СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_Toc289204035)………………………………………………………………………………………6

1 АНАЛИЗ СОСТОЯНИЯ ВОПРОСА…………………………………………………………7

1.1 Социальная сеть Вконтакте…………………………………………………………………7

1.2 ASP .NET MVC……………………………………………………………………………....9

1.2.1 Основные концепции ASP .NET……………………………………………………….....9

1.2.2 Архитектура Model View Controller…………………………………………………......11

1.2.3 Механизм визуализации Razor…………………………………………………………..11

[2 ПОСТАНОВКА ЗАДАЧИ](#_Toc289204036)……………………………………………………………………13

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ……………………………..14

3.1 Модели данных……………………………………………………………………………..14

3.2 Доступ к данным…………………………………………………………………………...14

3.3 Контроллеры……………………………………………………………………………......16

3.4 Представления……………... ………………………………………………………………19

4 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ…………………………………….21

4.1 Модели данных……………………………………………………………………………..21

4.2 Доступ к данным…………………………………………………………………………...21

4.3 Контроллеры……………………………………………………………………………......22

4.4 Представления……………... ………………………………………………………………23

5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ…………………………………25

5.1 Модульное тестирование…………………………………………………………………..25

5.2 Функциональное тестирование……………………………………………………………25

[ВЫВОДЫ](#_Toc289204081)……………………………………………………………………………………….28

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ…………………………………………..29

[ПРИЛОЖЕНИЕ А](#_Toc289204082)………………………………………………………………………………30

ВВЕДЕНИЕ

Количество зарегистрированных пользователей десяти ведущих социальных сетей на январь 2014 года в общей суме составило около 4.5 миллиарда [1], когда как население планеты земля на январь 2014 года около 7.1 миллиарда человек [2]. С учетом всех факторов можно заявлять, что больше половины человечества когда-либо пользовало или пользуется социальными сетями. Социальные сети не разрабатывают новые информационные технологии, как это делает корпорация Microsoft. Они и не делают научных открытий подобно NASA. Всего лишь предоставляя возможность интерактивного общения, социальные сети приносят миллиарды прибыли ежегодно.

Целью данной работы является написание веб-приложения, которое будет предоставлять возможности интерактивного общения. Использование ASP .NET MVC Framework должно стать основным достоинствам проекта.

В первом разделе производится анализ состояния вопроса, рассматривается существующая социальная сеть Вконтакте, описывается технология создания веб-приложений ASP .NET MVC. Во втором разделе представлен список задач. В третьем разделе производится проектирование веб-приложения. Четвёртый раздел описывает особенности реализации веб-приложения. В пятом разделе проводится модульное и функциональное тестирование разработанного веб-приложения.

1 АНАЛИЗ СОСТОЯНИЯ ВОПРОСА

* 1. Социальная сеть Вконтакте

Социальная сеть Вконтакте насчитывает свыше 250 миллионов зарегистрированных пользователей. Ежедневная аудитория составляет 60 миллионов человек [4]. Вконтакте это 21-ый сайт по посещаемости в мире. Финансовый оборот на 2013 год составил приблизительно 110 миллионов долларов США [1].

Вконтакте существует 3 способа размещения рекламы [5]:

- Непосредственная реклама на страницах пользователей;

- Разнообразные сообщества и группы;

- Приложения.

Особенно стоит отметить второй способ. Возможность создания меню группы и использование при этом wiki разметки делают группу Вконтакте не просто еще одной страницей социальной сети, а полноценным сайтом рисунок 1.1.1 [5].



Рисунок 1.1.1 – Меню группы Вконтакте

Выше сказанное подтверждается и тем, что каждая группа также может иметь сколь угодно большое число страниц, которые благодаря wiki разметке, по сути, ничем не отличаются от обычных страниц любого сайта рисунок 1.1.2 [5].

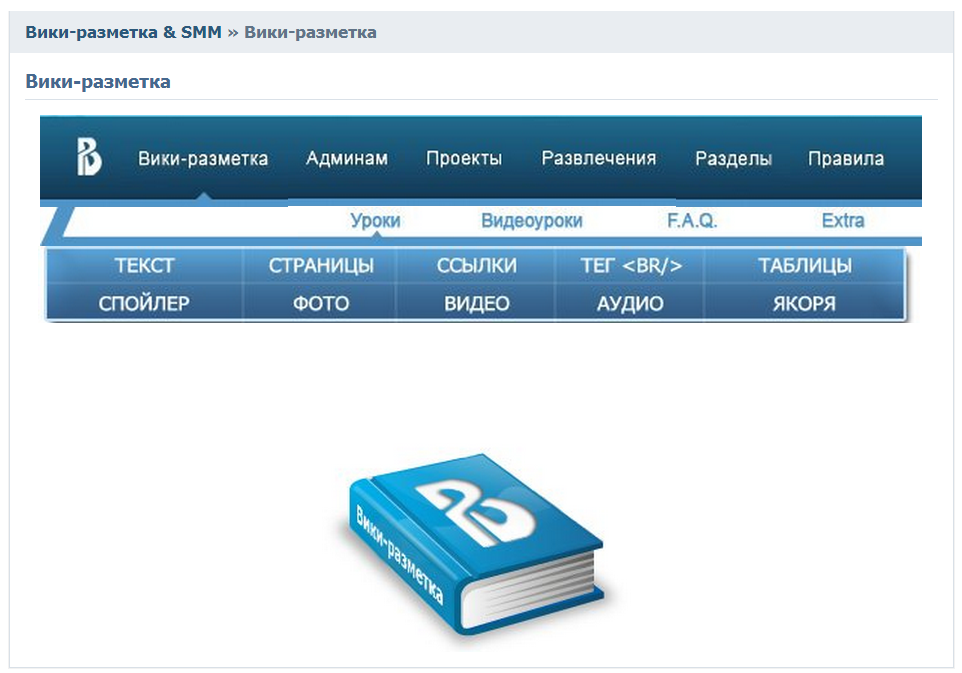


Рисунок 1.1.2 – Страница группы Вконтакте

В свою очередь разнообразные приложения и игры Вконтакте, собирающие у своих экранов миллионы пользователей, так же расширяют коммерческие возможности сайта рисунок 1.1.3 [5].



Рисунок 1.1.3 – Рейтинг приложений Вконтакте

1.2 ASP .NET MVC

1.2.1 Основные концепции ASP .NET

ASP .NET (Active Server Pages) – технология создания веб-приложений и веб-сервисов компании Microsoft. Она является составной частью платформы .NET и развитием более старой технологии ASP [1].

