

Министерство образования Республики Беларусь  
Учреждение образования «Белорусский государственный университет  
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

Веб-сайт доска объявлений

БГУИР КП 1-40 01 0 26 ПЗ

Студент: гр. 351002 Гучок О.А.

Руководитель: асс. Третьяков Ф.И.

Минск 2015

Учреждение образования  
«Белорусский государственный университет информатики и  
радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ  
Заведующий кафедрой ПОИТ

\_\_\_\_\_  
(подпись)

\_\_\_\_\_  
2015 г.

ЗАДАНИЕ  
по курсовому проектированию

Студенту Гучку Олегу Анатольевичу

1. Тема работы Веб-сайт доска объявлений
2. Срок сдачи студентом законченной работы DD.MM.YYYY
3. Исходные данные к работе Среда разработки Visual Studio 2013.
4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)  
Введение. 1. Анализ литературных источников. 2. Постановка задачи  
3.Разработка программного средства. 4. Руководство по  
использованию веб-сайта. Заключение. Приложения.
5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)  
1. Схема алгоритма
6. Консультант по курсовой работе  
Третьяков Ф.И.
7. Дата выдачи задания DD.MM.YYYY
8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):  
раздел 1 к DD.MM.YYYY – 15 % готовности работы;  
разделы 2, 3 к DD.MM.YYYY – 30 % готовности работы;  
раздел 4 к DD.MM.YYYY – 60 % готовности работы;  
раздел 5, 6 к DD.MM.YYYY – 90 % готовности работы;

оформление пояснительной записки и графического материала к  
DD.ММ.YYYY – 100 % готовности работы.  
Защита курсового проекта с DD по DD декабря YYYY г.

РУКОВОДИТЕЛЬ \_\_\_\_\_ Третьяков Ф.И.  
(подпись)

Задание принял к исполнению \_\_\_\_\_ Гучок О.А. DD.ММ.YYYYг.  
(дата и подпись студента)

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ . . . . .	5
1 АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ . . . . .	7
1.1 Платформа .Net Framework . . . . .	7
1.2 Веб-технология ASP.NET . . . . .	8
1.3 Реляционные базы данных . . . . .	9
1.4 MVC Framework . . . . .	9
1.5 Сервер IIS . . . . .	11
1.6 Dependency Injection . . . . .	11
2 СРАВНЕНИЕ АНАЛОГОВ И ПОСТАНОВКА ЗАДАЧИ . . . . .	12
2.1 Сравнение приложения с существующими аналогами . . . . .	12
2.2 Постановка задачи . . . . .	14
3 РАЗРАБОТКА ПРИЛОЖЕНИЯ . . . . .	16
3.1 Описание базы данных . . . . .	16
3.2 Используемые классы . . . . .	16
4 ТЕСТИРОВАНИЕ . . . . .	18
5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ . . . . .	19
ЗАКЛЮЧЕНИЕ . . . . .	25
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ . . . . .	26
ПРИЛОЖЕНИЕ А . . . . .	27
ПРИЛОЖЕНИЕ Б . . . . .	65

## ВВЕДЕНИЕ

Технологический прогресс и всемирная сеть способствовали значительному увеличению объема информации, углублению и расширению наших представлений об окружающем мире. До сих пор основным способом предъявления полученных новых научных знаний была публикация научных статей и разработок ученых. Гипермедиа ресурсы позволили осуществить быстрый доступ ко всей научной информации, размещенной в Интернете. Научные дискуссии, обмен мнениями и критическими замечаниями могут быть реализованы в режиме реального времени, что, несомненно, способствует более быстрому продвижению научных открытий в реальную жизнь. Сфера интернет-технологий является одной из наиболее динамично развивающихся направлений обработки информации. Технологии не успевают погнаться за своим полным распространением, как их сменяют другие, более совершенные.

Бесчисленное множество новых технологий делает нашу жизнь невозможной без быстрого доступа к информации. В наше время очень легко получить информацию, одним из способов быстрого доступа к ней является сайт.

Создание сайтов на сегодняшний день, становится одной из наиболее актуальных и востребованных услуг. Именно поэтому, большинство компаний уже оценили все преимущества такого предложения как создание сайтов и позаботились о разработке подходящего ресурса.

Пользователю приятно посещать те Web-страницы, которые имеют стильное оформление, не отягощены чрезмерно графикой и анимацией, быстро загружаются и правильно отображаются в окне Web-браузера. Но может возникнуть и другая проблема - сайт может оказаться не интересным пользователю и та информация, которую он несет, окажется не востребованной. Именно поэтому важно, чтобы сайт отвечал всем требованиям пользователя.

Актуальность данной работы заключается в том, что с учетом скорости развития сети Интернет и направления e-commerce, рынок WEB-разработки огромен и очень перспективен.

Целью работы является формирование теоретических знаний по проектированию web-сайта и практических навыков по его разработке. Выбранная технология - ASP.NET MVC.

Инфраструктура ASP.NET MVC Framework реализует шаблон MVC и при этом обеспечивает существенно улучшенное разделение ответственности. На самом деле в ASP.NET MVC внедрен современный вариант MVC, который особенно хорошо подходит для веб-приложений.

Для хранения информации пользователя будет использоваться база данных, доступ к которой будет контролироваться самим приложением.

# 1 АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ

## 1.1 Платформа .Net Framework

.NET Framework - программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общезыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Архитектура .NET. Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код Common Intermediate Language (CIL) (ранее назывался Microsoft Intermediate Language, MSIL). В терминах .NET получается сборка, англ. assembly. Затем код либо исполняется виртуальной машиной Common Language Runtime (CLR), либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора. Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части. В случае использования виртуальной машины CLR, встроенный в неё JIT-компилятор «на лету» (just in time) преобразует промежуточный байт-код в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина CLR также сама заботится о базовой безопасности, управлении памятью и системе исключений, избавляя разработчика от части работы. Архитектура .NET Framework описана и опубликована в спецификации Common Language Infrastructure (CLI), разработанной Microsoft и утверждённой ISO и ECMA. В CLI описаны типы данных .NET, формат метаданных о структуре программы, система исполнения байт-кода и многое другое.

Объектные классы .NET, доступные для всех поддерживаемых языков программирования, содержатся в библиотеке Framework Class Library (FCL). В FCL входят классы Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation и другие. Ядро FCL называется Base Class Library (BCL). Для выполнения данной курсовой работы была использована среда разработки Microsoft Visual Studio 2012 Express. Немного о самой среде разработки.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и

подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода.

## 1.2 Веб-технология ASP.NET

ASP.NET - технология создания веб-приложений и веб-сервисов от компании Майкрософт. Она является составной частью платформы Microsoft .NET и развитием более старой технологии Microsoft ASP. На данный момент последней версией этой технологии является ASP.NET 4.5.

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.

Преимущества ASP.NET:

- Компилируемый код выполняется быстрее, большинство ошибок отлавливается ещё на стадии разработки;
- Значительно улучшенная обработка ошибок во время выполнения запущенной готовой программы, с использованием блоков try..catch;
- Пользовательские элементы управления (controls) позволяют выделять часто используемые шаблоны, такие как меню сайта;
- Использование метафор, уже применяющихся в Windows-приложениях, например, таких как элементы управления и события;
- Расширяемый набор элементов управления и библиотек классов позволяет быстрее разрабатывать приложения;
- ASP.NET опирается на многоязыковые возможности .NET, что позволяет писать код страниц на VB.NET, Delphi.NET, Visual C#, J# и т. д.;
- Возможность кэширования всей страницы или её части для увеличения производительности;
- Возможность кэширования данных, используемых на странице;
- Возможность разделения визуальной части и бизнес-логики по разным файлам («code behind»);
- Расширяемая модель обработки запросов;
- Расширенная событийная модель;
- Расширяемая модель серверных элементов управления;
- Наличие master-страниц для задания шаблонов оформления страниц;
- Поддержка CRUD-операций при работе с таблицами через GridView;



- Встроенная поддержка AJAX.

### 1.3 Реляционные базы данных

База данных — это набор таблиц, состоящих из столбцов и строк, аналогично электронной таблице. Каждая строка содержит одну запись; каждый столбец содержит все экземпляры конкретного фрагмента данных всех строк.

Реляционные базы данных позволяют хранить информацию в нескольких «плоских» (двухмерных) таблицах, связанных между собой посредством совместно используемых полей данных, называемых ключами. Реляционные базы данных предоставляют более простой доступ к оперативно составляемым отчетам (обычно через SQL) и обеспечивают повышенную надежность и целостность данных благодаря отсутствию избыточной информации.

В реляционной базе данных каждая таблица должна иметь первичный ключ - поле или комбинацию полей, которые единственным образом идентифицируют каждую строку таблицы. Если ключ состоит из нескольких полей, он называется составным. Ключ должен быть уникальным и однозначно определять запись. По значению ключа можно отыскать единственную запись. Ключи служат также для упорядочивания информации в БД.

Для взаимодействия объектов программы с реляционной базой данных без использования SQL можно использовать инфраструктуру ORM (object-relational mapping), которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Одной из таких инфраструктур является Entity Framework. Entity Framework (EF) — это программная модель, которая пытается заполнить пробел между конструкциями базы данных и объектно-ориентированными конструкциями. При помощи Entity Framework таблицы, столбцы и строки реляционной базы данных представляются с помощью обычных объектов C#. Для работы с Entity Framework применяется интегрированный язык запросов LINQ, который существенно упрощает работу с базами данных.

### 1.4 MVC Framework

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 так, что модификация одного из компонентов оказывает минимальное воздействие на остальные. Данная схема проектирования (на рисунке 1.1) часто используется для построения архитектурного каркаса, когда переходят от

теории к реализации в конкретной предметной области.



Рис. 1: Паттерн MVC

Основная цель применения этой концепции состоит в разделении бизнес-логики (модели) от её визуализации (представления, вида). За счет такого разделения повышается возможность повторного использования. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

- К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;
- Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопку, ввод данных), для этого достаточно использовать другой контроллер.

Концепция MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

- **Модель** (англ. Model). Модель предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать;
- **Представление, вид** (англ. View). Отвечает за отображение информации (визуализацию). Часто в качестве представления выступает форма (окно) с графическими элементами;
- **Контроллер** (англ. Controller). Обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и

использует модель и представление для реализации необходимой реакции.

Важно отметить, что как представление, так и контроллер зависят от модели. Однако модель не зависит ни от представления, ни от контроллера.

### **1.5 Сервер IIS**

IIS Web Server встроен в Windows. IIS Express работает с VS 2012 и Visual Web Developer 2010 Express, запускаться на Windows XP и выше, не требует прав администратора и внесения изменений в код приложения. Позволяет работать со всеми типами ASP.NET приложений и разрабатывать, используя всю мощь возможностей IIS 7.x.

Используя IIS, как сервер для разработок получаем все возможности веб-сервера (SSL, URL Rewrite Rules и т.п.). IIS является полноценным веб-сервером, а это значит, что будет точно известно, как будет работать приложения на публичном сервере.

### **1.6 Dependency Injection**

Процесс предоставления внешней зависимости программному компоненту. Является специфичной формой «инверсии управления» (англ. Inversion of control, IoC), где изменение порядка связи осуществляется путём получения необходимой зависимости.

Работа фреймворка, обеспечивающая внедрение зависимости, описывается следующим образом. Приложение, независимо от оформления, выполняется внутри контейнера IoC, предоставляемого фреймворком. Часть объектов в программе по-прежнему создается обычным способом языка программирования, часть создается контейнером на основе предоставленной ему конфигурации.

Условно, если объекту нужно получить доступ к определенному сервису, объект берет на себя ответственность за доступ к этому сервису: он или получает прямую ссылку на местонахождение сервиса, или обращается к известному «сервис-локатору» и запрашивает ссылку на реализацию определенного типа сервиса. Используя же внедрение зависимости, объект просто предоставляет свойство, которое в состоянии хранить ссылку на нужный тип сервиса; и когда объект создается, ссылка на реализацию нужного типа сервиса автоматически вставляется в это свойство (поле), используя средства среды.

Внедрение зависимости более гибко, потому что становится легче создавать альтернативные реализации данного типа сервиса, а потом указывать, какая именно реализация должна быть использована в, например, конфигурационном файле, без изменений в объектах, которые этот сервис используют. Это особенно полезно в юнит-тестировании, потому что вставить реализацию «заглушки» сервиса в тестируемый объект очень просто.

## 2 СРАВНЕНИЕ АНАЛОГОВ И ПОСТАНОВКА ЗАДАЧИ

### 2.1 Сравнение приложения с существующими аналогами

В интернете множество сайтов объявлений, например: kufar.by, ixr.by, doska.by, onliner.by. Данные сайты предусматривают просмотр и поиск объявлений, а также добавление, удаление, изменение объявления. Любой пользователь может зарегистрироваться на сайте и добавить свое объявление (Рис. 2).

