Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

Кафедра информационных систем и программной инженерии

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

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

**на тему**

Проектирование и разработка программной системы   
«Кадастровое управление»

Выполнил: ст. гр. ПРИ-116

Мартынов Г. И. Принял: доц. кафедры ИСПИ

Вершинин В.В.

Владимир, 2019

**Аннотация**

В данной курсовой работе разрабатывается программная система «Кадастровое управление». С целью удобства использования были разработаны модели данных, база данных SQL, контроллеры и представления. В базе данных имеется 13 таблиц, заполненных данными. Разработана логическая схема базы данных, в соответствие с теорией нормализации, приведена к 3НФ. А также разработана диаграмма прецедентов и состояний.

Курсовой проект представлен на 33 страницах, и содержит 31 рисунок, 1 таблица, использованных источников – 9, приложений – 1, графический материал на 3 листах формата А4 масштаба 1:8.

**ANNOTATION**

In this term paper is developed software system "Cadastral management". For ease of use, data models, databases SQL, controllers and views have been developed. The database has 13 tables filled with data. The logical scheme of the database has been developed, in accordance with the theory of normalization, reduced to 3NF. And also developed a chart of precedents and states.

The course project is presented on 33 pages, and contains 31 figures, 1 table, used sources - 9, application - 1, graphic material on 3 sheets of A4 format in scale 1:8.

**Содержание**

ВВЕДЕНИЕ 3

1 ПОСТАНОВКА ЦЕЛИ И ЗАДАЧ 5

1.1 Цель 5

1.2. Задачи, решаемые подсистемой хранения данных 5

1.3 Сокращения 6

2 Описание предметной области 7

2.1 Общие сведения о предмете автоматизации 7

2.2 Словарь предметной области 7

2.3 Сценарий взаимодействия пользователя с системой 8

3 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 10

3.1 Диаграмма прецедентов 10

3.2 Диаграмма классов 13

3.3 Диаграмма объектов 14

3.4 Диаграмма состояний 14

3.5 Диаграмма последовательности 16

3.6 Структура базы данных 18

4 РЕАЛИЗАЦИЯ СИСТЕМЫ 20

4.1 Принципы организации проекта 20

4.2 Реализация модели 20

4.3 Реализация поддержки различных типов пользователей 20

4.4 Реализация подачи заявки 21

4.5 Реализация рассмотрения заявок 21

4.6 Реализация создания данных о КО 21

4.7 Пользовательский интерфейс 22

заключение 32

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33

ПРИЛОЖЕНИЕ А Исходный код программной системы 34

# ВВЕДЕНИЕ

Заданием данной курсовой работы является проектирование и реализация программной системы «Кадастровое управление». Данная система нужна для автоматизации повторяющихся процессов, которые может сделать программа и избежание ошибок человека.

Существенное преимущество построения веб-приложений заключается в том, что стандартные функции браузера должны выполняться независимо от операционной системы данного клиента. Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент-сервер». Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него. Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP.

Данная система позволяет пользователю подавать заявки на постановку кадастровых объектов на учет, оплатив ее, просматривать статус выполнения. А инженеру составлять документацию и создавать хранимые объекты учета.

В ходе выполнения курсового проекта, требуется решить следующие задачи:

* анализ предметной области;
* разработка проекта программной системы;
* разработка прототипа программной системы.

Результатом выполнения курсового проекта будет работающее программное обеспечение, позволяющее вести учет заявок и кадастровых объектов.

В ходе выполнения курсового проекта выполнена разработка проекта информационной системы с использованием языка UML, реализация диаграмм с использованием среды Microsoft Office Visio, реализация программного обеспечения с использованием технологий разработки ASP.NET MVC, Code First, Entity Framework, Razor, Bootstrap. Среда программирования – Visual Studio 2015.

**1 ПОСТАНОВКА ЦЕЛИ И ЗАДАЧ**

**1.1 Цель**

1.1.1 Целью создания программной системы «Кадастровое управление» является упрощение работы инженера и заказчиков, обеспечение целостности и сохранности информации, ускорение обработки и упрощение учета регулярно используемых данных.

**1.2 Задачи, решаемые подсистемой хранения данных**

1.2.1 Для достижения поставленной цели необходимо последовательно выполнить этапы анализа, проектирования и реализации системы.

1.2.2 Анализ предметной области

Результатом данного этапа работы является описание бизнес-процесса учета заявок, постановка их на учет и словарь предметной области.

1.2.3 Проектирование системы

Для проектирования системы необходимо использовать язык UML. Проектирование должно включать в себя следующие UML-диаграммы:

• диаграмма прецедентов;

• диаграмма классов;

• диаграмма состояний;

• диаграмма последовательностей.

Также этап проектирования включает схему базы данных в виде ER-диаграммы логической моделей данных.

1.2.4 Программная реализация.

Минимальный набор функций:

– ввод данных в систему о кадастровых объектах (земельный участок, объект недвижимости и т.д.)

– работа со справочником типов объектов учета в Кадастровом управлении;

– подача заявки на регистрацию кадастрового объекта;

– самостоятельная регистрация заявителя в системе;

– просмотр сведений о кадастровых объектах и изменение сведений о них;

– рассмотрение заявок и регистрация (или отказ).

Система должна быть реализована на языке C# в виде веб-приложения с использованием технологий ASP.NET MVC.

**1.****3 Сокращения**

− Кадастровое управление: КУ;

− Кадастровый объект: КО;

− Кадастровая работа: КР;

− Справочник типов объектов учета: СТОУ.

**2 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

**2.1 Общие сведения о предмете автоматизации**

2.1.1 Кадастровой деятельностью являются выполнение работ в отношении недвижимого имущества в соответствии с установленными федеральным законом требованиями, в результате которых обеспечивается подготовка документов, содержащих необходимые для осуществления государственного кадастрового учета недвижимого имущества (далее - кадастровый учет) сведения о таком недвижимом имуществе (далее - кадастровые работы), и оказание услуг в установленных федеральным законом случаях. Специальным правом на осуществление кадастровой деятельности обладает лицо, указанное в статье 29 настоящего Федерального закона (далее - кадастровый инженер). Федеральный закон о кадастровой деятельности №221-ФЗ от 24.07.2007 в редакции Федерального закона №361-ФЗ от 03.07.2016.

**2.2** **Словарь предметной области**

**2.2.1 Сущность «Пользователь (User)»**

2.2.1.1 Авторизованный пользователь системы, который имеет логин, пароль, личные данные, ему доступны определенные функции системы.

**2.2.2 Сущность «Заказчик (Customer)»**

2.2.2.1 Пользователь системы, наследует свое поведение от сущности пользователь.

**2.2.3 Сущность «Заявка (Application)»**

2.2.3.1 Взаимодействует с заказчиком, отражает его цели.

**2.2.4 Сущность «****Вид кадастровой работы (Type CW)»**

2.2.4.1 Задается вид: межевой план, технический план, акт обследования.

**2.2.5 Сущность «Кадастровый объект (Cadastral object)»**

2.2.5.1 Главная информационная база, с которой работает информационная система. Ее редакцией занимается администратор.

**2.2.6 Сущность «Статус (Status)»**

2.2.6.1 Определяет этапы выполнения заявки и кадастровой работы (черновик, ожидание оплаты, обработка, отменена, выполняется, завершена)

**2.2.7 Сущность «Инженер (Engineer)»**

2.2.7.1 Пользователь системы, основной задачей которого является выполнение кадастровых работ.

**2.2.8 Сущность «Кадастровая работа (Cadastral work)»**

2.2.8.1 Основная работа, которую выполняет инженер.

**2.2.9 Сущность «Справочник типов КО (Handbook of CO types)»**

2.2.9.1 Отражает список типов КО (земельные участки; здания, сооружения, помещения, квартиры).

**2.2.10 Сущность «Администратор (Administrator)»**

2.2.10.1 Пользователь системы, обладающий полным доступом к системе, имеет право вносить, редактировать и удалять любую информацию в системе.

**2.3** **Сценарий взаимодействия пользователя с системой**

2.3.1 В приложении используется разделение по ролям. На основе анализа системы были разработаны 3 роли:

• Заказчик;

• Инженеры;

• Администратор.

**2.3.1.1 Гость:**

− Зарегистрироваться;

− Войти в систему;

− Посмотреть главную страницу сайта.

**2.3.1.2 Пользователь с ролью Заказчик может:**

− Посмотреть сведения о заявках;

− Подать заявку;

− Редактировать заявку;

− Оплатить заявку;

− Отменить заявку;

− Посмотреть сведения о КО и о их типах.

**2.3.1.3 Пользователь с ролью Инженер может:**

− Обработать заявку (одобрить/отклонить)

− Создать КР;

− Редактировать КР;

− Удалить КР;

− Создать данные о КО;

− Изменить данные о КО;

− Удалить данные о КО;

− Работать со справочником типов объектов учета (СТОУ) в КУ.

**2.3.1.4 Администратор наследует инженера, а также может:**

− Сопровождать сайт.

**3 проектирование системы**

**3.1** **Диаграмма прецедентов**

Объект информатизации исследуется в границах одного города. Действующими лицами (в нотации UML – актерами) являются: Заказчик, Инженер, Администратор и Гость. Диаграмма прецедентов представлена на рисунке 1.



Рисунок 1 – Диаграмма прецедентов

Ниже приведен перечень и расширенное описание некоторых прецедентов в проектируемой системе.

**3.1.1 Название прецедента «Подать заявку»**

Предусловие: заказчик нажимает кнопку "Создать новую заявку".

Действующее лицо: заказчик

Основной поток: открывается окно приложения, отображающее форму заявки. Заказчик заполняет все нужные поля и нажимает кнопку "Отправить".

Постусловие: данные поданной заявки система сохраняет в БД и перенаправляет его инженеру в прецедент "Обработать заявку".

**3.1.2 Название прецедента «Обработать заявку»**

Предусловие: инженер нажимает кнопку меню "Заявки".

Действующее лицо: инженер

Основной поток: открывается окно приложения, отображающее список заявок. Инженер нажимает на кнопку "Принять" заявку и переходит в окно создания новой кадастровой работы. После заполняет все нужные поля и нажимает кнопку "Создать". Статус заявки изменяется на "Выполнение".

Постусловие: данные принятой заявки система сохраняет в БД и отображает в форме "Список кадастровых работ".

Альтернативный поток: открывается окно приложения, отображающее список заявок. Инженер нажимает на кнопку "Отклонить" заявку и переходит в окно подтверждения отмены. После подтверждения статус заявки изменяется на "Отменена".

**3.1.3 Название прецедента «Создать данные о КО»**

Предусловие: инженер нажимает "Завершить" КР на странице списка КР или кнопку "Создать новый КО" на странице списка КО.

Действующее лицо: инженер

Основной поток: открывается окно приложения, отображающее форму ввода данных о КО. Инженер вводит тип КО, кадастровый номер и другие данные в соответствующие поля. Нажимает кнопку "Сохранить".

Постусловие: данные о КО система сохраняет в БД, которые после можно посмотреть в прецеденте "Просмотреть сведения о КО".

**3.1.4 Название прецедента «Изменить данные о КО»**

Предусловие: администратор на кнопку меню "Кадастровые объекты".

Действующее лицо: администратор или инженер

Основной поток: Администратор пользуется средствами поиска по каталогу (тип КО, кадастровый номер) нажимает кнопку "Изменить" у нужной записи. Открывается окно приложения, отображающее форму изменения данных о КО. Изменяет необходимые данные в соотв. полях. Нажимает кнопку "Сохранить".

Постусловие: данные о КО система сохраняет в БД, которые после можно посмотреть в прецеденте "Просмотреть сведения о КО".