.NET (.NET Framework) – среда выполнения приложений, позволяющая использовать в разработке различные языки программирования (Common Language Runtime) [1]:

- C#;

- J#;

- VB .NET;

- JScript .NET;

- C++/CLI;

- F#.

.NET предлагает обширную библиотеку классов, которая позволяет обращаться к Win32 API. Кроме того .NET Framework включает в себя большое число расширений, среди которых ASP .NET. На рисунке 1.2.1.1 представлен стек технологий .NET.

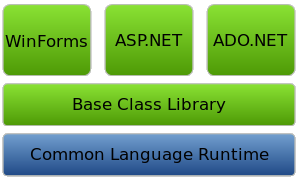


Рисунок 1.2.1.1 – Стек технологий .NET

На рисунке 1.2.1.2 представлена эволюция .NET Framework.

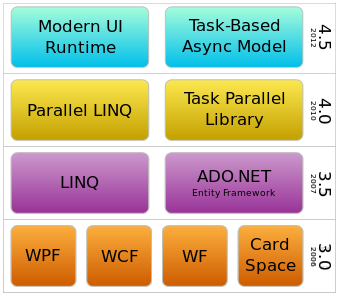


Рисунок 1.2.1.2 - Эволюция .NET Framework.

В таблице 1.2.1.1 представлена хронология развития технологий веб-разработки корпорации Microsoft [6].

Таблица 1.2.1.1

Эволюция технологий веб-разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Период | Технология | Достоинства | Недостатки |
| 1996 | ASP | Универсальность назначения | Интерпретация во время выполнения |
| 2002 2003 | ASP .NET  Web Forms 1.0 | Компилируемый пользовательский интерфейс с поддержкой состояний | - Передача состояния.  - Беспорядочный HTML-код  - Отсутствие возможности тестирования |
| 2005 | Web Forms 2.0 |  |  |
| 2007 | ASP. NET AJAX |  |  |
| 2008 | Web Forms 3.5 |  |  |
| 2009 | MVC 1.0 | - Архитектура Model View Controller  - Жесткий контроль над HTML и HTTP  - Тестируемость  - Система маршрутизации | Сложность разработки |

Продолжение таблицы 1.2.1.1

|  |  |  |  |
| --- | --- | --- | --- |
| 2010 | MVC 2.0  Web Forms 4.0 |  |  |
| 2011 | MVC 3.0 |  |  |
| 2012 | MVC 4.0  Web Forms 4.5 |  |  |

1.2.2 Архитектура Model View Controller

На рисунке 1.2.2.1 представлена схема архитектуры MVC [6].

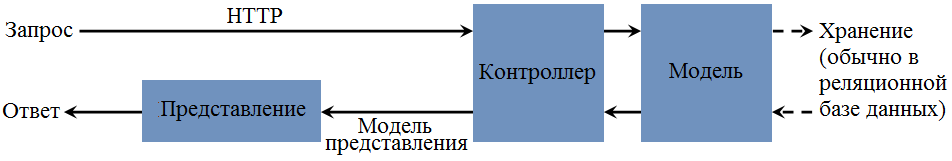


Рисунок 1.2.2.1 – Схема архитектуры MVC

Пользователь запрашивает по средствам HTTP какуюта информацию. Запрос поподает на сервер и перенапровляется приложению. Приложение в свою очередь передает управление соответсвующему контролеру по средствам маршрутизации. Контроллер достает необходимые данные из базы данных, при этом данные предствовляются в виде определенной модели (класс). Далие контролер обрабатывает полученые данные и передает их модель представлению. Модель передоваемая представлению и модель данных из базы данных не всегда совподают. Далие предстовление визиализирует модель полученую от контролера и все предстваление отпровляется клиенту в виде HTML страницы.

1.2.3 Механизм визуализации Razor

У каждого контролера (метода действия) есть модель данных (передаваемые параметры). И результат работы контролера имеет своё представление (html страницу). Таким образом, контроллер при помощи своего представления может визуализировать как модель, так и любые другие данные по средствам механизма визуализации Razor.

Для визуализации модели в представлении необходимо [3]:

- Подключить пространство имен, где эта модель описана @model <пространство имен модели>;

- Визуализировать модель @Model.<Имя пораметра моедил>.

Для визуализации любых других данных используется ViewBag [3]:

- В коде контроллера производиться инициализация данных ViewBag.newData = “BNTU”;

- В представлении производиться чтение данных @ViewBag.newData <!--BNTU-->.

Для создания формы редактирования всей модели в представлении [3]:

- @Html.EditorForModel().

Для создания формы редактирования отдельных параметров модели [3]:

- @Html.TextBoxFor(model => model.<имя поля>)

2 ПОСТАНОВКА ЗАДАЧИ

Разработать веб-приложение, которое будет предоставлять возможности интерактивного общения в сети Internet. Веб-приложение должно основываться на типе проекта ASP .NET MVC 3. Доступ к данным должен осуществляться при помощи ADO .NET Entity Framework. Разметка должна проводиться c использованием языка разметки HTML5. Описание внешнего вида веб-приложения должно осуществляться при помощи языка стилей CSS3. Сценарии должны реализовываться при помощи языка сценариев JavaScript. Для достижения поставленных задач необходимо:

- Спроектировать модели данных;

- Спроектировать алгоритмы доступа к данным;

- Спроектировать алгоритмы методов действия;

- Спроектировать разметку и стили веб-страниц;

- На основе спроектированных алгоритмов, моделей данных и разметки разработать веб-приложение реализующие возможности интерактивного общения в сети Internet.

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Модели данных

На рисунке 3.1.1 представлена логическая модель сообщений пользователей.



Рисунок 3.1.1 – Логическая модель сообщений

На рисунке 3.1.2 представлена физическая модель сообщений пользователи.



Рисунок 3.1.2 – Физическая модель сообщений

На рисунке 3.1.3 представлена логическая модель изображений пользователей.



Рисунок 3.1.3 – Логическая модель изображений

На рисунке 3.1.4 представлена физическая модель изображений.



Рисунок 3.1.4 – Физическая модель изображений

3.2 Доступ к данным

На рисунке 3.2.1 представлен алгоритм добавления сообщений.

