar](128) NOT NULL,

[Name] [nvarchar](256) NOT NULL,

CONSTRAINT [PK\_dbo.AspNetRoles] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[AspNetUserClaims] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUserClaims](

[Id] [int] IDENTITY(1,1) NOT NULL,

[UserId] [nvarchar](128) NOT NULL,

[ClaimType] [nvarchar](max) NULL,

[ClaimValue] [nvarchar](max) NULL,

CONSTRAINT [PK\_dbo.AspNetUserClaims] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[AspNetUserLogins] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUserLogins](

[LoginProvider] [nvarchar](128) NOT NULL,

[ProviderKey] [nvarchar](128) NOT NULL,

[UserId] [nvarchar](128) NOT NULL,

CONSTRAINT [PK\_dbo.AspNetUserLogins] PRIMARY KEY CLUSTERED

(

[LoginProvider] ASC,

[ProviderKey] ASC,

[UserId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[AspNetUserRoles] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUserRoles](

[UserId] [nvarchar](128) NOT NULL,

[RoleId] [nvarchar](128) NOT NULL,

CONSTRAINT [PK\_dbo.AspNetUserRoles] PRIMARY KEY CLUSTERED

(

[UserId] ASC,

[RoleId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[AspNetUsers] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUsers](

[Id] [nvarchar](128) NOT NULL,

[Email] [nvarchar](256) NULL,

[EmailConfirmed] [bit] NOT NULL,

[PasswordHash] [nvarchar](max) NULL,

[SecurityStamp] [nvarchar](max) NULL,

[PhoneNumber] [nvarchar](max) NULL,

[PhoneNumberConfirmed] [bit] NOT NULL,

[TwoFactorEnabled] [bit] NOT NULL,

[LockoutEndDateUtc] [datetime] NULL,

[LockoutEnabled] [bit] NOT NULL,

[AccessFailedCount] [int] NOT NULL,

[UserName] [nvarchar](256) NOT NULL,

CONSTRAINT [PK\_dbo.AspNetUsers] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Cadastr] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Cadastr](

[CadastrId] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](100) NOT NULL,

CONSTRAINT [PK\_Cadastr] PRIMARY KEY CLUSTERED

(

[CadastrId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Immovable] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Immovable](

[ImmovableId] [int] IDENTITY(1,1) NOT NULL,

[ImmovableTypeId] [int] NOT NULL,

[OwnerId] [int] NOT NULL,

[Cost] [decimal](18, 0) NOT NULL,

[CadastrId] [int] NOT NULL,

[Address] [varchar](100) NOT NULL,

CONSTRAINT [PK\_Immovable] PRIMARY KEY CLUSTERED

(

[ImmovableId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[ImmovableType] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[ImmovableType](

[ImmovableTypeId] [int] IDENTITY(1,1) NOT NULL,

[ImmovableName] [varchar](50) NOT NULL,

CONSTRAINT [PK\_ImmovableType] PRIMARY KEY CLUSTERED

(

[ImmovableTypeId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Land] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Land](

[LandId] [int] IDENTITY(1,1) NOT NULL,

[LandTypeId] [int] NOT NULL,

[OwnerId] [int] NOT NULL,

[Cost] [decimal](18, 0) NOT NULL,

[Area] [decimal](18, 0) NOT NULL,

[Address] [varchar](200) NOT NULL,

[CadastrId] [int] NOT NULL,

CONSTRAINT [FK\_Land] PRIMARY KEY CLUSTERED

(

[LandId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[LandType] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[LandType](

[LandTypeId] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](100) NULL,

CONSTRAINT [FK\_LandType] PRIMARY KEY CLUSTERED

(

[LandTypeId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[LicenseRequest] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[LicenseRequest](

[LicenseId] [int] IDENTITY(1,1) NOT NULL,

[LicenseReqState] [varchar](20) NULL,

[LandId] [int] NOT NULL,

CONSTRAINT [LicenseRequest\_PK] PRIMARY KEY CLUSTERED

(

[LicenseId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Owner] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Owner](

[OwnerId] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](50) NOT NULL,

[Surname] [varchar](100) NOT NULL,

[DateBirth] [datetime] NOT NULL,

[UserId] [nvarchar](128) NULL,

CONSTRAINT [PK\_Owner] PRIMARY KEY CLUSTERED

(

[OwnerId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [RoleNameIndex] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

CREATE UNIQUE NONCLUSTERED INDEX [RoleNameIndex] ON [dbo].[AspNetRoles]

(

[Name] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [IX\_UserId] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

CREATE NONCLUSTERED INDEX [IX\_UserId] ON [dbo].[AspNetUserClaims]

(

[UserId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [IX\_UserId] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

CREATE NONCLUSTERED INDEX [IX\_UserId] ON [dbo].[AspNetUserLogins]

(

[UserId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [IX\_RoleId] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

