**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**

**ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ  
ЮЖНОГО ФЕДЕРАЛНОГО УНИВЕРСИТЕТА в г. Таганроге**

Факультет автоматики и вычислительной техники

Кафедра «Математического обеспечения и применения ЭВМ»

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Курсовая работа**

по дисциплине «Программирование в Базах Данных»

**«Создание Web-приложения с использованием**

**ASP.NET MVC и Entity Framework»**

Выполнил:

студент группы А-91

Чепурной А.В.

Проверил:

Доцент кафедры МОП ЭВМ

Хашковский В. В.

Оценка

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

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ г.

Таганрог 2014

# Аннотация

В данной курсовой работе рассмотрен проект, который описывает этапы создания Web-приложения, которое является форумом, позволяющим делиться информацией между пользователями в рамках данного ресурса. Серверная часть приложения написана на ASP.NET с использованием языка программирования C# и движка Razor. Клиентская часть использует язык разметки HTML, язык скриптов JavaScript и CSS – каскадные таблицы стилей. Рассмотрена база данных, разработанная в ходе работы, обусловлен выбор средств разработки, дается краткое описание основных частей проекта, а также приведена полная инструкция пользователя.

Оглавление

[Аннотация 2](#_Toc389426595)

[Постановка задачи 4](#_Toc389426596)

[Введение 5](#_Toc389426597)

[1. Проектирование сервиса 9](#_Toc389426598)

[1.1. Описание функциональных возможностей сервиса. 9](#_Toc389426599)

[1.2. Описание структуры разработанной базы данных 10](#_Toc389426600)

[1.3. Диаграмма базы данных 12](#_Toc389426601)

[2. Описание программного продукта 13](#_Toc389426602)

[2.1. Программно-аппаратные требования 13](#_Toc389426603)

[2.2. Язык и среда программирования 13](#_Toc389426604)

[2.3. Структура приложения 14](#_Toc389426605)

[3. Описание интерфейса и инструкция пользования 15](#_Toc389426606)

[3.1. Описание интерфейса незарегистрированного пользователя 15](#_Toc389426607)

[3.2. Описание интерфейса зарегистрированного пользователя 19](#_Toc389426608)

[3.3. Описание интерфейса модератора 24](#_Toc389426609)

[3.4. Описание интерфейса администратора 25](#_Toc389426610)

[Заключение 27](#_Toc389426611)

[Библиографический список 28](#_Toc389426612)

[Приложение 1. Ход разработки проекта 29](#_Toc389426613)

[Приложение 2. Частичный листинг приложения 32](#_Toc389426614)

# Постановка задачи

Необходимо разработать Web-приложение, которое является форумом, позволяющим делиться информацией между пользователями в рамках данного ресурса. Пользователи должны без труда создавать новые темы, добавлять комментарии и оценивать уже существующие комментарии посредством выставления голосов. Пользователи должны иметь возможность кастомизировать свой аккаунт путем изменения настроек приватности и загрузки собственного аватара. Приложение должно содержать модули администрирования и модерирования, позволяющие производить настройку приложения должным образом.

Серверная часть приложения должна быть написана на ASP.NET с использованием языка программирования C# и движка Razor вкупе с Entity Framework. Клиентская часть должна использовать язык разметки HTML, язык скриптов JavaScript и CSS – каскадные таблицы стилей.

# Введение

**ASP** (англ. *Active Server Pages* — «активные серверные страницы») — технология, предложенная компанией Microsoft в 1996 году для создания Web-приложений. Эта технология основана на внедрении в обыкновенные веб-страницы специальных элементов управления, допускающих программное управление.

По своей сути, **ASP** — это технология динамического создания страниц на стороне сервера, приблизившая проектирование и реализацию Web-приложений к той модели, по которой проектируются и реализуются обычные приложения.

Для реализации приложений **ASP** используются языки сценариев (VBScript или JScript). Также допускается применение COM-компонентов.

Технология **ASP** разработана для операционных систем из семейства Windows NT и функционирует под управлением веб-сервера Microsoft IIS.

Технология **ASP** получила своё развитие в виде ASP.NET — технологии создания веб-приложений, основанной уже на платформе Microsoft .NET.

Хотя **ASP.NET** берёт своё название от старой технологии Microsoft ASP, она значительно от неё отличается. Microsoft полностью перестроила ASP.NET, основываясь на Common Language Runtime (CLR), которая является основой всех приложений Microsoft .NET. Разработчики могут писать код для ASP.NET, используя практически любые языки программирования, входящие в комплект .NET Framework (C#, Visual Basic.NET и JScript .NET). ASP.NET имеет преимущество в скорости по сравнению со скриптовыми технологиями, так как при первом обращении код компилируется и помещается в специальный кэш, и впоследствии только исполняется, не требуя затрат времени на парсинг, оптимизацию, и т. д.

**Model-view-controller** (**MVC**, «модель-представление-поведение», «модель-представление-контроллер», «модель-вид-контроллер») — схема использования нескольких шаблонов проектирования, с помощью которых модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Данная схема проектирования часто используется для построения архитектурного каркаса, когда переходят от теории к реализации в конкретной предметной области.

Рис. 1. Архитектура MVC

**ADO.NET Entity Framework** (EF) — объектно-ориентированная технология доступа к данным, является object-relational mapping(ORM) решением для .NET Framework от Microsoft. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как ADO.NET Data Services (*Astoria*), так и связка из Windows Communication Foundation и Windows Presentation Foundation, позволяющая строить многоуровневые приложения, реализуя один из шаблонов проектирования MVC, MVP или MVVM.

В данной курсовой работе использовался подход «Code-First» при работе с Entity Framework.

**Code-First** – новая возможность, стоящая в одном ряду с «Model-First» и «Database-First». Можно сначала написать код, описывающий классы-модели, а потом фреймворк автоматически создаст БД по такому коду.  
Самое приятное, что виды отношений будут распознаны – достаточно определить просто ссылки на объекты для 1:1, ICollection для отношения 1:n, и взаимные ICollection для m:n. В этом случае промежуточная таблица также создастся автоматически. Для обеспечения «ленивой» загрузки хватит ключевого слова virtual в определении свойства.

Также полностью поддерживаются атрибуты из пространства имен System.ComponentModel.DataAnnotations.

Основные из них:

* [Key] – указывает на то, что данное свойство будет ключевым.
* [NotMapped] – не отображать свойство на колонки в БД.
* [Column(“columnName”, TypeName=”typeName”] – указывает колонку в БД, на которую отображается поле и его тип.

Полезно когда в классе есть свойство типа byte[], хранящий, например рисунок, а в базе это отобразится на специальный тип image.   
Однако нельзя указать подмножество типа таким образом: например для строк по умолчанию в базе выбирается тип nvarchar(MAX) и с помощью этого атрибута нельзя явно указать длину поля nvarchar(30) [MinLength], [MaxLength], [Required] и другие для обеспечения проверки данных.

Code-First заточен под работу с новой версией SQL Server CE 4.0. Поддержка этой базы данных включена в ASP.NET MVC с версии 3.

Приложение было написано в соответствии с таким стандартным паттерном разработки программных приложений, как Repository.

**Repository** посредничает между уровнями области определения и распределения данных (domain and data mapping layers), используя интерфейс, схожий с коллекциями для доступа к объектам области определения.

Система со сложной моделью области определения может быть упрощена при помощи дополнительного уровня, например Data Mapper, который бы изолировал объекты от кода доступа к БД. В таких системах может быть полезным добавление ещё одного слоя абстракции поверх слоя распределения данных (Data Mapper), в котором бы был собран код создания запросов. Это становится ещё более важным, когда в области определения множество классов или при сложных, тяжелых запросах. В таких случаях добавление этого уровня особенно помогает сократить дублирование кода запросов.

Паттерн Repository посредничает между слоем области определения и слоем распределения данных, работая, как обычная коллекция объектов области определения. Объекты-клиенты создают описание запроса декларативно и направляют их к объекту-репозиторию (Repository) для обработки. Объекты могут быть добавлены или удалены из репозитория, как будто они формируют простую коллекцию объектов. А код распределения данных, скрытый в объекте Repository, позаботится о соответсвующих операциях в незаметно для разработчика. В двух словах, паттерн Repository инкапсулирует объекты, представленные в хранилище данных и операции, производимые над ними, предоставляя более объектно-ориентированное представление реальных данных. Repository также преследует цель достижения полного разделения и односторонней зависимости между уровнями области определения и распределения данных.

# Проектирование сервиса

## Описание функциональных возможностей сервиса.

Незарегистрированные пользователи практически не имеют никаких прав и могут лишь просматривать уже добавленные темы и оставленные комментарии к ним.

Зарегистрированные пользователи могут создавать темы (не больше одной темы в час), оставлять комментарии к ним (другие пользователи увидят их только после проверки модератором или администратором) и оценивать чужие комментарии. Из оценок к комментариям складывается рейтинг пользователя. Кроме того, зарегистрированным пользователям доступен раздел «Личный кабинет», где они могут кастомизировать свой аккаунт путем изменения настроек приватности и загрузки собственного аватара. Также зарегистрированным пользователям можно просматривать информацию о других зарегистрированных пользователях.

Обычному зарегистрированному пользователю может быть присвоена роль модератора главным администратором сайта (о котором позднее). Модератор обладает расширенными возможностями обычного пользователя. Так он может создавать темы без ограничения по времени, оставлять не нуждающиеся в проверке комментарии. Основной обязанностью модератора является премодерация сообщений. Также он может удалять существующие темы.

