МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Минский КОЛЛЕДЖ предпринимательства

Отделение «Программное обеспечение информационных технологий»

Пояснительная записка

к курсовому проекту

на тему

**ВЕБ-ПРИЛОЖЕНИЕ «АБИТУРИЕНТ»**

МКП КП 2-40 01 01 33 006 ПЗ

**Автор проекта**

Учащегося 4 курса группы П-41 /Д.А. Ефименко/

**Руководитель**

Преподаватель /И.Ю. Дроздова/

Минск, 2018

**МИНСКИЙ КОЛЛЕДЖ ПРЕДПРИНИМАТЕЛЬСТВА**

Отделение

«Программное обеспечение информационных технологий»

Специальность

«Программное обеспечение информационных технологий» 2-40 01 01

УТВЕРЖДАЮ

Председатель цикловой комиссии

\_\_\_\_\_\_\_\_\_\_/ \_Ю.А. Доманова\_\_\_\_\_\_\_\_\_\_\_\_/

**ЗАДАНИЕ**

**на курсовой проект**

Ефименко Даниилу Андреевичу

Учащемуся\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4

П-41

курса\_\_\_\_\_\_\_\_ группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Базы данных и системы управления базами данных

по учебной дисциплине \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Веб-приложение «Абитуриент»

Тема курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

логин, пароль, добавление данных в таблицы

Исходные данные: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Состав проекта**

|  |  |
| --- | --- |
| **Пояснительная записка** | |
| Содержание разделов | Срок выполнения |
| Введение | 08.10 – 15.10 |
| Постановка задачи | 16.10 – 22.10 |
| Вычислительная система | 23.10 – 29.10 |
| Проектирование | 30.10 – 12.11 |
| Описание программы | 13.11 – 26.11 |
| Отладка и испытание программы | 27.11 – 03.12 |
| Описание применения | 04.12 – 10.12 |
| Заключение | 11.12 – 17.12 |
| **Графическая часть проекта** | |
| ER-диаграмма - плакат | 18.12 – 20.12 |
| Диаграмма вариантов использования - плакат | 21.12 – 24.12 |

Дата выдачи « 08 »\_октября\_\_2018 г.

Срок завершения « 24 » декабря 2018 г.

**Преподаватель-руководитель курсового проекта** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись учащегося**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ**

*y*

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

МКП КП 2-40 01 01 33 006 ПЗ

Разраб.

Ефименко

Пров.

*Дроздова*

Реценз.

Н. контр.

Утв.

*Веб-приложение   
«Абитуриент»*

Лит.

Листов

49

МКП

[Введение 4](#_Toc532807863)

[1 Постановка задачи 6](#_Toc532807864)

[1.1 Формулировка задания 6](#_Toc532807865)

[1.2 Обоснование необходимости автоматизации 7](#_Toc532807866)

[1.3 Определение данных и их представление 7](#_Toc532807867)

[2 Вычислительная система 9](#_Toc532807868)

[2.1 Обоснование языка программирования 9](#_Toc532807869)

[2.2 Обоснование выбора среды разработки 10](#_Toc532807870)

[2.3 Требования к конфигурации программного и аппаратного обеспечения 11](#_Toc532807871)

[3 Проектирование 13](#_Toc532807872)

[3.1 Проектирование интерфейса 13](#_Toc532807873)

[3.2 Инфологическая модель базы данных 13](#_Toc532807874)

[3.3 Логическая модель базы данных 15](#_Toc532807875)

[3.4 Физическая модель базы данных 18](#_Toc532807876)

[4 Описание программы 21](#_Toc532807877)

[4.1 Логическая структура 21](#_Toc532807878)

[4.2 Физическая структура 21](#_Toc532807879)

[5 Отладка и испытания программы 22](#_Toc532807880)

[5.1 Тестовые примеры 22](#_Toc532807881)

[5.2 Анализ полученных результатов 23](#_Toc532807882)

[6 Описание применения программы 26](#_Toc532807883)

[6.1 Назначение программы и условия применения 26](#_Toc532807884)

[6.2 Описание работы программы 26](#_Toc532807885)

[Заключение 29](#_Toc532807886)

[Список используемых источников 30](#_Toc532807887)

[Приложение А (обязательное) Листинг программы 31](#_Toc532807888)

# ВВЕДЕНИЕ

В данном курсовом проекте будет создано информационное веб-приложение для получения и поиска информации об учебных заведениях.

Веб-приложение – клиент-серверное приложение, в котором клиент взаимодействует с сервером при помощи браузера, а за сервер отвечает веб-сервер. Логика веб-приложение распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами.

Базой данных называется совокупность самостоятельных материалов, систематизированных таким образом, чтобы их можно было легко найти и обработать с помощью электронной вычислительной машины (ЭВМ).

Реляционная модель данных (РМД) – логическая модель данных, прикладная теория построения баз данных, которая является приложением к задачам обработки данных таких разделов математики, как теория множеств и логика первого порядка.

Целью выполнения курсового проекта является расширение и углубление знаний в области баз данных и систем управления базами данных, а также языка запросов SQL.

Задачи выполнения курсового проекта:

* систематизация научных знаний;
* углубление уровня и расширение объема профессионально значимых знаний, умений и навыков;
* формирование умений и навыков самостоятельной организации научно-исследовательской работы;
* изучение технологии работы с базами данных;
* овладение современными методами поиска, обработки и использования информации.

Данная пояснительная записка содержит следующие разделы:

1. Раздел «Постановка задачи» содержит постановку задачи с указанием всех функций проекта, описание входных и выходных данных.
2. Раздел «Вычислительная система» содержит обоснования выбора языка программирования и среды разработки, описание операционной системы и требований к конфигурации программного и аппаратного обеспечения.
3. В разделе «Проектирование» приводится описание внешнего пользовательского интерфейса, описание алгоритма программы и структуры справочной системы.
4. Раздел «Описание программы» содержит описание всех модулей и подпрограмм, входящих в приложение. Здесь же описывается интерфейс программы.
5. В разделе «Отладка и испытание программы» описывается проверка работы программы на различных тестах и описание реакции программы на тесты.
6. Раздел «Описание применения программы» содержит сведения о назначении программного средства области применения и классе решаемых задач. Кроме того, приводятся требования к необходимым техническим и программным ресурсам.
7. В «Заключении» описывается краткая формулировка проблемы и пути её решения, использованные методы и средства, возможность дальнейшей модификации проекта.

# Постановка задачи

## Формулировка задания

В данном курсовом проекте необходимо разработать веб-приложение «Абитуриент». Приложение должно помочь абитуриентам в поиске желаемого учебного заведения по различным параметрам. Реализовать возможность выбора нескольких вариантов поиска.

Для поиска предусмотреть следующие критерии:

* город;
* специальность;
* балл;
* какие экзамены сданы;
* стоимость обучения;
* количество мест (платных, на бюджет);
* тип учебного заведения;
* форма обучения (дневная, вечерняя, заочная, сокращенная);
* платное/бюджет;
* обязательно ли общежитие.

Так же необходимо визуализировать результаты поиска в виде карты с расположением учебных заведений. Учебные заведения будут отображаться на карте в виде маркеров, по нажатию на которые во всплывающем окне будет отображаться краткая информация о данном заведении. В этом же окне реализовать возможность перехода на страницу с полной информацией об учебном заведении.

Страница с полной информацией должна содержать краткое и полное название учебного заведения и его описание. Должна присутствовать информация о факультетах, специальностях и требований для поступления. Так же реализовать возможность оставлять и просматривать комментарии к учебному заведению.

Также реализовать регистрацию и аутентификацию пользователя на сайте.

## Обоснование необходимости автоматизации

В настоящее время, с развитием информационных технологий, появилась возможность простого и быстрого получения практически любой информации в сети Интернет.

Поиск и получение информации об учебных заведениях имеет большую потенциальную клиентскую базу, что, в свою очередь, требует от сервиса возможность обработки больших массивов данных и быстрого отклика.

Большинство аналогов реализуют возможность фильтрации и просмотра информации об учебных заведениях, факультетах и специальностях.

Главным аналогом является сайт adukar.by. На данном сайте присутствует каталог из учебных заведений, которые можно фильтровать по различным параметрам, а полная информация о конкретном заведении доступна на отдельной странице.

Данный проект будет отличаться от аналогов тем, что каталог учебных заведений будет представлен в виде карты с маркерами. Будет присутствовать возможность оставить свой отзыв о заведении и поставить ему оценку. Также будет реализована функция предложения нового учебного заведения с последующим рассмотрением этого предложения администрацией сайта.

## Определение данных и их представление

Входные данные – это величины, которые задаются до начала работы алгоритма или определяются динамически во время его работы. Входными данными в курсовом проекте являются: пароль, логин, действия пользователя, его отзывы и оценки, настройка профиля, запросы на добавление новых и изменение уже существующих учебных заведений.

Выходные данные это то, что отобразится, или то, что мы получим в результате манипуляций при работе с программой. Выходными данными в курсовом проекте являются: страницы с информацией об учебных заведениях, главная страница, отзывы и оценки пользователей.

# Вычислительная система

## Обоснование языка программирования

Данный проект написан на языке программирования C#.

Язык программирования C# – объектно-ориентированный язык программирования с C-подобным синтаксисом, один из наиболее новых и быстро развивающихся языков программирования. Был разработан в компании Microsoft в 1998–2001 годах. [1, с.26]

Основная область использования – написание прикладного программного обеспечения.

Первоначально C# был ориентирован только на Microsoft Windows на платформе Microsoft .NET Framework, но в настоящее время позволяет строить собственные приложения на любой машине, где работает версия CLR и FCL, соответствующая стандарту ECMA: с архитектурой x86, x64, IA64 и т. д. [2, c.16]

К достоинствам C#, как языка программирования, относится:

* интерфейсная часть приложения разрабатывается очень быстро;
* синтаксический сахар, множество встроенных типов данных и функций;
* масштабируемость за счет интегрирования новых элементов и инструментов в данную среду;
* можно разрабатывать собственные компоненты и инструменты средствами самого C#;
* эргономичная проработка всей иерархии объектов;
* имеется возможность подключать сторонние визуальные компоненты в большом количестве;
* богатый инструментарий по отладке на всех уровнях;
* есть возможность работать над проектом сообща, в коллектив.

## Обоснование выбора среды разработки

Данный курсовой проект разработан в среде Microsoft Visual Studio 2017.

Среда программирования Visual Studio рассчитана на программирование различных приложений и предоставляет большое количество компонентов для этого. К тому же работодателей интересует, прежде всего, кроссплатформенность и качество создания программ, а эти характеристики может обеспечить только среда визуального проектирования, способная взять на себя значительные объемы рутинной работы по подготовке приложений, а также согласовать деятельность группы постановщиков, кодировщиков, тестеров и технических писателей. Возможности Visual Studio полностью отвечают подобным требованиям и подходят для создания систем любой сложности.