CREATE NONCLUSTERED INDEX [IX\_RoleId] ON [dbo].[AspNetUserRoles]

(

[RoleId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [IX\_UserId] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

CREATE NONCLUSTERED INDEX [IX\_UserId] ON [dbo].[AspNetUserRoles]

(

[UserId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [UserNameIndex] Script Date: 15.12.2017 2:56:01 \*\*\*\*\*\*/

CREATE UNIQUE NONCLUSTERED INDEX [UserNameIndex] ON [dbo].[AspNetUsers]

(

[UserName] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, DROP\_EXISTING = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

GO

ALTER TABLE [dbo].[AspNetUserClaims] WITH CHECK ADD CONSTRAINT [FK\_dbo.AspNetUserClaims\_dbo.AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[AspNetUserClaims] CHECK CONSTRAINT [FK\_dbo.AspNetUserClaims\_dbo.AspNetUsers\_UserId]

GO

ALTER TABLE [dbo].[AspNetUserLogins] WITH CHECK ADD CONSTRAINT [FK\_dbo.AspNetUserLogins\_dbo.AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[AspNetUserLogins] CHECK CONSTRAINT [FK\_dbo.AspNetUserLogins\_dbo.AspNetUsers\_UserId]

GO

ALTER TABLE [dbo].[AspNetUserRoles] WITH CHECK ADD CONSTRAINT [FK\_dbo.AspNetUserRoles\_dbo.AspNetRoles\_RoleId] FOREIGN KEY([RoleId])

REFERENCES [dbo].[AspNetRoles] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[AspNetUserRoles] CHECK CONSTRAINT [FK\_dbo.AspNetUserRoles\_dbo.AspNetRoles\_RoleId]

GO

ALTER TABLE [dbo].[AspNetUserRoles] WITH CHECK ADD CONSTRAINT [FK\_dbo.AspNetUserRoles\_dbo.AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[AspNetUserRoles] CHECK CONSTRAINT [FK\_dbo.AspNetUserRoles\_dbo.AspNetUsers\_UserId]

GO

ALTER TABLE [dbo].[Immovable] WITH CHECK ADD CONSTRAINT [FK\_Immovable\_CadastrId] FOREIGN KEY([CadastrId])

REFERENCES [dbo].[Cadastr] ([CadastrId])

GO

ALTER TABLE [dbo].[Immovable] CHECK CONSTRAINT [FK\_Immovable\_CadastrId]

GO

ALTER TABLE [dbo].[Immovable] WITH CHECK ADD CONSTRAINT [FK\_Immovable\_ImmovableTypeId] FOREIGN KEY([ImmovableTypeId])

REFERENCES [dbo].[ImmovableType] ([ImmovableTypeId])

GO

ALTER TABLE [dbo].[Immovable] CHECK CONSTRAINT [FK\_Immovable\_ImmovableTypeId]

GO

ALTER TABLE [dbo].[Immovable] WITH CHECK ADD CONSTRAINT [FK\_Immovable\_OwnerId] FOREIGN KEY([OwnerId])

REFERENCES [dbo].[Owner] ([OwnerId])

GO

ALTER TABLE [dbo].[Immovable] CHECK CONSTRAINT [FK\_Immovable\_OwnerId]

GO

ALTER TABLE [dbo].[Land] WITH CHECK ADD CONSTRAINT [FK\_Land\_CadastrId] FOREIGN KEY([CadastrId])

REFERENCES [dbo].[Cadastr] ([CadastrId])

GO

ALTER TABLE [dbo].[Land] CHECK CONSTRAINT [FK\_Land\_CadastrId]

GO

ALTER TABLE [dbo].[Land] WITH CHECK ADD CONSTRAINT [FK\_Land\_LandTypeId] FOREIGN KEY([LandTypeId])

REFERENCES [dbo].[LandType] ([LandTypeId])

GO

ALTER TABLE [dbo].[Land] CHECK CONSTRAINT [FK\_Land\_LandTypeId]

GO

ALTER TABLE [dbo].[Land] WITH CHECK ADD CONSTRAINT [FK\_Land\_OwnerId] FOREIGN KEY([OwnerId])

REFERENCES [dbo].[Owner] ([OwnerId])

GO

ALTER TABLE [dbo].[Land] CHECK CONSTRAINT [FK\_Land\_OwnerId]

GO

ALTER TABLE [dbo].[LicenseRequest] WITH CHECK ADD CONSTRAINT [FK\_LicenseRequest\_LandId] FOREIGN KEY([LandId])

REFERENCES [dbo].[Land] ([LandId])

GO

ALTER TABLE [dbo].[LicenseRequest] CHECK CONSTRAINT [FK\_LicenseRequest\_LandId]

GO

ALTER TABLE [dbo].[Owner] WITH CHECK ADD CONSTRAINT [FK\_Owner\_AspNetUsers] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