Главный администратор сайта обладает всеми возможностями модератора. Также он имеет право управлять пользователями, например, назначать модератором или снимать с этой должности. Администратор может заблокировать пользователя или восстановить уже заблокированного. Только ему разрешено управлять разделами форума, как то удалять раздел, переименовывать существующий раздел либо создавать новые разделы.

## Описание структуры разработанной базы данных

Ниже представлены лишь основные сущности, использованные при разработке приложения

Описание сущности «Пользователь» (UserProfile) представлено в таблице 1.

Таблица 1.Сущность «Пользователь».

|  |  |
| --- | --- |
| **Название поля** | **Описание поля** |
| UserId | Уникальный идентификатор |
| UserName | Имя пользователя |
| Mobile | Телефон пользователя |
| Email | Электронный адрес пользователя |
| RegistrationDate | Дата регистрации |
| ImageData | Аватар пользователя |
| ImageMimeType | Тип загружаемой картинки |

Описание сущности «Настройка пользователя» (UserProperty) представлено в таблице 2.

Таблица 2. Сущность «Настройка пользователя».

| **Название поля** | **Описание поля** |
| --- | --- |
| UserId | Уникальный идентификатор |
| ShowEmail | Показывать почту |
| ShowMobile | Показывать телефон |

Описание сущности «Раздел» (Section) представлено в таблице 3.

Таблица 3. Сущность «Раздел».

|  |  |
| --- | --- |
| **Название поля** | **Описание поля** |
| SectionId | Уникальный идентификатор |
| SectionTitle | Название раздела |

Описание сущности «Тема» (Theme) представлено в таблице 4.

Таблица 4. Cущность «Тема».

|  |  |
| --- | --- |
| **Название поля** | **Описание поля** |
| ThemeId | Уникальный идентификатор |
| SectionId | Раздел, к которому относится тема |
| UserId | Пользователь, создавший тему |
| ThemeTitle | Заголовок темы |
| ThemeText | Содержание темы |
| CreationDate | Дата создания |

Описание сущности «Комментарий» (Comment) представлено в таблице 5.

Таблица 5. Сущность «Комментарий».

|  |  |
| --- | --- |
| **Название поля** | **Описание поля** |
| CommentId | Уникальный идентификатор |
| ThemeId | Тема, к которой относится комментарий |
| UserId | Пользователь, создавший комментарий |
| CommentText | Содержание комментария |
| CreationDate | Дата создания |
| IsAdmitted | Допущен комментарий или нет |

Описание сущности «Лайк» (Like) представлено в таблице 6.

Таблица 6. Сущность «Лайк».

|  |  |
| --- | --- |
| **Название поля** | **Описание поля** |
| LikeId | Уникальный идентификатор |
| UserId | Идентификатор голосующего |
| CommentId | Идентификатор сообщения |
| Vote | Голос |

Описание сущности «Роль» (Role) представлено в таблице 7.

Таблица 7. Сущность «Роль».

|  |  |
| --- | --- |
| **Название поля** | **Описание поля** |
| RoleId | Уникальный идентификатор |
| RoleName | Роль |

Описание сущности «Пользователи в ролях» (UsersInRoles) представлено в таблице 8.

Таблица 8. Сущность «Пользователи в ролях».

|  |  |
| --- | --- |
| **Название поля** | **Описание поля** |
| UserId | Идентификатор пользователя |
| RoleId | Идентификатор роли |

## Диаграмма базы данных

Данная диаграмма создана на основании базы данных, автоматически созданной средствами Entity Framework. В ней присутствуют все главные таблицы, описанные выше, а также вспомогательные, необходимые для разрешения взаимосвязей между сущностями.

# Описание программного продукта

## Программно-аппаратные требования

Так как приложение является Web- сервисом, минимальные требования к программно-аппаратным средствам представлены для клиента и для сервера.

Со стороны сервера:

* ОС (операционная система): операционные системы семейства Windows с поддержкой развертывания IIS;
* ОЗУ объемом 1 ГБ (1,5 ГБ для работы на виртуальной машине);
* 5 Гб доступного пространства на жестком диске;
* .NET Framework 4.0, 4.5, 4.5.1;
* HTTP-сервер IIS;
* СУБД: MS SQL;
* Скоростное интернет - соединение.

Со стороны клиента:

* ОС: Любая операционная система, которая поддерживает интернет соединение и с возможностью установки браузера;
* Любой браузер: Chrome, Mazilla Firefox, Safari, Opera, IE( версия 9 и выше, версии ниже возможны искажения);
* Клавиатура, мышь.

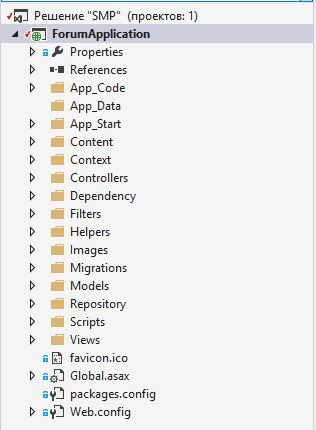
## Язык и среда программирования

Данный программный продукт был реализован при помощи языка программирования C# и движка Razor с использованием фреймворка ASP.NET MVC на серверной стороне и языка JavaScript (в частности с использованием jQuery) со стороны клиента. В качестве среды разработки была выбрана Microsoft Visual Studio 2013.

Выбор данных средств обусловлен удобностью их использования.

## Структура приложения

Само приложение представлено в виде решения, которое включает в себя ASP.NET MVC4 проект.

**App\_Data** – содержит локальную базу данных.

**App\_Code** –  позволяет создавать настраиваемые классы и другие файлы с исходным кодом и использовать их в создаваемом веб-приложении без необходимости их независимой компиляции.

**App\_Start** – содержит базовые классы, необходимые для корректной работы MVC приложения.

**Content** – содержит .css файлы и изображения.

**Context** – содержит такие классы, как Theme, Comment, Like и т.д.

**Controllers** – содержит классы контроллеров MVC приложения.

**Dependency & Repository** – содержит классы, необходимые для реализации паттерна репозиторий.

**Filters** – содержит фильтры приложения, например, фильтр авторизации.

**Helpers** – содержит статический класс, содержащий методы расширения.

**Migrations** – содержит классы миграций, созданных и примененных к базе данных.

**Models** – содержит модели MVC приложения.

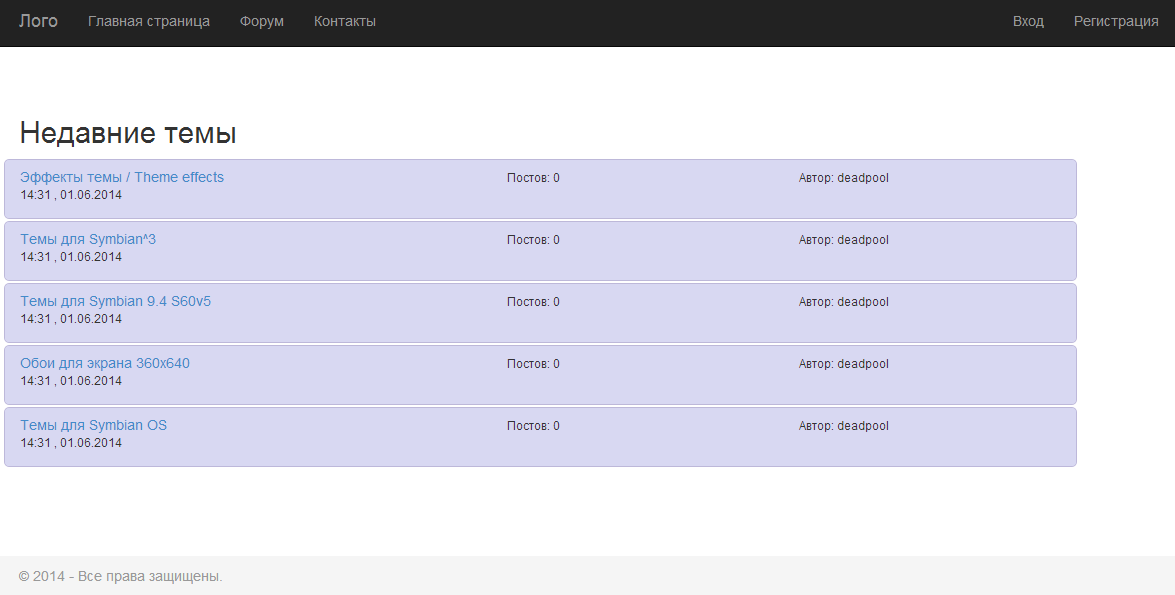
**Scripts –** cодержит скрипты MVC приложения.

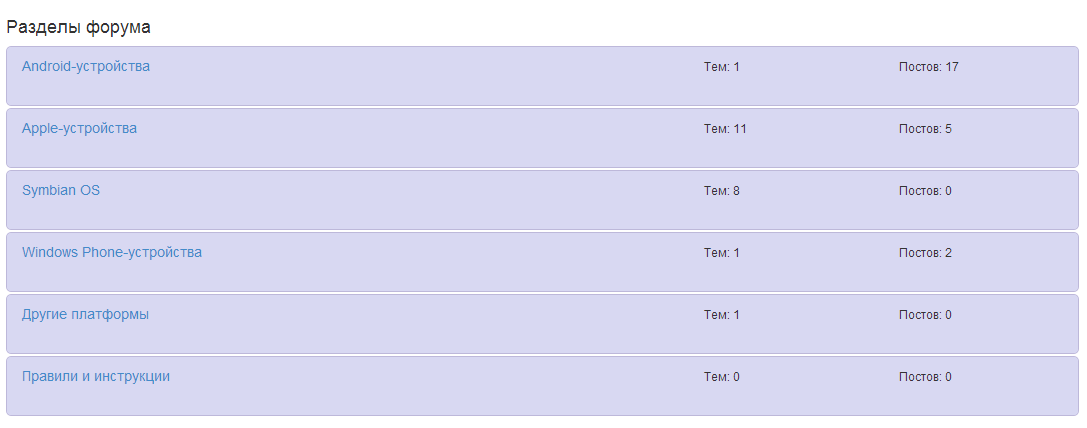
**Views** – содержит представления MVC приложения.

