*Федеральное государственное бюджетное образовательное учреждение высшего образования*

*«Владимирский государственный университет*

*имени Александра Григорьевича и Николая Григорьевича Столетовых»*

*Кафедра информационных систем и программной инженерии*

***ПОЯСНИТЕЛЬНАЯ ЗАПИСКА***

***к курсовому проекту по дисциплине   
"Технологии программирования"***

***на тему***

*Информационно-программная система «Рекламное агентство»*

*Выполнил: студент гр. ПРИ-115*

*Шопырев Д. Р.*

*Принял: доц. кафедры ИСПИ*

*Вершинин В.В.*

*Владимир, 2017*

**Аннотация**

Суть курсовой работы заключается в разработке информационно-программной системы для коммуникации менеджера, медиабайнера и клиента. Работа состоит из графической части и пояснительной записки. Графическая часть представляет собой 3 чертежа, выполненные на разных листах формата А4 в масштабе 1:8.

Пояснительная записка содержит:

* 40 страниц;
* 14 рисунков.

The essence of the course work is to develop information and software systems for communication manager, mediaminer and client. The work consists of a graphic part and an explanatory note. Graphical part is 3 drawings, made on different sheets of A4 format on a scale of 1: 8.

The explanatory memorandum contains:

* 40 pages;
* •14 figures

.

**Содержание**

[1. ВВЕДЕНИЕ 2](#_Toc501697091)

[2. Постановка задачи 3](#_Toc501697092)

[3. Описание предметной области 4](#_Toc501697093)

[3.1. Общие сведения о предмете автоматизации 4](#_Toc501697094)

[3.2. Словарь предметной области 5](#_Toc501697095)

[3.3. Сценарий взаимодействия пользователя с системой 5](#_Toc501697096)

[4. проектирование системы 8](#_Toc501697097)

[4.1 Диаграмма прецедентов 8](#_Toc501697098)

[4.2 Диаграмма классов 10](#_Toc501697099)

[4.3 Диаграмма состояний 12](#_Toc501697100)

[4.4 Диаграмма последовательности 13](#_Toc501697101)

[4.5 Структура базы данных 14](#_Toc501697102)

[5. реализация системы 16](#_Toc501697104)

[5.1 Принципы организации проекта 16](#_Toc501697105)

[5.2 Реализация модели 16](#_Toc501697106)

[5.3 Реализация поддержки различных типов пользователей 16](#_Toc501697107)

[5.4 Реализация поддержки различных справочников 17](#_Toc501697108)

[5.5 Реализация оформления продажи 17](#_Toc501697109)

[5.6 Пользовательский интерфейс 17](#_Toc501697110)

[6. заключение 22](#_Toc501697111)

[7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc501697112)

[ПРИЛОЖЕНИЕ А Исходный код программной системы 24](#_Toc501697113)

[ПРИЛОЖЕНИЕ Б Структура и Скрипты создания таблиц БД 40](#_Toc501697114)

1. ВВЕДЕНИЕ

Одной из актуальных проблем любого вида человеческой деятельности является проблема обработки увеличивающегося потока информации. Перспективным способом решения данной проблемы является автоматизация работы с информацией.

Автоматизировав работу с информацией в рекламном агенстве, мы получим программную систему, предназначенную для обработки данных о клиентах, менеджерах, медиабайнерах, СМИ, категориях рекламы и оформления заказов, все это не только повысит эффективность работы персонала, но и предоставит возможность удобного взаимодействия менеджера с клиентами.

Программная система в данном курсовом проекте разработана с использованием языка UML, реализация диаграмм с использованием среды Microsoft Office Visio и Enterprise Architect, реализация программного обеспечения с использованием технологий разработки ASP .NET MVC, Entity Framework. Среда программирования – Microsoft Visual Studio 2017.

1. Постановка задачи

Необходимо разработать программное обеспечение, ведущее учет поступавших заказов и обеспечивающее взаимодействие клиента с менеджером и менеджера с медиабайнером.

Для решения поставленной задачи, необходимо последовательно разработать следующие вопросы:

* выполнить исследование и анализ предметной области;
* разработать прототип программной системы;
* выполнить моделирование работы программной системы;
* разработать схему (разработка схем);
* реализовать программную систему с использованием выбранных средств и технологий.

Необходимо реализовать функции:

* авторизация;
* поддержка различных типов пользователей (администратор, менеджер, клиент, медиабайнер);
* поддержка справочника менеджеров(администратор);
* поддержка справочника категорий рекламы(для менеджера);
* поддержка справочника средств массовой информации(для медиабайнера);
* подача заказов и просмотр их состояния (для клиента);
* ответ на заказ пользователю (для менеджера).

Система должна быть реализована на языке C# в виде веб-приложения с использованием технологий ASP .NET MVC.

1. Описание предметной области
   1. Общие сведения о предмете автоматизации

Рекламное агентство является компанией независимого характера занимающиеся разработкой программ рекламного типа, непосредственно рекламой, а также созданием методик, благодаря которым можно эффективно продвигать товары.

Информационная система позволяет вести учет клиентов, СМИ, заказов, сотрудников. Предоставляет справочник категорий рекламы и справочник средств массовой информации.

Разработанная программная система предназначена для менеджеров, администраторов, медиабайнеров, клиентов. Администратор сопровождает и администрирует систему. Клиент делает заказ на проект и просматривает работу над заказом. Менеджер поддерживает справочник заказов и справочник категорий рекламы. Медиабайнер управляет списком СМИ и назначает его в категории рекламы. Каждый пользователь данной системы наделен уникальными правами на выполнения ряда функций.

Изначально пользователь заходит в систему как гость и выполняет одно из действий: регистрация, авторизация.

Если авторизация выполняется успешно, тогда клиенту назначаются права на просмотр предоставляемых услуг, заказ проекта и просмотр заказа, если он уже присутствует. В заказе проекта включаются такие функции как: создание, удаление, редактирование черновика. После этого он может отправить черновик в агентство. Так же пользователь может просмотреть все проекты, составленные им.

Менеджер просматривает заказ, отправленный пользователем, и приглашает клиента на прием. После этого менеджер оформляет заказ проекта и согласовывает его с клиентом. Клиент может посмотреть договор после авторизации. После этого менеджер оформляет заказ, прописывает основные пункты и, если ID клиента соответствует его ID в заказе, он может просмотреть его.

Справочник категорий рекламы представляется в виде разделов и содержит в себе: ID категории рекламы, ID средств массовой информации, визуальные средства, ID менеджера, срок действия рекламы. В справочнике заказов перечисляются ID заказа, ID менеджера, ID клиента, ID категории рекламы, имя продукта, описание пожеланий, статус. дата окончания заказа, дата снятия рекламы с оборота. Справочник СМИ имеет следующие атрибуты: ID средств массовой информации, наименование, предметная область, цена. Первые два справочники необходимы менеджеру для взаимодействия с клиентами. Третий справочник необходим для поддержки справочника категорий рекламы. Справочник категорий рекламы и справочник заказов обновляется и редактируется менеджером и медиабайнером. Справочник СМИ обновляется и редактируется медиабайнером.

Для сохранения и развития информационной системы предусмотрена роль администратора. Его основная функция в ИС является поддержка списка пользователей. Управляя списком пользователей, администратор назначает права, и имеет возможность редактирования прав.

Данная система необходима в решении проблемы обработки увеличивающегося потока информации. Автоматизировав работу с информацией по данной информационной системе, мы получим программную систему, предназначенную для обработки данных о сотрудниках, клиентах, заказах, доступных СМИ.

* 1. Словарь предметной области

Реклама – направление в маркетинговых коммуникациях, в рамках которого производится распространение информации для привлечения внимания к объекту рекламирования с целью формирования или поддержания интереса к нему.

Рекламное агентство – компания независимого характера занимающиеся разработкой программ рекламного типа, непосредственно рекламой, а также созданием методик, благодаря которым можно эффективно продвигать товары.

Средство массовой информации (СМИ) — средство донесения информации (словесной, звуковой, визуальной) по принципу широковещательного канала, охватывающее большую аудиторию и действующее на постоянной основе.   
К СМИ относятся: печатные издания (газеты, журналы и др.),   
электронные СМИ (телевидение, радио, интернет).

Менеджер – пользователь, зарегистрированный в системе являющийся лицом с высшим образованием. Имеет следующие характеристики: логин и пароль, адрес электронной почты, роль в системе, фамилию, имя, отчество, специальность, номер телефона. Основные функции менеджера: просмотр справочника заказов, оформление заказа проекта, поддержка справочника категорий рекламы.

Клиент – пользователь, зарегистрированный в системе желающий сделать заказ для создания рекламы товара или услуги. Имеет следующие характеристики: логин и пароль, адрес электронной почты, роль в системе, фамилию, имя, отчество, номер телефона, паспортные данные. Основные функции клиента: просмотр справочника услуг, заказ, просмотр заказа.

Администратор – пользователь, зарегистрированный в системе осуществляющий управление основными сущностями рекламного агентства. Имеет следующие характеристики: логин и пароль, адрес электронной почты, роль в системе, фамилию, имя, отчество. Основные функции администратора: поддержка справочника заказов, поддержка списка пользователей.

Гость – пользователь, не зарегистрированный или не авторизованный в системе. Основные функции гостя: регистрация, авторизация.

Медиабайнер - сотрудник, который занимается вопросами об использовании средств распространения рекламы. Самая основная задача медиапланера состоит в том, чтобы установилось соответствие между СМИ и целевой аудиторией. Таким образом, специалист ищет подходящую целевую аудиторию, а затем уже занимается подбором средств распространения рекламы. Имеет следующие характеристики: логин и пароль, адрес электронной почты, роль в системе, фамилию, имя, отчество, номер телефона. Основной функцией медиабайнера является поддержка справочника СМИ и добавление СМИ в категорию рекламы.

Справочник категорий рекламы – это список доступных услуг связанных с рекламой, который может постепенно расширятся. В справочнике категорий рекламы перечисляются визуальные средства, СМИ, срок действия рекламы.

* 1. Сценарий взаимодействия пользователя с системой

В спроектированной программной системе присутствуют следующие роли:

* администратор (Administrator);
* менеджер (Manager);
* медиабайнер (Mediabuiner)
* клиент (Client).

Подробное описание ролей с предусмотренными для них функциями представлено в словаре предметной области.

1. проектирование системы
   1. Диаграмма прецедентов

Для наглядного преставления требований к системе построим диаграмму прецедентов [1]. На рисунке 1 представлен перечень и описание некоторых прецедентов в проектируемой системе.

**Название:** Формирование заказа

**Краткое описание:** Клиент составляет заказ.

**Предусловие:** Пользователь находится на форме заказа. Ему предлагается ввести поля оформления заказа.

**Действующее лицо:** Клиент.

**Основной поток:**

1. Система перенаправляет клиента на форму создания заказа.

2. Пользователь заполняет все необходимые поля.

3. Пользователь выбирает один из пунктов: сохранить как черновик, отправить.

4. Пользователь нажимает отправить заказ.

5. При отправке заказа система сохраняет данные в БД.

**Альтернативный поток:** пользователь выбрал сохранить заказ

1. Альтернативный поток начинается с пункта 4 основного потока.

2. Система сохраняет введённые данные как шаблон.

3. Система сохраняет данные в БД.

**Альтернативный поток:** заполнены не все обязательные поля или заполнены неправильно.

1. Альтернативный поток начинается с пункта 4 основного потока.

2. Система сообщает пользователю о незаполненных полях, оповещая пользователя.

3. Система проверяет корректность заполненных полей и сообщает пользователю об этом.

**Постусловие:** Отправка заказа прошла успешно.

Диаграмма прецедентов представлена на рисунке 1.

**Название:** Поддержка справочника заказов.

**Краткое описание:** Менеджер оформляет заказ клиента для того, чтобы клиент ознакомился со всеми условиями создания, срока сдачи и прекращение рекламы заказа.

**Предусловие**: Пользователь находится на форме оформление заказа проекта. Ему предлагается ввести данные о заказе.

**Действующее лицо:** Менеджер.

**Основной поток:**

1. Открыть страницу со списком заказов

2. Пользователь выбирает отправленный заказ из списка.

3. Пользователь выбирает один из пунктов: обработать заказ, отклонить заказ.

4. Пользователь выбрал обработать заказ, и система перенаправляет его на форму добавления.

5. Пользователь заполняет необходимые поля.

6. Пользователь нажимает кнопку добавить.

**Альтернативный поток:** пользователь выбрал отклонить.

1. Альтернативный поток начинается с пункта 4 основного потока.

2. Статус заказа меняется на черновик.

**Альтернативный поток:** заполнены не все поля на формах или заполнены неправильно.

1. Альтернативный поток начинается с пункта 5 основного потока.

2. Система сообщает пользователю о незаполненных полях, оповещая пользователя.

3. Система проверяет корректность заполненных полей и сообщает пользователю об этом.

**Постусловие:** Пользователю по окончанию работы с заказом выдает одно из двух сообщений: заказ обработан или заказ откланен.

**Название:** Управление списком пользователей

**Краткое описание:** Данный прецедент выполняется при необходимости редактирования списка пользователя (назначение роли пользователю, удаление роли пользователя).

**Предусловие:** Пользователь находится на форме управления списком пользователей. Ему предлагается удалить, назначить роль.

**Действующие лицо:** Администратор.

**Основной поток:**

1. Прецедент начинается, когда пользователь нажимает на кнопку «Изменить роли пользователя».

2. Система перенаправляет пользователя на форму изменения роли.

3. Пользователь выбирает роль, которую хочет добавить.

4. При нажатии кнопки добавить роль система добавляет ее пользователю.

**Альтернативный поток: Удаление роли пользователя.**

1. Прецедент начинается, когда пользователь нажимает на кнопку «Изменить роли пользователя».

2. Система перенаправляет пользователя на форму изменения роли.

3. Пользователь выбирает роль, которую хочет удалить.

4. При нажатии кнопки удалить роль система удаляет ее у пользователя.

**Постусловие:** Редактирование прав пользователей прошло успешно.

**Название:** Поддержка списка СМИ

**Краткое описание:** Данный прецедент выполняется при необходимости редактирования списка СМИ.

**Предусловие:** Пользователь находится на форме редактирования списка СМИ. Ему предлагается добавить, изменить или удалить одно из СМИ.

**Действующее лицо:** Медиабайнер

**Основной поток:**

1. Система перенаправляет пользователя на форму добавления, изменения и редактирования списка.

2. Пользователь выбирает один из пунктов: добавить, изменить или удалить

3. Пользователь выбрал добавление, и система перенаправляет его на форму добавления.

4. Пользователь заполняет все необходимые поля на добавление.

5. При отправке система сохраняет данные в БД.

**Альтернативный поток:** пользователь выбрал изменение.

1. Альтернативный поток начинается с пункта 3 основного потока.

2. Система перенаправляет его на форму изменения данных.

3. Пользователь заполняет все необходимые поля на изменение.

4. При отправке система сохраняет данные в БД.

**Альтернативный поток:** пользователь выбрал удаление.

1. Альтернативный поток начинается с пункта 3 основного потока.

2. Система перенаправляет его на форму изменения удаления.

3. Пользователь выбирает СМИ, которое необходимо удалить.

4. При отправке система сохраняет данные в БД.

**Альтернативный поток:** заполнены не все поля на формах или заполнены неправильно.

1. Альтернативный поток начинается с пункта 3 основного потока.

2. Система сообщает пользователю о незаполненных полях, оповещая пользователя.

3. Система проверяет корректность заполненных полей и сообщает пользователю об этом.

