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

Курсового проекта

**по дисциплине   
"** **Технологии программирования "**

**на тему**

Проектирование и разработка программной системы   
**«Паспорт стул»**

Выполнил: ст. гр. ИСТ-116

Мануэл И.К.

Принял:

Вершинин В.В.

Владимир, 2018

**Аннотация**

В данной курсовой работе разрабатывается программная система «**Паспорт стул**». 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 37 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 "Passport table". 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 37 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

3.4 Диаграмма состояний 14

3.5 Диаграмма последовательности 16

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

4.7 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 «**Паспорт стул**». 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 “Draw.io”, реализация программного обеспечения с использованием технологий разработки ASP.NET MVC, Code First, Entity Framework, Razor, Bootstrap. Среда программирования – Visual Studio 2017.

**1 ПОСТАНОВКА ЦЕЛИ И ЗАДАЧ**

**1.1 Цель**

1.1.1 Целью создания программной системы «**Паспорт стул**» является упрощение работы инженера и клиент, 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.

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.

**2 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

**2.1** **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 паспорта (PasportRequest)»**

2.2.3.1 Взаимодействует с заказчиком, отражает его цели.**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.

**2.2.10 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:

• Администратор;

• Инженер;

• Заказчик (клиент);

• Кроме того, «Пользователь» как родитель остальных пользователей и «Гость» как незарегистрированный пользователь.

**2.3.1.1.1 Пользователь** системы может:

− Смотреть заявку

− Смотреть информация

**2.3.1.2 Пользователь с ролью Заказчик может:**

− Смотреть информация;

− Смотреть заявку;

− Создать заявку;

− Оплатить заявку;

− Отменить заявку;

− Редактировать заявку;

**2.3.1.3 Пользователь с ролью Инженер может:**

− Создать;

− Обработать заявку (отклонить/ одобрить)

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

− Ведуть (проводить) сайт.

**3 проектирование системы**

**3.1.1 Название прецедента «Создать 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".

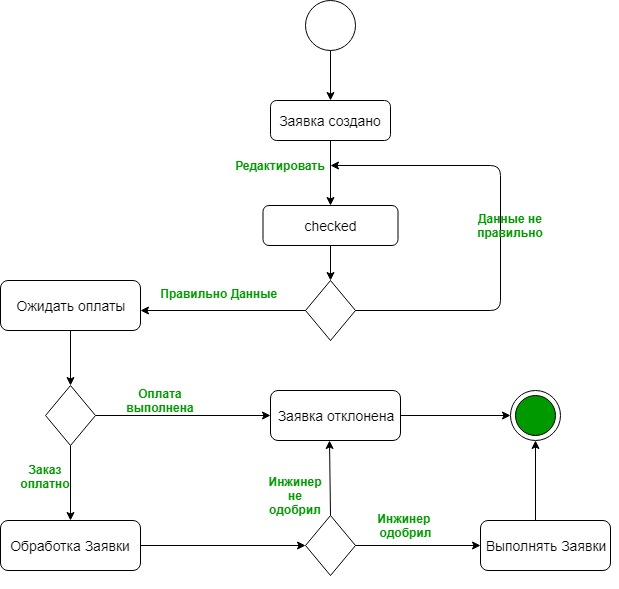


Рисунок 4 – Диаграмма состояний для объекта «Заявка»