**Global.asax** - содержит код для отклика на события уровня приложения и уровня сеанса.

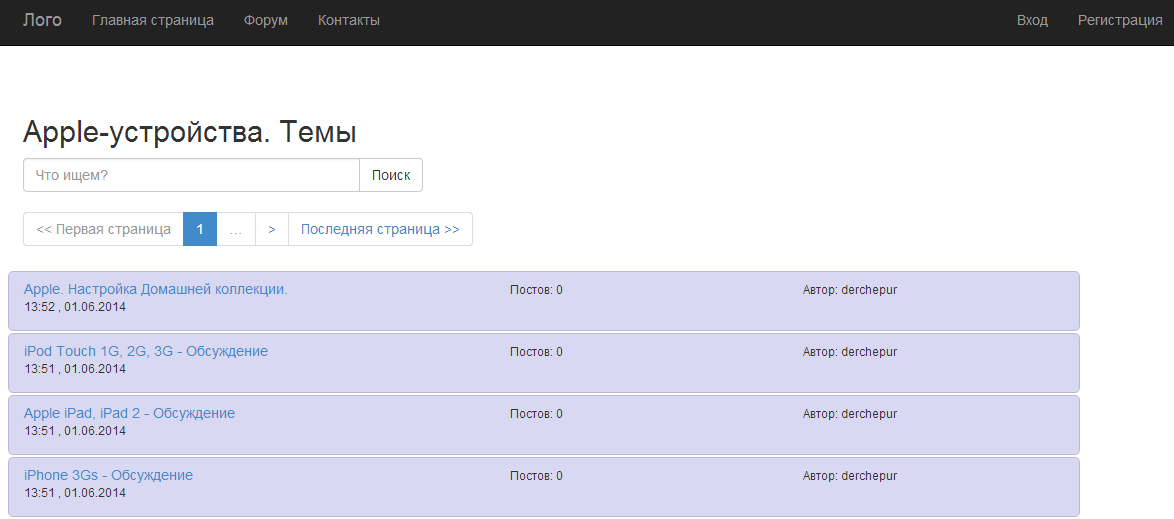
# Описание интерфейса и инструкция пользования

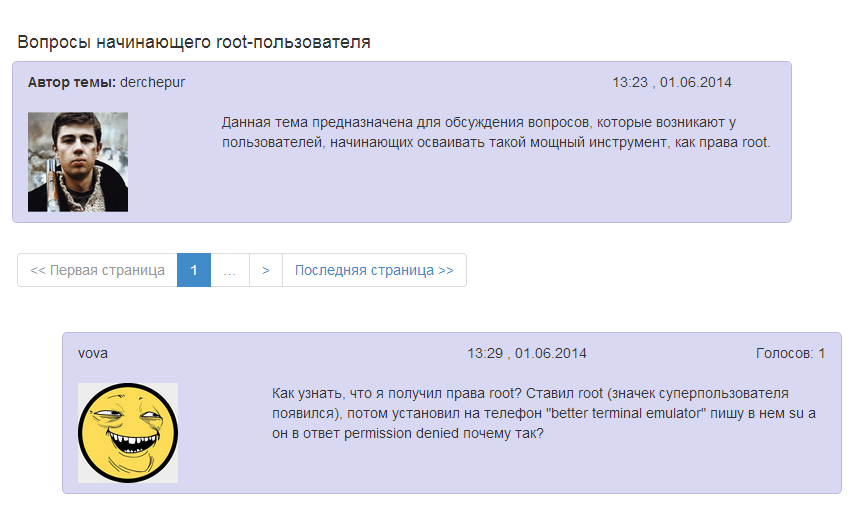
## Описание интерфейса незарегистрированного пользователя

Незарегистрированный пользователь при входе на сайт видит главную страницу с последними добавленными темами.

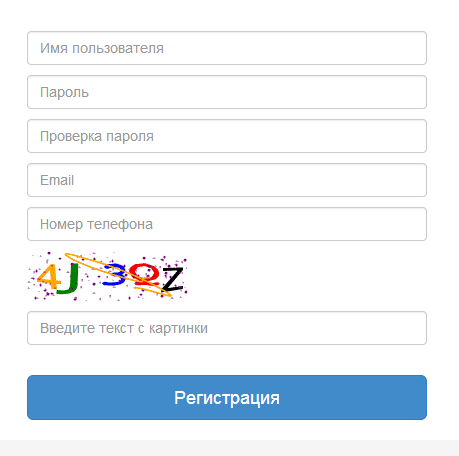
Раздел с секциями форума для незарегистрированного пользователя выглядит следующим образом.

Раздел с темами для незарегистрированного пользователя выглядит следующим образом.

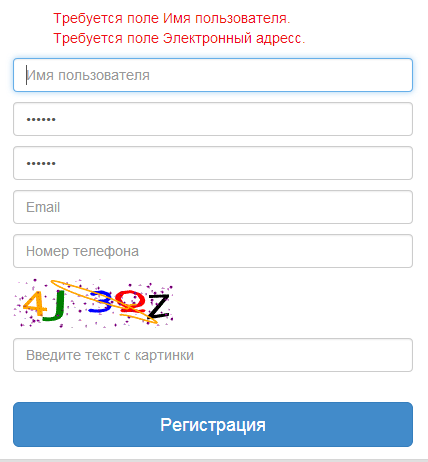


При щелчке по конкретной теме незарегистрированный пользователь увидит следующее.

Также незарегистрированный пользователь может зарегистрироваться, перейдя по соответствующей ссылке.

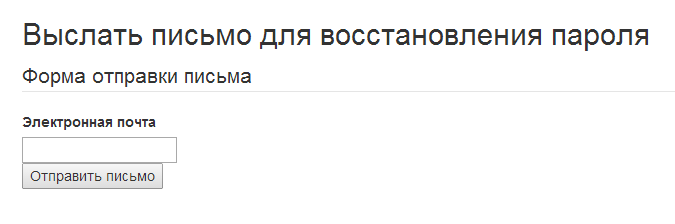


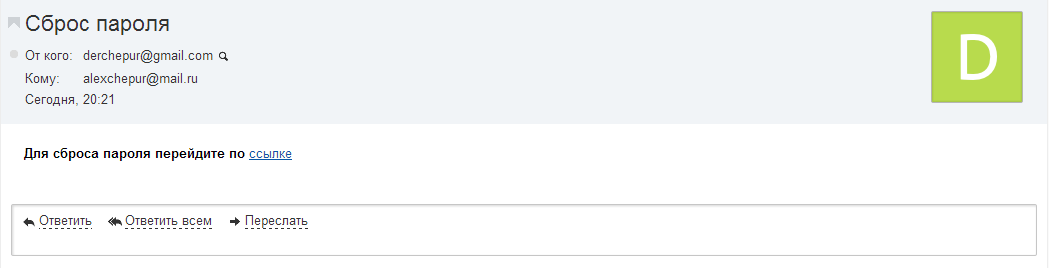
При введении некорректных данных, пользователь будет получать информацию, где и как он ошибся.



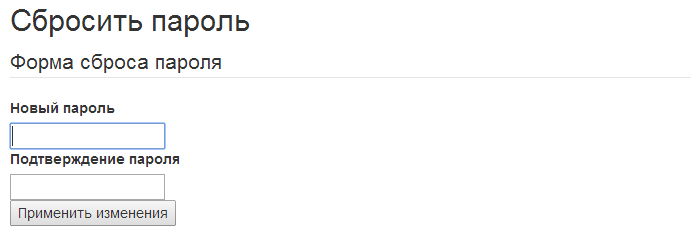
Было принято решение использовать капчу для предотвращения регистрации ботов на форуме. После регистрации пользователь считается активным и может принимать участие в жизнедеятельности ресурса.

Однако если пользователь забыл свои данные для входа, то он может их восстановить, воспользовавшись формой восстановления пароля.



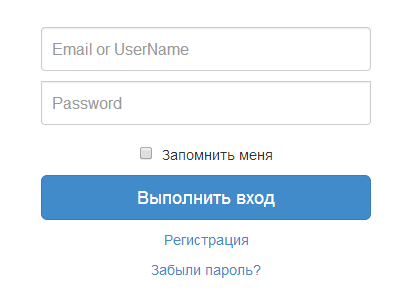
В этом случае на его почту будет высланы инструкции по смене пароля.