**3.1.5 Название прецедента «Удалить данные о КО»**

Предусловие: администратор на кнопку меню "Кадастровые объекты".

Действующее лицо: администратор или инженер

Основной поток: Администратор пользуется средствами поиска по каталогу (тип КО, кадастровый номер) нажимает кнопку "Удалить" у нужной записи. Открывается диалоговое окно подтверждения удаления, нажимает на кнопку "Подтвердить" или "Отменить".

Постусловие: данные о КО система сохраняет в БД, которые после можно посмотреть в прецеденте "Просмотреть сведения о КО".

**3.2 Диаграмма классов**

3.2.1 Для проектирования структур и моделей данных программной системы была разработана диаграмма классов, представляющая основные сущности системы. Диаграмма классов представлена на рисунке 2.

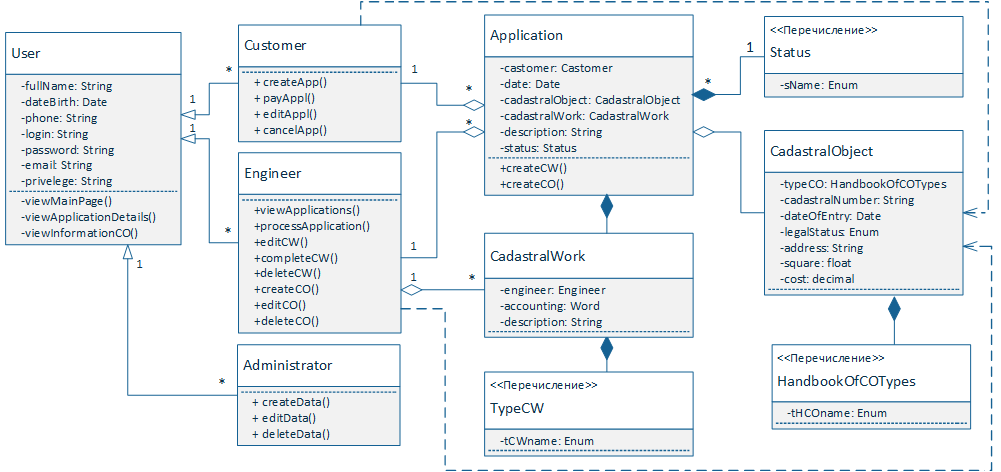


Рисунок 2 – Диаграмма классов

**3.2.2 Описание взаимоотношений между классами**.

Взаимоотношения между классами представлено в таблице 1.

Таблица 1 − Взаимоотношения между классами

|  |  |  |  |
| --- | --- | --- | --- |
| № | Взаимодействующий класс | Наследование | Классы, с которыми взаимодействуют |
| 1 | Пользователь | − | Главные вкладки сайта |
| 2 | Заказчик | Пользователь | Заявка (редактирование),  КО, типы КО |
| 3 | Инженер | Пользователь | КР, КО, типы КО – ред. |
| 4 | Администратор | Инженер | КР, типы КО, вид КР, типы КО |
| 5 | Заявка | − | вид КР, Статус, КО. |

**3.3 Диаграмма объектов**

3.3.1 Диаграмма объектов, полученная на основе диаграммы классов для объекта «Заявка» представлена на рисунке 3.

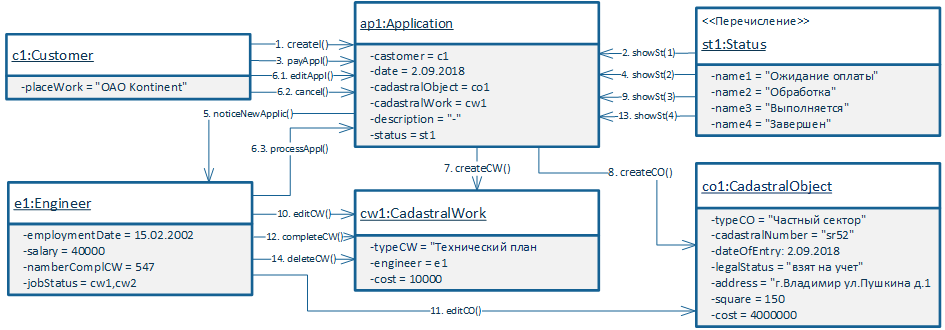


Рисунок 3 – Диаграмма объектов для объекта «Заявка»

**3.4 Диаграмма состояний**

3.4.1 Одним из важнейших свойств данной системы является возможность создавать и принимать заявки. Описание бизнес-процесса работы ИС с заявкой представлено ниже:

1. Пользователь (заказчик) оформил заявку, указав свои реквизиты, вид КО, КР и оплатил его. Объект «Заявка» создан в системе (объект в памяти и/или запись в БД. Данные, указанные заказчиком при оформлении заявки, передается во внешнюю систему проверки; заказ ожидает подтверждения о проверке.

2. От системы проверки пришло сообщение о неверном вводе нескольких полей. У заказчика на экране появляется соответствующее уведомление, он обязан исправить эти поля.

3. От системы проверки пришло подтверждение корректности ввода данных. Поскольку оплата заявки производится через внешнюю платежную систему, заявка не обрабатывается до момента получения подтверждения об оплате.

4. От платежной системы пришло сообщение об отказе в оплате или истек максимальный срок ожидания оплаты. Заказчику на указанный им e-mail отправляется соответствующее уведомление, после чего переходим к п. 9.

5. От платежной системы пришло подтверждение об оплате. Объект «Заявка» помечается как оплаченный. Заявка помечается как достоверная и переходит в состояние «Обработка заявки».

6. Инженер отклонил заявку. Переходим к п. 9.

7. Инженер подтвердил заявку, заявка переходит в состояние «Выполнение».

8. Инженер завершил КР и заказчику отправляется уведомление о выполнении.

9. Заявка помечается как отмененная.

3.4.2 Диаграмма состояний, полученная на основе диаграммы классов для объекта «Заявка» представлена на рисунке 4.

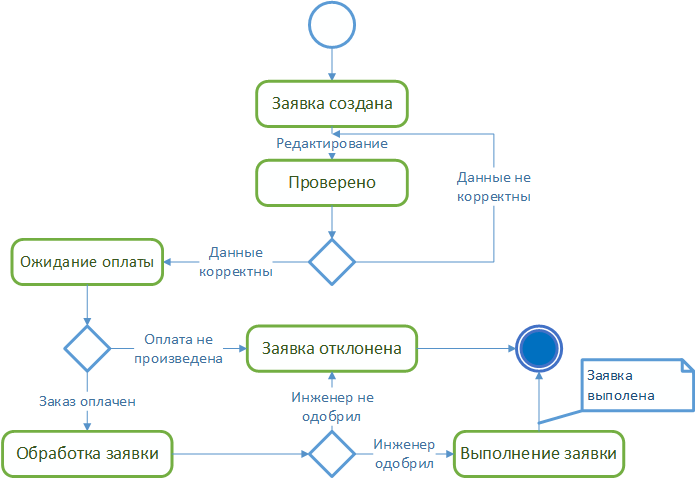


Рисунок 4 – Диаграмма состояний для объекта «Заявка»

**3.5 Диаграмма последовательностей**

3.5.1 Диаграмма последовательностей для объекта «Заявка» с действующим актером «Заказчик», полученная на основе диаграммы прецедентов представлена на рисунке 5.

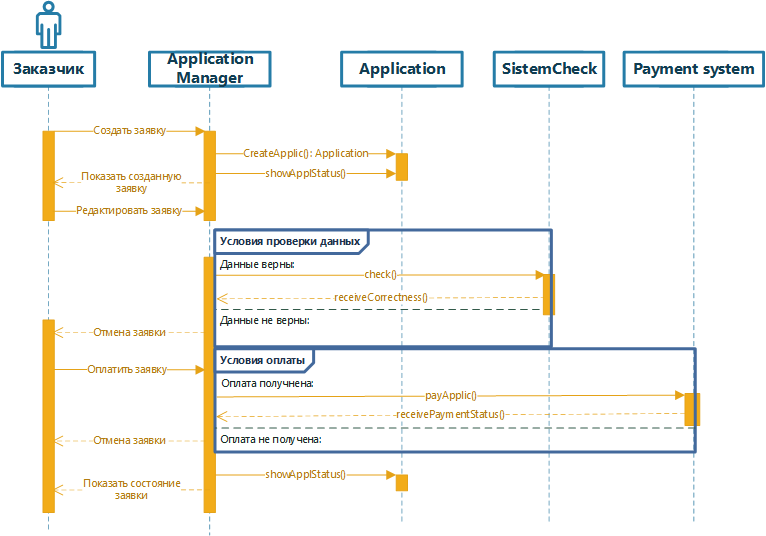


Рисунок 5 – Диаграмма последовательностей для объекта «Заявка» с актером «Заказчик»

3.5.1 Диаграмма последовательностей для объекта «Заявка» с действующим актером «Инженер», полученная на основе диаграммы прецедентов представлена на рисунке 6.

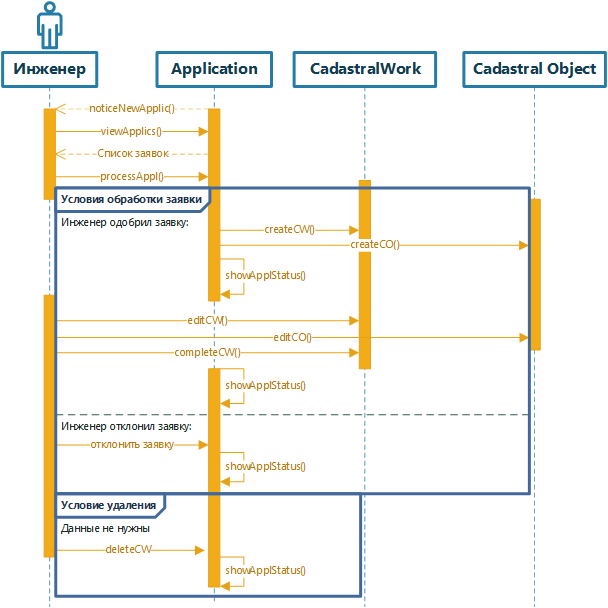


Рисунок 6 – Диаграмма последовательностей «Заявка» с актером «Инженер»

**3.6** **Структура базы данных**

3.6.1 Логическая ER диаграмма предметной области и все выделенные ранее сущности, атрибуты и связи, разработанная в среде Visual Studio 2015 и нормализованная к третьей нормальной форме, показана на рисунке 7.