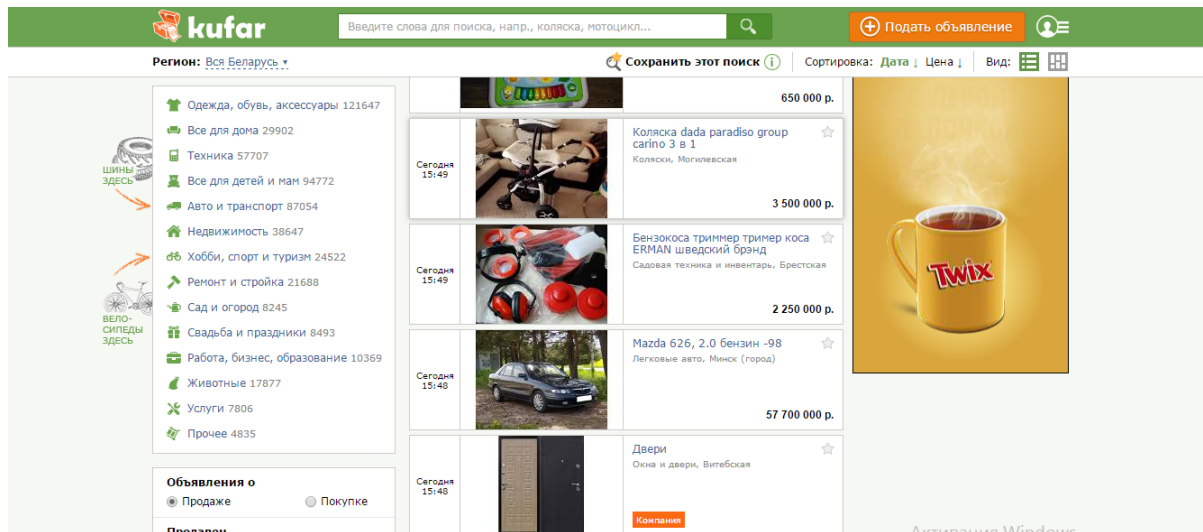


Рис. 2: Главная страница kufar.by

На некоторых сайтах предусмотрена функция отклика на объявления, оставленные другими пользователями (Рис. 3).

Рис. 3: Форма отклика на объявление

Также у каждого объявления должна быть контактная информация пользователя, разместившего его (Рис. 4).

На таких сайтах как auto.by необходимо подтверждение администратором объявления. Таким образом администратор

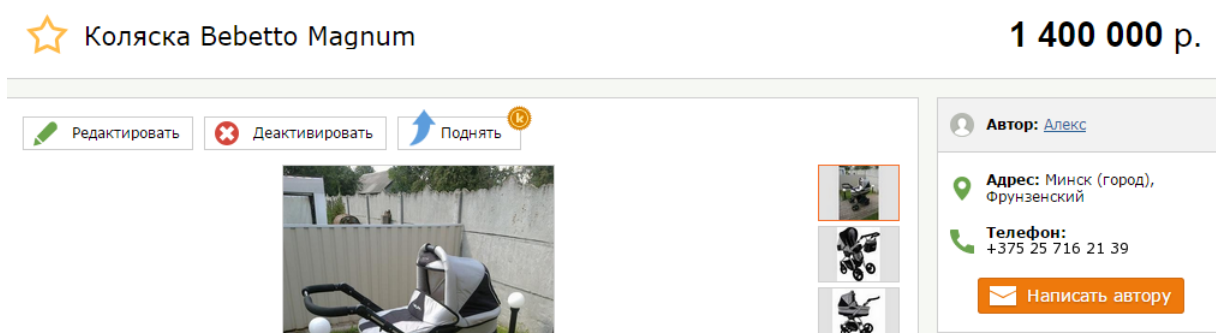


Рис. 4: Контактная информация владельца объявления

осуществляет проверку на цензуру, на сам контент объявления. Это делает сайты приемлимыми и культурными.

Также на многих сайтах объявлений обычно предусмотрена функция редактирования профиля. Пользователи могут указывать свои данные, к примеру: фамилию, имя, телефон, адрес, и тд, если это необходимо. Так клиенту проще связаться с лицом, разместившим объявление.

На продвинутых сайтах, таких как aliexpress.com можно регистрироваться через сторонние сервисы (Рис. 5). Эта область на данный момент развивается. На многие сайты можно зайти через facebook, vk, google+, yahoo, linkedIn, youtube. Пользователи находят это удобным и предпочитают аутентифицироваться через сторонние сервисы.

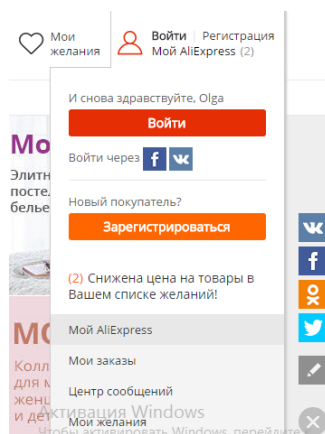


Рис. 5: Аутентификация через сторонние сервисы

На данном этапе развития информационных технологий большинство пользователей заинтересовались тэгами. Их можно оставлять везде и всему. Любой сущности можно оставить свой комментарий в виде тэга и не один. Данное приспособление привлекает пользователей, поскольку удобно осуществлять фильтрацию по тэгам. И любой объект можно пометить чем-то своим. Удобно осуществлять поиск по тэгам и таким образом сразу находить интересующий нас предмет/область/статью.

На сайте kufar.by реализован поиск по ключевым словам (Рис. 6). Это обеспечивает технология поиска по ключевым словам сущности, причем не важно, какое это ключевое слово: тэг, название или описание. Так же предусмотрена функция автодополнения, что делает поиск более удобным пользователю.

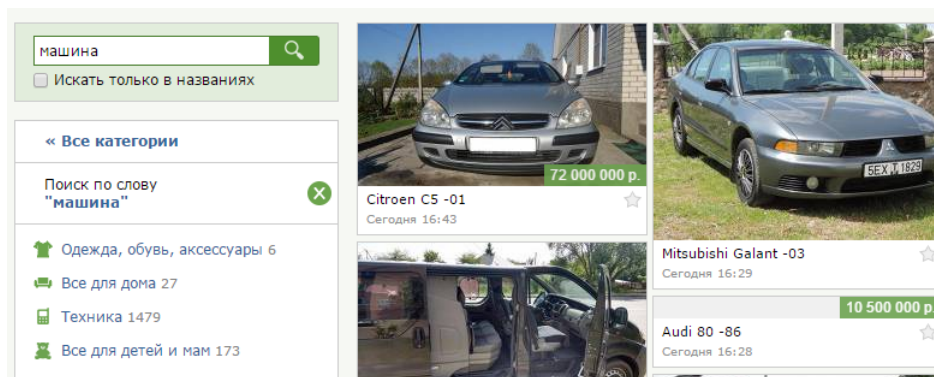


Рис. 6: Поиск объявлений

Большинство сайтов сейчас работают быстро, однако наиболее удобным является технология обновления какой-либо части веб-страницы без ее перезагрузки. К примеру, чтобы "лайк" ставился сразу, чтобы объявление помечалось как "избранное" по щелчку. Поэтому большинство разработчиков стараются предусмотреть такой поворот событий и сделать свое веб-приложение наиболее удобным и комфортным для пользователей (Рис. 7).

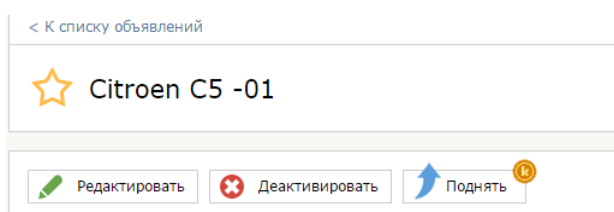


Рис. 7: Добавление в избранное

## 2.2 Постановка задачи

Разработать программное средство “Сайт Доска Объявлений” с использованием языка программирования C#.

Реализовать функциональный набор:

1. Регистрация/вход пользователя в систему;
2. Просмотр объявлений, размещенных на сайте;

3. Добавление, изменение, удаление объявлений для зарегистрированных пользователей;
4. Создание ролей пользователей: администратор и обычный пользователь;
5. Поиск объявлений;
6. Добавление изображений для объявлений;
7. Предусмотреть аутентификацию через сторонние сервисы(vk, google+, facebook), реализовать возможность редактирования профиля (изменение аутентификационных данных, загрузка аватара), реализовать админку (главный администратор сайта проверяет каждое объявление и разрешает или запрещает его публикацию)

Приложение разрабатывать на платформе ASP.NET MVC

## 3 РАЗРАБОТКА ПРИЛОЖЕНИЯ

### 3.1 Описание базы данных

В данном проекте применялся MS SQL Server для построения базы данных. Доступ к базе осуществляется с помощью Entity Framework (EF) - инфраструктуры ORM для платформы .NET. При помощи Entity Framework таблицы, столбцы и строки реляционной базы данных представляются с помощью обычных объектов C#. Платформа .NET поддерживает другие инфраструктуры ORM, которые используют разные подходы к организации и управлению данными, но была выбрана Entity Framework по следующим причинам:

- EF проста и ее легко запустить в работу;
- EF интегрирована с LINQ;
- EF обладает большими функциональными возможностями;

Диаграмма IDEF1X базы данных веб-приложения доска объявлений представлена на рисунке 8:

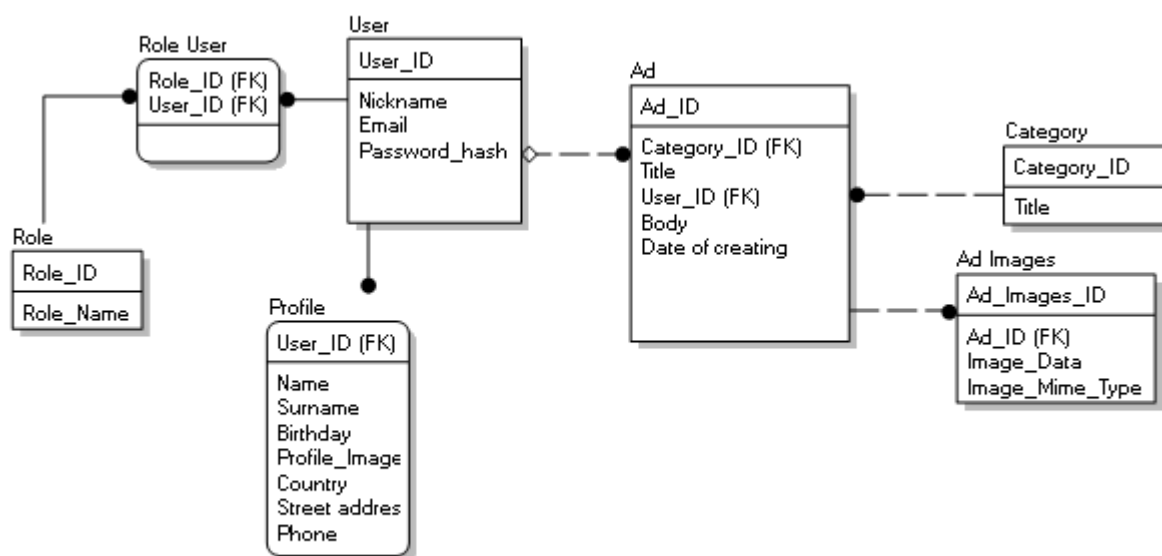


Рис. 8: Диаграмма IDEF1X базы данных

Основной таблицей является Ad. У отдельного экземпляра сущности из таблицы Ad может быть 1 категория объявления, название, описание, дата публикации и уникальный идентификатор пользователя, разместившего объявление.

Структура данных представлена в виде классов.



## 3.2 Используемые классы

Используемые классы:

1)EFAdRepository. Данный класс обеспечивает работу с данными таблиц Ad и Image базы данных. Предоставляет интерфейс для CRUD операций. Содержит поле контекста базы данных (EFDbContext), методы удаления, добавления, сохранения объявлений или картинок.

2)EFDbContext. Класс, устанавливающий связь с таблицами Ad и Image базы данных.

3)Ad, Image. Классы, которые являются объектно-ориентированными представлениями табли Ad и Image базы данных.

4)AccountController. Контроллер, предоставляющий методы регистрации, логина, изменения данных пользователя. Реализуется при помощи Identity 2.0.

5)AdController. Контроллер, предназначенный для работы пользователся с объявлениями (добавление, удаление, просмотр, изменение).

6)AdminController. Контроллер, предназначенный для работы с объявлениями пользователя, обладающего правами администратора.

7)ImageController. Контроллер, в котором содержатся методы загрузки и отображение картинок.

8)UserController. Контроллер, предназначенный для работы пользователя со своими персональными данными и личным кабинетом.

9)NavController. Контроллер, предоставляющий методы для работы с навигационной панелью сайта.

10)IdentityModel. Класс, предоставляющий стандартные модели пользователя в Identity 2.0.

11)UserAdsViewModel. Модель представления, содержащая в себе информацию о конкретном пользователе и все объявления данного пользователя.

12)PagingInfo. Класс, содержащий информацию о пагинации страниц в приложении (общее количество объявлений, максимальное количество объявлений, размещенных на одной странице).

Каждая модель имеет соответствующий контроллер: AdController, UserController, ImageController

Все роуты прописываются в классе RouteConfig, где они заносятся в таблицу роутов, по которой в дальнейшем определяется к какому контроллеру и действию был произведен запрос.

Для организации более эффективной передачи файлов скриптов и стилей, а также для минификации, в классе BundleConfig регистрируются необходимые бандлы.

Что касается представлений, то они написаны с использованием html и движка Razor. Также были разработаны html-хелперы, позволяющие добавлять необходимые html-теги с помощью кода на языке C#.

## 4 ТЕСТИРОВАНИЕ

Приложение Ad Board покрыто некоторыми тестами. Основная идея юнит тестирования – тестирование отдельных компонентов программы, т.е. классов и их методов. Имеются тесты контроллера, моделей. Для написания тестов использовался Unit Testing Framework NUnit и фреймворк Moq, который позволяют имитировать или эмулировать какую-то функциональность или создавать мок-объекты.

Тесты контроллера включают в себя тестирование:

- AdController;
- UserController;
- AdminController;

Тесты контроллера позволяют протестировать методы контроллера, такие как проверка генерации результата в виде представления, проверка содержимого объявлений и т.д.

Тесты моделей предусматривают тестирование методов класса. В приложении Ad Board была протестирована модель Paging Info.

В приложении были протестированы методы нескольких контроллеров, а также методы моделей. Это свидетельствует о том, что приложение не полностью покрыто тестами, следовательно добавление нового функционала может привести к особым трудностям, поскольку не весь код можно проверить тестами.

