**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное учреждение высшего образования

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Институт кибернетики

Кафедра информатики и проектирования систем

Направление «Информатика и вычислительная техника»

Отчет по лабораторной работе №2 по дисциплине

**«Современные платформы программирования»**

Разработка многостраничного веб-сайта по шаблону MVC

Выполнил:

Студент группы 8ВМ81 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Т. Хусаинова

Проверил:

Доцент ОИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Н. Попов

Томск 2018

# Задание

Разработать многостраничный web-сайт на C# с использованием IDE Microsoft Visual Studio, шаблона проектирования и технологии MVC ASP.NET.

Необходимо:

* Реализовать веб-сайт со следующими страницами: административная страница для удаления файлов и их описаний с сайта; пользовательская страница для загрузки файлов и их описаний на сайт; пользовательская страница для поиска файла по ключевым словам описания и скачивания файла.
* Защитить все административные страницы сайта парольным доступом.
* Дополнительно создать на сайте страницу с описанием контакта на разработчика сайта («Об авторе» / «Контакты» и т.п.) и страницу с описанием принципа работы сайта («О сайте»).
* Реализовать на web-сайте возможность протоколирования действия пользователя (нажатия на ссылки, заполнения полей) в файле (текстовом, xml, json) и (при потребности) отображать их перечень на странице приложения в табличном виде (дата, время, действие).

**ASP.NET MVC**

Платформа ASP.NET MVC представляет собой фреймворк для создания сайтов и веб-приложений с помощью реализации паттерна MVC. Концепция паттерна (шаблона) MVC (model – view – controller) предполагает разделение приложения на три компонента:

* Контроллер (controller) представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления.
* Представление (view) – это собственно визуальная часть или пользовательский интерфейс приложения. Как правило, html-страница, которую пользователь видит, зайдя на сайт.
* Модель (model) представляет класс, описывающий логику используемых данных.

Общую схему взаимодействия этих компонентов можно представить следующим образом:



Рисунок 1 – Схема взаимодействия компонентов ASP.NET MVC

В этой схеме модель является независимым компонентом - любые изменения контроллера или представления не затрагивают модель. Контроллер и представление являются относительно независимыми компонентами, и нередко их можно изменять независимо друг от друга.

Благодаря этому реализуется концепция разделение ответственности, в связи с чем легче построить работу над отдельными компонентами. Кроме того, вследствие этого приложение обладает лучшей тестируемостью. И если, к примеру, важна визуальная часть или фронтэнд, MVC позволяет тестировать представление независимо от контроллера. Либо можно сосредоточиться на бэкэнде и тестировать контроллер.

# Листинг

**Model**

using System.Data.Entity;

namespace WebApplication3.Models

{

public class Files

{

public int ID { get; set; }

public string Name { get; set; }

public string Path { get; set; }

public string Type { get; set; }

public string Description { get; set; }

}

public class FilesDbContext : DbContext

{

public DbSet<Files> Files { get; set; }

}

}

**Controller**

using System;

using System.Data.Entity;

using System.Linq;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

using WebApplication3.Models;

using NLog;

namespace WebApplication3.Controllers

{

public class FilesController : Controller

{

private FilesDbContext db = new FilesDbContext();

private Logger logger = LogManager.GetCurrentClassLogger();

public ActionResult Index()

{

logger.Info("Got List of available files");

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

}

[HttpPost]

public ActionResult Upload(HttpPostedFileBase upload, string description)

{

if (upload != null)

{

string fileName = System.IO.Path.GetFileName(upload.FileName);

var path = Server.MapPath("~/Media/" + fileName);

upload.SaveAs(path);

var model = new Files

{

Name = fileName,

Path = path,

Type = VirtualPathUtility.GetExtension(path),

Description = description

};

db.Files.Add(model);

db.SaveChanges();

logger.Info("A new file was uploaded: {0}", fileName);

}

return RedirectToAction("Index");

}

public FilePathResult DownloadFile(int id)

{

Files file = db.Files.Find(id);

logger.Info("A file was downloaded: {0}", file.Name);

return File(file.Path, file.Type, file.Name);

}

public async Task<ActionResult> Search(string searchInput)

{

var file = from f in db.Files

select f;

if (!String.IsNullOrEmpty(searchInput))

{

file = file.Where(s => s.Description.Contains(searchInput));

}

logger.Info("Searched by description: {0}", searchInput);

return View(await file.ToListAsync());

}

public ActionResult toDelete(int id)

{

Files file = db.Files.Find(id);

if (System.IO.File.Exists(file.Path))