Преимущества Visual Studio по сравнению с аналогичными программными продуктами:

* быстрота разработки приложения;
* огромное количество инструментов для разработки ПО;
* наращиваемость за счет встраивания новых компонент и инструментов в среду Visual Studio;
* возможность разработки новых компонентов и инструментов собственными средствами Visual Studio (существующие компоненты и инструменты доступны в исходных кодах);
* удачная проработка иерархии объектов.
* используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Архитектура MySQL очень отличается от архитектур иных серверов баз данных, что делает эту СУБД полезной для одних целей, но одновременно неудачным выбором для других. MySQL неидеальна, но достаточно гибка для того, чтобы хорошо работать в очень требовательных средах, например, в веб-приложениях. В то же время MySQL позволяет применять встраиваемые приложения, хранилища данных, индексирование содержимого, программное обеспечение для доставки, высоконадежные системы с резервированием, обработку транзакций в реальном времени (OLTP) и многое другое. [3, c.29]

## Требования к конфигурации программного и аппаратного обеспечения

Данный курсовой проект был реализован в операционной системе Windows 10.

Технические характеристики ОС:

1) Процессор: не менее 1 ГГц или SoC.

2) ОЗУ: 1 ГБ (для 32-разрядных систем) или 2 ГБ (для 64-разрядных систем).

3) Место на жестком диске: 16 ГБ (для 32-разрядных систем) или 20 ГБ (для 64-разрядных систем).

4) Видеокарта: DirectX версии 9 или выше с драйвером WDDM 1.0.

5) Дисплей: 1024 x 600 точек.

Достоинства Windows 10, по сравнению с другими ОС в семействе Windows:

1) Возможность использования виртуальных рабочих столов.

2) Прикрепление окон.

3) Менеджер фалов.

4) Улучшенная командная строка.

5) Просмотр задач.

6) Доработки проводника.

7) Универсальный поиск.

Недостатки ОС:

1) Система сбора данных (ОС отправляет огромное количество сведений о пользователе Microsoft, такие как местоположение, контакты и частоту разговоров, данные электронной почты и другое).

2) Сложность установки или частые проблемы в ходе работы с некоторыми приложениями.

Минимальные требования к аппаратным и операционным (программным) ресурсам:

* 2 ГБ оперативной памяти;
* Microsoft .NET Framework 4.6 и выше;
* от 3 ГБ свободного места на жестком диске;
* базовый графический процессор любого поставщика, класса DirectX 9.0 или выше (Pixel Shader 2.0);
* процессор Intel Pentium или совместимый, с частотой не менее 1,6 ГГц (рекомендуется 2 ГГц и выше);
* монитор с разрешением 1024×768 или выше;

мышь или другое указывающее устройство.

# Проектирование

## Проектирование интерфейса

Пользовательский интерфейс – интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы (ISO/IEC/IEEE 24765-2010).

Веб-приложение «Художник» будет выполнено в дизайне Material Design с цветами: белый, зеленый, розовый.

Шапка сайта будет содержать название приложения и кнопку входа в учетную запись справа.

Главная страница будет содержать карту с маркерами и фильтр по учебным заведениям.

Страница учебного заведения будет содержать короткое, полное название и фото здания. Будет присутствовать информация о специальностях в виде таблицы. Будет присутствовать описание заведения и контактные данные. Также будет присутствовать карта, на которой будут отображены маркерами места нахождения корпусов и общежитий заведения. В конце страницы будет располагаться блок

Страница авторизации и регистрации будет содержать поля для ввода данных и кнопки входа и регистрации.

## Инфологическая модель базы данных

Необходимо спроектировать базу данных для хранения, создания и отображения информации о рисунках и пользователях.

В базе данных необходимо хранить информацию о рисунках, пользователях, комментариев, лайков. Также необходимо хранить информацию о комнатах.

Главный субъект базы данных – авторизированный пользователь, который является автором комментариев, оценки и лайков. Пользователь будет иметь стандартные поля для авторизации на сайте, также дополнительные сведения о том, как он будет отображаться на сайте (его аватар, имя, дата создания).

Главный объект данного сайта, над которым работает пользователь – рисунок.

Комната – сущность, которая будет хранить в себе изображение, которое смогут редактировать все желающие.

Комментарий – способ оценки рисунок, должен хранить текст, пользователя.

Лайки - способ оценки комментария, должен хранить комментарий, пользователя

При проектировании базы данных необходимо предусмотреть следующее:

* логин пользователя должны быть уникальным;
* у одного отзыва один автор.

Таким образом, база данных содержит такие сущности, как Учебные заведения, Факультеты, Специальности, Отзывы, Пользователи, Фото, Корпусы, Общежития.

Учебное заведение содержит связи с Отзывами, Факультетами и Фото, так как эти поля применяются к конкретному заведению.

Факультет содержит связь с Специальностями, так как это поле применяется к конкретному факультету множество раз.

## Логическая модель базы данных

База данных должна содержать 5 сущностей: «Пользователь», «Рисунок», «Комната», «Лайк», «Комментарий»

Сущность «Пользователь», предназначенная для хранения информации о пользователе, представлена в таблице 3.1.

Таблица 3.1 – «Пользователь»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| UserId | Целое число | Id пользователя |
| UserName | Текст | Имя пользователя |
| UserPassword | Текст | Пароль пользователя |
| UserPhotoPath | Текст | Путь к аватарке |

Сущность «Рисунок», предназначенная для хранения рисунков, и данных к ним, представлена в таблице 3.2.

Таблица 3.2 – «Рисунок»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| SketchId | Целое число | Id рисунка |
| SketchName | Текст | Имя рисунка |
| SketchPhoto | Текст | Картинка рисунка |
| SketchDate | Текст | Дата создания |
| SketchCategory | Текст | Категория |
| UserId | Целое число | Путь к пользователю |

Сущность «Комментарий», предназначенная для хранения комментариев пользователей, представлена в таблице 3.3.

Таблица 3.3 – «Коментарий»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| CommentId | Целое число | Id комментария |
| CommentDescription | Текст | Содержание |
| CommentTrueFalse | Целое число | Значение комментария |
| UserId | Целое число | Путь к пользователю |
| SketchId | Целое число | Путь к рисунку |

Сущность «Лайк», предназначенная для хранения лайков, представлена в таблице 3.4.

Таблица 3.4 – «Лайк»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| LikesId | Целое число | Id лайка |
| CommentId | Целое число | Путь к комментарию |
| UserId | Целое число | Путь к пользователю |

Сущность «Комната», предназначенная для хранения комнат рисования, представлена в таблице 3.5.

Таблица 3.5 – «Комнтата»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Id | Целое число | Id комнаты |
| Name | Текст | Имя комнаты |
| Image | Текст | Изображение комнаты |
| UserId | Целое число | Путь к пользователю |

## Физическая модель базы данных

База данных проектируется в MySQL.

Создание таблиц «Пользователи», «Рисунки», «Комнаты», «Лайки», «Комментарии» представлено на рисунке 3.1.

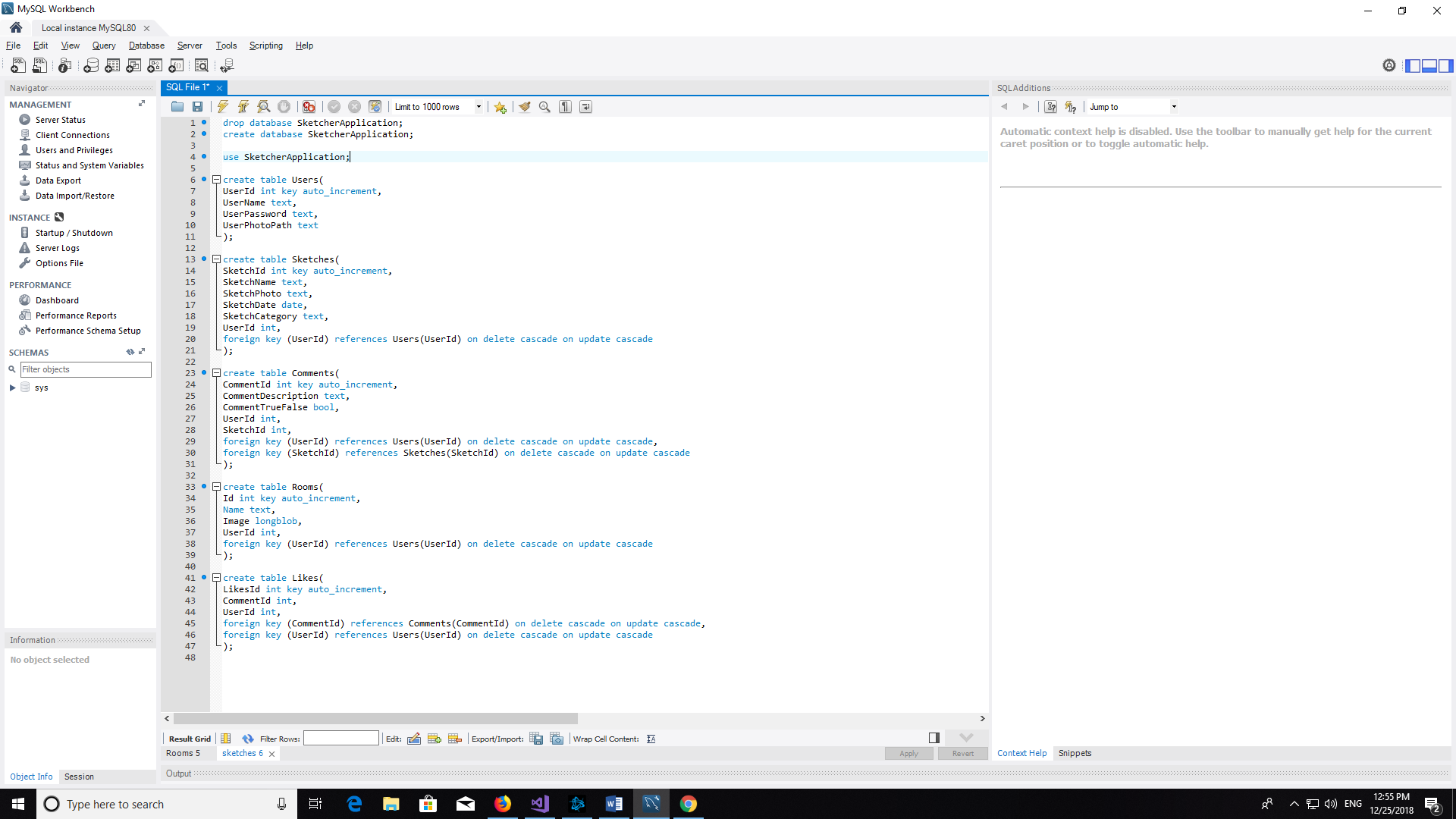


Рисунок 3.1 – Создание таблиц «Учебное заведение», «Общежития», «Корпусы»

# Описание программы

## Логическая структура

Данный разрабатываемый проект представлен следующей логической структурой:

HomeController:

1. Index – отображение главной страницы
2. News – отображение страницы новостей

ApplicationUserController:

1. Index – отображение всех пользователей
2. ShowUserPage – отображение страницы пользователя
3. Create – регистрация пользователя
4. Authorize – авторизация пользователя

SketchController:

1. Сreate – создание рисунка
2. ShowSketch – отображение рисунка

RoomController:

1. Index – отображение всех комнат
2. Create – создание комнаты

## Физическая структура

Физическая структура представлена следующими классами:

1. Users – определяет модель пользователя.
2. Sketches – определяет модель рисунка.
3. Comments – определяет модель комментария.
4. Rooms – определяет модель комнаты.

