МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ НФОРМАЦИОННЫХ ТЕХНОЛОГИЙ и РОБОТОТЕХНИКИ

Кафедра “Системы автоматизированного проектирования”

**Отчёт прохождения производственной практики**

**«**Веб-приложение типа “Вопрос-ответ” для изучения языков**»**

**Исполнитель:** студент гр.10702314 Азаревич В. А.

**Руководитель:** доц. Напрасников В. В.

Минск 2017

**СОДЕРЖАНИЕ**

[**1.** **ЦЕЛЬ** 3](#_Toc469407663)

[**2.** **ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ..** 3](#_Toc469407664)

[**3.** **ВЫБОР СУБД И СРЕДСТВ ПРОЕКТИРОВАНИЯ ПРИЛОЖЕНИЯ** 4](#_Toc469407665)

[**4.** **ПОРЯДОК РАЗРАБОТКИ ПРОЕКТА** 4](#_Toc469407666)

[**4.1** **СОЗДАНИЕ БАЗЫ ДАННЫХ** 5](#_Toc469407667)

[**4.2** **ДОСТУП К ДАННЫМ** 1](#_Toc469407668)1

[**4.3** **СОЗДАНИЕ ИНТЕРФЕЙСА И ПРИЛОЖЕНИЯ** 1](#_Toc469407669)2

[**5.** **ТЕСТИРОВАНИЕ**](#_Toc469407666) 24

[**ВЫВОД**](#_Toc469407670) 24

[**ЛИТЕРАТУРА** 25](#_Toc469407670)

# **ЦЕЛЬ**

Целью данной курсовой работы является создание web-приложения, в программе *Microsoft Visual studio 2013 ultimate* представляющего собой приложение типа «Вопрос-ответ» при изучении нового языка.

Данное web-приложение должно облегчить задачу поиска ответов на возникающие вопросы при изучении нового языка или культуры другой страны.

# **ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ. ЦЕЛЬ.**

Разработчики всегда пытались создавать шаблонные решения, ведь они позволяют ссылаться на известные конструкции, тем самым снижая количество ошибок. именно поэтому появились **паттерны**.

Паттерн (англ. design pattern) — повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста. Поэтому в данном курсовом проекте используется один из фундаментальных паттернов MVC (Model-View-Controller).

Модель MVC включает в себя три компонента: Модель, Представление и Контроллер.

Самое главное конечно здесь первое – это **Модель**. Модель представляет собой совокупность процедур и алгоритмов обработки данных. Сама по себе Модель не содержит данных, но как правило черпает их из базы данных и обрабатывает по заранее прописанным алгоритмам. Если говорить о Web разработке, то модель будет содержать набор классов и функций на каком-либо языке.

Второй элемент – это представление **View**. Позволяет отобразить информацию. Если это сайт, то информация отображается в браузере. Представление при разработке сайтов содержит HTML код, в который подставляются переменные, которые берутся, нет не из модели, а из контроллера.

* Итак, третий элемент – это Контроллер. Его главная функция это обеспечение связи с пользователем и моделью. . В приложении MVC представления только отображают данные, а контроллер обрабатывает вводимые данные и отвечает на действия пользователя. Например, контроллер может обрабатывать строковые значения запроса и передавать их в модель, которая может использовать эти значения для отправки запроса в базу данных.



# **ВЫБОР СУБД и СРЕДСТВ ПРОЕКТИРОВАНИЯ ПРИЛОЖЕНИЯ**

Одним из наиболее распространённым вопросом и проблемой, с которой сталкиваются разработчики, является «Выбор системы управления базами данных». При выборе СУБД следует учитывать множество различных факторов: "размер" разрабатываемого проекта, сложность структуры базы данных, сложность освоения, стоимость, скорость работы и др.

Для крупных web узлов предпочтительно выбирать СУБД, которые хорошо зарекомендовали себя в плане стабильной работы с потоками данных корпоративного уровня. В первое время как нельзя кстати подойдет СУБД Microsoft SQL Server Express 12. Ее преимуществом является то, что в случае большого роста числа клиентов интернет провайдера, все данные будет легко перевести в Microsoft SQL Server, а на данном этапе все ограничения никак не повлияют на процесс.

Само приложение будет разбалтываться при помощи Microsoft ASP.NET MVC 4 Web Application. Это мощная распространённая платформа, обладающая высокой надежностью и скоростью обработки запросов, а так же рядом дополнительных удобств. Конечно, и она имеет свои недостатки, один из которых это размер приложения, но достоинств эта платформа имеет куда больше, чем недостатков, при разработке web-приложений подобного типа.

# **ПОРЯДОК РАЗРАБОТКИ ПРОЕКТА**

Работа над данным проектом начинается с создания инфраструктуры базы данных, а так же с продумывания основных аспектов работы приложения. Прежде всего, создаём необходимые таблицы базы данных, а так же немного заполним её. Это позволяет лучше продумать её структуру и избежать некоторых ошибок.

Далее разрабатывается непосредственно web-приложение. Для этого была использована *Microsoft Visual Studio 2013 Ultimate*.