**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.

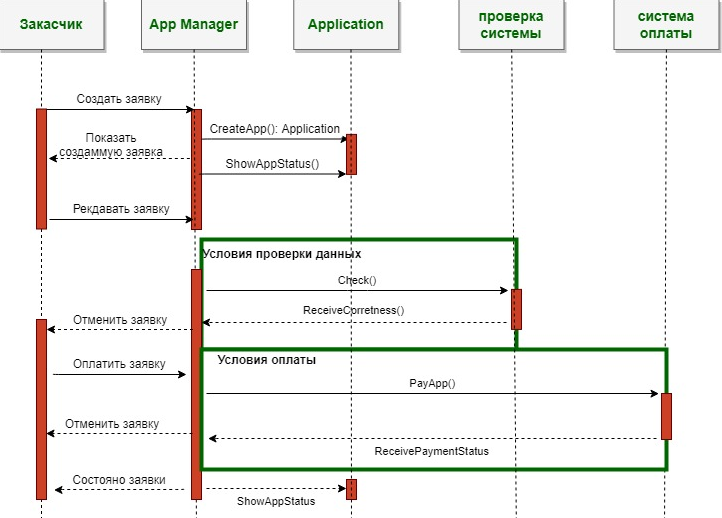


Рис. 5 – Диаграмма последовательностей для объекта «Заявка» с актером «Заказчик»

**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 2017 и нормализованная к третьей нормальной форме, показана на рисунке 7.

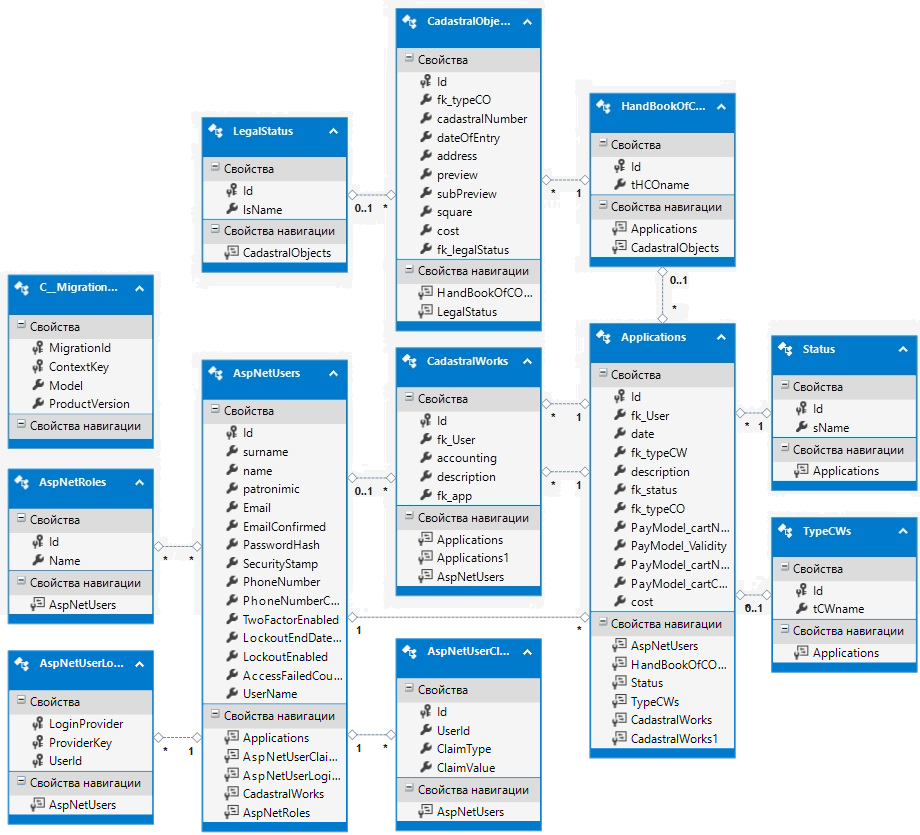


Рисунок 7 – Логическая схема модели базы данных

1. **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".

