*Министерство образования и науки Российской Федерации*

***не Федеральное государственное бюджетное образовательное учреждение высшего образования***

*«Владимирский государственный университет*

*имени Александра Григорьевича и Николая Григорьевича Столетовых»*

**(ВлГУ)**

*Кафедра информационных систем и программной инженерии*

***КУРСОВОЙ ПРОЕКТ***

Проектирование и разработка программной системе

″автоматизации документооборота малого предприятия″

*Выполнил: студент гр. ПРИ-121*

*Нгуамене Лонла В.В.*

*Принял: доц. Вершинин В.В.*

*Владимир, 2023*

***Аннотация***

*Суть курсового проекта заключает проектирование и реализация программной системе «автоматизации документооборота малого предприятия».* *В процессе работы разрабатывает необходимый функционала ПС..*

*Курсовой проект представлен на 35 страницах, рисунков - 18, источников - 3, приложений - 1, иллюстрационный материал на 3 листе формат A1.*

***Annotation***

*The essence of the course project is the design and implementation of the software system "automation of document flow of a small enterprise". In the process of work, the software system develops the necessary functionality.*

*The course project is presented on 35 pages, figures - 18, sources - 3, appendices - 1, illustrative material on 3 sheet A1 format.*

Содержание

[**ВВЕДЕНИЕ 4**](#_Toc93258893)

[**1. Постановка Цели и Задачи 5**](#_Toc93258894)

[**2.1 Цель 5**](#_Toc93258895)

[**2.2 Задачи 5**](#_Toc93258896)

[**2. Описание предметной области 6**](#_Toc93258897)

[**2.1 Словарь предметной области 6**](#_Toc93258898)

[**3. Проектирование Системы 7**](#_Toc93258899)

[**3.1 Диаграмма прецедентов 7**](#_Toc93258900)

[**3.2 Диаграмма Последовательность 8**](#_Toc93258901)

[**3.3 Диаграмма классов 10**](#_Toc93258902)

[**4. Реализация Программной Системе 11**](#_Toc93258903)

[**4.1 Принципы Организации Проекта 11**](#_Toc93258904)

[**4.2 Реализация Модели 11**](#_Toc93258905)

[**4.3 Реализация аутентификация 11**](#_Toc93258906)

[**4.4 Пользовательский Интерфейс 12**](#_Toc93258907)

[**5 ЗАКЛЮЧЕИИЕ 20**](#_Toc93258908)

[**6 СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ 21**](#_Toc93258909)

[**ПРИЛОЖЕНИЕ А 22**](#_Toc93258910)

[**ИХОДНЫЙ КОД ПРОГРАММННОЙ СИСТЕМЕ 22**](#_Toc93258911)

# **ВВЕДЕНИЕ**

Заданием данной курсовой проект является проектирование и реализация программной системе «автоматизации документооборота малого предприятия». Данная система нужна для того чтобы представить пользователю удобной инструмент для операции с документами.

В ходе выполнения курсового проекта выполнена разработки проекта информационной системы с использованием языка UML, реализация диаграмм с использованием среды Microsoft Office Visio, реализация программного обеспечения с использованием технологий разработки ASP.NET framework.

# **1. Постановка Цели и Задачи**

## 2.1 Цель

Целью данного курсового проект является разработка программной системе ″автоматизации документооборота малого предприятия″, которая обеспечивает удобной функционал для создания, хранения, исполнения документы. Система решает проблемы:

* + хранение и опасность потери документов при их передаче из одного подразделения в другое;
  + образование двух параллельных видов документооборота: бумажного и электронного (дублирование бумажных документов электронными).

## 2.2 Задачи

В процессе прототипа программной системы необходимо:

1. Выполнить исследование и анализ предметной область;
2. Разработать прототип ПС;
3. Выполнить моделирование работы ПС;
4. Разработать схему БД;
5. Реализовать ПС с использованием выбранных средств и технологии.

Исходной данные:

1. Язык моделирование UML;
2. Платформа разработке ASP.NET.

Функциональные требование представляемые к разрабатываемые ПС:

1. Авторизации различных типов пользователей (администратор, директор, начальник отдела);
2. Поддержка справочника отделов, пользователей (адм.);
3. Поддержка справочник типов документ;
4. Добавление нового документа и его учет по исполнителям(директор/начальник/система);
5. Редактирование существующий документ (директор/ начальник /система);
6. удаление существующий документ(директор/система);
7. Отчет по исполненным/неисполненным документом (директор/система).

Нефункциональные требование, предъявляемые к разрабатываемой ПС:

* Система должна быть реализована на языке C# в виде веб-приложения.

# **2. Описание предметной области**

Разрабатываемая система предназначен для движения документов с момента их создания или получения до завершения исполнения или отправки.

В процессе документооборота с документами совершаются определенные делопроизводственные операции, а не просто передача из одного подразделения в другое или от исполнителя к исполнителю.

На малых предприятиях обрабатывается и хранится огромное количество информации, связанной с обеспечением взаимодействия с партнёрами и клиентами.

Пользователи система имеют возможность добавление новый документ, редактировать или удалить существующий документ, и получить отчет по исполненным/неисполненным.

## 2.1 Словарь предметной области

2.1 **Сущности «Пользователь»**

Человек, который использует систем для добавления, редактирования, и удаления документ.

**2.2 Сущности «документ».**

информация на материальном носителе, имеющая юридическую силу.

**2.3 Сущности «Тип-Документ».**

Вид документа, который должен быть в системе.

**2.4 Сущности «Подготовил».**

человек, который поставил документ в системе.

**2.5 Сущности «подписал».**

человек, который подписал документ.

**2.6 Сущности «Должность».**

Рабочее место человека, который подготовил или подписал документ.

**2.7 Сущности «Статус-документа».**

В каком состояние находится документ.

**2.8 Сущности «Контрагент».**

физическое или юридическое лицо, которое выступает сделки.

**2.9 Сущности «Тип-Договор».**

Соглашение между собой предприятия и другой компании.

**2.10 Сущности «Сотрудник».**

Человек, который работает на предприятии.

# **3. Проектирование Системы**

## 3.1 Диаграмма прецедентов

Диаграмма прецедента программной системе ″автоматизации документооборота малого предприятия″, представлен на (Рисунке 1).



Рисунок 1 – диаграмма президентов

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

**Название: “**Добавление Документ в системе**”**

**Предусловия:** нет

**Действующее лицо:** Начальник

**Основной поток:** Добавление Документ в системе

сотрудник открывают форму (окно приложения), используется поля ввода текста, вводит документ, и нажимает кнопку сохранить.

**Альтернативный паток**: если у пользователя нет поддержка система или не авторизирован, то он не может добавить, редактировать документ. Ему надо получит поддержка.

**Пост условия**: если документ не подписан, то он может отправит сообщение на подпись документ к директору, после подписания документа он может подготовить и сохранить.

**Название: “**Операции с аккаунтом**”**

**Предусловия:** нет

**Действующее лицо:** Администратор

**Основной поток:** Операции с аккаунтом

Администратор открывают форму (окно приложения), используется форм он создает, либо редактирует аккаунт и нажимает кнопку создать, либо сохранить.

**Пост условия**: если аккаунт уже регистрирован, то он может дать поддержка система или его рол.

## 3.2 Диаграмма Последовательность

Для проектирования структура программной системе была разработана диаграмма последовательность, представляющая как добавить документ в системе (Рисунок 2).



Рис 2 - Диаграмма последовательность для прецедента «Добавить документ в системе»

## 3.3 Диаграмма классов

Диаграмма классов программной системе ″автоматизации документооборота малого предприятия″, представлен на (Рисунке 3).