GO

ALTER TABLE [dbo].[Owner] CHECK CONSTRAINT [FK\_Owner\_AspNetUsers]

GO

USE [master]

GO

ALTER DATABASE [CadastraDB] SET READ\_WRITE

GO

Приложение б. Исходный код программной системы

**CadastrDAO.cs**

public class CadastrDAO

{

private CadastraDBEntities \_edmx = new CadastraDBEntities();

public CadastrDAO()

{

}

public async Task<List<CadastrViewModel>> GetCadastras() =>

await (from cad in \_edmx.Cadastrs

where cad.CadastrId > 0

select new CadastrViewModel

{

CadastrId = cad.CadastrId,

CadastrName = cad.Name

})

.ToListAsync();

public async Task<CadastrViewModel> GetCadastrById(int cadId) =>

await (from cad in \_edmx.Cadastrs

where cad.CadastrId == cadId

select new CadastrViewModel

{

CadastrId = cad.CadastrId,

CadastrName = cad.Name

})

.FirstOrDefaultAsync();

public async Task EditCadastr(CadastrViewModel model)

{

var entity = await \_edmx.Cadastrs.FirstOrDefaultAsync(x => x.CadastrId == model.CadastrId);

if (entity == null)

throw new Exception("Не найдена модель для редактирования");

entity.Name = model.CadastrName;

await \_edmx.SaveChangesAsync();

}

public async Task RemoveCadastr(CadastrViewModel model)

{

var entity = await \_edmx.Cadastrs.FirstOrDefaultAsync(x => x.CadastrId == model.CadastrId);

if (entity == null)

throw new Exception("Не найдена модель для редактирования");

\_edmx.Cadastrs.Remove(entity);

await \_edmx.SaveChangesAsync();

}

public async Task CreateCadastr(CadastrViewModel model)

{

Cadastr entity = new Cadastr

{

Name = model.CadastrName

};

\_edmx.Cadastrs.Add(entity);

await \_edmx.SaveChangesAsync();

}

}

**ImmovableDAO.cs**

public class ImmovableDAO

{

private CadastraDBEntities \_edm = new CadastraDBEntities();

public async Task<IEnumerable<ImmovableViewModel>> GetImmovables() =>

await (from im in \_edm.Immovables

where im.ImmovableId > 0

select new ImmovableViewModel

{

ImmovableId = im.ImmovableId,

Address = im.Address,

Cost = im.Cost,

ImmovableType = new ImmovableTypeViewModel

{

ImmovableTypeId = im.ImmovableTypeId,

ImmovableTypeName = \_edm.ImmovableTypes.FirstOrDefault(x => x.ImmovableTypeId == im.ImmovableTypeId).ImmovableName

},

Onwer = new OwnerViewModel

{

OwnerId = im.OwnerId,

Name = \_edm.Owners.FirstOrDefault(x => x.OwnerId == im.OwnerId).Name,

Surname = \_edm.Owners.FirstOrDefault(x => x.OwnerId == im.OwnerId).Surname

},

Cadastr = new CadastrViewModel

{

CadastrId = im.CadastrId,

CadastrName = im.Cadastr.Name

}

})

.ToListAsync();

public async Task<ImmovableViewModel> GetImmovableById(int id) =>

await (from im in \_edm.Immovables

where im.ImmovableId == id

select new ImmovableViewModel

{

ImmovableId = im.ImmovableId,

Address = im.Address,

Cost = im.Cost,

ImmovableType = new ImmovableTypeViewModel

{

ImmovableTypeId = im.ImmovableTypeId,

ImmovableTypeName = \_edm.ImmovableTypes.FirstOrDefault(x => x.ImmovableTypeId == im.ImmovableTypeId).ImmovableName

},

Onwer = new OwnerViewModel

{

OwnerId = im.OwnerId,

Name = \_edm.Owners.FirstOrDefault(x => x.OwnerId == im.OwnerId).Name,

Surname = \_edm.Owners.FirstOrDefault(x => x.OwnerId == im.OwnerId).Surname

},

Cadastr = new CadastrViewModel

{

CadastrId = im.CadastrId,

CadastrName = im.Cadastr.Name

}

})

.FirstOrDefaultAsync();

public async Task EditImmovable(ImmovableViewModel model)

{

var entity = await \_edm.Immovables.FirstOrDefaultAsync(x => x.ImmovableId == model.ImmovableId);

if (entity == null)

throw new Exception("Не найдена сущность для редактирования");

entity.Address = model.Address;

entity.Cost = model.Cost;

entity.ImmovableTypeId = model.ImmovableType.ImmovableTypeId;

entity.OwnerId = model.Onwer.OwnerId;

entity.CadastrId = model.Cadastr.CadastrId;

await \_edm.SaveChangesAsync();

}

public async Task RemoveImmovable(int id)