**4.7 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.

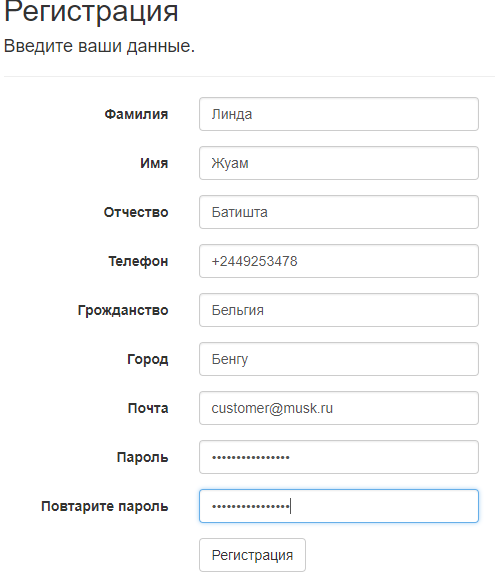


Рисунок 8 – Страница регистрации

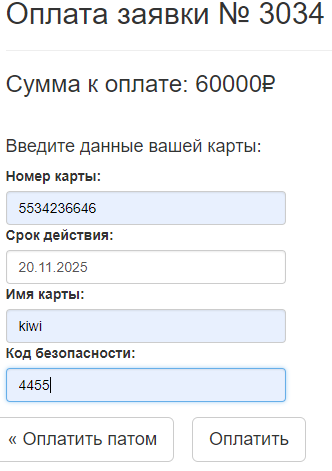


Рисунок 17 – Оплатить заявку

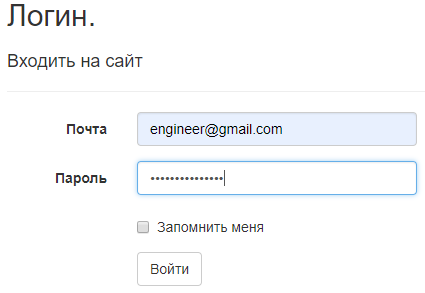


Рисунок 21 – Вход в роль инженера

Статус заявки изменится на «Выполняется» и кнопки пропадут

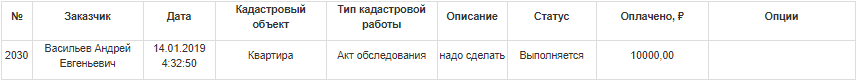


Рисунок 23 – Изменение статуса заявки на «Выполняется»

Статус заявки изменится на «Завершен».

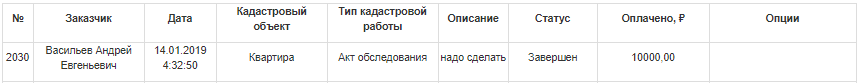


Рисунок 26 – Изменение статуса заявки на «Завершен»

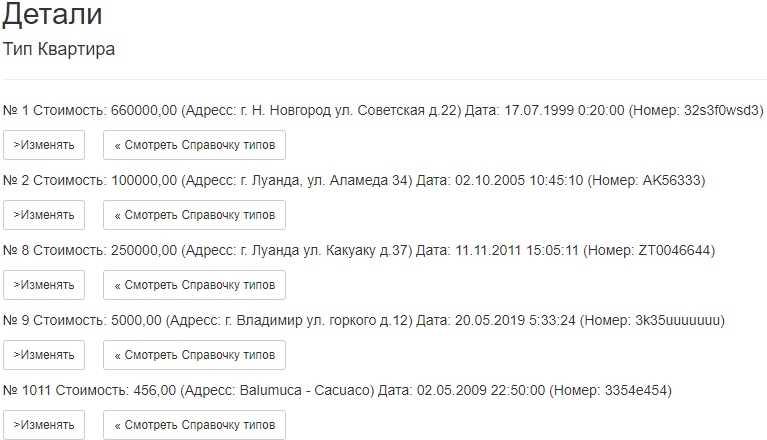


Рисунок 28 – Детали

# заключение

В результате выполнения работы была спроектирована и реализована программная система учета «**Паспорт стул**». 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 «Паспорт стул». 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);

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. Youtube: https://www.youtube.com/watch?v=eeTSG4Iu8ac
2. Stack overflow: https://stackoverflow.com/questions/17922945/how-to-enable-migration-to-update-my-database-in-mvc4
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. Microsoft docs: <https://docs.microsoft.com/en-us/aspnet/mvc/mvc5>