Рис 3 – Диаграмма классов

# **4. Реализация Программной Системе**

## 4.1 Принципы Организации Проекта

Приложение построена как single page application, что позволило обрабатывать значительной часть бизнес-логики, на клиентской стороне. В качество Front-end framework использовался HTML, Bootstrap. Back-end разработан на .Net Framework, в качестве СУБД применялся MS SQL. В качестве системе контроля версий использовался Git. Исходный код приложение приведен в приложение А.

## 4.2 Реализация Модели

Модель построена с использованием Database-First – позволяет писать приложения для существующих баз данных. Во многих отношениях, подход Database-First является противоположностью подходу Model-First. При подходе Database-First база данных уже существует, поэтому разработчик должен знать, где расположена база данных, а также иметь информацию об имени базы данных. Тем не менее, разработчик не должен понимать внутреннюю работу базы данных - Entity Framework по-прежнему скрывает внутреннюю реализацию из поля зрения.

При этом подходе, рабочий процесс создания модели начинается с создания и проектирования базы данных. После генерации сущностных классов модели из существующей базы данных, работа с Entity Framework аналогична подходам Code-First и Model-First. Это означает создание объекта класса контекста и использование этого объекта для выполнения необходимых задач.

## 4.3 Реализация аутентификация

Для реализации аутентификации и авторизации пользователей были использованы JSON Web Tokens, позволяющий реализовать удобной платформ независимый и легко расширяемый способ авторизации.

## 4.4 Пользовательский Интерфейс

Пользовательский интерфейс представлен на рисунках 4-13.

На рисунке 4 представлен Home Page без аутентифицированного.

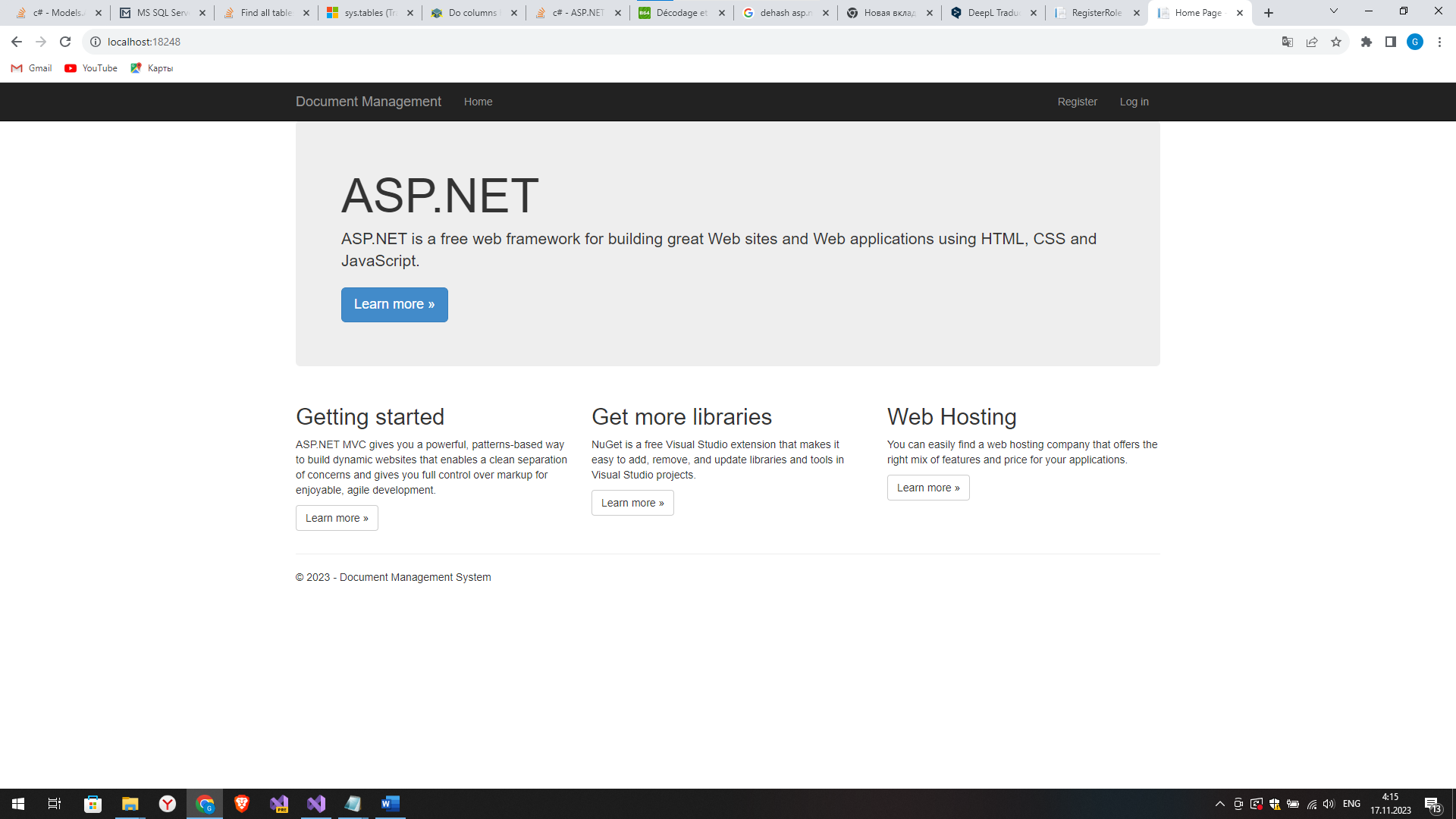


Рис 4 – Home Page без аутентифицированного.

На рисунке 5 представлен Страница входа в систему.