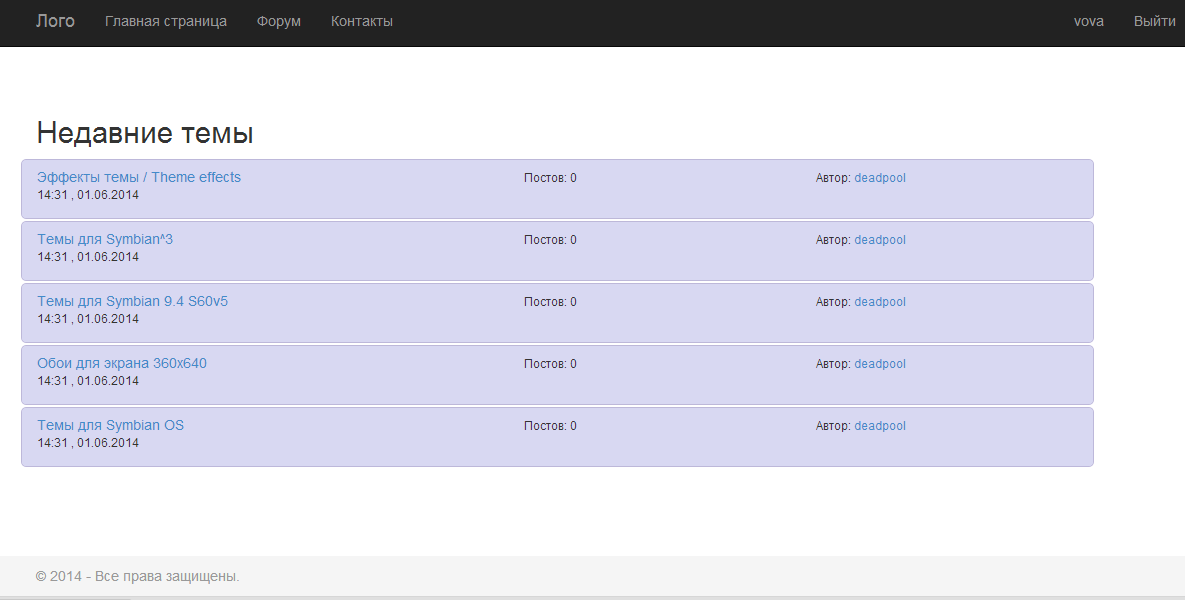
После того, как пользователь щелкнет по ссылке, перед ним откроется следующая форма и он сможет поменять пароль.



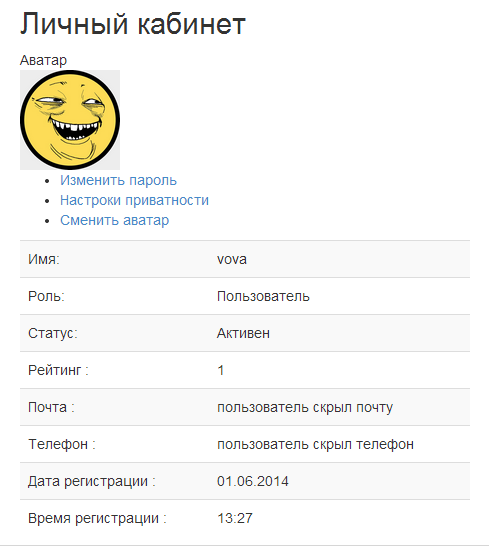
## Описание интерфейса зарегистрированного пользователя

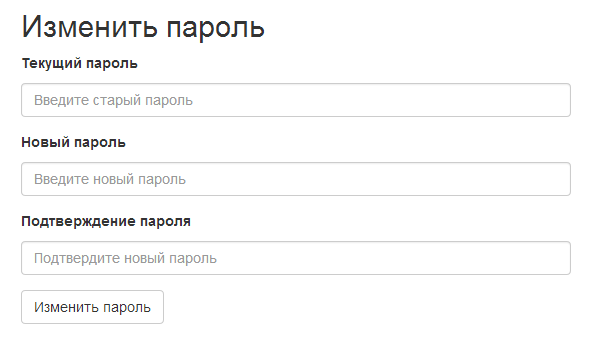
Так выглядит форма входа.

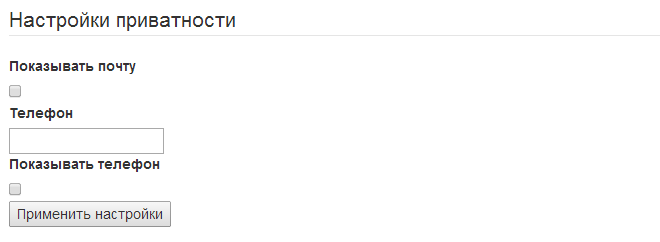


Зарегистрированный пользователь также видит главную страницу, но с одним отличием: он имеет возможность перейти в личный кабинет.

Перейдя на страницу личного кабинета, пользователь видит следующее.

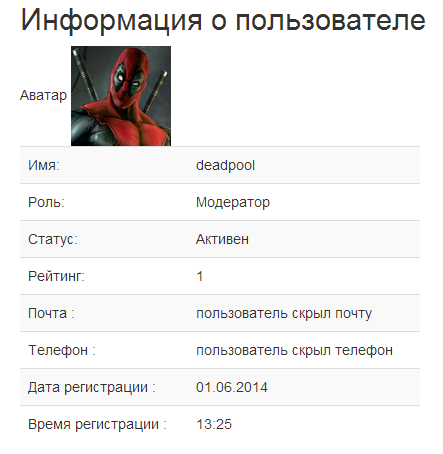


Пользователь может сменить аватар. Также пользователь может сменить пароль, но для этого необходимо помнить свой прошлый пароль.

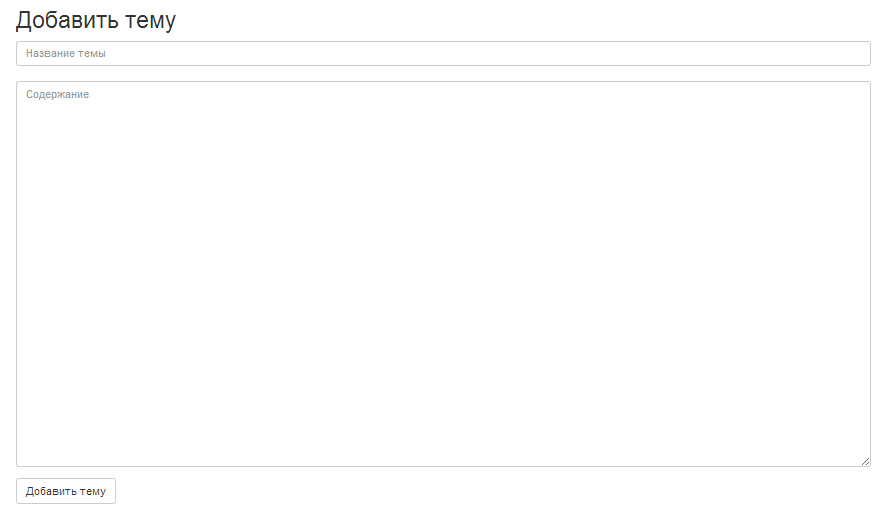
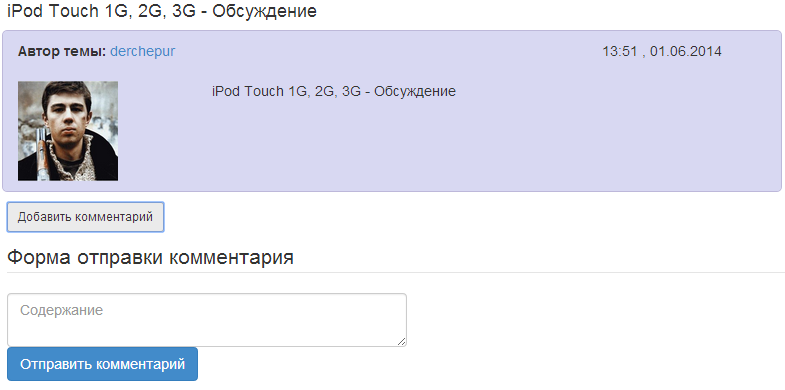
Из личного кабинета зарегистрированный пользователь может изменить настройки приватности.

Также в личном кабинете отображаются темы, созданные данным пользователем.

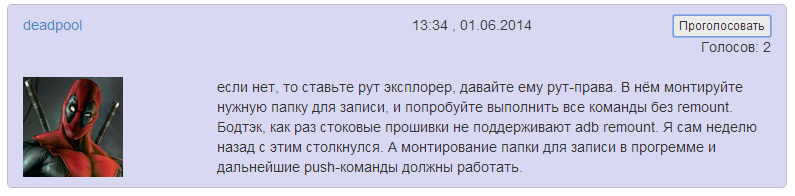
При щелчке на другом пользователе будет выведена информация о нем (наподобие личного кабинета, но без настроек аккаунта).



Также отличительной особенностью зарегистрированного пользователя является то, что он может добавлять темы.

А еще оставлять комментарии.

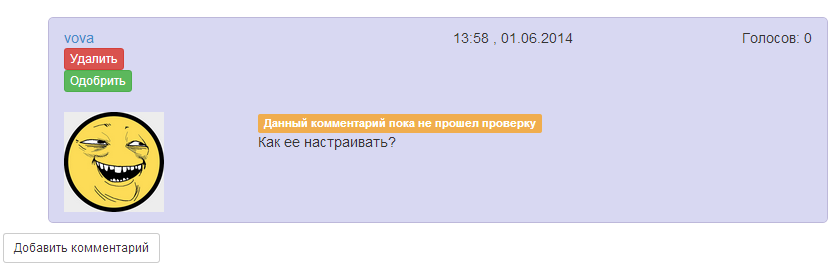
А еще голосовать за оставленные комментарии, тем самым повышая рейтинг пользователя, оставившего комментарий.



## Описание интерфейса модератора

Модератор обладает теми же возможностями, что и зарегистрированный пользователь. Кроме того он может удалять темы.