# Отладка и испытания программы

## Тестовые примеры

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий две различные цели:

1. Продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям.
2. Выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Тест – это последовательность одного или более подтестов, выполняемых последовательно. [4, c. 27-29]

Тест считается пройденным, если при корректно заданных критериях соответствия после его прохождения фактический итог совпадает с предсказанным и отсутствует симптом.

Тест считается не пройденным, если при его прохождении физический итог не совпадает с предсказанным и/или присутствует симптом. Причиной этого может быть использование неверного объекта, неверный ввод или прогнозируемый итог, неверное начальное состояние или критерии соответствия, некорректное применение этих критериев, ошибка при проектировании теста, выполнении теста, ошибка в процессе проверки соответствия и даже, каким бы это не могло показаться странным, ошибка в тестируемом объекте.

Тесты:

1. Попытка входа под несуществующим пользователем
2. Попытка входа под существующим пользователем
3. Попытка регистрации с незаполненными полями
4. Попытка регистрации с уже существующим логином
5. Попытка регистрации с неверным форматом почты

## Анализ полученных результатов

1. Пользователю выведется сообщение о том, что логин или пароль были введены неверно, представлено на рисунке 5.1.

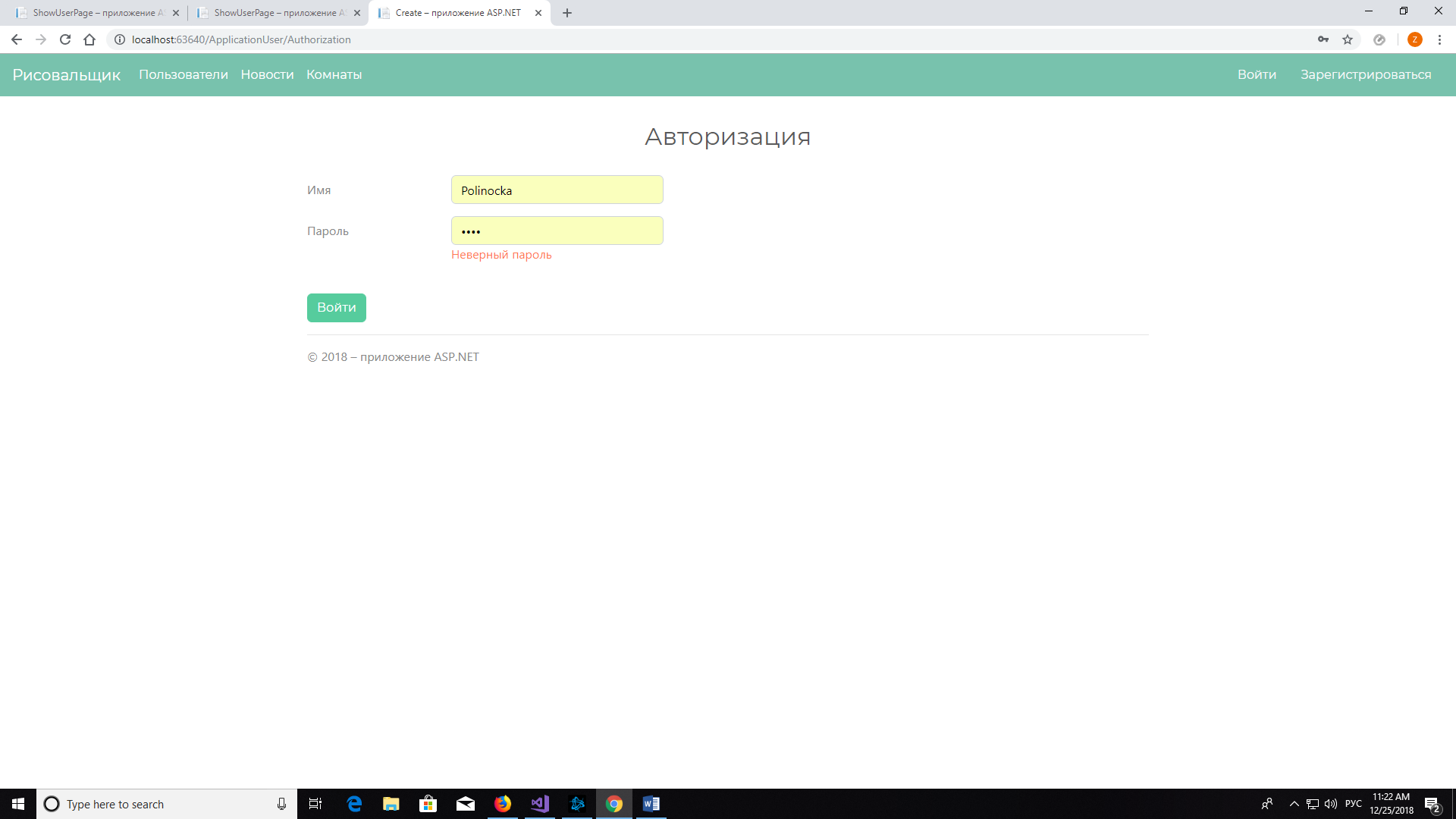


Рисунок 5.1 – Попытка входа под несуществующим пользователем

1. Пользователь будет перенаправлен на страницу, с которой он перешел на страницу авторизации, а также выведется имя и иконка профиля, представлено на рисунке 5.2.

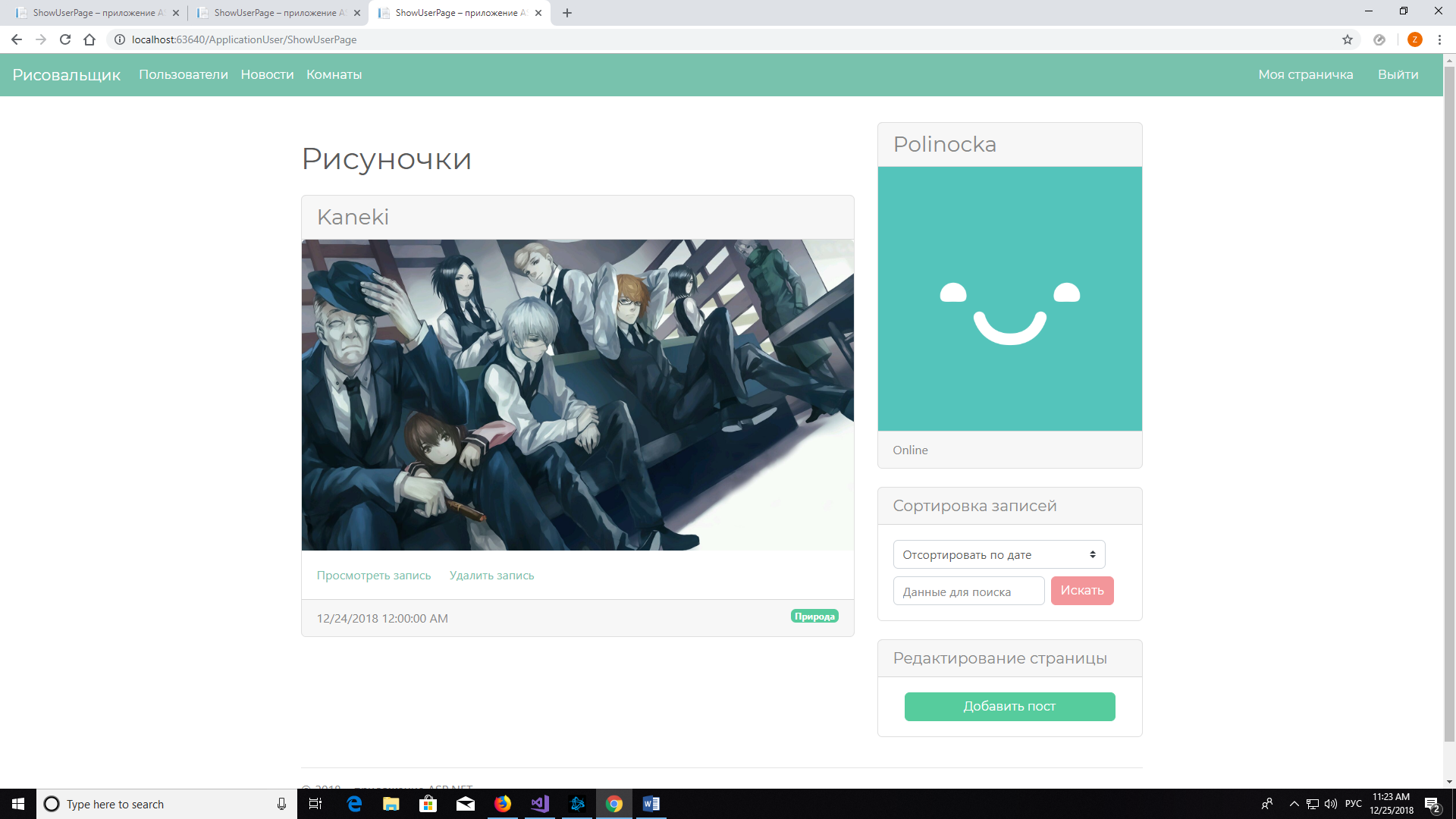


Рисунок 5.2 – Успешная авторизация

1. Будут выведены уведомления об ошибках, представлены на рисунке 5.3.