{

var entity = await \_edm.Immovables.FirstOrDefaultAsync(x => x.ImmovableId == id);

if (entity == null)

throw new Exception("Не найдена сущность для удаления");

\_edm.Immovables.Remove(entity);

await \_edm.SaveChangesAsync();

}

public async Task CreateImmovable(ImmovableViewModel model)

{

if (model == null)

throw new Exception("Модель для добавления нового недвижимого имущества пуста!");

var immovable = new Immovable

{

Address = model.Address,

Cost = model.Cost,

ImmovableTypeId = model.ImmovableType.ImmovableTypeId,

OwnerId = model.Onwer.OwnerId,

CadastrId = model.Cadastr.CadastrId

};

\_edm.Immovables.Add(immovable);

await \_edm.SaveChangesAsync();

}

}

**LandDAO.cs**

public class LandDAO

{

private CadastraDBEntities \_edmx = new CadastraDBEntities();

public LandDAO()

{

}

public async Task<List<LandViewModel>> GetLands() =>

await (from land in \_edmx.Lands

where land.CadastrId == 2

select new LandViewModel

{

Cadastr = new CadastrViewModel

{

CadastrId = land.CadastrId,

CadastrName = land.Cadastr.Name

},

LandId = land.LandId,

Address = land.Address,

Area = land.Area,

Cost = land.Cost,

LandType = new LandTypeViewModel

{

LandTypeId = land.LandTypeId,

LandTypeName = land.LandType.Name.ToString()

},

Owner = new OwnerViewModel

{

OwnerId = land.OwnerId,

Name = land.Owner.Name,

Surname = land.Owner.Surname,

BirthDate = land.Owner.DateBirth

}

})

.ToListAsync();

public async Task<LandViewModel> GetLandById(int landId) =>

await (from land in \_edmx.Lands

where land.LandId == landId

select new LandViewModel

{

Cadastr = new CadastrViewModel

{

CadastrId = land.CadastrId,

CadastrName = land.Cadastr.Name

},

LandId = land.LandId,

Address = land.Address,

Area = land.Area,

Cost = land.Cost,

LandType = new LandTypeViewModel

{

LandTypeId = land.LandTypeId,

LandTypeName = land.LandType.Name.ToString()

},

Owner = new OwnerViewModel

{

OwnerId = land.OwnerId,

Name = land.Owner.Name,

Surname = land.Owner.Surname,

BirthDate = land.Owner.DateBirth

}

})

.FirstOrDefaultAsync();

public async Task EditLand(LandViewModel model)

{

var entity = await (from land in \_edmx.Lands

where land.LandId == model.LandId

select land).FirstOrDefaultAsync();

if (entity == null)

throw new Exception("Модель для редактирования пустая");

entity.LandTypeId = model.LandType.LandTypeId > 0 ? model.LandType.LandTypeId : entity.LandTypeId;

entity.OwnerId = model.Owner.OwnerId > 0 ? model.Owner.OwnerId : entity.OwnerId;

entity.Address = model.Address;

entity.Area = model.Area;

entity.Cost = model.Cost;

entity.CadastrId = model.Cadastr.CadastrId > 0 ? model.Cadastr.CadastrId : entity.CadastrId;

if(entity.CadastrId == 2)

{

var license = new LicenseRequest

{

LandId = entity.LandId,

LicenseReqState = "Not confirmed"

};

\_edmx.LicenseRequests.Add(license);

}

await \_edmx.SaveChangesAsync();

}

public async Task CreateLand(LandViewModel model)

{

Land land = new Land

{

Address = model.Address,

Area = model.Area,

Cost = model.Cost,

LandTypeId = model.LandType.LandTypeId,

OwnerId = model.Owner.OwnerId,

CadastrId = model.Cadastr.CadastrId

};

\_edmx.Lands.Add(land);

await \_edmx.SaveChangesAsync();

var license = new LicenseRequest

{

LandId = land.LandId,

LicenseReqState = "Not Accepted"

};

\_edmx.LicenseRequests.Add(license);

await \_edmx.SaveChangesAsync();

}

public async Task RemoveLand(LandViewModel model)

{

var licensee = await \_edmx.LicenseRequests.FirstOrDefaultAsync(x => x.LandId == model.LandId);

var entity = await (from land in \_edmx.Lands

where land.LandId == model.LandId

select land).FirstOrDefaultAsync();

if (entity == null)

throw new Exception("Модель для удаления пустая!");

if(licensee == null)

throw new Exception("Модель для удаления пустая!");

\_edmx.LicenseRequests.Remove(licensee);

\_edmx.Lands.Remove(entity);

await \_edmx.SaveChangesAsync();

}

}

**LicenseDAO.cs**

public class LicenseDAO

{

private CadastraDBEntities \_edm = new CadastraDBEntities();

public LicenseDAO()