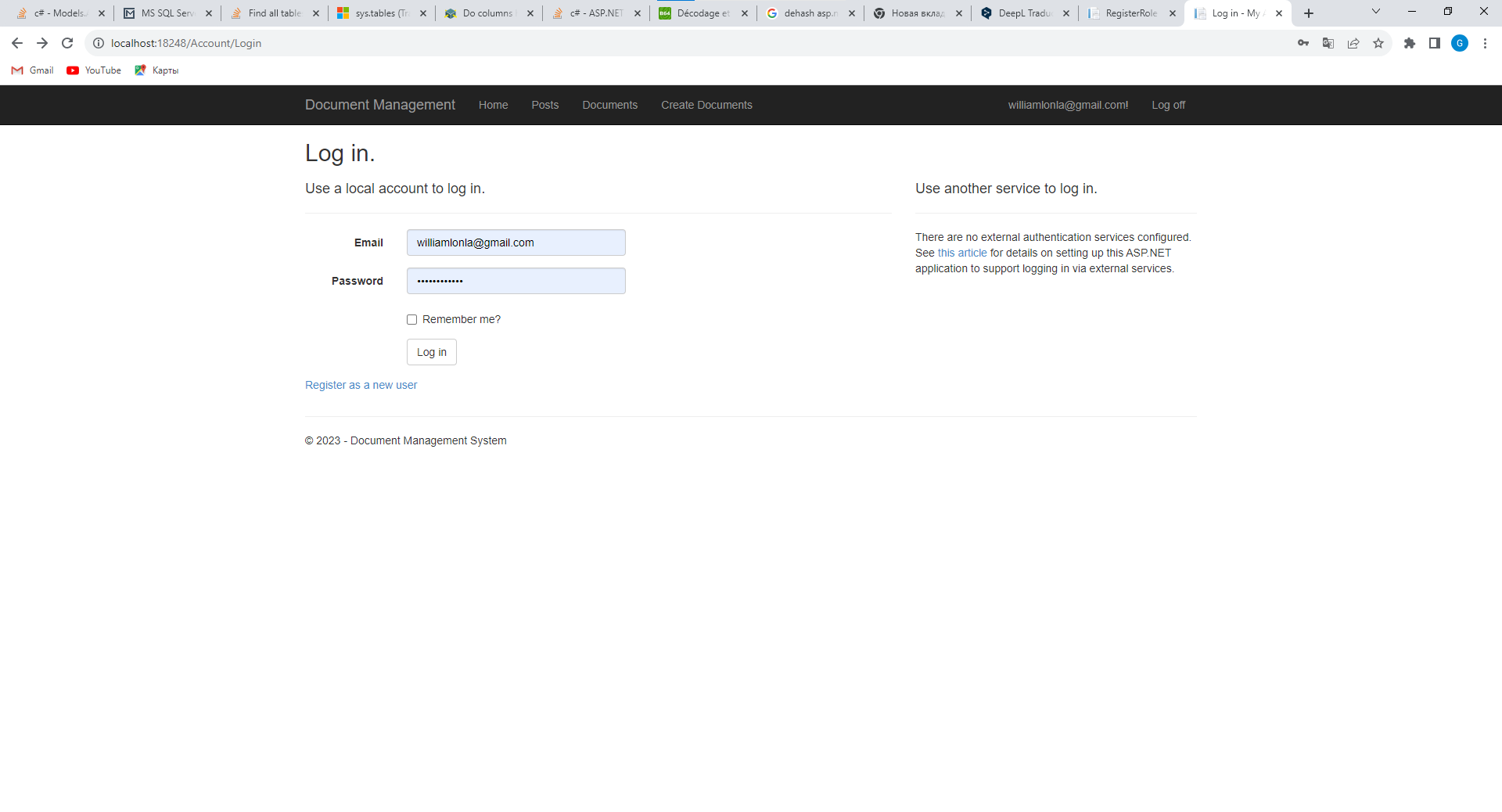


Рис 5 –Login Page

На рисунке 6 представлен Home Page аутентифицирован под сотрудником.

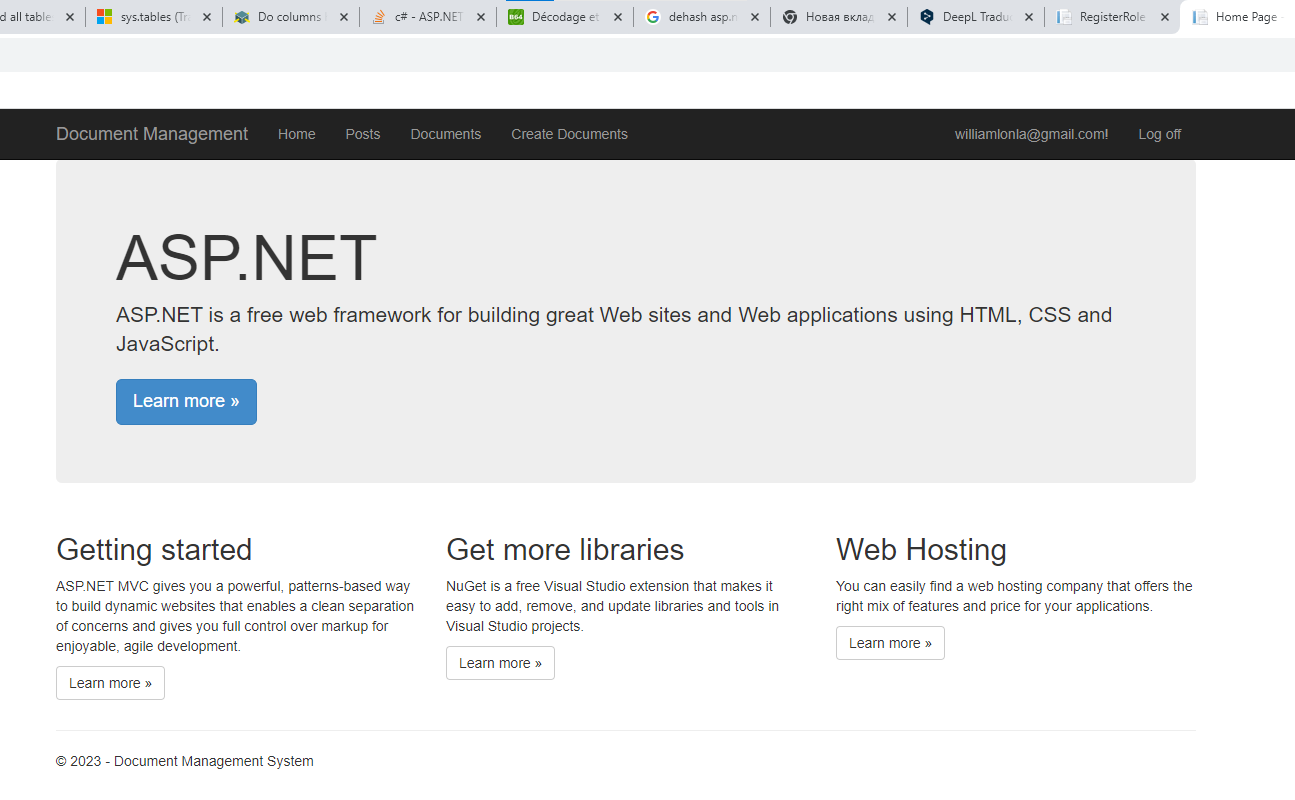


Рис 6 – Home Page аутентифицирован под сотрудником.

На рисунки 7,8 представлен форма создание документа.

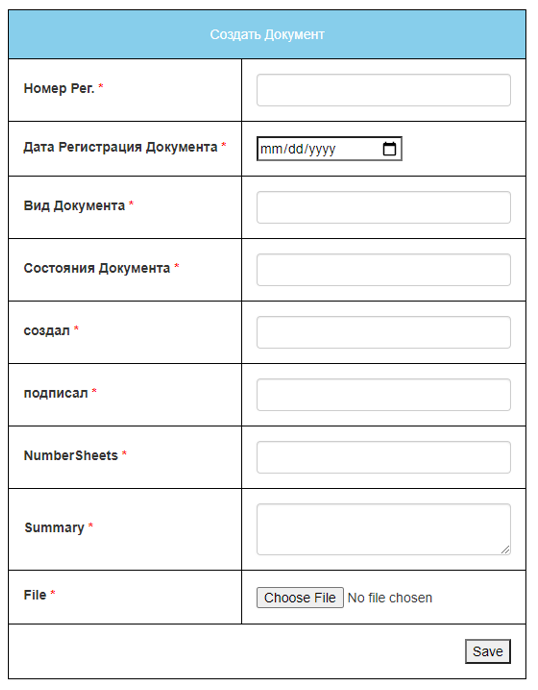


Рис 7 –Форма создание Документа.