{

System.IO.File.Delete(file.Path);

}

logger.Info("A file was deleted: {0}", file.Name);

db.Files.Remove(file);

db.SaveChanges();

return RedirectToAction("toRemove");

}

public ActionResult toRemove()

{

logger.Info("Got List of files to delete");

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

}

}

}

**View (Index)**

@model IEnumerable<WebApplication3.Models.Files>

@{

ViewBag.Title = "Index";

}

<h2>Загрузка и скачивание файлов</h2>

<div class="container" style="margin-top: 30px;">

@using (Html.BeginForm("Upload", "Files", FormMethod.Post, new { enctype = "multipart/form-data" }))

{

<div class="row">

<div class="col-md-4">

<p style="text-align: right; font-weight: bold;">Файл</p>

</div>

<div class="col-md-8">

<input type="file" name="upload" value="Выберите файл" />

</div>

</div>

<div class="row" style="margin-bottom: 5px;">

<div class="col-md-4">

<p style="text-align: right; font-weight: bold;">Описание</p>

</div>

<div class="col-md-8">

<input type="text" name="description" style="max-width: 780px; width: 100%;" />

</div>

</div>

<div class="row">

<div class="col-md-4">

</div>

<div class="col-md-8">

<input type="submit" value="Загрузить" />

</div>

</div>

<div class="row" style="margin-top: 30px;">

<table class="table">

<thead>

<tr>

<th class="col-md-4">Имя файла</th>

<th class="col-md-6">Описание</th>

<th class="col-md-2">Действие</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>@item.Name</td>

<td>@item.Description</td>

<td>@Html.ActionLink("Скачать", "DownloadFile", new { id = item.ID })</td>

</tr>

}

</tbody>

</table>

</div>

}

</div>

**View (Search)**

@model IEnumerable<WebApplication3.Models.Files>

@{

ViewBag.Title = "Search";

}

<h2>Поиск файлов по описанию</h2>

<div class="container" style="margin-top: 30px;">

<form asp-controller="Files" asp-action="Search">

<div class="row" style="margin-bottom: 5px;">

<div class="col-md-4">

<p style="text-align: right; font-weight: bold;">Поиск</p>

</div>

<div class="col-md-8">

<input type="text" name="searchInput" style="max-width: 780px; width: 100%;" />

</div>

</div>

<div class="row">

<div class="col-md-4">

</div>

<div class="col-md-8">

<input type="submit" value="Найти" />

</div>

</div>

</form>

<div class="row" style="margin-top: 30px;">

<table class="table">

<thead>

<tr>

<th class="col-md-4">Имя файла</th>

<th class="col-md-6">Описание</th>

<th class="col-md-2">Действие</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>@item.Name</td>

<td>@item.Description</td>

<td>@Html.ActionLink("Скачать", "DownloadFile", new { id = item.ID })</td>

</tr>

}

</tbody>

</table>

</div>

</div>

**View (toRemove)**

@model IEnumerable<WebApplication3.Models.Files>

@{

ViewBag.Title = "toRemove";

}

<h2>Поиск файлов по описанию</h2>

<div class="container" style="margin-top: 30px;">

<div class="row" style="margin-top: 30px;">

<table class="table">

<thead>

<tr>

<th class="col-md-4">Имя файла</th>

<th class="col-md-6">Описание</th>

<th class="col-md-2">Действие</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>@item.Name</td>