{

}

public IEnumerable<LicenseRequestModel> GetAllRequests()

{

var requests = (from license in \_edm.LicenseRequests

join land in \_edm.Lands on license.LandId equals land.LandId

join owner in \_edm.Owners on land.OwnerId equals owner.OwnerId

select new LicenseRequestModel

{

LicenseId = license.LicenseId,

LiicenseRequestState = license.LicenseReqState,

Cadastr = new CadastrViewModel

{

CadastrId = land.CadastrId,

CadastrName = land.Cadastr.Name

},

Land = new LandViewModel

{

LandId = land.LandId,

Address = land.Address,

Area = land.Area,

Cost = land.Cost,

LandType = new LandTypeViewModel

{

LandTypeId = land.LandTypeId,

LandTypeName = land.LandType.Name

}

},

Owner = new OwnerViewModel

{

OwnerId = owner.OwnerId,

Name = owner.Name,

Surname = owner.Surname,

BirthDate = owner.DateBirth

}

})

.ToList();

return requests;

}

public LicenseRequestModel GetLicenseById(int id)

{

var request = (from license in \_edm.LicenseRequests

join land in \_edm.Lands on license.LandId equals land.LandId

join owner in \_edm.Owners on land.OwnerId equals owner.OwnerId

where license.LicenseId == id

select new LicenseRequestModel

{

LicenseId = license.LicenseId,

LiicenseRequestState = license.LicenseReqState,

Cadastr = new CadastrViewModel

{

CadastrId = land.CadastrId,

CadastrName = land.Cadastr.Name

},

Land = new LandViewModel

{

LandId = land.LandId,

Address = land.Address,

Area = land.Area,

Cost = land.Cost,

LandType = new LandTypeViewModel

{

LandTypeId = land.LandTypeId,

LandTypeName = land.LandType.Name

}

},

Owner = new OwnerViewModel

{

OwnerId = owner.OwnerId,

Name = owner.Name,

Surname = owner.Surname,

BirthDate = owner.DateBirth

}

})

.FirstOrDefault();

return request;

}

public void EditLicenseData(LicenseRequestModel model)

{

var license = \_edm.LicenseRequests.FirstOrDefault(x => x.LicenseId == model.LicenseId);

if (license == null)

throw new Exception("Что-то пошло не так");

license.Land.LandId = model.Land.LandId;

license.Land.LandTypeId = model.Land.LandType.LandTypeId;

license.Land.Owner.OwnerId = model.Owner.OwnerId;

license.Land.Area = model.Land.Area;

license.Land.Cost = model.Land.Cost;

license.Land.Address = model.Land.Address;

\_edm.SaveChanges();

}

public void Accept(int id)

{

var licEntity = \_edm.LicenseRequests.FirstOrDefault(x => x.LicenseId == id);

licEntity.LicenseReqState = "Accepted";

\_edm.SaveChanges();

}

public void Reject(int id)

{

var licEntity = \_edm.LicenseRequests.FirstOrDefault(x => x.LicenseId == id);

licEntity.LicenseReqState = "Rejected";

\_edm.SaveChanges();

}

public void NotAccepted(int id)

{

var licEntity = \_edm.LicenseRequests.FirstOrDefault(x => x.LicenseId == id);

licEntity.LicenseReqState = "Not Accepted";

\_edm.SaveChanges();

}

public void SentToRevision(int id)

{

var licEntity = \_edm.LicenseRequests.FirstOrDefault(x => x.LicenseId == id);

licEntity.LicenseReqState = "Sened for revision";

\_edm.SaveChanges();

}

}

**OwnerDAO.cs**

public class OwnerDAO

{

private CadastraDBEntities \_edm = new CadastraDBEntities();

public OwnerDAO()

{

}

public async Task<List<OwnerViewModel>> GetOwners() =>

await (from owner in \_edm.Owners

where owner.OwnerId > 0

select new OwnerViewModel

{

OwnerId = owner.OwnerId,

Name = owner.Name,

Surname = owner.Surname,

BirthDate = owner.DateBirth

})

.ToListAsync();

public async Task<OwnerViewModel> GetOwnerById(int ownerId) =>

await (from owner in \_edm.Owners

where owner.OwnerId == ownerId

select new OwnerViewModel

{

OwnerId = owner.OwnerId,

Name = owner.Name,

Surname = owner.Surname,

BirthDate = owner.DateBirth

})

.FirstOrDefaultAsync();

public async Task EditOwner(OwnerViewModel model)

{

var entity = await \_edm.Owners.FirstOrDefaultAsync(x => x.OwnerId == model.OwnerId);

if (entity == null)

throw new Exception("Модель для редактирования пустая!");

entity.Name = model.Name;

entity.Surname = model.Surname;

entity.DateBirth = model.BirthDate;

await \_edm.SaveChangesAsync();