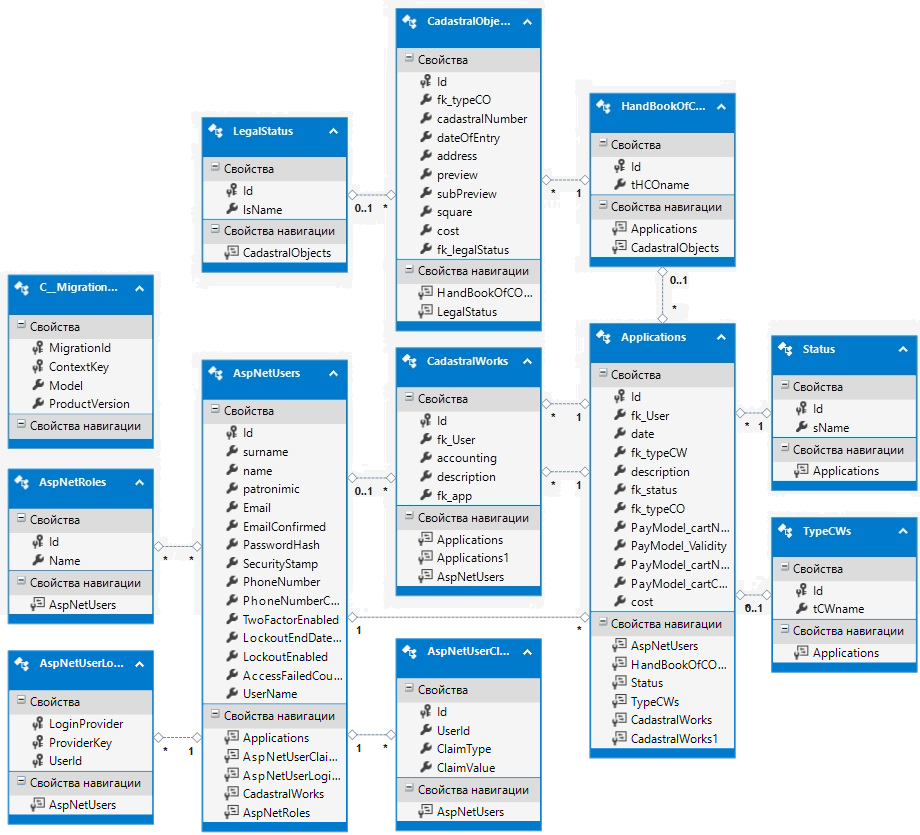


Рисунок 7 – Логическая схема модели базы данных

**4 РЕАЛИЗАЦИЯ СИСТЕМЫ**

**4.1 Принципы организации проекта**

Разрабатываемая система построена на архитектуре MVC с использованием технологии Entity Framework и языка интегрированных запросов LINQ. Система представляет собой веб-приложение, и, таким образом, разделяется на три части: контроллер, модель, представление.

Представления построены с использованием Razor – интеллектуальный обработчик программного кода и движок представлений в ASP.NET используемый для создания динамических веб-страниц с языком программирования C#. Также используется Bootstrap – это CSS фреймворк c хорошим набором инструментов для front-end разработки. Исходный код приложения приведен в приложении А.

**4.2 Реализация модели**

Модель построена с использованием Code First – подход, где главным становится код, а не база. То есть сначала создавались модели, а потом по ним с помощью добавления миграций строилась база данных. Работа с миграциями позволила изменять модели (добавлять новые, изменять существующие) без потери данных. Также данный метод позволяет не работать напрямую со скриптами создания и изменения таблиц.

**4.3 Реализация поддержки различных типов пользователей**

Для реализации регистрации, аутентификации и авторизации пользователей были использованы стандартные методы Microsoft Identity.

Методы и представления имеют ограниченный доступ. Пользователям, в зависимости от роли, доступны различные функции. Это обеспечивается ограничениями, реализованными с помощью встроенных средств.

Регистрация пользователей в системе производится программно.

**4.4** **Реализация подачи заявки**

Заказчик нажимает на кнопку меню «Заявки» и попадает на страницу, которая отображает данные о заявках, которые он уже подал. С помощью кнопки «Создать новую заявку» заказчик может пользуясь полями ввода и выпадающими списками выбора составить новую заявку и отправить на обработку. После воспользоваться опциями «Оплатить», «Изменить», «Отменить» или «Удалить» заявку.

**4.5** **Реализация рассмотрения заявок**

Инженер нажимает на кнопку меню «Заявки» и попадает на страницу, которая отображает список заявок от всех пользователей. Инженеру доступны две кнопки: «Принять» и «Отклонить». Нажав на кнопку "Принять" заявку, он переходит в окно создания новой кадастровой работы. Заполнив все нужные поля, нажимает кнопку "Создать". Статус заявки изменяется на "Выполнение".

Данные принятой заявки система сохраняет в БД и отображает в форме "Список кадастровых работ". Нажав на кнопку "Отклонить" заявку, система переходит в окно подтверждения отмены. После подтверждения статус заявки изменяется на "Отменена".

**4.6 Реализация создания данных о КО**

Инженер нажимает на кнопку "Завершить" КР на странице списка КР или кнопку "Создать новый КО" на странице списка КО. Открывается окно приложения, отображающее форму ввода данных о КО. Инженер вводит тип КО, кадастровый номер и другие данные в соответствующие поля. Нажимает кнопку "Сохранить". Данные о КО система сохраняет в БД, система переходит на страницу "Список КО" и отображает обновленную таблицу добавив данные нового объекта.

**4.7 Пользовательский интерфейс**

Пользовательский интерфейс представлен на рисунках 8-27. Формы изменения и удаления имеют схожий друг с другом интерфейс, поэтому некоторые страницы не были добавлены в примеры интерфейса.

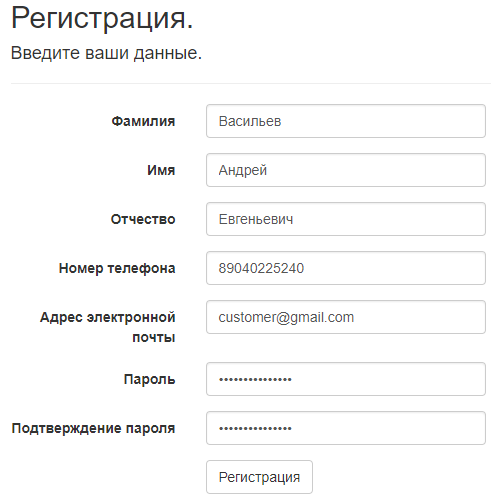


Рисунок 8 – Страница регистрации

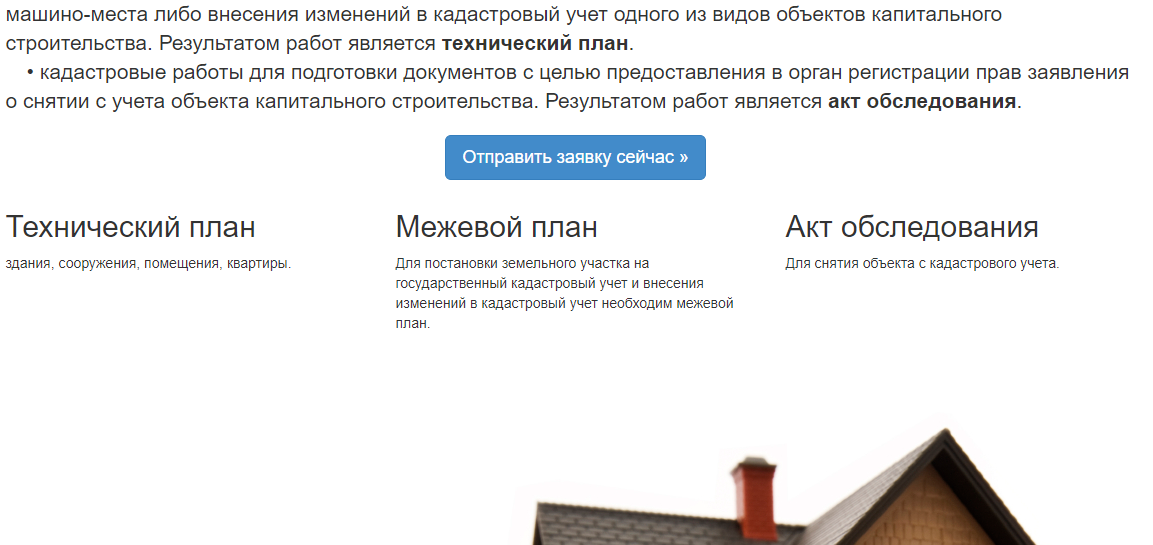
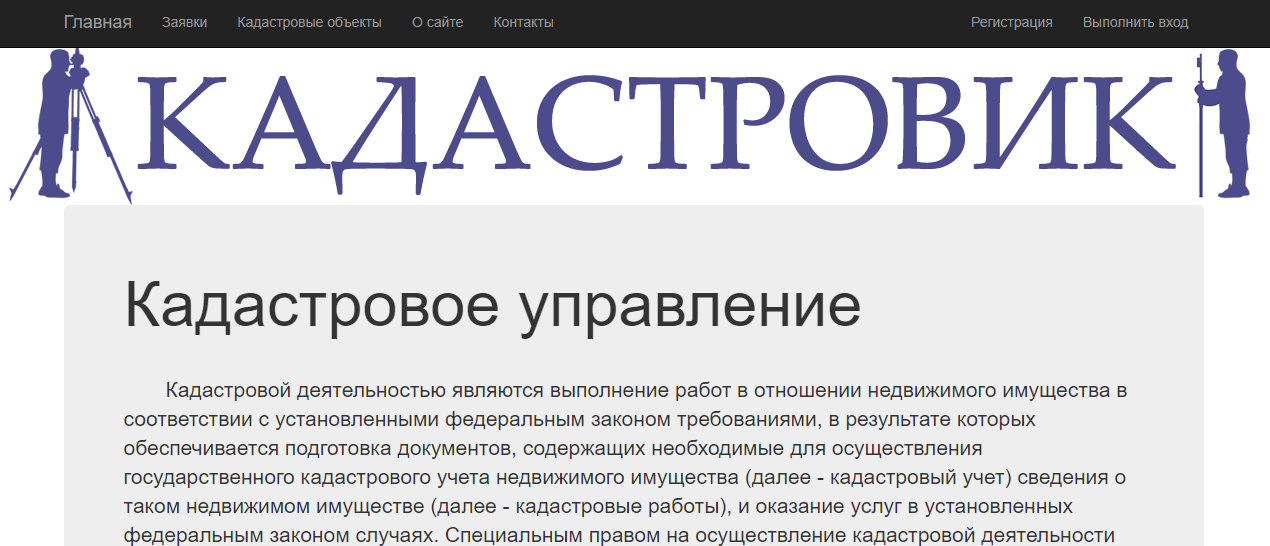


Рисунок 9 – Главная

При нажатии гостя на кнопку меню «Заявки» или кнопку «Отправить заявку сейчас» система перейдет в окно «Входа в систему», рисунок 10.

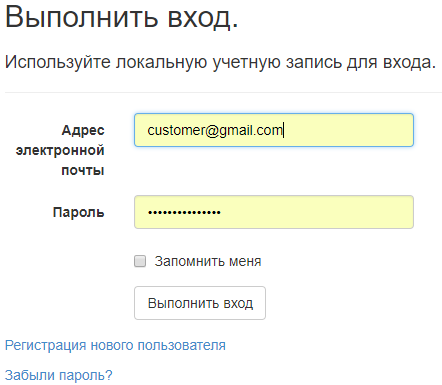


Рисунок 10 − Страница входа

Далее система перейдет в окно списка заявок, рисунок 11.

