Допущен к защите:

Старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ *Дипломный проект защищен:*

*(подпись) с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*«\_\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016года*

*«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2016 г од Зав.учебной частью \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись)*

**ДИПЛОМНИЙ ПРОЕКТ**

##### КА «Шаг» с15пз3. ПСТ. 12/2013.

*Тема:* ***Веб интерфейс для работы с базой данных модельного агентства***

*(на русском языке)*

*Тема:* ***The web interface to work with the database modeling Agency***

*(на английском языке)*

**Слушатель** \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ Светличный Сергей Алексеевич (подпись) (дата) (Ф.И.О на русском языке)

*Svitlychnyi Serhii*

*(Ф.И.О на английском языке)*

***Руководитель***  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ Василенко Игорь Григорьевич*

*(подпись) (дата) (Ф.И.О)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(рекомендуемая оценка)*

***Экзаменнационная комиссия по защите дипломных проектов:***

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись) (дата) (Ф.И.О)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись) (дата) (Ф.И.О)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись) (дата) (Ф.И.О)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись) (дата) (Ф.И.О)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

### ЗАДАНИЕ И КАЛЕНДАРНЫЙ ПЛАН

по выполнению дипломной работы

|  |  |  |
| --- | --- | --- |
| студентом группы |  |  |
|  | | |
| (Фамилия, Имя, Отчество) | | |

|  |  |
| --- | --- |
| Тема работы: |  |
|  | |
|  | |

Утверждена приказом директора филиала от \_\_\_\_\_\_\_\_\_\_\_\_\_ № \_\_\_\_

Срок сдачи студентом законченной работы:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Объем дипломной работы: 50-80 машинописных страниц.

Перечень прилагаемого материала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

|  |
| --- |
| Организации, на базе которых выполняется дипломная работа: |
| а) |
| б) |
| в) |

## Календарный план выполнения дипломной работы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование этапа работ | Срок выполнения | | Оценка выполненного этапа работы | Роспись руководителя |
| по плану | фактически |
| Написание введения |  |  |  |  |
| Написание главы 1 |  |  |  |  |
| Написание главы 2 |  |  |  |  |
| Написание главы 3 |  |  |  |  |
| Написание заключения |  |  |  |  |
| Оформление дипломной работы |  |  |  |  |
| Получение отзыва руководителя |  |  |  |  |
| Подготовка доклада и наглядных пособий |  |  |  |  |
| Защита дипломной работы |  |  |  |  |

УТВЕРЖДАЮ

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_ 201\_\_ г.

Ст.преподаватель\_\_\_\_\_\_\_\_\_\_ Задание принял к исполнению:

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ……. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лист замечаний**

|  |  |
| --- | --- |
| Ф.И.О. | Суть замечаний, оценка и подпись |
|  |  |

Содержание

Введение

Глава 1. Постановка и анализ задачи

1.1 Исследование предметов цели иособенности веб интерфейса

1.2 Требования к приложению

1.3 Технологии и средства разработки

1.4 Список используемых технологий

1.5 Обзор используемых технологий

Глава 2 Проектирование предметной области

2.1 Проектирование структуры сайта

2.2 Проектирование базы данных

2.2.1 Постановка задачи

2.2.2 Цели разработки базы данных

2.2.3 Пользовательские требования

2.2.4 Структура базы данных

Глава 3. Проектирование приложения и решение поставленных задач

3.1 Архитектура приложения

3.2 Алгоритм фильтрации данных

Заключение

Список литературы

**ВВЕДЕНИЕ**

Невозможно представить современный мир без компьютера, и в частности, без интернета. С каждым годом количество пользователей неуклонно растет. И это неудивительно, ведь интернет дает нам обилие возможностей. Одним из главных его достоинств является доступность огромного количества информации. Не выходя из дома, имея только персональный компьютер и доступ в интернет, можно совершать путешествия по другим странам, изучать историю, читать интересующие книги, смотреть кино, рассматривать миллионы фотографий! Но это не все, что может дать нам всемирная сеть, она также очень упрощает нашу жизнь. Теперь не нужно стоять в очередях, чтобы купить билет или оплатить коммунальные платежи, все это можно быстро и безопасно сделать прямо со своего рабочего места.

Естественно, что для осуществления электронной коммерции, необходимо современное программное обеспечение. Сегодня существует множество технологий, позволяющих создавать надёжные и эффективные web-приложения отвечающие самым современным требованиям, удобные в использовании, как со стороны продавца, так и со стороны покупателя.

Данная работа посвящена задачи проектирования и развития веб интерфейса для работы с базой данных модельного агентства.

Первый раздел содержит постановку и анализ решаемой задачи, а именно включает в себя исследование предмета, целей, особенностей и обзор технологий применяемых в ходе построения приложения.

Второй раздел посвящён проектированию сайта. Подробно рассматриваются такие вопросы, как проектирование таблиц базы данных, связи между ними. Касательно проекта веб приложения затрагивается проектирование слоев.

**Глава 1. Постановка и анализ задачи**

**1.1 Исследование предметов цели и особенности веб интерфейса**

Для любого пользователя веб интерфейса главными преимуществами являются скорость загрузки и простота навигации. К сожалению, иногда бывает сложно разобраться в запутанной иерархии сайта и найти нужную информацию. Ещё одним неприятным явлением иногда бывает плохой дизайн - слишком мелкий текст или яркие, кричащие цвета, от которых устают глаза.

Поэтому главной задачей является создание сайта, в котором заказчик мог бы с легкостью ориентироваться, получать фото интересующего кандидата и его персональные данные, осуществить поиск кандидатов по различным критериям, добавлять кандидатов в свою корзину и сохранение отобранных вакансий.

**1.2 Требования к приложению**

Разрабатываемая система должна удовлетворять следующим требованиям:

* + - * Иметь систему регистрации пользователей;
* Иметь систему аутентификации пользователей;
* Наличие расширенной системы поиска. Поиск по полу, возрасту, росту, роду деятельности;
* Предоставлять потенциальному покупателю всю необходимую информацию о кандидате;
* Иметь удобную и быструю систему отбора кандидата
* Возможность просмотра истории заказов;

**1.3 Технологии и средства разработки**

С развитием Интернета появилось большое количество средств разработки веб-приложений. Выбор средства для разработки - один из главных этапов при создании проекта. Для построения приложения была выбрана технология ASP .NET MVC по рядку причин. Во-первых, выбор этой технологии обусловлен возможностью легкой адаптации в будущем таких проектов к развертыванию в облаке. Во-вторых, платформа .Net позволяет, используя один язык программирования, такой, как например C#, создавать и веб-приложения, и десктопные программы, и службы, и сервисы. В-третьих, платформа .Net, весьма удобна в плане использования объектно-ориентированного программирования. В четвертых, паттерн MVC в настоящее время хорошо зарекомендовал себя как архитектурное решение.

