# Разработка Веб-приложения

## Средства разработки

Для разработки веб-приложения выбраны следующие средства:

* Microsoft Visual Studio 2022 Community – бесплатная версия среды разработки Visual Studio для разработки приложений на платформа .Net Framework;
* SQL Server 2016 Express – бесплатная версия СУБД Microsoft SQL Server;
* Microsoft SQL Server Management Studio – средство работы с СУБД Microsoft SQl Server;
* Entity Framework – объектно-ориентированная технология доступа к данным, является object-relational mapping (ORM) решением для .NET Framework от Microsoft.

## Создание нового проекта

Запустим Visual Studio и создадим новый проект веб-приложения. В качестве шаблона выбрано приложение ASP.NET Core MVC(модель-представление-контроллер) на платформе .NET 6.0.

## Подключение базы данных

Для работы с базой данных добавим строку подключения в файл конфигурации appsettings.json. Для работы с Entity Framework установим в приложение необходимый набор пакетов через менеджер пакетов NuGet:

* Microsoft.EntityFrameworkCore.Design;
* Microsoft.EntityFrameworkCore.SqlServer.

После установки пакетов для создания модели базы данных с помощью Entity Framework откроем утилиту Developer PowerShell через меню Visual Studio Средства->Командная строка. В командной строке выполним следующую команду «dotnet ef dbcontext scaffold --project WebApp Name=sql Microsoft.EntityFrameworkCore.Sqlerver --output-dir Models --context-dir Data --namespace WebApp.Models --context-namespace WebApp.Data --context DContext -f --no-onconfiguring». После выполнения команды в проекте будут автоматически сгенерированы классы объектной модели базы данных с учетом отношений между таблицами и необходимые классы для подключения и выполнения команд с БД.

## Инициализация подключения

После создания объектной модели необходимо добавить код инициализации подключения, для этого откроем файл Program.cs и добавим следующую строку «builder.Services.AddDbContext<SqliteContext>(options=> options.UseSqlServer("Name=sql"))».

## Структура приложения

Функционал приложения разделен на следующие основные страницы:

* Страница входа в систему;
* Страница администратора;
* Страница успеваемости.

## Страница входа в систему

Контроллер Home и действие Index вызываются по умолчанию при открытии пользователем веб-приложения, поэтому он будет использоваться в качестве страницы авторизации пользователей.

Для работы с БД добавим в контроллер поле «DContext \_context» (DContext – сгенерированный Entity Framework класс с коллекциями всех сущностей БД, и реализующий функции чтения и записи данных), добавим конструктор с таким же параметром и запишем параметр в поле. Данное поле будет добавлено во все контроллеры приложения.

В представлении Index добавим форму авторизации с полями для ввода логина и пароля и укажем элементу form атрибут method=”Post”. Для обработки формы добавим в контроллер действие Index с параметрами, передаваемыми из формы, и отметим его как «HttpPost».