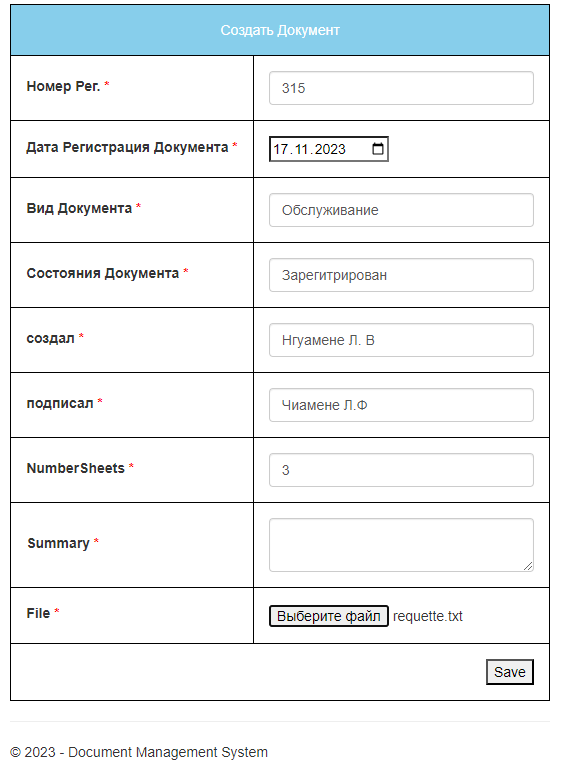


Рис 8 –Форма создание Документа с датами.

На рисунке 9 представлен Home Page аутентифицирован под администратором

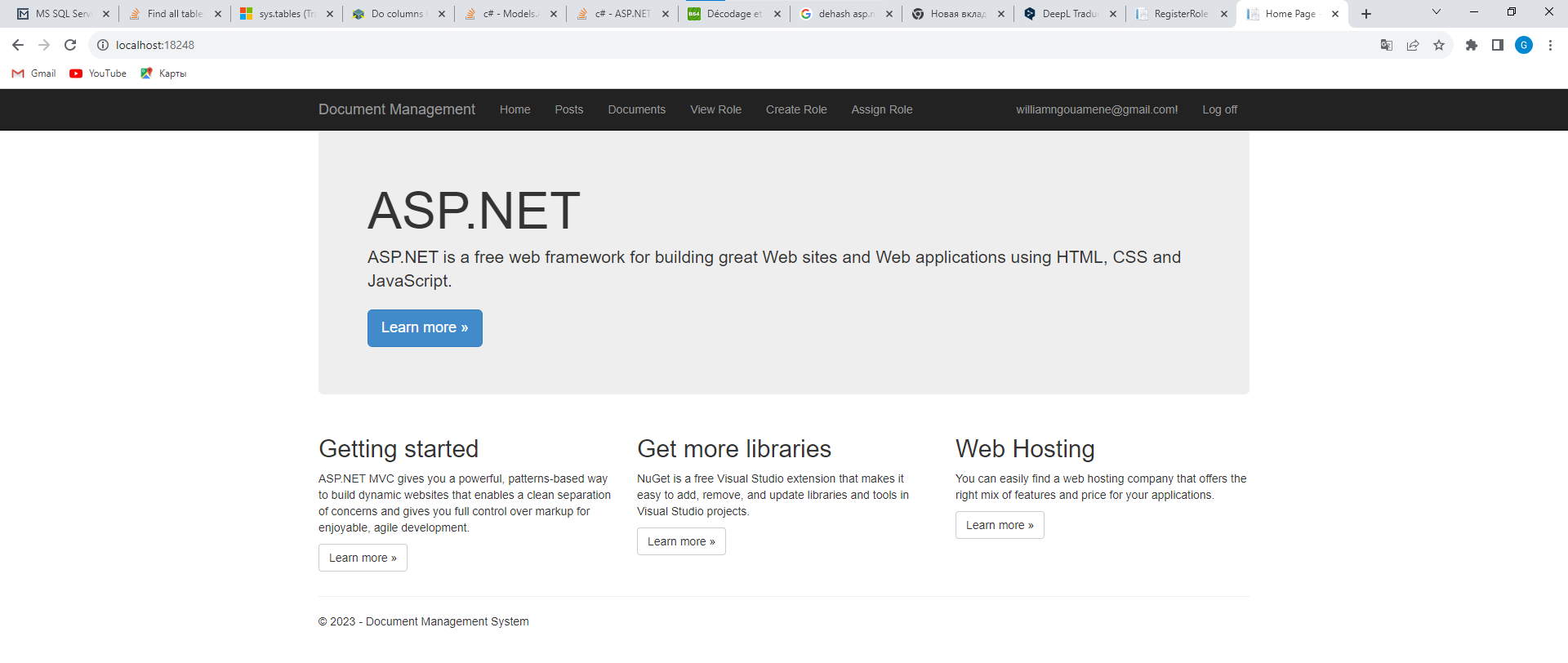


Рис 9 – Home Page аутентифицирован под администратором.

На рисунке 10 представлен список документ, сохранённый в системе.

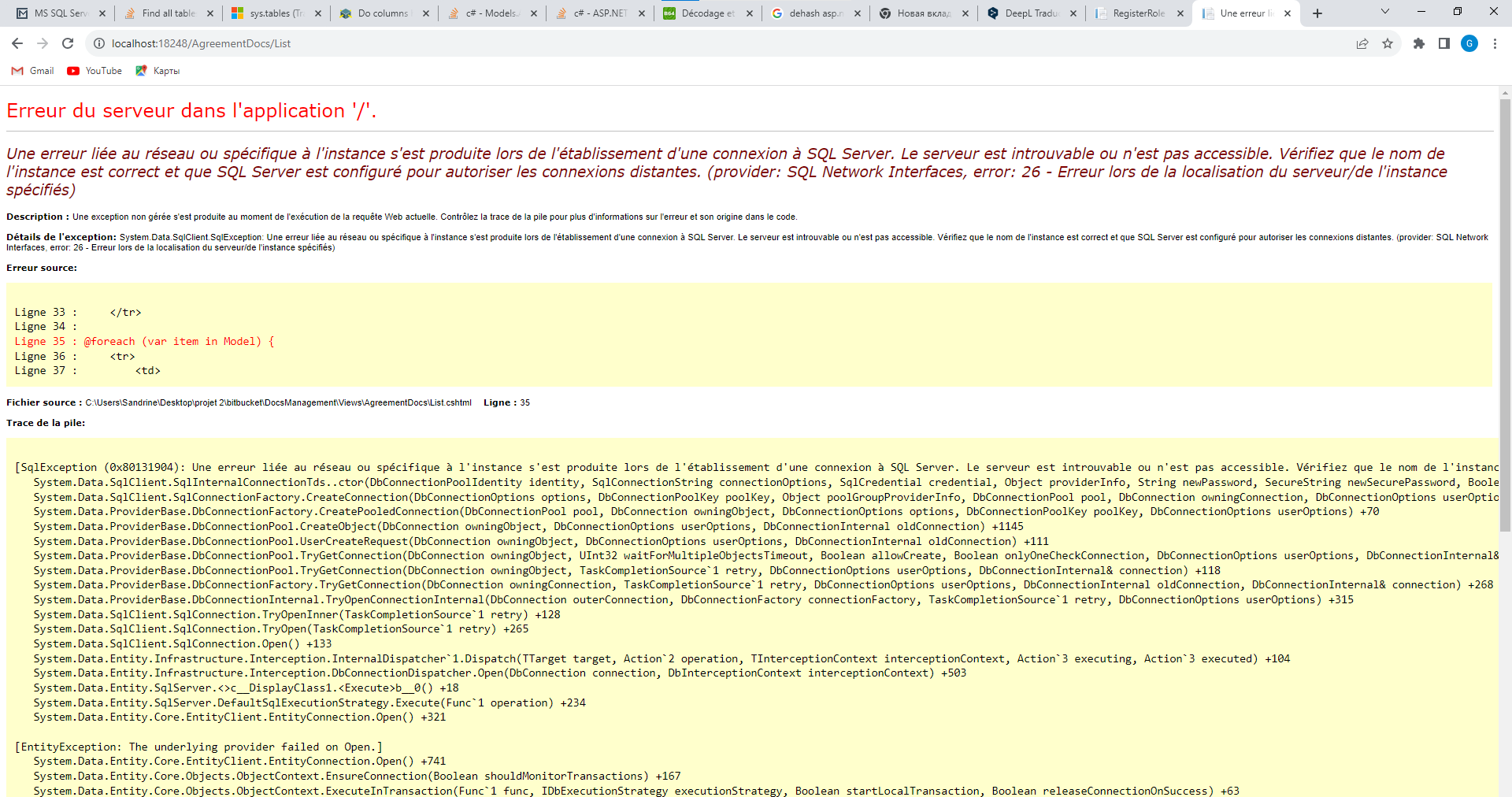


Рис 10 – список документов.