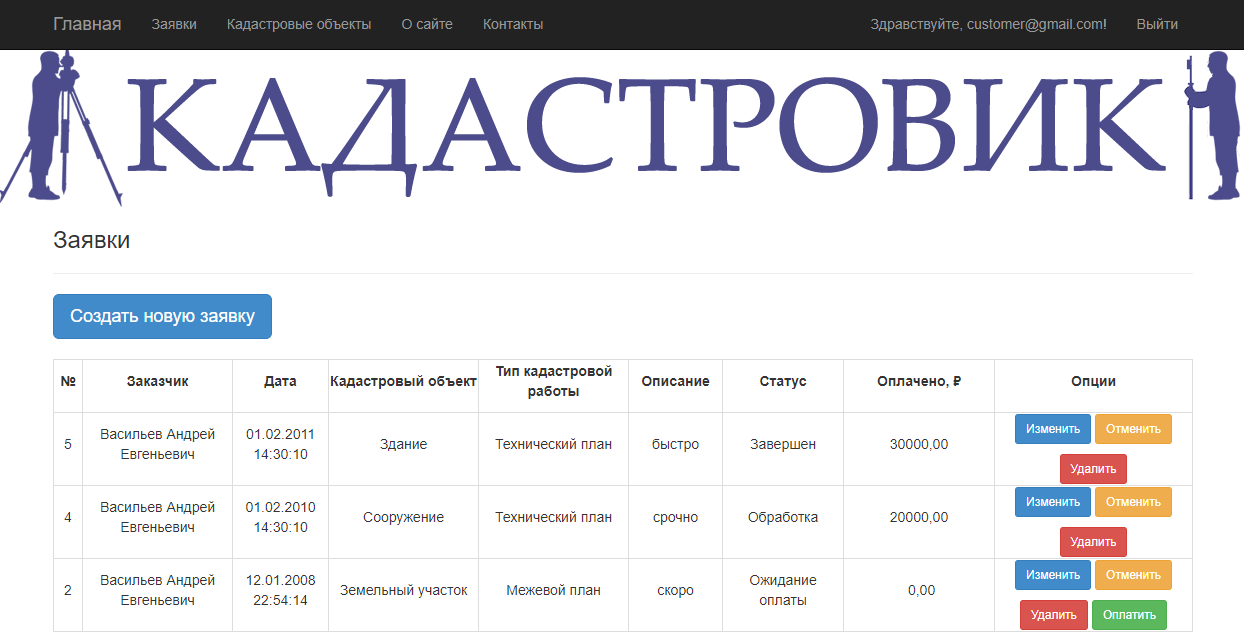


Рисунок 11 – Список заявок определенного пользователя

Нажмем на кнопку «Создать новую заявку», рисунок 12. Поля заказчик и дата формируются автоматически и не изменяемы, а поля КО и тип КР с возможностью выбора из выпадающего списка.

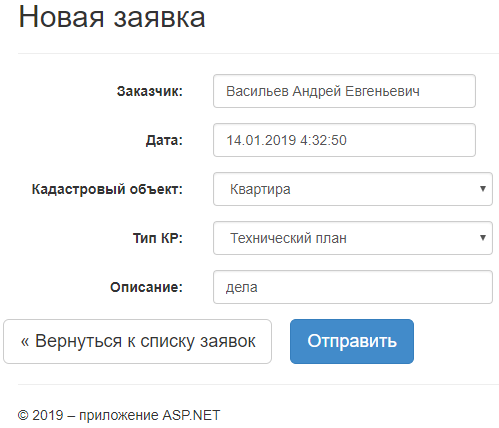


Рисунок 12 – Формирование новой заявки

После нажатия на кнопку «Отправить» в первой строке таблицы появляется новая запись, рисунок 13. Статус заявки «Ожидание опалаты».

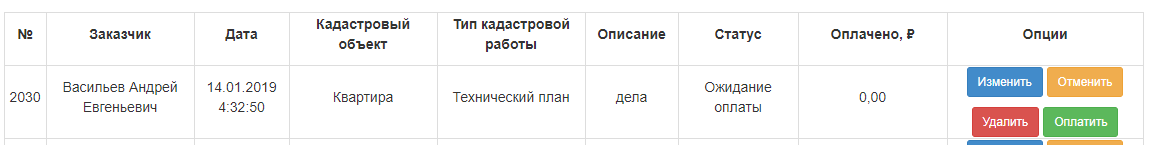


Рисунок 13 – Новая запись в таблице

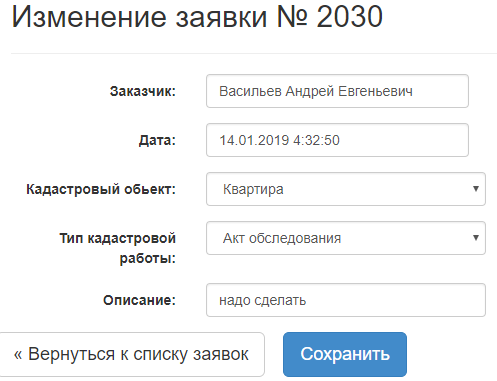


Рисунок 14 – Изменение заявки

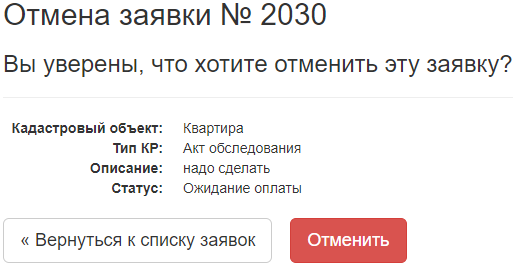


Рисунок 15 – Отмена заявки, по виду аналогична удалению

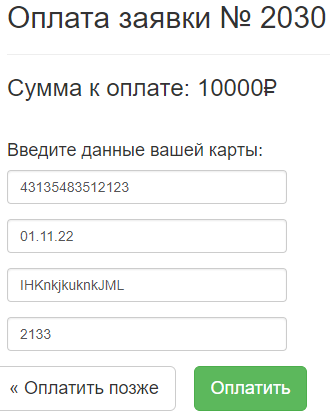


Рисунок 16 – Оплата заявки

После оплаты статус заявки изменится на «Обработка» и исчезнет кнопка «Оплатить», рисунок 17.

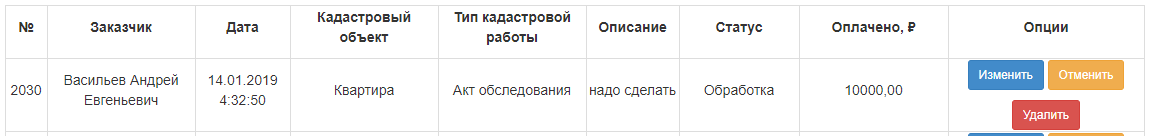


Рисунок 17 – Оплачено

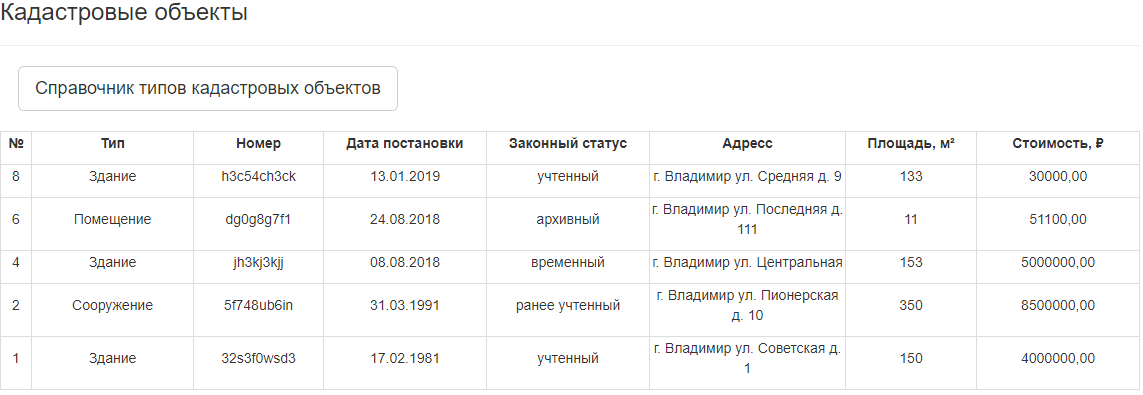


Рисунок 18 – Просмотр сведений о КО, ролью «Заказчик»

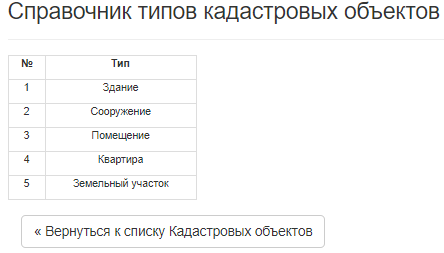


Рисунок 19 – Справочник типов КО

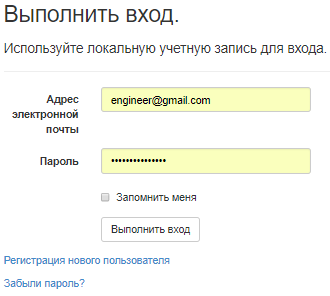


Рисунок 20 – Вход под ролью инженера

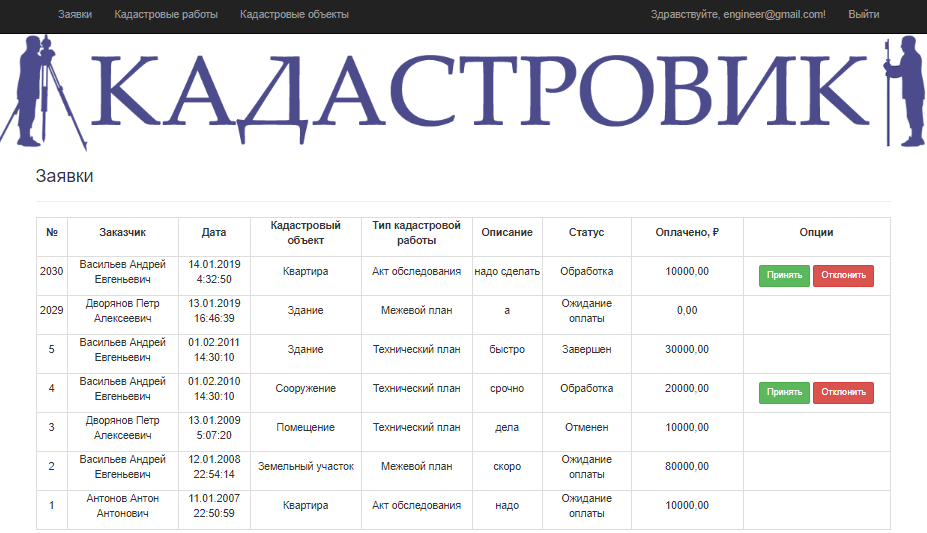


Рисунок 21 – Список всех заявок, отображаемых инженеру и админу

Инженеру доступны лишь 3 кнопки меню, это «Заявки», «Кадастровые работы» и «Кадастровые объекты».

Нажимаем на кнопку «Принять» у последней записи. Система переходит в окно создания новой КР, все поля заполняются автоматически, рисунок 22.

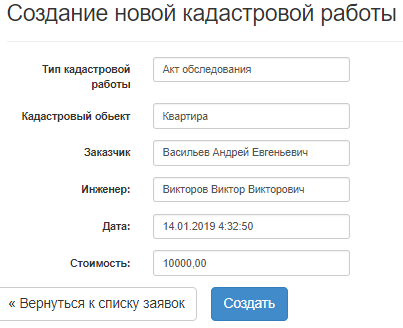


Рисунок 22 – Создание новой КР

Нажав на кнопку «Создать», система сохраняет КР в БД и переходит в окно списка КР отображая и так же новую КР, рисунок 23.

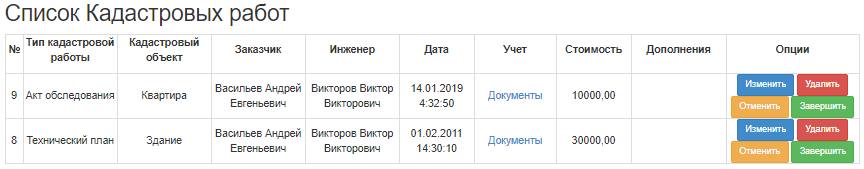


Рисунок 23 – Список «Кадастровых работ»

Статус заявки изменится на «Выполняется» и кнопки пропадут, рисунок 24.