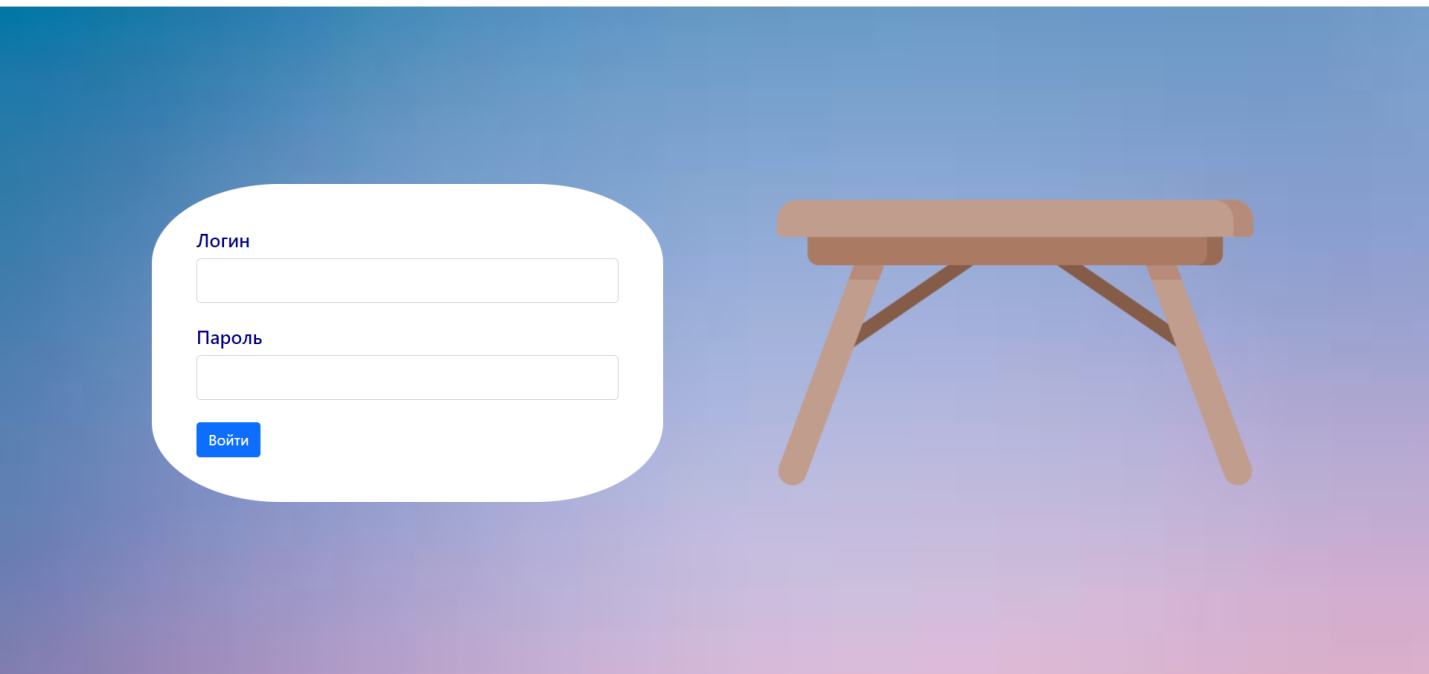


Рис. 1. Страница авторизации.

## Авторизация пользователей

Обработка формы авторизации проводит поиск пользователя по введенным логину и паролю среди всех пользователей с помощью метода FirstOrDefault. Поскольку пароль хранится в БД в зашифрованном виде, добавим в папку Models проекта статический класс Helper для реализации методов, необходимых в разных частях приложения.

В класс Helper добавим статический метод CreateMD5 для шифрования введенного на форме авторизации пароля по алгоритму MD5.

Для хранения данных авторизованного пользователя на время работы приложения запишем его данные, найденные в коллекции пользователей, в сессию.

Для работы с сессией добавим в файл Program.cs строку «builder.Services.AddSession(options => { options.IdleTimeout = TimeSpan.FromHours(1);});». Свойство IdleTimeout определяет время хранения сессии в памяти если пользователь не совершает никаких действий.

## Ограничение доступа неавторизованных пользователей

Доступ к функционалу веб-приложения может осуществляться не только через интерфейс html-страниц, но и через адресную строку браузера. Для того чтобы пользователь не мог обращаться к функциям приложения без авторизации через адресную строку создадим метод ShowPage в классы Helper и передадим ему текущую сессию приложения и имя запрашиваемой страницы. Метод будет возвращать значения true/false в зависимости от доступности функционала пользователю и перенаправлять на страницу авторизации или иную доступную страницу.

Данных метод будет вызываться во всех действиях контроллеров.