На рисунке 11 представлен редактирование документа.

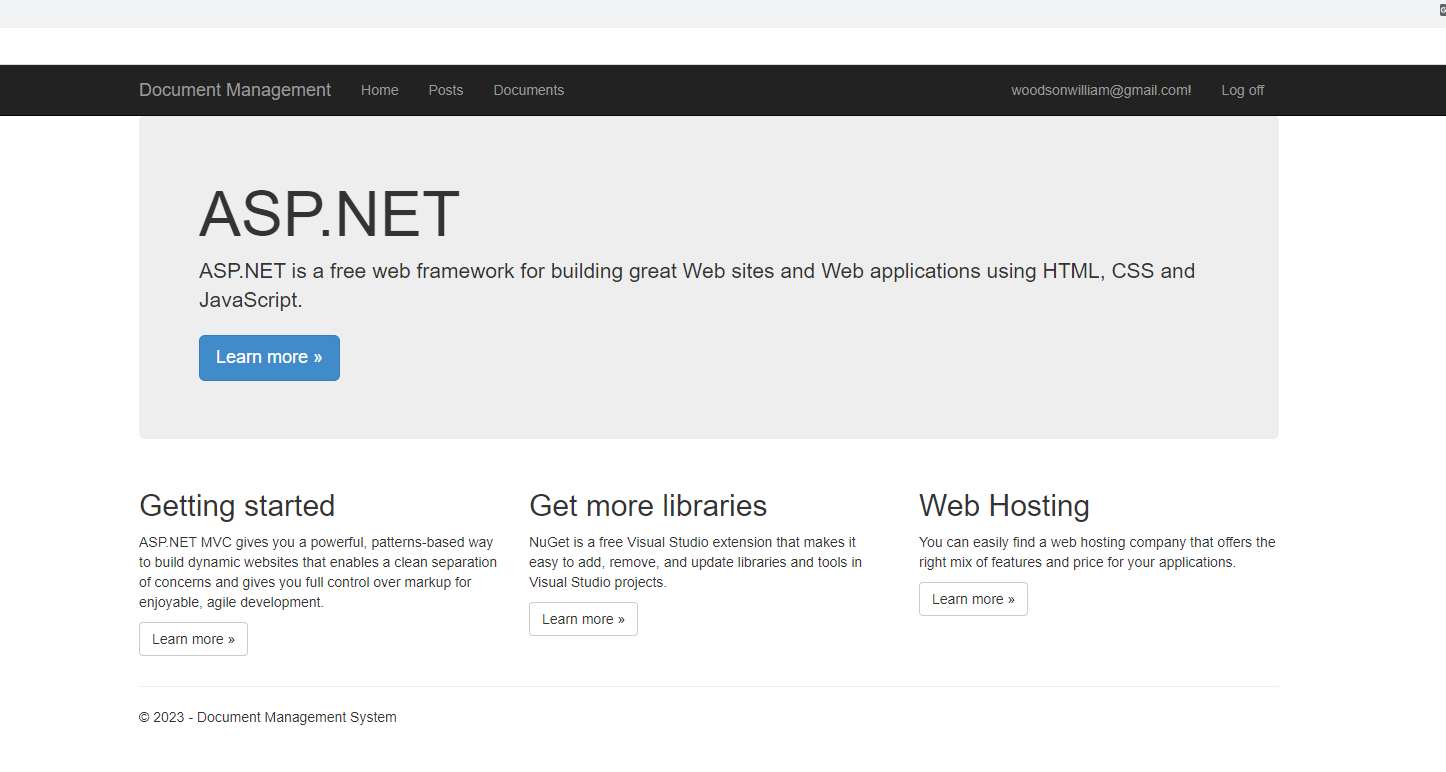
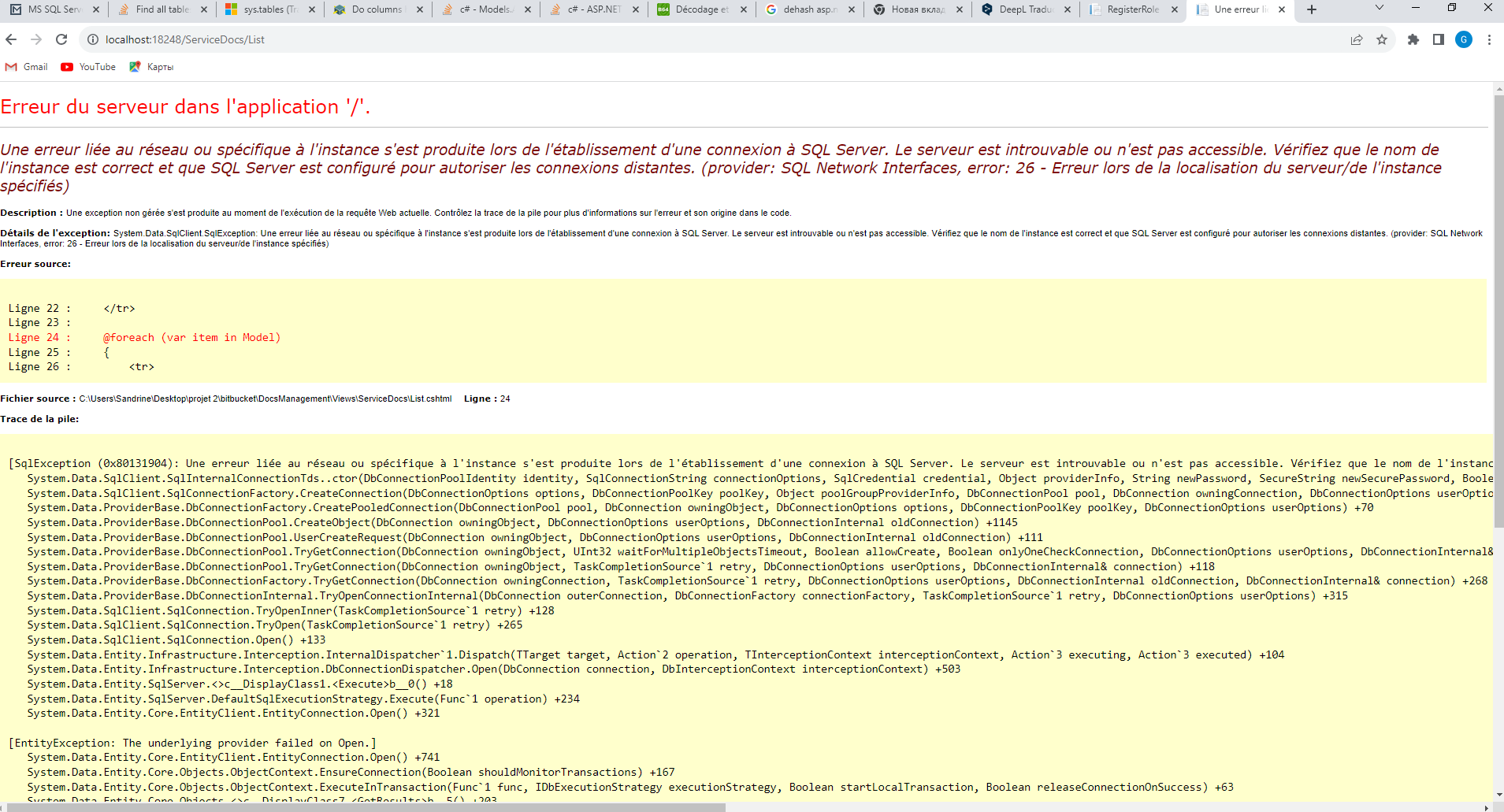


Рис 11 – редактирование документа. 