}

public async Task RemoveOwner(OwnerViewModel model)

{

var entity = await \_edm.Owners.FirstOrDefaultAsync(x => x.OwnerId == model.OwnerId);

if (entity == null)

throw new Exception("Модель для редактирования пустая!");

\_edm.Owners.Remove(entity);

await \_edm.SaveChangesAsync();

}

public async Task CreateOwner(OwnerViewModel model)

{

Owner owner = new Owner

{

Name = model.Name,

Surname = model.Surname,

DateBirth = model.BirthDate,

UserId = model.UserId

};

\_edm.Owners.Add(owner);

await \_edm.SaveChangesAsync();

}

}

**CadastrController.cs**

[Authorize]

public class CadastrController : Controller

{

CadastrDAO \_cad = new CadastrDAO();

// GET: Cadastr

public async Task<ActionResult> Index()

{

var cadastras = await \_cad.GetCadastras();

return View(cadastras);

}

[Authorize(Roles = "Administration")]

public async Task<ActionResult> Edit(int id)

{

var cadastr = await \_cad.GetCadastrById(id);

return View(cadastr);

}

[HttpPost]

[Authorize(Roles = "Administration")]

public async Task<ActionResult> Edit(CadastrViewModel model)

{

await \_cad.EditCadastr(model);

return RedirectToAction("Index");

}

}

**ImmovableController.cs**

[Authorize]

public class ImmovableController : Controller

{

ImmovableDAO \_im = new ImmovableDAO();

private CadastraDBEntities \_edmx = new CadastraDBEntities();

// GET: Immovable

public async Task<ActionResult> Index(string searchString)

{

var immovables = (await \_im.GetImmovables()).ToList();

List<ImmovableViewModel> result = new List<ImmovableViewModel>();

result = immovables;

if (!string.IsNullOrEmpty(searchString))

{

//попробуем найти по адерсу

result = immovables.Where(x => x.Address.Contains(searchString)).ToList();

//если не получилось найти по адресу

if (!result.Any())

//попробуем найти по имени

result = immovables.Where(x => x.Onwer.Name.Contains(searchString)).ToList();

//если не получилось найти по имени

if (!result.Any())

//попробуем найти по фамилии

result = immovables.Where(x => x.Onwer.Surname.Contains(searchString)).ToList();

//если не получилось найти по фамилии

if (!result.Any())

//попробуем найти по имени и фамилии

result = immovables.Where(x => x.Onwer.Owner.Contains(searchString)).ToList();

decimal costOrArea = 0;

//а может входная переменная число?!

bool val = decimal.TryParse(searchString, out costOrArea);

//если да

if (val)

{

//попробуем найти по цене

result = immovables.Where(x => x.Cost <= costOrArea).ToList();

}

}

return View(result);

}

[HttpGet]

[Authorize]

public ActionResult Create()

{

InitDynamicViewBag();

var user = \_edmx.AspNetUsers.FirstOrDefault(x => x.UserName == User.Identity.Name);

var owner = \_edmx.Owners.FirstOrDefault(x => x.UserId == user.Id);

var immovable = new ImmovableViewModel

{

Name = owner.Name,

Surname = owner.Surname,

CurrentUserId = user.Id,

Cadastr = new CadastrViewModel

{

CadastrId = 2,

CadastrName = "Immovable"

}

};

return View(immovable);

}

[HttpPost]

[Authorize]

public async Task<ActionResult> Create(ImmovableViewModel model)

{

if (model.Onwer == null)

{

var user = \_edmx.AspNetUsers.FirstOrDefault(x => x.UserName == User.Identity.Name);

var owner = \_edmx.Owners.FirstOrDefault(x => x.UserId == user.Id);

model.Onwer = new OwnerViewModel();

model.Onwer.OwnerId = owner.OwnerId;

model.Cadastr = new CadastrViewModel();

model.Cadastr.CadastrId = 1;

}

if (model != null)

{

await \_im.CreateImmovable(model);

}

return RedirectToAction("Index");

}

[HttpGet]

[Authorize]

public async Task<ActionResult> Edit(int id)

{

InitDynamicViewBag();

var im = await \_im.GetImmovableById(id);

return View(im);

}

private void InitDynamicViewBag()

{

var immovables = new SelectList(\_edmx.ImmovableTypes.ToList(), "ImmovableTypeId", "ImmovableName");

var owns = \_edmx.Owners.Select(x => new OwnerViewModel

{

OwnerId = x.OwnerId,

Name = x.Name,

Surname = x.Surname

});

var owners = new SelectList(owns.ToList(), "OwnerId", "Owner");

var cadastras = new SelectList(\_edmx.Cadastrs.ToList(), "CadastrId", "Name");

ViewBag.ImmovableTypes = immovables;

ViewBag.Owners = owners;

ViewBag.Cadastras = cadastras;

}

[HttpPost]