**Постусловие:** Редактирование списка СМИ прошло успешно.

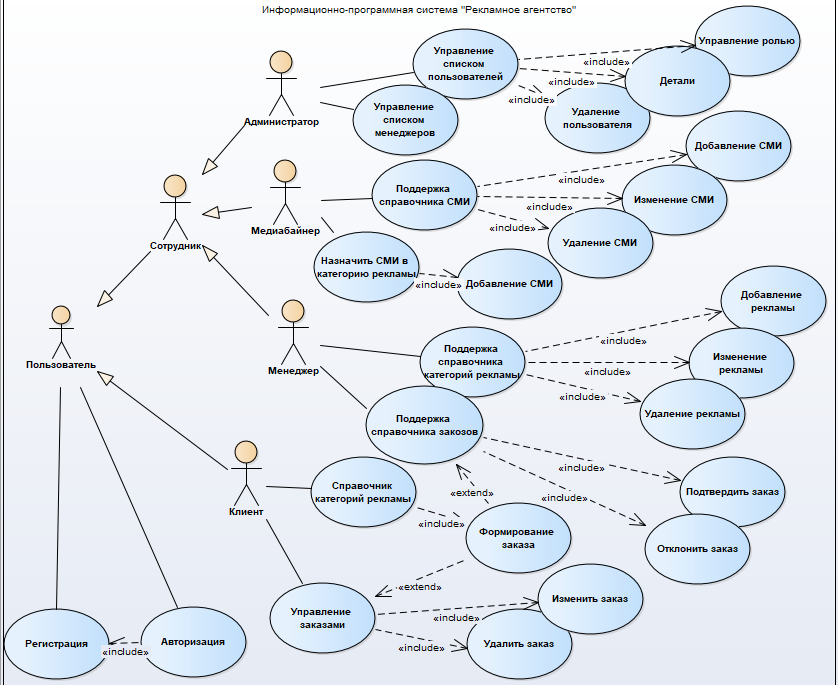


Рисунок 1. Диаграмма прецедентов.

* 1. Диаграмма классов

Моделирование предметной области позволяют сконцентрировать внимание на наиболее важных деталях проекта.

Краткое описание основных сущностей с атрибутами для информационной системы «Рекламное агентство» представлено ниже:

User (Пользователь системы) – пользователь, зарегистрированный в системе.

Атрибуты:

* password (пароль)
* role (роль в системе)
* e-mail (почта)
* surname (фамилия)
* name (имя)
* patronymic (отчество)
* phone (номер телефона)

Client (клиент) – пользователь в системе подающий заявку для заказа. Просматривает проект.

Атрибуты:

* id\_client (идентификатор клиента)
* passport (паспортные данные клиента)
* address (место жительства клиента)

Mediabuiner (медиабайнер) – пользователь в системе осуществляющий управление списком СМИ.

Атрибуты:

* id\_mediabuiner (идентификатор медиабайнера)
* work\_experience (стаж работы)

Manager (менеджер) – пользователь в системе осуществляющий просмотр справочника СМИ, поддержка справочника заказов, поддержка справочника категорий рекламы.

Атрибуты:

* id\_manager (идентификатор менеджера)
* number\_of\_processed\_client (число обработанных клиентов)
* work\_experience (стаж работы)

Mass-media (СМИ) – средство массовой информации.

Атрибуты:

* id\_mass-media (идентификатор СМИ)
* id\_mediabuiner (идентификатор медиабайнера)
* name\_mass-media (название СМИ)
* price (цена работы со СМИ)
* subject\_area (предметная область)

Administrator (администратор) – пользователь в системе осуществляющий управление основными сущностями рекламного агенства.

Category\_advertising (категория рекламы) – характеристика определенный рекламы.

* id\_category\_advertising (идентификатор категорий рекламы)
* id\_manager (идентификатор менеджера)
* id\_mass\_media (СМИ)
* visual\_aid(визуальные средства)
* advertising\_period (срок действия рекламы)

Order (заказ) – заказ клиента в разных статусах.

* id\_order (идентификатор заказа)
* id\_manager (идентификатор менеджера)
* id\_client (идентификатор клиента)
* id\_categoty\_advertising (идентификатор категории рекламы)
* product\_name(название продукта)
* description\_wish(описание пожеланий)
* term\_of\_delivery (срок сдачи заказа)
* date (дата заказа)
* date\_of\_withdrawal\_of\_advertising\_turnover (дата снятия рекламы с оборота)

<<enumeretion>> Role (<<перечисление>> состояние пользователя)

* administrator
* client
* manager
* mediabuiner

<<enumeretion>> Status (<<перечисление>> состояние заказа)

* sent (отправлен)
* draft (черновик)
* treatment (обработан)

Диаграмма классов предметной области представлена на рисунке 2.

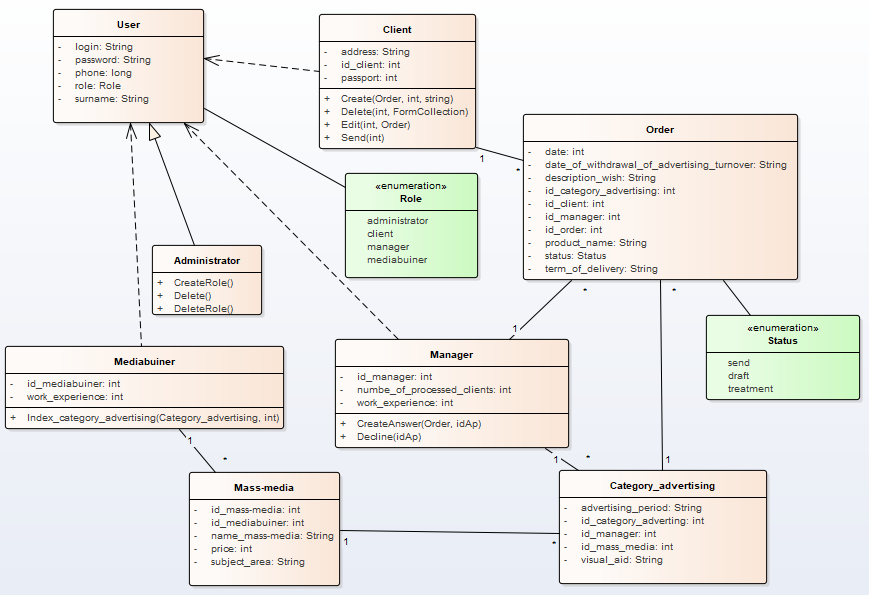


Рисунок 2. Диаграмма классов.

* 1. Диаграмма состояний

Состояние объекта «Заказ».

1. Пользователь (Клиент) составляет заказ для рекламного агентства, заполняя все необходимые поля. Объект «Заказ» создан в системе (объект в памяти).
2. Производится либо отправка заказа, либо сохранение заказа в черновик. То есть объект добавляется в БД с соответствующим статусом («Черновик», «Отправлен»).
3. Если заказ находится в статусе черновик, то возможно ее удаление. Так же можно изменить заказ или отправить заказ – переход в статус «Отправлен».
4. После отправки заказ ждет подтверждения. Заказ либо подтверждается, так если менеджер обрабатывает заказ, либо отклоняется (меняет статус на черновик) менеджером. То есть объект обновляется в БД с соответствующим статусом («Черновик», «Обработан»).

Диаграмма состояния объекта заявка представлена на рисунке 3.

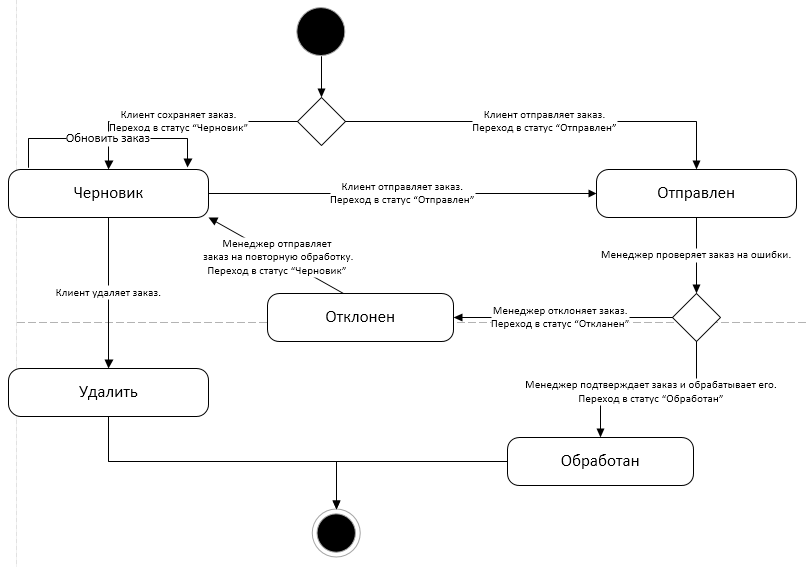


Рисунок 3. Диаграмма состояний объекта «Заявка».

* 1. Диаграмма последовательности

Диаграмма последовательности показывает участников взаимодействия и последовательность сообщений, которыми они обмениваются. Она описывает взаимодействие системы с действующими лицами в процессе полного или частичного выполнения варианта использования [1].

Диаграммы последовательности, показывающая участников взаимодействия и последовательность сообщений разработана для прецедентов «Создание и отправка заказа», «Ответ на заказ» представлены на рисунках 4-5.

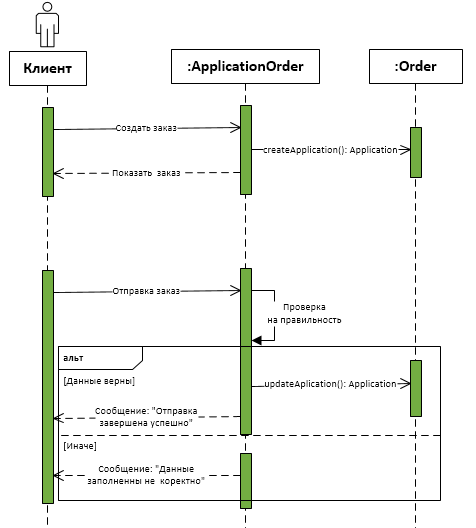


Рисунок 4. Диаграмма последовательностей. Создание и отправка заказа.

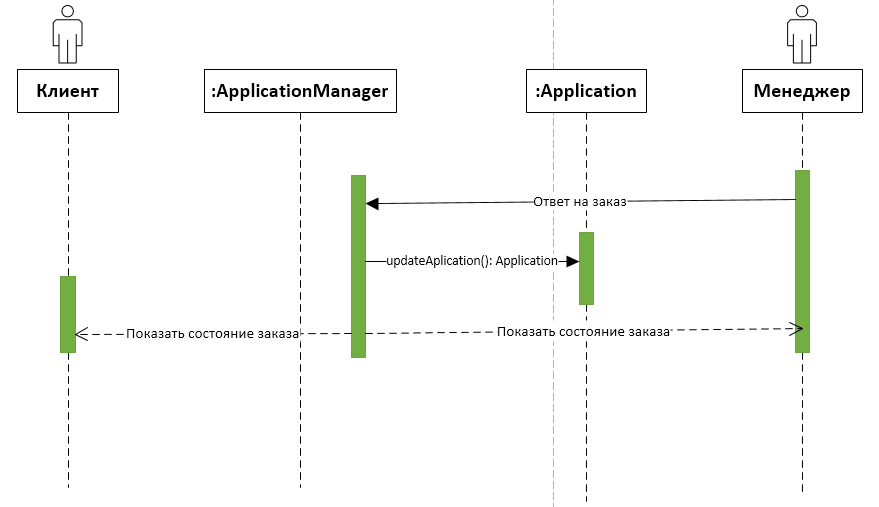


Рисунок 5. Диаграмма последовательностей. Ответ на заявку.

* 1. Структура базы данных

Разработанная физическая структура базы данных приведена к третей нормальной форме. Схема физической модели базы данных приведена на рисунке 6 [2]. Скрипты создания таблиц базы данных приведены в приложении Б.

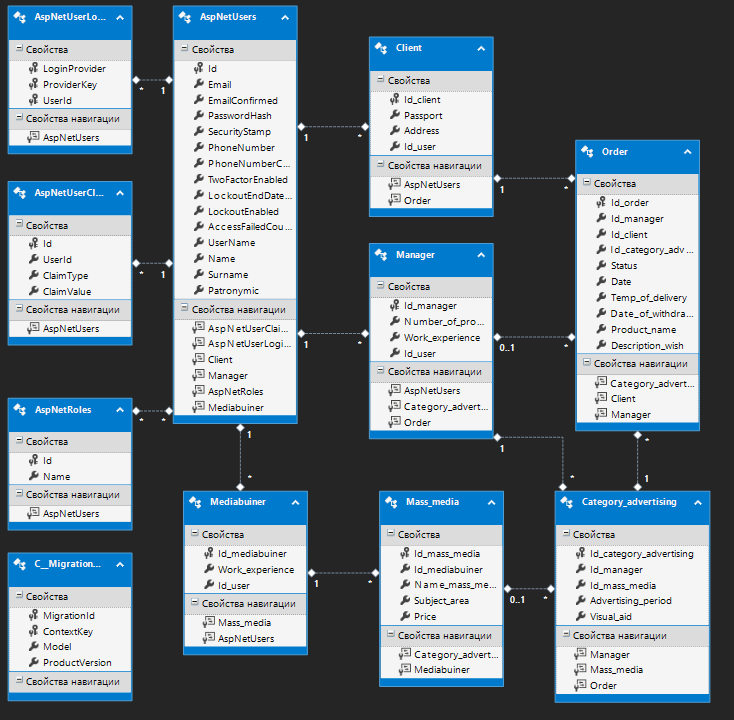


Рисунок 6. Модель базы данных.

1. реализация системы
   1. Принципы организации проекта

Разрабатываемая система построена на архитектуре Model – View – Controller и таким образом, веб-приложение было разделено на три части: контроллер, модель, представление. А так же в приложении используется технологии Entity Framework, и язык интегрированных запросов LINQ[3].

А также стоит отметить, приложение использует сторонние веб-сервисы, представляющие собой справочник диагнозов и справочник болезней. Исходный код веб-сервиса приведен в приложении В.

При разработке кода для того что бы отслеживать состояние работы программы использовался NLog а так же написаны тесты представленные в приложении Г.

* 1. Реализация модели

На основе физической структуры базы данных были созданы таблицы в базе данных с указанием первичных ключей (установленных как автоинкрементенты) и вторичных ключей. После чего на основе созданной базы данных создавалась модель с использованием технологии ADO.NET Entity Framework.

* 1. Реализация поддержки различных типов пользователей

Для реализации регистрации, аутентификации и авторизации пользователей были использованы стандартные методы шаблона Microsoft Identity. По умолчанию при регистрации первого пользователя создается набор таблиц, позволяющий осуществлять ряд функций, перечисленных выше.

Пользователям, в зависимости от роли, доступны различные функции, с помощью ограничения методов в контроллере и ограничения кнопок в меню Mvc.sitemap.

Неавторизованному пользователю доступны страницы просмотра справочников, главная станица, а также страница авторизации и страница регистрации.

Зарегистрированному пользователю автоматически присваивается роль посетителя. Роль администратора, менеджера и медиабайнера назначается исключительно администратором.

* 1. Пользовательский интерфейс

Пользовательский интерфейс представлен на Рисунках 7-31.

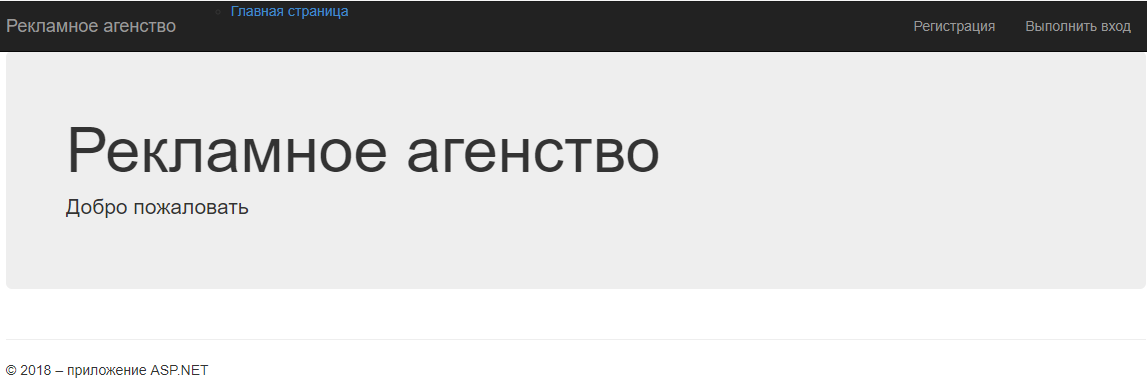


Рисунок 7. Главная страница.