**1.4 Список используемых технологий**

В процессе разработки использовались следующие технологии для реализации:

Основные технологии:

* [программная платформа](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0#.D0.BA.D1.80.D0.BE.D1.81.D1.81.D0.BF.D0.BB.D0.B0.D1.82.D1.84.D0.BE.D1.80.D0.BC.D0.B5.D0.BD.D0.BD.D0.BE.D0.B5_.D0.BF.D1.80.D0.BE.D0.B3.D1.80.D0.B0.D0.BC.D0.BC.D0.BD.D0.BE.D0.B5_.D0.BE.D0.B1.D0.B5.D1.81.D0.BF.D0.B5.D1.87.D0.B5.D0.BD.D0.B8.D0.B5) .NET Framework 4.5
* язык программирования C# 5.0
* технология создания [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5" \o "Веб-приложение) и [веб-сервисов](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81" \o "Веб-сервис) ASP.NET MVC 5.0
* среда разработки Microsoft Visual Studio 2015
* Microsoft SQL Server 2012

Вспомогательные технологии:

* Microsoft ASP.NET Identity
* Ninject
* Entity Framework 6.0.0
* CSS3
* HTML 5
* JQuery

**1.5 Обзор используемых технологий**

* [Программная платформа](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0#.D0.BA.D1.80.D0.BE.D1.81.D1.81.D0.BF.D0.BB.D0.B0.D1.82.D1.84.D0.BE.D1.80.D0.BC.D0.B5.D0.BD.D0.BD.D0.BE.D0.B5_.D0.BF.D1.80.D0.BE.D0.B3.D1.80.D0.B0.D0.BC.D0.BC.D0.BD.D0.BE.D0.B5_.D0.BE.D0.B1.D0.B5.D1.81.D0.BF.D0.B5.D1.87.D0.B5.D0.BD.D0.B8.D0.B5) .NET Framework 4.5

.NET Framework — [программная платформа](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0#.D0.BA.D1.80.D0.BE.D1.81.D1.81.D0.BF.D0.BB.D0.B0.D1.82.D1.84.D0.BE.D1.80.D0.BC.D0.B5.D0.BD.D0.BD.D0.BE.D0.B5_.D0.BF.D1.80.D0.BE.D0.B3.D1.80.D0.B0.D0.BC.D0.BC.D0.BD.D0.BE.D0.B5_.D0.BE.D0.B1.D0.B5.D1.81.D0.BF.D0.B5.D1.87.D0.B5.D0.BD.D0.B8.D0.B5), выпущенная компанией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft) в [2002 году](https://ru.wikipedia.org/wiki/2002_%D0%B3%D0%BE%D0%B4). Основой платформы является общеязыковая среда исполнения [Common Language Runtime (CLR)](https://ru.wikipedia.org/wiki/Common_Language_Runtime" \o "Common Language Runtime), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

NET Framework состоит из двух основных компонентов

Common Language Runtime (CLR) - это общеязыковая среда выполнения, в которой функционируют приложения NET. Это виртуальная машина.

Base Class Library (BCL) - набор библиотек классов .NET Framework. Все языки программирования в среде NET используют эту библиотеку.

* язык программирования C# 5.0

C# (произносится «си шарп») — [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Разработан в [1998](https://ru.wikipedia.org/wiki/1998)—[2001 годах](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) группой инженеров под руководством [Андерса Хейлсберга](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D0%B9%D0%BB%D1%81%D0%B1%D0%B5%D1%80%D0%B3,_%D0%90%D0%BD%D0%B4%D0%B5%D1%80%D1%81) в компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft) как язык разработки приложений для платформы [Microsoft .NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework" \o ".NET Framework).

Это относительно новый язык программирования, который характеризуется двумя следующими преимуществами:

- C# спроектирован и разработан специально для применения с Microsoft .NET Framework (развитой платформой разработки, развертывания и выполнения распределенных приложений).

- C# — язык, основанный на современной объектно-ориентированной методологии проектирования, при разработке которого специалисты из Microsoft опирались на опыт создания подобных языков, построенных в соответствии с предложенными около 20 лет назад объектно-ориентированными принципами.

* Среда разработки Microsoft Visual Studio 2015

Продукт от компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft), включающих [интегрированную среду разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) программного обеспечения и ряд других инструментальных средств. Данный продукт позволяет разрабатывать как [консольные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) [приложения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), так и приложения с [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), в том числе с поддержкой технологии [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms" \o "Windows Forms), а также [веб-сайты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82" \o "Сайт), [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5" \o "Веб-приложение), [веб-службы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0" \o "Веб-служба) как в [родном](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), так и в [управляемом](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) кодах для всех платформ, поддерживаемых [Windows](https://ru.wikipedia.org/wiki/Windows" \o "Windows), [Windows Mobile](https://ru.wikipedia.org/wiki/Windows_Mobile" \o "Windows Mobile), [Windows CE](https://ru.wikipedia.org/wiki/Windows_CE" \o "Windows CE), [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework), [Xbox](https://ru.wikipedia.org/wiki/Xbox" \o "Xbox), [Windows Phone](https://ru.wikipedia.org/wiki/Windows_Phone" \o "Windows Phone) [.NET Compact Framework](https://ru.wikipedia.org/wiki/.NET_Compact_Framework) и[Silverlight](https://ru.wikipedia.org/wiki/Silverlight).

Visual Studio включает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) с поддержкой технологии [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense" \o "IntelliSense) и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и дизайнер [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Visual Studio позволяет создавать и подключать сторонние дополнения ([плагины](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD" \o "Плагин)) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем [контроля версий исходного кода](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8) (как, например, [Subversion](https://ru.wikipedia.org/wiki/Subversion" \o "Subversion) и [Visual SourceSafe](https://ru.wikipedia.org/wiki/Microsoft_Visual_SourceSafe" \o "Microsoft Visual SourceSafe)), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на [предметно-ориентированных языках программирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)) или инструментов для прочих аспектов [процесса разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) (например, клиент Team Explorer для работы с [Team Foundation Server](https://ru.wikipedia.org/wiki/Team_Foundation_Server" \o "Team Foundation Server)).