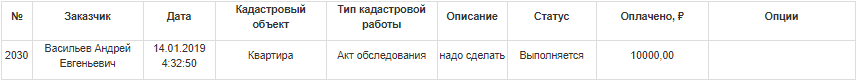


Рисунок 24 – Изменение статуса заявки на «Выполняется»

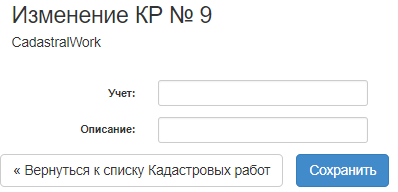


Рисунок 25 – Изменение КР

Нажмем на кнопку «Завершить» в окне списка КР у последней записи. Система перенаправит инженера на страницу создания нового кадастрового объекта, рисунок 26. Поля «Тип объекта», «Дата постановки на учет» и «Стоимость» заполняется автоматически, а поле «Правовой статус» с возможностью выбора из выпадающего списка.

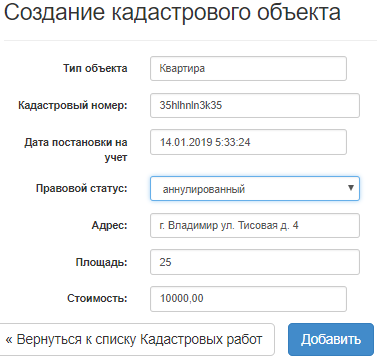


Рисунок 26 – Создание нового КО

Нажав кнопку добавить, система сохранит данные в БД и перенаправит инженера на страницу «списка Кадастровых объектов», рисунок 27.

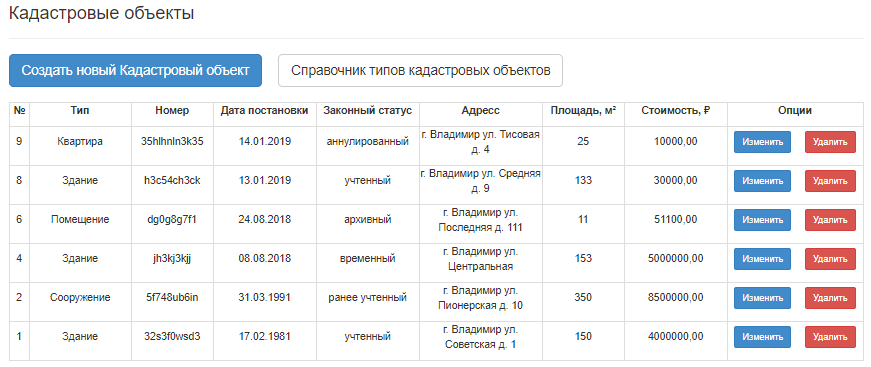


Рисунок 27 – Список Кадастровых объектов

Статус заявки изменится на «Завершен», рисунок 28.

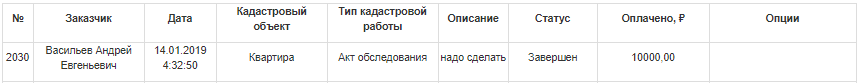


Рисунок 28 – Изменение статуса заявки на «Завершен»



Рисунок 29 – Справочник типов КО для инженера и админа

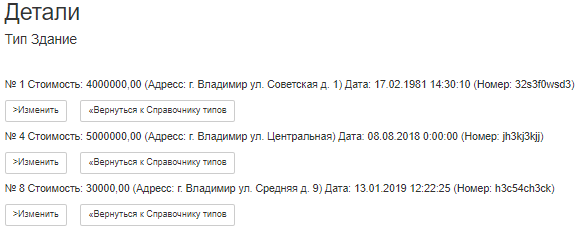


Рисунок 30 – Детали для типа «Здание»

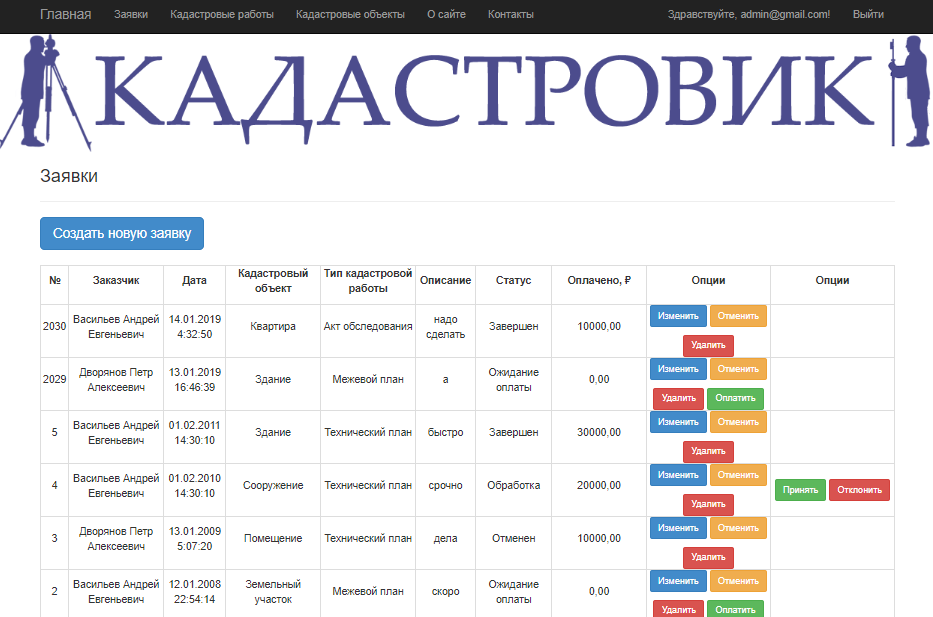


Рисунок 31 – Вход под Администратором в окно списка заявок

Администратор может видеть все кнопки и выполнять все действия предоставляемой системы.

# заключение

В результате выполнения работы была спроектирована и реализована программная система учета «Кадастровых объектов». Она во многом удобна на практике, так как позволяет хранить, добавлять, изменять, удалять, обеспечивать целостность и сохранность информации. Разработанный функционал позволяет просматривать различные данные и автоматизировать процесс подачи и обработки заявок, что упрощает работу инженера и заказчиков системы «Кадастрового управления». Позволяет ускорить обработку и упростить учет регулярно используемых данных. Для выполнения поставленной задачи был выполнен:

* Анализ предметной области и формирование требований к системе;
* Разработка проекта программной системы
* Реализация программного обеспечения

Разработка проекта программного продукта была выполнена на языке UML. Программное обеспечение было реализовано на языке C# в виде веб-приложения, построенного на архитектуре MVC на базе программной платформы ASP.NET.

Разработанная система позволяет:

• Самостоятельно зарегистрироваться заявителю в системе;

• Подавать заявки на регистрацию кадастрового объекта;

• Рассматривать заявки и регистрировать (или отклонять);

• Просматривать сведения о КО и изменять сведения о них;

• Вводить данные в систему о КО;

• Работать со справочником типов объектов учета в КУ;

Данная программная система может быть использована организациями занимающимися кадастровым делом с целью электронного представления данных учета заявок, упрощение работы инженера и заказчиков, обеспечение целостности и сохранности информации, ускорение обработки и упрощение учета регулярно используемых данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Информация о Кадастровом учете, видам работ и объектах: http://kadastrovik.com/kadastr.html
2. Использование Code First Migrations:

https://docs.microsoft.com/ru-ru/ef/ef6/modeling/code-first/migrations/automatic, https://metanit.com/sharp/entityframework/3.12.php

1. Справочник по синтаксису Razor для ASP.NET Core:

https://docs.microsoft.com/ru-ru/aspnet/core/mvc/views/razor?view=aspnetcore -2.2

1. Модели со сложной структурой: https://metanit.com/sharp/mvc/5.6.php
2. Решение проблем с разработкой системы http://qaru.site/questions/
3. Руководство по ASP.NET MVC 5: https://metanit.com/sharp/mvc5/
4. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов г.2013 стр.688
5. ASP.NET MVC 4. Разработка реальных веб-приложений с помощью

ASP.NET MVC. г. 2013 стр.432.

1. Pro jQuery Адам Фриман.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММНОЙ СИСТЕМЫ**

**Модели:**

**Application.cs**

namespace WebAppCM.Models

{

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

public partial class Application

{

[Key, Display(Name = "№:")]

public int Id { get; set; }

[ForeignKey(nameof(User)), Display(Name = "Заказчик:")]

public string fk\_User { get; set; }

[Display(Name = "Дата:")]

public System.DateTime date { get; set; }

[ForeignKey(nameof(HandBookOfCOType)),Display(Name = "Кадастровый объект:")]

public int? fk\_typeCO { get; set; }

[ForeignKey(nameof(TypeCW)), Display(Name = "Тип КР:")]

public int? fk\_typeCW { get; set; }

[Display(Name = "Описание:")]

public string description { get; set; }

[ForeignKey(nameof(Status)), Display(Name = "Статус:")]

public int fk\_status { get; set; }

[Display(Name = "Стоимость:")]

public decimal cost { get; set; }

[InverseProperty(nameof(CadastralWork.app))]

public virtual ICollection<CadastralWork> CadastralWorks { get; set; } = new HashSet<CadastralWork>();

public virtual HandBookOfCOType HandBookOfCOType { get; set; }

public virtual ApplicationUser User { get; set; }

public virtual Status Status { get; set; }

public virtual TypeCW TypeCW { get; set; }

public virtual PayModel PayModel { get; set; }

}

}

**CadastralObject.cs**

namespace WebAppCM.Models

{

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

public partial class CadastralObject

{

[Key, Display(Name = "№:")]

public int Id { get; set; }

[ForeignKey(nameof(HandBookOfCOType)), Display(Name = "Тип объекта")]

public int? fk\_typeCO { get; set; }

[Display(Name = "Кадастровый номер:")]

public string cadastralNumber { get; set; }

[Display(Name = "Дата внесения:")]

public System.DateTime dateOfEntry { get; set; }

[ForeignKey(nameof(LegalStatus)), Display(Name = "Правовой статус:")]

public int? fk\_legalStatus { get; set; }

[Display(Name = "Адрес:")]

public string address { get; set; }

[Display(Name = "Кадастровый номер:")]

public string preview { get; set; }

public string subPreview { get; set; }

[Display(Name = "Площадь:")]

public double square { get; set; }

[Display(Name = "Стоимость:")]

public decimal cost { get; set; }

public virtual HandBookOfCOType HandBookOfCOType { get; set; }

public virtual LegalStatus LegalStatus { get; set; }

}

}

**HandBookOfCOType.cs**

namespace WebAppCM.Models

{

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

public partial class HandBookOfCOType

{

[Key]

public int Id { get; set; }

public string tHCOname { get; set; }

[InverseProperty(nameof(CadastralObject.HandBookOfCOType))]

public virtual ICollection<CadastralObject> CadastralObjects

{ get; set; } = new HashSet<CadastralObject>();

[InverseProperty(nameof(Application.HandBookOfCOType))]

public virtual ICollection<Application> Applications { get; set; } = new HashSet<Application>();

public IEnumerable<CadastralObject> co;

public IEnumerable<Application> app;

}

}

**LegalStatus.cs**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace WebAppCM.Models

{

public class LegalStatus

{

[Key]

public int Id { get; set; }

public string lsName { get; set; }

[InverseProperty(nameof(CadastralObject.LegalStatus))]

public virtual ICollection<CadastralObject> CadastralObjects { get; set; } = new HashSet<CadastralObject>();

}

}

**Status.cs**

namespace WebAppCM.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

public partial class Status

{

[Key]

public int Id { get; set; }

public string sName { get; set; }

[InverseProperty(nameof(Application.Status))]

public virtual ICollection<Application> Applications { get; set; } = new HashSet<Application>();

}

}

**TypeCW.cs**

namespace WebAppCM.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

public partial class TypeCW

{

[Key]

public int Id { get; set; }

public string tCWname { get; set; }

[InverseProperty(nameof(Application.TypeCW))]

public virtual ICollection<Application> Applications { get; set; } = new HashSet<Application>();

public IEnumerable<Application> app;

public IEnumerable<CadastralWork> cw;