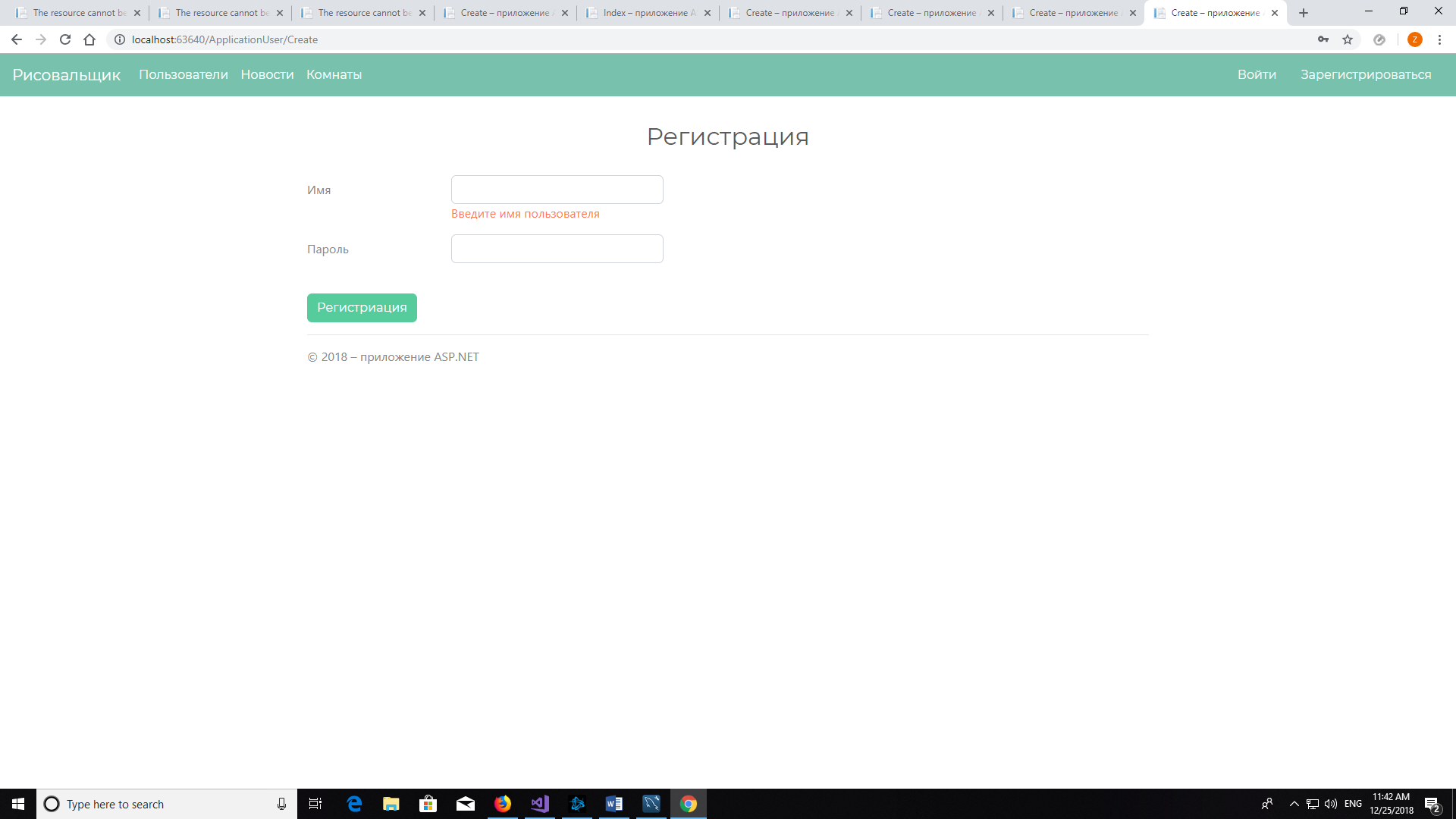


Рисунок 5.3 – Попытка регистрации с незаполненными полями

1. Будет выведено уведомление об ошибке, что пользователь с данным логином уже существует, представлено на рисунке 5.4.

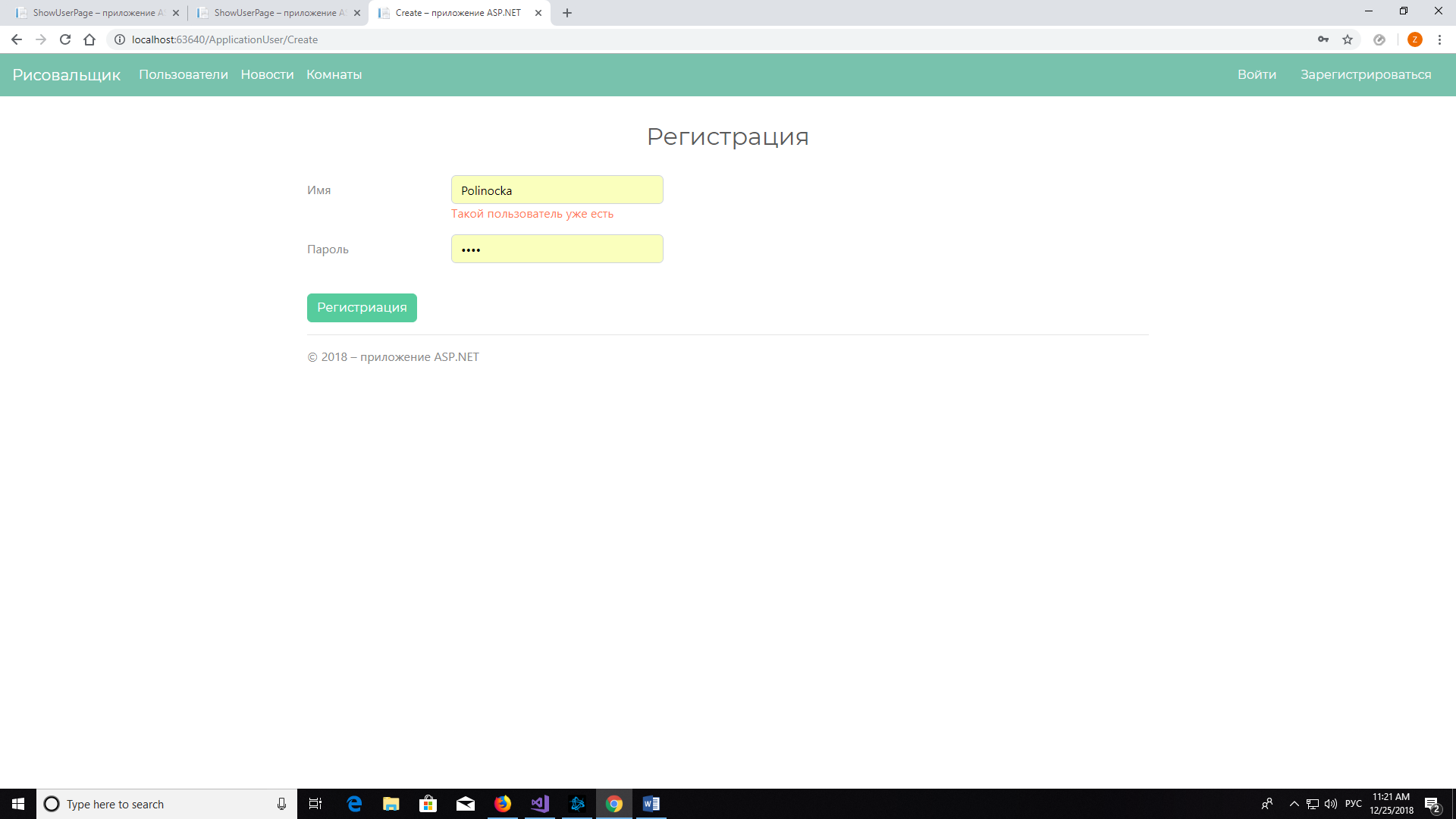


Рисунок 5.4 – Попытка регистрации с уже существующим логином

1. Будет выведено уведомление об ошибке, что формат почты неверный, представлено на рисунке 5.5.

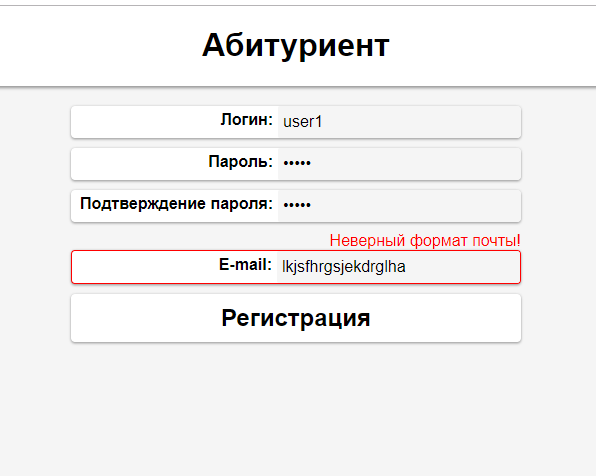


Рисунок 5.5 – Попытка регистрации с неверным форматом почты

# Описание применения программы

## Назначение программы и условия применения

Веб-приложение является информационным ресурсом для обмена, и обсуждения своих и чужих рисунков.

Для работы с веб-приложением необходимо:

* мобильный телефон или стационарный компьютер;
* браузер старшего поколения с поддержкой JavaScript;
* интернет соединение;
* устройства манипулирования: тачскрин, компьютерная мышь;
* устройство ввода: экранная клавиатура, клавиатура.

## Описание работы программы

Страница авторизации представлена на рисунке 6.1.