## Страница администратора

После успешной авторизации пользователь с ролью «Администратор» автоматически перенаправляется на страницу администратора. На странице будут отображаться данные пользователей и информация об учебных заведениях. В действии Index запишем список пользователей и учебных заведений из БД в соответствующие переменные «users» и «institutuions» с помощью метода ToListAsync у коллекций Users и Institutions, доступных в поле \_context, для включения связанных сущностей будет использован метод Include с указанием поля представляющего связанную сущность. Поскольку представления могут иметь только один тип модели, создадим переменную анонимного класса и запишем в свойства с соответствующими названиями переменные «users» и «institutuions» и передадим как модель представления в методе View.

Для действия Index создадим представление Index.cshtml. При создании модели укажем тип «пустое представление», поскольку передаваемая модель не является строго типизированным классом, определенным в проекте.

Для отображения данных модели в представлении создадим в папке Shared частичное представление для каждого из свойств модели и выведем их с помощью метода «Html.RenderPartial» указав имя представления, модель и дополнительные параметры. В частичных представлениях выведем данные в виде таблице с кнопками действий добавления, удаления и редактирования.

Для каждой кнопки действия в представлении Index добавим соответствующие формы, которые будут отображаться при нажатии кнопки с помощью стилей CSS и заполняться при необходимости с помощью javascript функций, вызываемых при срабатывании события onclick.

Все используемые javascript функции будут определены в файле site.js проекта. Для каждой формы добавим в контроллер Administration действия с параметрами, идентичными указанным на формах и атрибутом HttpPost.

Формы добавления и редактирования для каждой сущности объединим в одну форму, для определения необходимого действия добавления или редактирования в поле идентификатора на форме с помощью функции javascipt будет добавлен идентификатор из модели представления или значение «-1».

Для полей форм с выбором из списка в действии Index добавим в коллекцию ViewBag необходимые списки типа SelectList и привяжем с помощью html-атрибута asp-items в представлении.

Список доступных действий:

* Для сущности Пользователь:
  + Добавление;
  + Смена пароля;
  + Добавление учебного заведения;
  + Удаление;
* Для сущности Учебное заведение:
  + Добавление;
  + Редактирование названия;
  + Удаление;
  + Добавление специальности/направления;
  + Редактирование специальности/направления;
  + Переход к просмотру специальности/направления;
  + Удаление специальности/направления.