* технология создания [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5" \o "Веб-приложение) и [веб-сервисов](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81" \o "Веб-сервис) ASP.NET MVC 5.0

Платформа ASP.NET MVC представляет собой фреймворк для создания сайтов и веб-приложений с помощью реализации паттерна MVC.

Концепция паттерна (шаблона) MVC (model - view - controller) предполагает разделение приложения на три компонента:

Контроллер (controller) представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления.

Представление (view) - это собственно визуальная часть или пользовательский интерфейс приложения. Как правило, html-страница, которую пользователь видит, зайдя на сайт.

Модель (model) представляет класс, описывающий логику используемых данных.

Общая схема взаимодействия этих компонентов представлена на рис. 1.

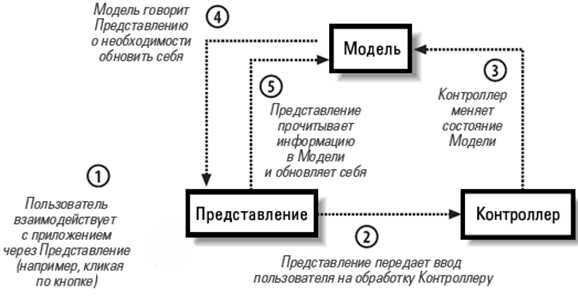


Рис. 1 - Схема взаимодействия компонентов паттерна MVC

Преимущества ASP.NET MVC:

* Концепция разделения. ASP.NET MVC поддерживает конструкцию, в которой пользовательский интерфейс (представление) сохраняет свою изолированность от кода, управляющего им (контроллера). При хорошей реализации это означает, что разработчикам легче передвигаться по приложениям, а также такая возможность упрощает процесс сохранения приложения – то, что вы внесете изменения в контроллер, еще не означает, что вам придется изменять пользовательский интерфейс.
* Тестируемость. С помощью отделения логики приложения от его пользовательского интерфейса ASP.NET MVC упрощает тестирование отдельных компонентов приложения. Классы контроллеров могут быть протестированы без тестирования реального пользовательского интерфейса. Контроллеры не имеют прямой зависимости от нетестируемого класса HttpContext, а вместо этого полагаются на абстракцию, что упрощает процесс написания автоматизированных тестов.
* Web-разработка на ASP.NET. Способ использования классов в .NET Framework в ASP.NET ничем не отличаемся от способа их применения в любом другом типе приложения .NET. Это значит, что, изучив программирование стандартных Windows-приложений, мы сможем применять большую часть полученных знаний и в программировании Web. Приложения ASP.NET всегда компилируются. В действительности каждый раз проходят два этапа компиляции. На первом этапе написанный нами код С# компилируется в код промежуточного языка под названием Microsoft Intermediate Language (MSIL), или просто IL. Этот первый шаг является фундаментальной причиной взаимозависимости .NET от языков. По сути, все языки .NET (включая С#, Visual Basic и многие другие) компилируются в фактически идентичный код IL. Этот первый этап компиляции может произойти автоматически при первом запросе страницы, или же его можно выполнить заранее (этот процесс известен как предварительная компиляция). Скомпилированный файл с кодом IL является сборкой. Второй этап компиляции наступает непосредственно перед фактическим выполнением страницы. На этом этапе код IL компилируется в низкоуровневый собственный машинный код. Этот этап известен как оперативная компиляция "точно к нужному моменту" (Just-In-Time — JIT) и он проходит одинаково для всех приложений .NET. Вся среда .NET Framework — то есть все пространства имен, приложения и классы - называется управляемой. Это значит, что происходят следующие процессы:
* Автоматическое управление памятью и сборка мусора.
* Безопасность типов. При компиляции приложения .NET добавляет к сборке информацию о доступных классах, их членах, типах данных и т.д.
* Расширяемые метаданные. Информация о классах и элементах является лишь одним из типов метаданных, хранимых .NET в скомпилированной сборке.
* Структурированная обработка ошибок.
* Многопоточностъ. .NET Framework предоставляет пул потоков, которые могут использоваться различными классами. Например, можно асинхронно вызывать методы, читать файлы либо взаимодействовать с Web-службами без необходимости явного создания новых потоков.
* Объектная-ориентированность. ASP.NET является объектно-ориентированной средой. Ваш код не только имеет полный доступ ко всем объектам в .NET Framework, возможность эксплуатировать все условные обозначения объектно-ориентированного программирования (ООП). Например, можно создавать повторно используемые классы, стандартизовать код в соответствии с интерфейсами и объединять полезные функции в распределяемом скомпилированном компоненте.
* Microsoft SQL Server 2012

Microsoft SQL Server — [система управления реляционными базами данных (РСУБД)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94), разработанная корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft). Основной используемый язык запросов — [Transact-SQL](https://ru.wikipedia.org/wiki/Transact-SQL" \o "Transact-SQL), создан совместно Microsoft и [Sybase](https://ru.wikipedia.org/wiki/Sybase). Transact-SQL является реализацией стандарта [ANSI](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2)/[ISO](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%BE_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8) по структурированному языку запросов ([SQL](https://ru.wikipedia.org/wiki/SQL)) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия.

* Microsoft ASP.NET Identity

Релиз ASP.NET MVC 5 ознаменовался выходом новой системой авторизации и аутентификации в .NET приложениях под названием ASP.NET Identity. Эта система пришла на смену провайдерам Simple Membership, которые были введены в ASP.NET MVC 4. Основные преимущества использования ASP.NET Identity Framework 2.0

Использование единой системы авторизации как для MVC проекта, так и для WebAPI. Более того, можно на одном сайте (домене) настроить авторизации в зависимости от URL. Так например если все вызовы WebAPI начинаются с /api - для этой части сайта можно настроить проверку авторизации на основе токенов. Тогда как для основного сайта удобнее будет использовать Cookies;

Простота подключения. Спорный момент, но единожды осознав как работает система подключать ее стало довольно просто. Роли и авторизация на основе утверждений (Claims), Возможность авторизации через социальные сети;

* Ninject

Ninject представляет собой легкий каркас для внедрения зависимостей. NET приложений. Это поможет вам разделить ваше приложение в коллекцию слабосвязанных, высоко сплоченной куски, а затем склеить их вместе на гибкой основе. С помощью Ninject поддерживать архитектуру программного обеспечения вашего, ваш код станет проще писать, повторное использование, тестировать и модифицировать.