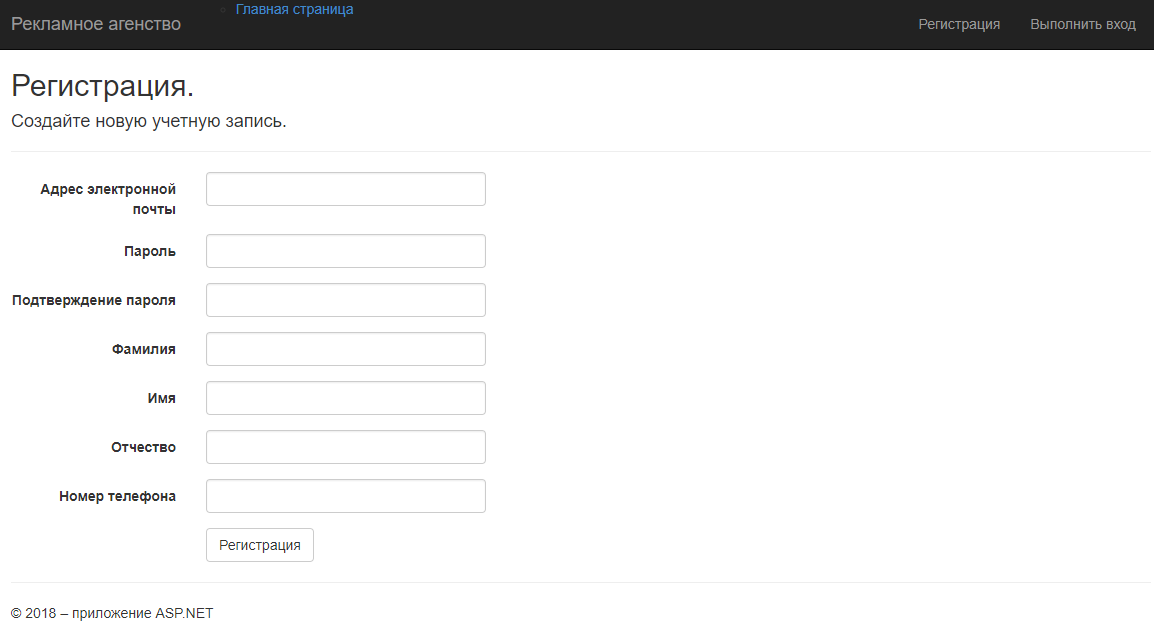


Рисунок 8. Форма регистрации.

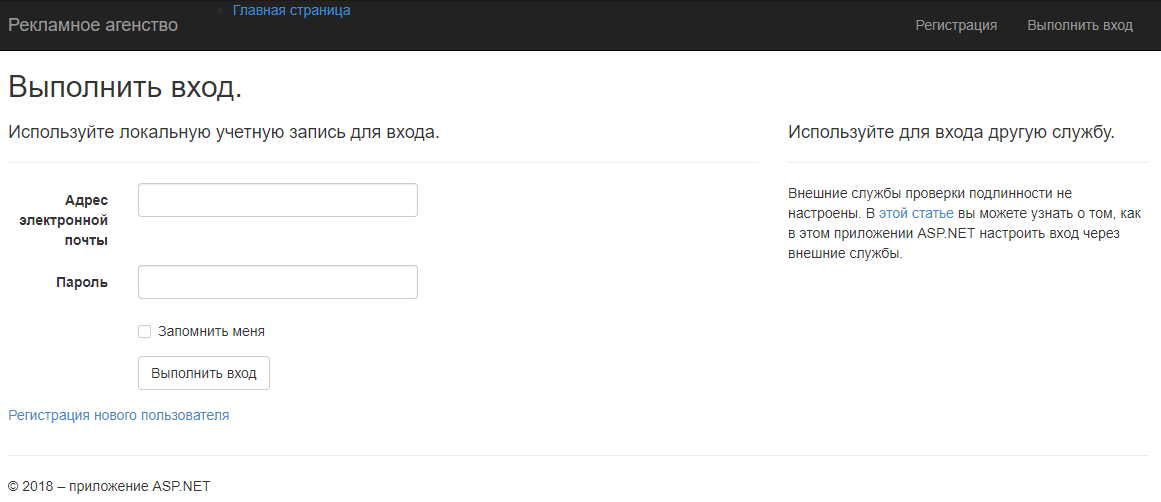


Рисунок 9. Форма входа в систему.

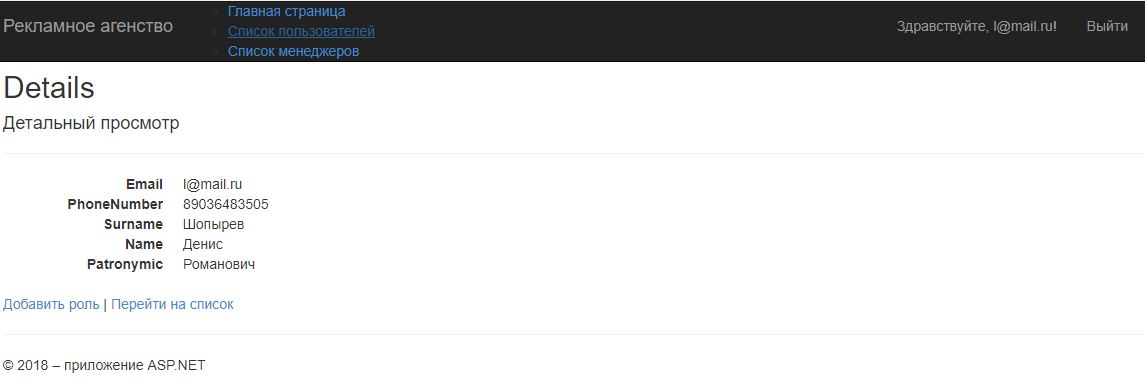


Рисунок 10. Личная информация пользователя в системе.

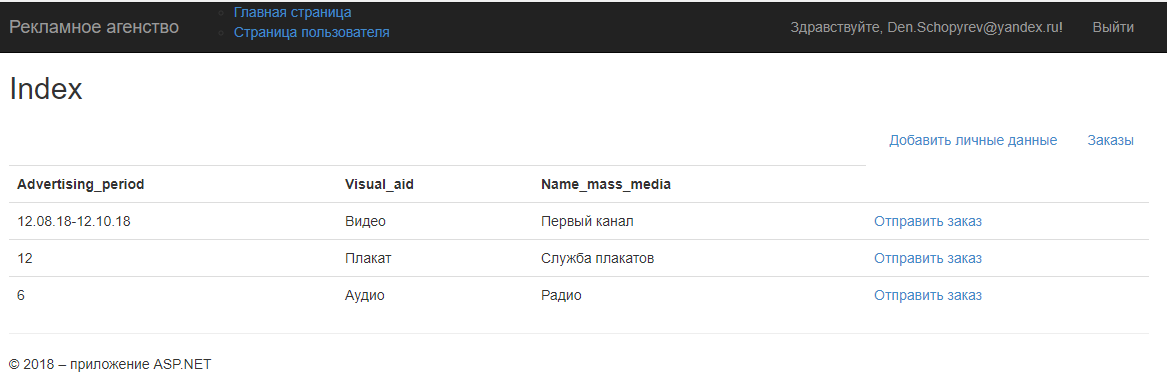


Рисунок 11. Страница пользователя.

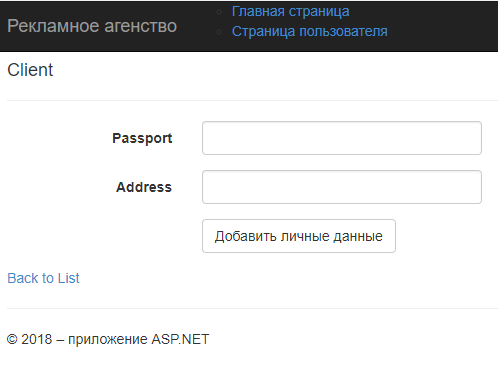


Рисунок 12. Добавление личной информации.

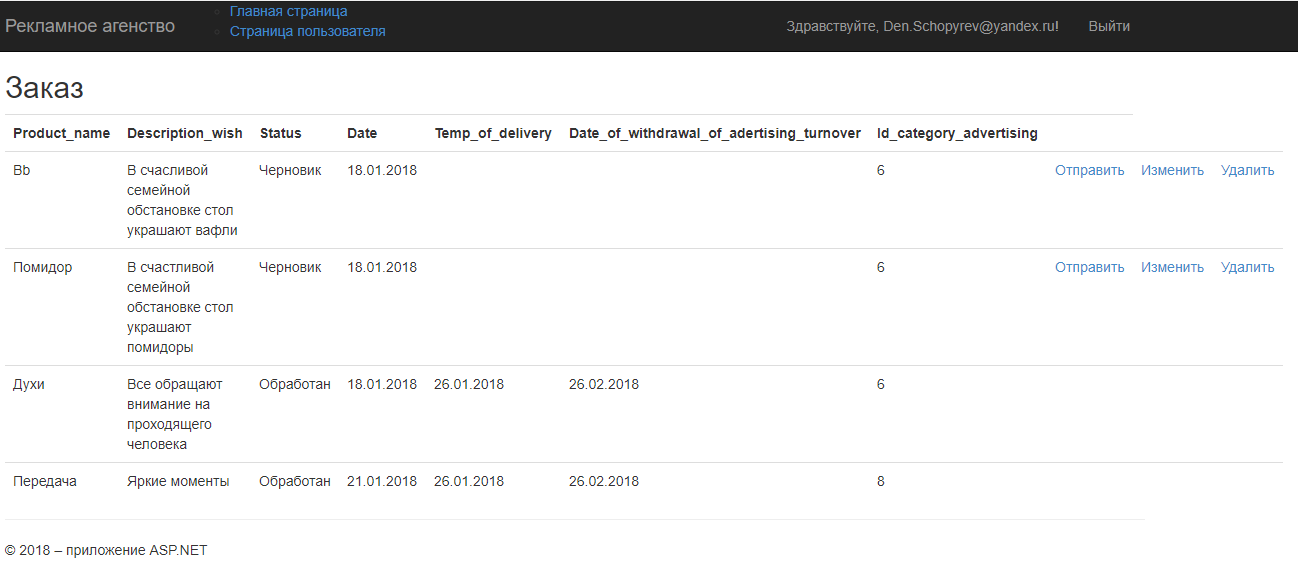


Рисунок 13. Заказы пользователя.

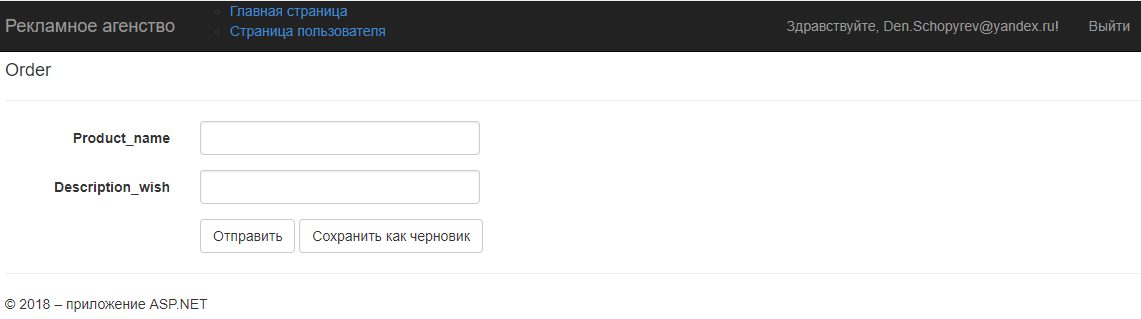


Рисунок 14. Создание заказа.

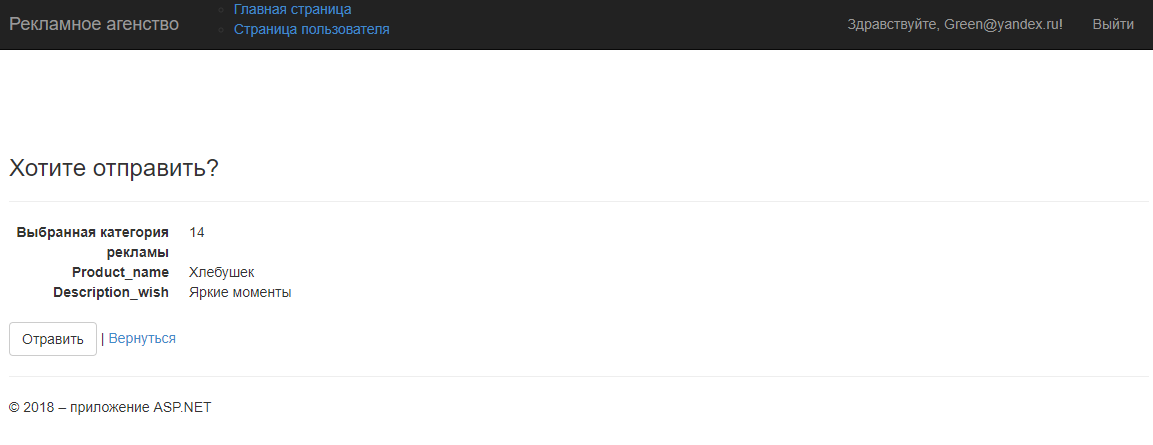


Рисунок 15. Отправить заказ из черновика.

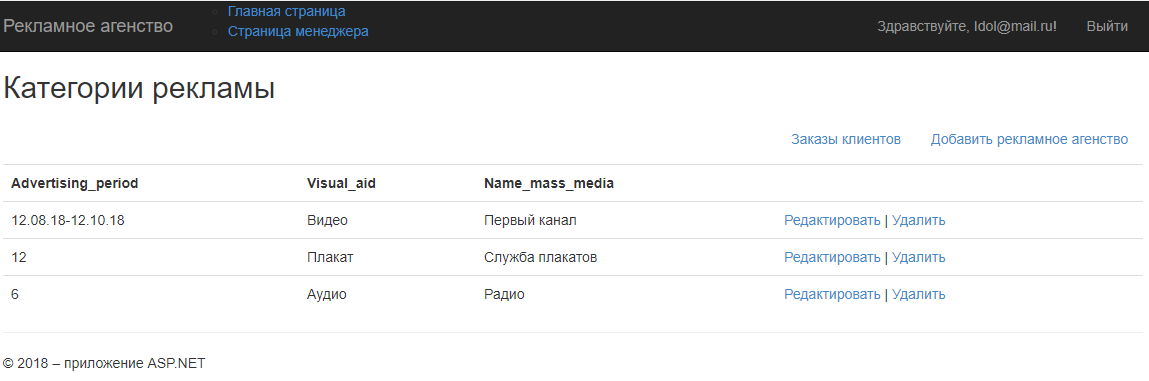


Рисунок 16. Страница менеджера.

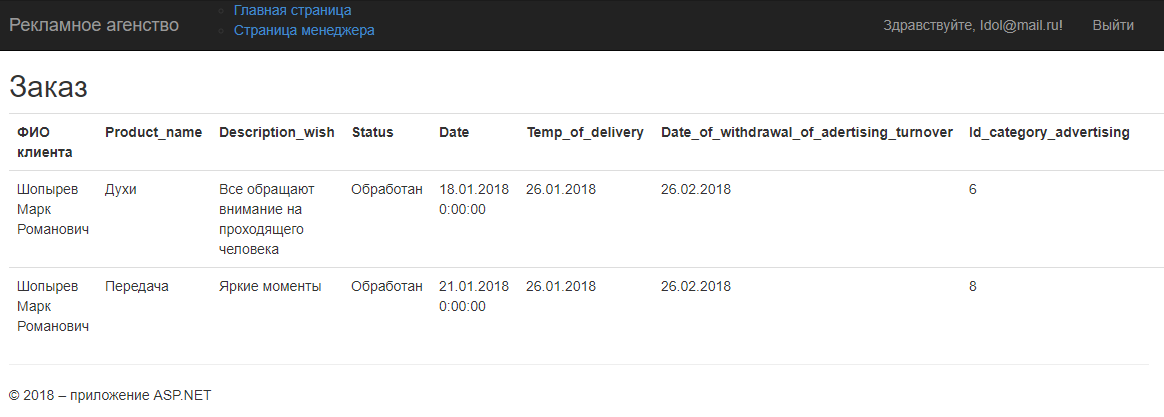


Рисунок 16. Заказы клиентов.