## 5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Так выглядит главная страница сайта.

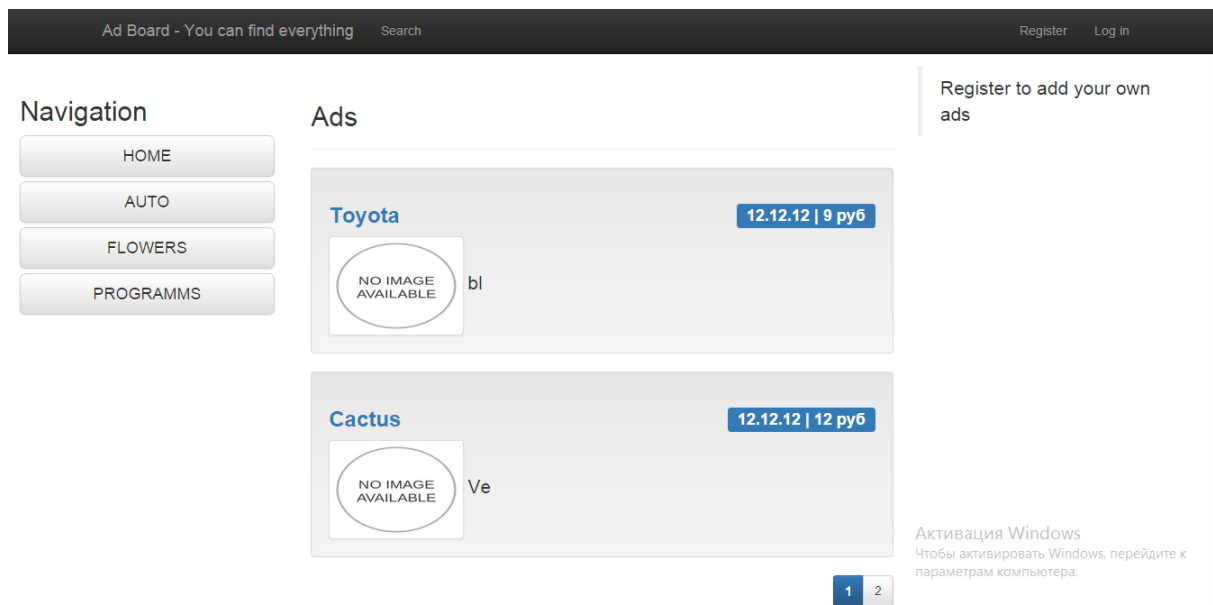


Рис. 9: Главная страница Ad Board

Для того что бы разместить объявление пользователю необходимо войти на сайт (Рис.10). Для этого он может использовать сторонние сервисы.

### Log in.

Use a local account to log in.

Email

Password

☐

Remember me?

LOG IN

[Register as a new user](#)

Use another service to log in.

VKONTAKTE

GOOGLE

FACEBOOK

Рис. 10: Вход на сайт

Если пользователь хочет зарегистрироваться на сайте он должен заполнить данную форму (Рис. 11).

---

## Register.

Create a new account.

---

Email*	<input type="text"/>
Password*	<input type="password"/>
Confirm password*	<input type="password"/>
Name*	<input type="text"/>
Surname*	<input type="text"/>
Country*	<input type="text"/>
Street Address*	<input type="text"/>
MobilePhone*	<input type="text"/>
Image	<a href="#">CHOOSE IMAGE...</a>

Fields marked without asterisk(\*) are optional


---

Рис. 11: Регистрация

Для изменения своих контактных данных пользователь может воспользоваться редактированием профиля (Рис. 12).

## Edit Profile

---

 <input type="button" value="Выберите файл"/> Файл не выбран	<b>Personal info</b>	
	Name	<input type="text" value="admin"/>
	Surname	<input type="text" value="admin"/>
	Email	<input type="text" value="admin@google.com"/>
	Country	<input type="text" value="Country"/>
	Street Address	<input type="text" value="Address"/>
	MobilePhone	<input type="text" value="375 (21) 312-31-23"/>
	Current password	<input type="password"/>
	Confirm new password	<input type="password"/>
	New password	<input type="password"/>
<input type="button" value="REGISTER"/>		

---

Рис. 12: Редактирование профиля

Для того чтобы найти необходимые объявления используется поиск (Рис. 13).

The interface is titled "Search Ads". It features a search bar with the text "t" and a dropdown arrow, followed by a "SEARCH" button. Below the search bar, there are two search results displayed in a list. The first result is for "Toyota" with the text "blaallblalblab" and a price tag "12.12.12 | 9 руб". The second result is for "Cactus" with the text "Very very small cactus hitler" and a price tag "12.12.12 | 12 руб".

Рис. 13: Поиск объявлений

Для просмотра объявлений по нужной категории используется навигационная панель. Соответствующая категория закрашивается, когда данная категория выбрана (Рис. 14).


The interface is divided into two main sections: "Navigation" and "Ads". The "Navigation" section on the left contains four buttons: "HOME", "AUTO", "FLOWERS", and "PROGRAMMS". The "AUTO" button is highlighted in blue, indicating it is the selected category. The "Ads" section on the right displays a search result for "Toyota" with the text "bl" and a price tag "12.12.12 | 9 руб". A placeholder image for the Toyota car is shown with the text "NO IMAGE AVAILABLE". A small blue button with the number "1" is located at the bottom right of the "Ads" section.

Рис. 14: Выбранная категория

Полная информация об объявлении предоставлена на рисунке 15.

## Toyota

**Author:**




Карта кодов № 38790			
1 - 9244	11 - 4459	21 - 9423	31 - 5000
2 - 9979	12 - 7804	22 - 9834	32 - 8954
3 - 7413	13 - 9195	23 - 6259	33 - 5388
4 - 7926	14 - 9566	24 - 6050	34 - 5998
5 - 0070	15 - 2881	25 - 3356	35 - 5965
6 - 4958	16 - 6031	26 - 1122	36 - 3704
7 - 1399	17 - 2902	27 - 9389	37 - 1017
8 - 3856	18 - 2201	28 - 1844	38 - 2583
9 - 3220	19 - 5575	29 - 6391	39 - 6005
10 - 0016	20 - 0045	30 - 5182	40 - 4990

Asdk Saodk

aodi  
sioa  
P: 375 (12) 321-31-23 user@gmail.com

**Toyota**



NO IMAGE AVAILABLE

blaallblalblab

12.12.12 | 9 руб

Рис. 15: Информация об объявлении


Просмотр объявлений залогинившегося пользователя предоставлен на рисунке 16.

## My Ads

[+ ADD NEW](#)

### Your ads:

**Cactus**



NO IMAGE AVAILABLE

12.12.12 | 12 руб

Very very small cactus hitler

[EDIT](#)

[EDIT PHOTOS](#)

[DELETE](#)

1 2

Рис. 16: Мои объявления

Добавление нового объявления может осуществлять только залогинившийся пользователь. Форма добавления объявления показана на рисунке 17.

The form is titled 'Name' and contains several input fields. The 'Name' field has the value 'Pudge'. The 'Description' field has the value 'asdasjd'. The 'Date' field has the value '12/12/12'. The 'Category' field has the value 'Dota'. The 'Price' field has the value '12'. At the bottom, there are two buttons: 'СОХРАНИТЬ' (Save) and 'CANCEL CHANGES AND RETURN TO LIST'.

Рис. 17: Добавление объявления

Форма редактирования объявления предоставлена на рисунке 18.

The form is titled 'Editing Ad Cactus'. It contains several input fields. The 'Name' field has the value 'Cactus'. The 'Description' field has the value 'Very very small cactus hitler'. The 'Date' field has the value '12//', and there is a message below it: 'The value '12//' is not valid for Date.'. The 'Category' field has the value 'Flowers'. The 'Price' field has the value '12'. At the bottom, there are two buttons: 'СОХРАНИТЬ' (Save) and 'CANCEL CHANGES AND RETURN TO LIST'.

Рис. 18: Редактирование объявления

Также в данном приложении есть панель администратора, которая доступна только тем пользователям, которые наделены соответствующими правами администратора(Рис.19).

Рис. 19: Панель администратора

Администратор может удалять (Рис. 20) и редактировать объявления (Рис. 21).

List of ads			
ID	Name	Price	Action
18	<a href="#">Toyota</a>	9	<a href="#">Delete</a>
20	<a href="#">Cactus</a>	12	<a href="#">Delete</a>
21	<a href="#">Skype</a>	10012	<a href="#">Delete</a>

Рис. 20: Панель администратора: список объявлений

### Editing of Toyota

Name

Description

Date

Cateogry

Price

Рис. 21: Панель администратора: редактирование объявления



## ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была изучена возможность регистрации пользователей через Identity. Также были изучены основы реляционных баз данных, в частности MSSql. Были выполнены следующие требования:

- Возможность создания и удаления пользователя;
- Возможность редактирования профиля пользователя (включает загрузку аватара и изменения личных и контактных данных);
- Возможность создания объявления;
- Возможность редактирования и удаления объявления (загрузка фотографий, добавление подробного описания, личных данных человека, разместившего объявление);
- Создание ролей пользователей: админ и простой пользователь. У админа больше полномочий чем у обычного пользователя;
- Технология поиска объявлений ( поиск по названию, категории);
- Предусмотрена аутентификация через сторонние сервисы (vk, google+, facebook);
- Реализованы дополнительные возможности администраторов сайта (запрещение нецензурных публикаций, просмотр чужих объявлений перед публикацией);

## СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

- [1] Википедия [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://ru.wikipedia.org/wiki/>
- [2] Pro Asp.Net MVC - Adam Freeman
- [3] CLR via C# - Jeffrey Richter
- [4] Pro C# 5.0 and the .NET 4.5 Framework - Andrew Troelsen

**ПРИЛОЖЕНИЕ А**  
**(обязательное)**  
**Исходный код программы**

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.ComponentModel.DataAnnotations;
7 using System.Web.Mvc;
8
9
10 namespace AdBoard.Domain.Entities
11 {
12     public class Ad
13     {
14         [HiddenInput(DisplayValue=false)]
15         public int Id { get; set; }
16
17         [Required(ErrorMessage="Please , enter ad name")]
18         [Display(Name="Name")]
19         public string Name { get; set; }
20
21         [DataType(DataType.MultilineText)]
22         [Required(ErrorMessage="Please , enter description for ad")]
23         public string Description { get; set; }
24
25         [DataType(DataType.Date)]
26         [DisplayFormat(DataFormatString="{0:dd/MM/yyyy}",
27             ApplyFormatInEditMode=true)]
28         [Required(ErrorMessage="Please , enter date for ad(use slash
29             \"/\")")]
30         public DateTime Date { get; set; }
31
32         [Required(ErrorMessage="Please , enter ad category")]
33         [Display(Name="Cateogry")]
34         public string Category { get; set; }
35
36         [HiddenInput(DisplayValue=false)]
37         public string UserId { get; set; }
38
39         [Required(ErrorMessage = "Please , enter ad price")]
40         [DataType(DataType.Currency)]
41         public decimal Price { get; set; }
42
43         [HiddenInput(DisplayValue = true)]
44         public IEnumerable<Image> Images { get; set; }
```

```

43     }
44 }
45
46 using System;
47 using System.Collections.Generic;
48 using System.Linq;
49 using System.Text;
50
51 namespace AdBoard.Domain.Entities
52 {
53     public class Image
54     {
55         public int Id { get; set; }
56         public int AddId { get; set; }
57         public byte[] ImageData { get; set; }
58         public string ImageMimeType { get; set; }
59     }
60 }
61
62 using AdBoard.Domain.Entities;
63 using System;
64 using System.Collections.Generic;
65 using System.Linq;
66 using System.Text;
67 using System.Threading.Tasks;
68
69 namespace AdBoard.Domain.Abstract
70 {
71     public interface IAdRepository
72     {
73         IEnumerable<Ad> Ads { get; }
74         IEnumerable<Image> Images { get; }
75         void SaveAd(Ad ad);
76         Ad DeleteAd(int addId);
77         void SaveImage(Image image);
78     }
79 }
80
81 using AdBoard.Domain.Entities;
82 using System;
83 using System.Collections.Generic;
84 using System.Data.Entity;
85 using System.Linq;
86 using System.Text;
87 using System.Threading.Tasks;
88
89 namespace AdBoard.Domain.Concrete
90 {
91     public class EFDbContext : DbContext
92     {
93         public DbSet<Ad> Ads { get; set; }
94         public DbSet<Image> Images { get; set; }

```

```

95     }
96 }
97
98 using AdBoard.Domain.Abstract;
99 using AdBoard.Domain.Entities;
100 using System;
101 using System.Collections.Generic;
102 using System.Linq;
103 using System.Text;
104 using System.Threading.Tasks;
105
106 namespace AdBoard.Domain.Concrete
107 {
108     public class EFAdRepository : IAdRepository, IDisposable
109     {
110         EFDbContext context = new EFDbContext();
111
112         public virtual IEnumerable<Entities.Ad> Ads
113         {
114             get { return context.Ads; }
115         }
116
117         public IEnumerable<Entities.Image> Images
118         {
119             get { return context.Images; }
120         }
121
122         public void SaveAd(Ad ad)
123         {
124             if (ad.Id == 0)
125                 context.Ads.Add(ad);
126             else
127             {
128                 Ad dbEntry = context.Ads.Find(ad.Id);
129                 if (dbEntry != null)
130                 {
131                     dbEntry.Name = ad.Name;
132                     dbEntry.Description = ad.Description;
133                     dbEntry.Price = ad.Price;
134                     dbEntry.Category = ad.Category;
135                     dbEntry.Date = ad.Date;
136                 }
137             }
138             context.SaveChanges();
139         }
140
141         public Ad DeleteAd(int adId)
142         {
143             Ad dbEntry = context.Ads.Find(adId);
144             if (dbEntry != null)
145             {
146                 context.Ads.Remove(dbEntry);

```