Ninject является:

1. **Сосредоточенным.** Слишком много существующих проектов внедрения зависимостей принести в жертву удобство и простоту использования для функций, которые не часто необходимы. Каждый раз , когда функция добавляется к Ninject, его преимущество взвешивается против сложности он добавляет к повседневного использования. Наша цель состоит в том, чтобы сохранить барьер для доступа - базовый уровень знаний ,необходимых для использования Ninject - как можно более низкой. Ninject имеет множество дополнительных функций, но их понимание не требуется , чтобы использовать основные функции.
2. **Sleek.** Framework наворотов является серьезной проблемой для некоторых проектов, и как таковой, все основные функциональные возможности Ninject находится в одной сборке без зависимостей вне. NET библиотеки базовых классов (есть также построить для Client Profile).След этого единого узла составляет приблизительно 100KB при компиляции для выпуска.
3. **Быстро.** Вместо того , чтобы полагаться на отражение для вызова, Ninject использует легкий генерации кода в CLR . Это может привести к резкому (8-50x) улучшение производительности во многих ситуациях.
4. **Точная.** Ninject помогает разработчикам сделать все правильно в первый раз вокруг.Вместо того , чтобы полагаться на XML - файлов и отображения строк идентификаторов телеграфировать компонентов, Ninject обеспечивает надежную предметно-ориентированный язык. Это означает , что Ninject воспользовался преимуществом возможностей языка (например , типа-безопасности) и IDE (например , IntelliSense и завершение кода).
5. **Проворный.** Ninject строится вокруг компонентной архитектуры, с настройкой и эволюции в виду. Многие аспекты системы могут быть дополнены или изменены , чтобы соответствовать требованиям каждого проекта.
6. **Скрытность.** Ninject не вторгнуться в ваш код. Вы можете легко изолировать зависимость от Ninject к одной сборке в вашем проекте.
7. **Мощный.** Ninject включает в себя множество дополнительных функций. Например, Ninject был первый инжектор зависимость для поддержки контекстная связывания, в котором другой конкретной реализации услуги может быть введен в зависимости от контекста , в котором она запрашивается. Ninject поддерживает большинство основных объектов ,предлагаемых конкурирующими рамок (хотя многие такие элементы живут в *расширений* - плагинов модулей , которые слой на объекты поверх ядра).

* Entity Framework

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

* CSS 3

CSS3 (Cascading Style Sheets 3 — каскадные таблицы стилей третьего поколения) — активно разрабатываемая спецификация [CSS](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D0%BA%D0%B0%D0%B4%D0%BD%D1%8B%D0%B5_%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%8B_%D1%81%D1%82%D0%B8%D0%BB%D0%B5%D0%B9). Представляет собой [формальный язык](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA), реализованный с помощью [языка разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8). Самая масштабная редакция по сравнению с CSS1, CSS2 и CSS2.1. Главной особенностью CSS3 является возможность создавать анимированные элементы без использования [JS](https://ru.wikipedia.org/wiki/JavaScript), поддержка линейных и радиальных градиентов, теней, сглаживания и многое другое.

* HTML5

HTML5 ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) HyperText Markup Language, version 5) — [язык для структурирования и представления](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) содержимого[всемирной паутины](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0). Это пятая версия [HTML](https://ru.wikipedia.org/wiki/HTML). Хотя стандарт был завершён (рекомендованная версия к использованию) только в 2014 году (предыдущая, четвёртая, версия опубликована в 1999 году[[3]](https://ru.wikipedia.org/wiki/HTML5#cite_note-3)), ещё с 2013 года браузерами оперативно осуществлялась поддержка, а разработчиками — использование рабочего стандарта ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) HTML Living Standard). Цель разработки HTML5 — улучшение уровня поддержки [мультимедиа](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D0%B8%D0%B0)-технологий с одновременным сохранением обратной совместимости, удобочитаемости кода для человека и простоты анализа для [парсеров](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7" \o "Синтаксический анализ).

* JQuery

**jQuery** — [библиотека JavaScript](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript), фокусирующаяся на взаимодействии [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) и [HTML](https://ru.wikipedia.org/wiki/HTML). Библиотека jQuery помогает легко получать доступ к любому элементу [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), обращаться к атрибутам и содержимому элементов[DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), манипулировать ими. Также библиотека jQuery предоставляет удобный [API](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для работы с [AJAX](https://ru.wikipedia.org/wiki/AJAX). Сейчас разработка jQuery ведётся командой jQuery во главе с [Джоном Резигом](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B7%D0%B8%D0%B3,_%D0%94%D0%B6%D0%BE%D0%BD).

**Глава 2 Проектирование предметной области**

**2.1 Проектирование структуры сайта**

Структура сайта - древовидное иерархическое представление всех элементов сайта, включая информационные разделы и их подразделы.

Линейная - страницы располагаются в определенном порядке. Переход с одной страницы на другую строго определен. Такая структура обоснована, например, при обучении рис. 2.



Рис. 2 – Линейная структура сайта

Иерархическая - страницы разбиты по категориям и подкатегориям. Такая структура наиболее удобна рис. 3.



Рис. 3 - Иерархическая структура сайта

Произвольная - страницы расположены в свободном порядке. Такая структура оправдана только для небольших сайтов рис. 4.



Рис. 4 – Произвольная структура сайта

В ходе анализа вышеперечисленных архитектур сайта, для создания интернет магазина была выбрана иерархическая структура сайта рис. 5.

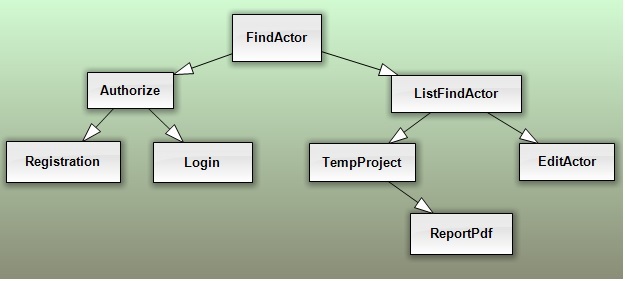


Рис. 5 – Структура веб интерфейса “Wow cast”

# 2.2 Проектирование базы данных

## 2.2.1 Постановка задачи

Для автоматизации работы модельного агентства требуется разработать базу данных (БД) и систему управления базой данных (СУБД). СУБД должна предусматривать основные функции работы с данными БД.

**2.2.2 Цели разработки базы данных**

Целью разработки базы данных является - хранение и использование информации о какой-либо предметной области. Так же одной из важнейших целей проектирования БД является сокращение избыточности хранимых данных, а следовательно, экономия объема используемой памяти, уменьшение затрат на многократные операции обновления избыточных копий и устранение возможности возникновения противоречий из-за хранения в разных местах сведений об одном и том же объекте. Так называемый, "чистый" проект БД ("Каждый факт в одном месте") можно создать, используя методологию нормализации отношений.