[Authorize]

public async Task<ActionResult> Edit(ImmovableViewModel model)

{

if (model.Onwer == null)

{

var user = \_edmx.AspNetUsers.FirstOrDefault(x => x.UserName == User.Identity.Name);

var owner = \_edmx.Owners.FirstOrDefault(x => x.UserId == user.Id);

model.Onwer = new OwnerViewModel();

model.Onwer.OwnerId = owner.OwnerId;

model.Cadastr = new CadastrViewModel();

model.Cadastr.CadastrId = 1;

}

if (model != null)

{

await \_im.EditImmovable(model);

}

return RedirectToAction("Index");

}

[HttpGet]

[Authorize(Roles = "Administration,Moderator")]

public async Task<ActionResult> Delete(int id)

{

var im = await \_im.GetImmovableById(id);

return View(im);

}

[HttpPost]

[Authorize(Roles = "Administration,Moderator")]

public async Task<ActionResult> Delete(ImmovableViewModel model)

{

try

{

if (ModelState.IsValid && model != null)

await \_im.RemoveImmovable(model.ImmovableId);

else

throw new Exception();

}

catch (Exception ex)

{

ModelState.AddModelError("", ex);

}

return RedirectToAction("Index");

}

}

**LandController.cs**

public class LandController : Controller

{

private CadastraDBEntities \_edmx = new CadastraDBEntities();

private LandDAO \_land = new LandDAO();

// GET: Land

public async Task<ActionResult> Index(string searchString)

{

var lands = await \_land.GetLands();

List<LandViewModel> result = new List<LandViewModel>();

result = lands;

//если переменная searchString не пустая

if (!string.IsNullOrEmpty(searchString))

{

//попробуем найти по адерсу

result = lands.Where(x => x.Address.Contains(searchString)).ToList();

//если не получилось найти по адресу

if (!result.Any())

//попробуем найти по имени

result = lands.Where(x => x.Owner.Name.Contains(searchString)).ToList();

//если не получилось найти по имени

if (!result.Any())

//попробуем найти по фамилии

result = lands.Where(x => x.Owner.Surname.Contains(searchString)).ToList();

//если не получилось найти по фамилии

if (!result.Any())

//попробуем найти по имени и фамилии

result = lands.Where(x => x.Owner.Owner.Contains(searchString)).ToList();

decimal costOrArea = 0;

//а может входная переменная число?!

bool val = decimal.TryParse(searchString, out costOrArea);

//если да

if (val)

{

//попробуем найти по цене

result = lands.Where(x => x.Cost <= costOrArea).ToList();

//если не получилось

if (!result.Any())

//попробуем найти по площади

result = lands.Where(x => x.Area <= costOrArea).ToList();

}

}

return View(result);

}

[HttpGet]

[Authorize]

public ActionResult CreateLand()

{

InitDynamicViewBag();

var user = \_edmx.AspNetUsers.FirstOrDefault(x => x.UserName == User.Identity.Name);

var owner = \_edmx.Owners.FirstOrDefault(x => x.UserId == user.Id);

var land = new LandViewModel

{

Name = owner.Name,

Surname = owner.Surname,

CurrentUserId = user.Id,

Cadastr = new CadastrViewModel

{

CadastrId = 2,

CadastrName = "Land"

}

};

return View(land);

}

private void InitDynamicViewBag()

{

var landTypes = new SelectList(\_edmx.LandTypes.ToList(), "LandTypeId", "Name");

var owns = \_edmx.Owners.Select(x => new OwnerViewModel

{

OwnerId = x.OwnerId,

Name = x.Name,

Surname = x.Surname

});

var owners = new SelectList(owns.ToList(), "OwnerId", "Owner");

var cadastras = new SelectList(\_edmx.Cadastrs.ToList(), "CadastrId", "Name");

ViewBag.LandTypes = landTypes;

ViewBag.Owners = owners;

ViewBag.Cadastras = cadastras;

}

[HttpPost]

[Authorize]

public async Task<ActionResult> CreateLand(LandViewModel model)

{

try

{

if (model.Owner == null)

{

var user = \_edmx.AspNetUsers.FirstOrDefault(x => x.UserName == User.Identity.Name);

var owner = \_edmx.Owners.FirstOrDefault(x => x.UserId == user.Id);

model.Owner = new OwnerViewModel();

model.Owner.OwnerId = owner.OwnerId;

model.Cadastr = new CadastrViewModel();

model.Cadastr.CadastrId = 2;

}

if (model != null)

await \_land.CreateLand(model);

else

throw new Exception();

}

catch (Exception ex)

{

ModelState.AddModelError("", ex);

}

return RedirectToAction("Index");

}

[HttpGet]

[Authorize]

public async Task<ActionResult> EditLand(int id)

{

InitDynamicViewBag();

LandViewModel land = new LandViewModel();

try

{

land = await \_land.GetLandById(id);