```

147         context.SaveChanges();
148     }
149     return dbEntry;
150 }
151
152 public void SaveImage(Image image)
153 {
154     if (image.Id == 0)
155         context.Images.Add(image);
156     else
157     {
158         Image dbEntry = context.Images.Find(image.Id);
159         if (dbEntry != null)
160         {
161             dbEntry.AddId = image.AddId;
162             dbEntry.Id = image.Id;
163             dbEntry.ImageData = image.ImageData;
164             dbEntry.ImageMimeType = image.ImageMimeType;
165         }
166     }
167     context.SaveChanges();
168 }
169
170 public void Save()
171 {
172     context.SaveChanges();
173 }
174
175 protected void DisposeContext(bool disposing)
176 {
177     if (disposing)
178     {
179         if (context != null)
180         {
181             context.Dispose();
182             context = null;
183         }
184     }
185 }
186
187 public void Dispose()
188 {
189     DisposeContext(true);
190     GC.SuppressFinalize(this);
191 }
192 }
193 }
194 }
195
196 using System.Web;
197 using System.Web.Optimization;
198

```

```

199 namespace AdBoard.WebUI
200 {
201     public class BundleConfig
202     {
203         // For more information on bundling, visit http://go.
204         microsoft.com/fwlink/?LinkId=301862
205         public static void RegisterBundles(BundleCollection bundles)
206         {
207             bundles.Add(new ScriptBundle("~/bundles/jquery").Include
208                 (
209                     "~/Scripts/jquery-{version}.js"));
210
211             bundles.Add(new ScriptBundle("~/bundles/jqueryval").
212                 Include(
213                     "~/Scripts/jquery.validate*"));
214
215             // Use the development version of Modernizr to develop
216             with and learn from. Then, when you're
217             // ready for production, use the build tool at http://
218             modernizr.com to pick only the tests you need.
219             bundles.Add(new ScriptBundle("~/bundles/modernizr").
220                 Include(
221                     "~/Scripts/modernizr-*"));
222
223             bundles.Add(new ScriptBundle("~/bundles/bootstrap").
224                 Include(
225                     "~/Scripts/bootstrap.js",
226                     "~/Scripts/respond.js"));
227
228             bundles.Add(new StyleBundle("~/Content/css").Include(
229                 "~/Content/bootstrap.css",
230                 "~/Content/site.css"));
231
232             bundles.Add(new ScriptBundle("~/bundles/jquerymask").
233                 Include(
234                     "~/Scripts/jquery.mask.min.js",
235                     "~/Scripts/maskedinput-binder.js"));
236
237             bundles.Add(new ScriptBundle("~/bundles/jasny-bootstrap")
238                 ).Include(
239                 "~/Scripts/jasny-bootstrap.min.js"));
240
241             bundles.Add(new ScriptBundle("~/bundles/imageload").
242                 Include(
243                     "~/Scripts/imageload.js"));
244
245             bundles.Add(new ScriptBundle("~/bundles/sitescripts").
246                 Include(
247                     "~/Scripts/sitescripts.js"));
248
249             // Set EnableOptimizations to false for debugging. For
250             more information ,

```

```

239         // visit http://go.microsoft.com/fwlink/?LinkId=301862
240         BundleTable.EnableOptimizations = true;
241     }
242 }
243 }
244
245 using System.Web;
246 using System.Web.Mvc;
247
248 namespace AdBoard.WebUI
249 {
250     public class FilterConfig
251     {
252         public static void RegisterGlobalFilters(
253             GlobalFilterCollection filters)
254         {
255             filters.Add(new HandleErrorAttribute());
256         }
257     }
258
259     using System.Threading.Tasks;
260     using Microsoft.AspNet.Identity;
261     using Microsoft.AspNet.Identity.EntityFramework;
262     using Microsoft.AspNet.Identity.Owin;
263     using Microsoft.Owin;
264     using AdBoard.WebUI.Models;
265
266     namespace AdBoard.WebUI
267     {
268         // Configure the application user manager used in this
269         application. UserManager is defined in ASP.NET Identity and
270         is used by the application.
271
272         public class ApplicationUserManager : UserManager<
273             ApplicationUser>
274         {
275             public ApplicationUserManager(IUserStore<ApplicationUser>
276                 store)
277                 : base(store)
278             {
279             }
280
281             public static ApplicationUserManager Create(
282                 IdentityFactoryOptions<ApplicationUserManager> options,
283                 IOwinContext context)
284             {
285                 var manager = new ApplicationUserManager(new UserStore<
286                     ApplicationUser>(context.Get<ApplicationDbContext>())
287                 );
288                 // Configure validation logic for usernames
289                 manager.UserValidator = new UserValidator<

```



```

282         ApplicationUser >(manager)
283     {
284         AllowOnlyAlphanumericUserNames = false ,
285         RequireUniqueEmail = true
286     };
287     // Configure validation logic for passwords
288     manager.PasswordValidator = new PasswordValidator
289     {
290         RequiredLength = 6,
291         RequireNonLetterOrDigit = false ,
292         RequireDigit = true ,
293         RequireLowercase = true ,
294         RequireUppercase = false ,
295     };
296     // Register two factor authentication providers. This
297     // application uses Phone and Emails as a step of
298     // receiving a code for verifying the user
299     // You can write your own provider and plug in here.
300     manager.RegisterTwoFactorProvider("PhoneCode", new
301     PhoneNumberTokenProvider<ApplicationUser>
302     {
303         MessageFormat = "Your security code is: {0}"
304     });
305     manager.RegisterTwoFactorProvider("EmailCode", new
306     EmailTokenProvider<ApplicationUser>
307     {
308         Subject = "Security Code",
309         BodyFormat = "Your security code is: {0}"
310     });
311     manager.EmailService = new EmailService();
312     manager.SmsService = new SmsService();
313     var dataProtectionProvider = options.
314     DataProtectionProvider;
315     if (dataProtectionProvider != null)
316     {
317         manager.UserTokenProvider = new
318         DataProtectorTokenProvider<ApplicationUser>(
319             dataProtectionProvider.Create("ASP.NET Identity")
320         );
321     }
322     return manager;
323 }
324 }

```

```

317 public class EmailService : IIdentityMessageService
318 {
319     public Task SendAsync(IdentityMessage message)
320     {
321         // Plug in your email service here to send an email.
322         return Task.FromResult(0);
323     }
324 }

```

```

325
326 public class SmsService : IIdentityMessageService
327 {
328     public Task SendAsync(IIdentityMessage message)
329     {
330         // Plug in your sms service here to send a text message.
331         return Task.FromResult(0);
332     }
333 }
334 }
335
336 [assembly: WebActivatorEx.PreApplicationStartMethod(typeof(AdBoard.
    WebUI.App_Start.NinjectWebCommon), "Start")]
337 [assembly: WebActivatorEx.ApplicationShutdownMethodAttribute(typeof(
    AdBoard.WebUI.App_Start.NinjectWebCommon), "Stop")]
338
339 namespace AdBoard.WebUI.App_Start
340 {
341     using System;
342     using System.Web;
343
344     using Microsoft.Web.Infrastructure.DynamicModuleHelper;
345
346     using Ninject;
347     using Ninject.Web.Common;
348     using System.Web.Mvc;
349     using AdBoard.WebUI.Infrastructure;
350
351     public static class NinjectWebCommon
352     {
353         private static readonly Bootstrapper bootstrapper = new
            Bootstrapper();
354
355         /// <summary>
356         /// Starts the application
357         /// </summary>
358         public static void Start()
359         {
360             DynamicModuleUtility.RegisterModule(typeof(
                OnePerRequestHttpModule));
361             DynamicModuleUtility.RegisterModule(typeof(
                NinjectHttpModule));
362             bootstrapper.Initialize(CreateKernel);
363         }
364
365         /// <summary>
366         /// Stops the application.
367         /// </summary>
368         public static void Stop()
369         {
370             bootstrapper.ShutDown();
371         }

```

```

372
373     /// <summary>
374     /// Creates the kernel that will manage your application.
375     /// </summary>
376     /// <returns>The created kernel.</returns>
377     private static IKernel CreateKernel()
378     {
379         var kernel = new StandardKernel();
380         try
381         {
382             kernel.Bind<Func<IKernel>>().ToMethod(ctx => () =>
383                 new Bootstrapper().Kernel);
384             kernel.Bind<IHttpModule>().To<
385                 HttpApplicationInitializationHttpModule>();
386
387             RegisterServices(kernel);
388             return kernel;
389         }
390         catch
391         {
392             kernel.Dispose();
393             throw;
394         }
395     }
396
397     /// <summary>
398     /// Load your modules or register your services here!
399     /// </summary>
400     /// <param name="kernel">The kernel.</param>
401     private static void RegisterServices(IKernel kernel)
402     {
403         DependencyResolver.SetResolver(new
404             NinjectDependencyResolver(kernel));
405     }
406 }
407
408 using System;
409 using System.Collections.Generic;
410 using System.Linq;
411 using System.Web;
412 using System.Web.Mvc;
413 using System.Web.Routing;
414
415 namespace AdBoard.WebUI
416 {
417     public class RouteConfig
418     {
419         public static void RegisterRoutes(RouteCollection routes)
420         {
421             routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

```

```

421 routes.MapRoute(null,
422     "",
423     new
424     {
425         controller = "Ad",
426         action = "List",
427         category = (string)null,
428         page = 1
429     });
430
431 routes.MapRoute(
432     null,
433     "UserAds{page}",
434     new { controller = "User", action = "UserAds" },
435     constraints: new { page = @"\\d+" }
436 );
437
438 routes.MapRoute(
439     null,
440     "UserAds",
441     new { controller = "User", action = "UserAds" }
442 );
443
444 routes.MapRoute(
445     null,
446     "EditProfile",
447     new { controller = "Account", action = "EditProfile"
448         }
449 );
450
451 routes.MapRoute(
452     null,
453     "Search",
454     new { controller = "Ad", action = "SearchAds" }
455 );
456
457 routes.MapRoute(
458     name: null,
459     url: "AdInfo{id}",
460     defaults: new { controller = "Ad", action = "AdInfo"
461         },
462     constraints: new { id = @"\\d+" }
463 );
464
465 routes.MapRoute(
466     name: null,
467     url: "Page{page}",
468     defaults: new { controller = "Ad", action = "List",
469         category = (string)null },
470     constraints: new { page = @"\\d+" }
471 );

```

```

470         routes.MapRoute(
471             null,
472             "{category}",
473             new { controller = "Ad", action = "List", page = 1 }
474         );
475
476         routes.MapRoute(
477             null,
478             "{category}/Page{page}",
479             new { controller = "Ad", action = "List" },
480             new { page = @"\d+" }
481         );
482
483         routes.MapRoute(null, "{controller}/{action}");
484     }
485 }
486 }
487
488 using Microsoft.AspNet.Identity;
489 using Microsoft.AspNet.Identity.EntityFramework;
490 using Microsoft.AspNet.Identity.Owin;
491 using Microsoft.Owin;
492 using Microsoft.Owin.Security.Cookies;
493 using Microsoft.Owin.Security.DataProtection;
494 using Microsoft.Owin.Security.Google;
495 using Owin;
496 using System;
497 using AdBoard.WebUI.Models;
498
499 namespace AdBoard.WebUI
500 {
501     public partial class Startup
502     {
503         // For more information on configuring authentication ,
504         please visit http://go.microsoft.com/fwlink/?LinkId
505         =301864
506         public void ConfigureAuth(IAppBuilder app)
507         {
508             // Configure the db context and user manager to use a
509             single instance per request
510             app.CreatePerOwinContext(ApplicationDbContext.Create);
511             app.CreatePerOwinContext<ApplicationUserManager>(
512                 ApplicationUserManager.Create);
513
514             // Enable the application to use a cookie to store
515             information for the signed in user
516             // and to use a cookie to temporarily store information
517             about a user logging in with a third party login
518             provider
519             // Configure the sign in cookie
520             app.UseCookieAuthentication(new
521                 CookieAuthenticationOptions

```

```

514 {
515     AuthenticationType = DefaultAuthenticationTypes.
        ApplicationCookie,
516     LoginPath = new PathString("/Account/Login"),
517     Provider = new CookieAuthenticationProvider
518     {
519         OnValidateIdentity = SecurityStampValidator.
            OnValidateIdentity<ApplicationUserManager,
            ApplicationUser>(
520             validateInterval: TimeSpan.FromMinutes(30),
521             regenerateIdentity: (manager, user) => user.
                GenerateUserIdentityAsync(manager))
522     };
523 });
524
525 app.UseExternalSignInCookie(DefaultAuthenticationTypes.
    ExternalCookie);
526
527 // Uncomment the following lines to enable logging in
    with third party login providers
528 //app.UseMicrosoftAccountAuthentication(
529 //     clientId: "",
530 //     clientSecret: "");
531
532 //app.UseTwitterAuthentication(
533 //     consumerKey: "",
534 //     consumerSecret: "");
535
536 app.UseFacebookAuthentication(
537     appId: "1636839839860813",
538     appSecret: "74abcaff1ced5606bade7524a2cf64");
539
540 app.UseGoogleAuthentication(new
    GoogleOAuth2AuthenticationOptions()
541 {
542     ClientId = "1051820490822-
        b8mo77iknruoecm45kftnncr0l2s2qp1.apps.
        googleusercontent.com",
543     ClientSecret = "dW-HN0Vi-vJFOTmv1pEMiQXQ"
544 });
545
546 app.UseVkontakteAuthentication(appId: "4856605",
    appSecret: "mEtzKiP8KSNKi6hNigJV", scope: "email");
547 }
548 }
549 }
550
551 using System;
552 using System.Collections.Generic;
553 using System.Linq;
554 using System.Security.Claims;
555 using System.Threading.Tasks;

```