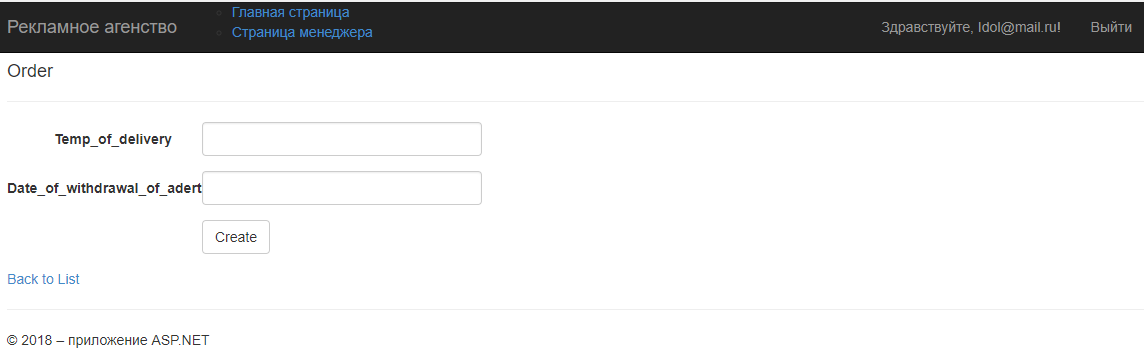


Рисунок 17. Страница обработки заказа.

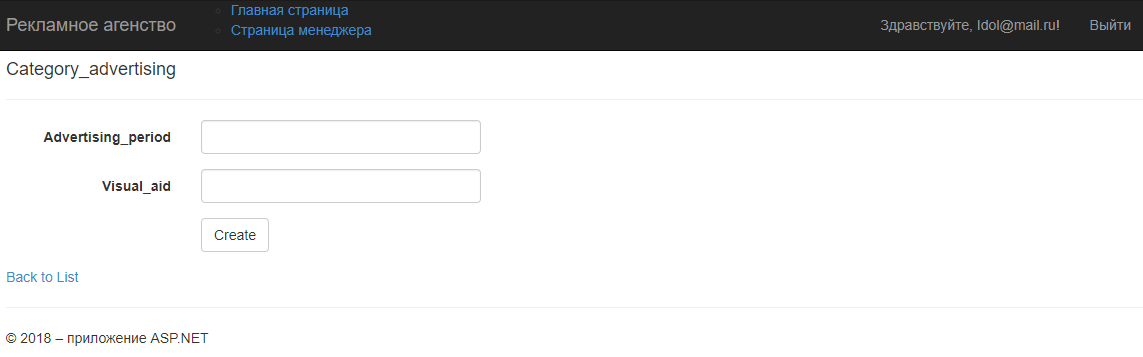


Рисунок 18. Добавить рекламное агентство.

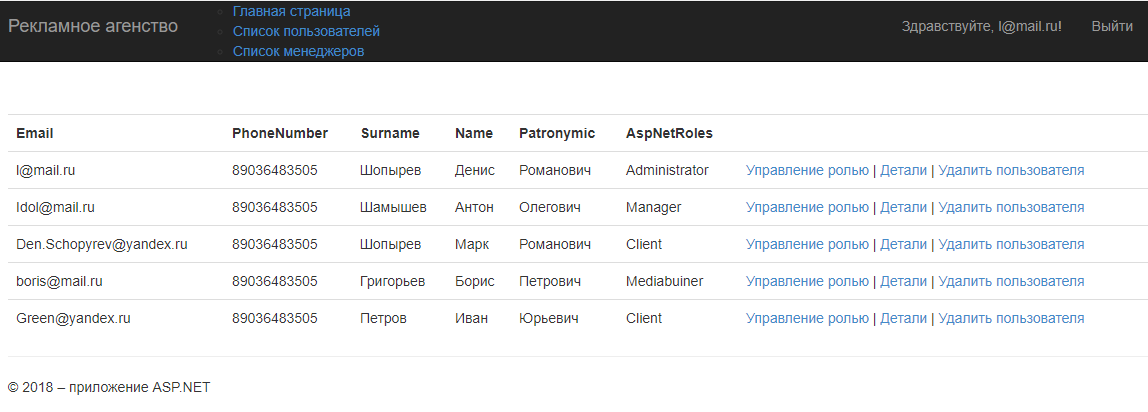


Рисунок 19. Список пользователей для администратора.

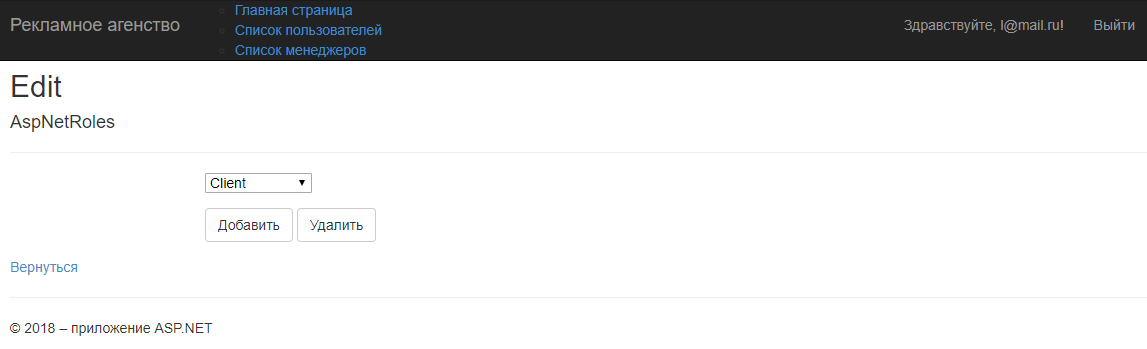


Рисунок 25. Управление ролью.

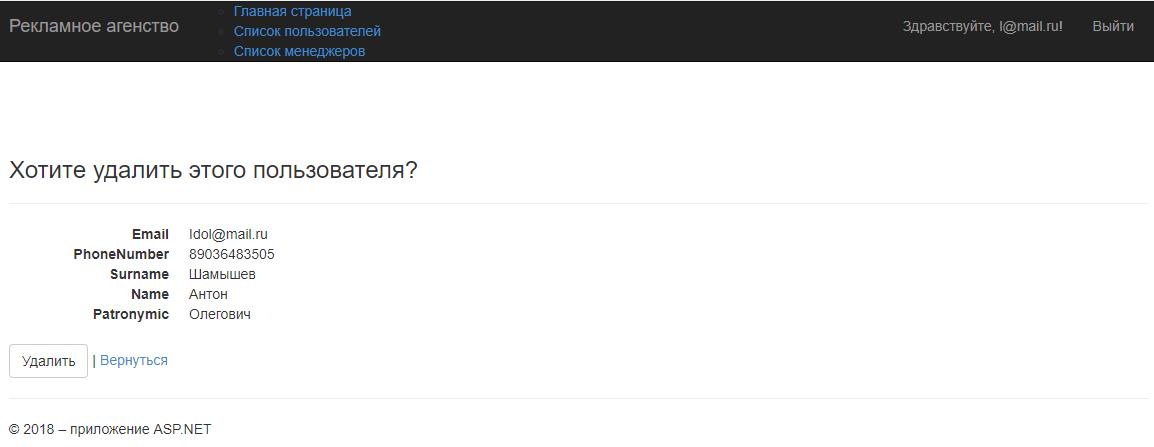


Рисунок 26. Удаление пользователя.

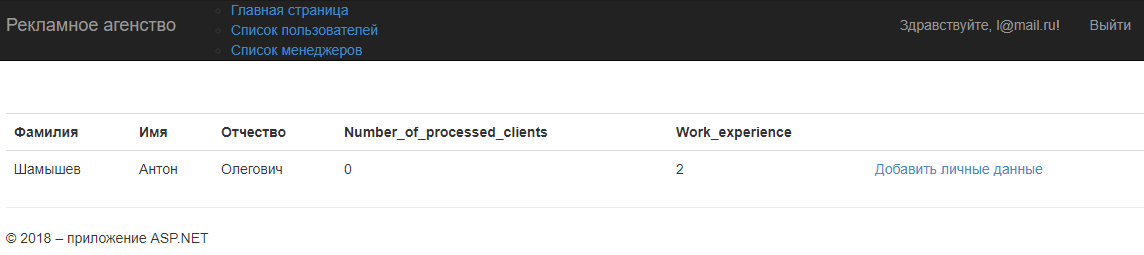


Рисунок 27. Список менеджеров для администратора.

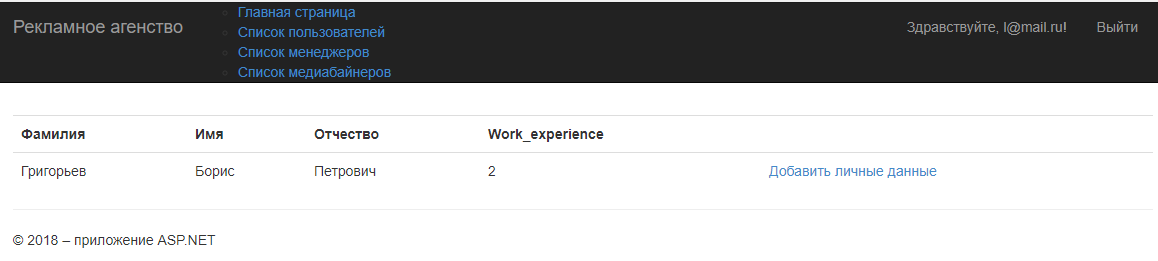


Рисунок 28. Список медиабаинеров для администратора.

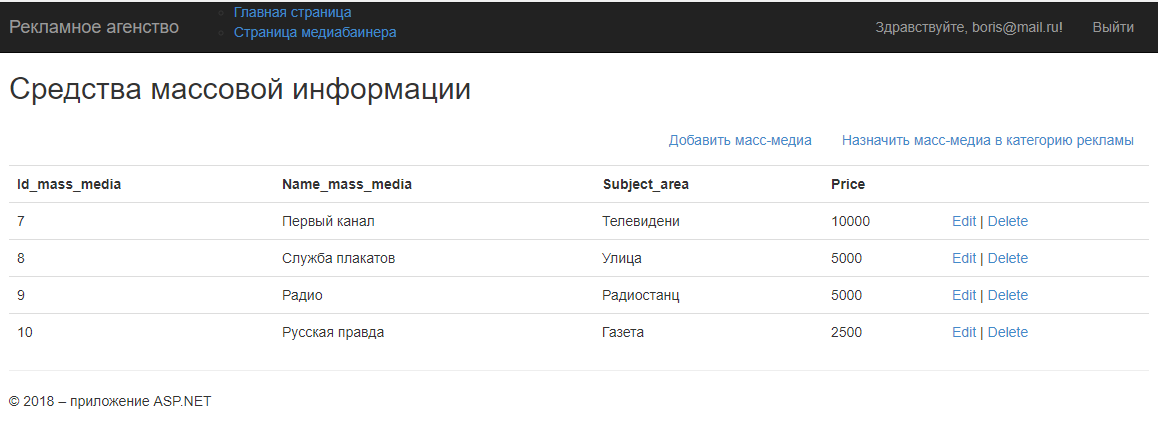


Рисунок 29. Страница медиабайнера.

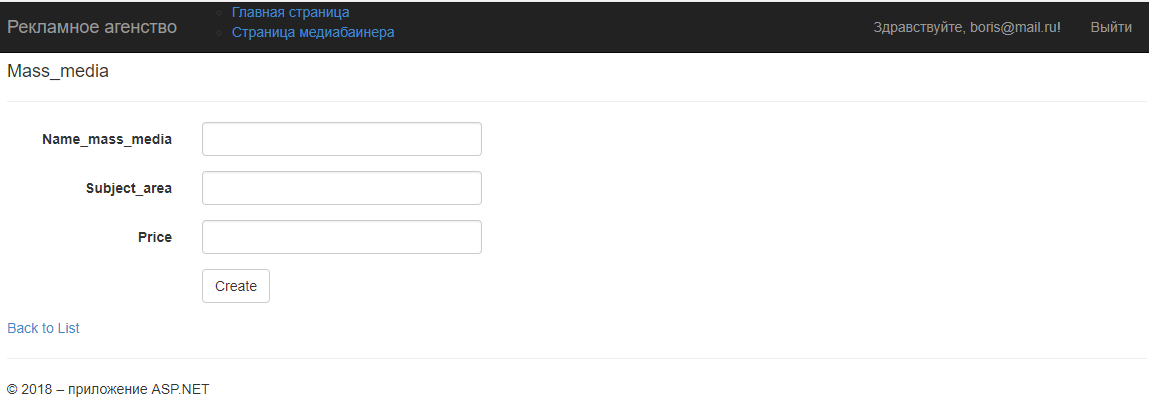


Рисунок 30. Добавить СМИ.

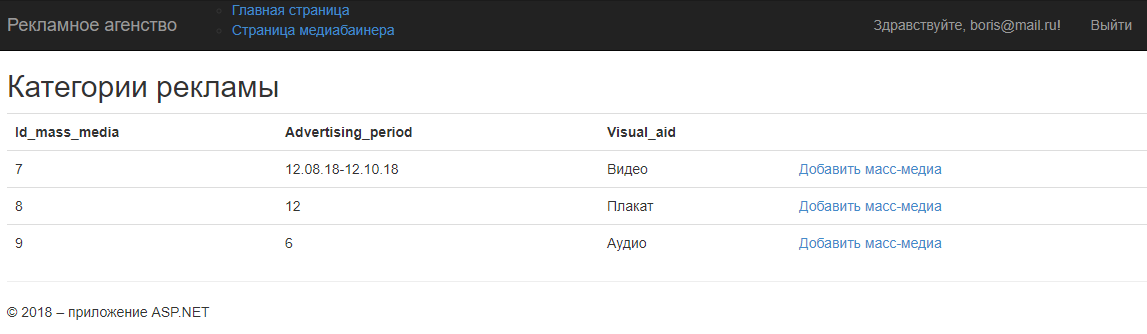


Рисунок 31. Назначить СМИ в категорию рекламы.

* 1. заключение

В курсовом проекте была спроектирована и реализована программная система. В процессе выполнения был проведен анализ предметной области, выполнена разработка программной системы, а также выполнена реализация программного обеспечения.

Система была реализована на языке C# в виде веб-приложения с использованием технологий ASP .NET MVC в процессе проектирования использовался язык UML.

Разработанная программная система позволяет работать со средствами массовой информации, категориями рекламы. А также реализована коммуникация между клиентом и менеджером по средствам отправки заказа пользователями и ответами на заказы менеджерами.