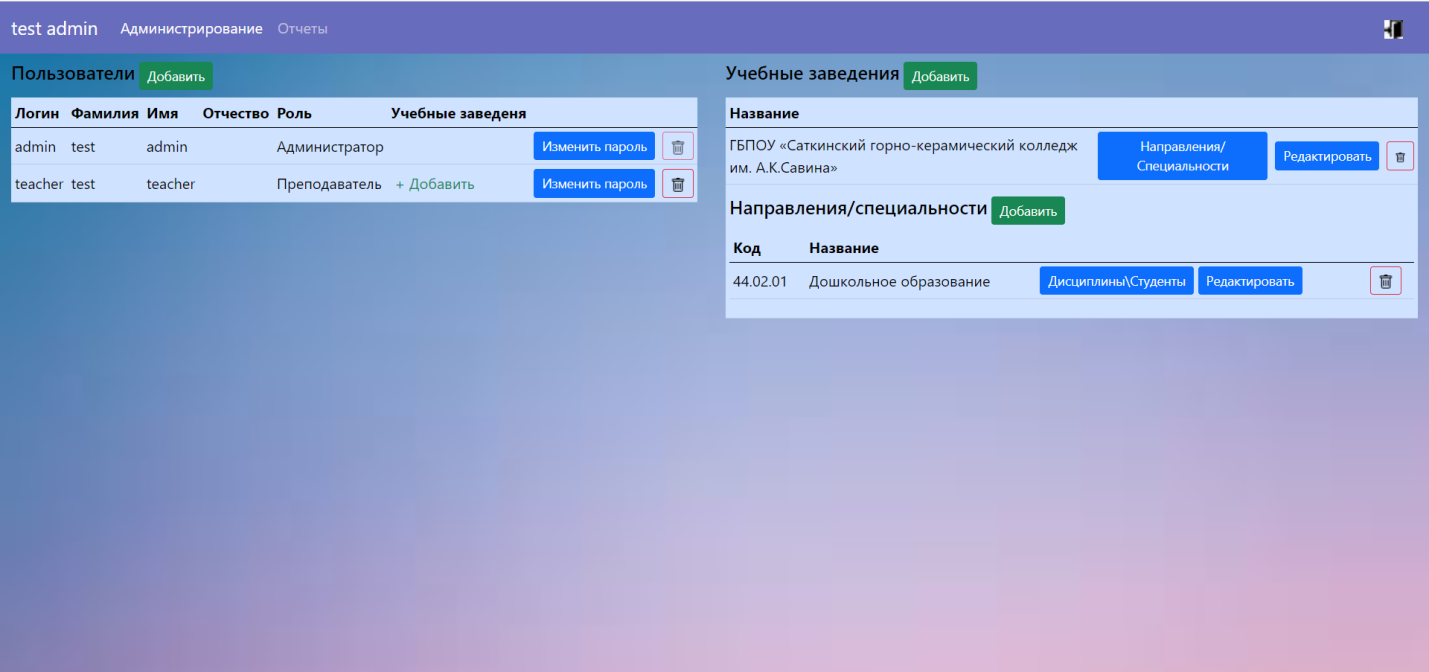


Рис. 2. Страница администрирования

## Страница специальности

На странице специальности будет отображаться следующая информация:

* Список модулей специальности;
* Список практик;
* Список студентов.

Поскольку страница является подразделом страницы администратора, для нее не будет создан отдельный контроллер, а добавлено действие Speciality в контроллер Administration с параметром id, переданным при нажатии кнопки просмотра специальности.

В действии Speciality с помощью метода Where, указав переданный параметр id, выберем и запишем в соответствующие переменные список модулей специальностей, практик и студентов, запишем переменные в свойства анонимного класса и передадим в метод View.

Аналогично основной странице администратора для действия Speciality добавим представление с пустой моделью, а для каждого списка добавим частичные представления с таблицами и кнопками действий.

Список возможный действий для каждой сущности на странице специальности:

* Для сущности Модуль дисциплины:
  + Добавление;
  + Редактирование кода и названия;
  + Удаление;
* Для сущности Практика:
  + Добавление;
  + Редактирование;
  + Удаление;
* Для сущности студент:
  + Добавление;
  + Редактирование;
  + Удаление.

Для каждого действия в представлении Speciality добавим формы и действия в контроллере, и в действии Speciality добавим необходимые списки для выбора в коллекцию ViewBag.