```

556 using System.Web;
557 using System.Web.Mvc;
558 using Microsoft.AspNet.Identity;
559 using Microsoft.AspNet.Identity.EntityFramework;
560 using Microsoft.AspNet.Identity.Owin;
561 using Microsoft.Owin.Security;
562 using Owin;
563 using AdBoard.WebUI.Models;
564
565 namespace AdBoard.WebUI.Controllers
566 {
567     [Authorize]
568     public class AccountController : Controller
569     {
570         private ApplicationUserManager _userManager;
571
572         public AccountController()
573         {
574         }
575
576         public AccountController(ApplicationUserManager userManager)
577         {
578             UserManager = userManager;
579         }
580
581         public ApplicationUserManager UserManager {
582             get
583             {
584                 return _userManager ?? HttpContext.GetOwinContext().
                    GetUserIdentityManager<ApplicationUserManager>();
585             }
586             private set
587             {
588                 _userManager = value;
589             }
590         }
591
592         //
593         // GET: /Account/Login
594         [AllowAnonymous]
595         public ActionResult Login(string returnUrl)
596         {
597             ViewBag.ReturnUrl = returnUrl;
598             return View();
599         }
600
601         //
602         // POST: /Account/Login
603         [HttpPost]
604         [AllowAnonymous]
605         [ValidateAntiForgeryToken]
606         public async Task<ActionResult> Login(LoginViewModel model,

```

```

607         string returnUrl)
608     {
609         if (ModelState.IsValid)
610         {
611             var user = await UserManager.FindAsync(model.Email,
612                 model.Password);
613             if (user != null)
614             {
615                 await SignInAsync(user, model.RememberMe);
616                 return RedirectToLocal(returnUrl);
617             }
618             else
619             {
620                 ModelState.AddModelError("", "Invalid username
621                     or password.");
622             }
623         }
624         // If we got this far, something failed, redisplay form
625         return View(model);
626     }
627
628     // GET: /Account/Register
629     [AllowAnonymous]
630     public ActionResult Register()
631     {
632         return View();
633     }
634
635     // POST: /Account/Register
636     [HttpPost]
637     [AllowAnonymous]
638     [ValidateAntiForgeryToken]
639     public async Task<ActionResult> Register(RegisterViewModel
640         model, HttpPostedFileBase image = null)
641     {
642         if (ModelState.IsValid)
643         {
644             var user = new ApplicationUser() { UserName = model.
645                 Email, Email = model.Email, Name = model.Name,
646                 Surname = model.Surname, Country = model.Country
647                 , StreetAddress = model.StreetAddress,
648                 MobilePhone = model.MobilePhone, ImageData =
649                 model.ImageData, ImageMimeType = model.
650                 ImageMimeType,
651                 FavoritesAds = model.FavoritesAds };
652
653             if (image != null)
654             {
655                 user.ImageMimeType = image.ContentType;

```



```

651         user.ImageData = new byte[image.ContentLength];
652         image.InputStream.Read(user.ImageData, 0, image.
            ContentLength);
653     }
654
655     IdentityResult result = await UserManager.
        CreateAsync(user, model.Password);
656     if (result.Succeeded)
657     {
658         await UserManager.AddToRoleAsync(user.Id, "user"
            );
659         await SignInAsync(user, isPersistent: false);
660
661         // For more information on how to enable account
            confirmation and password reset please visit
            http://go.microsoft.com/fwlink/?LinkID
            =320771
662         // Send an email with this link
663         // string code = await UserManager.
            GenerateEmailConfirmationTokenAsync(user.Id);
664         // var callbackUrl = Url.Action("ConfirmEmail",
            "Account", new { userId = user.Id, code =
            code }, protocol: Request.Url.Scheme);
665         // await UserManager.SendEmailAsync(user.Id, "
            Confirm your account", "Please confirm your
            account by clicking <a href=\"" + callbackUrl
            + "\">here</a>");
666
667         return RedirectToAction("UserAds", "User");
668     }
669     else
670     {
671         AddErrors(result);
672     }
673 }
674
675 // If we got this far, something failed, redisplay form
676 return View(model);
677 }
678
679 //
680 // GET: /Account/ConfirmEmail
681 [AllowAnonymous]
682 public async Task<ActionResult> ConfirmEmail(string userId,
    string code)
683 {
684     if (userId == null || code == null)
685     {
686         return View("Error");
687     }
688
689     IdentityResult result = await UserManager.

```

```

        ConfirmEmailAsync(userId, code);
        if (result.Succeeded)
        {
            return View("ConfirmEmail");
        }
        else
        {
            AddErrors(result);
            return View();
        }
    }

    //
    // GET: /Account/ForgotPassword
    [AllowAnonymous]
    public ActionResult ForgotPassword()
    {
        return View();
    }

    //
    // POST: /Account/ForgotPassword
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> ForgotPassword(
        ForgotPasswordViewModel model)
    {
        if (ModelState.IsValid)
        {
            var user = await UserManager.FindByNameAsync(model.
                Email);
            if (user == null || !(await UserManager.
                IsEmailConfirmedAsync(user.Id)))
            {
                ModelState.AddModelError("", "The user either
                    does not exist or is not confirmed.");
                return View();
            }

            // For more information on how to enable account
            // confirmation and password reset please visit http
            // ://go.microsoft.com/fwlink/?LinkId=320771
            // Send an email with this link
            // string code = await UserManager.
            //     GeneratePasswordResetTokenAsync(user.Id);
            // var callbackUrl = Url.Action("ResetPassword", "
            //     Account", new { userId = user.Id, code = code },
            //     protocol: Request.Url.Scheme);
            // await UserManager.SendEmailAsync(user.Id, "Reset
            //     Password", "Please reset your password by
            //     clicking <a href=\"" + callbackUrl + "\">here</a>

```

```

730         >");
       // return RedirectToAction("
       ForgotPasswordConfirmation", "Account");
731     }
732
733     // If we got this far, something failed, redisplay form
734     return View(model);
735 }
736
737 //
738 // GET: /Account/ForgotPasswordConfirmation
739 [AllowAnonymous]
740 public ActionResult ForgotPasswordConfirmation()
741 {
742     return View();
743 }
744
745 //
746 // GET: /Account/ResetPassword
747 [AllowAnonymous]
748 public ActionResult ResetPassword(string code)
749 {
750     if (code == null)
751     {
752         return View("Error");
753     }
754     return View();
755 }
756
757 //
758 // POST: /Account/ResetPassword
759 [HttpPost]
760 [AllowAnonymous]
761 [ValidateAntiForgeryToken]
762 public async Task<ActionResult> ResetPassword(
    ResetPasswordViewModel model)
763 {
764     if (ModelState.IsValid)
765     {
766         var user = await UserManager.FindByNameAsync(model.
            Email);
767         if (user == null)
768         {
769             ModelState.AddModelError("", "No user found.");
770             return View();
771         }
772         IdentityResult result = await UserManager.
            ResetPasswordAsync(user.Id, model.Code, model.
            Password);
773         if (result.Succeeded)
774         {
775             return RedirectToAction("

```

```

776         ResetPasswordConfirmation", "Account");
777     }
778     else
779     {
780         AddErrors(result);
781         return View();
782     }
783 }
784 // If we got this far, something failed, redisplay form
785 return View(model);
786 }
787
788 //
789 // GET: /Account/ResetPasswordConfirmation
790 [AllowAnonymous]
791 public ActionResult ResetPasswordConfirmation()
792 {
793     return View();
794 }
795
796 //
797 // POST: /Account/Disassociate
798 [HttpPost]
799 [ValidateAntiForgeryToken]
800 public async Task<ActionResult> Disassociate(string
    loginProvider, string providerKey)
801 {
802     ManageMessageId? message = null;
803     IdentityResult result = await UserManager.
        RemoveLoginAsync(User.Identity.GetUserId(), new
        UserLoginInfo(loginProvider, providerKey));
804     if (result.Succeeded)
805     {
806         var user = await UserManager.FindByIdAsync(User.
            Identity.GetUserId());
807         await SignInAsync(user, isPersistent: false);
808         message = ManageMessageId.RemoveLoginSuccess;
809     }
810     else
811     {
812         message = ManageMessageId.Error;
813     }
814     return RedirectToAction("Manage", new { Message =
        message });
815 }
816
817 //
818 // GET: /Account/Manage
819 public ActionResult Manage(ManageMessageId? message)
820 {
821     ViewBag.StatusMessage =

```

```

822         message == ManageMessaged.ChangePasswordSuccess ? "
            Your password has been changed."
823         : message == ManageMessaged.SetPasswordSuccess ? "
            Your password has been set."
824         : message == ManageMessaged.RemoveLoginSuccess ? "
            The external login was removed."
825         : message == ManageMessaged.Error ? "An error has
            occurred."
826         : "";
827     ViewBag.HasLocalPassword = HasPassword();
828     ViewBag.ReturnUrl = Url.Action("Manage");
829     return View();
830 }
831
832 //
833 // POST: /Account/Manage
834 [HttpPost]
835 [ValidateAntiForgeryToken]
836 public async Task<ActionResult> Manage(ManageUserViewModel
    model)
837 {
838     bool hasPassword = HasPassword();
839     ViewBag.HasLocalPassword = hasPassword;
840     ViewBag.ReturnUrl = Url.Action("Manage");
841     if (hasPassword)
842     {
843         if (ModelState.IsValid)
844         {
845             IdentityResult result = await UserManager.
                ChangePasswordAsync(User.Identity.GetUserId()
                    , model.OldPassword, model.NewPassword);
846             if (result.Succeeded)
847             {
848                 var user = await UserManager.FindByIdAsync(
                    User.Identity.GetUserId());
849                 await SignInAsync(user, isPersistent: false)
                    ;
850                 return RedirectToAction("Manage", new {
                    Message = ManageMessaged.
                        ChangePasswordSuccess });
851             }
852             else
853             {
854                 AddErrors(result);
855             }
856         }
857     }
858     else
859     {
860         // User does not have a password so remove any
            validation errors caused by a missing OldPassword
            field

```

```

861         ModelState state = ModelState["OldPassword"];
862         if (state != null)
863         {
864             state.Errors.Clear();
865         }
866
867         if (ModelState.IsValid)
868         {
869             IdentityResult result = await UserManager.
                AddPasswordAsync(User.Identity.GetUserId(),
                model.NewPassword);
870             if (result.Succeeded)
871             {
872                 return RedirectToAction("Manage", new {
                    Message = ManageMessageId.
                        SetPasswordSuccess });
873             }
874             else
875             {
876                 AddErrors(result);
877             }
878         }
879     }
880
881     // If we got this far, something failed, redisplay form
882     return View(model);
883 }
884
885 public ActionResult EditProfile()
886 {
887     ManageUserViewModel model = new ManageUserViewModel();
888     var user = UserManager.FindById(User.Identity.GetUserId(
        ));
889
890     model.Name = user.Name;
891     model.Surname = user.Surname;
892     model.MobilePhone = user.MobilePhone;
893     model.Email = user.Email;
894     model.Country = user.Country;
895     model.StreetAddress = user.StreetAddress;
896     model.OldPassword = string.Empty;
897     model.NewPassword = string.Empty;
898     return View(model);
899 }
900
901 [HttpPost]
902 [ValidateAntiForgeryToken]
903 public async Task<ActionResult> EditProfile(
    ManageUserViewModel model, HttpPostedFileBase image =
    null)
904 {
905     if (ModelState.IsValid)

```

```

906     {
907         var user = UserManager.FindById(User.Identity.
            GetUserId());

908
909         user.Name = model.Name;
910         user.Surname = model.Surname;
911         user.MobilePhone = model.MobilePhone;
912         user.Email = model.Email;
913         user.Country = model.Country;
914         user.StreetAddress = model.StreetAddress;
915         if (image != null)
916         {
917             user.ImageMimeType = image.ContentType;
918             user.ImageData = new byte[image.ContentLength];
919             image.InputStream.Read(user.ImageData, 0, image.
                ContentLength);
920         }
921
922         var result = await UserManager.UpdateAsync(user);
923
924         if (!result.Succeeded)
925         {
926             AddErrors(result);
927         }
928     }
929
930     return RedirectToAction("EditProfile");
931 }
932
933 //
934 // POST: /Account/ExternalLogin
935 [HttpPost]
936 [AllowAnonymous]
937 [ValidateAntiForgeryToken]
938 public ActionResult ExternalLogin(string provider, string
    returnUrl)
939 {
940     // Request a redirect to the external login provider
941     return new ChallengeResult(provider, Url.Action("
        ExternalLoginCallback", "Account", new { ReturnUrl =
            returnUrl }));
942 }
943
944 //
945 // GET: /Account/ExternalLoginCallback
946 [AllowAnonymous]
947 public async Task<ActionResult> ExternalLoginCallback(string
    returnUrl)
948 {
949     var loginInfo = await AuthenticationManager.
        GetExternalLoginInfoAsync();
950     if (loginInfo == null)

```

```

951     {
952         return RedirectToAction("Login");
953     }
954
955     // Sign in the user with this external login provider if
956     the user already has a login
957     var user = await UserManager.FindAsync(loginInfo.Login);
958     if (user != null)
959     {
960         user.Country = String.Empty;
961         user.MobilePhone = String.Empty;
962         user.Name = String.Empty;
963         user.Surname = String.Empty;
964         user.StreetAddress = String.Empty;
965         await SignInAsync(user, isPersistent: false);
966         return RedirectToLocal(returnUrl);
967     }
968     else
969     {
970         // If the user does not have an account, then prompt
971         the user to create an account
972         ViewBag.ReturnUrl = returnUrl;
973         ViewBag.LoginProvider = loginInfo.Login.
974             LoginProvider;
975         return View("ExternalLoginConfirmation", new
976             ExternalLoginConfirmationViewModel { Email =
977                 loginInfo.Email });
978     }
979 }
980
981 //
982 // POST: /Account/LinkLogin
983 [HttpPost]
984 [ValidateAntiForgeryToken]
985 public ActionResult LinkLogin(string provider)
986 {
987     // Request a redirect to the external login provider to
988     link a login for the current user
989     return new ChallengeResult(provider, Url.Action("
990         LinkLoginCallback", "Account"), User.Identity.
991         GetUserId());
992 }
993
994 //
995 // GET: /Account/LinkLoginCallback
996 public async Task<ActionResult> LinkLoginCallback()
997 {
998     var loginInfo = await AuthenticationManager.
999         GetExternalLoginInfoAsync(XsrfKey, User.Identity.
1000             GetUserId());
1001     if (loginInfo == null)
1002     {

```