* 1. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка Питер, 2007
2. Кузнецов С. Д. Основы баз данных. — 2-е изд. — М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007. — 484 с. — ISBN 978-5-94774-736-2
3. Кармайкл Э., Хейвуд Д. Быстрая и качественная разработка программного обеспечения. М.: изд. дом «Вильямс, 2003»

ПРИЛОЖЕНИЕ А Исходный код программной системы

**Модели:**

**AccountViewModels.cs**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace WebApplication4.Models

{

public class ExternalLoginConfirmationViewModel

{

[Required]

[Display(Name = "Адрес электронной почты")]

public string Email { get; set; }

}

public class ExternalLoginListViewModel

{

public string ReturnUrl { get; set; }

}

public class SendCodeViewModel

{

public string SelectedProvider { get; set; }

public ICollection<System.Web.Mvc.SelectListItem> Providers { get; set; }

public string ReturnUrl { get; set; }

public bool RememberMe { get; set; }

}

public class VerifyCodeViewModel

{

[Required]

public string Provider { get; set; }

[Required]

[Display(Name = "Код")]

public string Code { get; set; }

public string ReturnUrl { get; set; }

[Display(Name = "Запомнить браузер?")]

public bool RememberBrowser { get; set; }

public bool RememberMe { get; set; }

}

public class ForgotViewModel

{

[Required]

[Display(Name = "Адрес электронной почты")]

public string Email { get; set; }

}

public class LoginViewModel

{

[Required]

[Display(Name = "Адрес электронной почты")]

[EmailAddress]

public string Email { get; set; }

[Required]

[DataType(DataType.Password)]

[Display(Name = "Пароль")]

public string Password { get; set; }

[Display(Name = "Запомнить меня")]

public bool RememberMe { get; set; }

}

public class RegisterViewModel

{

[Required]

[EmailAddress]

[Display(Name = "Адрес электронной почты")]

public string Email { get; set; }

[Required]

[StringLength(100, ErrorMessage = "Значение {0} должно содержать не менее {2} символов.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Пароль")]

public string Password { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Подтверждение пароля")]

[Compare("Password", ErrorMessage = "Пароль и его подтверждение не совпадают.")]

public string ConfirmPassword { get; set; }

[Required]

// Валидация

[Display(Name = "Фамилия")]

public string Surname { get; set; }

[Required]

// Валидация

[Display(Name = "Имя")]

public string Name { get; set; }

[Required]

// Валидация

[Display(Name = "Отчество")]

public string Patronymic { get; set; }

[Required]

// Валидация

[Display(Name = "Номер телефона")]

public string PhoneNumber { get; set; }

}

public class ResetPasswordViewModel

{

[Required]

[EmailAddress]

[Display(Name = "Адрес электронной почты")]

public string Email { get; set; }

[Required]

[StringLength(100, ErrorMessage = "Значение {0} должно содержать не менее {2} символов.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Пароль")]

public string Password { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Подтверждение пароля")]

[Compare("Password", ErrorMessage = "Пароль и его подтверждение не совпадают.")]

public string ConfirmPassword { get; set; }

public string Code { get; set; }

}

public class ForgotPasswordViewModel

{

[Required]

[EmailAddress]

[Display(Name = "Почта")]

public string Email { get; set; }

}

}

**IdentityModels.cs**

using System.Data.Entity;

using System.Security.Claims;

using System.Threading.Tasks;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.EntityFramework;

namespace WebApplication4.Models

{

// Чтобы добавить данные профиля для пользователя, можно добавить дополнительные свойства в класс ApplicationUser. Дополнительные сведения см. по адресу: http://go.microsoft.com/fwlink/?LinkID=317594.

public class ApplicationUser : IdentityUser

{

public string Name { get; set; } // Имя

public string Surname { get; set; } // Фамилия

public string Patronymic { get; set; } // Отчество

public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)

{

// Обратите внимание, что authenticationType должен совпадать с типом, определенным в CookieAuthenticationOptions.AuthenticationType

var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);

// Здесь добавьте утверждения пользователя

return userIdentity;

}

}

public class ApplicationDbContext : IdentityDbContext<ApplicationUser>

{

public ApplicationDbContext()

: base("DefaultConnection", throwIfV1Schema: false)

{

}

public static ApplicationDbContext Create()

{

return new ApplicationDbContext();

}

}

}

**ManageViewModels.cs**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using Microsoft.AspNet.Identity;

using Microsoft.Owin.Security;

namespace WebApplication4.Models

{

public class IndexViewModel

{

public bool HasPassword { get; set; }

public IList<UserLoginInfo> Logins { get; set; }

public string PhoneNumber { get; set; }

public bool TwoFactor { get; set; }

public bool BrowserRemembered { get; set; }

}

public class ManageLoginsViewModel

{

public IList<UserLoginInfo> CurrentLogins { get; set; }

public IList<AuthenticationDescription> OtherLogins { get; set; }

}

public class FactorViewModel

{

public string Purpose { get; set; }

}

public class SetPasswordViewModel

{

[Required]

[StringLength(100, ErrorMessage = "Значение {0} должно содержать символов не менее: {2}.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Новый пароль")]

public string NewPassword { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Подтверждение нового пароля")]

[Compare("NewPassword", ErrorMessage = "Новый пароль и его подтверждение не совпадают.")]

public string ConfirmPassword { get; set; }

}

public class ChangePasswordViewModel

{

[Required]

[DataType(DataType.Password)]

[Display(Name = "Текущий пароль")]

public string OldPassword { get; set; }

[Required]

[StringLength(100, ErrorMessage = "Значение {0} должно содержать символов не менее: {2}.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Новый пароль")]

public string NewPassword { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Подтверждение нового пароля")]

[Compare("NewPassword", ErrorMessage = "Новый пароль и его подтверждение не совпадают.")]

public string ConfirmPassword { get; set; }

}

public class AddPhoneNumberViewModel

{

[Required]

[Phone]

[Display(Name = "Номер телефона")]

public string Number { get; set; }

}

public class VerifyPhoneNumberViewModel

{

[Required]

[Display(Name = "Код")]

public string Code { get; set; }

[Required]

[Phone]

[Display(Name = "Номер телефона")]

public string PhoneNumber { get; set; }

}

public class ConfigureTwoFactorViewModel

{

public string SelectedProvider { get; set; }

public ICollection<System.Web.Mvc.SelectListItem> Providers { get; set; }

}

}

**AspNetRoles.cs**

namespace WebApplication4.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

public partial class AspNetRoles

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public AspNetRoles()

{

this.AspNetUsers = new HashSet<AspNetUsers>();

}

public string Id { get; set; }

[Display(Name = "Наименование роли")]

public string Name { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AspNetUsers> AspNetUsers { get; set; }

}

}

**AspNetUserClaims.cs**

namespace WebApplication4.Models

{

using System;

using System.Collections.Generic;

public partial class AspNetUserClaims

{

public int Id { get; set; }

public string UserId { get; set; }

public string ClaimType { get; set; }

public string ClaimValue { get; set; }

public virtual AspNetUsers AspNetUsers { get; set; }

}

}

**AspNetUserLogins.cs**

namespace WebApplication4.Models

{

using System;

using System.Collections.Generic;

public partial class AspNetUserLogins

{

public string LoginProvider { get; set; }

public string ProviderKey { get; set; }

public string UserId { get; set; }

public virtual AspNetUsers AspNetUsers { get; set; }

}

}

**AspNetUsers.cs**

namespace WebApplication4.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

public partial class AspNetUsers

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public AspNetUsers()

{

this.AspNetUserClaims = new HashSet<AspNetUserClaims>();

this.AspNetUserLogins = new HashSet<AspNetUserLogins>();

this.Client = new HashSet<Client>();

this.Manager = new HashSet<Manager>();

this.AspNetRoles = new HashSet<AspNetRoles>();

this.Mediabuiner = new HashSet<Mediabuiner>();

}

public string Id { get; set; }

[Display(Name = "Электронная почта")]

public string Email { get; set; }

public bool EmailConfirmed { get; set; }

public string PasswordHash { get; set; }

public string SecurityStamp { get; set; }

[Display(Name = "Номер телефона")]

public string PhoneNumber { get; set; }

public bool PhoneNumberConfirmed { get; set; }

public bool TwoFactorEnabled { get; set; }

public Nullable<System.DateTime> LockoutEndDateUtc { get; set; }

public bool LockoutEnabled { get; set; }

public int AccessFailedCount { get; set; }

[Display(Name = "Имя пользователя в системе")]

public string UserName { get; set; }

[Display(Name = "Имя")]

public string Name { get; set; }

[Display(Name = "Фамилия")]

public string Surname { get; set; }

[Display(Name = "Отчество")]

public string Patronymic { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AspNetUserClaims> AspNetUserClaims { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AspNetUserLogins> AspNetUserLogins { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Client> Client { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Manager> Manager { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<AspNetRoles> AspNetRoles { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Mediabuiner> Mediabuiner { get; set; }

}

}

**C\_\_MigrationHistory.cs**

namespace WebApplication4.Models

{

using System;

using System.Collections.Generic;

public partial class C\_\_MigrationHistory

{

public string MigrationId { get; set; }

public string ContextKey { get; set; }

public byte[] Model { get; set; }

public string ProductVersion { get; set; }

}

}

**Category\_advertising.cs**

namespace WebApplication4.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

public partial class Category\_advertising

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Category\_advertising()

{

this.Order = new HashSet<Order>();

}

public int Id\_category\_advertising { get; set; }

public int Id\_manager { get; set; }

public Nullable<int> Id\_mass\_media { get; set; }

[Display(Name = "Срок действия рекламы")]

public string Advertising\_period { get; set; }

[Display(Name = "Визуальные средства")]

public string Visual\_aid { get; set; }

public virtual Manager Manager { get; set; }

public virtual Mass\_media Mass\_media { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Order> Order { get; set; }

}

}

**Client.cs**

namespace WebApplication4.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

public partial class Client

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Client()

{

this.Order = new HashSet<Order>();

}

public int Id\_client { get; set; }

[Display(Name = "Паспорт")]

public string Passport { get; set; }

[Display(Name = "Место проживания")]

public string Address { get; set; }

public string Id\_user { get; set; }

public virtual AspNetUsers AspNetUsers { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Order> Order { get; set; }

}

}

**Manager.cs**

namespace WebApplication4.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

public partial class Manager

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Manager()

{

this.Category\_advertising = new HashSet<Category\_advertising>();

this.Order = new HashSet<Order>();

}

public int Id\_manager { get; set; }

[Display(Name = "Число обработанных клиентов")]

public int Number\_of\_processed\_clients { get; set; }

[Display(Name = "Стаж работы (в месяцах)")]

public int Work\_experience { get; set; }

public string Id\_user { get; set; }

public virtual AspNetUsers AspNetUsers { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Category\_advertising> Category\_advertising { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Order> Order { get; set; }

}

}

**Mass\_media.cs**

namespace WebApplication4.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

public partial class Mass\_media

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Mass\_media()

{

this.Category\_advertising = new HashSet<Category\_advertising>();

}

public int Id\_mass\_media { get; set; }

public int Id\_mediabuiner { get; set; }

[Display(Name = "Название СМИ")]

public string Name\_mass\_media { get; set; }

[Display(Name = "Предметная область")]

public string Subject\_area { get; set; }

[Display(Name = "Цена")]

public int Price { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Category\_advertising> Category\_advertising { get; set; }

public virtual Mediabuiner Mediabuiner { get; set; }

}

}

**Mediabuiner.cs**

namespace WebApplication4.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

public partial class Mediabuiner

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Mediabuiner()

{

this.Mass\_media = new HashSet<Mass\_media>();

}

public int Id\_mediabuiner { get; set; }

[Display(Name = "Стаж работы (в месяцах)")]

public int Work\_experience { get; set; }

public string Id\_user { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Mass\_media> Mass\_media { get; set; }

public virtual AspNetUsers AspNetUsers { get; set; }

}

}

**Order.cs**

namespace WebApplication4.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

public partial class Order

{

public int Id\_order { get; set; }

public Nullable<int> Id\_manager { get; set; }

public int Id\_client { get; set; }

public int Id\_category\_advertising { get; set; }

[Display(Name = "Статус")]

public string Status { get; set; }

[Display(Name = "Дата заказа")]

public System.DateTime Date { get; set; }

[Display(Name = "Срок сдачи заказа")]

public string Temp\_of\_delivery { get; set; }

[Display(Name = "Дата снятия рекламы с оборота")]

public string Date\_of\_withdrawal\_of\_adertising\_turnover { get; set; }

[Display(Name = "Название продукта")]

public string Product\_name { get; set; }

[Display(Name = "Описание пожеланий")]

public string Description\_wish { get; set; }

public virtual Category\_advertising Category\_advertising { get; set; }

public virtual Client Client { get; set; }

public virtual Manager Manager { get; set; }

}

}

**Контроллеры:**

**AccountController.cs**

using System;

using System.Globalization;

using System.Linq;

using System.Security.Claims;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin.Security;

using WebApplication4.Models;

namespace WebApplication4.Controllers

{

[Authorize]

public class AccountController : Controller

{

private ApplicationSignInManager \_signInManager;

private ApplicationUserManager \_userManager;

public AccountController()

{

}

public AccountController(ApplicationUserManager userManager, ApplicationSignInManager signInManager )

{

UserManager = userManager;

SignInManager = signInManager;

}

public ApplicationSignInManager SignInManager

{

get

{

return \_signInManager ?? HttpContext.GetOwinContext().Get<ApplicationSignInManager>();

}

private set

{

\_signInManager = value;

}

}

public ApplicationUserManager UserManager

{

get

{

return \_userManager ?? HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();

}

private set

{

\_userManager = value;

}

}

//

// GET: /Account/Login

[AllowAnonymous]

public ActionResult Login(string returnUrl)

{

ViewBag.ReturnUrl = returnUrl;

return View();

}

//

// POST: /Account/Login

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)

{

if (!ModelState.IsValid)

{

return View(model);

}

// Сбои при входе не приводят к блокированию учетной записи

// Чтобы ошибки при вводе пароля инициировали блокирование учетной записи, замените на shouldLockout: true

var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, shouldLockout: false);

switch (result)

{

case SignInStatus.Success:

return RedirectToLocal(returnUrl);

case SignInStatus.LockedOut:

return View("Lockout");

case SignInStatus.RequiresVerification:

return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = model.RememberMe });

case SignInStatus.Failure:

default:

ModelState.AddModelError("", "Неудачная попытка входа.");

return View(model);

}

}

//

// GET: /Account/VerifyCode

[AllowAnonymous]

public async Task<ActionResult> VerifyCode(string provider, string returnUrl, bool rememberMe)

{

// Требовать предварительный вход пользователя с помощью имени пользователя и пароля или внешнего имени входа

if (!await SignInManager.HasBeenVerifiedAsync())

{

return View("Error");

}

return View(new VerifyCodeViewModel { Provider = provider, ReturnUrl = returnUrl, RememberMe = rememberMe });

}

//

// POST: /Account/VerifyCode

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> VerifyCode(VerifyCodeViewModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

// Приведенный ниже код защищает от атак методом подбора, направленных на двухфакторные коды.

// Если пользователь введет неправильные коды за указанное время, его учетная запись

// будет заблокирована на заданный период.