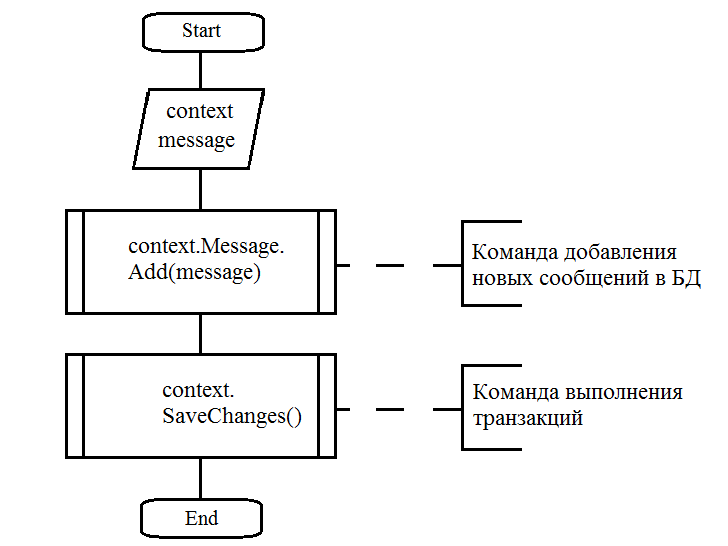


Рисунок 3.2.1 – Алгоритм добавления сообщений

На рисунке 3.2.2 представлен алгоритм добавления изображений.

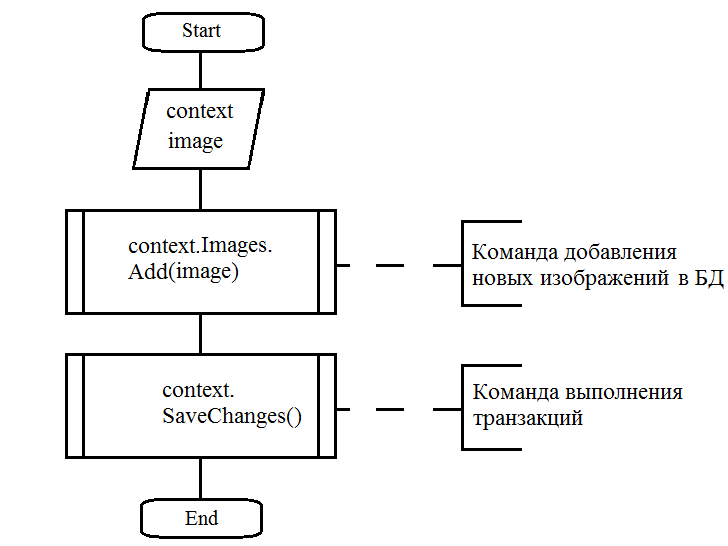


Рисунок 3.2.2 – Алгоритм добавления изображений

На рисунке 3.2.3 представлен алгоритм редактирования изображений.

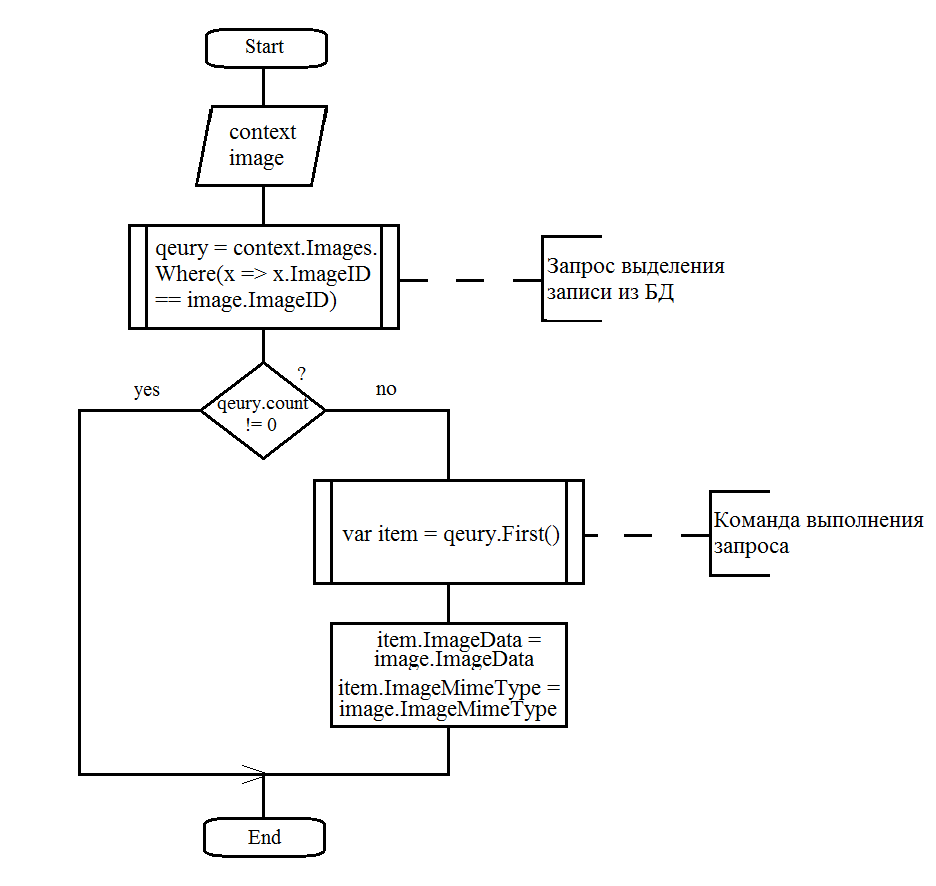


Рисунок 3.2.3 – Алгоритм редактирования изображений

3.3 Контроллеры

На рисунке 3.3.1 представлен алгоритм регистрации пользователя.