## 2.2.3 Пользовательские требования

В данном проекте необходимо организовать работу двух пользователей: администратора и покупателя. И реализовать процесс авторизации.

Обеспечить "покупателю" возможность исполнять свою трудовую деятельность, при этом запретив всякое самовольное изменение данных.

Пользователь с правами администратора имеет полный доступ ко всем данным.

## 2.2.4 Структура базы данных

Информация в приложении хранится в базе данных Actor.

Actor – отвечает за хранение данных об кандидатах их фотографиях, информации о пользователях, а также отобранных кандидатов каждого пользователя и включает в себя 8 таблицах рис 6.

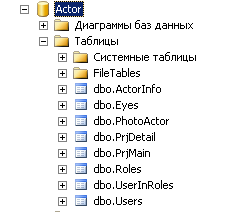


Рис. 6 – Перечень таблиц базы данных Actor

Обеспечение целостности и определение типов связи между таблицами базы данных Actor.

При планировании таблиц имеются два важных шага: определить допустимые значения для столбца и решить, каким образом обеспечить целостность данных в этом столбце. Целостность данных подразделяется на следующие категории.

* Сущностная целостность
* Доменная целостность
* Ссылочная целостность
* Пользовательская целостность

С целью сохранения связей между таблицами при добавлении или удалении строк возникла необходимость в определении ссылочной целостности.

ActorInfo – Eyes. Каждый актер может принадлежать только к одним глазам, отношение многие к одному.

ActorInfo – PhotoActor. Каждый актер может содержать множество фотографий, отношение один ко многим.

Схема отношений представлена на рис. 7.

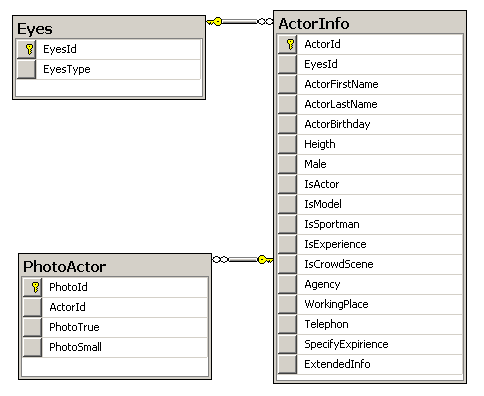


Рис. 7 - Отношение связей таблиц ActorInfo, Eyes, PhotoActor.

PrjDetail – PhotoActor. Каждая детальная информация проэкта может принадлежать только в одной фотографии, отношение многие к одному.

PrjDetail – PrjMain. Каждая детальная информация проэкта может принадлежать только к одному проэкту, отношение многие к одному.

PrjMain – Users. Каждый проэкт может принадлежать только одному пользователю, отношение многие к одному.

Схема отношений представлена на рис. 8.

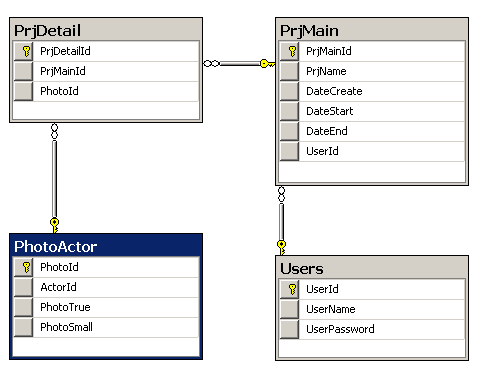


Рис. 8 - Отношение связей таблиц PrjDetail, PrjMain, PhotoActor, Users.

UserInRoles – Users - Roles. Каждая роль может принадлежать каждому пользователю и каждый пользователь может принадлежать каждой роли, отношение многие ко многим.

Схема отношений представлена на рис. 9.

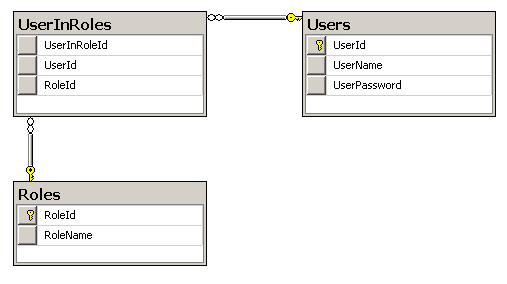


Рис. 9 - Отношение связей таблиц UserInRoles, Users, Roles.

**Глава 3. Проектирование приложения и решение поставленных задач**

**3.1 Архитектура приложения**

В ходе построения приложения была выбрана архитектуры, которая называется луковая (onion architecture). Эта архитектура идеально подходит для приложений с длительным жизненным циклом и сложной бизнес логикой. Считается, что ее использование в подобных проектах приводит к превосходным результатам, в следствии изначально заложенного в архитектуру акцента на разделение различных аспектов приложения. В луковой архитектуре уделяется особое внимание к описанию поведения системы в терминах контрактного программирования и выносу инфраструктурного кода во внешние модули. На диаграмме ниже изображена луковая архитектура рис 10.

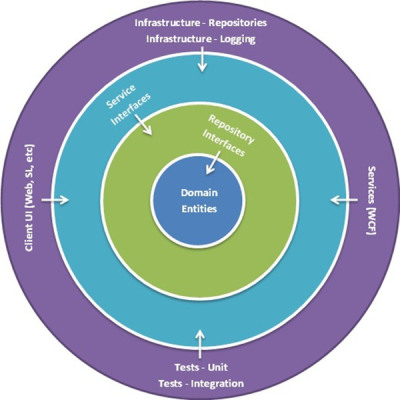


Рис 10 – Оnion архитектура

Данный подход был применен в построении архитектуры приложения

рис. 11

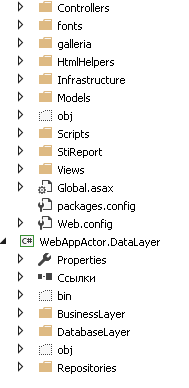


Рис 11 – Схема слоев приложения

Ее ключевая особенность в управлении связанностью. Фундаментальное правило — любой модуль приложения может зависеть от более близких к центру луковицы модулей, но не может зависеть от более дальних. Иными словами, любая связанность должна быть направлена к центру луковицы.   
В самом центре находится доменная модель приложения.

К доменной модели относятся сущности посредством которых происходит взаимодействие с удаленным хранилищем данным. В приложении хранилищем данных является база данных SQL. Сущности же в свою очередь ничего не знают о своей принадлежности к хранилищу, которое забегая вперед будет определена в слое DatabaseLayer.