}

}

**PayModel.cs**

using System.ComponentModel.DataAnnotations;

namespace WebAppCM.Models

{

public class PayModel

{

[Display(Name = "Номер карты:")]

public string cartNumber { get; set; }

[Display(Name = "Срок действия:"), DisplayFormat(DataFormatString = "{0:dd.MM.yy}",

ApplyFormatInEditMode = true)]

public System.DateTime Validity { get; set; }

[Display(Name = "Имя карты:")]

public string cartName { get; set; }

[Display(Name = "Код безопасности:")]

public string cartCod { get; set; }

}

}

**IdentityModels.cs**

using System.Data.Entity;

using System.Security.Claims;

using System.Threading.Tasks;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.EntityFramework;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

namespace WebAppCM.Models

{

public class ApplicationUser : IdentityUser

{

public string surname { get; set; }

public string name { get; set; }

public string patronimic { get; set; }

public IEnumerable<Application> app;

public IEnumerable<CadastralWork> cw;

public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)

{

var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);

return userIdentity;

}

}

public class ApplicationDbContext : IdentityDbContext<ApplicationUser>

{

public ApplicationDbContext()

: base("DefaultConnection", throwIfV1Schema: false)

{}

public virtual DbSet<Application> Applications { get; set; }

public virtual DbSet<CadastralObject> CadastralObjects { get; set; }

public virtual DbSet<HandBookOfCOType> HandBookOfCOTypes { get; set; }

public virtual DbSet<CadastralWork> CadastralWorks { get; set; }

public virtual DbSet<Status> Status { get; set; }

public virtual DbSet<TypeCW> TypeCWs { get; set; }

public virtual DbSet<LegalStatus> LegalStatus { get; set; }

public static ApplicationDbContext Create()

{ return new ApplicationDbContext();

}

}

}

Изменения в **AccountViewModels.cs**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace WebAppCM.Models

{

public class LoginViewModel

{

[Required]

[Display(Name = "Адрес электронной почты")]

[EmailAddress]

public string Email { get; set; }

[Required]

[DataType(DataType.Password)]

[Display(Name = "Пароль")]

public string Password { get; set; }

[Display(Name = "Запомнить меня")]

public bool RememberMe { get; set; }

}

public class RegisterViewModel

{

[Required, Display(Name = "Фамилия")]

public string surname { get; set; }

[Required, Display(Name = "Имя")]

public string name { get; set; }

[Required, Display(Name = "Отчество")]

public string patronymic { get; set; }

[Required, Display(Name = "Номер телефона")]

public string PhoneNumber { get; set; }

[Required, EmailAddress, Display(Name = "Адрес электронной почты")]

public string Email { get; set; }

[Required]

[StringLength(100, ErrorMessage = "Значение {0} должно содержать не менее {2} символов.", MinimumLength = 6)]

[DataType(DataType.Password), Display(Name = "Пароль")]

public string Password { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Подтверждение пароля")]

[Compare("Password", ErrorMessage = "Пароль и его подтверждение не совпадают.")]

public string ConfirmPassword { get; set; }

}

}

**Контроллеры:**

**AppController.cs**

using System;

using System.Web;

using System.Web.Mvc;

using WebAppCM.Models;

using System.Data.Entity;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

namespace WebAppCM.Controllers.AppController

{

public class AppController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: App

[HttpGet, Authorize(Roles = "Admin, Engineer,Customer")]

public ActionResult ListApp()

{

var items = db.Applications.Include(p => p.HandBookOfCOType).

Include(p => p.User).Include(p => p.Status).Include(p => p.TypeCW);

ApplicationUser user = UserManager.FindByEmail(User.Identity.Name);

ViewBag.User = user;

return View(items);

}

private ApplicationUserManager UserManager

{ get

{ return HttpContext.GetOwinContext().

GetUserManager<ApplicationUserManager>();

}

}

// GET: App/Create

[HttpGet, Authorize(Roles = "Customer, Admin")]

public ActionResult AppCreate()

{

ApplicationUser user = UserManager.FindByEmail(User.Identity.Name);

ViewBag.User = user;

ViewBag.Date = DateTime.Now;

// Формируем список типов КО для передачи в представление

SelectList typeCO = new SelectList(db.HandBookOfCOTypes, "Id",

"tHCOname");

ViewBag.HandBookOfCOTypes = typeCO;

// Формируем список типов КР для передачи в представление

SelectList typeCW = new SelectList(db.TypeCWs, "Id", "tCWname");

ViewBag.TypeCW = typeCW;

ViewBag.Status = 2;

return View();

}

// POST: App/Create

[HttpPost]

public ActionResult AppCreate(Application App)

{

db.Applications.Add(App);

db.SaveChanges();

return RedirectToAction("ListApp");

}

// GET: App/Edit/5

[HttpGet]

public ActionResult AppEdit(int? id)

{

if (id == null)

return HttpNotFound();

// Находим в бд выбранную заявку

Application app = db.Applications.Find(id);

if (app != null)

{

// Создаем список типо КО для передачи в представление

SelectList typeCO = new SelectList(db.HandBookOfCOTypes, "Id",

"tHCOname", app.fk\_typeCO);

ViewBag.HandBookOfCOTypes = typeCO;

// Создаем список типо КР для передачи в представление

SelectList typeCW = new SelectList(db.TypeCWs, "Id", "tCWname",

app.fk\_typeCW);

ViewBag.TypeCW = typeCW;

return View(app);

}

return RedirectToAction("ListApp");

}

// POST: App/Edit/5

[HttpPost]

public ActionResult AppEdit(Application app)

{

db.Entry(app).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("ListApp");

}

// GET: App/Cancel/5

[HttpGet]

public ActionResult AppCancel(int id)

{

Application app = db.Applications.Find(id);

if (app == null)

return HttpNotFound();

return View(app);

}

// POST: App/Cancel/5

[HttpPost, ActionName("AppCancel")]

public ActionResult AppCancelConfirmed(int id)

{

Application app = db.Applications.Find(id);

if (app == null)

return HttpNotFound();

app.fk\_status=5;

db.Entry(app).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("ListApp");

}

// GET: App/Delete/5

[HttpGet]

public ActionResult AppDelete(int id)

{

Application app = db.Applications.Find(id);

if (app == null)

return HttpNotFound();

return View(app);

}

// POST: App/Delete/5

[HttpPost, ActionName("AppDelete")]

public ActionResult AppDeleteConfirmed(int id)

{

Application app = db.Applications.Find(id);

if (app == null)

return HttpNotFound();

db.Applications.Remove(app);

db.SaveChanges();

return RedirectToAction("ListApp");

}

// GET: App/AppPay/5

[HttpGet]

public ActionResult AppPay(int? id)

{

if (id == null)

return HttpNotFound();

// Находим в бд выбранную заявку

Application app = db.Applications.Find(id);

if (app != null)

return View(app);

return RedirectToAction("ListApp");

}

// POST: App/AppPay/5

[HttpPost]

public ActionResult AppPay(Application app)

{

db.Entry(app).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("ListApp");

}

}

}

**COController.cs**

using System.Data.Entity;

using System.Linq;

using System.Web.Mvc;

using WebAppCM.Models;

namespace WebAppCM.Controllers.COController

{

public class COController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: CO

[HttpGet, Authorize(Roles = "Admin, Engineer,Customer")]

public ActionResult ListCO()

{

var items = db.CadastralObjects.Include(p => p.HandBookOfCOType).

Include(p => p.LegalStatus);

return View(items.ToList());

}

// GET: CO/Create

[HttpGet]

public ActionResult COCreate()

{

// Формируем список типов КО для передачи в представление

SelectList type = new SelectList(db.HandBookOfCOTypes, "Id",

"tHCOname");

ViewBag.HandBookOfCOTypes = type;

// Формируем список статусов для передачи в представление

SelectList status = new SelectList(db.LegalStatus, "Id", "lsName");

ViewBag.LegalStatus = status;

return View();

}

// POST: CO/Create

[HttpPost]

public ActionResult COCreate(CadastralObject co)

{

//Добавляем КО в таблицу

db.CadastralObjects.Add(co);

db.SaveChanges();

return RedirectToAction("ListCO");

}

// GET: CO/Edit/5

[HttpGet]

public ActionResult COEdit(int? id)

{

if (id == null)

return HttpNotFound();

// Находим в бд выбранный KO

CadastralObject co = db.CadastralObjects.Find(id);

if (co != null)

{

// Создаем список типо КО для передачи в представление

SelectList type = new SelectList(db.HandBookOfCOTypes, "Id", "tHCOname", co.fk\_typeCO);

ViewBag.HandBookOfCOTypes = type;

// Формируем список статусов для передачи в представление

SelectList status = new SelectList(db.LegalStatus, "Id", "lsName", co.fk\_legalStatus);

ViewBag.LegalStatus = status; return View(co);

}

return RedirectToAction("ListCO");

}

// POST: CO/Edit/5

[HttpPost]

public ActionResult COEdit(CadastralObject co)

{

db.Entry(co).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("ListCO");

}

// GET: CO/Delete/5

[HttpGet]

public ActionResult CODelete(int id)

{

CadastralObject co = db.CadastralObjects.Find(id);

if (co == null)

{

return HttpNotFound();

}

return View(co);

}

// POST: CO/Delete/5

[HttpPost, ActionName("CODelete")]

public ActionResult CODeleteConfirmed(int id)

{

CadastralObject co = db.CadastralObjects.Find(id);

if (co == null)

{

return HttpNotFound();

}

db.CadastralObjects.Remove(co);

db.SaveChanges();

return RedirectToAction("ListCO");

}

}

}

**COTypeController.cs**

using System.Data.Entity;

using System.Linq;

using System.Web.Mvc;

using WebAppCM.Models;

namespace WebAppCM.Controllers.COTypeController

{

public class COTypeController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: COType

[HttpGet]

public ActionResult ShowHandBookOfCOType()

{

return View(db.HandBookOfCOTypes.ToList());

}

// GET: COType/Details/5

public ActionResult typeCODetails(int? id)

{

if (id == null)

return HttpNotFound();

HandBookOfCOType typeCO = db.HandBookOfCOTypes.Find(id);

if (typeCO == null)

return HttpNotFound();

typeCO.co = db.CadastralObjects.Where(m => m.fk\_typeCO==typeCO.Id);

return View(typeCO);

}

// GET: COType/Create

[HttpGet]

public ActionResult Create()

{

return View();

}

// POST: COType/Create

[HttpPost]

public ActionResult Create(HandBookOfCOType type)

{

db.HandBookOfCOTypes.Add(type);

db.SaveChanges();

return RedirectToAction("ShowHandBookOfCOType");

}

// GET: COType/Edit/5

[HttpGet]

public ActionResult Edit(int? id)

{

if (id == null)

return HttpNotFound();

HandBookOfCOType type = db.HandBookOfCOTypes.Find(id);

if (type != null)

return View(type);

return HttpNotFound();

}

// POST: COType/Edit/5

[HttpPost]

public ActionResult Edit(HandBookOfCOType type)

{

db.Entry(type).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("ShowHandBookOfCOType");

}

// GET: COType/Delete/5

[HttpGet]

public ActionResult Delete(int id)

{

HandBookOfCOType type = db.HandBookOfCOTypes.Find(id);

if (type == null)

return HttpNotFound();

return View(type);

}

// POST: COType/Delete/5

[HttpPost, ActionName("Delete")]

public ActionResult typeDelete(int id)

{

HandBookOfCOType type = db.HandBookOfCOTypes.Find(id);

if (type == null)

return HttpNotFound();

db.HandBookOfCOTypes.Remove(type);

db.SaveChanges();

return RedirectToAction("ShowHandBookOfCOType");