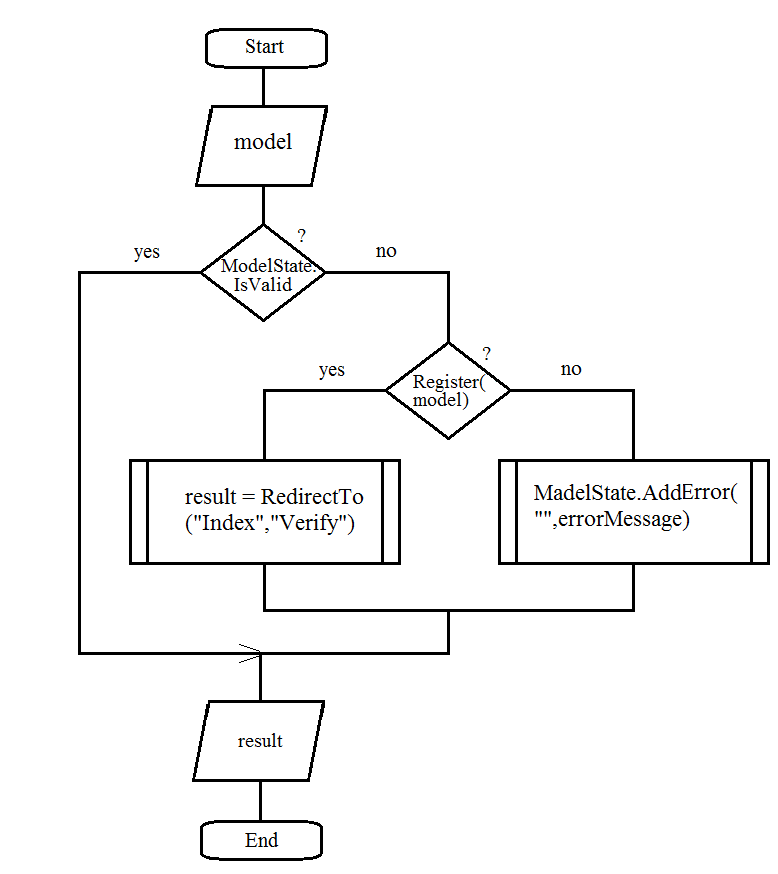


Рисунок 3.3.1 – Алгоритм регистрации пользователя

На рисунке 3.3.2 представлен алгоритм авторизации пользователя.

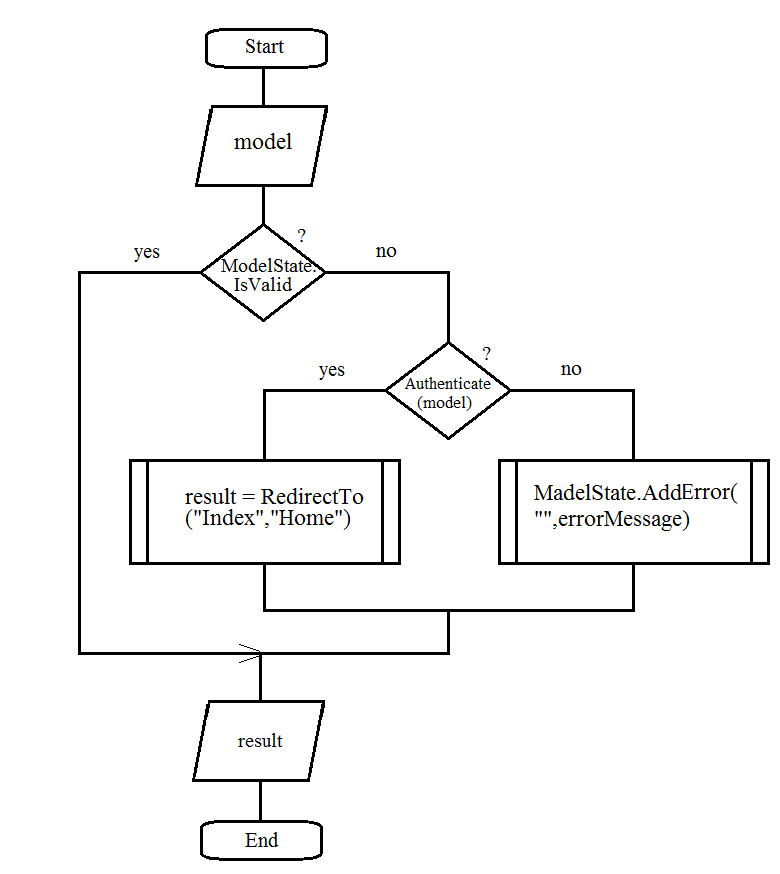


Рисунок 3.3.2 – Алгоритм авторизации пользователя

На рисунке 3.3.3 представлен алгоритм выхода пользователя из приложения.

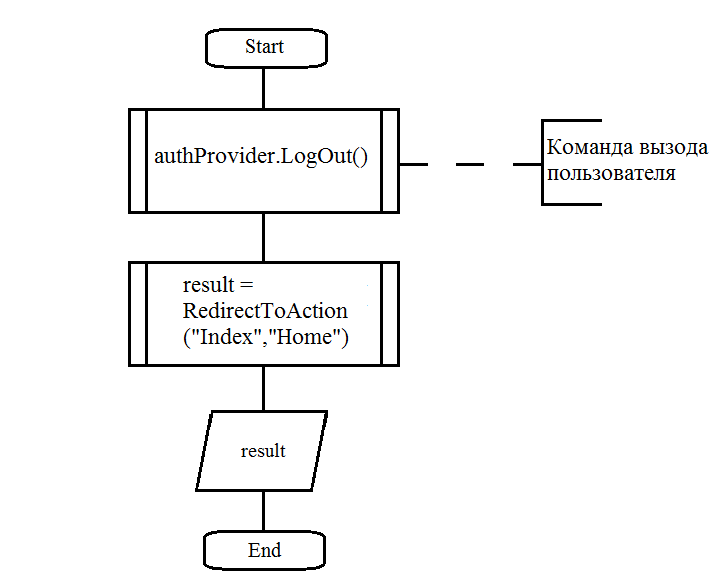


Рисунок 3.3.3 – Алгоритм выхода пользователя

На рисунке 3.3.4 представлен алгоритм верификации пользователя.