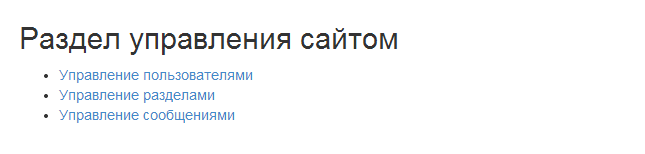


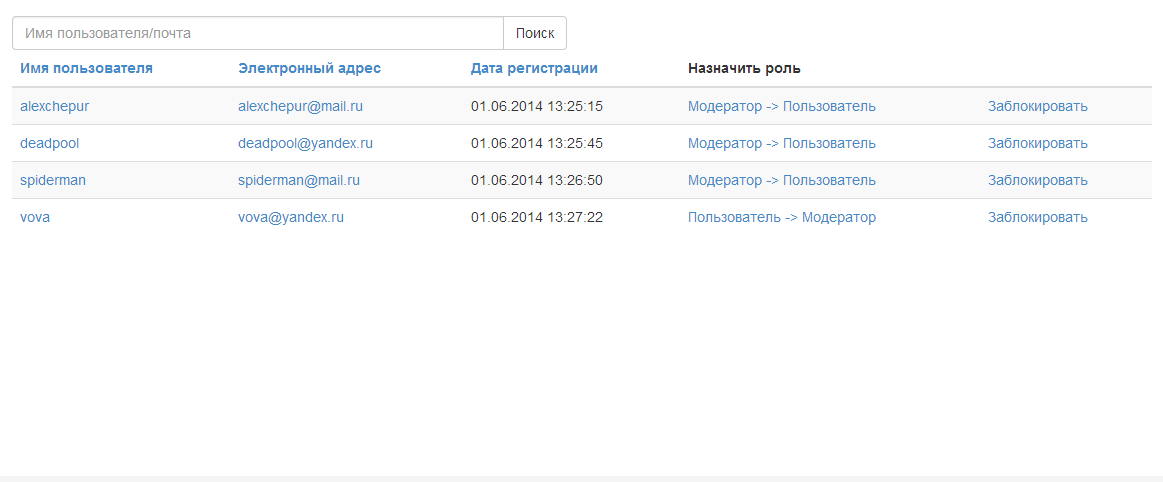
Однако основными обязанностями модератора является модерация сообщений.

Кроме того, для всех модераторов сайта существует раздел модерации сообщений, где отображаются все комментарии, непрошедшие проверку.



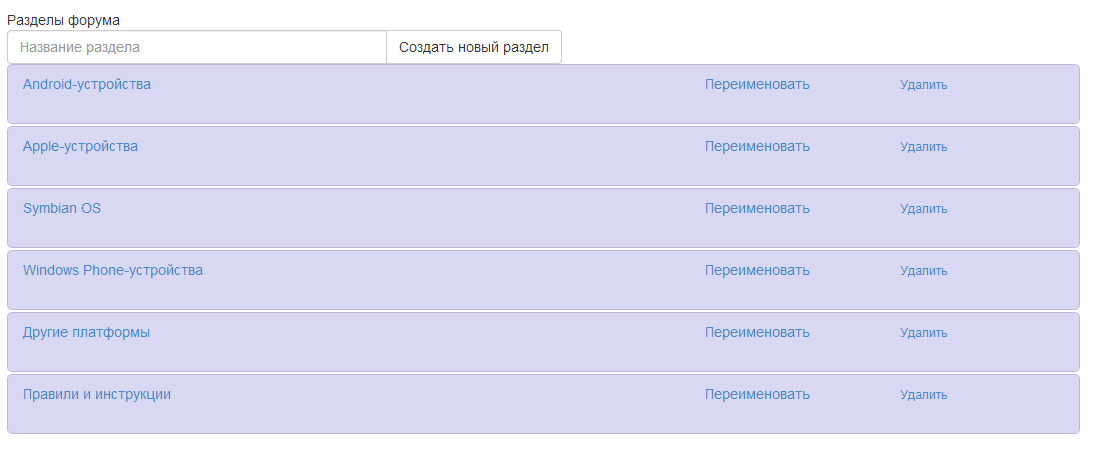
## Описание интерфейса администратора

Администратор обладает теми же возможностями, что и модератор. Кроме того, ему доступен раздел управления сайтом.

При переходе по ссылке «Управление пользователями» открывается следующая страница.

Администратор может блокировать\восстанавливать пользователей, делать обычных пользователей модераторами или наоборот.

При переходе по ссылке «Управление разделами» открывается следующая страница.



Администратор может переименовывать либо удалять существующие разделы, а так же создавать новые.

# Заключение

В рамках данной курсовой работы было разработано Web-приложение, которое позиционирует себя как форум. Разработанное приложение позволяет пользователю осуществлять быстрый поиск тем по названию.

Любой пользователь может без предварительной подготовки работать с данным приложением. Интуитивно-понятный интерфейс позволяет достаточно просто добавлять новые темы и комментарии. В свою очередь, со стороны модератора\администратора легко следить за активностью пользователей.

Приложение создано с использованием ASP.NET MVC4 фреймворка, разработчики которого уделяют большое внимание сетевой безопасности.

В приложение заложена возможность его модернизации посредством увеличения эффективности поиска тем (за счет увеличения количества полей, по которым осуществляется поиск и ввода полнотекстового поиска), ввода дополнительной функциональности и настроек.

# Библиографический список

1. Паттерны проектирования программных приложений [http://design-pattern.ru/patterns]
2. Всплывающие окна подсказки без использования картинок [http://habrahabr.ru/post/161041/]
3. Компоненты Bootstrap [http://bootstrap-ru.com/components.php]
4. Джеффри Палермо 2012 г. ASP.NET MVC4 in Action
5. Мартин Фаулер. Москва 2006 г. Архитектура программных приложений.

# Приложение 1. Ход разработки проекта

Разработка с использованием Git

Разработка проекта велась в команде из двух человек. На этапе проектирования каждому участнику разработки были назначены компоненты, за которые он будет ответственен. Учитывая то, что при разработке необходимо будет пользоваться MVC паттерном, то решено было использовать систему управления версиями файлов Git. В качестве git хостинга был выбран Github (<https://github.com/morow93/WorkForum>). С помощью сетевого графика коммитов можно проследить все этапы разработки проекта, а так же когда и кем был сделан тот или иной модуль.

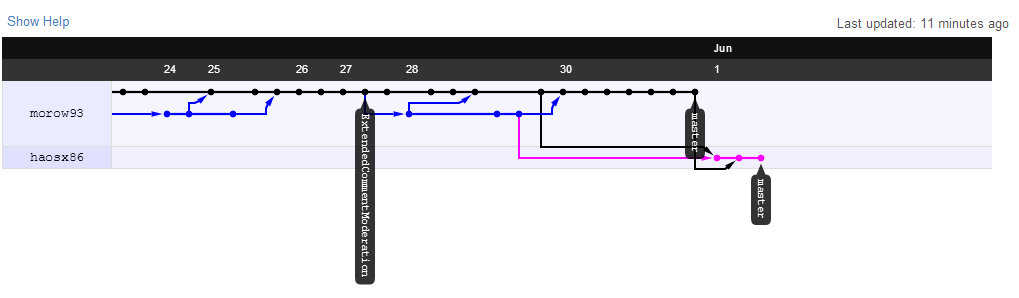


Рис. 1. Фрагмент сетевого графика коммитов

Участники проекта на Github

* Morow93 – Чепурной Александр
* Haosx86 – Терехов Владимир

Активность

GitHub позволяет просмотреть историю активности. Это можно сделать как для отдельного участника, так и для всего проекта в целом. На рис. 1 приведён график активности для проекта в целом.

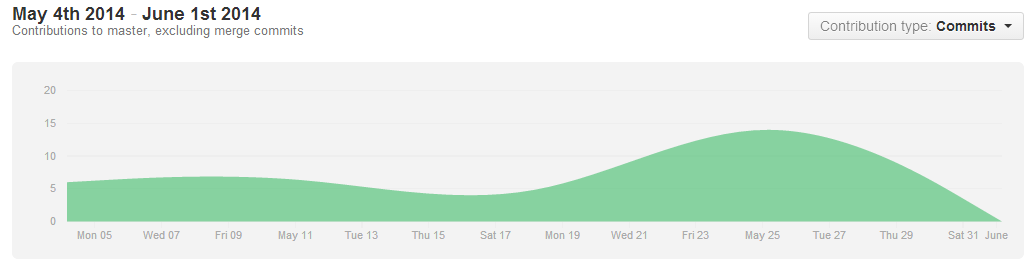


Рис. 2. График активности всех участников проекта за всё время.

Таким образом, можно убедится, что на всю разработку ушло чуть меньше месяца и основная часть работы не была выполнена в последний день, что, как правило, приводит к очень плохому качеству кода.

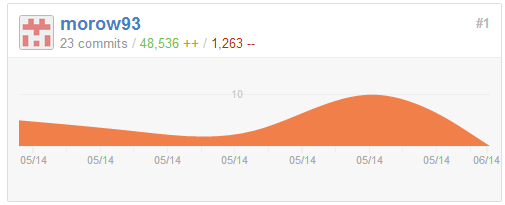


Рис. 3. Активность коммитов Чепурного Александра

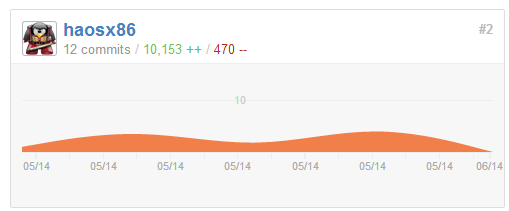


Рис. 4 Активность коммитов Терехова Владимира

Разработка основного функционала заняла только первую половину месяца. В этом можно убедиться, если взглянуть на график кол-ва добавлений для всего проекта.