**ПРИЛОЖЕНИЕ**

**ИСХОДНЫЙ КОД ПРОГРАММНОЙ СИСТЕМЫ**

**Модели**

**PasportRequest.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace PasportProvide.Models.Databases

{

public enum RequestType

{

Create,

Recreate

}

public enum RequstResult

{

NotResult,

Success,

Dismiss,

IncorrectData,

Close

}

public class PasportRequest

{

public int ID { get; set; }

public User User { get; set; }

public User Executor { get; set; }

public DateTime DateTime { get; set; }

public RequestType RequestType { get; set; }

public string Comment { get; set; }

public RequstResult RequstResult { get; set; }

}

}

**Application.cs**

namespace CadaManage.Models

{

using System;

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; }

}

}

**Status.cs**

namespace CadaManage.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>();

}

}

**User.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace PasportProvide.Models.Databases

{

public enum UserType

{

Admin,

Provider,

User

}

public class User

{

public int ID { get; set; }

public string Login { get; set; }

public string PasswordHash { get; set; }

public DateTime Birthday { get; set; }

public string PasportNum { get; set; }

public UserType UserType { get; set; }

}

}

**DatabaseContext.cs**

using Microsoft.EntityFrameworkCore;

using PasportProvide.Models.Databases;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace PasportProvide.Models

{

public class DatabaseContext : DbContext

{

public DbSet<User> Users { get; set; }

public DbSet<PasportRequest> PasportRequests { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

base.OnConfiguring(optionsBuilder);

optionsBuilder.UseSqlServer(@"Data Source=ACER-PC\SQLEXPRESS;Initial Catalog=PasportProvider;Integrated Security=True");

}

}

}

**LegalStatus.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace CadaManage.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>();

}

}

**ErrorViewModel.cs**

using System;

namespace PasportProvide.Models

{

public class ErrorViewModel

{

public string RequestId { get; set; }

public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

}

}

**AccountViewModels.cs**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace WebAppCM.Models

{

public class LoginViewModel

{

[Required]

[Display(Name = "Почта")]

[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 = "Телефон")]

public string PhoneNumber { get; set; }

//[Required]

[Display(Name = "Город")]

public string City { get; set; }

//[Required]

[Display(Name = "Грожданство")]

public string Nationality { get; set; }

[Required]

[EmailAddress]

[Display(Name = "Почта")]

public string Email { get; set; }

[Required]

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

[DataType(DataType.Password)]

[Display(Name = "Пароль")]

public string Password { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Повтарите пароль")]

[Compare("Password", ErrorMessage = "Пароль не совпадают!!")]

public string ConfirmPassword { get; set; }

}

}

**Контроллеры:**

**PasportProvideController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Claims;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Logging;

using PasportProvide.Models;

using PasportProvide.Models.Databases;

namespace PasportProvide.Controllers

{

[Authorize(Roles = "Provider")]

public class PasportProvideController : Controller

{

private int UserID => int.Parse(User.Claims.FirstOrDefault(x => x.Type == ClaimTypes.NameIdentifier).Value);

ILogger<PasportProvideController> logger;

public PasportProvideController(ILogger<PasportProvideController> logger)

{

this.logger = logger ?? throw new ArgumentNullException(nameof(logger));

}

public IActionResult Index()

{

DatabaseContext databaseContext = new DatabaseContext();

var requests = databaseContext.PasportRequests.Include(x => x.Executor).Include(x => x.User).Where(x => x.Executor.ID == UserID

&& x.RequstResult == RequstResult.NotResult).ToList();

return View(requests);

}

public IActionResult RequestView(int id)

{

DatabaseContext databaseContext = new DatabaseContext();

return View(databaseContext.PasportRequests.Where(x => x.ID == id)

.Include(x => x.User).Include(x => x.Executor).FirstOrDefault());

}

[HttpPost]

public IActionResult RequestView(PasportRequest pasportRequest)

{

DatabaseContext databaseContext = new DatabaseContext();

var pReq = databaseContext.PasportRequests.Find(pasportRequest.ID);

pReq.Comment = pasportRequest.Comment;

pReq.RequstResult = pasportRequest.RequstResult;

databaseContext.Update(pReq);

databaseContext.SaveChanges();