На рисунке 12 представлен Удаление документа



Рис 12 – Удаление документа.

На рисунки 13 представлены Создание Роли, Назначение роль пользователю, и просмотра Доступные Роли.

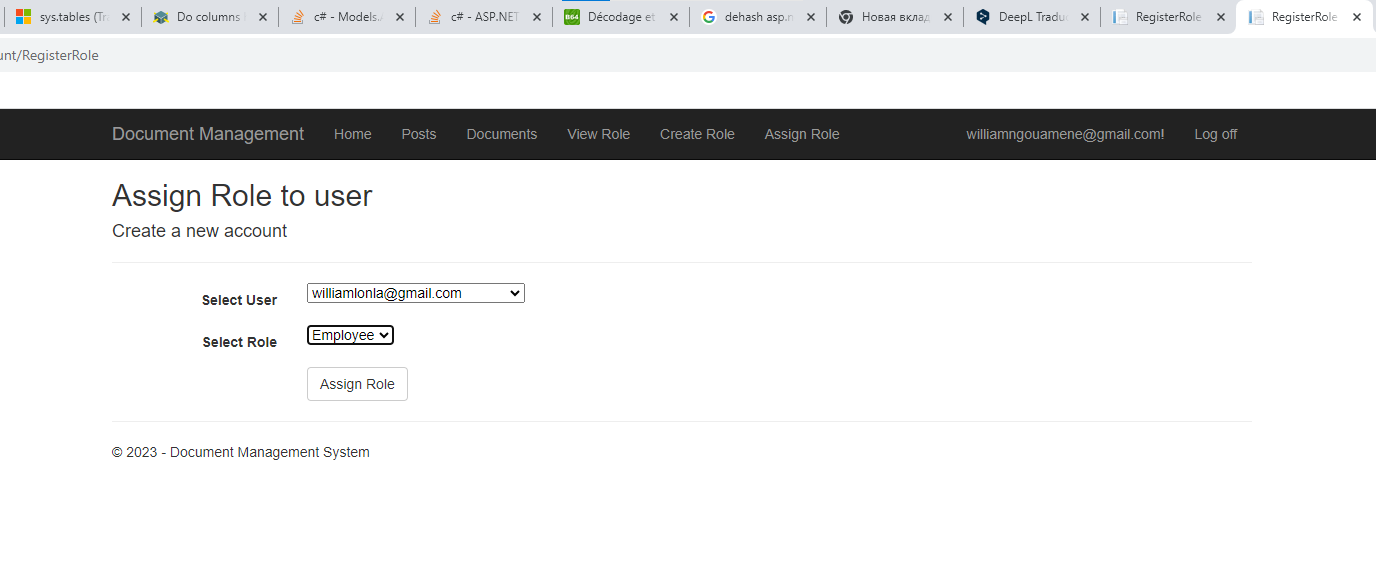


Рис 13 – Назначение роль пользователю

# **5 ЗАКЛЮЧЕИИЕ**

В результате выполнения работы была спроектировано и реализовано программная система «автоматизации документооборота малого предприятия». Она во многом удобна на практике, так как позволяет хранить, добавлять, редактировать, и удалять документы. Для выполнения доставленной задачи был выполнен:

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

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

# **6 СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ**

1. Pro ASP.NET Core. — 7-е изд. — Develop Cloud-Ready Web Applications Using MVC 2, Blazor, and Razor Pages, 2017, 1024 с. — ISBN-13 (pbk): 978-1-4842-3149-4.
2. <https://www.youtube.com/playlist?list=PLp1Emx1rT4z9MYuP7U8GVUMvKYz_NM9AY>.
3. https://www.compilemode.com/2017/01/downloading-files-from-database-in-asp-net-mvc.htm

# **ПРИЛОЖЕНИЕ А**

## ИХОДНЫЙ КОД ПРОГРАММННОЙ СИСТЕМЕ

**Startup.cs**

using Microsoft.Owin;

using Owin;

[assembly: OwinStartupAttribute(typeof(DocsManagement.Startup))]

namespace DocsManagement

{

public partial class Startup

{

public void Configuration(IAppBuilder app)

{

ConfigureAuth(app);

}

}

}

**Documents.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Threading.Tasks;

using System.Web;

namespace DocsManagement.Models

{

public class Documents

{

[Key]

[Required]

[Display(Name = "Номер Рег.")]

public int RegistrationNomer { get; set; }

[Required]

[Display(Name = "Дата Регистрация Документа")]

public System.DateTime RegistrationData { get; set; }

[Required]

[Display(Name = "Вид Документа")]

public String TypeDocument { get; set; }

[Required]

[Display(Name = "Состояния Документа")]

public String StateDocument { get; set; }

[Required]

[Display(Name = "создал")]

public String CreatedUser { get; set; }

[Required]

[Display(Name = "подписал")]

public String SignedUser { get; set; }

public Nullable<int> NumberSheets { get; set; }

public String Summary { get; set; }

[Display(Name = "File")]

public HttpPostedFileBase File { get; set; }

//public HttpPostedFileBase File { get; set; }

}

}

**AgreementDocument.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace DocsManagement.Models

{

public class AgreementDocument : Documents

{

[Required]

[Display(Name = "Вид Договора")]

public string TypeAgreement { get; set; }

[Required]

[Display(Name = "срок Договора")]

public System.DateTime DeadlineAgreement { get; set; }

[Required]

[Display(Name = "контрагента")]

public string Contractor { get; set; }

[Required]

[Range(0.01, double.MaxValue,

ErrorMessage = "Введите, Положительную Цену")]

public decimal Amount { get; set; }

}

}

**ServiceDocument.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace DocsManagement.Models

{

public class ServiceDocument : Documents

{

}

}

**IServiceDocsRepository.cs**

using System;

using System.Linq;

namespace DocsManagement.Models

{

public interface IServiceDocsRepository

{

IQueryable<Service> ServicesDocuments { get; }

void SaveServicesDocuments(ServiceDocument servicesDocuments, String File);

void DeleteServicesDocuments(int serviceDocsID);

}

}

**IAgreementDocsRepository.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace DocsManagement.Models

{

public interface IAgreementDocsRepository

{

IQueryable<Agreement> AgreementDocuments { get; }

void SaveAgreementDocuments(AgreementDocument agreementDocuments, String File);

void DeleteAgreementDocuments(int agreementDocsID);