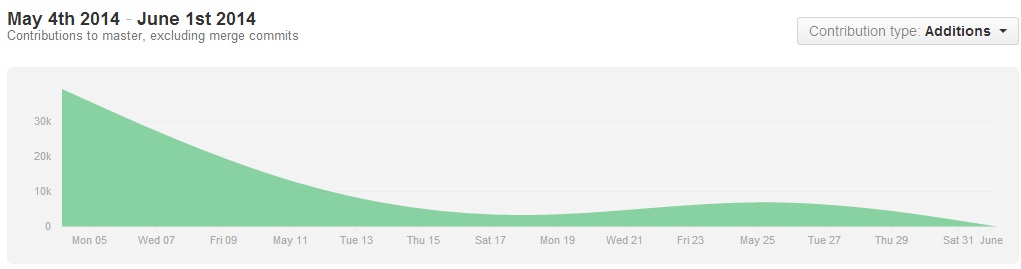


Рис. 5. График количества добавленных строк

# Приложение 2. Частичный листинг приложения

1. HomeController

namespace BeginApplication.Controllers

{

[Authorize]

[InitializeSimpleMembership]

public class HomeController : Controller

{

public IForumRepository repository;

public HomeController(IForumRepository \_repository)

{

repository = \_repository;

}

[AllowAnonymous]

public ActionResult Index()

{

int? curUserId = null;

if (Request.IsAuthenticated &&

WebSecurity.UserExists(WebSecurity.CurrentUserName)) curUserId =

WebSecurity.CurrentUserId;

return View(new ThemesModel { Themes = repository.GetRecentThemes(5,

curUserId) });

}

[AllowAnonymous]

public ActionResult Contact()

{

return View();

}

}

}

1. AccountController

namespace BeginApplication.Controllers

{

[Authorize]

[InitializeSimpleMembership]

public class AccountController : Controller

{

public IForumRepository repository;

public AccountController(IForumRepository \_repository)

{

repository = \_repository;

}

#region Личный кабинет

public ActionResult Index()

{

var model = repository.GetPrivateCabinet(WebSecurity.CurrentUserId);

return View(model);

}

public PartialViewResult GetThemesByUser(int? id, int page = 1)

{

int \_id = id ?? WebSecurity.CurrentUserId;

var userThemes = repository.GetThemesByUser(\_id);

var model = new UserThemesModel { PagedThemes =

(PagedList<ShortThemeInfo>)userThemes.ToPagedList(page, 10), TotalItems = userThemes.Count, UserId = \_id };

return PartialView(model);

}

#endregion

#region Вход

[AllowAnonymous]

public ActionResult Login()

{

if (Request.IsAuthenticated) return RedirectToAction("Index", "Home");

return View();

}

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public ActionResult Login(LoginModel model)

{

var user = new UserProfile();

using(var context = new SimpleMembershipContext())

{

user = context.UserProfiles.FirstOrDefault(u => u.UserName ==

model.UserName || u.Email == model.UserName);

}

if (user != null && ModelState.IsValid)

{

if (WebSecurity.Login(model.UserName, model.Password, persistCookie:

model.RememberMe))

{

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

}

else

{

if (WebSecurity.Login(user.UserName, model.Password, persistCookie:

model.RememberMe))

{

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

}

}

}

ModelState.AddModelError("", "Данные для входа указаны неверно.");

return View(model);

}

#endregion

#region Выход

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Logoff(string returnUrl)

{

WebSecurity.Logout();

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

}

#endregion

#region Регистрация

[AllowAnonymous]

public ActionResult Captcha()

{

return new CaptchaResult();

}

[AllowAnonymous]

public ActionResult Register()

{

if (Request.IsAuthenticated) return RedirectToAction("Index", "Home");

return View();

}

[Captcher(MessageText = "Неверный код с картинки")]

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public ActionResult Register(RegisterModel model)

{

if (ModelState.IsValid)

{

try

{

var context = new SimpleMembershipContext();

var user = context.UserProfiles.FirstOrDefault(u => u.Email.ToLower()

== model.Email.ToLower());

context.Dispose();

if (user == null)

{

WebSecurity.CreateUserAndAccount(

model.UserName,

model.Password,

new {

Email = model.Email,

Mobile = model.Mobile,

RegistrationDate = DateTime.Now

}

);

Roles.AddUserToRole(model.UserName, "active");

WebSecurity.Login(model.UserName, model.Password);

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

}

else

{

ModelState.AddModelError("", "Электронный адрес уже существует.

Введите другой электронный адрес.");

}

}

catch (MembershipCreateUserException e)

{

ModelState.AddModelError("", ErrorCodeToString(e.StatusCode));

}

}

return View(model);

}

#endregion

#region Смена пароля из личного кабинета

public ActionResult ChangePassword()

{

ViewBag.HasLocalAccount =

OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name))

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult ChangePassword(ChangePasswordModel model)

{

bool hasLocAcc = ViewBag.HasLocalAccount =

OAuthWebSecurity.HasLocalAccount(WebSecurity.GetUserId(User.Identity.Name))

if (hasLocAcc)

{

if (ModelState.IsValid)

{

bool changePasswordSucceeded;

try

{

changePasswordSucceeded =

WebSecurity.ChangePassword(User.Identity.Name, model.OldPassword, model.NewPassword);

}

catch (Exception)

{

changePasswordSucceeded = false;

}

if (changePasswordSucceeded)

{

TempData["success"] = "Пароль был успешно изменен";

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError("", "Неправильный текущий пароль или

недопустимый новый пароль.");

}

}

}

else

{

ModelState.AddModelError("", "У пользователя нет локальной учетной

записи.");

}

return View(model);

}

#endregion

#region Высылка пароля на почту

[AllowAnonymous]

public ActionResult ForgotPassword()

{

return View();

}

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public ActionResult ForgotPassword(string Email)

{

var user = new SimpleMembershipContext().UserProfiles.FirstOrDefault(u =>

u.Email.ToLower() == Email.ToLower());

if (user == null)

{

TempData["failure"] = "Не существует пользователя с такой почтой.";

}

else

{

var token = WebSecurity.GeneratePasswordResetToken(user.UserName);

var resetLink = "<a href='" + Url.Action("ResetPassword", "Account", new

{ userName = user.UserName, resetToken = token }, "http") + "'>ссылке</a>";

var subject = "Сброс пароля";

var body = "<b>Для сброса пароля перейдите по </b>" + resetLink;

bool flag = true;

try {

SendEMail(Email, subject, body);

}

catch (Exception) {

flag = false;

}

if (flag)

TempData["success"] = "Письмо для сброса пароля было отправлено на

указанную почту";

else

TempData["failure"] = "При отправке письме для сброса пароля

произошла ошибка";

}

return View();

}

[HttpGet]

[AllowAnonymous]

public ActionResult ResetPassword(string userName, string resetToken)

{

return View(new RecoveryPasswordModel { Token = resetToken, UserName =

userName });

}

[HttpPost]

[AllowAnonymous]

public ActionResult ResetPassword(RecoveryPasswordModel model)

{

if (ModelState.IsValid)

{

SimpleMembershipContext db = new SimpleMembershipContext();

//ищем айди юзера по его имени

var userid = (from i in db.UserProfiles

where i.UserName == model.UserName

select i.UserId).FirstOrDefault();

//проверка соответствия айди юзера и токена

bool any = (from j in db.webpages\_Memberships

where (j.UserId == userid)

&& (j.PasswordVerificationToken == model.Token)

select j).Any();

if (any == true)

{

//сброс пароля

bool response = WebSecurity.ResetPassword(model.Token,

model.NewPassword);

if (response == true)

TempData["success"] = "Пароль успешно изменен. Теперь можете

войти.";

else

{

TempData["failure"] = "Увы, пароль не был изменен.";

}

}

else

{

TempData["failure"] = "Выявлено несоотвествие пользователя и токена

сброса пароля.";

}

}

else

{

ModelState.AddModelError("", "Невозможно выполнить сброс пароля.");

}

return View();

}

private void SendEMail(string email, string subject, string body)

{

SmtpClient client = new SmtpClient();

client.DeliveryMethod = SmtpDeliveryMethod.Network;

client.EnableSsl = true;

client.Host = "smtp.gmail.com";

client.Port = 587;

System.Net.NetworkCredential credentials = new

System.Net.NetworkCredential("derchepur@gmail.com", "\*\*\*\*\*\*\*");

client.UseDefaultCredentials = false;

client.Credentials = credentials;

MailMessage msg = new MailMessage();

msg.From = new MailAddress("derchepur@gmail.com");

msg.To.Add(new MailAddress(email));

msg.Subject = subject;

msg.IsBodyHtml = true;

msg.Body = body;

client.Send(msg);

}

#endregion

#region Настройки приватности

public ActionResult EditPrivateProperties()

{

var model = repository.GetPrivateProperties(WebSecurity.CurrentUserId);

return View(model);

}

[HttpPost]

public ActionResult EditPrivateProperties(PrivatePropertiesModel model)

{

repository.SetPrivateProperties(model, WebSecurity.CurrentUserId);

return RedirectToAction("Index");

}

#endregion

#region Аватар

[AllowAnonymous]

public FileContentResult GetImage(int userId)

{

var user = repository.UserProfiles.FirstOrDefault(u => u.UserId == userId);

if (user != null && user.ImageData != null)

{

return File(user.ImageData, user.ImageMimeType);

}

else

{

return null;

}

}

[HttpPost]

public ActionResult ChangeAvatar(PrivateCabinetModel model,

HttpPostedFileBase file)

{

if (ModelState.IsValid)

{

if (file != null)

{

using (var context = new SimpleMembershipContext())