}

catch (Exception ex)

{

ModelState.AddModelError("", ex);

}

return View(land);

}

[HttpPost]

[Authorize]

public async Task<ActionResult> EditLand(LandViewModel model)

{

try

{

if (model.Owner == null)

{

var user = \_edmx.AspNetUsers.FirstOrDefault(x => x.UserName == User.Identity.Name);

var owner = \_edmx.Owners.FirstOrDefault(x => x.UserId == user.Id);

model.Owner = new OwnerViewModel();

model.Owner.OwnerId = owner.OwnerId;

model.Cadastr = new CadastrViewModel();

model.Cadastr.CadastrId = 2;

}

if (model != null)

await \_land.EditLand(model);

else

throw new Exception();

}

catch (Exception ex)

{

ModelState.AddModelError("", ex);

}

return RedirectToAction("Index");

}

[HttpGet]

[Authorize(Roles = "Administration,Moderator")]

public async Task<ActionResult> Delete(int id)

{

var land = await \_land.GetLandById(id);

return View(land);

}

[HttpPost]

[Authorize(Roles = "Administration,Moderator")]

public async Task<ActionResult> Delete(LandViewModel model)

{

try

{

if (model != null)

await \_land.RemoveLand(model);

else

throw new Exception();

}

catch (Exception ex)

{

ModelState.AddModelError("", ex);

}

return RedirectToAction("Index");

}

}

**LicenseController.cs**

[Authorize]

public class LicenseController : Controller

{

private CadastraDBEntities \_edmx = new CadastraDBEntities();

LicenseDAO \_license = new LicenseDAO();

public ActionResult Index()

{

var licenses = \_license.GetAllRequests();

return View(licenses);

}

[HttpGet]

[Authorize]

public ActionResult EditLicense(int id)

{

var landTypes = new SelectList(\_edmx.LandTypes.ToList(), "LandTypeId", "Name");

var owns = \_edmx.Owners.Select(x => new OwnerViewModel

{

OwnerId = x.OwnerId,

Name = x.Name,

Surname = x.Surname

});

var owners = new SelectList(owns.ToList(), "OwnerId", "Owner");

var cadastras = new SelectList(\_edmx.Cadastrs.ToList(), "CadastrId", "Name");

ViewBag.LandTypes = landTypes;

ViewBag.Owners = owners;

ViewBag.Cadastras = cadastras;

var license = \_license.GetLicenseById(id);

return View(license);

}

[Authorize]

public ActionResult EditLicense(LicenseRequestModel model)

{

if (ModelState.IsValid && model != null)

{

\_license.EditLicenseData(model);

}

return RedirectToAction("Index");

}

[Authorize(Roles = "Administration, Moderator")]

public ActionResult Accept(int id)

{

\_license.Accept(id);

return RedirectToAction("Index");

}

[Authorize(Roles = "Administration, Moderator")]

public ActionResult Reject(int id)

{

\_license.Reject(id);

return RedirectToAction("Index");

}

[Authorize(Roles = "Administration, Moderator")]

public ActionResult SentToRevision(int id)

{

\_license.SentToRevision(id);

return RedirectToAction("Index");

}

}

**OwnerController.cs**

[Authorize]

public class OwnerController : Controller

{

OwnerDAO \_owner = new OwnerDAO();

[Authorize]

public async Task<ActionResult> Index()

{

var owners = await \_owner.GetOwners();

return View(owners);

}

[HttpGet]

[Authorize(Roles = "Administration,Moderator")]

public ActionResult CreateOwner()

{

return View();

}

[HttpPost]

[Authorize(Roles = "Administration,Moderator")]

public async Task<ActionResult> CreateOwner(OwnerViewModel model)

{

if (ModelState.IsValid && model != null)

{

await \_owner.CreateOwner(model);

}

return RedirectToAction("Index");

}

[HttpGet]

[Authorize(Roles = "Administration,Moderator")]

public async Task<ActionResult> EditOwnerData(int id)

{

var owner = await \_owner.GetOwnerById(id);

return View(owner);

}

[HttpPost]

[Authorize(Roles = "Administration,Moderator")]

public async Task<ActionResult> EditOwnerData(OwnerViewModel model)

{

if (ModelState.IsValid && model != null)

{

await \_owner.EditOwner(model);

}

return RedirectToAction("Index");

}

[HttpGet]

[Authorize(Roles = "Administration,Moderator")]

public async Task<ActionResult> Delete(int id)

{

var owner = await \_owner.GetOwnerById(id);

return View(owner);

}

[HttpPost]

[Authorize(Roles = "Administration,Moderator")]

public async Task<ActionResult> Delete(OwnerViewModel model)

{

if (ModelState.IsValid && model != null)

{

await \_owner.RemoveOwner(model);

}

return RedirectToAction("Index");

}

}