}

}

**EFServicesDocsRepository.cs**

using DocsManagement.Controllers;

using DocsManagement.LogHelper;

using System;

using System.Collections.Generic;

using System.Data.Entity.Infrastructure;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

namespace DocsManagement.Models

{

public class EFServicesDocsRepository : IServiceDocsRepository

{

private static readonly log4net.ILog log = LogHellper.GetLogger();

private DocumentsDBEntities context = new DocumentsDBEntities();

public IQueryable<Service> ServicesDocuments => context.Services;

public void SaveServicesDocuments(ServiceDocument servicesDocS, String File)

{

try

{

FileInfo fil = new FileInfo(File);

using (DocumentsDBEntities doc = new DocumentsDBEntities())

{

var serdocs = new Service()

{

RegistrationNomer = servicesDocS.RegistrationNomer,

RegistrationData = servicesDocS.RegistrationData,

TypeDocument = servicesDocS.TypeDocument,

StateDocument = servicesDocS.StateDocument,

CreatedUser = servicesDocS.CreatedUser,

SignedUser = servicesDocS.SignedUser,

NumberSheets = servicesDocS.NumberSheets,

Summary = servicesDocS.Summary,

FileName = Path.GetFileName(File),

FileType = fil.Extension,

FileContent = System.IO.File.ReadAllBytes(File)

};

doc.Services.Add(serdocs);

doc.SaveChanges();

}

}

catch (DbUpdateException ex)

{

log.Error("Can not use existing Registration nomer!", ex);

}

}

public void DeleteServicesDocuments(int id)

{

Service doc = context.Services.Find(id);

if (doc != null)

{

context.Services.Remove(doc);

context.SaveChanges();

}

}

}

}

**EFAgreementDocsRepository.cs**

using DocsManagement.LogHelper;

using System;

using System.Data.Entity.Infrastructure;

using System.IO;

using System.Linq;

[assembly: log4net.Config.XmlConfigurator(Watch = true)]

namespace DocsManagement.Models

{

public class EFAgreementDocsRepository : IAgreementDocsRepository

{

private static readonly log4net.ILog log = LogHellper.GetLogger();

private DocumentsDBEntities context = new DocumentsDBEntities();

public IQueryable<Agreement> AgreementDocuments => context.Agreements;

public void SaveAgreementDocuments(AgreementDocument agreementDocs, String File)

{

Executor executor = new Executor();

try

{

FileInfo fil = new FileInfo(File);

using (DocumentsDBEntities doc = new DocumentsDBEntities())

{

var agDoc = new Agreement()

{

RegistrationNomer = agreementDocs.RegistrationNomer,

RegistrationData = agreementDocs.RegistrationData,

TypeDocument = agreementDocs.TypeDocument,

StateDocument = agreementDocs.StateDocument,

TypeAgreement = agreementDocs.TypeAgreement,

DeadlineAgreement = agreementDocs.DeadlineAgreement,

Conractor = agreementDocs.Contractor,

Amount = agreementDocs.Amount,

CreatedUser = agreementDocs.CreatedUser,

SignedUser = agreementDocs.SignedUser,

NumberSheets = agreementDocs.NumberSheets,

Summary = agreementDocs.Summary,

FileName = Path.GetFileName(File),

FileType = fil.Extension,

FileContent = System.IO.File.ReadAllBytes(File)

};

doc.Agreements.Add(agDoc);

doc.SaveChanges();

}

}

catch (DbUpdateException ex)

{

log.Error("Can not use existing Registration nomer!", ex);

}

}

public void DeleteAgreementDocuments(int agreementDocsID)

{

Agreement doc = context.Agreements.Find(agreementDocsID);

if(doc != null)

{

context.Agreements.Remove(doc);

context.SaveChanges();

}

}

}

}

**Agreement.cs**

namespace DocsManagement.Models

{

using System;

using System.Collections.Generic;

public partial class Agreement

{

public int RegistrationNomer { get; set; }

public System.DateTime RegistrationData { get; set; }

public string TypeDocument { get; set; }

public string StateDocument { get; set; }

public string TypeAgreement { get; set; }

public System.DateTime DeadlineAgreement { get; set; }

public string Conractor { get; set; }

public decimal Amount { get; set; }

public string CreatedUser { get; set; }

public string SignedUser { get; set; }

public Nullable<int> NumberSheets { get; set; }

public string Summary { get; set; }

public string FileName { get; set; }

public string FileType { get; set; }

public byte[] FileContent { get; set; }

}

}

**Service.cs**

namespace DocsManagement.Models

{

using System;

using System.Collections.Generic;

public partial class Service

{

public int RegistrationNomer { get; set; }

public System.DateTime RegistrationData { get; set; }

public string TypeDocument { get; set; }

public string StateDocument { get; set; }

public string CreatedUser { get; set; }

public string SignedUser { get; set; }

public Nullable<int> NumberSheets { get; set; }

public string Summary { get; set; }

public string FileName { get; set; }

public string FileType { get; set; }

public byte[] FileContent { get; set; }

}

}

**Messages.cs**

using System.ComponentModel.DataAnnotations;

using System.Web;

namespace DocsManagement.Models

{

public class Messages

{

public string Sender { get; set; }

public string Reciever { get; set; }

public string Message1 { get; set; }

[Display(Name = "File")]

public HttpPostedFileBase File { get; set; }

}

}

**Executor.cs**

namespace DocsManagement.Models

{

using System;

using System.Collections.Generic;

public partial class Executor

{

public int ID { get; set; }

public string Name { get; set; }

public string Email { get; set; }

public string Phone { get; set; }

public string Position { get; set; }

}

}

**Prepared.cs**

namespace DocsManagement.Models

{

using System;

using System.Collections.Generic;

public partial class Prepared

{

public int ID { get; set; }

public string Name { get; set; }

public string Email { get; set; }

public string Phone { get; set; }

}

}

**DocumentsDBEntities.cs**

namespace DocsManagement.Models

{

using System;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

public partial class DocumentsDBEntities : DbContext

{

public DocumentsDBEntities()

: base("name=DocumentsDBEntities")

{

}

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

throw new UnintentionalCodeFirstException();

}

public virtual DbSet<Agreement> Agreements { get; set; }

public virtual DbSet<Executor> Executors { get; set; }

public virtual DbSet<Message> Messages { get; set; }

public virtual DbSet<Prepared> Prepareds { get; set; }

public virtual DbSet<Service> Services { get; set; }

}

}

**AgreementDocsController.cs**

using DocsManagement.Models;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace DocsManagement.Controllers

{

public class AgreementDocsController : Controller

{

private DocumentsDBEntities context;

EFAgreementDocsRepository ef;

public AgreementDocsController()

{

context = new DocumentsDBEntities();

ef = new EFAgreementDocsRepository();

}

// GET: AgreementDocs

[Authorize(Roles = "Employee, Admin, Manager")]

public ActionResult List()

{

return View(ef.AgreementDocuments);

}

// GET: AgreementDocs/Details/5

[Authorize(Roles = "Employee, Admin, Manager")]

public ActionResult Details(int id)

{

Agreement doc = context.Agreements.Find(id);

if(doc == null)

{

return HttpNotFound();

}

return View(doc);

}

[Authorize(Roles = "Employee")]

[HttpGet]

// GET: AgreementDocs/Create

public ActionResult Create()

{

//ViewBag.prepared = new SelectList(context.Prepareds.ToList(), "Name", "Name");

//ViewBag.executors = new SelectList(context.Executors.ToList(), "Name", "Name");

return View();

}

// POST: AgreementDocs/Create

[HttpPost]

public ActionResult Create(AgreementDocument agreementDocs)

{

if (ModelState.IsValid)