<td>@item.Description</td>

<td>@Html.ActionLink("Удалить", "toDelete", new { id = item.ID })</td>

</tr>

}

</tbody>

</table>

</div>

</div>

# Результаты работы

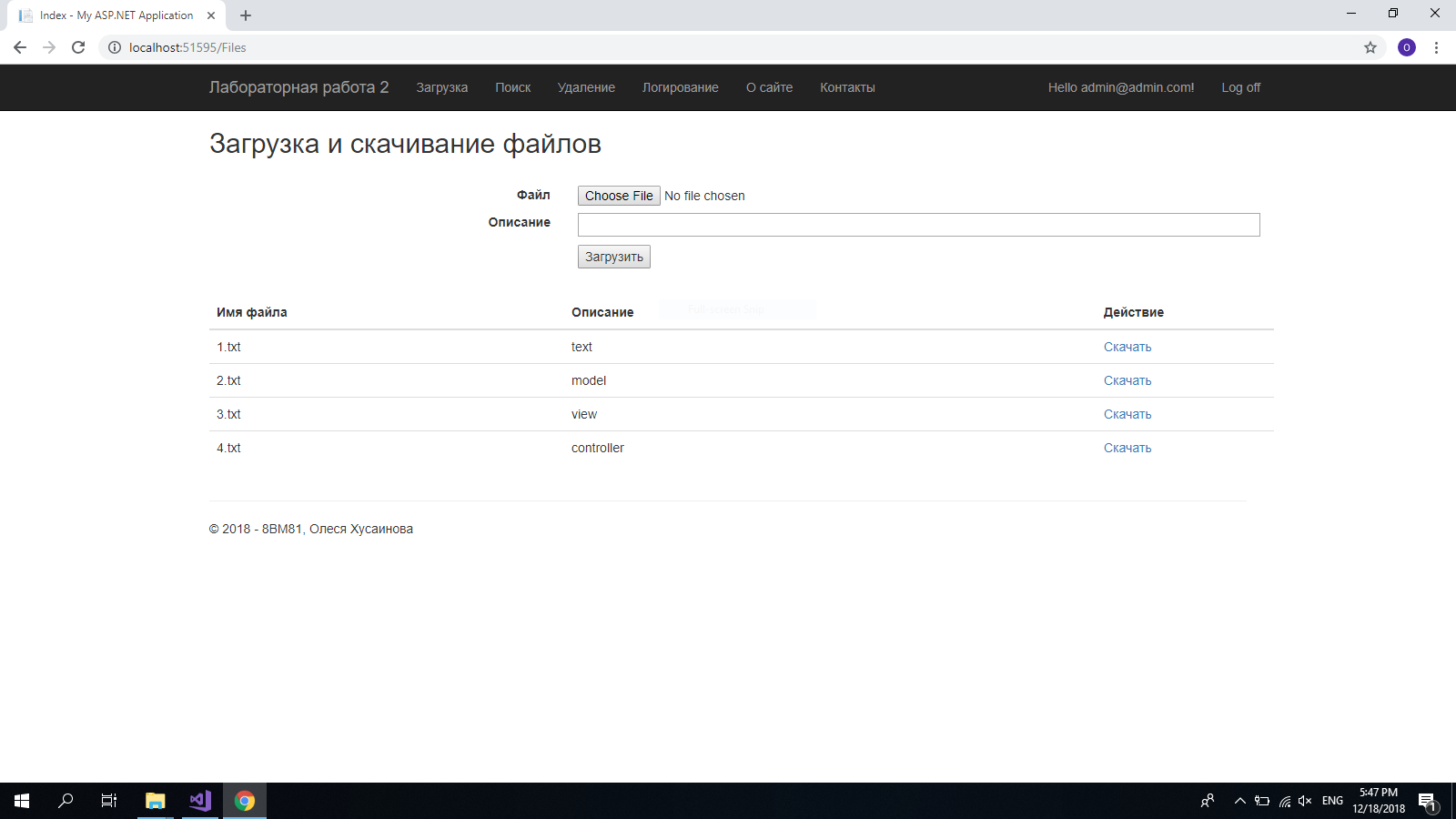