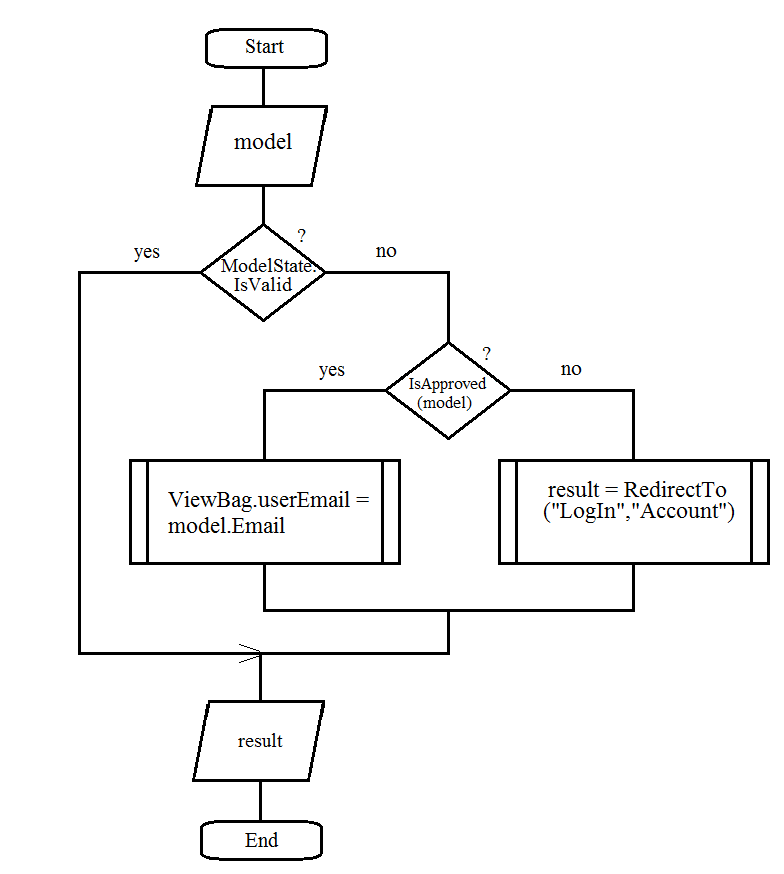


Рисунок 3.3.4 – Алгоритм верификации пользователя

На рисунке 3.3.5 представлен алгоритм подтверждения верификации пользователя.

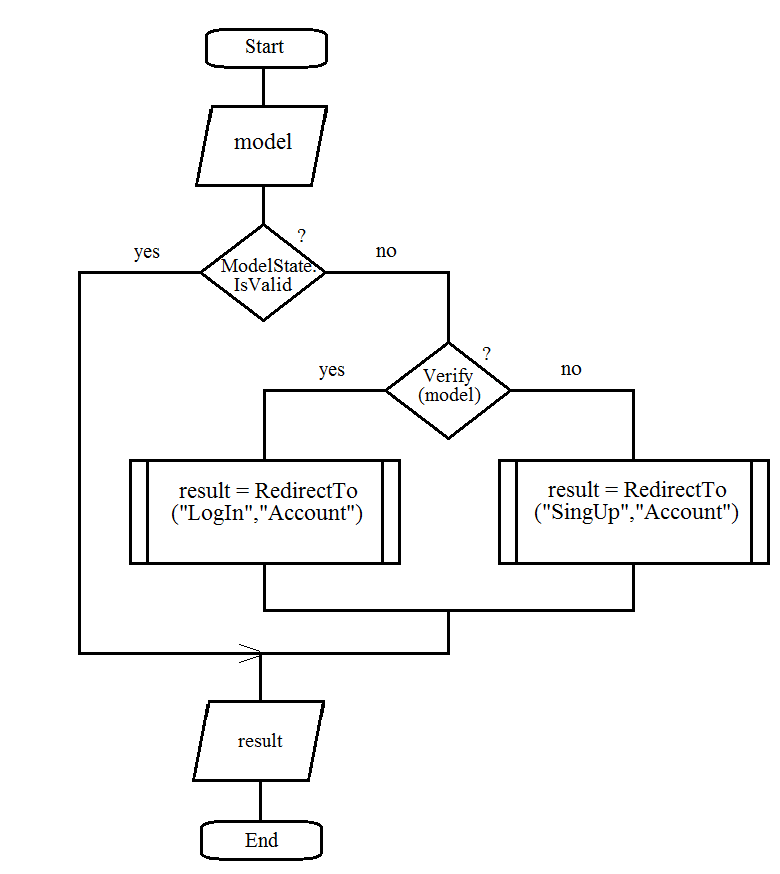


Рисунок 3.3.5 – Алгоритм подтверждения верификации пользователя

3.4 Представления

На рисунке 3.4.1 представлен внешний вид страницы регистрации пользователя.

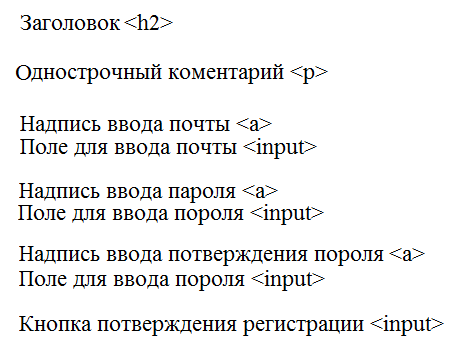


Рисунок 3.4.1 – Внешний вид страницы регистрации пользователя

На рисунке 3.4.2 представлен внешний вид страницы аутентификации пользователя.

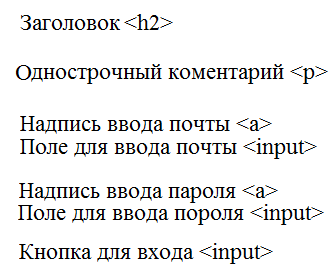


Рисунок 3.4.2 – Внешний вид страницы аутентификации пользователя

На рисунке 3.4.3 представлен внешний вид главной страницы.