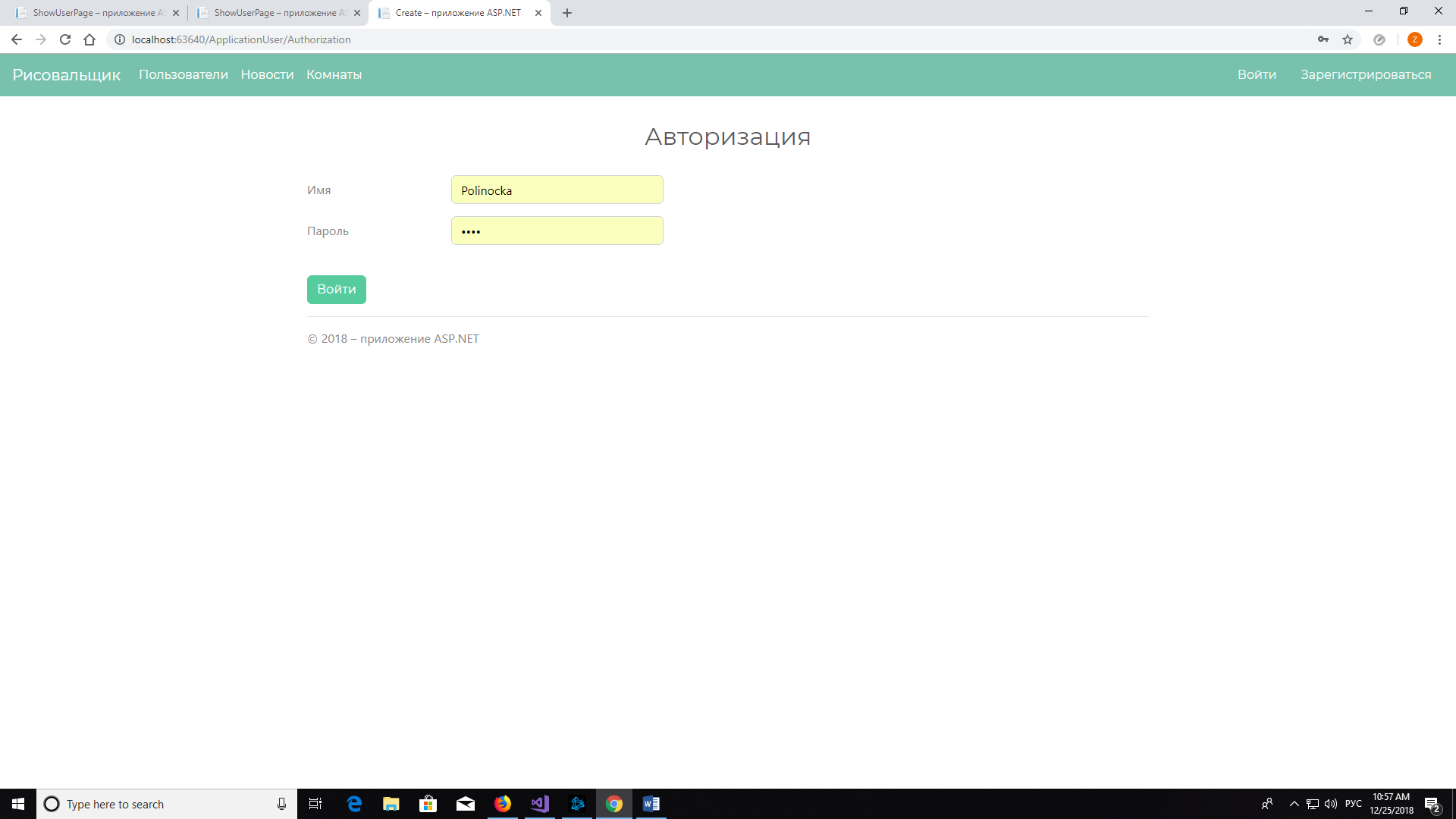


Рисунок 6.1 – Страница авторизации

Страница регистрации представлена на рисунке 6.2.

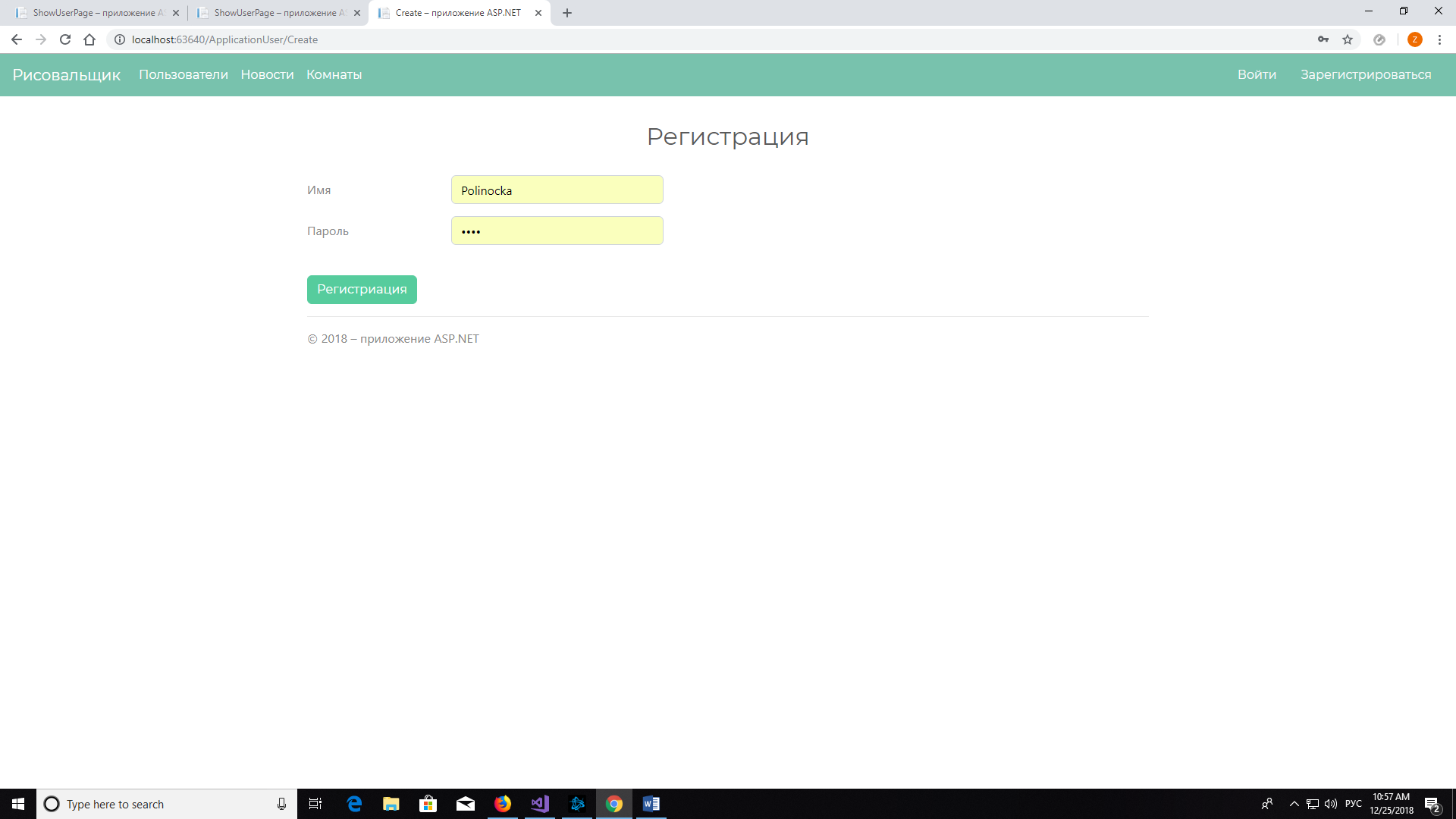


Рисунок 6.2 – Страница регистрации

Страница новостей представлена на рисунке 6.3. на ней отображаются самые актуальные, и последние записи пользователей.

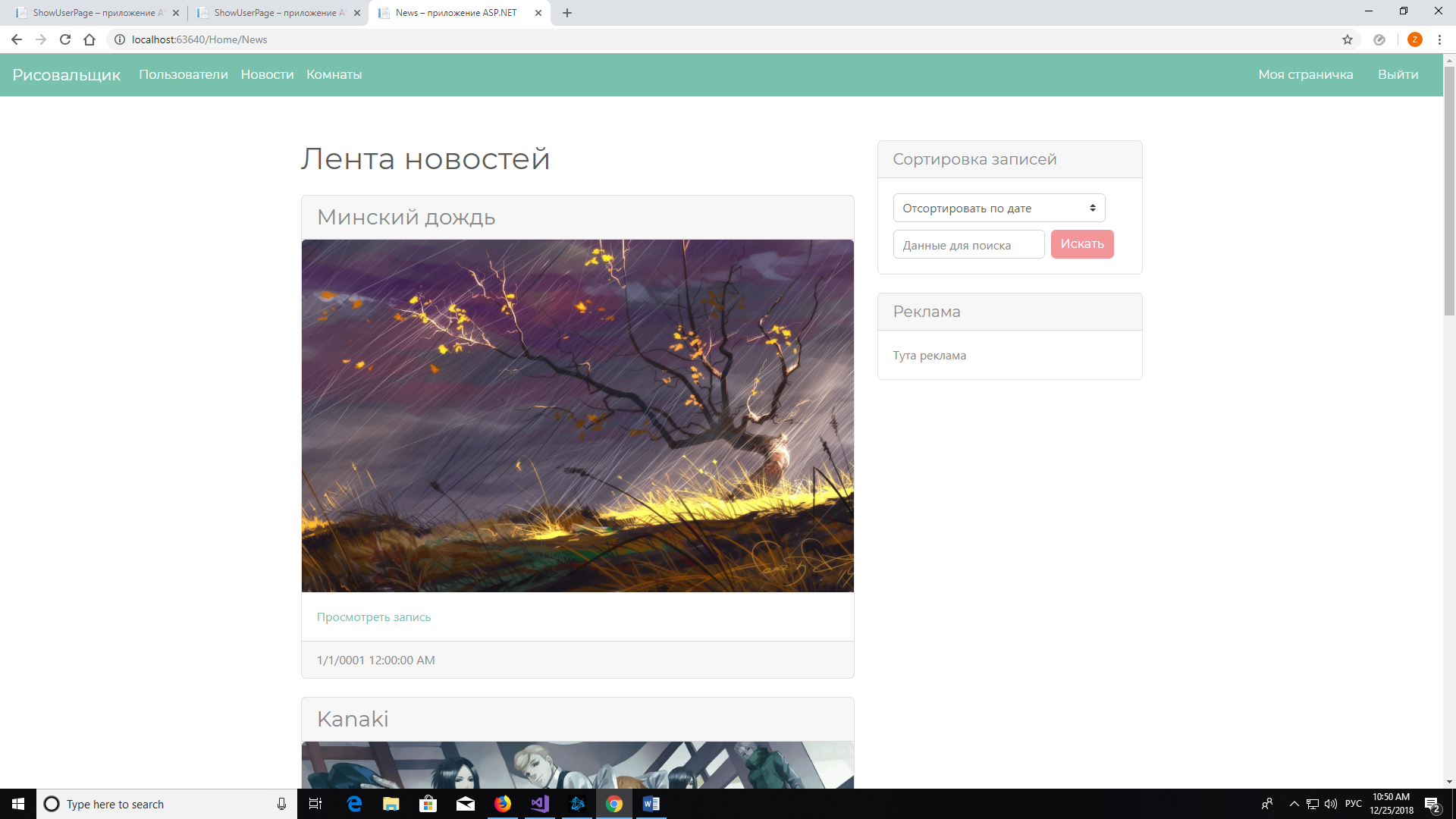


Рисунок 6.3 – Страница новостей

Страница пользователя представлена на рисунке 6.4. на ней отображается информация о пользователе и его записях.