Рисунок 1 – Загрузка и скачивание файлов

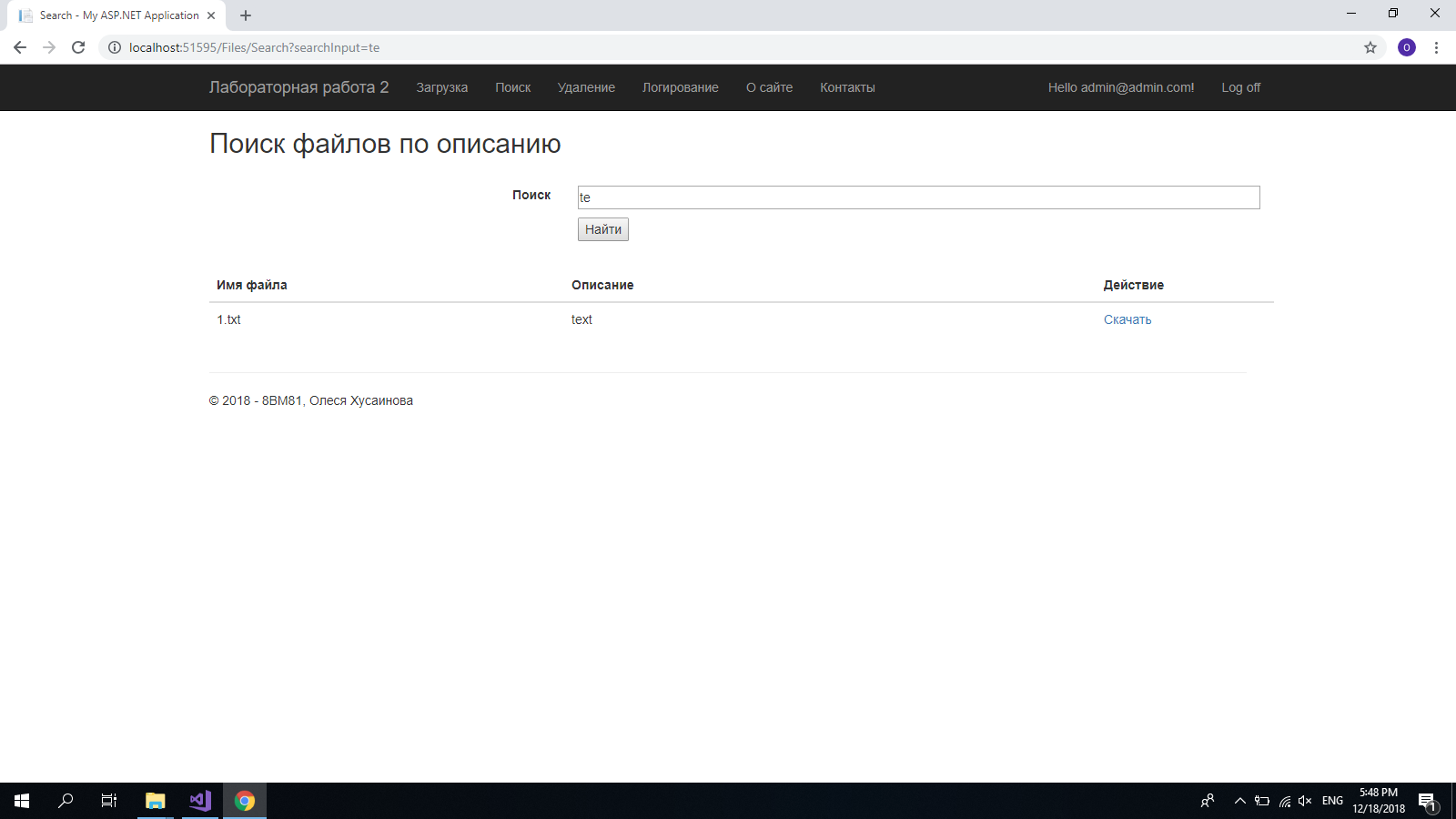


Рисунок 2 – Поиск файлов

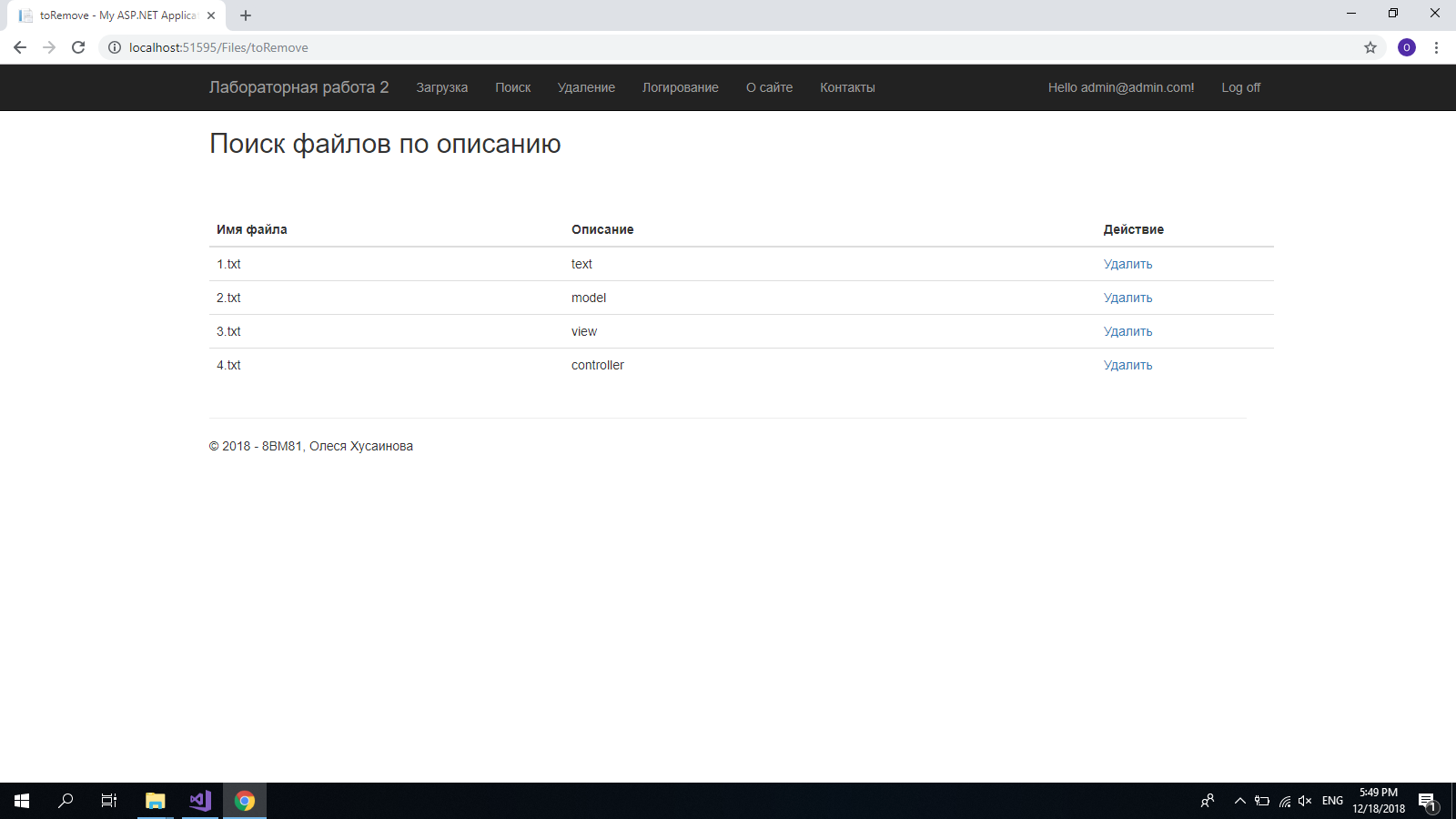