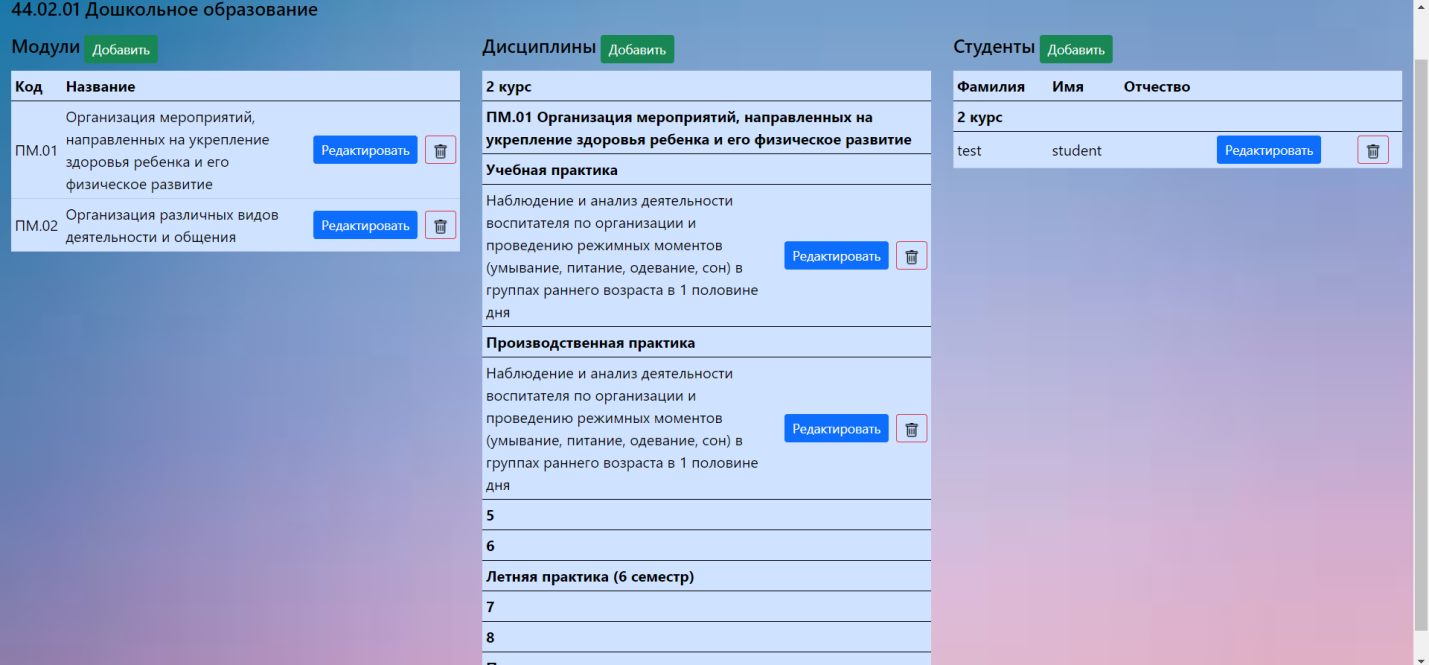


Рис. 3. Страница специальности.

## Страница успеваемости

После авторизации пользователя с ролью «Преподаватель», автоматически будет открыта страница успеваемости. Формирование содержимого страницы зависит от выбора параметров на форме в верхней части страницы.

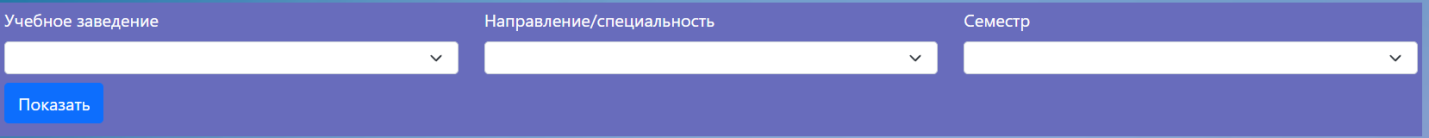


Рис. 4. Форма выбора параметров.

Список учебных заведений, доступных пользователю в зависимости от идентификатора и роли, хранящихся в сессии приложения, будет передан в коллекции ViewBag из действия Index контроллера Examination.

При выборе значения из списка учебных заведений будет вызвана javascript функция из файла site.js для заполнения списка специальностей с помощью вызова действия в контроллере, переданного в параметрах функции. Аналогично заполняется список семестров при выборе специальности.

После выбора параметров и нажатия кнопки «Показать» будет повторно вызвано действие Index, но уже с параметрами указанными на форме. В модели с помощью методов Include и Where будет передан список оценок по практикам за выбранный семестр по выбранной специальности для студентов, обучающихся на соответствующем семестру курсе или список практик, если студенты не были ранее добавлены на странице специальности.

В представлении Index контроллера Examination с помощью цикла foreach и метода GroupBy у модели сгруппируем оценки по модулям дисциплин и видам практик. Для получившегося сгруппированного набора создадим частичное представление Examination и выведем с помощью метода Html.RenderPartialAsync.

Для вывода оценки в представлении Examination создадим частичное представление Mark с полем отображения оценки и скрытым полем редактирования и добавим его в представление Examination с помощью html тэга partial, указав имя представления и модель.