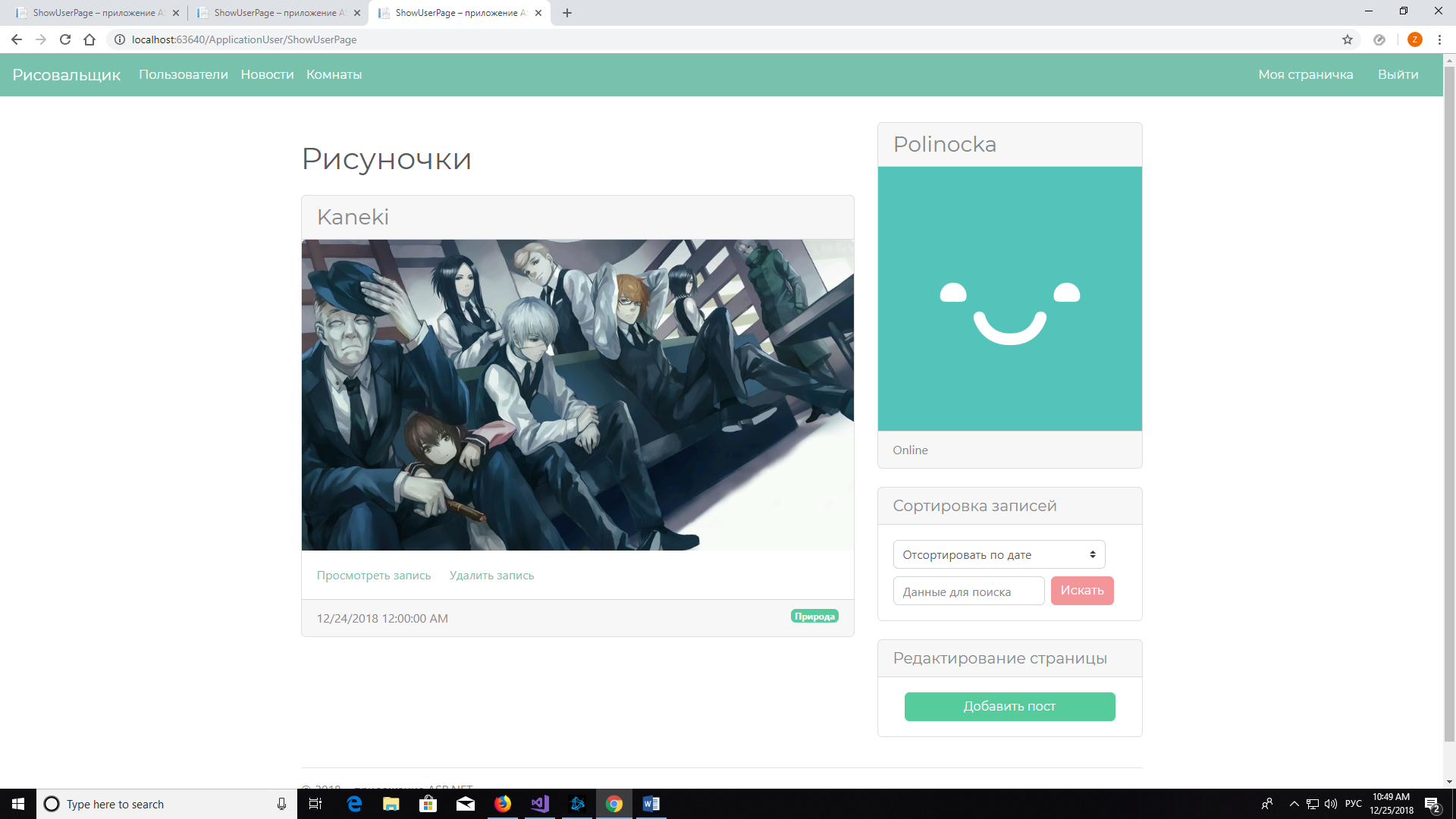


Рисунок 6.4 – Страница пользователя

# Заключение

В данном курсовом проекте было разработано информационное веб-приложение для получения и поиска информации об учебных заведениях на язык C# с использованием технологии ASP.NET и паттерна MVC. Была спроектирована и разработана база данных, используя язык SQL и СУБД MySQL для хранения информации об учебных заведениях, факультетах, специальностях, общежитиях, корпусах, отзывах и пользователях. Была реализована регистрация, авторизация и возможность оставлять отзывы. Предусмотрены следующие роли: «Администратор», «Пользователь».

В приложении реализован просмотр страниц, поиск с фильтрами, добавление отзывов, просмотр подробной информации об учебном заведении.

В ходе разработки данного проекта получилось реализовать не все задуманные функции. Получилось оптимизировать код для дальнейшего использования и продвижения.

# Список используемых источников

1. Дж. Рихтер, 2016 CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. – С. 25-30
2. Либерти Д. Язык программирования C# // Программирование на C#. – Санкт-Петербург. – 2003: Символ-Плюс. – С. 40–42.
3. Шварц Б. MySQL по максимуму. – СПБ.: Питер, 2018. – 864 с.
4. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения. – СПб.: Питер, 2004. – 318 с.

# Приложение А

**(обязательное)**

**Листинг программы**

**AddInstitutionController.cs**

**using System.Web;**

**using System.Web.Mvc;**

**using AbiturSite.DataBase;**

**using AbiturSite.Models;**

**namespace AbiturSite.Controllers**

**{**

**public class AddInstitutionController : Controller**

**{**

**// GET: AddInstitution**

**public ActionResult Index()**

**{**

**return View();**

**}**

**[HttpPost]**

**public ActionResult Index(string shortName, string fullName, string descrition, HttpPostedFileBase mainPhoto,**

**HttpPostedFileBase emblem)**

**{**

**Database.ExecNonQuery($"insert into offers.Institution values (default, '{shortName}', '{fullName}', '{descrition}')");**

**return Preview(shortName, fullName, descrition, mainPhoto, emblem);**

**}**

**[HttpPost]**

**public ActionResult Preview(string shortName, string fullName, string descrition, HttpPostedFileBase mainPhoto, HttpPostedFileBase emblem)**

**{**

**var model = new Models.Institution**

**{**

**ShortName = shortName,**

**FullName = fullName,**

**Description = descrition,**

**MainPhoto = mainPhoto.FileName,**

**Emblem = emblem.FileName,**

**Faculties = new Faculty[0],**

**Reviews = new Review[0]**

**};**

**ViewBag.Title = model.ShortName + " " + model.FullName;**

**return View(model);**

**}**

**}**

**}**

**ControlPanelController.cs**

**using System.Data;**

**using System.Linq;**

**using System.Web.Mvc;**

**using AbiturSite.DataBase;**

**namespace AbiturSite.Controllers**

**{**

**public class ControlPanelController : Controller**

**{**

**// GET: AdminPanel**

**public ActionResult Index()**

**{**

**return View();**

**}**

**[HttpPost]**

**public ActionResult Index(string token)**

**{**

**var uid = Sessions.GetUserId(token);**

**if (uid == null)**

**return Redirect("404");**

**var user = Models.User.GetById((long) uid);**

**if (user.Group == "user")**

**return Redirect("404");**

**return View();**

**}**

**[HttpPost]**

**public ActionResult GetOffers()**

**{**

**var data = Database.ExecQuery("select i\_id, i\_short\_name, substring(i\_description,1,200) as 'description' from offers.institutions").Rows**