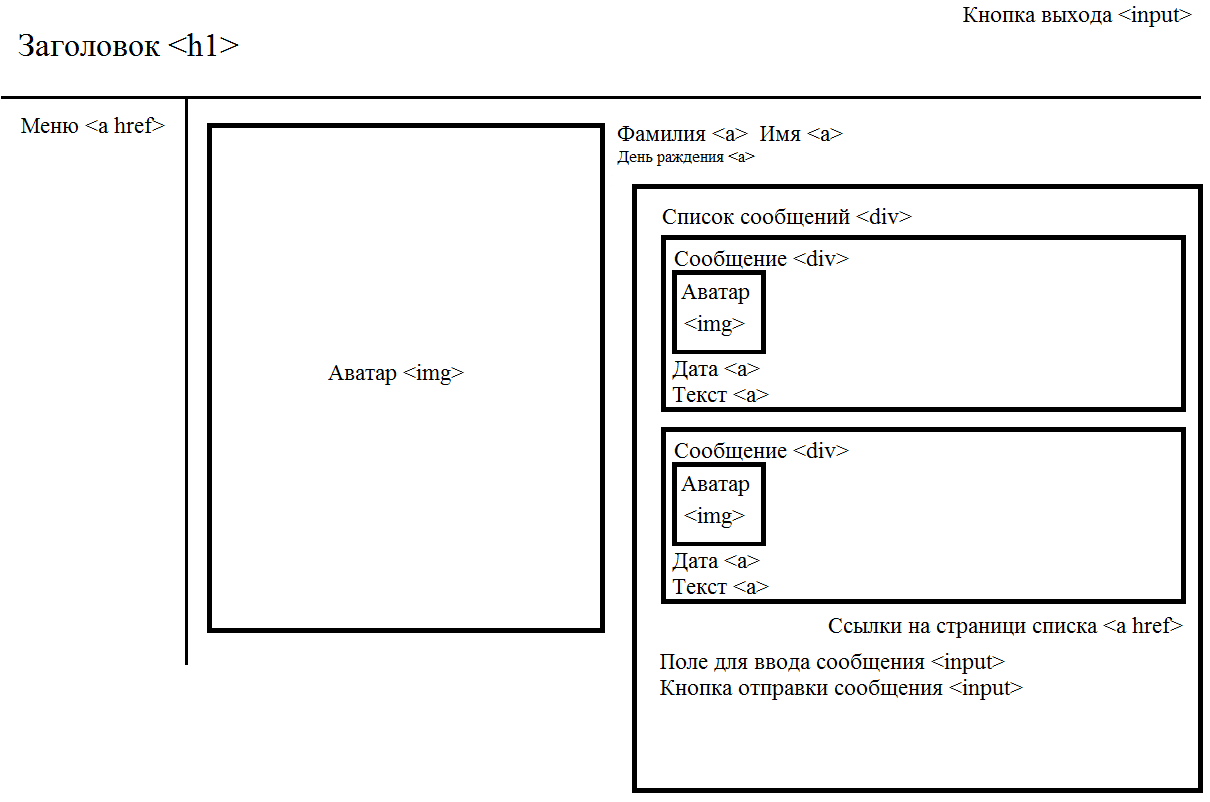


Рисунок 3.4.3 – Внешний вид главной страницы

4 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

4.1 Модели данных

Исходный код сущности Message:

public class Message

{

public int MessageID { get; set; }

public string UserID { get; set; }

public string DialogID { get; set; }

public string Text { get; set; }

public DateTime Time { get; set; }

}

Исходный код сущности Image:

public class Image

{

public string ImageID { get; set; }

public byte[] ImageData { get; set; }

public string ImageMimeType { get; set; }

}

4.2 Доступ к данным

Исходный код метода добавления сообщений в базу данных:

public void SaveMessage(Message message)

{

context.Messages.Add(message);

context.SaveChanges();

}

Исходный код метода добавления изображений:

public void SaveImage(Message image)

{

context.Images.Add(image);

context.SaveChanges();

}

Исходный код метода редактирования изображений:

internal void ChangeImage(Image image){

var qeury = Images.Where(x => x.ImageID == image.ImageID);

if (qeury.Count() != 0)

{

var item = qeury.First();

item.ImageData = image.ImageData;

item.ImageMimeType = image.ImageMimeType;

}

}

4.3 Контроллеры

Исходный код метода регистрации пользователя контроллера Account:

[HttpPost]

public ActionResult SignUp(SignUpModel model)

{

if (ModelState.IsValid)

{

string errorMessage;

if (authProvider.Register(model.Email, model.Password, model.Email, model.PasswordQuestion, model.PasswordAnswer, false, null, out errorMessage))

{

if (Url != null)

{

return RedirectToAction("Index", "Verify", new ActivateModel { Email = model.Email });

}

else

{

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

}

}

else

{

ModelState.AddModelError("", errorMessage);

}

}

return View(model);

}

Исходный код метода авторизации пользователя контроллера Account:

[HttpPost]

public ActionResult LogIn(LogInModel model, string returnUrl)

{

if (ModelState.IsValid)

{

if (authProvider.Authenticate(model.UserName, model.Password, true))

{

return Redirect(returnUrl ?? Url.Action("Index", "Home"));

}

else

{

ModelState.AddModelError("", "Incorrect username or password");

}

}

return View();

}

Исходный код метода выхода пользователя из приложения контроллера Account:

public ActionResult LogOut()

{

authProvider.LogOut();

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

}

Исходный код метода верификации пользователя контроллера Verify:

[OutputCache(Duration = 60)]

public ActionResult Index(ActivateModel model)

{

if (model.Email != null)

{

if (verifyProvider.IsApproved(model.Email))

{

MailAddress mailAddress = new MailAddress(model.Email);

if (mailAddress != null)

{

ViewBag.userHost = mailAddress.Host;

ViewBag.userEmail = model.Email;

return View();

}

}

}

return RedirectToAction("LogIn","Account");

}

Исходный код метода подтверждения верификации пользователя контроллера Verify:

public ActionResult Verify(ActivateModel model)

{

if (model.Email != null && model.SecretCode != null)

{

if (verifyProvider.Verify(model.Email, model.SecretCode))

{

return RedirectToAction("LogIn", "Account");

}

}

return RedirectToAction("SignUp", "Account");

}

4.4 Представления

На рисунке 4.4.1 представлен внешний вид страницы регистрации пользователя.

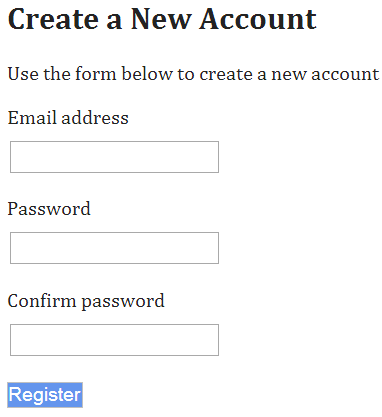


Рисунок 4.4.1 – Внешний вид страницы регистрации пользователя

На рисунке 4.4.2 представлен внешний вид страницы аутентификации пользователя.

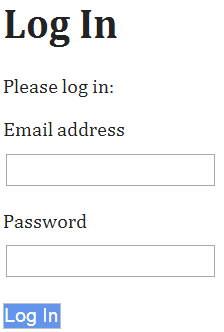


Рисунок 4.4.2 – Внешний вид страницы аутентификации пользователя

На рисунке 4.4.3 представлен внешний вид главной страницы.