// Параметры блокирования учетных записей можно настроить в IdentityConfig

var result = await SignInManager.TwoFactorSignInAsync(model.Provider, model.Code, isPersistent: model.RememberMe, rememberBrowser: model.RememberBrowser);

switch (result)

{

case SignInStatus.Success:

return RedirectToLocal(model.ReturnUrl);

case SignInStatus.LockedOut:

return View("Lockout");

case SignInStatus.Failure:

default:

ModelState.AddModelError("", "Неправильный код.");

return View(model);

}

}

//

// GET: /Account/Register

[AllowAnonymous]

public ActionResult Register()

{

return View();

}

//

// POST: /Account/Register

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

var user = new ApplicationUser { UserName = model.Email, Email = model.Email };

var result = await UserManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

await SignInManager.SignInAsync(user, isPersistent:false, rememberBrowser:false);

// Дополнительные сведения о том, как включить подтверждение учетной записи и сброс пароля, см. по адресу: http://go.microsoft.com/fwlink/?LinkID=320771

// Отправка сообщения электронной почты с этой ссылкой

// string code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);

// var callbackUrl = Url.Action("ConfirmEmail", "Account", new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);

// await UserManager.SendEmailAsync(user.Id, "Подтверждение учетной записи", "Подтвердите вашу учетную запись, щелкнув <a href=\"" + callbackUrl + "\">здесь</a>");

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

}

AddErrors(result);

}

// Появление этого сообщения означает наличие ошибки; повторное отображение формы

return View(model);

}

//

// GET: /Account/ConfirmEmail

[AllowAnonymous]

public async Task<ActionResult> ConfirmEmail(string userId, string code)

{

if (userId == null || code == null)

{

return View("Error");

}

var result = await UserManager.ConfirmEmailAsync(userId, code);

return View(result.Succeeded ? "ConfirmEmail" : "Error");

}

//

// 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)))

{

// Не показывать, что пользователь не существует или не подтвержден

return View("ForgotPasswordConfirmation");

}

// Дополнительные сведения о том, как включить подтверждение учетной записи и сброс пароля, см. по адресу: http://go.microsoft.com/fwlink/?LinkID=320771

// Отправка сообщения электронной почты с этой ссылкой

// 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, "Сброс пароля", "Сбросьте ваш пароль, щелкнув <a href=\"" + callbackUrl + "\">здесь</a>");

// return RedirectToAction("ForgotPasswordConfirmation", "Account");

}

// Появление этого сообщения означает наличие ошибки; повторное отображение формы

return View(model);

}

//

// GET: /Account/ForgotPasswordConfirmation

[AllowAnonymous]

public ActionResult ForgotPasswordConfirmation()

{

return View();

}

//

// GET: /Account/ResetPassword

[AllowAnonymous]

public ActionResult ResetPassword(string code)

{

return code == null ? View("Error") : View();

}

//

// POST: /Account/ResetPassword

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> ResetPassword(ResetPasswordViewModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

var user = await UserManager.FindByNameAsync(model.Email);

if (user == null)

{

// Не показывать, что пользователь не существует

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

}

var result = await UserManager.ResetPasswordAsync(user.Id, model.Code, model.Password);

if (result.Succeeded)

{

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

}

AddErrors(result);

return View();

}

//

// GET: /Account/ResetPasswordConfirmation

[AllowAnonymous]

public ActionResult ResetPasswordConfirmation()

{

return View();

}

//

// POST: /Account/ExternalLogin

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public ActionResult ExternalLogin(string provider, string returnUrl)

{

// Запрос перенаправления к внешнему поставщику входа

return new ChallengeResult(provider, Url.Action("ExternalLoginCallback", "Account", new { ReturnUrl = returnUrl }));

}

//

// GET: /Account/SendCode

[AllowAnonymous]

public async Task<ActionResult> SendCode(string returnUrl, bool rememberMe)

{

var userId = await SignInManager.GetVerifiedUserIdAsync();

if (userId == null)

{

return View("Error");

}

var userFactors = await UserManager.GetValidTwoFactorProvidersAsync(userId);

var factorOptions = userFactors.Select(purpose => new SelectListItem { Text = purpose, Value = purpose }).ToList();

return View(new SendCodeViewModel { Providers = factorOptions, ReturnUrl = returnUrl, RememberMe = rememberMe });

}

//

// POST: /Account/SendCode

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> SendCode(SendCodeViewModel model)

{

if (!ModelState.IsValid)

{

return View();

}

// Создание и отправка маркера

if (!await SignInManager.SendTwoFactorCodeAsync(model.SelectedProvider))

{

return View("Error");

}

return RedirectToAction("VerifyCode", new { Provider = model.SelectedProvider, ReturnUrl = model.ReturnUrl, RememberMe = model.RememberMe });

}

//

// GET: /Account/ExternalLoginCallback

[AllowAnonymous]

public async Task<ActionResult> ExternalLoginCallback(string returnUrl)

{

var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync();

if (loginInfo == null)

{

return RedirectToAction("Login");

}

// Выполнение входа пользователя посредством данного внешнего поставщика входа, если у пользователя уже есть имя входа

var result = await SignInManager.ExternalSignInAsync(loginInfo, isPersistent: false);

switch (result)

{

case SignInStatus.Success:

return RedirectToLocal(returnUrl);

case SignInStatus.LockedOut:

return View("Lockout");

case SignInStatus.RequiresVerification:

return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = false });

case SignInStatus.Failure:

default:

// Если у пользователя нет учетной записи, то ему предлагается создать ее

ViewBag.ReturnUrl = returnUrl;

ViewBag.LoginProvider = loginInfo.Login.LoginProvider;

return View("ExternalLoginConfirmation", new ExternalLoginConfirmationViewModel { Email = loginInfo.Email });

}

}

//

// POST: /Account/ExternalLoginConfirmation

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> ExternalLoginConfirmation(ExternalLoginConfirmationViewModel model, string returnUrl)

{

if (User.Identity.IsAuthenticated)

{

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

}

if (ModelState.IsValid)

{

// Получение сведений о пользователе от внешнего поставщика входа

var info = await AuthenticationManager.GetExternalLoginInfoAsync();

if (info == null)

{

return View("ExternalLoginFailure");

}

var user = new ApplicationUser { UserName = model.Email, Email = model.Email };

var result = await UserManager.CreateAsync(user);

if (result.Succeeded)

{

result = await UserManager.AddLoginAsync(user.Id, info.Login);

if (result.Succeeded)

{

await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

return RedirectToLocal(returnUrl);

}

}

AddErrors(result);

}

ViewBag.ReturnUrl = returnUrl;

return View(model);

}

//

// POST: /Account/LogOff

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult LogOff()

{

AuthenticationManager.SignOut(DefaultAuthenticationTypes.ApplicationCookie);

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

}

//

// GET: /Account/ExternalLoginFailure

[AllowAnonymous]

public ActionResult ExternalLoginFailure()

{

return View();

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

if (\_userManager != null)

{

\_userManager.Dispose();

\_userManager = null;

}

if (\_signInManager != null)

{

\_signInManager.Dispose();

\_signInManager = null;

}

}

base.Dispose(disposing);

}

#region Вспомогательные приложения

// Используется для защиты от XSRF-атак при добавлении внешних имен входа

private const string XsrfKey = "XsrfId";

private IAuthenticationManager AuthenticationManager

{

get

{

return HttpContext.GetOwinContext().Authentication;

}

}

private void AddErrors(IdentityResult result)

{

foreach (var error in result.Errors)

{

ModelState.AddModelError("", error);

}

}

private ActionResult RedirectToLocal(string returnUrl)

{

if (Url.IsLocalUrl(returnUrl))

{

return Redirect(returnUrl);

}

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

}

internal class ChallengeResult : HttpUnauthorizedResult

{

public ChallengeResult(string provider, string redirectUri)

: this(provider, redirectUri, null)

{

}

public ChallengeResult(string provider, string redirectUri, string userId)

{

LoginProvider = provider;

RedirectUri = redirectUri;

UserId = userId;

}

public string LoginProvider { get; set; }

public string RedirectUri { get; set; }

public string UserId { get; set; }

public override void ExecuteResult(ControllerContext context)

{

var properties = new AuthenticationProperties { RedirectUri = RedirectUri };

if (UserId != null)

{

properties.Dictionary[XsrfKey] = UserId;

}

context.HttpContext.GetOwinContext().Authentication.Challenge(properties, LoginProvider);

}

}

#endregion

}

}

**ClientController.cs**

using WebApplication4.DAO;

using WebApplication4.Models;

using System;

using Microsoft.AspNet.Identity;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication4.DAO

{

public class ClientController : Controller

{

private DAO\_client clientDAO = new DAO\_client();

private DAO\_category\_advertising category\_advertisingDAO = new DAO\_category\_advertising();

// GET: Client

[Authorize(Roles = "Client")]

[HttpGet]

public ActionResult Index()

{

return View(category\_advertisingDAO.GetAll());

}

[Authorize(Roles = "Client")]

[HttpGet]

public ActionResult Create(string id)

{

return View();

}

[Authorize(Roles = "Client")]

[HttpPost]

public ActionResult Create(Client collection)

{

if (clientDAO.Add(collection, User.Identity.GetUserId()))

{

return RedirectToAction("Message", "Message", new { str = "Информация пользователя добавлена!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

}

}

**HomeController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication4.Controllers

{

public class HomeController : Controller

{

public ActionResult Index()

{

return View();

}

public ActionResult About()

{

ViewBag.Message = "Your application description page.";

return View();

}

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

return View();

}

}

}

**ListManagerController.cs**

using WebApplication4.DAO;

using WebApplication4.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication4.Controllers

{

public class ListManagerController : Controller

{

private DAO\_manager managerDAO = new DAO\_manager();

// GET: ListManager

[Authorize(Roles = "Administrator")]

[HttpGet]

public ActionResult Index(int? id)

{

return View(managerDAO.GetAll());

}

[Authorize(Roles = "Administrator")]

[HttpGet]

public ActionResult CreateInfo(string id)

{

return View();

}

// GET: Manager

[Authorize(Roles = "Administrator")]

[HttpPost]

public ActionResult CreateInfo(Manager collection, string id)

{

if (managerDAO.Add(collection, id))

{

return RedirectToAction("Message", "Message", new { str = "Личная информация добавлена!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

}

}

**ListMediabuinerController.cs**

using WebApplication4.DAO;

using WebApplication4.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication4.Controllers

{

public class ListMediabuinerController : Controller

{

private DAO\_mediabuiner mediabuinerDAO = new DAO\_mediabuiner();

[Authorize(Roles = "Administrator")]

[HttpGet]

public ActionResult Index(int? id)

{

return View(mediabuinerDAO.GetAll());

}

[Authorize(Roles = "Administrator")]

[HttpGet]

public ActionResult CreateInfo(string id)

{

return View();

}

[Authorize(Roles = "Administrator")]

[HttpPost]

public ActionResult CreateInfo(Mediabuiner collection, string id)

{

if (mediabuinerDAO.Add(collection, id))

{

return RedirectToAction("Message", "Message", new { str = "Личная информация добавлена!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

}

}

**ListUserController.cs**

using WebApplication4.DAO;

using WebApplication4.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication4.Controllers

{

public class ListUserController : Controller

{

private DAO\_user userDAO = new DAO\_user();

[Authorize(Roles = "Administrator")]

[HttpGet]

public ActionResult Index()

{

return View(userDAO.GetAll());

}

[Authorize(Roles = "Administrator")]

[HttpGet]

public ActionResult Details(string id)

{

return View(userDAO.Get(id));

}

[Authorize(Roles = "Administrator")]

[HttpGet]

public ActionResult Edit(string id)

{

return View();

}

[Authorize(Roles = "Administrator")]

[HttpPost]

public ActionResult Edit(string id, AspNetRoles collection, string submitButton)

{

switch (submitButton)

{

case "Добавить":

if (userDAO.UpdateRoles(id, collection))

{

return RedirectToAction("Message", "Message", new { str = "Добавление роли прошло успешно!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

case "Удалить":

if (userDAO.DeleteRoles(id, collection))

{

return RedirectToAction("Message", "Message", new { str = "Удаление прошло успешно!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

default:

return (View());

}

}

[Authorize(Roles = "Administrator")]

[HttpGet]

public ActionResult Delete(string id)

{

return View("Delete", userDAO.Get(id));

}

// POST: List\_user/Delete/5

[Authorize(Roles = "Administrator")]

[HttpPost]

public ActionResult Delete(string id, FormCollection collection)

{

if (userDAO.Delete(id))

{

return RedirectToAction("Message", "Message", new { str = "Удаленее пользователя прошло успешно!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

}

}

**ManageController.cs**

using System;

using System.Linq;

using System.Threading.Tasks;

using System.Web;

using System.Web.Mvc;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin.Security;

using WebApplication4.Models;

namespace WebApplication4.Controllers

{

[Authorize]

public class ManageController : Controller

{

private ApplicationSignInManager \_signInManager;

private ApplicationUserManager \_userManager;

public ManageController()

{

}

public ManageController(ApplicationUserManager userManager, ApplicationSignInManager signInManager)

{

UserManager = userManager;

SignInManager = signInManager;

}

public ApplicationSignInManager SignInManager

{

get

{

return \_signInManager ?? HttpContext.GetOwinContext().Get<ApplicationSignInManager>();

}

private set

{

\_signInManager = value;

}

}

public ApplicationUserManager UserManager

{

get

{

return \_userManager ?? HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();

}

private set

{

\_userManager = value;

}

}

//

// GET: /Manage/Index

public async Task<ActionResult> Index(ManageMessageId? message)

{

ViewBag.StatusMessage =

message == ManageMessageId.ChangePasswordSuccess ? "Ваш пароль изменен."

: message == ManageMessageId.SetPasswordSuccess ? "Пароль задан."

: message == ManageMessageId.SetTwoFactorSuccess ? "Настроен поставщик двухфакторной проверки подлинности."

: message == ManageMessageId.Error ? "Произошла ошибка."

: message == ManageMessageId.AddPhoneSuccess ? "Ваш номер телефона добавлен."

: message == ManageMessageId.RemovePhoneSuccess ? "Ваш номер телефона удален."

: "";

var userId = User.Identity.GetUserId();

var model = new IndexViewModel

{

HasPassword = HasPassword(),

PhoneNumber = await UserManager.GetPhoneNumberAsync(userId),

TwoFactor = await UserManager.GetTwoFactorEnabledAsync(userId),

Logins = await UserManager.GetLoginsAsync(userId),

BrowserRemembered = await AuthenticationManager.TwoFactorBrowserRememberedAsync(userId)

};

return View(model);

}

//

// POST: /Manage/RemoveLogin

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> RemoveLogin(string loginProvider, string providerKey)

{

ManageMessageId? message;

var result = await UserManager.RemoveLoginAsync(User.Identity.GetUserId(), new UserLoginInfo(loginProvider, providerKey));

if (result.Succeeded)

{

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null)

{

await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

}

message = ManageMessageId.RemoveLoginSuccess;

}

else

{

message = ManageMessageId.Error;

}

return RedirectToAction("ManageLogins", new { Message = message });

}

//

// GET: /Manage/AddPhoneNumber

public ActionResult AddPhoneNumber()

{

return View();

}

//

// POST: /Manage/AddPhoneNumber

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> AddPhoneNumber(AddPhoneNumberViewModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

// Создание и отправка маркера

var code = await UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(), model.Number);

if (UserManager.SmsService != null)

{

var message = new IdentityMessage

{

Destination = model.Number,

Body = "Ваш код безопасности: " + code

};

await UserManager.SmsService.SendAsync(message);

}

return RedirectToAction("VerifyPhoneNumber", new { PhoneNumber = model.Number });

}

//

// POST: /Manage/EnableTwoFactorAuthentication

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> EnableTwoFactorAuthentication()

{

await UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), true);

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null)

{

await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

}

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

}

//

// POST: /Manage/DisableTwoFactorAuthentication

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> DisableTwoFactorAuthentication()

{

await UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), false);

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null)

{

await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

}

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

}

//

// GET: /Manage/VerifyPhoneNumber

public async Task<ActionResult> VerifyPhoneNumber(string phoneNumber)

{

var code = await UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(), phoneNumber);

// Отправка SMS через поставщик SMS для проверки номера телефона

return phoneNumber == null ? View("Error") : View(new VerifyPhoneNumberViewModel { PhoneNumber = phoneNumber });

}

//

// POST: /Manage/VerifyPhoneNumber

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> VerifyPhoneNumber(VerifyPhoneNumberViewModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

var result = await UserManager.ChangePhoneNumberAsync(User.Identity.GetUserId(), model.PhoneNumber, model.Code);

if (result.Succeeded)

{

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null)

{

await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

}

return RedirectToAction("Index", new { Message = ManageMessageId.AddPhoneSuccess });

}

// Это сообщение означает наличие ошибки; повторное отображение формы

ModelState.AddModelError("", "Не удалось проверить телефон");

return View(model);

}

//

// GET: /Manage/RemovePhoneNumber

public async Task<ActionResult> RemovePhoneNumber()

{

var result = await UserManager.SetPhoneNumberAsync(User.Identity.GetUserId(), null);

if (!result.Succeeded)

{

return RedirectToAction("Index", new { Message = ManageMessageId.Error });

}

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null)

{

await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

}

return RedirectToAction("Index", new { Message = ManageMessageId.RemovePhoneSuccess });

}

//

// GET: /Manage/ChangePassword

public ActionResult ChangePassword()

{

return View();

}

//

// POST: /Manage/ChangePassword

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> ChangePassword(ChangePasswordViewModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

var result = await UserManager.ChangePasswordAsync(User.Identity.GetUserId(), model.OldPassword, model.NewPassword);

if (result.Succeeded)

{

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null)

{

await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

}

return RedirectToAction("Index", new { Message = ManageMessageId.ChangePasswordSuccess });

}

AddErrors(result);

return View(model);

}

//

// GET: /Manage/SetPassword

public ActionResult SetPassword()

{

return View();

}

//

// POST: /Manage/SetPassword

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> SetPassword(SetPasswordViewModel model)

{

if (ModelState.IsValid)

{

var result = await UserManager.AddPasswordAsync(User.Identity.GetUserId(), model.NewPassword);

if (result.Succeeded)

{

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user != null)

{

await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

}

return RedirectToAction("Index", new { Message = ManageMessageId.SetPasswordSuccess });

}

AddErrors(result);

}

// Это сообщение означает наличие ошибки; повторное отображение формы

return View(model);

}

//

// GET: /Manage/ManageLogins

public async Task<ActionResult> ManageLogins(ManageMessageId? message)

{

ViewBag.StatusMessage =

message == ManageMessageId.RemoveLoginSuccess ? "Внешнее имя входа удалено."