# **Создание базы данных**

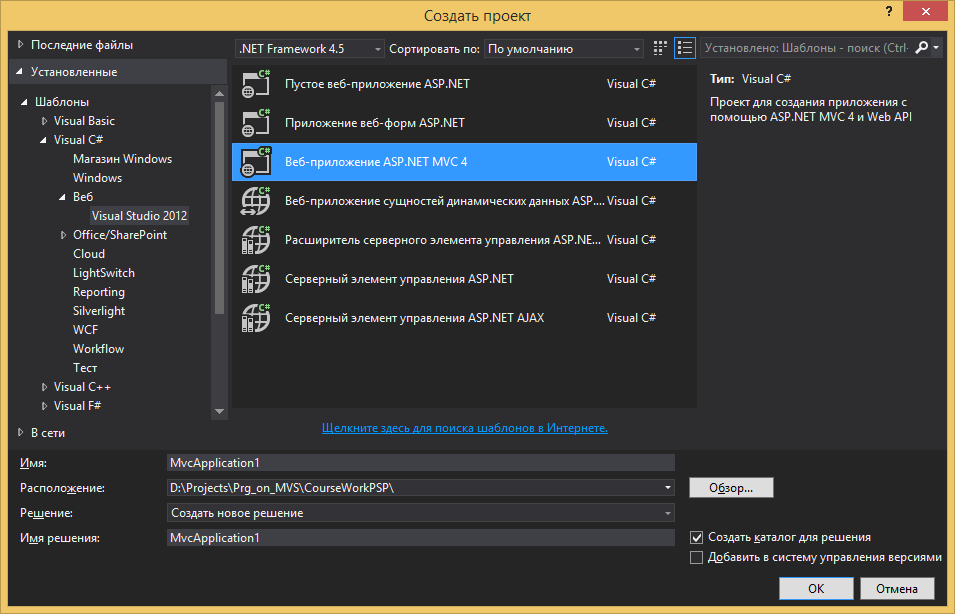
Будем использовать локальную базу данных, встроенную в Microsoft Visual Studio 2013 Ultimate. Для это создадим проект выбирается NewProject, выбираетсяASP.NET,MVC 4, что продемонстрировано на Рисунок 1. 

Рисунок 1 – Создание проекта

Далее в папку App\_Data необходимо добавить базу данных. Для этого нажимаем правой клавишей на папку App\_Data ->Add ->NewItem. В всплывшем диалоговом окне выбираем База данных SQLServer и называем DataBase1, что показано на Рисунок 2.

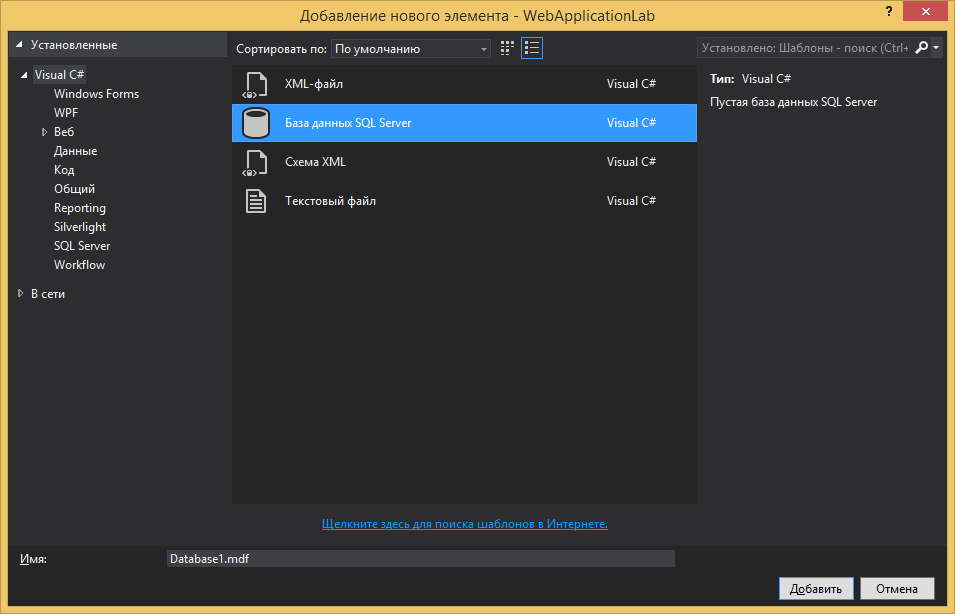


Рисунок 2 – Добавление базы данных

Теперь можем открыть базу данных в Обозревателе Серверов и, нажав правой клавишей мыши на пункт Таблицы, выбрать в всплывающем меню Add New Table Рисунок 3