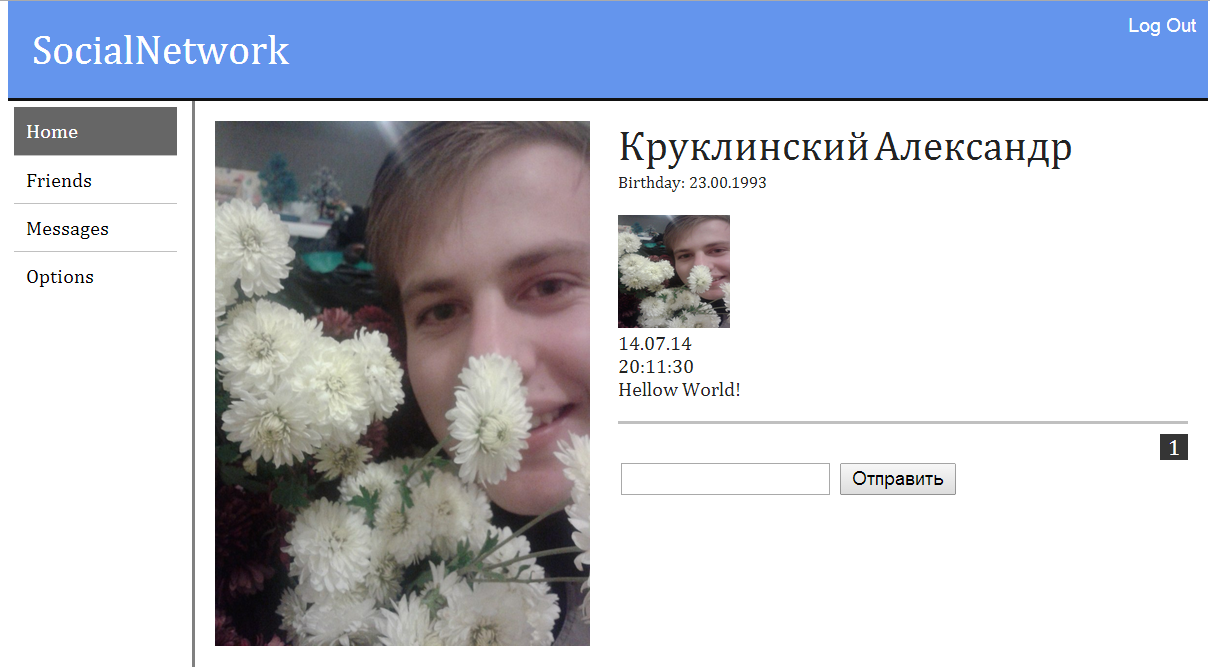


Рисунок 4.4.3 – Внешний вид главной страницы

5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

5.1 Модульное тестирование

Для улучшения качества программного обеспечение было решено провести модульное тестирование основных компонентов веб-приложения. Список тестов и результаты модульного тестирования представлены на рисунок 5.1.1

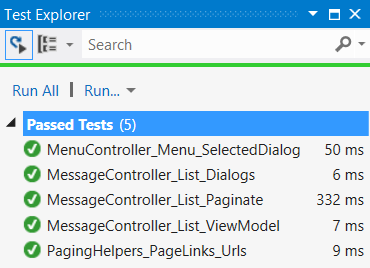


Рисунок 5.1.1 – Результаты модульного тестирования

5.2 Функциональное тестирование.

Регистрация нового пользователя представлена на рисунке 5.2.1.

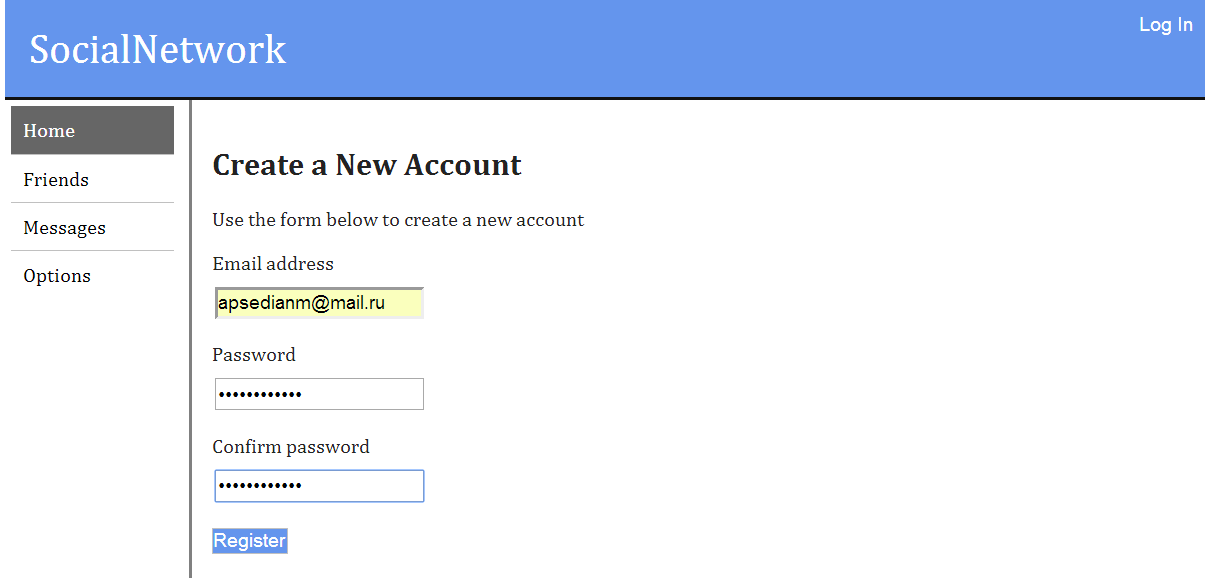


Рисунок 5.2.1 – Регистрация нового пользователя

На рисунке 5.2.2 представлены результаты регистрации.

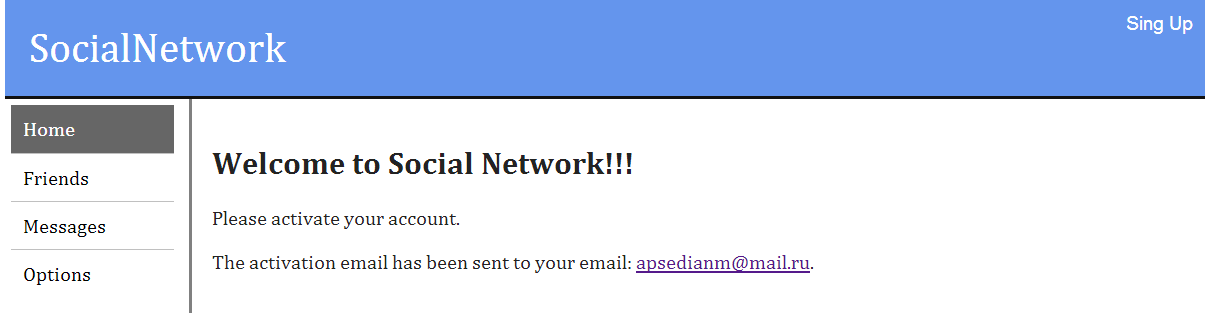


Рисунок 5.2.2 – Результаты регистрации

На рисунке 5.2.3 представлены результаты верификации зарегистрированного пользователя.