: message == ManageMessageId.Error ? "Произошла ошибка."

: "";

var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

if (user == null)

{

return View("Error");

}

var userLogins = await UserManager.GetLoginsAsync(User.Identity.GetUserId());

var otherLogins = AuthenticationManager.GetExternalAuthenticationTypes().Where(auth => userLogins.All(ul => auth.AuthenticationType != ul.LoginProvider)).ToList();

ViewBag.ShowRemoveButton = user.PasswordHash != null || userLogins.Count > 1;

return View(new ManageLoginsViewModel

{

CurrentLogins = userLogins,

OtherLogins = otherLogins

});

}

//

// POST: /Manage/LinkLogin

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult LinkLogin(string provider)

{

// Запрос перенаправления к внешнему поставщику входа для связывания имени входа текущего пользователя

return new AccountController.ChallengeResult(provider, Url.Action("LinkLoginCallback", "Manage"), User.Identity.GetUserId());

}

//

// GET: /Manage/LinkLoginCallback

public async Task<ActionResult> LinkLoginCallback()

{

var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey, User.Identity.GetUserId());

if (loginInfo == null)

{

return RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });

}

var result = await UserManager.AddLoginAsync(User.Identity.GetUserId(), loginInfo.Login);

return result.Succeeded ? RedirectToAction("ManageLogins") : RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });

}

protected override void Dispose(bool disposing)

{

if (disposing && \_userManager != null)

{

\_userManager.Dispose();

\_userManager = null;

}

base.Dispose(disposing);

}

#region Вспомогательные приложения

// Используется для защиты от XSRF-атак при добавлении внешних имен входа

private const string XsrfKey = "XsrfId";

private IAuthenticationManager AuthenticationManager

{

get

{

return HttpContext.GetOwinContext().Authentication;

}

}

private void AddErrors(IdentityResult result)

{

foreach (var error in result.Errors)

{

ModelState.AddModelError("", error);

}

}

private bool HasPassword()

{

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

if (user != null)

{

return user.PasswordHash != null;

}

return false;

}

private bool HasPhoneNumber()

{

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

if (user != null)

{

return user.PhoneNumber != null;

}

return false;

}

public enum ManageMessageId

{

AddPhoneSuccess,

ChangePasswordSuccess,

SetTwoFactorSuccess,

SetPasswordSuccess,

RemoveLoginSuccess,

RemovePhoneSuccess,

Error

}

#endregion

}

}

**ManagerController.cs**

using Microsoft.AspNet.Identity;

using WebApplication4.DAO;

using WebApplication4.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication4.Controllers

{

public class ManagerController : Controller

{

private DAO\_category\_advertising category\_advertisingDAO = new DAO\_category\_advertising();

[Authorize(Roles = "Manager")]

[HttpGet]

public ActionResult Index()

{

return View(category\_advertisingDAO.GetAll());

}

[Authorize(Roles = "Manager")]

[HttpGet]

public ActionResult Create()

{

return View();

}

[Authorize(Roles = "Manager")]

[HttpPost]

public ActionResult Create(Category\_advertising st, string id)

{

if (category\_advertisingDAO.Add(st, id))

{

return RedirectToAction("Message", "Message", new { str = "Добавлена категория рекламы" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

[Authorize(Roles = "Manager")]

[HttpGet]

public ActionResult Delete(int id)

{

return View("Delete", category\_advertisingDAO.Get(id));

}

[Authorize(Roles = "Manager")]

[HttpPost]

public ActionResult Delete(int id, FormCollection collection)

{

if (category\_advertisingDAO.Delete(id))

{

return RedirectToAction("Message", "Message", new { str = "Категория рекламы успешно удалена" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

[Authorize(Roles = "Manager")]

[HttpGet]

public ActionResult Edit(int id)

{

return View();

}

[Authorize(Roles = "Manager")]

[HttpPost]

public ActionResult Edit(int id, Category\_advertising collection)

{

if (category\_advertisingDAO.Update(id, collection))

{

return RedirectToAction("Message", "Message", new { str = "Обновление прошло успешно" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

}

}

**MediabuinerController.cs**

using Microsoft.AspNet.Identity;

using WebApplication4.DAO;

using WebApplication4.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication4.Controllers

{

public class MediabuinerController : Controller

{

private DAO\_category\_advertising category\_advertisingDAO = new DAO\_category\_advertising();

private DAO\_mass\_media mass\_mediaDAO = new DAO\_mass\_media();

[Authorize(Roles = "Mediabuiner")]

[HttpGet]

public ActionResult Index()

{

return View(mass\_mediaDAO.GetAll());

}

[Authorize(Roles = "Mediabuiner")]

[HttpGet]

public ActionResult Create()

{

return View();

}

[Authorize(Roles = "Mediabuiner")]

[HttpPost]

public ActionResult Create(Mass\_media st, string id)

{

if (mass\_mediaDAO.Add(st, id))

{

return RedirectToAction("Message", "Message", new { str = "Средство массовой информации добавлено" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

[Authorize(Roles = "Mediabuiner")]

[HttpGet]

public ActionResult Index\_category\_advertising()

{

return View(category\_advertisingDAO.GetAll());

}

[Authorize(Roles = "Mediabuiner")]

[HttpGet]

public ActionResult Create\_id\_mass\_media()

{

return View();

}

[Authorize(Roles = "Mediabuiner")]

[HttpPost]

public ActionResult Create\_id\_mass\_media(Category\_advertising st, int id)

{

if (category\_advertisingDAO.AddMass\_media(st, id))

{

return RedirectToAction("Message", "Message", new { str = "Добавлено средство массовой информации в категорию рекламы" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

[Authorize(Roles = "Mediabuiner")]

[HttpGet]

public ActionResult Delete(int id)

{

return View("Delete", mass\_mediaDAO.Get(id));

}

[Authorize(Roles = "Mediabuiner")]

[HttpPost]

public ActionResult Delete(int id, FormCollection collection)

{

if (mass\_mediaDAO.Delete(id))

{

return RedirectToAction("Message", "Message", new { str = "Средство массовой информации успешно удалено" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

[Authorize(Roles = "Mediabuiner")]

[HttpGet]

public ActionResult Edit(int id)

{

return View();

}

[Authorize(Roles = "Mediabuiner")]

[HttpPost]

public ActionResult Edit(int id, Mass\_media collection)

{

if (mass\_mediaDAO.Update(id, collection))

{

return RedirectToAction("Message", "Message", new { str = "Обновление прошло успешно" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

}

}

**MessageController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication4.Controllers

{

public class MessageController : Controller

{

// GET: Message

public ActionResult Message(string str)

{

ViewBag.Message = str;

return View();

}

// GET: Error

public ActionResult Error(string str)

{

ViewBag.Message = str;

return View();

}

}

}

**OrderController.cs**

using Microsoft.AspNet.Identity;

using WebApplication4.DAO;

using WebApplication4.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace WebApplication4.Controllers

{

public class OrderController : Controller

{

private DAO\_order orderDAO = new DAO\_order();

[Authorize(Roles = "Client")]

[HttpGet]

public ActionResult IndexClient(string id)

{

return View(orderDAO.GetAllClient(id));

}

[Authorize(Roles = "Manager")]

[HttpGet]

public ActionResult IndexManager(string id)

{

return View(orderDAO.GetAllManager(id));

}

[Authorize(Roles = "Client")]

[HttpGet]

public ActionResult Create()

{

return View();

}

[Authorize(Roles = "Client")]

[HttpPost]

public ActionResult Create(Order obj, int id\_ca, string submitButton)

{

string id = User.Identity.GetUserId();

obj.Id\_category\_advertising = id\_ca;

switch (submitButton)

{

case "Отправить":

if (orderDAO.Add(obj, id, "Отправлен"))

{

return RedirectToAction("Message", "Message", new { str = "Заявка отправлена!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

case "Сохранить как черновик":

if (orderDAO.Add(obj, id, "Черновик"))

{

return RedirectToAction("Message", "Message", new { str = "Заказ сохранен как черновик!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

default:

return (View());

}

}

[Authorize(Roles = "Manager")]

[HttpGet]

public ActionResult Decline(int idAp)

{

if (orderDAO.Decline(idAp))

{

return RedirectToAction("Message", "Message", new { str = "Заказ отклонен!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

[Authorize(Roles = "Client")]

[HttpGet]

public ActionResult Edit(int id)

{

return View();

}

[Authorize(Roles = "Client")]

[HttpPost]

public ActionResult Edit(int id, Order collection)

{

collection.Id\_order = id;

if (orderDAO.Update(id, collection))

{

return RedirectToAction("Message", "Message", new { str = "Обновление прошло успешно!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

[Authorize(Roles = "Client")]

[HttpGet]

public ActionResult Delete(int id)

{

return View("Delete", orderDAO.Get(id));

}

[Authorize(Roles = "Client")]

[HttpPost]

public ActionResult Delete(int id, FormCollection collection)

{

if (orderDAO.Delete(id))

{

return RedirectToAction("Message", "Message", new { str = "Удаление прошло успешно!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

[Authorize(Roles = "Client")]

[HttpGet]

public ActionResult Send(int id)

{

return View("Send", orderDAO.Get(id));

}

[Authorize(Roles = "Client")]

[HttpPost]

public ActionResult Send(int id, FormCollection collection)

{

if (orderDAO.Send(id))

{

return RedirectToAction("Message", "Message", new { str = "Вы отправили заказ!" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

[Authorize(Roles = "Manager")]

[HttpGet]

public ActionResult CreateAnswer()

{

return View();

}

[Authorize(Roles = "Manager")]

[HttpPost]

public ActionResult CreateAnswer(Order obj, int idAp)

{

string id = User.Identity.GetUserId();

obj.Status = "Обработан";

if (orderDAO.AddAnswer(obj, id, idAp))

{

return RedirectToAction("Message", "Message", new { str = "Заказ обработан" });

}

else

{

return RedirectToAction("Error", "Message", new { str = "Произошла ошибка!" });

}

}

}

}

**Объекты доступа к данным(DAO)**

**DAO\_category\_advertising.cs**

using NLog;

using WebApplication4.Models;

using System.Collections.Generic;

using System.Linq;

namespace WebApplication4.DAO

{

public class DAO\_category\_advertising

{

private Entities1 \_entities = new Entities1();

private static Logger logger = LogManager.GetCurrentClassLogger();

public bool AddMass\_media(Category\_advertising st, int id)

{

try

{

var Entity = \_entities.Category\_advertising.FirstOrDefault(n => n.Id\_category\_advertising == id);

Entity.Id\_mass\_media = st.Id\_mass\_media;

\_entities.SaveChanges();

logger.Debug("Добавление масс-медиа");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

public Category\_advertising Get(int? id)

{

try

{

Category\_advertising item = \_entities.Category\_advertising.Where(n => n.Id\_category\_advertising == id).First();

logger.Debug("Получена категория рекламы");

return item;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public IEnumerable<Category\_advertising> GetAll()

{

try

{

IEnumerable<Category\_advertising> list = \_entities.Category\_advertising.Select(n => n);

logger.Debug("Получен список категорий рекламы");

return list;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public bool Add(Category\_advertising st, string id)

{

try

{

var currentUser = new Entities1().AspNetUsers.Where(n => n.Id.Equals(id)).FirstOrDefault();

var EntityManager = \_entities.Manager.FirstOrDefault(n => n.Id\_user == currentUser.Id);

st.Id\_manager = EntityManager.Id\_manager;

\_entities.Category\_advertising.Add(st);

\_entities.SaveChanges();

logger.Debug("Добавлена категория рекламы");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return false;

}

public bool Delete(int id)

{

try

{

Category\_advertising applicationToDelete = Get(id);

\_entities.Category\_advertising.Remove(applicationToDelete);

\_entities.SaveChanges();

logger.Debug("Категория рекламы удалена");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

public bool Update(int id, Category\_advertising st)

{

try

{

var Entity = \_entities.Category\_advertising.FirstOrDefault(n => n.Id\_category\_advertising == id);

Entity.Advertising\_period = st.Advertising\_period;

Entity.Visual\_aid = st.Visual\_aid;

\_entities.SaveChanges();

logger.Debug("Категория рекламы обнавлена");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

}

}

**DAO\_client.cs**

using NLog;

using WebApplication4.Models;

using System.Collections.Generic;

using System.Linq;

namespace WebApplication4.DAO

{

public class DAO\_client

{

private Entities1 \_entities = new Entities1();

private static Logger logger = LogManager.GetCurrentClassLogger();

public IEnumerable<Client> GetAll()

{

try

{

IEnumerable<Client> list = \_entities.Client.Select(n => n);

logger.Debug("Получение списка клиентов");

return list;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public bool Add(Client st, string id)

{

try

{

st.Id\_user = id;

\_entities.Client.Add(st);

\_entities.SaveChanges();

logger.Debug("Добавление информации клиента");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

public Client Get(string id)

{

try

{

var currentUser = new Entities1().AspNetUsers.Where(n => n.Id.Equals(id)).FirstOrDefault();

Client item = \_entities.Client.Where(n => n.Id\_user == currentUser.Id).First();

logger.Debug("Получение текущего клиента");

return item;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

}

}

**DAO\_manager.cs**

using NLog;

using WebApplication4.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace WebApplication4.DAO

{

public class DAO\_manager

{

private Entities1 \_entities = new Entities1();

private static Logger logger = LogManager.GetCurrentClassLogger();

public IEnumerable<Manager> GetAll()

{

try

{

IEnumerable<Manager> list = \_entities.Manager.Select(n => n);

logger.Debug("Получен список менеджеров");

return list;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public Manager Get(int id)

{

try

{

Manager item = \_entities.Manager.Where(n => n.Id\_manager == id).First();

logger.Debug("Получен менеджер");

return item;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public bool Add(Manager st, string id)

{

try

{

var currentUser = new Entities1().AspNetUsers.Where(n => n.Id.Equals(id)).FirstOrDefault();

st.Id\_user = currentUser.Id;

st = \_entities.Manager.Add(st);

\_entities.SaveChanges();

logger.Debug("Добавлен");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

}

}

**DAO\_mass\_media.cs**

using NLog;

using WebApplication4.Models;

using System.Collections.Generic;

using System.Linq;

namespace WebApplication4.DAO

{

public class DAO\_mass\_media

{

private Entities1 \_entities = new Entities1();

private static Logger logger = LogManager.GetCurrentClassLogger();

public Mass\_media Get(int id)

{

try

{

Mass\_media item = \_entities.Mass\_media.Where(n => n.Id\_mass\_media == id).First();

logger.Debug("Получено стедство массовой информации");

return item;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public IEnumerable<Mass\_media> GetAll()

{

try

{

IEnumerable<Mass\_media> list = \_entities.Mass\_media.Select(n => n);

logger.Debug("Получен список массовой информации");

return list;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public bool Add(Mass\_media st, string id)

{

try

{

var currentUser = new Entities1().AspNetUsers.Where(n => n.Id.Equals(id)).FirstOrDefault();

var EntityMediabuiner = \_entities.Mediabuiner.FirstOrDefault(n => n.Id\_user == currentUser.Id);

st.Id\_mediabuiner = EntityMediabuiner.Id\_mediabuiner;

\_entities.Mass\_media.Add(st);

\_entities.SaveChanges();

logger.Debug("Добавлено стредство массовой информации");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return false;

}

public bool Delete(int id)

{

try

{

Mass\_media applicationToDelete = Get(id);

\_entities.Mass\_media.Remove(applicationToDelete);

\_entities.SaveChanges();

logger.Debug("Средство массовой информации удалено");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

public bool Update(int id, Mass\_media st)

{

try

{

var Entity = \_entities.Mass\_media.FirstOrDefault(n => n.Id\_mass\_media == id);

Entity.Name\_mass\_media = st.Name\_mass\_media;