{

String File = SaveToPhysicalLocation(agreementDocs.File);

ef.SaveAgreementDocuments(agreementDocs, File);

return RedirectToAction("List");

}

return View(agreementDocs);

}

private string SaveToPhysicalLocation(HttpPostedFileBase file)

{

if (file.ContentLength > 0)

{

var fileName = Path.GetFileName(file.FileName);

var path = Path.Combine(Server.MapPath("~/App\_Data/UploadedFiles"), fileName);

file.SaveAs(path);

return path;

}

return string.Empty;

}

[HttpGet]

public FileResult Download(int id)

{

Agreement doc = context.Agreements.Find(id);

//string fullpath = Path.Combine(Server.MapPath("~/App\_Data/UploadedFiles"), doc.FileName);

//FileInfo file = new FileInfo(fullpath);

//String fileName = Path.GetFileName(fullpath);

//byte[] fileContent = System.IO.File.ReadAllBytes(fullpath);

//String ext = file.Extension;

//String contentType = MimeMapping.GetMimeMapping(fullpath);

//return File(fileContent, ext, fileName);

return File(doc.FileContent, doc.FileType, doc.FileName);

}

// GET: AgreementDocs/Edit/5

[Authorize(Roles = "Manager")]

public ActionResult Edit(int id)

{

Agreement doc = context.Agreements.Find(id);

if(doc == null)

{

return HttpNotFound();

}

return View(doc);

}

// POST: AgreementDocs/Edit/5

[HttpPost]

public ActionResult Edit(Agreement doc)

{

if (ModelState.IsValid)

{

context.Entry(doc).State = EntityState.Modified;

context.SaveChanges();

return RedirectToAction("List");

}

return View(doc);

}

[Authorize(Roles = "Manager")]

// GET: AgreementDocs/Delete/5

public ActionResult Delete(int id)

{

Agreement doc = context.Agreements.Find(id);

if(doc == null)

{

return HttpNotFound();

}

return View(doc);

}

// POST: AgreementDocs/Delete/5

[HttpPost, ActionName("Delete")]

public ActionResult DeleteConfirmed(int id)

{

ef.DeleteAgreementDocuments(id);

return RedirectToAction("List");

}

}

}

**ServiceDocsController.cs**

using DocsManagement.Models;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace DocsManagement.Controllers

{

public class ServiceDocsController : Controller

{

DocumentsDBEntities context = new DocumentsDBEntities();

EFServicesDocsRepository ef = new EFServicesDocsRepository();

// GET: AgreementDocs

[Authorize(Roles = "Employee, Admin, Manager")]

public ActionResult List()

{

return View(ef.ServicesDocuments);

}

// GET: AgreementDocs/Details/5

[Authorize(Roles = "Employee, Admin, Manager")]

public ActionResult Details(int id)

{

Service doc = context.Services.Find(id);

if (doc == null)

{

return HttpNotFound();

}

return View(doc);

}

// GET: AgreementDocs/Create

[Authorize(Roles = "Employee")]

public ActionResult Create()

{

return View();

}

// POST: AgreementDocs/Create

[HttpPost]

public ActionResult Create(ServiceDocument serviceDocs)

{

if (ModelState.IsValid)

{

String File = SaveToPhysicalLocation(serviceDocs.File);

ef.SaveServicesDocuments(serviceDocs, File);

return RedirectToAction("List");

}

return View(serviceDocs);

}

public string SaveToPhysicalLocation(HttpPostedFileBase file)

{

if (file.ContentLength > 0)

{

var fileName = Path.GetFileName(file.FileName);

var path = Path.Combine(Server.MapPath("~/App\_Data/ServiceDocFiles"), fileName);

file.SaveAs(path);

return path;

}

return string.Empty;

}

[HttpGet]

public FileResult Download(int id)

{

Service doc = context.Services.Find(id);

return File(doc.FileContent, doc.FileType, doc.FileName);

}

[Authorize(Roles = "Manager")]

public ActionResult Edit(int id)

{

Service doc = context.Services.Find(id);

if (doc == null)

{

return HttpNotFound();

}

return View(doc);

}

[HttpPost]

public ActionResult Edit(Service doc)

{

if (ModelState.IsValid)

{

context.Entry(doc).State = EntityState.Modified;

context.SaveChanges();

return RedirectToAction("List");

}

return View(doc);

}

[Authorize(Roles = "Manager")]

public ActionResult Delete(int id)

{

Service doc = context.Services.Find(id);

if (doc == null)

{

return HttpNotFound();

}

return View(doc);

}

[HttpPost, ActionName("Delete")]

public ActionResult DeleteConfirmed(int id)

{

ef.DeleteServicesDocuments(id);

return RedirectToAction("List");

}

}

}

**MessagesController.cs**

using DocsManagement.Models;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace DocsManagement.Controllers

{

public class MessagesController : Controller

{

DocumentsDBEntities context = new DocumentsDBEntities();

// GET: Messages

public ActionResult List()

{

return View(context.Messages);

}

// GET: Messages/Details/5

public ActionResult Details(int id)

{

Message post = context.Messages.Find(id);

if (post == null)

{

return HttpNotFound();

}

return View(post);

}

// GET: Messages/Create

public ActionResult Create()

{

return View();

}

// POST: Messages/Create

[HttpPost]

public ActionResult Create(Messages post)

{

if (ModelState.IsValid)

{

String File = SaveToPhysicalLocation(post.File);

FileInfo fil = new FileInfo(File);

using (DocumentsDBEntities entity = new DocumentsDBEntities())

{

var message = new Message()

{

Sender = post.Sender,

Reciever = post.Reciever,

Message1 = post.Message1,

FileName = Path.GetFileName(File),

FileType = fil.Extension,

FileContent = System.IO.File.ReadAllBytes(File)

};

entity.Messages.Add(message);

entity.SaveChanges();

}

return RedirectToAction("List");

}

return View(post);

}

private string SaveToPhysicalLocation(HttpPostedFileBase file)

{

if (file.ContentLength > 0)

{

var fileName = Path.GetFileName(file.FileName);

var path = Path.Combine(Server.MapPath("~/App\_Data/PostFiles"), fileName);

file.SaveAs(path);

return path;

}

return string.Empty;

}

[HttpGet]

public FileResult Download(int id)

{

Message doc = context.Messages.Find(id);

return File(doc.FileContent, doc.FileType, doc.FileName);

}

// GET: Messages/Delete/5

public ActionResult Delete(int id)

{

Message post = context.Messages.Find(id);

if (post == null)

{

return HttpNotFound();

}

return View(post);

}

// POST: Messages/Delete/5

[HttpPost, ActionName("Delete")]

public ActionResult DeleteConfirmed(int id)

{

Message post = context.Messages.Find(id);

if (post != null)

{

context.Messages.Remove(post);

context.SaveChanges();

}

return RedirectToAction("List");

}

}

}

**RoleController.cs**

using DocsManagement.Models;

using Microsoft.AspNet.Identity.EntityFramework;

using System.Linq;

using System.Web.Mvc;

namespace DocsManagement.Controllers

{

public class RoleController : Controller

{

ApplicationDbContext context;

public RoleController()

{

context = new ApplicationDbContext();

}

// GET: Role

public ActionResult Index()

{

var Roles = context.Roles.ToList();

return View(Roles);

}

public ActionResult Create()

{

var Role = new IdentityRole();

return View(Role);

}

[HttpPost]

public ActionResult Create(IdentityRole Role)

{

context.Roles.Add(Role);

context.SaveChanges();

return RedirectToAction("Index");

}

}

}