```

993         return RedirectToAction("Manage", new { Message =
994             ManageMessaged.Error });
995     }
996     IdentityResult result = await UserManager.AddLoginAsync(
997         User.Identity.GetUserId(), loginInfo.Login);
998     if (result.Succeeded)
999     {
1000         return RedirectToAction("Manage");
1001     }
1002     return RedirectToAction("Manage", new { Message =
1003         ManageMessaged.Error });
1004 }
1005 //
1006 // POST: /Account/ExternalLoginConfirmation
1007 [HttpPost]
1008 [AllowAnonymous]
1009 [ValidateAntiForgeryToken]
1010 public async Task<ActionResult> ExternalLoginConfirmation(
1011     ExternalLoginConfirmationViewModel model, string
1012     returnUrl)
1013 {
1014     if (User.Identity.IsAuthenticated)
1015     {
1016         return RedirectToAction("Manage");
1017     }
1018     if (ModelState.IsValid)
1019     {
1020         // Get the information about the user from the
1021         // external login provider
1022         var info = await AuthenticationManager.
1023             GetExternalLoginInfoAsync();
1024         if (info == null)
1025         {
1026             return View("ExternalLoginFailure");
1027         }
1028         var user = new ApplicationUser() { Username = model.
1029             Email, Email = model.Email };
1030         IdentityResult result = await UserManager.
1031             CreateAsync(user);
1032         if (result.Succeeded)
1033         {
1034             result = await UserManager.AddLoginAsync(user.Id
1035                 , info.Login);
1036             if (result.Succeeded)
1037             {
1038                 await SignInAsync(user, isPersistent: false)
1039                     ;
1040             }
1041             // For more information on how to enable
1042             // account confirmation and password reset

```

```

1033         please visit http://go.microsoft.com/
1034         fwlink/?LinkID=320771
1035         // Send an email with this link
1036         // string code = await UserManager.
1037         GenerateEmailConfirmationTokenAsync(user.
1038         Id);
1039         // var callbackUrl = Url.Action("
1040         ConfirmEmail", "Account", new { userId =
1041         user.Id, code = code }, protocol: Request
1042         .Url.Scheme);
1043         // SendEmail(user.Email, callbackUrl, "
1044         Confirm your account", "Please confirm
1045         your account by clicking this link");
1046
1047         return RedirectToLocal(returnUrl);
1048     }
1049     }
1050     AddErrors(result);
1051 }
1052
1053 ViewBag.ReturnUrl = returnUrl;
1054 return View(model);
1055 }
1056
1057 //
1058 // POST: /Account/LogOff
1059 [HttpPost]
1060 [ValidateAntiForgeryToken]
1061 public ActionResult LogOff()
1062 {
1063     AuthenticationManager.SignOut();
1064     return RedirectToAction("List", "Ad");
1065 }
1066
1067 //
1068 // GET: /Account/ExternalLoginFailure
1069 [AllowAnonymous]
1070 public ActionResult ExternalLoginFailure()
1071 {
1072     return View();
1073 }
1074
1075 [ChildActionOnly]
1076 public ActionResult RemoveAccountList()
1077 {
1078     var linkedAccounts = UserManager.GetLogins(User.Identity
1079     .GetUserId());
1080     ViewBag.ShowRemoveButton = HasPassword() ||
1081     linkedAccounts.Count > 1;
1082     return (ActionResult) PartialView("_RemoveAccountPartial",
1083     linkedAccounts);
1084 }

```

```

1073
1074     protected override void Dispose(bool disposing)
1075     {
1076         if (disposing && UserManager != null)
1077         {
1078             UserManager.Dispose();
1079             UserManager = null;
1080         }
1081         base.Dispose(disposing);
1082     }
1083
1084     #region Helpers
1085     // Used for XSRF protection when adding external logins
1086     private const string XsrfKey = "XsrfId";
1087
1088     private IAuthenticationManager AuthenticationManager
1089     {
1090         get
1091         {
1092             return HttpContext.GetOwinContext().Authentication;
1093         }
1094     }
1095
1096     private async Task SignInAsync(ApplicationUser user, bool
1097         isPersistent)
1098     {
1099         AuthenticationManager.SignOut(DefaultAuthenticationTypes
1100             .ExternalCookie);
1101         AuthenticationManager.SignIn(new
1102             AuthenticationProperties() { IsPersistent =
1103                 isPersistent }, await user.GenerateUserIdentityAsync(
1104                 UserManager));
1105     }
1106
1107     private void AddErrors(IdentityResult result)
1108     {
1109         foreach (var error in result.Errors)
1110         {
1111             ModelState.AddModelError("", error);
1112         }
1113     }
1114
1115     private bool HasPassword()
1116     {
1117         var user = UserManager.FindById(User.Identity.GetUserId
1118             ());
1119         if (user != null)
1120         {
1121             return user.PasswordHash != null;
1122         }
1123         return false;
1124     }
1125

```

```

1119
1120     private void SendEmail(string email, string callbackUrl,
1121                             string subject, string message)
1122     {
1123         // For information on sending mail, please visit http://
1124         go.microsoft.com/fwlink/?LinkID=320771
1125     }
1126
1127     public enum ManageMessageld
1128     {
1129         ChangePasswordSuccess,
1130         SetPasswordSuccess,
1131         RemoveLoginSuccess,
1132         Error
1133     }
1134
1135     private ActionResult RedirectToLocal(string returnUrl)
1136     {
1137         if (Url.IsLocalUrl(returnUrl))
1138         {
1139             return Redirect(returnUrl);
1140         }
1141         else
1142         {
1143             return RedirectToAction("UserAds", "User");
1144         }
1145     }
1146
1147     private class ChallengeResult : HttpUnauthorizedResult
1148     {
1149         public ChallengeResult(string provider, string
1150                                 redirectUri) : this(provider, redirectUri, null)
1151         {
1152         }
1153
1154         public ChallengeResult(string provider, string
1155                                 redirectUri, string userId)
1156         {
1157             LoginProvider = provider;
1158             RedirectUri = redirectUri;
1159             UserId = userId;
1160         }
1161
1162         public string LoginProvider { get; set; }
1163         public string RedirectUri { get; set; }
1164         public string UserId { get; set; }
1165
1166         public override void ExecuteResult(ControllerContext
1167                                             context)
1168         {
1169             var properties = new AuthenticationProperties() {
1170                 RedirectUri = RedirectUri };

```

```

1165         if (UserId != null)
1166         {
1167             properties.Dictionary[XsrfKey] = UserId;
1168         }
1169         context.HttpContext.GetOwinContext().Authentication.
            Challenge(properties, LoginProvider);
1170     }
1171 }
1172 #endregion
1173
1174 }
1175 }
1176
1177 using AdBoard.Domain.Abstract;
1178 using AdBoard.Domain.Concrete;
1179 using AdBoard.Domain.Entities;
1180 using AdBoard.WebUI.Models;
1181 using System;
1182 using System.Collections.Generic;
1183 using System.IO;
1184 using System.Linq;
1185 using System.Web;
1186 using System.Web.Mvc;
1187 using Microsoft.AspNet.Identity;
1188
1189
1190 namespace AdBoard.WebUI.Controllers
1191 {
1192     public class AdController : Controller
1193     {
1194         IAdRepository repository;
1195         public int pageSize = 2;
1196         protected ApplicationDbContext ApplicationDbContext { get;
            set; }
1197
1198         public AdController(IAdRepository repo)
1199         {
1200             this.ApplicationDbContext = new ApplicationDbContext();
1201             repository = repo;
1202         }
1203
1204         public ActionResult Edit(int id)
1205         {
1206             Ad ad = repository.Ads
1207                 .FirstOrDefault(a => a.Id == id);
1208             return View(ad);
1209         }
1210
1211         [HttpPost]
1212         public ActionResult Edit(Ad ad)
1213         {
1214             if (ModelState.IsValid)

```

```

1215         {
1216             repository.SaveAd(ad);
1217             TempData["message"] = string.Format("Changing in ad
1218             \"{0}\" was saved", ad.Name);
1219             return RedirectToAction("UserAds", "User");
1220         }
1221         else
1222         {
1223             return View(ad);
1224         }
1225     }
1226     [HttpPost]
1227     public ActionResult Delete(int add)
1228     {
1229         Ad deletedAd = repository.DeleteAd(add);
1230         if (deletedAd != null)
1231         {
1232             TempData["message"] = string.Format("Ad \"{0}\" was
1233             deleted",
1234             deletedAd.Name);
1235         }
1236         return RedirectToAction("UserAds", "User");
1237     }
1238     public ViewResult Create()
1239     {
1240         Ad ad = new Ad();
1241         ad.UserId = User.Identity.GetUserId();
1242         return View("Edit", ad);
1243     }
1244
1245     public ViewResult List(string category, int page = 1)
1246     {
1247         AdListViewModel model = new AdListViewModel
1248         {
1249             Ads = repository.Ads
1250                 .Where(p => category == null || p.Category ==
1251                 category)
1252                 .OrderBy(ads => ads.Id)
1253                 .Skip((page - 1) * pageSize)
1254                 .Take(pageSize).ToList(),
1255             PagingInfo = new PagingInfo
1256             {
1257                 CurrentPage = page,
1258                 ItemsPerPage = pageSize,
1259                 TotalItems = category == null ?
1260                 repository.Ads.Count() :
1261                 repository.Ads.Where(a => a.Category == category)
1262                 .Count(),
1263             },
1264             CurrentCategory = category

```

```

1263     };
1264     foreach (Ad ad in model.Ads)
1265     {
1266         ad.Images = repository.Images
1267             .Where(i => i.AddId == ad.Id);
1268     }
1269     ViewBag.IsInfo = false;
1270     ViewBag.IsUserAd = false;
1271     return View(model);
1272 }
1273
1274 public ActionResult AdInfo(int id)
1275 {
1276     var ad = repository.Ads.Where(a => a.Id == id).
1277         FirstOrDefault();
1278     ad.Images = repository.Images.Where(m => m.AddId == id);
1279     UserAdViewModel model = new UserAdViewModel
1280     {
1281         Ad = ad,
1282         User = ApplicationDbContext.Users.FirstOrDefault(u
1283             => u.Id == ad.UserId);
1284     };
1285     ViewBag.IsInfo = true;
1286     ViewBag.IsUserAd = false;
1287     if (User.Identity.IsAuthenticated)
1288     {
1289         if (model.Ad.UserId == User.Identity.GetUserId())
1290             ViewBag.IsUserAd = true;
1291         else
1292             ViewBag.IsUserAd = false;
1293     }
1294     return View(model);
1295 }
1296
1297 public ActionResult SearchAds(string adName, string category
1298 )
1299 {
1300     ViewBag.Category = new SelectList(repository.Ads, "
1301         Category", "Category");
1302     ViewBag.IsInfo = false;
1303     ViewBag.IsUserAd = false;
1304     var ads = from a in repository.Ads
1305         select a;
1306     if (!String.IsNullOrEmpty(adName))
1307     {
1308         ads = ads.Where(a => a.Name.ToLower().Contains(
1309             adName.ToLower()));
1310     }
1311     if (!String.IsNullOrEmpty(category))
1312     {
1313         return View(ads.Where(x => x.Category == category));
1314     }

```

```

1310         else
1311             return View(ads);
1312     }
1313
1314     public FileContentResult GetAdImage(int imageId)
1315     {
1316         var Image = repository.Images.Where(i => i.Id == imageId)
1317             .FirstOrDefault();
1318         if (Image != null)
1319         {
1320             return File(Image.ImageData, Image.ImageMimeType);
1321         }
1322         else
1323         {
1324             return null;
1325             /*string path = AppDomain.CurrentDomain.
1326                BaseDirectory + "/Content/Images/No_image.png";
1327             return File(System.IO.File.ReadAllBytes(path), "
1328                image/jpg");*/
1329         }
1330     }
1331
1332     public ActionResult EditImages(int addId)
1333     {
1334         var images = repository.Images.Where(i => i.AddId == addId)
1335             );
1336         ViewBag.AddId = addId;
1337         return View(images);
1338     }
1339
1340     [HttpPost]
1341     [ValidateAntiForgeryToken]
1342     public ActionResult EditImages(int addId, HttpPostedFileBase
1343         file = null)
1344     {
1345         Image image = new Image();
1346         if (ModelState.IsValid)
1347         {
1348             if (file != null)
1349             {
1350                 image.AddId = addId;
1351                 image.ImageMimeType = file.ContentType;
1352                 image.ImageData = new byte[file.ContentLength];
1353                 file.InputStream.Read(image.ImageData, 0, file.
1354                     ContentLength);
1355                 repository.SaveImage(image);
1356             }
1357         }
1358         else
1359         {

```