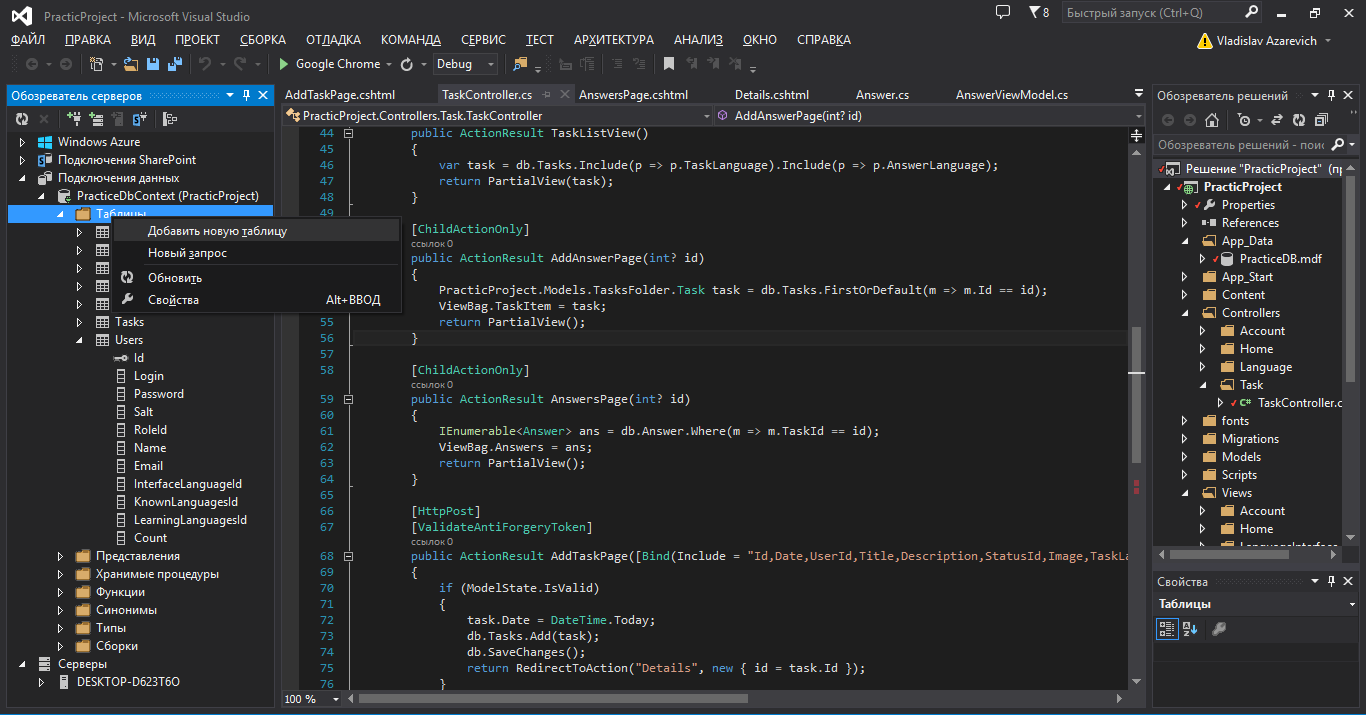


Рисунок 3 – Добавление новой таблицы базы данных

Таблица Answers состоит из нескольких полей:

* Поле Id – устанавливает порядковый номер модели и также является первичным ключом;
* Поле Description – Поле с ответом.
* Поле Image – Прикреплённая картинка.
* Поле Likes – Рейтинг ответа**.**
* Поле TaskId ­– ИД задания, к которому относится ответ.
* Поле Date – Дата ответа.

Таблица Answers показана на рисунке 4.

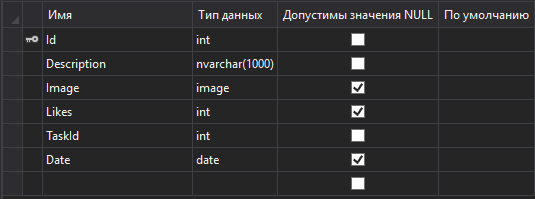


Рисунок 4 – Таблица Answers

Таблица Languages состоит из нескольких полей:

* Поле Id – устанавливает ID языка;
* Поле Name – наименование.

Таблица Languages показана на рисунке 5.

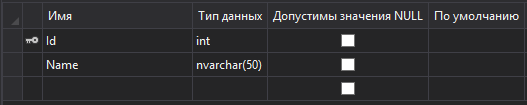


Рисунок 5 – Таблица Languages

Таблица Roles состоит из 2 полей:

* Поле Id – ID производителя.
* Поле Name – имя производителя.

Таблица Roles показана на рисунке 6.

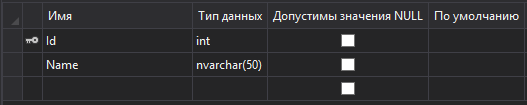


Рисунок 6 – Таблица Roles

Таблица Tasks состоит из нескольких полей:

* Поле Id – ID задачи
* Поле Date – дата публикации задачи
* Поле UserId – Id пользователя добавившего задачу
* Поле Title – заголовок задачи
* Поле Description– полное описание задачи
* Поле Image– картинка для задания
* Поле TaskLanguageId– Id языка, на котором составлена задача
* Поле AnswerLanguageId– Id языка, требующегося для ответа.

Таблица Tasks показана на рисунке 7.

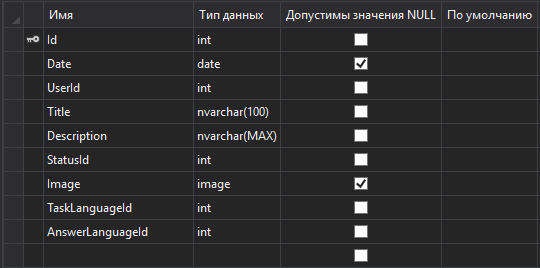


Рисунок 7 – Таблица Tasks

Таблица Users состоит из следующих полей:

* Поле Id – устанавливает порядковый номер пользователя;
* Поле Login – хранит логин пользователя;
* Поле Password – хранит пароль пользователя;
* Поле Salt – хранит шифр пароля для защиты данных;
* Поле RoleId – хранит номер роли пользователя.
* Поле Name – хранить имя пользователя
* Поле Email – хранит почту пользователя
* Поле InterfaceLanguageId– хранит Id интерфейса пользователя
* Поле KnownLanguagesId– хранит Id родного языка пользователя
* Поле LearningLanguagesId– хранит Id изучаемого языка пользователя
* Поле Count – хранит количество задач, которые он может добавить

Таблица Users показана на рисунке 8.

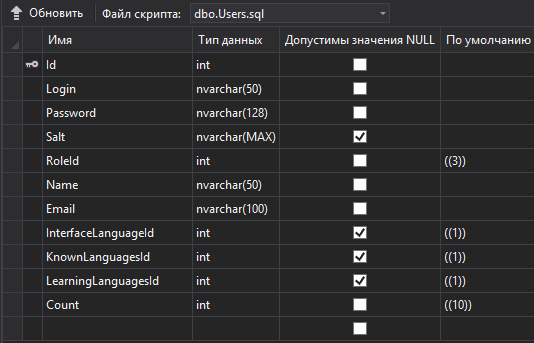


Рисунок 8 – Таблица Users

Для более комфортной работы с базой данных установим автоматическое инкрементирование поле Id всех таблиц. Для этого в свойствах поля Id каждой таблицы в поле спецификация идентификатора устанавливаем значение “true”.

# **Доступ к данным**

После создания таблиц создается модель классов базы данных с помощью дополнения EntityFramework. Пользуясь моделью можно взаимодействовать с самой базой данных, абстрагируясь от SQL запросов.

При создании данного приложения использовался подход «Code First», который означает, что мы пишем (либо продумываем) контекст данных, затем описываем его в коде, после чего фреймворк сопоставляет его с нашей базой данных(рисунок 9).