**.Cast<DataRow>().Select(row => new**

**{**

**Id = (int)row["i\_id"],**

**ShortName = (string) row["i\_short\_name"],**

**Description = (string) row["description"]**

**}).ToArray();**

**return Json(data);**

**}**

**}**

**}**

**HomeController.cs**

**using System.Collections.Generic;**

**using System.Data;**

**using System.IO.Compression;**

**using System.Linq;**

**using System.Net;**

**using System.Text;**

**using System.Web.Mvc;**

**using AbiturSite.DataBase;**

**using MySql.Data.MySqlClient;**

**namespace AbiturSite.Controllers**

**{**

**public class HomeController : Controller**

**{**

**// GET: Home**

**public ActionResult Index()**

**{**

**return View();**

**}**

**internal static StringBuilder QueryConstructor(bool? university, bool? paid, string[] studyForm, decimal? minCost,**

**decimal? maxCost, int? minMark, int? maxMark, string city, string specName)**

**{**

**var ret = new StringBuilder();**

**var whereConds = new List<string>();**

**if (university != null)**

**whereConds.Add($"i\_type = '{(university == true ? "university" : "colledge")}'");**

**if (specName != null)**

**whereConds.Add($"s\_name like '%{specName}%'");**

**if (paid != null)**

**{**

**if (paid == true)**

**{**

**whereConds.Add($"s\_paid\_places > 0");**

**if (maxMark != null)**

**whereConds.Add($"s\_paid\_mark <= {maxMark}");**

**if (minMark != null)**

**whereConds.Add($"s\_paid\_mark >= {minMark}");**

**if (maxCost != null)**

**whereConds.Add($"s\_cost <= {maxCost}");**

**if (minCost != null)**

**whereConds.Add($"s\_cost >= {minCost}");**

**}**

**else**

**{**

**whereConds.Add($"s\_budget\_places > 0");**

**if (maxMark != null)**

**whereConds.Add($"s\_budget\_mark <= {maxMark}");**

**if (minMark != null)**

**whereConds.Add($"s\_budget\_mark >= {minMark}");**

**}**

**}**

**else**

**{**

**if (maxMark != null)**

**whereConds.Add($"(s\_paid\_mark <= {maxMark} or s\_budget\_mark <= {maxMark})");**

**if (minMark != null)**

**whereConds.Add($"(s\_paid\_mark >= {minMark} or s\_budget\_mark >= {minMark})");**

**}**

**if (studyForm != null)**

**whereConds.Add($"(s\_study\_form = '{string.Join("' or s\_study\_form = '", studyForm)}')");**

**if (city != null)**

**whereConds.Add($"i\_city = '{city}'");**

**//**

**if (whereConds.Count > 0)**

**ret.AppendFormat(" where {0}", string.Join(" and ", whereConds));**

**return ret;**

**}**

**[HttpPost]**

**public ActionResult InstitutionsFilter(bool? university = null, bool? paid = null, string[] studyForm = null,**

**decimal? minCost = null, decimal? maxCost = null, int? minMark = null, int? maxMark = null,**

**string city = null, string specName = null)**

**{**

**if (city != null)**

**city = MySqlHelper.EscapeString(city);**

**if (specName != null)**

**specName = MySqlHelper.EscapeString(specName);**

**var institutionsQuery = QueryConstructor(university, paid, studyForm, minCost, maxCost, minMark, maxMark,**

**city, specName);**

**institutionsQuery.Insert(0,**

**"select distinct i\_id from TapedTables");**

**var query = GetInstitutionQuery(paid, studyForm, minCost, maxCost, minMark, maxMark, specName);**

**var institutions = Database.ExecQuery(institutionsQuery.ToString()).Rows.Cast<DataRow>()**

**.Select(row => new**

**{**

**i = (int) row[0],**

**q = query**

**}).ToArray();**

**return Json(institutions);**

**}**

**[HttpPost]**

**public ActionResult GetInstitutionsData()**

**{**

**var institutions = Database**

**.ExecQuery("select i\_id, i\_short\_name, i\_address, i\_latitude, i\_longitude from Institutions")**

**.Rows.Cast<DataRow>().ToArray();**

**var institutionsLength = institutions.Length;**

**var outp = new object[institutionsLength];**

**for (var i = 0; i < institutionsLength; i++)**

**{**

**var institution = institutions[i];**

**outp[i] = new**

**{**

**i = (int) institution["i\_id"],**

**a = (string) institution["i\_address"],**

**c = new[] {(float) institution["i\_latitude"], (float) institution["i\_longitude"]},**

**n = (string) institution["i\_short\_name"]**

**};**

**}**

**return Json(outp);**

**}**

**private static StringBuilder GetInstitutionQuery(bool? paid, string[] studyForm, decimal? minCost,**

**decimal? maxCost, int? minMark, int? maxMark, string specName)**

**{**

**var query = new StringBuilder();**

**if (paid != null)**

**{**

**query.Append("&paid=").Append(paid);**

**if (paid == true)**

**{**

**if (minCost != null)**

**query.Append("&minCost=").Append(minCost);**

**if (maxCost != null)**

**query.Append("&maxCost=").Append(maxCost);**

**}**

**}**

**if (studyForm != null)**

**query.Append("&studyForm=").Append(string.Join("&studyForm=", studyForm));**

**if (minMark != null)**

**query.Append("&minMark=").Append(minMark);**

**if (maxMark != null)**

**query.Append("&maxMark=").Append(maxMark);**

**if (specName != null)**

**query.Append("&specName=").Append(specName);**

**return query;**

**}**

**[HttpPost]**

**public ActionResult GetCities(bool university)**

**{**

**var cities = Database.ExecQuery(**

**$"select i\_city from Cities where i\_type='{(university ? "university" : "colledge")}'");**

**return Json(cities.Rows.Cast<DataRow>().Select(x => ((string) x[0]).Replace("&nbsp;", " ")).ToArray());**

**}**

**}**

**}**

**LoginController.cs**

**using AbiturSite.DataBase;**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Web;**

**using System.Web.Mvc;**

**using MySql.Data.MySqlClient;**

**namespace AbiturSite.Controllers**

**{**

**public class LoginController : Controller**

**{**

**public ActionResult Index()**

**{**

**return View();**

**}**

**[HttpPost]**

**public ActionResult Index(string login, string pass)**

**{**

**login = MySqlHelper.EscapeString(login);**

**pass = MySqlHelper.EscapeString(pass);**

**var token = Sessions.Login(login, pass);**

**return Content(token);**

**}**

**[HttpPost]**

**public ActionResult ValidateToken(string token)**

**{**

**var userId = Sessions.GetUserId(token);**

**if (token == null || userId == null)**

**return Json(new {IsValid = false});**

**var user = Models.User.GetById((long) userId);**

**return Json(new**

**{**

**IsValid = true,**

**user.Group,**

**Id = userId,**

**user.Name,**

**user.Avatar**

**});**

**}**

**}**

**}**

**RegisterController.cs**

**using System;**

**using System.ComponentModel.DataAnnotations;**

**using System.Net;**

**using System.Net.Mail;**

**using System.Web.Mvc;**

**using AbiturSite.App\_Code;**

**using AbiturSite.DataBase;**

**using MySql.Data.MySqlClient;**

**namespace AbiturSite.Controllers**

**{**

**public class RegisterController : Controller**

**{**

**// GET**

**public ActionResult Index()**

**{**

**return View();**

**}**

**[HttpPost]**

**public ActionResult Index(string login, string pass, string passSubmit, string email)**

**{**

**ViewData["login"] = login;**

**ViewData["pass"] = pass;**

**ViewData["passSubmit"] = passSubmit;**

**ViewData["email"] = email;**

**bool loginE = false, passE = false, emailE = false, otherE = false;**

**#region Validation**

**if (string.IsNullOrWhiteSpace(pass))**

**{**

**ViewData["pass\_error"] = "Пароль обязателен!";**

**passE = true;**

**}**

**else if (pass.Length > 50)**

**{**

**ViewData["pass\_error"] = "Пароль должен содержать не более 50 символов!";**

**passE = true;**

**}**

**if (string.IsNullOrWhiteSpace(login))**

**{**

**ViewData["login\_error"] = "Логин обязателен!";**

**loginE = true;**

**}**

**else if (login.Length > 50)**

**{**

**ViewData["login\_error"] = "Логин должен содержать не более 50 символов!";**

**loginE = true;**

**}**

**else if (Database.ExecQuery(**

**$"select u\_id from Users where u\_login='{MySqlHelper.EscapeString(login)}'")**

**.Rows.Count > 0)**

**{**

**ViewData["login\_error"] = "Такой логин уже занят!";**

**loginE = true;**

**}**

**var passSubmitE = pass != passSubmit;**

**if (passSubmitE)**

**ViewData["passSubmit\_error"] = "Пароли не совпадают!";**

**if (string.IsNullOrWhiteSpace(email))**

**{**

**ViewData["email\_error"] = "E-mail обязателен!";**

**emailE = true;**

**}**

**else if (!new EmailAddressAttribute().IsValid(email))**

**{**

**ViewData["email\_error"] = "Неверный формат почты!";**

**emailE = true;**

**}**

**#endregion**

**if (!loginE && !passE && !passSubmitE && !emailE)**

**{**

**var token = Sessions.GenerateToken(out \_);**

**var error = Utils.SendEmail(email, "Подтверждение регистрации",**

**$@"<html><body>Нажмите на кнопку для подтверждения регистрации <a href=""http://localhost:63704/Register/Confirm?token={token}"" style=""display:block;border-width:1px;border-style:solid;cursor:pointer;outline:none;box-shadow:0 2px 4px 0 rgba(0,0,0,0.14),0 3px 4px 0 rgba(0,0,0,0.12),0 1px 5px 0 rgba(0,0,0,0.2);border-radius:4px;padding:10px;background-color:white;text-decoration:none;text-align:center;font-size:16px;font-weight:bold;font-family:Helvetica,sans-serif;transition:0.2s linear;border-color:#4CAF50;color:#4CAF50"">Подтвердить регистрацию</a></body></html>",**

**true);**

**if (error != null)**

**{**

**otherE = true;**

**ViewData["email\_send\_error"] =**

**"При отправке письма на указанную вами почту произошла ошибка. Проверьте правильность адреса почты или попробуйте зарегистрироваться чуть позже.";**

**}**

**login = MySqlHelper.EscapeString(login);**

**pass = MySqlHelper.EscapeString(pass);**

**email = MySqlHelper.EscapeString(email);**

**token = MySqlHelper.EscapeString(token);**

**Database.ExecNonQuery(**

**$"insert into NotSubmittedUsers values ('{token}', '{login}', '{pass}', '{email}')");**

**}**

**ViewData["has\_errors"] = loginE || passE || passSubmitE || emailE || otherE;**

**return View();**

**}**

**[HttpGet]**

**public ActionResult Confirm(string token)**

**{**

**Database.ExecNonQuery($"call SubmitNewUser('{token}')");**

**return View();**

**}**

**}**

**}**

**Ajax.cs**

**using System.Data;**

**using System.Linq;**

**using System.Web.Mvc;**

**using AbiturSite.DataBase;**

**using MySql.Data.MySqlClient;**

**namespace AbiturSite.Controllers.Institution**

**{**

**public partial class InstitutionController**

**{**

**[HttpPost]**

**public ActionResult SendReview(int id, string token, string text, int raiting)**

**{**

**var uid = Sessions.GetUserId(token);**

**if (uid == null)**

**{**

**return Json(new**

**{**

**Success = false,**

**IsAutorized = false,**

**Message = "Токен не валидный"**

**});**

**}**

**if (raiting == 0)**

**return Json(new**

**{**

**Success = false,**

**IsAutorized = true,**

**Message = "Необходимо поставить оценку учебному заведению!"**

**});**

**if (raiting <= 0 || raiting > 10)**

**return Json(new**

**{**

**Success = false,**

**IsAutorized = true,**

**Message = "Оценка учебного заведения может быть только в диапазоне от 1 до 10 включительно!"**

**});**

**if (string.IsNullOrWhiteSpace(text))**

**return Json(new**

**{**

**Success = false,**

**IsAutorized = true,**

**Message = "Пустой комментарий"**

**});**

**text = MySqlHelper.EscapeString(text);**

**Database.ExecNonQuery($"insert into reviews values (default, {id}, {uid}, {raiting}, '{text}', now())");**

**return Json(new**

**{**

**Success = true,**

**IsAutorized = true,**

**Message = ""**

**});**

**}**

**[HttpPost]**

**public ActionResult GetBuildingsInfo(int id)**

**{**

**var hostelsData = Database**

**.ExecQuery(**

**$"select h\_id, h\_latitude, h\_longitude, h\_address from Hostels where h\_institution = {id}")**

**.Rows.Cast<DataRow>()**

**.Select(h => new**

**{**

**i = (int) h["h\_id"],**

**c = new[] {(float) h["h\_latitude"], (float) h["h\_longitude"]},**

**a = (string) h["h\_address"]**

**}).ToArray();**

**var buildingsData = Database**

**.ExecQuery(**

**$"select b\_id, b\_latitude, b\_longitude, b\_address from Buildings where b\_institution = {id}")**

**.Rows.Cast<DataRow>()**

**.Select(h => new**

**{**

**i = (int) h["b\_id"],**

**c = new[] {(float) h["b\_latitude"], (float) h["b\_longitude"]},**

**a = (string) h["b\_address"],**

**}).ToArray();**

**return Json(new**

**{**

**Hostels = hostelsData,**

**Buildings = buildingsData**

**});**

**}**

**}**

**}**

**InstitutionController.cs**

**using System.Data;**

**using System.Linq;**

**using System.Web.Mvc;**

**using AbiturSite.DataBase;**

**using AbiturSite.Models;**

**using MySql.Data.MySqlClient;**

**namespace AbiturSite.Controllers.Institution**

**{**

**public partial class InstitutionController : Controller**

**{**

**[HttpGet]**

**public ActionResult Index(int id, bool? paid, string[] studyForm, decimal? minCost, decimal? maxCost,**

**int? minMark, int? maxMark, string specName)**

**{**

**var facultiesRows = Database.ExecQuery($"select f\_id, f\_name from Faculties where f\_university={id}").Rows**

**.Cast<DataRow>().ToArray();**

**#region Specialities**

**if (specName != null)**

**specName = MySqlHelper.EscapeString(specName);**

**var specQuery = HomeController.QueryConstructor(null, paid, studyForm, minCost, maxCost, minMark, maxMark,**

**null, specName);**

**if (specQuery.Length == 0)**

**specQuery.Append(" where ");**

**else specQuery.Append(" and ");**

**specQuery.AppendFormat("(s\_faculty={0})",**

**string.Join(" or s\_faculty=", facultiesRows.Select(x => x["f\_id"])));**

**specQuery.Insert(0, "select \* from Specialities");**

**var specialities = Database.ExecQuery(specQuery.ToString()).Rows.Cast<DataRow>();**

**#endregion**

**var faculties = facultiesRows.Select(f => new Faculty**

**{**

**Name = (string) f["f\_name"],**

**Specialities = specialities.Where(s => (int) s["s\_faculty"] == (int) f["f\_id"]).Select(s =>**

**new Speciality**

**{**

**Name = (string) s["s\_name"],**

**StudyForm = (string) s["s\_study\_form"],**

**BudgetPlaces = (uint) s["s\_budget\_places"],**

**BudgetMark = (uint) s["s\_budget\_mark"],**

**PaidPlaces = (uint) s["s\_paid\_places"],**

**PaidMark = (uint) s["s\_paid\_mark"],**

**Cost = (decimal) s["s\_cost"]**

**}).ToArray()**

**}).Where(f => f.Specialities.Length > 0).ToArray();**

**var reviews = Review.GetByInstitution(id);**

**var institution = Database.ExecQuery($"select \* from institutions where i\_id={id}").Rows[0];**

**var model = new Models.Institution**

**{**

**Id = id,**

**ShortName = (string) institution["i\_short\_name"],**

**FullName = (string) institution["i\_full\_name"],**

**Description = (string) institution["i\_description"],**

**MainPhoto = (string) institution["i\_main\_photo"],**

**Emblem = (string) institution["i\_logo"],**

**Faculties = faculties,**

**Reviews = reviews**

**};**

**return View(model);**

**}**

**public ActionResult Offer(int id, bool? accept)**

**{**

**// ReSharper disable once ConvertIfStatementToSwitchStatement**

**if (accept == null)**

**return View(Models.Institution.GetById(id, "offers"));**

**if (accept == false)**

**{**

**Database.ExecNonQuery($"delete from offers.institutions where i\_id={id}");**

**return Redirect("/ControlPanel");**

**}**

**Database.ExecNonQuery(**

**"insert into institutions (i\_type,i\_short\_name,i\_full\_name,i\_city,i\_address,i\_latitude,i\_longitude,i\_description,i\_main\_photo,i\_logo) values " +**