```

1356         return null;
1357     }
1358     return RedirectToAction("AdInfo", new { id = image.AddId
1359         });
1360 }
1361 }
1362
1363 using AdBoard.Domain.Abstract;
1364 using AdBoard.Domain.Entities;
1365 using System;
1366 using System.Collections.Generic;
1367 using System.Linq;
1368 using System.Web;
1369 using System.Web.Mvc;
1370
1371 namespace AdBoard.WebUI.Controllers
1372 {
1373     public class AdminController : Controller
1374     {
1375         IAdRepository repository;
1376
1377         public AdminController(IAdRepository repo)
1378         {
1379             repository = repo;
1380         }
1381
1382         public ActionResult Index()
1383         {
1384             return View(repository.Ads);
1385         }
1386
1387         public ActionResult Edit(int addId)
1388         {
1389             Ad ad = repository.Ads.FirstOrDefault(a => a.Id == addId)
1390                 ;
1391             return View(ad);
1392         }
1393
1394         [HttpPost]
1395         public ActionResult Edit(Ad ad)
1396         {
1397             if (ModelState.IsValid)
1398             {
1399                 repository.SaveAd(ad);
1400                 TempData["message"] = string.Format("Changes in ad
1401                     {0} was saved",
1402                         ad.Name);
1403                 return RedirectToAction("Index");
1404             }
1405             else
1406             {

```

```

1405         return View(ad);
1406     }
1407 }
1408
1409 [HttpPost]
1410 public ActionResult Delete(int add)
1411 {
1412     Ad deletedAd = repository.DeleteAd(add);
1413     if (deletedAd != null)
1414     {
1415         TempData["message"] = string.Format("Ad {0} was
1416             deleted",
1417             deletedAd.Name);
1418     }
1419     return RedirectToAction("Index");
1420 }
1421 }
1422
1423 using AdBoard.WebUI.Models;
1424 using System;
1425 using System.Collections.Generic;
1426 using System.Linq;
1427 using System.Web;
1428 using System.Web.Mvc;
1429
1430 namespace AdBoard.WebUI.Controllers
1431 {
1432     public class ImageController : Controller
1433     {
1434         private ApplicationDbContext contextIdentity = new
1435             ApplicationDbContext();
1436
1437         public ImageController()
1438         {
1439         }
1440
1441         public FileContentResult GetImage(string id)
1442         {
1443             ApplicationUser user = contextIdentity.Users.
1444                 FirstOrDefault(u => u.Id == id);
1445             if (user != null)
1446             {
1447                 if (user.ImageData != null)
1448                     return File(user.ImageData, user.ImageMimeType);
1449                 else
1450                 {
1451                     string path = AppDomain.CurrentDomain.
1452                         BaseDirectory + "/Content/Images/User.png";
1453                     return File(System.IO.File.ReadAllBytes(path), "
1454                         image/jpeg");
1455                 }
1456             }
1457         }
1458     }
1459 }

```

```

1452         }
1453     }
1454     else
1455         return null;
1456 }
1457
1458 protected override void Dispose(bool disposing)
1459 {
1460     contextIdentity.Dispose();
1461     base.Dispose(disposing);
1462 }
1463 }
1464 }
1465
1466 using AdBoard.Domain.Abstract;
1467 using AdBoard.Domain.Concrete;
1468 using System;
1469 using System.Collections.Generic;
1470 using System.Linq;
1471 using System.Web;
1472 using System.Web.Mvc;
1473
1474 namespace AdBoard.WebUI.Controllers
1475 {
1476     public class NavController : Controller
1477     {
1478         private IAdRepository repository;
1479
1480         public NavController(IAdRepository repo)
1481         {
1482             repository = repo;
1483         }
1484
1485         public PartialViewResult Menu(string category = null)
1486         {
1487             ViewBag.SelectedCategory = category;
1488
1489             IEnumerable<string> categories = repository.Ads
1490                 .Select(a => a.Category)
1491                 .Distinct()
1492                 .OrderBy(x => x);
1493
1494             return PartialView("FlexMenu", categories);
1495         }
1496     }
1497 }
1498
1499 using AdBoard.Domain.Abstract;
1500 using AdBoard.Domain.Entities;
1501 using AdBoard.WebUI.Models;
1502 using System;
1503 using System.Collections.Generic;

```

```

1504 using System.Linq;
1505 using System.Web;
1506 using System.Web.Mvc;
1507 using System.Web.Security;
1508 using Microsoft.AspNet.Identity;
1509 using Microsoft.AspNet.Identity.EntityFramework;
1510 using AdBoard.Domain.Concrete;
1511
1512 namespace AdBoard.WebUI.Controllers
1513 {
1514     public class UserController : Controller
1515     {
1516         IAdRepository repository;
1517         public int pageSize = 1;
1518         protected ApplicationDbContext ApplicationDbContext { get;
            set; }
1519
1520         public UserController(IAdRepository repo)
1521         {
1522             this.ApplicationDbContext = new ApplicationDbContext();
1523             repository = repo;
1524         }
1525
1526         public ActionResult UserAds(int page = 1)
1527         {
1528             var userId = User.Identity.GetUserId();
1529
1530             UserAdsViewModel model = new UserAdsViewModel
1531             {
1532                 Ads = repository.Ads
1533                     .Where(a => a.UserId == userId)
1534                     .OrderBy(a => a.Name)
1535                     .Skip((page - 1) * pageSize)
1536                     .Take(pageSize)
1537                     .ToList(),
1538                 PagingInfo = new PagingInfo
1539                 {
1540                     CurrentPage = page,
1541                     ItemsPerPage = pageSize,
1542                     TotalItems = repository.Ads
1543                         .Where(a => a.UserId == userId)
1544                         .Count()
1545                 },
1546                 User = ApplicationDbContext.Users.FirstOrDefault(x
                    => x.Id == userId)
1547             };
1548             foreach (Ad ad in model.Ads)
1549             {
1550                 ad.Images = repository.Images
1551                     .Where(i => i.AdId == ad.Id);
1552             }
1553             ViewBag.IsInfo = true;

```

```

1554         ViewBag.IsUserAd = true;
1555         return View(model);
1556     }
1557
1558     public PartialViewResult Profile()
1559     {
1560         var userId = User.Identity.GetUserId();
1561         var user = ApplicationDbContext.Users.FirstOrDefault(x
            => x.Id == userId);
1562         return PartialView(user);
1563     }
1564 }
1565 }
1566
1567 using System.ComponentModel.DataAnnotations;
1568
1569 namespace AdBoard.WebUI.Models
1570 {
1571     public class ExternalLoginConfirmationViewModel
1572     {
1573         [Required]
1574         [EmailAddress]
1575         [Display(Name = "Email")]
1576         public string Email { get; set; }
1577     }
1578
1579     public class ExternalLoginListViewModel
1580     {
1581         public string Action { get; set; }
1582         public string ReturnUrl { get; set; }
1583     }
1584
1585     public class ManageUserViewModel
1586     {
1587         [Required]
1588         [EmailAddress]
1589         [Display(Name="Email")]
1590         public string Email { get; set; }
1591
1592         [Required]
1593         [Display(Name = "Name")]
1594         public string Name { get; set; }
1595
1596         [Required]
1597         [Display(Name = "Surname")]
1598         public string Surname { get; set; }
1599
1600         [Required]
1601         [Display(Name = "Country")]
1602         public string Country { get; set; }
1603
1604         [Required]

```

```

1605     [Display(Name = "Street Address")]
1606     public string StreetAddress { get; set; }
1607
1608     [Required]
1609     [Display(Name = "MobilePhone")]
1610     public string MobilePhone { get; set; }
1611
1612     [Required]
1613     [DataType(DataType.Password)]
1614     [Display(Name = "Current password")]
1615     public string OldPassword { get; set; }
1616
1617     [Required]
1618     [StringLength(100, ErrorMessage = "The {0} must be at least
1619         {2} characters long.", MinimumLength = 6)]
1619     [DataType(DataType.Password)]
1620     [Display(Name = "New password")]
1621     public string NewPassword { get; set; }
1622
1623     [DataType(DataType.Password)]
1624     [Display(Name = "Confirm new password")]
1625     [Compare("NewPassword", ErrorMessage = "The new password and
1626         confirmation password do not match.")]
1626     public string ConfirmPassword { get; set; }
1627 }
1628
1629 public class LoginViewModel
1630 {
1631     [Required]
1632     [EmailAddress]
1633     [Display(Name = "Email")]
1634     public string Email { get; set; }
1635
1636     [Required]
1637     [DataType(DataType.Password)]
1638     [Display(Name = "Password")]
1639     public string Password { get; set; }
1640
1641     [Display(Name = "Remember me?")]
1642     public bool RememberMe { get; set; }
1643 }
1644
1645 public class RegisterViewModel
1646 {
1647     [Required]
1648     [EmailAddress]
1649     [Display(Name = "Email*")]
1650     public string Email { get; set; }
1651
1652     [Required]
1653     [Display(Name = "Name*")]
1654     public string Name { get; set; }

```

```

1655
1656     [Required]
1657     [Display(Name = "Surname*")]
1658     public string Surname { get; set; }
1659
1660     [Required]
1661     [Display(Name = "Country*")]
1662     public string Country { get; set; }
1663
1664     [Required]
1665     [Display(Name="Street Address*")]
1666     public string StreetAddress { get; set; }
1667
1668     [Required]
1669     [Display(Name = "MobilePhone*")]
1670     public string MobilePhone { get; set; }
1671
1672     public int[] FavoritesAds { get; set; }
1673
1674     [Display(Name = "Image")]
1675     public byte[] ImageData { get; set; }
1676
1677     public string ImageMimeType { get; set; }
1678
1679
1680     [Required]
1681     [StringLength(100, ErrorMessage = "The {0} must be at least
1682         {2} characters long.", MinimumLength = 6)]
1683     [DataType(DataType.Password)]
1684     [Display(Name = "Password*")]
1685     public string Password { get; set; }
1686
1687     [DataType(DataType.Password)]
1688     [Display(Name = "Confirm password*")]
1689     [Compare("Password", ErrorMessage = "The password and
1690         confirmation password do not match.")]
1691     public string ConfirmPassword { get; set; }
1692 }
1693
1694 public class ResetPasswordViewModel
1695 {
1696     [Required]
1697     [EmailAddress]
1698     [Display(Name = "Email")]
1699     public string Email { get; set; }
1700
1701     [Required]
1702     [StringLength(100, ErrorMessage = "The {0} must be at least
1703         {2} characters long.", MinimumLength = 6)]
1704     [DataType(DataType.Password)]
1705     [Display(Name = "Password")]
1706     public string Password { get; set; }

```

```

1704         [DataType(DataType.Password)]
1705         [Display(Name = "Confirm password")]
1706         [Compare("Password", ErrorMessage = "The password and
1707             confirmation password do not match.")]
1708         public string ConfirmPassword { get; set; }
1709
1710         public string Code { get; set; }
1711     }
1712
1713     public class ForgotPasswordViewModel
1714     {
1715         [Required]
1716         [EmailAddress]
1717         [Display(Name = "Email")]
1718         public string Email { get; set; }
1719     }
1720 }
1721
1722 using System;
1723 using System.Collections.Generic;
1724 using System.Linq;
1725 using System.Web;
1726
1727 namespace AdBoard.WebUI.Models
1728 {
1729     public class PagingInfo
1730     {
1731         public int TotalItems { get; set; }
1732         public int ItemsPerPage { get; set; }
1733         public int CurrentPage { get; set; }
1734         public int TotalPages
1735         {
1736             get { return (int)Math.Ceiling((decimal)TotalItems /
1737                 ItemsPerPage); }
1738         }
1739     }

```



## ПРИЛОЖЕНИЕ А Исходный код тестов

```
1  using System;
2  using Microsoft.VisualStudio.TestTools.UnitTesting;
3  using Moq;
4  using AdBoard.Domain.Abstract;
5  using AdBoard.Domain.Entities;
6  using System.Collections.Generic;
7  using AdBoard.WebUI.Controllers;
8  using System.Linq;
9  using System.Web.Mvc;
10 using AdBoard.WebUI.Models;
11 using AdBoard.WebUI.HtmlHelpers;
12 using Microsoft.AspNet.Identity;
13 using AdBoard.Domain.Concrete;
14 using System.Security.Principal;
15 using System.Threading;
16 using System.Web;
17 using System.Web.Routing;
18 using System.Security.Claims;
19
20 namespace AdBoard.UnitTests
21 {
22     [TestClass]
23     public class UnitTest1
24     {
25         [TestMethod]
26         public void CanPaginate()
27         {
28             //arrange
29             Mock<IAdRepository> mock = new Mock<IAdRepository>();
30             mock.Setup(m => m.Ads).Returns(new List<Ad>
31             {
32                 new Ad { Id = 1, Name = "A1" },
33                 new Ad { Id = 2, Name = "A2" },
34                 new Ad { Id = 3, Name = "A3" },
35                 new Ad { Id = 4, Name = "A4" },
36                 new Ad { Id = 5, Name = "A5" }
37             });
38             AdController controller = new AdController(mock.Object);
39             controller.pageSize = 3;
40
41             //act
42             AdListViewModel result = (AdListViewModel)controller.
43                 List(null, 2).Model;
44
45             //assert
46             List<Ad> ads = result.Ads.ToList();
47             Assert.IsTrue(ads.Count == 2);
```