Рисунок 3 – Удаление файлов

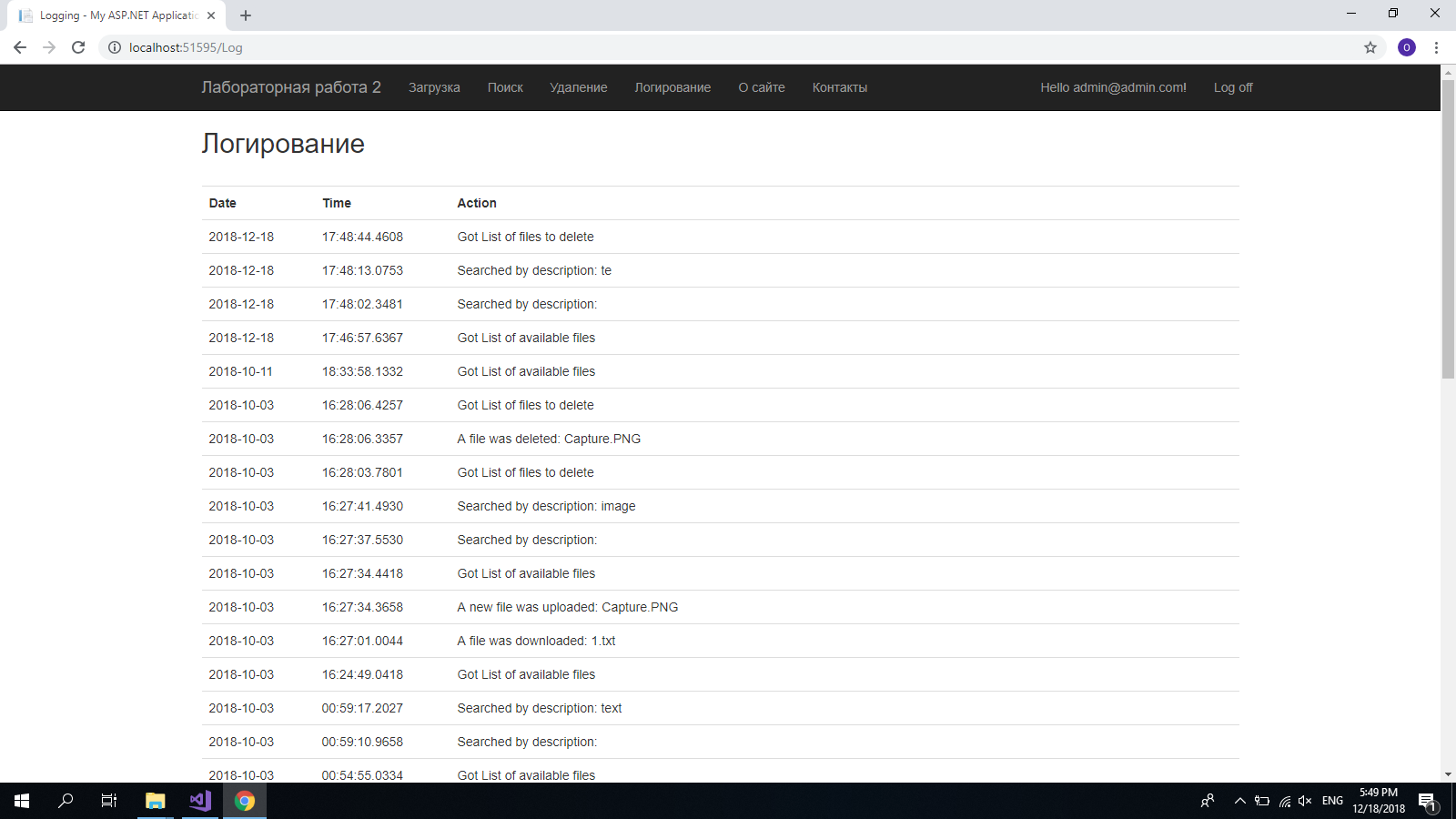


Рисунок 4 – Логирование действий

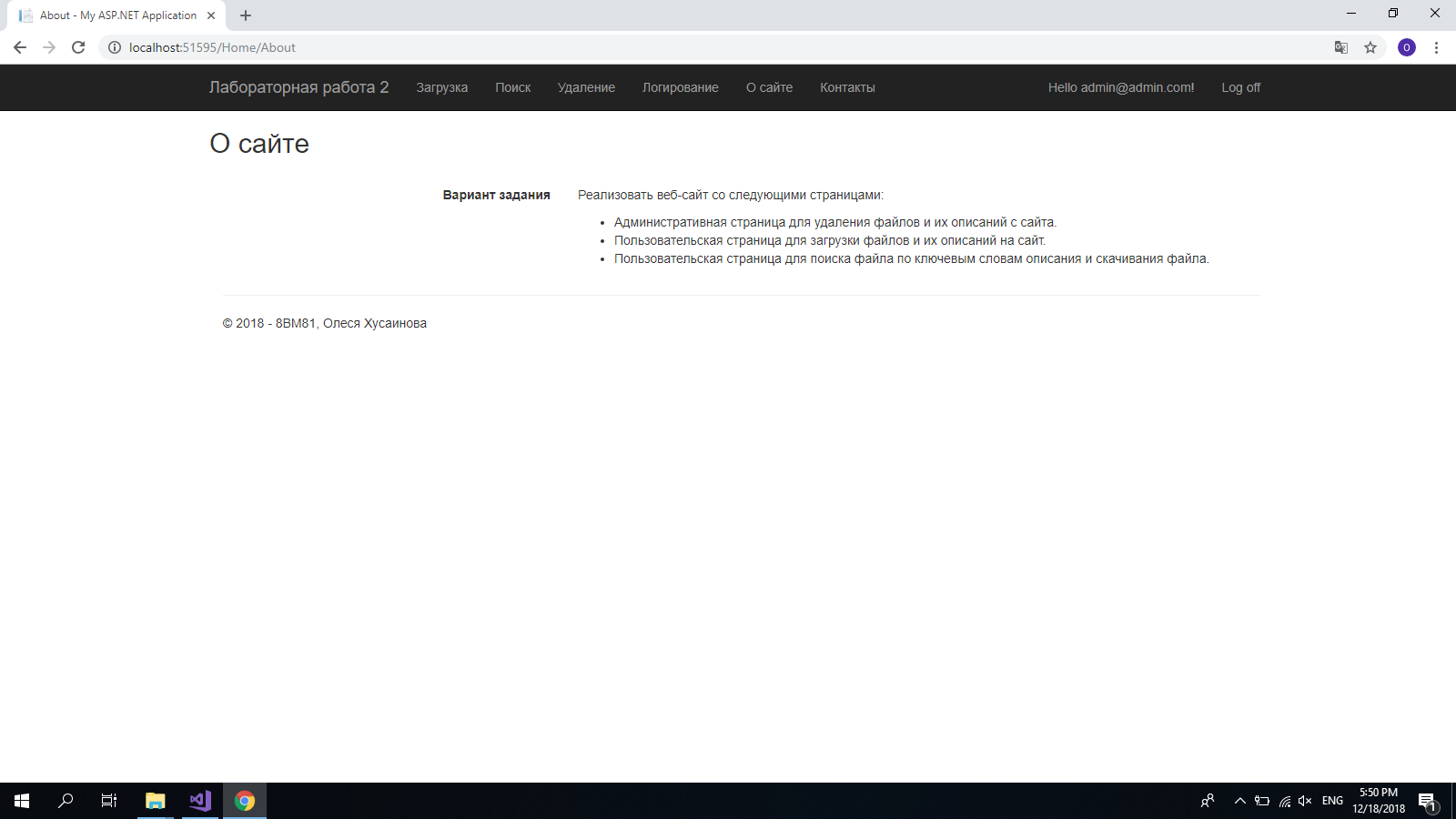


Рисунок 5 – О сайте