На доменном уровне так же располагаются интерфейсы репозиториев, которые является обслуживающим звеном к конкретным сущностям. Все интерфейса репозиториев наследуются от базового интерфейса IGlobalRepository, который содержит в себе четко определенные концептуальные границы, отличая его от всех других объектов, тем самым избавляя от дублировании кода и предоставляет базовый функционал, который присущ каждому из репозиториев.

Ко второму уровню архитектуры приложения относится слой сервисных интерфесов, которые предоставляют конкретный функционал для взаимодейтсвия с слоем данных. Сервисы как и репозитории имеют базовый интерфейс, который содержит в себе только значимые для остальных интерфейсов сигнатуры методов.

На уровне слоя сервисов располагаются интерфейсы, которые являются единицей работы манипуляцией над данными. Иными словами в базовом интерфейсе IActorRepositoryнаходится функционал работы с сущностью Actor.

Третий слой носит название DatabaseLayer и в нем принимается решение с каким хранилищем будут взаимодействовать репозитории и сущности и определяется контекст для доступа к этим хранилищам. В приложении имеется один контекст, одна база данных MsSql. Для сопоставления сущностей и хранилищем данных выбран объектно-реляционный модуль Entity Framework версии 6.0, позволяющий работать с реляционными данными с помощью объектов, специализированных для доменов. Соединение приложения с б/д проложено при помощи провайдера SqlProviderServices, который идет с пакетом Entity Framework.

Этот контекст отвечает за взаимодействие с б/д Actor. В сборке WebAppActor.DataLayer .DatabaseLayer определен контекст для доступа к б/д и в теле класса контекста находятся сущности из доменного слоя. На данном этапе построения приложения был определен способ доступа к данным и их хранилище. В связи с этим сконфигурированы сопоставления к таблицам, указаны зависимости между сущностями и настроены ограничения на их свойства.

Далее слой инфраструктуры содержит классы которые используются внутри нашего приложения для облегчения взаимодействия с сущностями, так называемые бизнес классы в сборке WebAppActor.DataLayer. BusinessLayerю

Далее слой инфраструктуры, содержит классы реализующие интерфейсы репозиториев и сервисов. Эти классы тесно связаны с конкретной технологией доступа к данным и именно поэтому они вынесены за пределы ядра приложения и располагаются в слое, где был выбрано хранилище и доступ к данным.

Следующий слой под номером четыре носит название Infrastructure. Луковая архитектура основана на принципе инверсии зависимостей. Приложению для работы необходимы реализации расположенных в ядре интерфейсов, а поскольку конкретные реализации находятся на внешнем радиусе приложения, приложению также необходим и механизм для инжектирования зависимостей. В указанном слое реализован IoC контейнер, являющийся сингелтоном в приложении. Посредством Ioc контейнера Ninject частично происходит регистрация сопоставлений конкретных и абстрактных типов реализованных к данному этапу приложению.

В данном слое находится интерфейсы и реализации классов, относящиеся к конкретному приложению и решающие прямые задачи, характерны только для него.

Пятый слой WebAppActor. На этом уровне происходит непосредственное взаимодействие с пользователем. Этот уровень включает компоненты пользовательского интерфейса, механизм получения событий от пользователя. В данном слое расположены представления и все те компоненты, который составляют пользовательский интерфейс (стили, javascript), а также модели представлений, контроллеры.

Контроллером по умолчанию является Home контроллер. Он предоставляет метод для отображения фильтра выбора актеров, метода отображения списка актеров по параметрам фильтра, а также метода отображения всех фотографий конкретного актера рис 12.

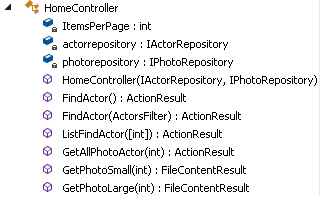


Рис 12 – Структура класса HomeController.cs

При вызове метода “ListFindActor” из метода действия “FindActor” вся информация по фильтрам записывается в бизнес класс “ActorFilter” рис. 13, а затем через “reposirory”, а именно через метод поиска “actorFilters” отбирает выбранных актеров для отображения

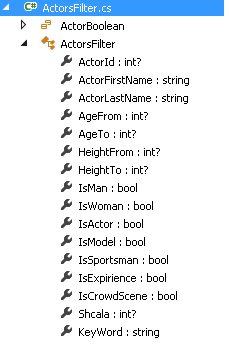


Рис 13. Бизнес класс ActorFilter

Далее обновляется страница, на которой после этого отображается вертикальный список актеров с фотографией и номером актера, списком всех фотографий выбранного актера кнопками добавить в проект и кнопкой просмотра проекта рис 14.

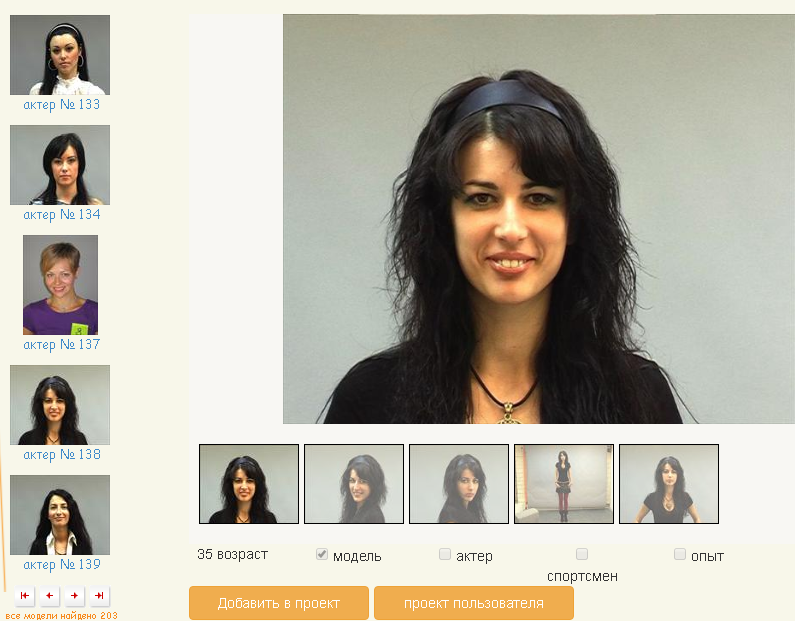


Рис 14. Отображения страницы ListFindActor.

Если нажать на номер актера под фотографией сработает метод “EditActor” контроллера “Actor” рис 15, если у пользователя есть права администратора.

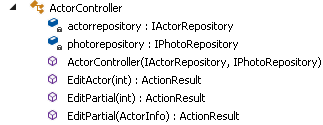


Рис 15. Контроллер Actor.