**"select (i\_type,i\_short\_name,i\_full\_name,i\_city,i\_address,i\_latitude,i\_longitude,i\_description,i\_main\_photo,i\_logo) from offers.institutions where i\_id=" +**

**id);**

**return Redirect("/Institution?id=" +**

**Database.ExecuteScalar("select i\_id from institutions order by i\_id desc limit 1"));**

**}**

**}**

**}**

**Institution.cs**

**using System;**

**using System.Data;**

**using System.Linq;**

**using AbiturSite.DataBase;**

**namespace AbiturSite.Models**

**{**

**public class Institution**

**{**

**public Faculty[] Faculties;**

**public int Id;**

**public Review[] Reviews;**

**public string ShortName, FullName, Description, MainPhoto, Emblem, Address;**

**public static Institution GetById(int id, string database = "kek")**

**{**

**var facultiesRows = Database.ExecQuery($"select f\_id, f\_name from {database}.Faculties where f\_university={id}").Rows**

**.Cast<DataRow>().ToArray();**

**var specialities = Database**

**.ExecQuery(**

**$"select \* from {database}.Specialities where s\_faculty in ({string.Join(",", facultiesRows.Select(x => x["f\_id"]))})")**

**.Rows.Cast<DataRow>();**

**var faculties = facultiesRows.Select(f => new Faculty**

**{**

**Name = f["f\_name"] as string,**

**Specialities = specialities.Where(s => (int) s["s\_faculty"] == (int) f["f\_id"]).Select(s =>**

**new Speciality**

**{**

**Name = s["s\_name"] as string,**

**StudyForm = s["s\_study\_form"] as string,**

**BudgetPlaces = (uint) s["s\_budget\_places"],**

**BudgetMark = (uint) s["s\_budget\_mark"],**

**PaidPlaces = (uint) s["s\_paid\_places"],**

**PaidMark = (uint) s["s\_paid\_mark"],**

**Cost = (decimal) s["s\_cost"]**

**}).ToArray()**

**}).Where(f => f.Specialities.Length > 0).ToArray();**

**var reviews = Review.GetByInstitution(id, database);**

**var institution = Database.ExecQuery($"select \* from {database}.institutions where i\_id={id}").Rows[0];**

**var result = new Institution**

**{**

**Id = id,**

**ShortName = institution["i\_short\_name"] as string,**

**FullName = institution["i\_full\_name"] as string,**

**Description = institution["i\_description"] as string,**

**MainPhoto = institution["i\_main\_photo"] as string,**

**Emblem = institution["i\_logo"] as string,**

**Faculties = faculties,**

**Reviews = reviews**

**};**

**return result;**

**}**

**}**

**public class Review**

**{**

**public DateTime Datetime;**

**public byte Raiting;**

**public string Text;**

**public User User;**

**public static Review[] GetByInstitution(int institution, string database = "kek")**

**{**

**return Database.ExecQuery($"call {database}.GetReviewsByInstitution({institution})").Rows.Cast<DataRow>()**

**.Select(r => new Review**

**{**

**Text = r["r\_text"] as string,**

**User = User.GetById((long) r["r\_profile\_id"]),**

**Raiting = (byte) r["r\_raiting"],**

**Datetime = (DateTime) r["r\_datetime"]**

**}).ToArray();**

**}**

**}**

**public class User**

**{**

**public long Id, Vk;**

**public string Login;**

**public string Name, Surname, Email, Avatar, Group;**

**public static User GetById(long id)**

**{**

**var result = Database.ExecQuery($"select \* from Users where u\_id={id}").Rows;**

**if (result.Count == 0)**

**return null;**

**var userData = result[0];**

**var user = new User**

**{**

**Id = (long) userData["u\_id"],**

**Vk = (long) userData["u\_vk"],**

**Login = (string) userData["u\_login"],**

**Name = (string) userData["u\_name"],**

**Surname = (string) userData["u\_surname"],**

**Email = (string) userData["u\_email"],**

**Avatar = (string) userData["u\_avatar"],**

**Group = (string) userData["u\_group"]**

**};**

**return user;**

**}**

**}**

**public class Faculty**

**{**

**public string Name;**

**public Speciality[] Specialities;**

**}**

**public class Speciality**

**{**

**public uint BudgetMark, PaidMark;**

**public uint BudgetPlaces, PaidPlaces;**

**public decimal Cost;**

**public string Name, StudyForm;**

**}**

**}**

**Database.cs**

**using System;**

**using System.Data;**

**using MySql.Data.MySqlClient;**

**namespace AbiturSite.DataBase**

**{**

**public static class Database**

**{**

**private static MySqlCommand cmd;**

**private static MySqlConnection con;**

**public static DataTable UniversityTable, FacultyTable, SpecialityTable;**

**static Database()**

**{**

**UniversityTable = new DataTable();**

**FacultyTable = new DataTable();**

**SpecialityTable = new DataTable();**

**}**

**public static void Connect(string login, string pass)**

**{**

**con = new MySqlConnection($"Server=127.0.0.1;Database=kek;Uid={login};Pwd={pass};");**

**con.Open();**

**cmd = con.CreateCommand();**

**Refresh(true);**

**}**

**public static void Refresh(bool refreshColumns)**

**{**

**FillTable(UniversityTable, "select \* from institutions", refreshColumns);**

**FillTable(FacultyTable, "select \* from Faculties", refreshColumns);**

**FillTable(SpecialityTable, "select \* from Specialities", refreshColumns);**

**}**

**public static void FillTable(DataTable table, string query, bool refreshColumns)**

**{**

**cmd.CommandText = query;**

**table.Clear();**

**var reader = cmd.ExecuteReader();**

**if (refreshColumns)**

**{**

**var schemaTableRows = reader.GetSchemaTable().Rows;**

**table.Columns.Clear();**

**var count = schemaTableRows.Count;**

**for (var i = 0; i < count; i++)**

**{**

**var row = schemaTableRows[i];**

**table.Columns.Add(new DataColumn((string) row["ColumnName"], (Type) row["DataType"]));**

**}**

**}**

**var fieldCount = reader.FieldCount;**

**while (reader.Read())**

**{**

**var values = new object[fieldCount];**

**reader.GetValues(values);**

**var row = table.NewRow();**

**row.ItemArray = values;**

**table.Rows.Add(row);**

**}**

**reader.Close();**

**}**

**public static void ExecNonQuery(string command)**

**{**

**cmd.CommandText = command;**

**var transaction = con.BeginTransaction();**

**cmd.Transaction = transaction;**

**try**

**{**

**cmd.ExecuteNonQuery();**

**transaction.Commit();**

**}**

**catch (Exception e)**

**{**

**transaction.Rollback();**

**Console.WriteLine(e);**

**}**

**}**

**public static DataTable ExecQuery(string command)**

**{**

**var table = new DataTable();**

**FillTable(table, command, true);**

**return table;**

**}**

**public static object ExecuteScalar(string command)**

**{**

**cmd.CommandText = command;**

**return cmd.ExecuteScalar();**

**}**

**public static T? ExecuteScalar<T>(string command) where T: struct**

**{**

**cmd.CommandText = command;**

**var result = cmd.ExecuteScalar();**

**return (T?) result;**

**}**

**}**

**}**

**Sessions.cs**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**namespace AbiturSite.DataBase**

**{**

**public static class Sessions**

**{**

**public static long? GetUserId(string token)**

**{**

**return Database.ExecuteScalar<long>("select uid from Sessions");**

**}**

**public static string Login(string login, string pass)**

**{**

**var uid = Database.ExecuteScalar<long>($"select LoginUser('{login}', '{pass}')");**

**if (uid is null)**

**return string.Empty;**

**var token = (string) Database.ExecuteScalar("select token from Sessions where uid=" + uid);**

**// ReSharper disable once InvertIf**

**if (token == null)**

**{**

**token = GenerateToken(out var generationTime);**

**Database.ExecNonQuery($"insert into Sessions values ({token}, {uid}, {generationTime:yy-MM-dd})");**

**}**

**return token;**

**}**

**public static string GenerateToken(out DateTime generationTime)**

**{**

**generationTime = DateTime.UtcNow;**

**var time = BitConverter.GetBytes(generationTime.ToBinary());**

**var key = Guid.NewGuid().ToByteArray();**

**return Convert.ToBase64String(time.Concat(key).ToArray());**

**}**

**}**

**}**

**Utils.cs**

**using System;**

**using System.Net;**

**using System.Net.Mail;**

**namespace AbiturSite.App\_Code**

**{**

**public class Utils**

**{**

**public static Exception SendEmail(string to, string subject, string body, bool isBodyHtml)**

**{**

**var fromAddress = new MailAddress("mr.dan1999@mail.ru", "Сайт Абитуриент");**

**var toAddress = new MailAddress(to);**

**var credential = new NetworkCredential("mr.dan1999", "bymedion7722846dan");**

**var smtp = new SmtpClient**

**{**

**Host = "smtp.mail.ru",**

**Port = 25,**

**EnableSsl = true,**

**DeliveryMethod = SmtpDeliveryMethod.Network,**

**UseDefaultCredentials = false,**

**Credentials = credential,**

**Timeout = 5000**

**};**

**var mail = new MailMessage(fromAddress, toAddress)**

**{**

**IsBodyHtml = isBodyHtml,**

**Body = body,**

**Subject = subject**

**};**

**try**

**{**

**smtp.Send(mail);**

**}**

**catch (Exception e)**

**{**

**return e;**

**}**

**finally**

**{**

**smtp.Dispose();**

**mail.Dispose();**

**}**

**return null;**

**}**

**}**

**}**