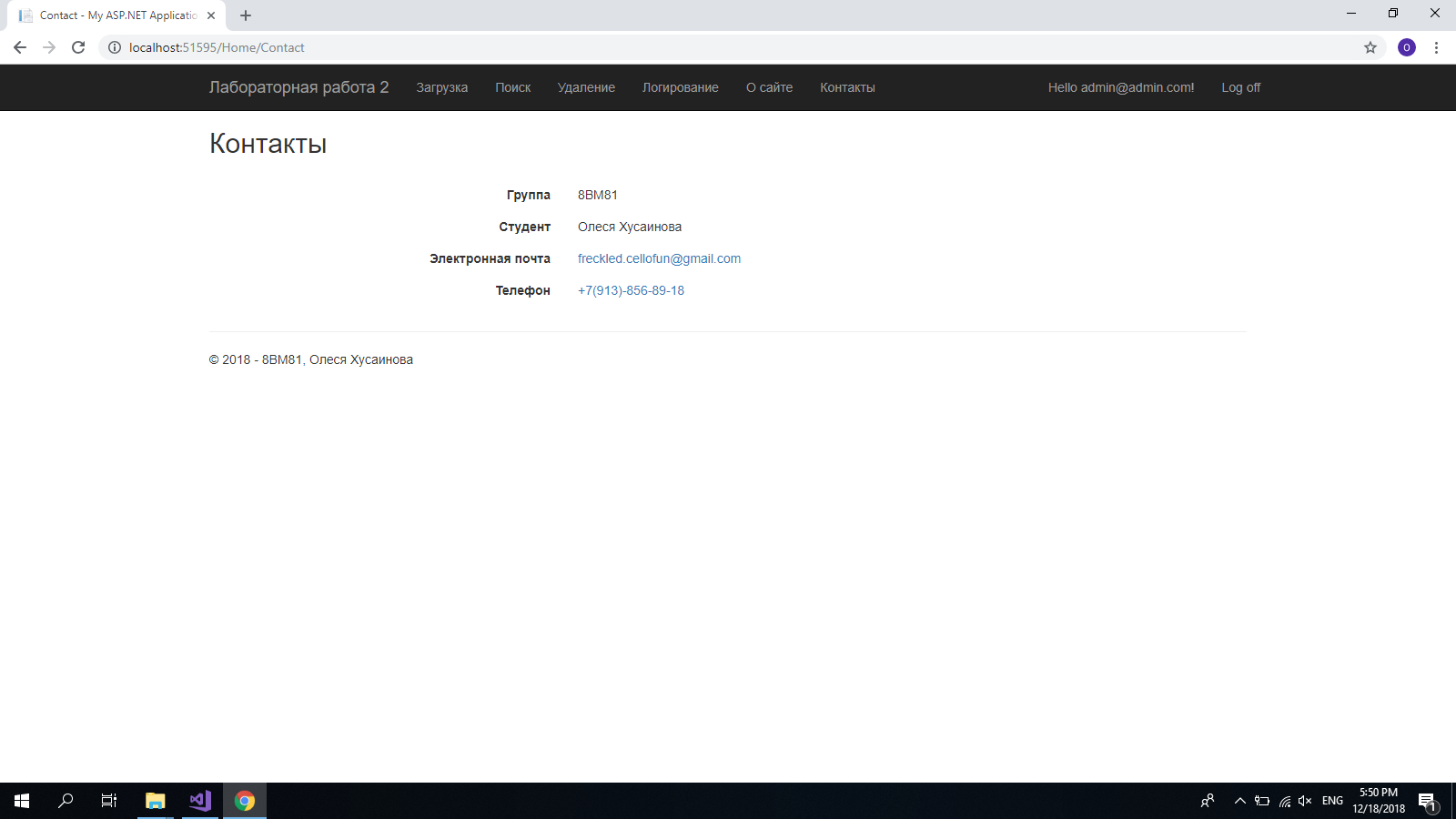


Рисунок 6 – Контакты

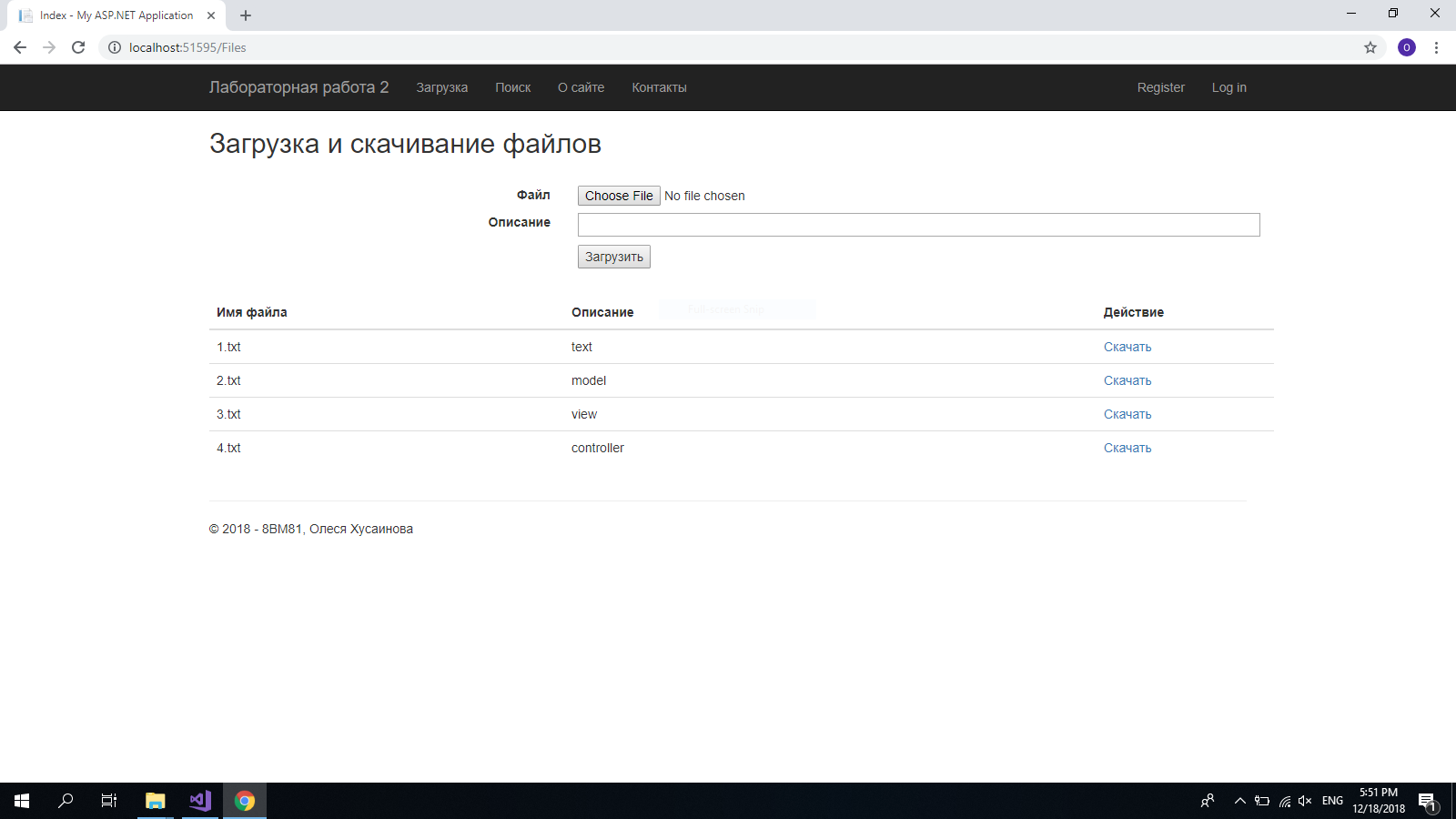


Рисунок 7 – Страницы, доступные для обычных пользователей

# Выводы

На основе Microsoft ASP.Net MVC был реализован веб-сайт со следующими страницами:

* административная страница для удаления файлов и их описаний с сайта;
* пользовательская страница для загрузки файлов и их описаний на сайт;
* пользовательская страница для поиска файла по ключевым словам описания и скачивания файла.

Все основные действия пользователя логируются на дополнительной странице сайта.