Далее отображается представление “EditActor” внутри которого вызывается метод действия “EditPartia” контроллера “Actor” который в свою очередь отображает частичное представление с данными выбранного актера рис 16.

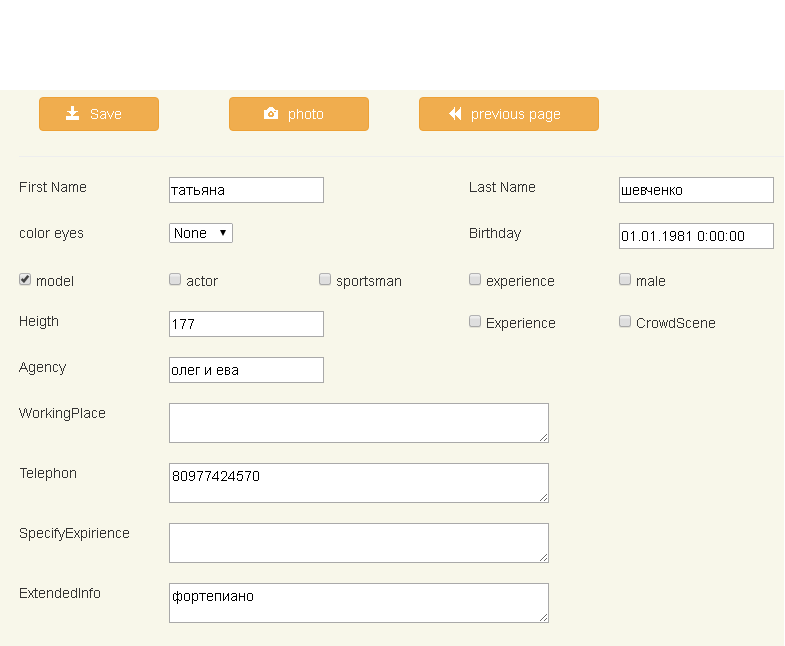


Рис 16. Данные актера для редактирования.

При нажатии на кнопку “add to project” в случае того если пользователь залогинелся сработает метод “AddCart” контроллера “Cart” рис 17

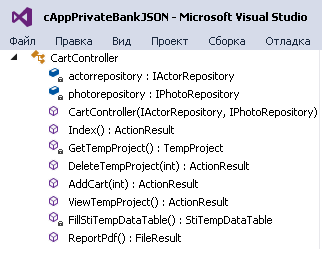


Рис 17. Контроллер Cart.

В котором вызывается метод GetTempProject() который возвращает экземпляр класа “TempProject” рис 18, который мы создали для временного хранения и редактирования, на протяжении существующей сессии, проекта пользователя

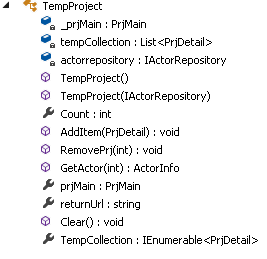


Рис 18. Класс TempProject

Происходит добавление актера до проекта текущего пользователя, в противоположном же случае пользователь будет перенаправлен на страницу авторизации

Кнопка “UserProject” выполняет метод действия “ViewTempProject” контроллера “Cart” который получает с текущей сессии проект пользователя и отображает всех отобранных кандидатов и краткую информацию о них рис 19.

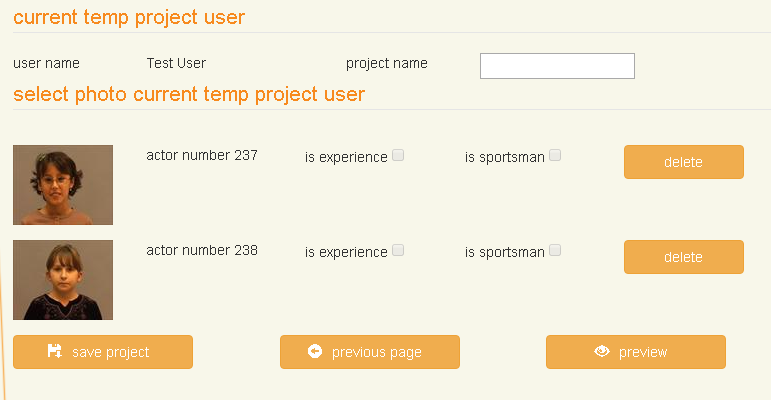


Рис 19. Страница ViewTempProject

Здесь можно удалить кандидата из списка нажатием кнопки delete послечего сработает метод “DeleteTempProject” контроллера “Cart”.

С помощью кнопки “preview” возможен вариант предпросмотра к андидатов в pdf формате рис. 20. При этом срабатывает метод “ReportPdf” контроллера “Cart”.

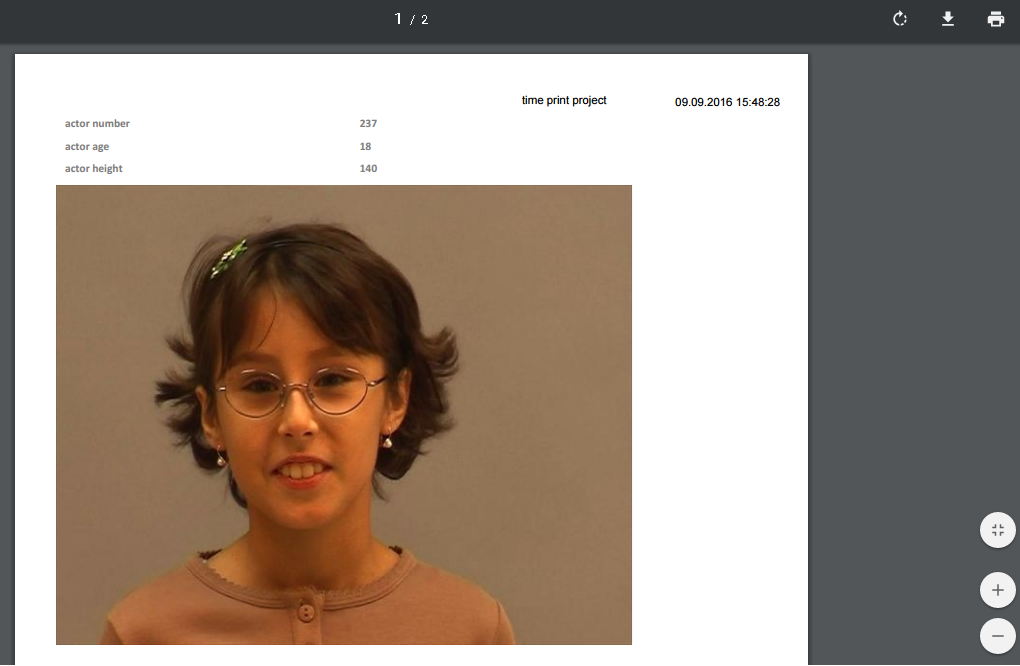


Рис 20. Отображение пред просмотра проекта.