{

var user = context.UserProfiles.FirstOrDefault(x => x.UserId ==

WebSecurity.CurrentUserId);

user.ImageData = new byte[file.ContentLength];

file.InputStream.Read(user.ImageData, 0, file.ContentLength);

user.ImageMimeType = file.ContentType;

context.SaveChanges();

}

}

}

return RedirectToAction("Index");

}

#endregion

#region Вспомогательные методы

private ActionResult RedirectToLocal(string returnUrl)

{

if (Url.IsLocalUrl(returnUrl))

{

return Redirect(returnUrl);

}

else

{

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

}

}

private static string ErrorCodeToString(MembershipCreateStatus createStatus)

{

switch (createStatus)

{

case MembershipCreateStatus.DuplicateUserName:

return "Имя пользователя уже существует.";

case MembershipCreateStatus.DuplicateEmail:

return "Имя пользователя для электронной почты уже существует.";

case MembershipCreateStatus.InvalidPassword:

return "Указан недопустимый пароль. Введите допустимое значение.";

case MembershipCreateStatus.InvalidEmail:

return "Указан недопустимый адрес электронной почты.";

case MembershipCreateStatus.InvalidAnswer:

return "Указан недопустимый ответ для восстановления пароля.";

case MembershipCreateStatus.InvalidQuestion:

return "Указан недопустимый вопрос для восстановления пароля.";

case MembershipCreateStatus.InvalidUserName:

return "Указано недопустимое имя пользователя.";

case MembershipCreateStatus.ProviderError:

return "Поставщик проверки подлинности вернул ошибку.";

case MembershipCreateStatus.UserRejected:

return "Запрос создания пользователя был отменен.";

default:

return "Произошла неизвестная ошибка.”;

}

}

#endregion

}

}

1. ForumController

namespace BeginApplication.Controllers

{

[Authorize]

[InitializeSimpleMembership]

public class ForumController : Controller

{

public IForumRepository repository;

public ForumController(IForumRepository \_repository)

{

repository = \_repository;

}

[AllowAnonymous]

public ActionResult Index()

{

int? curUserId = null;

if (Request.IsAuthenticated &&

WebSecurity.UserExists(WebSecurity.CurrentUserName)) curUserId = WebSecurity.CurrentUserId;

return View(new SectionsModel { Sections =

repository.GetForumSections(curUserId) });

}

/// <summary>

/// Получить темы выбранной секции

/// </summary>

[AllowAnonymous]

public ActionResult Section(string searching, int id, int page = 1)

{

int? curUserId = null;

if (Request.IsAuthenticated &&

WebSecurity.UserExists(WebSecurity.CurrentUserName)) curUserId = WebSecurity.CurrentUserId;

var themesBySection = repository.GetThemesBySection(id, curUserId);

if (!String.IsNullOrEmpty(searching))

{

themesBySection = themesBySection.Where(s =>

s.ThemeTitle.ToUpper().Contains(searching.ToUpper())).ToList();

}

var model = new ThemesWithSectionModel

{

TotalItems = themesBySection.Count,

PagedThemes = (PagedList<ThemeInfo>)themesBySection.ToPagedList(page, 10),

Section = repository.Sections.FirstOrDefault(s => s.SectionId == id),

SearchingString = searching

};

int countPages = (int)Math.Ceiling((double)model.TotalItems/(double)10);

if (countPages != 0 && page > countPages)

{

while (page > countPages) page--;

return RedirectToAction("Section", "Forum", new { searching = searching,

id = id, page = page });

}

return View(model);

}

/// <summary>

/// Получить сообщения выбранной темы

/// </summary>

[AllowAnonymous]

public ActionResult Theme(int id, int page = 1)

{

if (page == 0) ++page;

var showAll = User.IsInRole("moder") || User.IsInRole("admin");

int? curUserId = null;

if (Request.IsAuthenticated &&

WebSecurity.UserExists(WebSecurity.CurrentUserName)) curUserId =

WebSecurity.CurrentUserId;

var commentsByTheme = repository.GetCommentsByTheme(id, showAll, curUserId);

var model = new CommentsWithThemeModel

{

TotalItems = commentsByTheme.Count,

PagedComments =

(PagedList<CommentInfo>)commentsByTheme.ToPagedList(page, 10),

Theme = repository.Themes.Where(t => t.ThemeId == id).Select(x => new

CurrentTheme

{

UserId = x.User.UserId,

UserName = x.User.UserName ?? "Удаленный пользователь",

ThemeId = x.ThemeId,

ThemeTitle = x.ThemeTitle,

ThemeText = x.ThemeText,

CreationDate = x.CreationDate,

isAvatarExists = x.User.ImageData == null ? false : true

}).FirstOrDefault()

};

int countPages = (int)Math.Ceiling((double)model.TotalItems/(double)10);

if (countPages != 0 && page > countPages)

{

while (page > countPages) page--;

return RedirectToAction("Theme", "Forum", new { id = id, page = page });

}

return View(model);

}

/// <summary>

/// Получить рейтинг и дополнительную информацию о сообщении

/// </summary>

/// <param name="model"></param>

/// <returns></returns>

[HttpPost]

public PartialViewResult \_GetCommentData(CommentInfo model)

{

if (model.UserId != WebSecurity.CurrentUserId)

{

using (var context = new SimpleMembershipContext())

{

var record = context.Likes.FirstOrDefault(l => l.UserId ==

WebSecurity.CurrentUserId && l.CommentId == model.CommentId);

if (record == null)

{

context.Likes.Add(new Like { CommentId = model.CommentId, UserId

= WebSecurity.CurrentUserId, Vote = 1 });

model.CommentVote++;

}

else

{

if (record.Vote == 0)

{

record.Vote++;

model.CommentVote++;

}

else

{

model.CommentVote--;

record.Vote--;

}

}

context.SaveChanges();

}

}

return PartialView("\_GetCommentData", model);

}

#region Добавление темы

public ActionResult AddTheme(int id)

{

if (!Roles.IsUserInRole("admin") && !Roles.IsUserInRole("moder"))

{

var isExist = repository.Themes.Where(t => t.UserId ==

WebSecurity.CurrentUserId).

OrderByDescending(t => t.CreationDate).

FirstOrDefault();

if (isExist != null)

{

if ((DateTime.Now - isExist.CreationDate).TotalMinutes < 60)

{

TempData["failure"] = "Новую тему можно создать один раз в час.";

return RedirectToAction("Section", new { id = id, page = 1 });

}

}

}

var model = new AddThemeModel {

UserId = (int)WebSecurity.CurrentUserId,

SectionId = id

};

return View(model);

}

[HttpPost]

public ActionResult AddTheme(AddThemeModel theme)

{

if (ModelState.IsValid)

{

var themeToAdd = new Theme

{

SectionId = theme.SectionId,

UserId = theme.UserId,

ThemeTitle = theme.ThemeTitle,

ThemeText = theme.ThemeText,

CreationDate = DateTime.Now

};

repository.AddTheme(themeToAdd);

TempData["success"] = "Новая тема успешно создана.";

}

else

{

return View(theme);

}

return RedirectToAction("Section", "Forum", new { id = theme.SectionId });

}

#endregion

#region Удаление темы

[Authorize(Roles="admin,moder")]

public ActionResult RemoveTheme(int id, int section, string path)

{

var result = repository.RemoveTheme(id);

if (result)

TempData["success"] = "Тема была удалена.";

else

TempData["failure"] = "Тема не была удалена.";

if (path.RemoteFileExists())

return Redirect(path);

else

return RedirectToAction("Section", "Forum", new { id = section, page =

Math.Ceiling((double)repository.Themes.Where(t=>t.SectionId ==

section).Count()/10)});

}

#endregion

#region Добавление комментария

public PartialViewResult \_AddComment(int th)

{

var model = new AddCommentModel

{

UserId = (int)WebSecurity.CurrentUserId,

ThemeId = th

};

return PartialView(model);

}

[HttpPost]

public ActionResult \_AddComment(AddCommentModel comment)

{

if (ModelState.IsValid)

{

var ToAdd = new Comment

{

ThemeId = comment.ThemeId,

UserId = comment.UserId,

CommentText = comment.CommentText,

CreationDate = DateTime.Now

};

if (User.IsInRole("moder") || User.IsInRole("admin"))

{

ToAdd.IsAdmitted = true;

repository.AddComment(ToAdd);

TempData["success"] = "Сообщение создано.";

}

else

{

repository.AddComment(ToAdd);

TempData["success"] = "Сообщение добавлено, но другие пользователи

увидят его только после проверки.";

}

}

else

{

TempData["failure"] ="Сообщение не было создано.";

}

return RedirectToAction("Theme", "Forum", new { id = comment.ThemeId, page =

Math.Ceiling((double)repository.Comments.Where(c=>c.ThemeId ==

comment.ThemeId).Count()/10) });

}

#endregion

#region Удаление комментария

[Authorize(Roles="admin,moder")]

public ActionResult RemoveComment(int id, int theme, string path)

{

var result = repository.RemoveComment(id);

if (result)

TempData["success"] = "Комментарий был удален.";

else

TempData["failure"] = "Комментарий не был удален.";

if (path.RemoteFileExists())

return Redirect(path);

else

return RedirectToAction("Theme", "Forum", new { id = theme, page =

Math.Ceiling((double)repository.Comments.Where(t => t.ThemeId ==

theme).Count() / 10) });

}

#endregion

#region Допуск комментария

[Authorize(Roles = "admin,moder")]

public ActionResult AdmittComment(int id, int theme, string path)

{

var result = repository.AdmittComment(id);

if (result)

TempData["success"] = "Комментарий был допущен.";

else

TempData["failure"] = "Комментарий не был допущен.";

if (path.RemoteFileExists())

return Redirect(path);

else

return RedirectToAction("Theme", "Forum", new { id = theme, page =

Math.Ceiling((double)repository.Comments.Where(t => t.ThemeId ==

theme).Count() / 10) });

}

#endregion

/// <summary>

/// Получить информацию о конкретном пользователе

/// </summary>

public ActionResult UserInfo(int id)

{

var model = repository.GetUserSummary(id);

return View(model);

}

}

}

1. AdminController

namespace BeginApplication.Controllers

{

[Authorize(Roles = "admin,moder")]

[InitializeSimpleMembership]

public class AdminController : Controller

{

public IForumRepository repository;

public AdminController(IForumRepository \_repository)

{

repository = \_repository;

}

[Authorize(Roles = "admin,moder")]

public ActionResult Index()

{

return View();

}

#region Получить данные для изменения

[Authorize(Roles = "admin")]

public ActionResult GetUsers(string sortOrder, string currentFilter, string

searchString, int page = 1)

{

ViewBag.CurrentSort = sortOrder;

ViewBag.EmailSort = (sortOrder == "EmailAsc") ? "EmailDesc" : "EmailAsc";

ViewBag.UserNameSort = (sortOrder == "UserNameAsc") ? "UserNameDesc" :

“UserNameAsc";

ViewBag.RegistrationSort = (sortOrder == "RegistrationAsc") ?