```

47         Assert.AreEqual(ads[0].Name, "A4");
48         Assert.AreEqual(ads[1].Name, "A5");
49     }
50
51     [TestMethod]
52     public void Can_Generate_Page_Links()
53     {
54         //arrange
55         HtmlHelper helper = null;
56
57         PagingInfo pagingInfo = new PagingInfo
58         {
59             CurrentPage = 2,
60             ItemsPerPage = 10,
61             TotalItems = 28
62         };
63
64         Func<int, string> pageUrlDelegate = i => "Page" + i;
65
66         //act
67         MvcHtmlString result = helper.PageLinks(pagingInfo,
68             pageUrlDelegate);
69
70         //asset
71         Assert.AreEqual(result.ToString(), @"<a class=""btn btn-
72             default"" href=""Page1"">1</a>"
73             + @"<a class=""btn btn-default btn-primary selected"
74             " href=""Page2"">2</a>"
75             + @"<a class=""btn btn-default"" href=""Page3"">3</a
76             >"");
77     }
78
79     [TestMethod]
80     public void Can_Send_Pagination_View_Model()
81     {
82         //arrange
83         Mock<IAdRepository> mock = new Mock<IAdRepository>();
84         mock.Setup(m => m.Ads).Returns(new List<Ad>
85         {
86             new Ad { Id = 1, Name = "A1"},
87             new Ad { Id = 2, Name = "A2"},
88             new Ad { Id = 3, Name = "A3"},
89             new Ad { Id = 4, Name = "A4"},
90             new Ad { Id = 5, Name = "A5"}
91         });
92         AdController controller = new AdController(mock.Object);
93         controller.pageSize = 3;
94
95         //act
96         AdListViewModel result = (AdListViewModel)controller.
97             List(null, 2).Model;
98     }

```

```

94         //assert
95         PagingInfo pagingInfo = result.PagingInfo;
96         Assert.AreEqual(pagingInfo.CurrentPage, 2);
97         Assert.AreEqual(pagingInfo.TotalItems, 5);
98         Assert.AreEqual(pagingInfo.TotalPages, 2);
99         Assert.AreEqual(pagingInfo.ItemsPerPage, 3);
100     }
101
102     [TestMethod]
103     public void Can_Filter_Ads()
104     {
105         //arrange
106         Mock<IAdRepository> mock = new Mock<IAdRepository>();
107         mock.Setup(m => m.Ads).Returns(new List<Ad>
108         {
109             new Ad { Id = 1, Name = "Ad1", Category = "C1"},
110             new Ad { Id = 2, Name = "Ad2", Category = "C2"},
111             new Ad { Id = 3, Name = "Ad3", Category = "C1"},
112             new Ad { Id = 4, Name = "Ad4", Category = "C2"},
113             new Ad { Id = 5, Name = "Ad5", Category = "C1"}
114         });
115         AdController controller = new AdController(mock.Object);
116         controller.pageSize = 3;
117
118         //act
119         List<Ad> result = ((AdListViewModel)controller.List("C2",
120             , 1).Model).Ads.ToList();
121
122         //assert
123         Assert.AreEqual(result.Count, 2);
124         Assert.IsTrue(result[0].Name == "Ad2" && result[0].
125             Category == "C2");
126         Assert.IsTrue(result[1].Name == "Ad4" && result[1].
127             Category == "C2");
128     }
129
130     [TestMethod]
131     public void Can_Generate_List_of_Categories()
132     {
133         Mock<IAdRepository> mock = new Mock<IAdRepository>();
134         mock.Setup(m => m.Ads).Returns(new List<Ad>
135         {
136             new Ad { Id = 1, Category = "C1"},
137             new Ad { Id = 2, Category = "C3"},
138             new Ad { Id = 3, Category = "C1"},
139             new Ad { Id = 4, Category = "C2"},
140             new Ad { Id = 5, Category = "C3"},
141         });
142         NavController controller = new NavController(mock.Object);
143
144         //act

```

```

142         List<string> result = ((IEnumerable<string>)controller.
143             Menu(null).Model).ToList();
144
145         //assert
146         Assert.IsTrue(result.Count == 3);
147         Assert.AreEqual(result[0], "C1");
148         Assert.AreEqual(result[1], "C2");
149         Assert.AreEqual(result[2], "C3");
150     }
151
152     [TestMethod]
153     public void Indicates_Selected_Category()
154     {
155         //arrange
156         Mock<IAdRepository> mock = new Mock<IAdRepository>();
157         mock.Setup(m => m.Ads).Returns(new List<Ad>
158         {
159             new Ad { Id = 1, Category = "C1"},
160             new Ad { Id = 2, Category = "C2"},
161             new Ad { Id = 3, Category = "C1"}
162         });
163         NavController controller = new NavController(mock.Object);
164
165         string categoryToSelect = "C1";
166
167         //act
168         string result = controller.Menu(categoryToSelect).
169             ViewBag.SelectedCategory;
170
171         //assert
172         Assert.AreEqual(categoryToSelect, result);
173     }
174
175     [TestMethod]
176     public void Generate_Category_Specific_Count()
177     {
178         //arrange
179         Mock<IAdRepository> mock = new Mock<IAdRepository>();
180         mock.Setup(m => m.Ads).Returns(new List<Ad>
181         {
182             new Ad { Id = 1, Category = "C1"},
183             new Ad { Id = 2, Category = "C2"},
184             new Ad { Id = 3, Category = "C1"},
185             new Ad { Id = 4, Category = "C2"},
186             new Ad { Id = 5, Category = "C3"}
187         });
188         AdController controller = new AdController(mock.Object);
189         controller.pageSize = 3;
190
191         //act
192         int res1 = ((AdListViewModel)controller.List("C1").Model

```

```

191         ).PagingInfo.TotalItems;
192         int res2 = ((AdListViewModel) controller.List("C2").Model
193             ).PagingInfo.TotalItems;
194         int res3 = ((AdListViewModel) controller.List("C3").Model
195             ).PagingInfo.TotalItems;
196         int res4 = ((AdListViewModel) controller.List(null).Model
197             ).PagingInfo.TotalItems;
198
199         //assert
200         Assert.AreEqual(res1, 2);
201         Assert.AreEqual(res2, 2);
202         Assert.AreEqual(res3, 1);
203         Assert.AreEqual(res4, 5);
204     }
205
206     [TestMethod]
207     public void Can_Edit_Ad()
208     {
209         Mock<IAdRepository> mock = new Mock<IAdRepository>();
210         mock.Setup(m => m.Ads).Returns(new List<Ad>
211             {
212                 new Ad { Id = 1, Name = "A1"},
213                 new Ad { Id = 2, Name = "A2"},
214                 new Ad { Id = 3, Name = "A3"},
215             });
216
217         AdController controller = new AdController(mock.Object);
218
219         Ad ad1 = controller.Edit(1).ViewData.Model as Ad;
220         Ad ad2 = controller.Edit(2).ViewData.Model as Ad;
221         Ad ad3 = controller.Edit(3).ViewData.Model as Ad;
222
223         Assert.AreEqual(1, ad1.Id);
224         Assert.AreEqual(2, ad2.Id);
225         Assert.AreEqual(3, ad3.Id);
226     }
227
228     [TestMethod]
229     public void Can_Create_Ad()
230     {
231         var context = new Mock<HttpContextBase>();
232         var mockIdentity = new Mock<IIdentity>();
233         context.SetupGet(x => x.User.Identity).Returns(
234             mockIdentity.Object);
235         mockIdentity.Setup(x => x.Name).Returns("test_name");
236         Mock<IAdRepository> mock = new Mock<IAdRepository>();
237         mock.Setup(m => m.Ads).Returns(new List<Ad>
238             {
239                 new Ad { Id = 0, Name="A1", UserId = "id"}
240             });

```

```

238     var identity = new GenericIdentity("dominik.ernst@xyz123
239         .de");
240     identity.AddClaim(new Claim("http://schemas.xmlsoap.org/
241         ws/2005/05/identity/claims/nameidentifier", "id"));
242     var principal = new GenericPrincipal(identity, new[] { "
243         user" });
244     context.Setup(s => s.User).Returns(principal);
245
246     AdController controller = new AdController(mock.Object);
247     controller.ControllerContext = new ControllerContext(
248         context.Object, new RouteData(), controller);
249
250     var result = controller.Create().Model as Ad;
251
252     Assert.AreEqual(result.UserId, "id");
253 }
254
255 [TestMethod]
256 public void Can_Delete_Ad()
257 {
258     Ad ad = new Ad { Id = 1, Name = "A2" };
259
260     Mock<IAdRepository> mock = new Mock<IAdRepository>();
261     mock.Setup(m => m.Ads).Returns(new List<Ad>
262     {
263         new Ad { Id = 0, Name="A1"},
264         new Ad { Id = 1, Name = "A2"},
265         new Ad { Id = 2, Name = "A3"}
266     });
267
268     AdController controller = new AdController(mock.Object);
269
270     controller.Delete(1);
271
272     mock.Verify(m => m.DeleteAd(ad.Id));
273 }
274
275 [TestMethod]
276 public void Cannot_Edit_Nonexistent_Game()
277 {
278     Mock<IAdRepository> mock = new Mock<IAdRepository>();
279     mock.Setup(m => m.Ads).Returns(new List<Ad>
280     {
281         new Ad { Id = 1, Name = "Ad1"},
282         new Ad { Id = 2, Name = "Ad2"},
283         new Ad { Id = 3, Name = "Ad3"},
284         new Ad { Id = 4, Name = "Ad4"},
285         new Ad { Id = 5, Name = "Ad5"}
286     });
287
288     AdController controller = new AdController(mock.Object);

```

```

286         Ad result = controller.Edit(6).ViewData.Model as Ad;
287
288         // Assert
289     }
290 }
291 }
292
293 using AdBoard.Domain.Abstract;
294 using AdBoard.Domain.Entities;
295 using AdBoard.WebUI.Controllers;
296 using Microsoft.VisualStudio.TestTools.UnitTesting;
297 using Moq;
298 using System;
299 using System.Collections.Generic;
300 using System.Linq;
301 using System.Text;
302 using System.Threading.Tasks;
303 using System.Web.Mvc;
304
305 namespace AdBoard.UnitTests
306 {
307     [TestClass]
308     public class AdminTests
309     {
310         [TestMethod]
311         public void Can_Edit_Admin_Ad()
312         {
313             Mock<IAdRepository> mock = new Mock<IAdRepository>();
314             mock.Setup(m => m.Ads).Returns(new List<Ad>
315             {
316                 new Ad { Id = 1, Name = "A1"},
317                 new Ad { Id = 2, Name = "A2"},
318                 new Ad { Id = 3, Name = "A3"},
319             });
320
321             AdminController controller = new AdminController(mock.Object);
322
323             Ad ad1 = controller.Edit(1).ViewData.Model as Ad;
324             Ad ad2 = controller.Edit(2).ViewData.Model as Ad;
325             Ad ad3 = controller.Edit(3).ViewData.Model as Ad;
326
327             Assert.AreEqual(1, ad1.Id);
328             Assert.AreEqual(2, ad2.Id);
329             Assert.AreEqual(3, ad3.Id);
330         }
331
332         [TestMethod]
333         public void Cannot_Edit_Nonexistent_Ad()
334         {
335             Mock<IAdRepository> mock = new Mock<IAdRepository>();
336             mock.Setup(m => m.Ads).Returns(new List<Ad>

```

```

337         {
338             new Ad { Id = 1, Name = "Ad1"},
339             new Ad { Id = 2, Name = "Ad2"},
340             new Ad { Id = 3, Name = "Ad3"},
341             new Ad { Id = 4, Name = "Ad4"},
342             new Ad { Id = 5, Name = "Ad5"}
343         });
344
345         AdminController controller = new AdminController(mock.
            Object);
346
347         Ad ad = controller.Edit(6).ViewData.Model as Ad;
348     }
349
350     [TestMethod]
351     public void Index_Contains_All_Ads()
352     {
353         Mock<IAdRepository> mock = new Mock<IAdRepository>();
354         mock.Setup(m => m.Ads).Returns(new List<Ad>
355         {
356             new Ad { Id = 1, Name = "Ad1"},
357             new Ad { Id = 2, Name = "Ad2"},
358             new Ad { Id = 3, Name = "Ad3"},
359             new Ad { Id = 4, Name = "Ad4"},
360             new Ad { Id = 5, Name = "Ad5"}
361         });
362
363         AdminController controller = new AdminController(mock.
            Object);
364
365         List<Ad> ads = ((IEnumerable<Ad>)controller.Index().
            ViewData.Model).ToList();
366
367         Assert.AreEqual(ads.Count, 5);
368         Assert.AreEqual(ads[0].Name, "Ad1");
369         Assert.AreEqual(ads[4].Name, "Ad5");
370     }
371
372     [TestMethod]
373     public void Can_Save_Valid_Changes()
374     {
375         Mock<IAdRepository> mock = new Mock<IAdRepository>();
376
377         AdminController controller = new AdminController(mock.
            Object);
378
379         Ad ad = new Ad { Name = "Test" };
380
381         ActionResult result = controller.Edit(ad);
382
383         mock.Verify(m => m.SaveAd(ad));
384

```



```

385         Assert.IsNotInstanceOfType(result, typeof(ViewResult));
386     }
387
388     [TestMethod]
389     public void Can_Save_Invalid_Changes()
390     {
391         Mock<IAdRepository> mock = new Mock<IAdRepository>();
392
393         AdminController controller = new AdminController(mock.Object);
394
395         Ad ad = new Ad { Name = "Test" };
396
397         controller.ModelState.AddModelError("error", "error");
398
399         ActionResult result = controller.Edit(ad);
400
401         mock.Verify(m => m.SaveAd(It.IsAny<Ad>()), Times.Never());
402
403         Assert.IsInstanceOfType(result, typeof(ViewResult));
404     }
405
406     [TestMethod]
407     public void Can_Delete_Valid_Ads()
408     {
409         Ad ad = new Ad { Id = 2, Name = "Ad2" };
410
411         Mock<IAdRepository> mock = new Mock<IAdRepository>();
412         mock.Setup(m => m.Ads).Returns(new List<Ad>
413         {
414             new Ad { Id = 1, Name = "Ad1" },
415             new Ad { Id = 2, Name = "Ad2" },
416             new Ad { Id = 3, Name = "Ad3" },
417             new Ad { Id = 4, Name = "Ad4" },
418             new Ad { Id = 5, Name = "Ad5" }
419         });
420
421         AdminController controller = new AdminController(mock.Object);
422
423         controller.Delete(ad.Id);
424
425         mock.Verify(m => m.DeleteAd(ad.Id));
426     }
427 }
428 }

```