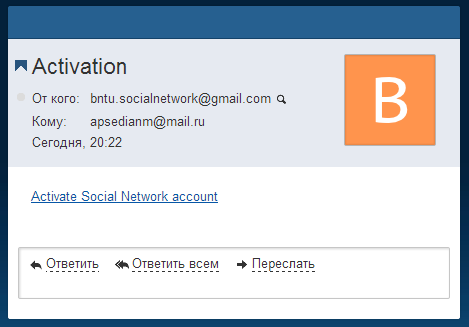


Рисунок 5.2.3 – Результаты верификации

Авторизация зарегистрированного пользователя представлена на рисунке 5.2.4.

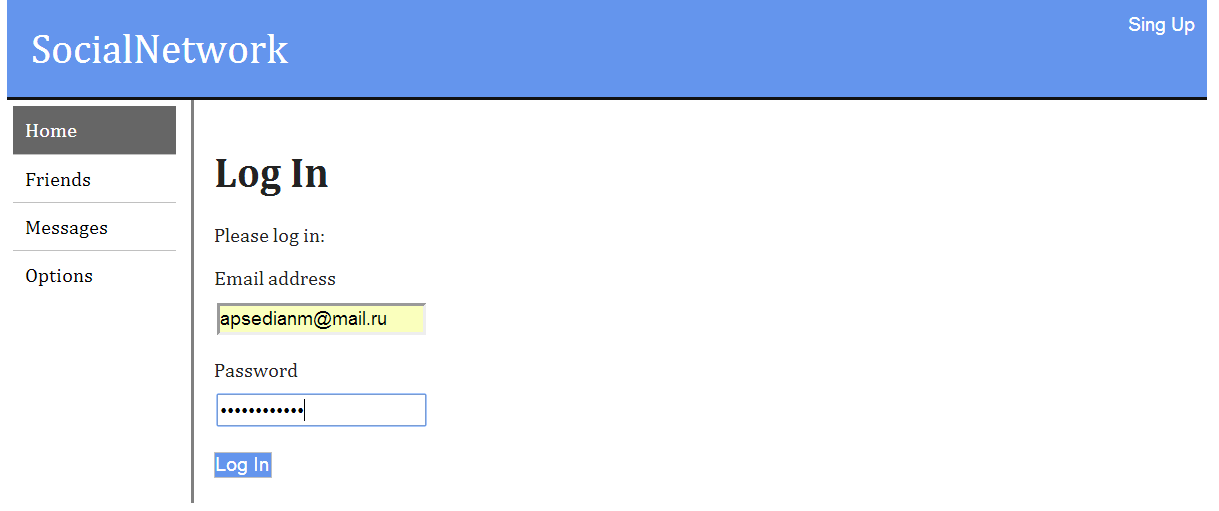


Рисунок 5.2.4 – Авторизация пользователя

На рисунке 5.2.5 представлены результаты авторизации.

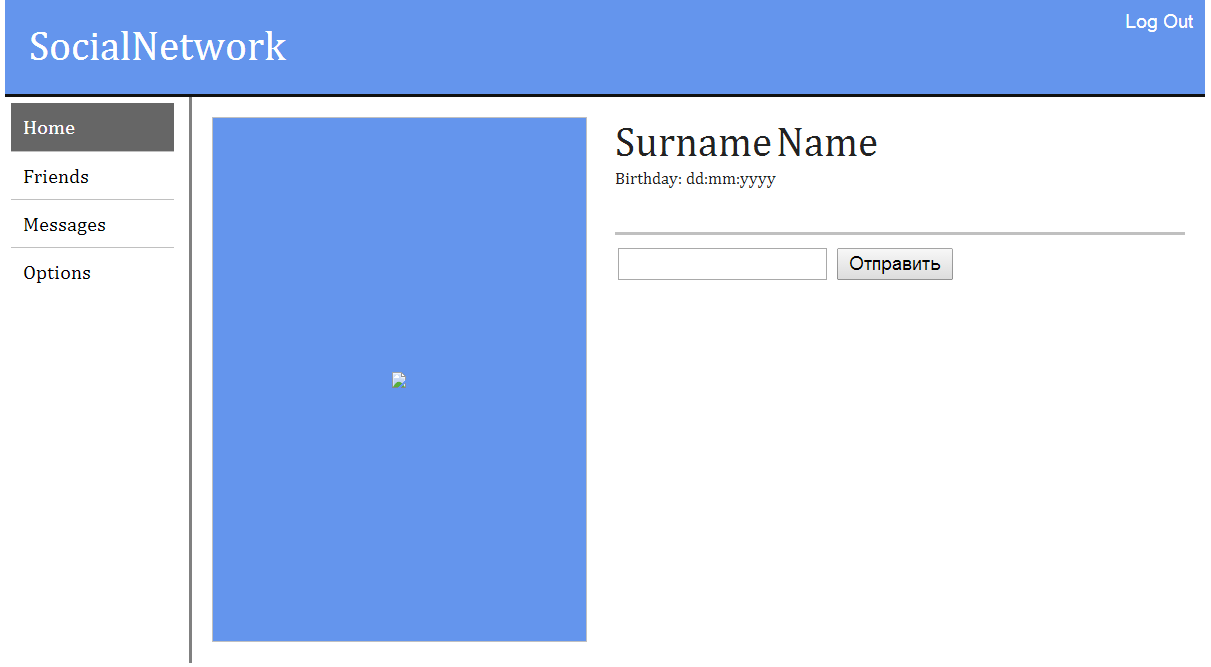


Рисунок 5.2.5 – Результаты авторизации

ВЫВОДЫ

В периуд производственной практики преобрел навыки разработки современных веб-приложений с использованием технологии ASP .NET MVC Framework. Так же изучил вопросы обеспечения правил техники безопасности и противопожарной техники при организации практик на кафедре ПОВТиАС.

Разработанное веб-приложение прошло функциональное и модульное тестирования. Модульное тестирование, на всем протяжении разработки, облегчало процесс нахождения и исправления ошибок. Функциональное тестирование позволило оценивать правильность работы реализованных алгоритмов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. <http://ru.wikipedia.org>
2. http://www.rosbalt.ru
3. <http://msdn.microsoft.com>
4. <http://www.liveinternet.ru>
5. <https://vk.com>
6. Адам Ф., Стивен С. ASP.NET MVC 3 Framework с примерами на C# для профессионалов. - М.: Вильямс, 2012. – 622с.
7. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. Для программной документации.