}

}

}

**CWController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using WebAppCM.Models;

using System.Data.Entity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.AspNet.Identity;

namespace WebAppCM.Controllers.CWController

{

public class CWController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: CW

[HttpGet, Authorize(Roles = "Admin, Engineer")]

public ActionResult ListCW()

{

var items = db.CadastralWorks.Include(p => p.User).Include(p => p.app);

return View(items);

}

private ApplicationUserManager UserManager

{ get

{return HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();

}

}

// GET: CW/Create

[HttpGet]

public ActionResult CWCreate(int id)

{

ApplicationUser user = UserManager.FindByEmail(User.Identity.Name);

ViewBag.User = user;

Application app = db.Applications.Find(id);

ViewBag.App = app;

return View();

}

// POST: CW/Create

[HttpPost]

public ActionResult CWCreate(CadastralWork cw, int id)

{

Application app = db.Applications.Find(id);

if (app == null)

return HttpNotFound();

app.fk\_status = 4;

db.Entry(app).State = EntityState.Modified;

db.SaveChanges();

db.CadastralWorks.Add(cw);

db.SaveChanges();

return RedirectToAction("ListCW");

}

// GET: CW/Edit/5

[HttpGet]

public ActionResult CWEdit(int? id)

{

if (id == null)

return HttpNotFound();

// Находим в бд выбранную КР

CadastralWork cw = db.CadastralWorks.Find(id);

if (cw != null)

return View(cw);

return RedirectToAction("ListCW");

}

// POST: CW/Edit/5

[HttpPost]

public ActionResult CWEdit(CadastralWork cw)

{

db.Entry(cw).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("ListCW");

}

// GET: CW/Delete/5

[HttpGet]

public ActionResult CWDelete(int id)

{

CadastralWork cw = db.CadastralWorks.Find(id);

if (cw == null)

return HttpNotFound();

return View(cw);

}

// POST: CW/Delete/5

[HttpPost, ActionName("CWDelete")]

public ActionResult CWDeleteConfirmed(int id)

{

CadastralWork cw = db.CadastralWorks.Find(id);

if (cw == null)

{

return HttpNotFound();

}

db.CadastralWorks.Remove(cw);

db.SaveChanges();

return RedirectToAction("ListCW");

}

// GET: CW/Complete

[HttpGet]

public ActionResult CWComplete(int id)

{

CadastralWork cw = db.CadastralWorks.Find(id);

ViewBag.CW = cw;

ViewBag.Date = DateTime.Now;

// Формируем список статусов для передачи в представление

SelectList status = new SelectList(db.LegalStatus, "Id", "lsName");

ViewBag.LegalStatus = status;

return View();

}

// POST: CW/Complete

[HttpPost]

public ActionResult CWComplete(CadastralObject co, int id)

{

CadastralWork cw = db.CadastralWorks.Find(id);

if (cw == null)

return HttpNotFound();

cw.app.fk\_status = 6;

db.Entry(cw).State = EntityState.Modified;

db.SaveChanges();

db.CadastralObjects.Add(co);

db.SaveChanges();

return RedirectToAction("ListCO", "CO");

}

}

}

Изменения в **AccountController.cs**

namespace WebAppCM.Controllers

{

[Authorize]

public class AccountController : Controller

{

// GET: /Account/Register

[AllowAnonymous]

public ActionResult Register()

{

return View();

}

// POST: /Account/Register

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

var user = new ApplicationUser {

surname = model.surname,

name = model.name,

patronimic = model.patronymic,

UserName = model.Email,

Email = model.Email,

PhoneNumber = model.PhoneNumber };

var result = await UserManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

await SignInManager.SignInAsync(user, isPersistent:false, rememberBrowser:false);

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

}

AddErrors(result);

}

// Появление этого сообщения означает наличие ошибки; повторное отображение формы

return View(model);

}

}

}

**Представления:**

**Register.cshtml**

@model WebAppCM.Models.RegisterViewModel

@{

ViewBag.Title = "Регистрация";

}

<h2>@ViewBag.Title</h2>

@using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))

{

@Html.AntiForgeryToken()

<h4>Введите ваши данные.</h4>

<hr />

<div class="form-group">

@Html.LabelFor(m => m.surname, new { @class = "col-md-2 control-label" })

<div class="col-md-10">

@Html.TextBoxFor(m => m.surname, new { @class = "form-control" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.name, new { @class = "col-md-2 control-label" })

<div class="col-md-10">

@Html.TextBoxFor(m => m.name, new { @class = "form-control" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.patronymic, new { @class = "col-md-2 control-label" })

<div class="col-md-10">

@Html.TextBoxFor(m => m.patronymic, new { @class = "form-control" })

</div>

</div>

<div class="form-group">

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

<div class="col-md-10">

@Html.TextBoxFor(m => m.PhoneNumber, new { @class = "form-control" })

</div>

</div>

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

<div class="form-group">

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

<div class="col-md-10">

@Html.TextBoxFor(m => m.Email, new { @class = "form-control" })

</div>

</div>

<div class="form-group">

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

<div class="col-md-10">

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

</div>

</div>

<div class="form-group">

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

<div class="col-md-10">

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

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" class="btn btn-default" value="Регистрация" />

</div>

</div>

}

@section Scripts {

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

}

**\_Layout.cshtml**

<!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">

<title>Кадастровое управление</title>

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

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

</head>

<body>

<div class="navbar navbar-inverse navbar-fixed-top">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

@if (!ViewContext.HttpContext.User.IsInRole("Engineer"))

{@Html.ActionLink("Главная", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })}

</div>

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li>@Html.ActionLink("Заявки", "ListApp", "App")</li>

@if (ViewContext.HttpContext.User.IsInRole("Engineer") || ViewContext.HttpContext.User.IsInRole("Admin"))

{<li>@Html.ActionLink("Кадастровые работы", "ListCW", "CW")</li>}

<li>@Html.ActionLink("Кадастровые объекты", "ListCO", "CO")</li>

@if (!ViewContext.HttpContext.User.IsInRole("Engineer"))

{<li>@Html.ActionLink("О сайте", "About", "Home")</li>

<li>@Html.ActionLink("Контакты", "Contact", "Home")</li>}

@\*@if (ViewContext.HttpContext.User.IsInRole("Admin"))

{<li>@Html.ActionLink("Пользователи", "ListUsers", "Manage")</li>}\*@

</ul>

@Html.Partial("\_LoginPartial")

</div>

</div>

</div>

<div id="header" class="text-center"><img src="/Content/images/Header.png" /></div>

<div 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>

**ListApp.cshtml**

@model IEnumerable<WebAppCM.Models.Application>

@{

ViewBag.Title = "Список заявок";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div>

<h3>Заявки </h3><hr />

@if (ViewContext.HttpContext.User.IsInRole("Customer") || ViewContext.HttpContext.User.IsInRole("Admin"))

{<a href="/App/AppCreate" class="btn btn-primary btn-lg">Создать новую заявку</a><br/><br/>}

<table class="text-center table-bordered">

<tr>

<td style="width: 2%;"><p><b>№</b></p></td>

<td style="width: 10%;"><p><b>Заказчик</b></p></td>

<td style="width: 6%;"><p><b>Дата</b></p></td>

<td style="width: 10%;"><p><b>Кадастровый объект</b></p></td>

<td style="width: 10%;"><p><b>Тип кадастровой работы</b></p></td>

<td style="width: 6%;"><p><b>Описание</b></p></td>

<td style="width: 8%;"><p><b>Статус</b></p></td>

<td style="width: 10%;"><p><b>Оплачено, ₽</b></p></td>

<td style="width: 13%;"><p><b>Опции</b></p></td>

@if (ViewContext.HttpContext.User.IsInRole("Admin"))

{ <td style="width: 13%;"><p><b>Опции</b></p></td>}

</tr>

@foreach (var m in Model.Reverse())

{<tr>

@if (!ViewContext.HttpContext.User.IsInRole("Customer"))

{ <td><p>@Html.DisplayFor(model => m.Id)</p></td>

<td><p>@m.User.surname @m.User.name @m.User.patronimic</p></td>

<td><p>@Html.DisplayFor(model => m.date)</p></td>

<td><p>@Html.DisplayFor(model => m.HandBookOfCOType.tHCOname)</p></td>

<td><p>@Html.DisplayFor(model => m.TypeCW.tCWname)</p></td>

<td><p>@Html.DisplayFor(model => m.description)</p></td>

<td><p>@Html.DisplayFor(model => m.Status.sName)</p></td>

<td><p>@Html.DisplayFor(model => m.cost)</p></td>}

@if (ViewBag.User.Id == m.User.Id && ViewContext.HttpContext.User.IsInRole("Customer"))

{ <td><p>@Html.DisplayFor(model => m.Id)</p></td>

<td><p>@m.User.surname @m.User.name @m.User.patronimic</p></td>

<td><p>@Html.DisplayFor(model => m.date)</p></td>

<td><p>@Html.DisplayFor(model => m.HandBookOfCOType.tHCOname)</p></td>

<td><p>@Html.DisplayFor(model => m.TypeCW.tCWname)</p></td>

<td><p>@Html.DisplayFor(model => m.description)</p></td>

<td><p>@Html.DisplayFor(model => m.Status.sName)</p></td>

<td><p>@Html.DisplayFor(model => m.cost)</p></td>}

@if ((ViewBag.User.Id == m.User.Id && ViewContext.HttpContext.User.IsInRole("Customer")) ||

ViewContext.HttpContext.User.IsInRole("Admin"))

{<td><p><a href="/App/AppEdit/@m.Id" class="btn btn-primary btn-sm">Изменить</a>

<a href="/App/AppCancel/@m.Id" class="btn btn-warning btn-sm">Отменить</a></p>

<a href="/App/AppDelete/@m.Id" class="btn btn-danger btn-sm">Удалить</a>

@if (m.fk\_status == 2)

{<a href="/App/AppPay/@m.Id" class="btn btn-success btn-sm">Оплатить</a>}

</td>

}

@if (!ViewContext.HttpContext.User.IsInRole("Customer"))

{<td>@if (m.fk\_status == 3)

{<a href="/CW/CWCreate/@m.Id" class="btn btn-success btn-sm">Принять</a>

<a href="/App/AppCancel/@m.Id" class="btn btn-danger btn-sm">Отклонить</a>}

</td>

}

</tr>

}

</table>

</div>

**AppCreate.cshtml −** аналогичен AppEdit.cshtml

@model WebAppCM.Models.Application

@{

ViewBag.Title = "Новая заявка";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Новая заявка</h2>

@using (Html.BeginForm("AppCreate", "App", FormMethod.Post))

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<hr />

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

<input type="hidden" value="@ViewBag.User.Id" name="fk\_User" />

<input type="hidden" value="@ViewBag.Date" name="date" />

<input type="hidden" value="@ViewBag.Status" name="fk\_status" />

<input type="hidden" value="01.01.1900" name="PayModel.Validity" />

<div class="form-group">

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

<div class="col-md-3">

<div class="form-control">

@ViewBag.User.surname @ViewBag.User.name @ViewBag.User.patronimic

</div>

</div>

</div>

<div class="form-group">

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

<div class="col-md-3">

<div class="form-control">

@ViewBag.Date

</div>

</div>

</div>

<div class="form-group">

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

<div class="col-md-10">

@Html.DropDownListFor(m => m.fk\_typeCO, ViewBag.HandBookOfCOTypes as SelectList, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(m => m.fk\_typeCO, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

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

<div class="col-md-10">

@Html.DropDownListFor(m => m.fk\_typeCW, ViewBag.TypeCW as SelectList, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(m => m.fk\_typeCW, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