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

}

}

}

**AppController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using CadaManage.Models;

using System.Data.Entity;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

namespace CadaManage.Controllers.AppController

{

public class AppController : Controller

{

private ApplicationDbContext db = new ApplicationDbContext();

// GET: App

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

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");

}

}

}

**AdminController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using PasportProvide.Models;

using PasportProvide.Models.Databases;

using System;

using System.Linq;

namespace PasportProvide.Controllers

{

[Authorize(Roles = "Admin")]

public class AdminController : Controller

{

ILogger<AdminController> logger;

public AdminController(ILogger<AdminController> logger)

{

this.logger = logger ?? throw new ArgumentNullException(nameof(logger));

}

public IActionResult Index()

{

DatabaseContext databaseContext = new DatabaseContext();

return View(databaseContext.Users.Where(x => x.UserType != UserType.Admin).ToList());

}

public IActionResult ChangeType(int id, [FromForm]UserType userType)

{

DatabaseContext databaseContext = new DatabaseContext();

var user = databaseContext.Users.Find(id);

user.UserType = userType;

databaseContext.Update(user);

databaseContext.SaveChanges();

return Redirect("Index");

}

}

}

**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 CadaManage.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>

<div class="form-group">

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

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

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

</div>

</div>

<div class="form-group">

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

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

@Html.TextBoxFor(m => m.City, 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 charset="utf-8" />

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

<title>@ViewData["Title"] - PasportProvide</title>

<environment include="Development">

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />

<link rel="stylesheet" href="~/css/site.css" />

</environment>

<environment exclude="Development">

<link rel="stylesheet" href="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/css/bootstrap.min.css"

asp-fallback-href="~/lib/bootstrap/dist/css/bootstrap.min.css"

asp-fallback-test-class="sr-only" asp-fallback-test-property="position" asp-fallback-test-value="absolute" />

<link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true" />

</environment>

</head>

<body>

<nav 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="sr-only">Toggle navigation</span>

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

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

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

</button>

<a asp-area="" asp-controller="Home" asp-action="Index" class="navbar-brand">PasportProvide</a>

</div>

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li><a asp-area="" asp-controller="Home" asp-action="Index">Home</a></li>

@{

var userType = (UserType)Enum.Parse(typeof(UserType), User.Claims.FirstOrDefault(x => x.Type == System.Security.Claims.ClaimTypes.Role).Value);

switch (userType)

{

case UserType.Admin:

<li><a asp-area="" asp-controller="Admin" asp-action="Index">Admin</a></li>

break;

case UserType.Provider:

<li><a asp-area="" asp-controller="PasportProvide" asp-action="Index">Pasport Provide</a></li>

break;

case UserType.User:

<li><a asp-area="" asp-controller="Home" asp-action="Notification">Notification</a></li>

<li><a asp-area="" asp-controller="Account" asp-action="ChangeUser">Change User</a></li>

break;

}

}

<li><a asp-area="" asp-controller="Account" asp-action="Signout">Signout</a></li>

</ul>

</div>

</div>

</nav>

<div class="container body-content">

@RenderBody()

<hr />

<footer>

<p>&copy; 2019 - PasportProvide</p>

</footer>

</div>

<environment include="Development">

<script src="~/lib/jquery/dist/jquery.js"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

</environment>

<environment exclude="Development">

<script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.3.1.min.js"

asp-fallback-src="~/lib/jquery/dist/jquery.min.js"

asp-fallback-test="window.jQuery"

crossorigin="anonymous"

integrity="sha384-tsQFqpEReu7ZLhBV2VZlAu7zcOV+rXbYlF2cqB8txI/8aZajjp4Bqd+V6D5IgvKT">

</script>

<script src="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/bootstrap.min.js"

asp-fallback-src="~/lib/bootstrap/dist/js/bootstrap.min.js"

asp-fallback-test="window.jQuery && window.jQuery.fn && window.jQuery.fn.modal"

crossorigin="anonymous"

integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa">

</script>

<script src="~/js/site.min.js" asp-append-version="true"></script>

</environment>

@RenderSection("Scripts", required: false)