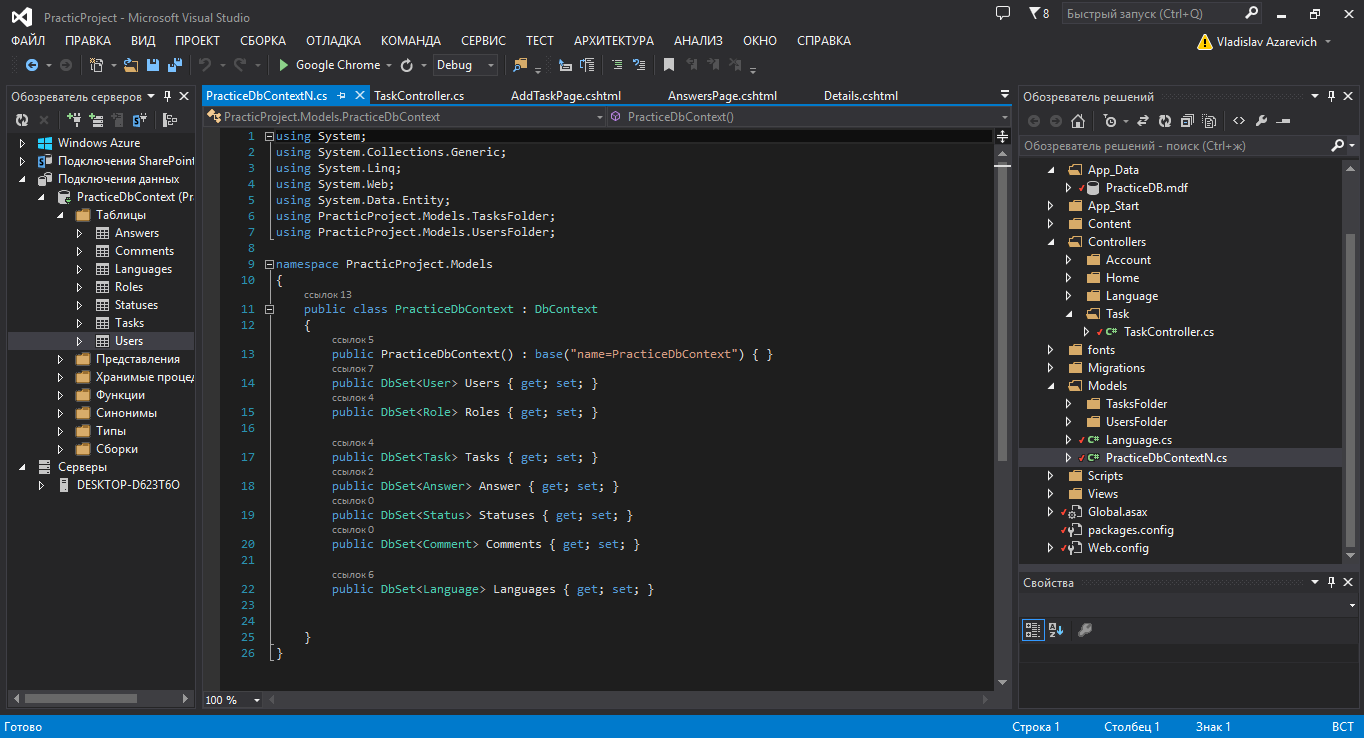


Рисунок 9 – контекст данных для связи с БД

**4.3 Создание интерфейса и приложения**

Данное web-приложение использует страницу-шаблон. Она полезна тем, что позволяет избегать повторения, сэкономить время, составить общее оформление для создания последующих страниц.

Для создания мастер-страницы в папку Views/shared добавляем представление и называем его \_layout.cshtml.

MasterPage будет состоять из блоков. Для этого используется тэг <div>.

1. Шапка (class = “container”)

Шапка статична и содержит кнопки для входа/выхода, кнопку регистрации, а также меню, в котором есть ссылки для перехода на другие страницы web-приложения. Шапка показана на рисунке 10



Рисунок 10 – Шапка

<!DOCTYPEhtml>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@ViewBag.Title</title>

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

</head>

<body>

<div class="navbar navbar-inverse navbar-fixed-top">

<div class="container">

<div class="navbar-collapse collapse">

<ul class="nav nav-pills nav-justified" style="padding-top:5px">

<li>@Html.ActionLink("Главная", "Index", "Home")</li>

<li>@Html.ActionLink("Добавить задачу", "AddTaskPage", "Task")</li>

@if (User.IsInRole("admin"))

{

<li>@Html.ActionLink("Пользователи", "Index", "Users")</li>

<li>@Html.ActionLink("Магазины", "Index", "Shop")</li>

}

@if (!User.Identity.IsAuthenticated)

{

<li>@Html.ActionLink("Войти", "Login", "Account")</li>

<li>@Html.ActionLink("Регистрация", "Register", "Account")</li>

}

else

{

<li>@Html.ActionLink("Мои задачи", "MyTasks", "Task")</li>

<li>@Html.ActionLink("Выход", "LogOff", "Account")</li>

}

</ul>

</div>

</div>

</div>

<div class="container body-content">

@RenderBody()

</div>

<hr />

<div class="navbar-bottom">

<div class="navbar-inner">

<div class="container">

@Html.ActionLink("Сменить язык интерфейса", "LanguageInterface", "LanguageInterface", null, new { @class = "btn" })

</div>

</div>

</div>

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/scripts/jquery.unobtrusive-ajax.js")

@Scripts.Render("~/bundles/bootstrap")

@Scripts.Render("~/bundles/bootstrap-select.js")

@RenderSection("scripts", required: false)

</body>

</html>

Код тела страницы

Здесь @RenderBody() – функция для вызова тела других страниц.

Также добавим в наш проект таблицу стилей CSS, чтобы украсить оформления web-приложения. Для этого создадим папку Content и в этой папке создадим документ Styles.css

Основной функционал web-приложения состоит из общедоступных страниц:

1. Страница, состоящая из таблицы всех задач;
2. Страница, содержащая подробную информацию о каждой задачи;
3. Страница входа в аккаунт;
4. Страница регистрации.