С этой страницы также мы имеем возможность сразу скачать, распечатать, повернуть изображение, увеличить или уменьшить.

С помощью кнопки “Save project” текущий проэкт уже записывается в базу данных на постоянное хранение.

Примечание перед сохранением нужно в поле project name указать имя проэкта.

**3.2 Алгоритм фильтрации данных**

В ходе разработки приложения возникла необходимость создать фильтр, который мог бы обрабатывать данные полученные из заполненных полей на странице FindActor и на основании их формировать предикат запроса к хранилищу. Для исключения какой либо логики с тела методы контроллера было принято решение создать метод Predicate в классе ActorRepository рис. 21

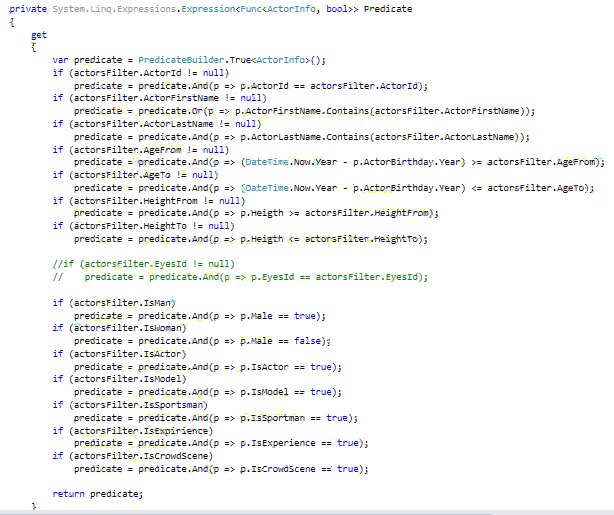


Рис 21. Метод Predicate

Которые используеься далее в этом же классе в методах: “Actors” и “CountActorByFilter” рис. 22

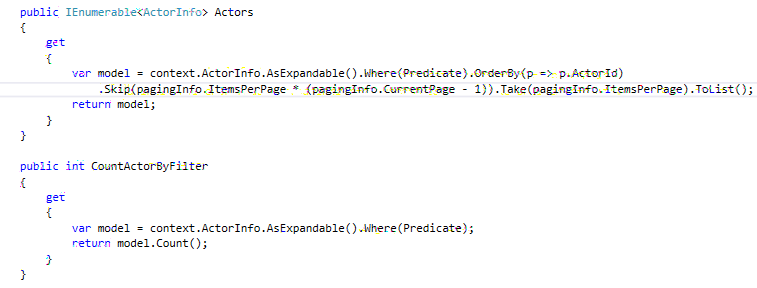


Рис 22. Методы: “Actors” и “CountActorByFilter”

Которые в свою очередь далее в программе используются в методах “HomeController” – ра “FindActor” и “ListFindActor” рис. 23.

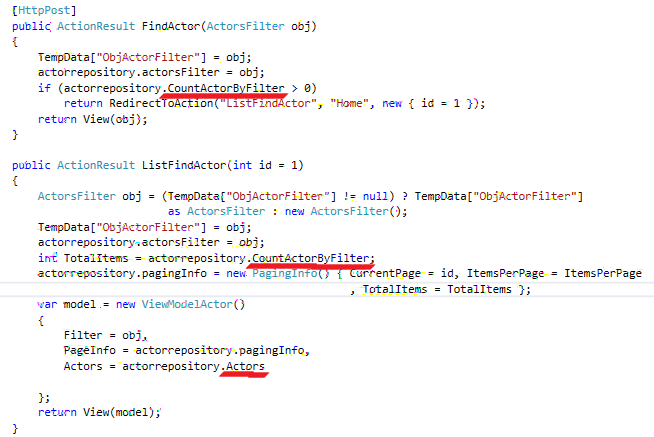


Рис 23. Использование методов CountByFilter и Actors в методах контроллера FindActor и ListFindActor

**Заключение**

В результате работы над дипломным проектом в соответствие с требованиями, изложенными в пункте 1.2, было разработано web-приложение и рассмотрены актуальные вопросы разработки и создания Web-сайта с использованием различных современных технологий.

В результате проведенных работ на базе выбранных технологий был создан веб интерфейс для работы с базой данных модельного агенства Wow cast.

При этом были решены следующие частные задачи:

* ознакомление с понятием электронной коммерции и её видами;
* изучение инструментов веб-разработки;
* выбор стратегии разработки и создания сайта;
* выбор архитектуры разработки приложения и его создание.

Таким образом, данная дипломная работа показывает, что поставленные задачи решены полностью. В результате работы над дипломным проектом был получен огромный опыт программирования на основе технологии ASP.NET MVC, что позволит в будущем создавать высокопроизводительные, профессиональные web-приложения.

**Список литературы**

1. [Джон Скит - C# программирование для профессионалов](http://www.proklondike.com/books/dotnet/skit_csharp_2011.html) 2013 г. ASP.NET MVC 4 Framework с примерами на C# 5.0 для профессионалов. ( Фримен А., Сандерсон С. ) - 2013 г.
2. C# 5.0. Справочник. Полное описание языка Джозеф Албахари, Бен Албахари 2014 г.
3. Programming Entity Framework: Code First Julia Lerman and Rowan Miller 2011 г.
4. Microsoft SQL Server 2012 Бондарь А. Г. 2013 г.
5. Оптимизация приложений на платформе .NET Голдштейн С., Зурбалев Д., Флатов И. - 2014 г.
6. Beginning ASP.NET Security [Barry Dorrans](http://eu.wiley.com/WileyCDA/Section/id-302479.html?query=Barry+Dorrans) 2010 г.
7. Приемы объектно-ориентированного проектирования Александр Шевчук, Дмитрий Охрименко, Андрей Касьянов 2015 г.
8. Приемы ООП Паттерны проектирования Э.Гамма, Р.Хелм, Р.Джонсон 2001 г.
9. ASP.NET MVC in Action 1st Edition [Jeffrey Palermo](http://www.amazon.com/Jeffrey-Palermo/e/B003T0DSGG/ref=dp_byline_cont_book_1) 2011 г.