</body>

</html>

**IndexPassportProvider.cshtml**

@model List<PasportRequest>

@{

ViewData["Title"] = "Index";

}

<h2>Index</h2>

<style>

table {

width: 100%;

}

td {

border: 1px solid grey;

padding: 10px 15px 10px 15px;

text-align: center;

}

.hide {

/\*border: none;\*/

}

</style>

<table>

<tr>

<td>Request Type</td>

<td>Datetime</td>

<td>User Login</td>

<td></td>

</tr>

@foreach (var req in Model)

{

<tr>

<td>@req.RequestType.ToString()</td>

<td>@req.DateTime.Date.ToString()</td>

<td>@req.User.Login</td>

<td>

@Html.ActionLink("Review", "RequestView", "PasportProvide", new { id = req.ID })

</td>

</tr>

}

</table>

**Notification.cshtml**

@model List<PasportRequest>

@{

ViewData["Title"] = "Notification";

}

<style>

table {

width: 100%;

}

td {

border: 1px solid grey;

padding: 10px 15px 10px 15px;

text-align: center;

}

.hide {

/\*border: none;\*/

}

</style>

<h2>Notification</h2>

<table>

<tr>

<td>Request Type</td>

<td>Datetime</td>

<td>Request Result</td>

<td>Comment</td>

<td></td>

</tr>

@foreach (var req in Model)

{

<tr>

<td>@req.RequestType.ToString()</td>

<td>@req.DateTime.Date.ToString()</td>

<td>@req.RequstResult.ToString()</td>

<td>@req.Comment</td>

<td>@(req.RequstResult == RequstResult.Dismiss ? Html.ActionLink("ReSend", "Recreate", "Home", new { id = req.ID }) : Html.ActionLink("", ""))</td>

</tr>

}

</table>

**CreateRequest.cshtml**

@model PasportRequest

@{

ViewData["Title"] = "CreateRequest";

}

<h2>CreateRequest</h2>

@using (Html.BeginForm("CreateRequest", "Home", FormMethod.Post))

{

<div class="form-group">

<label asp-for="RequestType" class="control-label"></label>

<select asp-for="RequestType" asp-items="Html.GetEnumSelectList<RequestType>()" class="form-control"></select>

<span asp-validation-for="RequestType" class="text-danger"></span>

</div>

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

}

**Login.cshtml**

@model LoginModel

@{

ViewData["Title"] = "Login";

Layout = null;

}

<head>

<link rel="stylesheet" href="~/css/unlayout.css" />

</head>

<body>

@using (Html.BeginForm("Login", "Account", FormMethod.Post))

{

<h2>Login</h2>

<div class="label">

@Html.LabelFor(x => x.Login)

</div>

<div class="input">

@Html.TextBoxFor(x => x.Login)

@Html.ValidationMessageFor(x => x.Login)

</div>

<div class="label">

@Html.LabelFor(x => x.Password)

</div>

<div class="input">

@Html.PasswordFor(x => x.Password)

@Html.ValidationMessageFor(x => x.Password)

</div>

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

@Html.ActionLink("Register", "Register", "Account", new { }, new { @class = "btn" })

}

</body>

**Loggers:**

**Logger.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using log4net;

using log4net.Config;

namespace CadaManage

{

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 log4net.Config;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Optimization;

using System.Web.Routing;

using CadaManage.Models;

namespace CadaManage

{

//public class MvcApplication : System.Web.HttpApplication

public class MvcApplication: 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.ИСТ-116.15.03.ГП

Разработал

.Мануэл И.

Проверил

Вершинин В.В.

Н. Контр.

Утвердил

Прототип программной системы «Паспорт стул»

Графическое приложение.

Лит.

Масштаб

Листов 1

ИСТ -116

У

12.03

Вершинин В.В.

Т. Контр.

Реценз.