На главную страницу выводим список всех задач с указанием некоторой информации (Даты добавления задания, требуемый язык ответа, язык самого задания). При этом в есть кнопка «Подробнее», ведущая на страницу об этой задаче, где можно увидеть более подробную информаци о задании, комментарии и ответы (рисунок 11). Здесь Layout = "/Views/ShowContent.cshtml" значит, что страница будет использовать мастер страницу. А @modelIEnumerable<webapplicationlab.Models.Component>, страница использует перечисление объектов типа Task из таблицы Tasks.

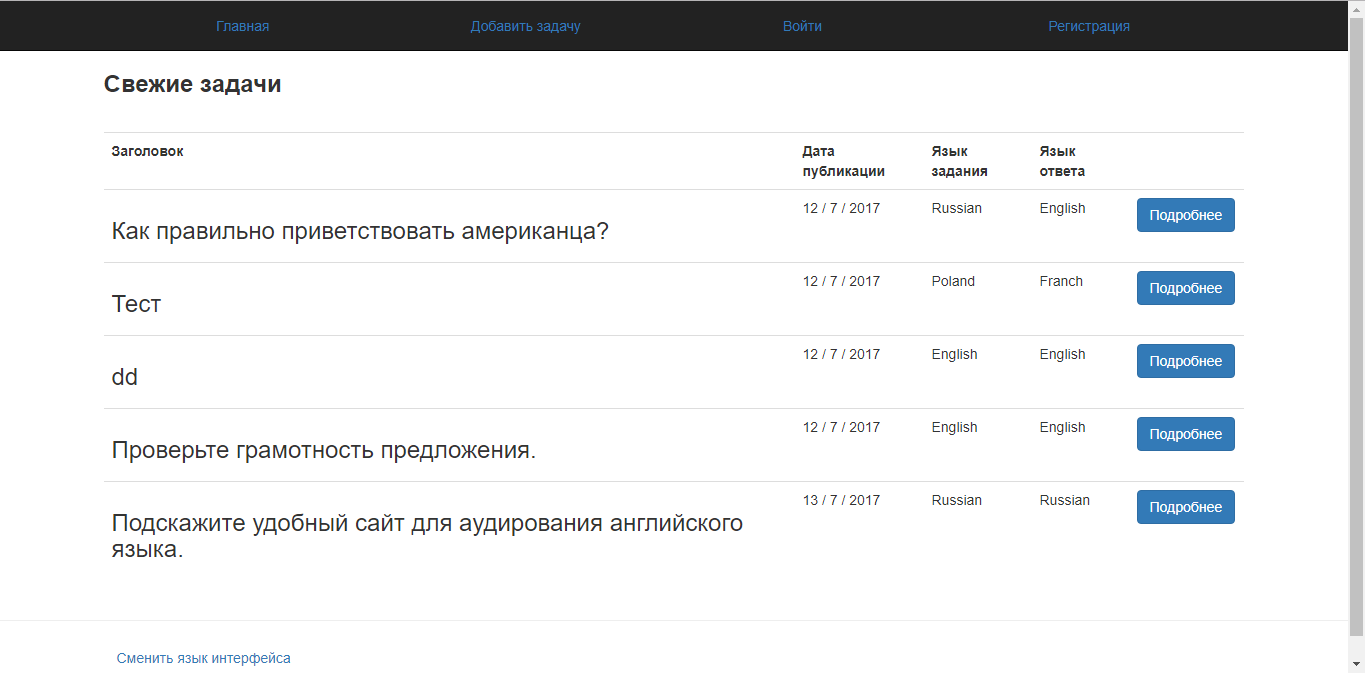


Рисунок 11– Страница просмотра задач

@model IEnumerable<PracticProject.Models.TasksFolder.Task>

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.Title)

</th>

<th>

@Html.DisplayNameFor(model => model.Date)

</th>

<th>

@Html.DisplayNameFor(model => model.TaskLanguage)

</th>

<th>

@Html.DisplayNameFor(model => model.AnswerLanguage)

</th>

<th></th>

</tr>

@foreach (var item in Model)

{

<tr>

<td>

<h3> @Html.DisplayFor(modelItem => item.Title)</h3>

</td>

<td>

@item.Date.Day / @item.Date.Month / @item.Date.Year

</td>

<td>

@Html.DisplayFor(modelItem => item.TaskLanguage.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.AnswerLanguage.Name)

</td>

<td>

@Html.ActionLink("Подробнее", "Details", new { id = item.Id }, new { @class = "btn btn-primary" })

</td>

</tr>

}

</table>

Рассмотрим подробнее страницу добавления задачи. Страница содержит поля для заполнения. Пример такой страницы приведен на рисунке 12 ниже.