Entity.Subject\_area = st.Subject\_area;

Entity.Price = st.Price;

\_entities.SaveChanges();

logger.Debug("Средство массовой информации обнавлено");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

}

}

**DAO\_mediabuiner.cs**

using NLog;

using WebApplication4.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace WebApplication4.DAO

{

public class DAO\_mediabuiner

{

private Entities1 \_entities = new Entities1();

private static Logger logger = LogManager.GetCurrentClassLogger();

public IEnumerable<Mediabuiner> GetAll()

{

try

{

IEnumerable<Mediabuiner> list = \_entities.Mediabuiner.Select(n => n);

logger.Debug("Получен список медиабайнеров");

return list;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public Mediabuiner Get(int id)

{

try

{

Mediabuiner item = \_entities.Mediabuiner.Where(n => n.Id\_mediabuiner == id).First();

logger.Debug("Получен медиабайнер");

return item;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public bool Add(Mediabuiner st, string id)

{

try

{

var currentUser = new Entities1().AspNetUsers.Where(n => n.Id.Equals(id)).FirstOrDefault();

st.Id\_user = currentUser.Id;

st = \_entities.Mediabuiner.Add(st);

\_entities.SaveChanges();

logger.Debug("Добавлен");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

}

}

**DAO\_order.cs**

using NLog;

using WebApplication4.Models;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System;

using System.Data.Entity.Validation;

namespace WebApplication4.DAO

{

public class DAO\_order

{

private Entities1 \_entities = new Entities1();

private static Logger logger = LogManager.GetCurrentClassLogger();

public IEnumerable<Order> GetAllClient(string id)

{

try

{

var currentUser = new Entities1().AspNetUsers.Where(n => n.Id.Equals(id)).FirstOrDefault();

var EntityClient = \_entities.Client.FirstOrDefault(n => n.Id\_user == currentUser.Id);

IEnumerable<Order> list = \_entities.Order.Where(n => n.Id\_client == EntityClient.Id\_client).Select(n => n);

logger.Debug("Получен список заказов");

return list;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public IEnumerable<Order> GetAllManager(string id)

{

try

{

IEnumerable<Order> list = \_entities.Order.Select(n => n);

logger.Debug("Получен список заказов");

return list;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public bool Add( Order Entity, string id, string status)

{

try

{

Entity.Status = status;

Entity.Date = DateTime.Now;

var currentUser = new Entities1().AspNetUsers.Where(n => n.Id.Equals(id)).FirstOrDefault();

var EntityClient = \_entities.Client.FirstOrDefault(n => n.Id\_user == currentUser.Id);

Entity.Id\_client = EntityClient.Id\_client;

\_entities.Order.Add(Entity);

\_entities.SaveChanges();

logger.Debug("Заказ добавлен");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

public bool Decline(int idAp)

{

try

{

var EntityApplication = \_entities.Order.FirstOrDefault(n => n.Id\_order == idAp);

string st = "Черновик";

EntityApplication.Status = st;

\_entities.SaveChanges();

logger.Debug("Изменен статус заказа");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

public Order Get(int id)

{

try

{

Order item = \_entities.Order.Where(n => n.Id\_order == id).First();

logger.Debug("Получен заказ");

return item;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public bool Update(int id, Order st)

{

try

{

var Entity = \_entities.Order.FirstOrDefault(n => n.Id\_order == id);

Entity.Product\_name = st.Product\_name;

Entity.Description\_wish = st.Description\_wish;

\_entities.SaveChanges();

logger.Debug("Заказ обновлен");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

public bool Delete(int id)

{

try

{

Order applicationToDelete = Get(id);

\_entities.Order.Remove(applicationToDelete);

\_entities.SaveChanges();

logger.Debug("Заказ удален");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

public bool Send(int id)

{

try

{

var Entity = \_entities.Order.FirstOrDefault(n => n.Id\_order == id);

if (new DAO\_category\_advertising().Get(Entity.Id\_category\_advertising) == null) return false;

Entity.Date = DateTime.Now;

Entity.Status = "Отправлен";

\_entities.SaveChanges();

logger.Debug("Изменен статус заказа");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

public bool AddAnswer(Order st, string id, int idAp)

{

try

{

var EntityOrder = \_entities.Order.FirstOrDefault(n => n.Id\_order == idAp);

var currentUser = new Entities1().AspNetUsers.Where(n => n.Id.Equals(id)).FirstOrDefault();

var EntityManager = \_entities.Manager.FirstOrDefault(n => n.Id\_user == currentUser.Id);

EntityOrder.Id\_manager = EntityManager.Id\_manager;

EntityOrder.Status = "Обработан";

EntityOrder.Temp\_of\_delivery = st.Temp\_of\_delivery;

EntityOrder.Date\_of\_withdrawal\_of\_adertising\_turnover = st.Date\_of\_withdrawal\_of\_adertising\_turnover;

\_entities.SaveChanges();

logger.Debug("Заказ обработан");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

}

}

**DAO\_user.cs**

using WebApplication4.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using NLog;

namespace WebApplication4.DAO

{

public class DAO\_user

{

private Entities1 \_entities = new Entities1();

private static Logger logger = LogManager.GetCurrentClassLogger();

public IEnumerable<AspNetUsers> GetAll()

{

try

{

IEnumerable<AspNetUsers> list = \_entities.AspNetUsers.Select(n => n);

logger.Debug("Получение списка пользователей");

return list;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public AspNetUsers Get(string id)

{

try

{

AspNetUsers item = \_entities.AspNetUsers.Where(n => n.Id == id).First();

logger.Debug("Получение пользователя");

return item;

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

}

return null;

}

public bool UpdateRoles(string id, AspNetRoles st)

{

try

{

var Entity = \_entities.AspNetUsers.FirstOrDefault(n => n.Id == id);

var EntityRoles = \_entities.AspNetRoles.FirstOrDefault(n => n.Name == st.Name);

Entity.AspNetRoles.Add(EntityRoles);

\_entities.SaveChanges();

logger.Debug("Обновление роли пользователя");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

public bool DeleteRoles(string id, AspNetRoles st)

{

try

{

var Entity = \_entities.AspNetUsers.FirstOrDefault(n => n.Id == id);

var EntityRoles = \_entities.AspNetRoles.FirstOrDefault(n => n.Name == st.Name);

if (Entity.AspNetRoles.Contains(EntityRoles))

{

Entity.AspNetRoles.Remove(EntityRoles);

\_entities.SaveChanges();

}

logger.Debug("Удаление роли пользователя");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

public bool Delete(string id)

{

try

{

AspNetUsers originalMachines = Get(id);

\_entities.AspNetUsers.Remove(originalMachines);

\_entities.SaveChanges();

logger.Debug("Удаление пользователя");

}

catch (System.ServiceModel.CommunicationException ex)

{

logger.Error("Ошибка: ", ex);

return false;

}

return true;

}

}

}

ПРИЛОЖЕНИЕ Б Структура и Скрипты создания таблиц БД

\_**\_MigrationHistory**

CREATE TABLE [dbo].[\_\_MigrationHistory] (

[MigrationId] NVARCHAR (150) NOT NULL,

[ContextKey] NVARCHAR (300) NOT NULL,

[Model] VARBINARY (MAX) NOT NULL,

[ProductVersion] NVARCHAR (32) NOT NULL,

CONSTRAINT [PK\_dbo.\_\_MigrationHistory] PRIMARY KEY CLUSTERED ([MigrationId] ASC, [ContextKey] ASC)

);

**AspNetRoles**

CREATE TABLE [dbo].[AspNetRoles] (

[Id] NVARCHAR (128) NOT NULL,

[Name] NVARCHAR (256) NOT NULL,

CONSTRAINT [PK\_dbo.AspNetRoles] PRIMARY KEY CLUSTERED ([Id] ASC)

);

GO

CREATE UNIQUE NONCLUSTERED INDEX [RoleNameIndex]

ON [dbo].[AspNetRoles]([Name] ASC);

**AspNetUserClaims //**не используется

CREATE TABLE [dbo].[AspNetUserClaims] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[UserId] NVARCHAR (128) NOT NULL,

[ClaimType] NVARCHAR (MAX) NULL,

[ClaimValue] NVARCHAR (MAX) NULL,

CONSTRAINT [PK\_dbo.AspNetUserClaims] PRIMARY KEY CLUSTERED ([Id] ASC),

CONSTRAINT [FK\_dbo.AspNetUserClaims\_dbo.AspNetUsers\_UserId] FOREIGN KEY ([UserId]) REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE

);

GO

CREATE NONCLUSTERED INDEX [IX\_UserId]

ON [dbo].[AspNetUserClaims]([UserId] ASC);

**AspNetUserLogins //**не используется

CREATE TABLE [dbo].[AspNetUserLogins] (

[LoginProvider] NVARCHAR (128) NOT NULL,

[ProviderKey] NVARCHAR (128) NOT NULL,

[UserId] NVARCHAR (128) NOT NULL,

CONSTRAINT [PK\_dbo.AspNetUserLogins] PRIMARY KEY CLUSTERED ([LoginProvider] ASC, [ProviderKey] ASC, [UserId] ASC),

CONSTRAINT [FK\_dbo.AspNetUserLogins\_dbo.AspNetUsers\_UserId] FOREIGN KEY ([UserId]) REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE

);

GO

CREATE NONCLUSTERED INDEX [IX\_UserId]

ON [dbo].[AspNetUserLogins]([UserId] ASC);

**AspNetUserRoles**

CREATE TABLE [dbo].[AspNetUserRoles] (

[UserId] NVARCHAR (128) NOT NULL,

[RoleId] NVARCHAR (128) NOT NULL,

CONSTRAINT [PK\_dbo.AspNetUserRoles] PRIMARY KEY CLUSTERED ([UserId] ASC, [RoleId] ASC),

CONSTRAINT [FK\_dbo.AspNetUserRoles\_dbo.AspNetRoles\_RoleId] FOREIGN KEY ([RoleId]) REFERENCES [dbo].[AspNetRoles] ([Id]) ON DELETE CASCADE,

CONSTRAINT [FK\_dbo.AspNetUserRoles\_dbo.AspNetUsers\_UserId] FOREIGN KEY ([UserId]) REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE

);

GO

CREATE NONCLUSTERED INDEX [IX\_UserId]

ON [dbo].[AspNetUserRoles]([UserId] ASC);

**AspNetUsers**

CREATE TABLE [dbo].[AspNetUsers] (

[Id] NVARCHAR (128) NOT NULL,

[Email] NVARCHAR (256) NULL,

[EmailConfirmed] BIT NOT NULL,

[PasswordHash] NVARCHAR (MAX) NULL,

[SecurityStamp] NVARCHAR (MAX) NULL,

[PhoneNumber] NVARCHAR (MAX) NULL,

[PhoneNumberConfirmed] BIT NOT NULL,

[TwoFactorEnabled] BIT NOT NULL,

[LockoutEndDateUtc] DATETIME NULL,

[LockoutEnabled] BIT NOT NULL,

[AccessFailedCount] INT NOT NULL,

[UserName] NVARCHAR (256) NOT NULL,

[Name] NVARCHAR (MAX) NULL,

[Surname] NVARCHAR (MAX) NULL,

[Patronymic] NVARCHAR (MAX) NULL,

CONSTRAINT [PK\_dbo.AspNetUsers] PRIMARY KEY CLUSTERED ([Id] ASC)

);

**Category\_advertising**

CREATE TABLE [dbo].[Category\_advertising] (

[Id\_category\_advertising] INT IDENTITY (1, 1) NOT NULL,

[Id\_manager] INT NOT NULL,

[Id\_mass\_media] INT NULL,

[Advertising\_period] NCHAR (20) NOT NULL,

[Visual\_aid] NCHAR (10) NOT NULL,

PRIMARY KEY CLUSTERED ([Id\_category\_advertising] ASC),

FOREIGN KEY ([Id\_mass\_media]) REFERENCES [dbo].[Mass\_media] ([Id\_mass\_media]),

FOREIGN KEY ([Id\_manager]) REFERENCES [dbo].[Manager] ([Id\_manager])

);

**Client**

CREATE TABLE [dbo].[Client] (

[Id\_client] INT IDENTITY (1, 1) NOT NULL,

[Passport] NVARCHAR (50) NOT NULL,

[Address] NVARCHAR (50) NULL,

[Id\_user] NVARCHAR (128) NOT NULL,

PRIMARY KEY CLUSTERED ([Id\_client] ASC),

FOREIGN KEY ([Id\_user]) REFERENCES [dbo].[AspNetUsers] ([Id])

);

**Manager**

CREATE TABLE [dbo].[Manager] (

[Id\_manager] INT IDENTITY (1, 1) NOT NULL,

[Number\_of\_processed\_clients] INT NOT NULL,

[Work\_experience] INT NOT NULL,

[Id\_user] NVARCHAR (128) NOT NULL,

PRIMARY KEY CLUSTERED ([Id\_manager] ASC),

FOREIGN KEY ([Id\_user]) REFERENCES [dbo].[AspNetUsers] ([Id])

);

**Mass\_media**

CREATE TABLE [dbo].[Mass\_media] (

[Id\_mass\_media] INT IDENTITY (1, 1) NOT NULL,

[Id\_mediabuiner] INT NOT NULL,

[Name\_mass\_media] NCHAR (30) NOT NULL,

[Subject\_area] NCHAR (30) NOT NULL,

[Price] INT NOT NULL,

PRIMARY KEY CLUSTERED ([Id\_mass\_media] ASC),

FOREIGN KEY ([Id\_mediabuiner]) REFERENCES [dbo].[Mediabuiner] ([Id\_mediabuiner])

);

**Mediabuiner**

CREATE TABLE [dbo].[Mediabuiner] (

[Id\_mediabuiner] INT IDENTITY (1, 1) NOT NULL,

[Work\_experience] INT NOT NULL,

[Id\_user] NVARCHAR (128) NOT NULL,

PRIMARY KEY CLUSTERED ([Id\_mediabuiner] ASC),

FOREIGN KEY ([Id\_user]) REFERENCES [dbo].[AspNetUsers] ([Id])

);

**Order**

CREATE TABLE [dbo].[Order] (

[Id\_order] INT IDENTITY (1, 1) NOT NULL,

[Id\_manager] INT NULL,

[Id\_client] INT NOT NULL,

[Id\_category\_advertising] INT NOT NULL,

[Product\_name] NVARCHAR (20) NOT NULL,

[Description\_wish] NVARCHAR (MAX) NOT NULL,

[Status] NVARCHAR (45) NOT NULL,

[Date] DATE NOT NULL,

[Temp\_of\_delivery] NVARCHAR (50) NULL,

[Date\_of\_withdrawal\_of\_adertising\_turnover] NVARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([Id\_order] ASC),

FOREIGN KEY ([Id\_category\_advertising]) REFERENCES [dbo].[Category\_advertising] ([Id\_category\_advertising]),

FOREIGN KEY ([Id\_client]) REFERENCES [dbo].[Client] ([Id\_client]),

FOREIGN KEY ([Id\_manager]) REFERENCES [dbo].[Manager] ([Id\_manager])

);

ПРИЛОЖЕНИЕ С ТЕсты

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Web.Mvc;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using WebApplication4;

using WebApplication4.Controllers;

using WebApplication4.DAO;

namespace WebApplication4.Tests.Controllers

{

[TestClass]

public class ControllerTest

{

[TestMethod]

public void IndexCategoryAdvertisingGetAllIndexManagerNotNull()//Возвращает ли список категорий рекламы

{

ManagerController controller = new ManagerController();

ViewResult result = controller.Index() as ViewResult;

Assert.IsNotNull(result);

}

[TestMethod]

public void IndexMassMediaGetAllIndexMediabuinerNotNull()//Возвращает ли список СМИ

{

MediabuinerController controller = new MediabuinerController();

ViewResult result = controller.Index() as ViewResult;

Assert.IsNotNull(result);

}

[TestMethod]

public void IndexUserGetAllIndexAdministratorNotNull()//Возвращает ли список пользователей

{

ListUserController controller = new ListUserController();

ViewResult result = controller.Index() as ViewResult;

Assert.IsNotNull(result);

}

}

}