“RegistrationDesc" : "RegistrationAsc";

if (searchString == null)

searchString = currentFilter;

ViewBag.CurrentFilter = searchString;

var admins = Roles.GetUsersInRole("admin");

var users = repository.UserProfiles.Where(u => !admins.Contains(u.UserName));

if (!String.IsNullOrEmpty(searchString))

{

users = users.Where(s =>

s.UserName.ToUpper().Contains(searchString.ToUpper())

||

s.Email.ToUpper().Contains(searchString.ToUpper())

);

}

switch (sortOrder)

{

case "EmailDesc":

users = users.OrderBy(u => u.Email);

break;

case "EmailAsc":

users = users.OrderByDescending(u => u.Email);

break;

case "UserNameDesc":

users = users.OrderBy(u => u.UserName);

break;

case "UserNameAsc":

users = users.OrderByDescending(u => u.UserName);

break;

case "RegistrationDesc":

users = users.OrderBy(u => u.RegistrationDate);

break;

case "RegistrationAsc":

users = users.OrderByDescending(u => u.RegistrationDate);

break;

default:

users = users.OrderBy(s => s.UserName);

break;

}

var model = new ManagementUsersModel

{

TotalItems = users.Count(),

PagedUsers = (PagedList<UserModel>)users.Select(x => new UserModel

{

UserId = x.UserId,

UserName = x.UserName,

Email = x.Email,

RegistrationDate = x.RegistrationDate

}).ToPagedList(page, 10)

};

int countPages = (int)Math.Ceiling((double)model.TotalItems/(double)10);

if (countPages != 0 && page > countPages)

{

while (page > countPages) page--;

return RedirectToAction("GetUsers", "Admin", new {

sortOrder = sortOrder,

currentFilter = currentFilter,

searchString = searchString,

page = page

});

}

return View(model);

}

[Authorize(Roles = "admin")]

public ActionResult GetSections()

{

return View(repository.Sections.Select(x => new ChangeSectionModel {

SectionId = x.SectionId, SectionTitle = x.SectionTitle }).OrderBy(s =>

s.SectionTitle).ToList());

}

[Authorize(Roles = "admin,moder")]

public ActionResult GetNotAdmittedComments(int page = 1)

{

var notAdmittedComments = repository.GetNotAdmittedComments();

var model = new CommentsAdmittedModel { PagedComments =

(PagedList<CommentAdmittedInfo>)notAdmittedComments.ToPagedList(page, 10),

TotalItems = notAdmittedComments.Count };

int countPages = (int)Math.Ceiling((double)model.TotalItems / (double)10);

if (countPages != 0 && page > countPages)

{

while (page > countPages) page--;

return RedirectToAction("GetNotAdmittedComments", "Admin", new { page =

page });

}

return View(model);

}

#endregion

#region Модерация сообщений

[Authorize(Roles = "admin,moder")]

public ActionResult RemoveComment(int id, string path)

{

var result = repository.RemoveComment(id);

if (result)

TempData["success"] = "Комментарий был удален.";

else

TempData["failure"] = "Комментарий не был удален.";

if (path.RemoteFileExists())

return Redirect(path);

else

return RedirectToAction("GetNotAdmittedComments", "Admin", new { page =

Math.Ceiling((double)repository.Comments.Where(c => !c.IsAdmitted).Count() / 10) });

}

[Authorize(Roles = "admin,moder")]

public ActionResult AdmittComment(int id, string path)

{

var result = repository.AdmittComment(id);

if (result)

TempData["success"] = "Комментарий был допущен.";

else

TempData["failure"] = "Комментарий не был допущен.";

if (path.RemoteFileExists())

return Redirect(path);

else

return RedirectToAction("GetNotAdmittedComments", "Admin", new { page =

Math.Ceiling((double)repository.Comments.Where(c =>

!c.IsAdmitted).Count() / 10) });

}

#endregion

#region Переименование раздела

[Authorize(Roles = "admin")]

public ActionResult \_RenameSection(ChangeSectionModel section)

{

return PartialView(section);

}

[HttpPost]

[Authorize(Roles = "admin")]

public ActionResult \_SubmitSectionChange(ChangeSectionModel section)

{

using (var context = new SimpleMembershipContext())

{

var db\_section = context.Sections.FirstOrDefault(x => x.SectionId ==

section.SectionId);

if (ModelState.IsValid)

{

try

{

db\_section.SectionTitle = section.SectionTitle;

context.SaveChanges();

}

catch

{

Response.StatusCode = (int)HttpStatusCode.BadRequest;

}

}

else

{

section.SectionTitle = db\_section.SectionTitle;

Response.StatusCode = (int)HttpStatusCode.BadRequest;

}

}

return PartialView("\_Section", section);

}

[Authorize(Roles = "admin")]

public ActionResult \_HideRenameForm(ChangeSectionModel section)

{

return PartialView("\_Section", section);

}

#endregion

#region Изменить роль пользователя

[Authorize(Roles = "admin")]

public string GetChangeRoleUrl(int id, string path)

{

var linkText = String.Empty;

if (Roles.IsUserInRole(repository.UserProfiles.FirstOrDefault(u => u.UserId

== id).UserName, "moder"))

{

linkText = "<a class='moder-to-user' title='Разжаловать' href='"

+ Url.Action("ChangeRole", "Admin", new { id = id, role =

String.Empty, path = path }, "http")

+ "'>Модератор -> Пользователь</a>";

}

else

{

linkText = "<a class='user-to-moder' title='Повысить' href='"

+ Url.Action("ChangeRole", "Admin", new { id = id, role = "moder",

path = path }, "http")

+ "'>Пользователь -> Модератор</a>";

}

return linkText;

}

[Authorize(Roles = "admin")]

public ActionResult ChangeRole(int id, string role, string path)

{

if (string.IsNullOrEmpty(role))

Roles.RemoveUserFromRole(repository.UserProfiles.FirstOrDefault(u =>

u.UserId == id).UserName, "moder");

else

Roles.AddUserToRole(repository.UserProfiles.FirstOrDefault(u => u.UserId

== id).UserName, "moder");

if (path.RemoteFileExists())

return Redirect(path);

else

return RedirectToAction("GetUsers", "Admin");

}

#endregion

/// <summary>

/// Удаление юзера

/// </summary>

[Authorize(Roles = "admin")]

public ActionResult RemoveUser(UserModel user)

{

if (ModelState.IsValid)

{

try

{

Roles.RemoveUserFromRole(user.UserName, "active");

}

catch

{

Response.StatusCode = (int)HttpStatusCode.BadRequest;

}

}

else

{

Response.StatusCode = (int)HttpStatusCode.BadRequest;

}

return PartialView("\_UserManage", user);

}

/// <summary>

/// Восстановление юзера

/// </summary>

[Authorize(Roles = "admin")]

public ActionResult RecoveryUser(UserModel user)

{

if (ModelState.IsValid)

{

try

{

Roles.AddUserToRole(user.UserName, "active");

}

catch

{

Response.StatusCode = (int)HttpStatusCode.BadRequest;

}

}

else

{

Response.StatusCode = (int)HttpStatusCode.BadRequest;

}

return PartialView("\_UserManage", user);

}

/// <summary>

/// Удаление раздела

/// </summary>

[Authorize(Roles = "admin")]

public ActionResult RemoveSection(ChangeSectionModel section)

{

if (ModelState.IsValid)

{

var result = repository.RemoveSection(section.SectionId);

if (result)

{

section = null;

}

else

{

Response.StatusCode = (int)HttpStatusCode.BadRequest;

}

}

else

{

Response.StatusCode = (int)HttpStatusCode.BadRequest;

}

return PartialView("\_Section", section);

}

/// <summary>

/// Создание нового раздела

/// </summary>

[HttpPost]

[Authorize(Roles = "admin")]

public ActionResult CreateSection(ChangeSectionModel model)

{

if (ModelState.IsValid)

{

var result = repository.AddSection(new Section { SectionTitle =

model.SectionTitle });

if (!result) {

TempData["failure"] = "Раздел не был добавлен.";

}

else

{

TempData["success"] = "Раздел был добавлен.";

}

}

else

{

TempData["failure"] = "Раздел не был добавлен.";

}

ModelState.Clear();

return RedirectToAction("GetSections");

}

}

}