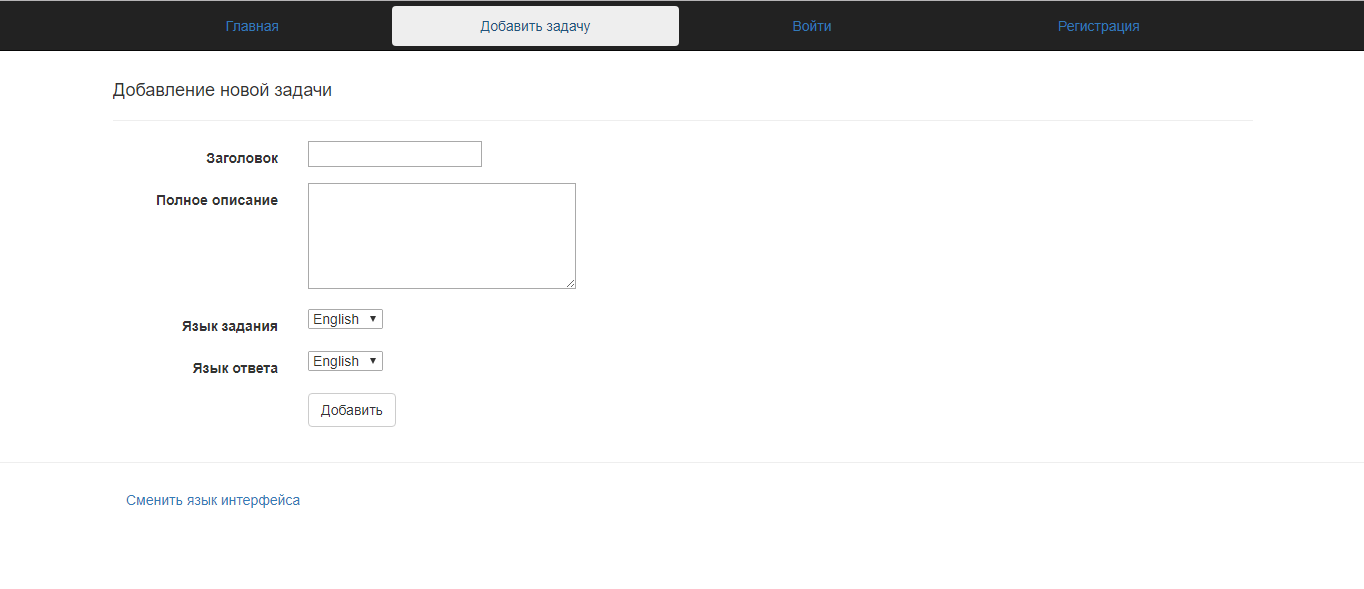


Рисунок 12– Страница добавления задачи

@model PracticProject.Models.TasksFolder.TaskViewModel

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<br />

<h4>Добавление новой задачи</h4>

<hr />

@Html.ValidationSummary(true)

<div class="form-group">

@Html.LabelFor(model => model.Title, new { @class = "control-label col-sm-2" })

<div class="col-sm-10">

@Html.EditorFor(model => model.Title)

@Html.ValidationMessageFor(model => model.Title)

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Description, new { @class = "control-label col-md-2" })

<div class="col-md-10">

<!--Html.EditorFor(model => model.Description)-->

@Html.TextAreaFor(model => model.Description, new { cols = 35, @rows = 5 })

@Html.ValidationMessageFor(model => model.Description)

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.TaskLanguage.Name, "Язык задания", new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("TaskLanguageId")

@Html.ValidationMessageFor(model => model.TaskLanguageId)

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.AnswerLanguage.Name, "Язык ответа", new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("AnswerLanguageId")

@Html.ValidationMessageFor(model => model.AnswerLangId)

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Добавить" class="btn btn-default" />

</div>

</div>

</div>

}

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

Рассмотрим страницу, которое содержит информацию о конкретной задаче. Пример такой страницы приведен на рисунке 13 ниже.

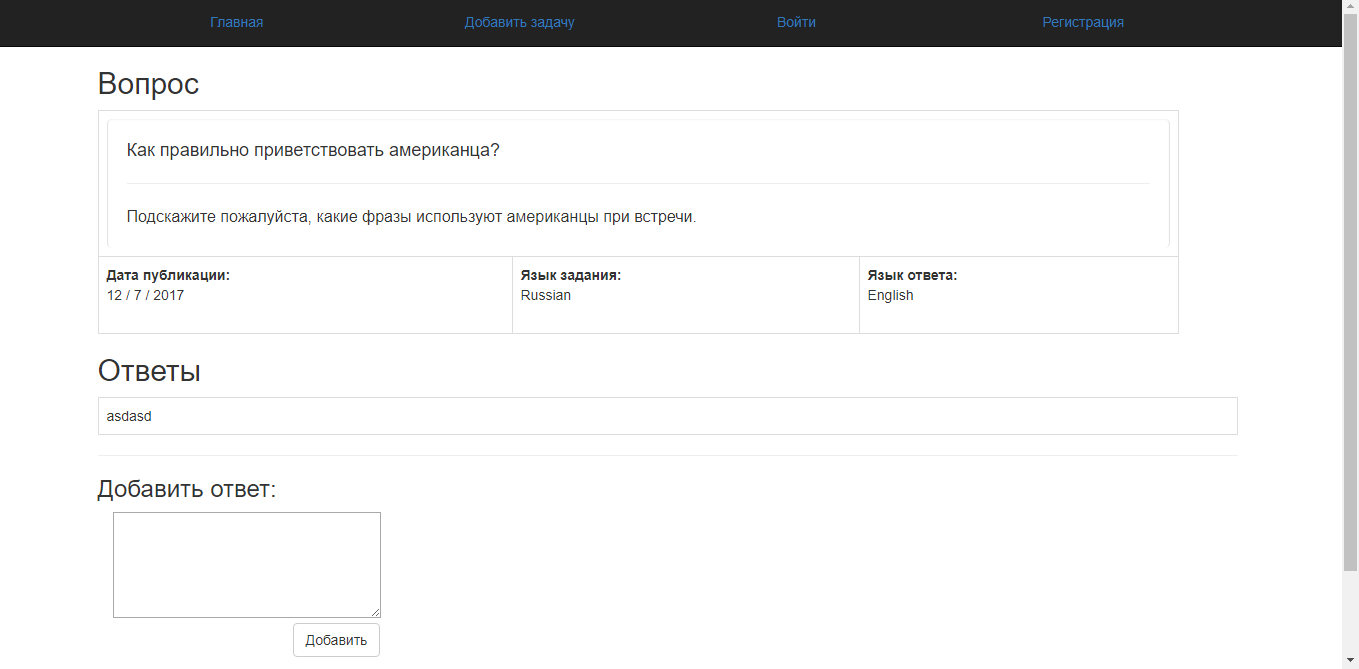


Рисунок 13 – Страница, содержащая информацию о конкретной задаче

@model PracticProject.Models.TasksFolder.Task

@{

ViewBag.Title = "Details";

}

<h2>Вопрос</h2>

<div>

<table class="table table-bordered" >

<tr>

<td colspan="3">

<div class="well" style="background-color: transparent; font-size: large; border-bottom: none; border-top: none; margin-bottom: 0px; border-radius: 5px;">

@Html.DisplayFor(model => model.Title)

<hr>

<span style="font-size: medium">@Html.DisplayFor(model => model.Description)</span>

</div>

</td>

<td style="visibility:hidden"></td>

<td style="visibility:hidden"></td>

</tr>

<tr>

<td>

<dl>

<dt>

@Html.DisplayNameFor(model => model.Date):