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

<div class="col-md-10">

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

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

</div>

</div>

<div class="form-group">

<a href="/App/ListApp" class="btn btn-default btn-lg">&laquo; Вернуться к списку заявок</a>&emsp;

<input type="submit" value="Отправить" class="btn btn-primary btn-lg" />

</div>

</div>

}

**AppCancel.cshtml −** аналогичен AppDelete.cshtml

@model WebAppCM.Models.Application

@{

ViewBag.Title = "Отмена заявки";

}

<h2>Отмена заявки № @Model.Id</h2>

<h3>Вы уверены, что хотите отменить эту заявку?</h3>

<div><hr />

<dl class="dl-horizontal">

<dt>@Html.DisplayNameFor(model => model.fk\_typeCO)</dt>

<dd>@Html.DisplayFor(model => model.HandBookOfCOType.tHCOname)</dd>

<dt>@Html.DisplayNameFor(model => model.fk\_typeCW)</dt>

<dd>@Html.DisplayFor(model => model.TypeCW.tCWname)</dd>

<dt>@Html.DisplayNameFor(model => model.description)</dt>

<dd>@Html.DisplayFor(model => model.description)</dd>

<dt>@Html.DisplayNameFor(model => model.fk\_status)</dt>

<dd>@Html.DisplayFor(model => model.Status.sName)</dd>

</dl>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-actions no-color">

<a href="/App/ListApp" class="btn btn-default btn-lg">&laquo; Вернуться к списку заявок</a>&emsp;

<input type="submit" value="Отменить" class="btn btn-danger btn-lg" />

</div>

}

</div>

**AppPay.cshtml**

@model WebAppCM.Models.Application

@{

ViewBag.Title = "Оплата заявки";

}

<h2>Оплата заявки № @Model.Id</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal"><hr />

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

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

@{int cost = 0;}

@switch (Model.fk\_typeCW)

{

case 1: cost = 30000; break;

case 2: cost = 10000; break;

case 3: cost = 5000; break;

case 4: cost = 50000; break;

}

@switch (Model.fk\_typeCO)

{

case 1: cost = cost + 20000; break;

case 2: cost = cost + 10000; break;

case 4: cost = cost + 5000; break;

case 5: cost = cost + 50000; break;

}

<h3>Сумма к оплате: @cost₽</h3><br/>

<h4>Введите данные вашей карты: </h4>

<div class="form-group">

<div class="col-md-10">

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

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

</div>

</div>

<input type="hidden" value="@Model.fk\_User" name="fk\_User" />

<input type="hidden" value="@Model.date" name="date" />

<input type="hidden" value="@Model.fk\_typeCO" name="fk\_typeCO" />

<input type="hidden" value="@Model.fk\_typeCW" name="fk\_typeCW" />

<input type="hidden" value="@Model.description" name="description" />

<input type="hidden" value=3 name="fk\_status" />

<input type="hidden" value="@cost" name="cost" />

<div class="form-group">

<a href="/App/ListApp" class="btn btn-default btn-lg">&laquo; Оплатить позже</a>&emsp;

<input type="submit" value="Оплатить" class="btn btn-success btn-lg" />

</div>

</div>

}

**ListCW.cshtml**

@model IEnumerable<WebAppCM.Models.CadastralWork>

@{

ViewBag.Title = "ListCW";

}

<h2>Список Кадастровых работ</h2>

<table class="text-center table-bordered">

<tr>

<td style="width: 2%;"><p><b>№</b></p></td>

<td style="width: 10%;"><p><b>Тип кадастровой работы</b></p></td>

<td style="width: 10%;"><p><b>Кадастровый объект</b></p></td>

<td style="width: 10%;"><p><b>Заказчик</b></p></td>

<td style="width: 10%;"><p><b>Инженер</b></p></td>

<td style="width: 8%;"><p><b>Дата</b></p></td>

<td style="width: 6%;"><p><b>Учет</b></p></td>

<td style="width: 8%;"><p><b>Стоимость</b></p></td>

<td style="width: 10%;"><p><b>Дополнения</b></p></td>

<td style="width: 14%;"><p><b>Опции</b></p></td>

</tr>

@foreach (var m in Model.Reverse())

{

<tr><td>@Html.DisplayFor(modelItem => m.Id)</td>

<td>@Html.DisplayFor(modelItem => m.app.TypeCW.tCWname)</td>

<td>@Html.DisplayFor(modelItem => m.app.HandBookOfCOType.tHCOname)</td>

<td>@m.app.User.surname @m.app.User.name @m.app.User.patronimic</td>

<td>@m.User.surname @m.User.name @m.User.patronimic</td>

<td>@m.app.date</td>

<td><a href="/CW/CWDoc/@m.Id" class="btn btn-group-sm">Документы</a></td>

<td>@Html.DisplayFor(modelItem => m.app.cost)</td>

<td>@Html.DisplayFor(modelItem => m.description)</td>

<td><a href="/CW/CWEdit/@m.Id" class="btn btn-primary btn-sm">Изменить</a>

<a href="/CW/CWDelete/@m.Id" class="btn btn-danger btn-sm">Удалить</a>

<a href="/CW/CWCancel/@m.Id" class="btn btn-warning btn-sm">Отменить</a>

<a href="/CW/CWComplete/@m.Id" class="btn btn-success btn-sm">Завершить</a></td>

</tr>

}

</table>

**CWComplete.cshtml –** почти аналагично **COCreate.cshtml**

@model WebAppCM.Models.CadastralObject

@{

ViewBag.Title = "Создание кадастрового объекта";

}

<h2>Создание кадастрового объекта</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal"><hr />

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

<input type="hidden" value="@ViewBag.CW.app.fk\_typeCO" name="fk\_typeCO" />

<input type="hidden" value="@ViewBag.Date" name="dateOfEntry" />

<input type="hidden" value="@ViewBag.CW.app.cost" name="cost" />

<div class="form-group">

@Html.LabelFor(model => model.fk\_typeCO, "Тип объекта", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-3">

<div class ="form-control">

@ViewBag.CW.app.HandBookOfCOType.tHCOname

</div>

</div>

</div>

<div class="form-group">

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

<div class="col-md-10">

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

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

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.dateOfEntry,"Дата постановки на учет", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-3">

<div class="form-control">

@ViewBag.Date

</div>

</div>

</div>

<div class="form-group">

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

<div class="col-md-10">

@Html.DropDownListFor(model => model.fk\_legalStatus, ViewBag.LegalStatus as SelectList, htmlAttributes: new { @class = "form-control" })

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

</div>

</div>

<div class="form-group">

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

<div class="col-md-10">

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

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

</div>

</div>

<div class="form-group">

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

<div class="col-md-10">

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

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

</div>

</div>

<div class="form-group">

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

<div class="col-md-3">

<div class="form-control">

@ViewBag.CW.app.cost

</div>

</div>

</div>

<div class="form-group">

<a href="/CW/ListCW" class="btn btn-default btn-lg">&laquo; Вернуться к списку Кадастровыx работ</a>&emsp;

<input type="submit" value="Добавить" class="btn btn-primary btn-lg" />

</div>

</div>

}

**ListCO.cshtml**

@model IEnumerable<WebAppCM.Models.CadastralObject>

@{

ViewBag.Title = "Кадастровые объекты";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div>

<h3>Кадастровые объекты</h3><hr />

@if (!ViewContext.HttpContext.User.IsInRole("Customer"))

{<a href="/CO/COCreate" class="btn btn-primary btn-lg">Создать новый Кадастровый объект </a> }&emsp;

<a href="/COType/ShowHandBookOfCOType" class="btn btn-default btn-lg">Справочник типов кадастровых объектов</a> <br /><br />

<table class="text-center table-bordered">

<tr>

<td style="width: 2%;"><p><b>№</b></p></td>

<td style="width: 10%;"><p><b>Тип</b></p></td>

<td style="width: 8%;"><p><b>Номер</b></p></td>

<td style="width: 10%;"><p><b>Дата постановки</b></p></td>

<td style="width: 10%;"><p><b>Законный статус</b></p></td>

<td style="width: 12%;"><p><b>Адресс</b></p></td>

<td style="width: 8%;"><p><b>Площадь, м²</b></p></td>

<td style="width: 10%;"><p><b>Стоимость, ₽</b></p></td>

@if (!ViewContext.HttpContext.User.IsInRole("Customer"))

{<td style="width: 13%;"><p><b>Опции</b></p></td>}

</tr>

@foreach (var m in Model.Reverse())

{ <tr>

<td><p>@Html.DisplayFor(model => m.Id)</p></td>

<td><p>@Html.DisplayFor(model => m.HandBookOfCOType.tHCOname)</p></td>

<td><p>@Html.DisplayFor(model => m.cadastralNumber)</p></td>

<td><p>@Html.ValueFor(model => m.dateOfEntry, "{0:dd/MM/yyyy}")</p></td>

<td><p>@Html.DisplayFor(model => m.LegalStatus.lsName)</p></td>

<td><p>@Html.DisplayFor(model => m.address)</p></td>

<td><p>@Html.DisplayFor(model => m.square)</p></td>

<td><p>@Html.DisplayFor(model => m.cost)</p></td>

@if (!ViewContext.HttpContext.User.IsInRole("Customer"))

{<td><p><a href="/CO/COEdit/@m.Id" class="btn btn-primary btn-sm">Изменить</a>&emsp;

<a href="/CO/CODelete/@m.Id" class="btn btn-danger btn-sm">Удалить</a></p></td>}

</tr>

}

</table>

</div>

**typeCODetails.cshtml**

@model WebAppCM.Models.HandBookOfCOType

@{

ViewBag.Title = "Тип @Model.tHCOname";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Детали</h2>

<div>

<h4>Тип @Model.tHCOname</h4><hr />

@foreach (var co in Model.CadastralObjects)

{

<p> № @co.Id Стоимость: @co.cost (Адресс: @co.address)

Дата: @co.dateOfEntry (Номер: @co.cadastralNumber)</p>

<p> <a href="/COType/Edit/@Model.Id" class="btn btn-default btn-sm">>Изменить</a>&emsp;

<a href="/COType/ShowHandBookOfCOType" class="btn btn-default btn-sm">&laquo;Вернуться к Справочнику типов</a></p>}

</div>

**Loggers:**

**Logger.cs**

using log4net;

using log4net.Config;

namespace WebAppCM

{

public class Logger

{

private static readonly ILog logger = LogManager.GetLogger("MyLogger");

public static ILog Log

{

get { return logger; }

}

public static void Init()

{

XmlConfigurator.Configure();

logger.Info("Logger Init...");

}

}

}

**Global.asax**

using log4net;

using System.Web.Mvc;

using System.Web.Optimization;

using System.Web.Routing;

namespace WebAppCM {

public class MvcApplication : System.Web.HttpApplication {

private static readonly ILog logger = LogManager.GetLogger("MyLogger");

protected void Application\_Start(){

logger.Info("Application Start");

AreaRegistration.RegisterAllAreas();

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

}

override public void Init(){

logger.Info("Application Init");

}

override public void Dispose(){

logger.Info("Application Dispose");

}

protected void Application\_Error(){

logger.Info("Application Error");

logger.Error("Application Error");

}

protected void Application\_End(){

logger.Info("Application End");

}

}}

Изм.

Лист

№ Докум.

Подпись

Дата

Масса

Лист 1

ВлГУ 09.03.04.ПРИ-116.11.01.ГП

Разработал

Мартынов Г. И.

Проверил

Вершинин В.В.

Н. Контр.

Утвердил

Прототип программной системы «Кадастровое управление»

Графическое приложение.

Лит.

Масштаб

Листов 1

ПРИ -116

У

14.01

Вершинин В.В.

Т. Контр.

Реценз.

Диаграмма прецедентов