</dt>

<dd>

@Model.Date.Day / @Model.Date.Month / @Model.Date.Year

</dd>

</dl>

</td>

<td>

<dl>

<dt>

@Html.DisplayNameFor(model => model.TaskLanguage):

</dt>

<dd>

@Html.DisplayFor(model => model.TaskLanguage.Name)

</dd>

</dl>

</td>

<td>

<dl>

<dt>

@Html.DisplayNameFor(model => model.AnswerLanguage):

</dt>

<dd>

@Html.DisplayFor(model => model.AnswerLanguage.Name)

</dd>

</dl>

</td>

</tr>

</table>

<h2>Ответы</h2>

@{Html.RenderAction("AnswersPage", "Task", new { id = Model.Id });}

@{Html.RenderAction("AddAnswerPage", "Task", new {id = Model.Id}); }

</div>

Здесь @modelwebapplicationlab.Models.Game.Task означает, что на странице используется конкретный объект класса Task. Объект выбивается с помощью функции ActionResult Details():

public ActionResult Details(int? id)

{

if (id == null) { id = 10; }

var task = db.Tasks.Include(p => p.TaskLanguage).Include(p => p.AnswerLanguage).FirstOrDefault(p => p.Id == id);

return View(task);

}

Функция принимает в качестве параметра id задачи подробную информацию о которой нужно вывести.

Страница входа в аккаунт. На этой странице есть поля для ввода логина, пароля и возможность запомнить данные в «куках» браузера, данные из которых посылаются на котроллер. Рисунок страницы рисунок 14 .

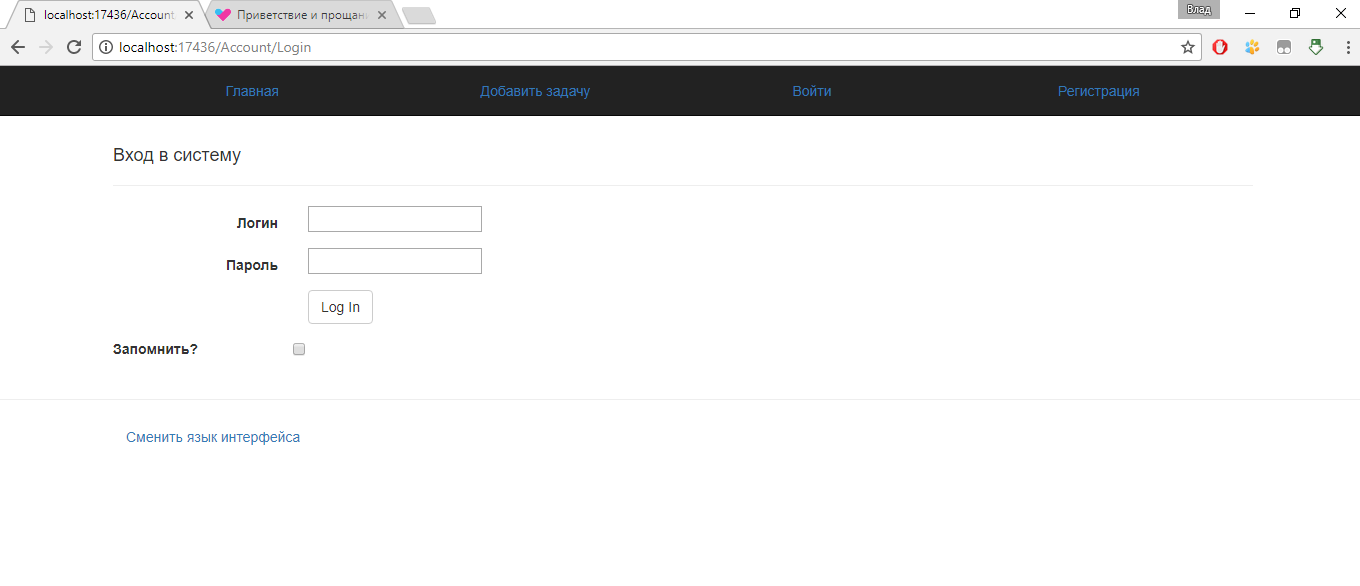


Рисунок 14 - Страница входа в аккаунт.

@model PracticProject.Models.LogViewModel

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<br />

<h4>Вход в систему</h4>

<hr />

@Html.ValidationSummary(true)

<div class="form-group">

@Html.LabelFor(model => model.UserName, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.UserName)

@Html.ValidationMessageFor(model => model.UserName)

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Password, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Password)

@Html.ValidationMessageFor(model => model.Password)

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Log In" class="btn btn-default" />

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.RememberMe, new { @class = "Control-label col-md-2" })

<div>

@Html.EditorFor(model => model.RememberMe)

</div>

</div>

</div>

}

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

Контроллер проверяет пользователя в базе данных. Если пользователь с таким логином и паролем существует, то происходит вход в аккаунт, иначе ошибка.

[HttpPost]

public ActionResult Login(LogViewModel model, string returnUrl)

{

if (ModelState.IsValid)

{

if (ValidateUser(model.UserName, model.Password))

{

return SetAuthCookie(model.UserName, model.RememberMe, returnUrl);

}

ModelState.AddModelError("", "Неправильный пароль или логин");

}

return View(model);

}

private ActionResult SetAuthCookie(string userName, bool rememberMe, string returnUrl)

{

FormsAuthentication.SetAuthCookie(userName, rememberMe);

if (Url.IsLocalUrl(returnUrl))

return Redirect(returnUrl);

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

}

private bool CreateUser(RegisterViewModel model, string password, string salt)

{

using (PracticeDbContext db = new PracticeDbContext())

{

try

{

User newUser = new User

{

Login = model.UserName,

Password = password,

RoleId = 3,

Salt = salt,

Name = model.Name,

Email = model.Email,

Count = 10,

InterfaceLanguageId = model.InterfaceLanguageId,

KnownLanguagesId = model.KnownLanguagesId,

LearningLanguagesId = model.LearningLanguagesId

};

db.Users.Add(newUser);

db.SaveChanges();

}

catch

{

return true;//false;

}

}

return true;

}

protected bool ValidateUser(string login, string password)

{

bool isValid = false;

try

{

IEnumerable<User> userList = db.Users.Where(u => u.Login == login);

User user = userList.FirstOrDefault();

if (user != null && password == null || user != null && Equals(user.Password, GenerateHash(password, user.Salt)))

isValid = true;

}

catch

{

isValid = false;

}

return isValid;

}

}

Следующая изображена страница регистрации. (Рис 24) На этой странице есть поля для введения логина, пароля и повтора пароля.

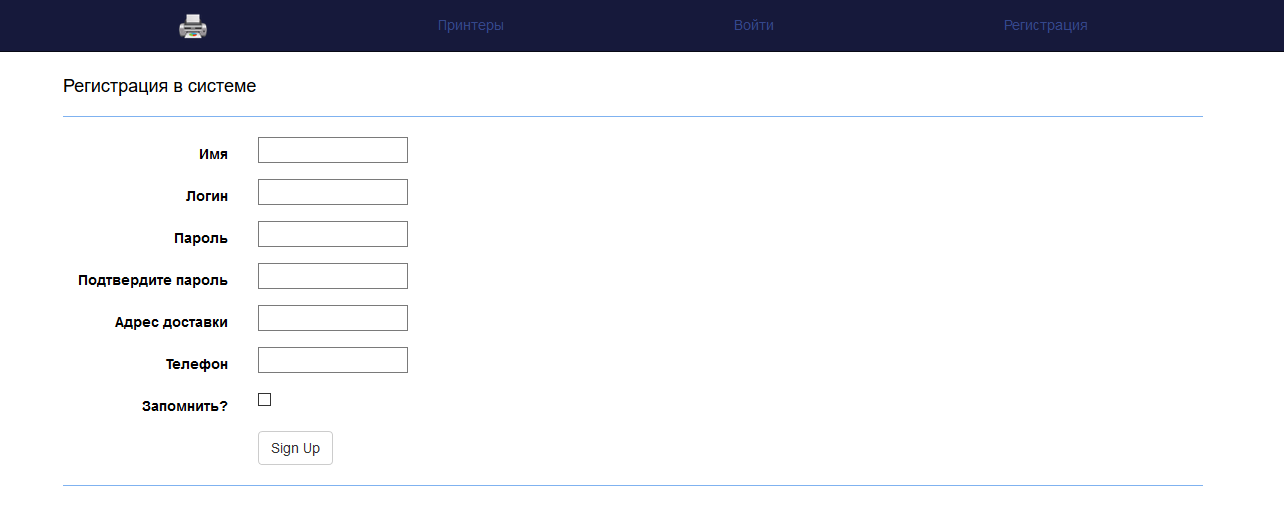


Рисунок 15 – Страница регистрации

@model PracticProject.Models.RegisterViewModel

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<br />

<h4>Регистрация в системе</h4>

<hr />

@Html.ValidationSummary(true)

<div class="form-group">

@Html.LabelFor(model => model.Name, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Name)

@Html.ValidationMessageFor(model => model.Name)

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.UserName, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.UserName)

@Html.ValidationMessageFor(model => model.UserName)

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Password, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Password)

@Html.ValidationMessageFor(model => model.Password)

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.PasswordConfirm, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.PasswordConfirm)

@Html.ValidationMessageFor(model => model.PasswordConfirm)

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Email, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Email)

@Html.ValidationMessageFor(model => model.Email)

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.InterfaceLanguage.Name, "Язык интерфейса", new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("InterfaceLanguageId")

@Html.ValidationMessageFor(model => model.InterfaceLanguageId)

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.KnownLanguages.Name, "Родной язык", new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("KnownLanguagesId")

@Html.ValidationMessageFor(model => model.KnownLanguagesId)

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LearningLanguages.Name, "Изучаемый язык", new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("LearningLanguagesId")

@Html.ValidationMessageFor(model => model.LearningLanguagesId)

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Sign Up" class="btn btn-default" />

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.RememberMe, new { @class = "Control-label col-md-2" })

<div>

@Html.EditorFor(model => model.RememberMe)

</div>

</div>

</div>

}

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

На этой странице есть поля для введения логина, пароля и повтора пароля.Если пароли совпадают, то данные посылаются в контроллер. Контроллер записывает данные в таблицу Users и сохраняет.

[HttpPost]

publicActionResult Register(RegisterViewModel model, stringreturnUrl)

{

if (ModelState.IsValid)

{

if (!ValidateUser(model.UserName, null))

{

string salt = GenerateSalt();

if (CreateUser(model, GenerateHash(model.Password, salt), salt))

{

returnSetAuthCookie(model.UserName, model.RememberMe, returnUrl);

}

}

else

{

ModelState.AddModelError("", "Пользователь с данным логином уже существует");

}

}

returnView(model);

}

Зарегистрировавшимся пользователям присваивается RoleId = 3. Это означает, что он будет иметь права пользователя. При присваивании параметра RoleId = 1 пользователь будет иметь статус администратора, в соответствии с нашей таблицей Roles.

# **5. ТЕСТИРОВАНИЕ**

Для данного приложения было выполнено визуальное и функциональное тестирование, в результате которых не было выявлено критических ошибок. Все визуальные неточности исправлены.

Функциональное тестирование — это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

Часть кода была отрефакторена, с целью уменьшения объёма кода.

# **ВЫВОД**

В ходе данного курсового проекта было разработано Web-приложение, реализующие следующие задачи:

* Отображение списка задач, с возможностью их добавления.
* Отображение ответов каждой задачи, с возможностью их добавления
* Регистрация пользователей с их ранжированием, и предоставление пользователям разных групп разные возможности.
* В результате разработки проекта были достигнуты поставленные цели и получен ожидаемый результат.

**Список литературы**:

ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов. Адам Фримен.

https://msdn.microsoft.com/ru-ru/

http://getbootstrap.com/