Режим редактирования оценки вызывается с помощью javascript функции beginEdit при двойном нажатии левой кнопкой мыши по полю отображения.

Сохранение изменения оценки происходит с помощью javascript функции endEdit, вызываемой при снятии фокуса с поля ввода.

При отсутствии предварительно добавленных студентов их можно добавить прямо в таблицу успеваемости, нажав Новый рисунок (6).bmp в любой из таблиц успеваемости.

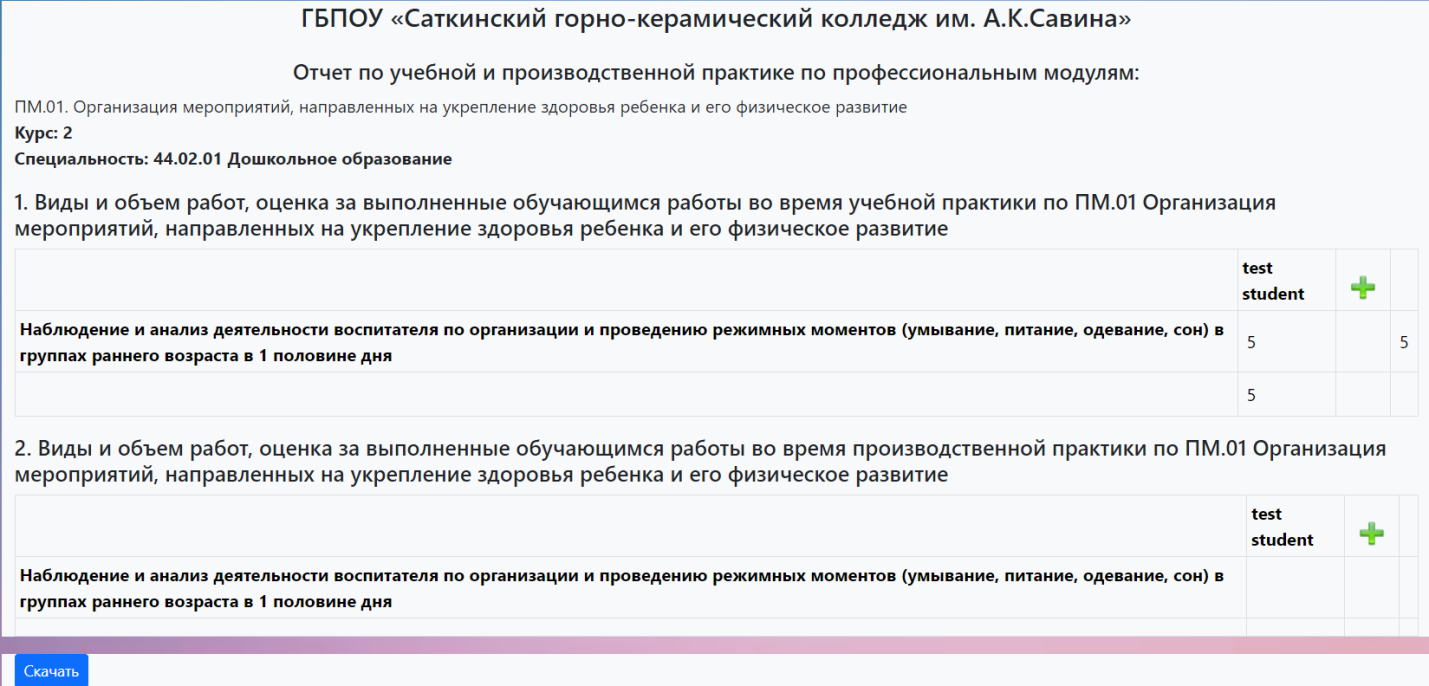


Рис. 5. Страница успеваемости.

## Скачивание отчета

Для скачивания отчета в формате pdf на странице успеваемости необходимо нажать кнопку «Скачать» внизу страницы.

Скачиваемый отчет формируется с помощью открытого генератора отчетов FastReport OpenSource, добавленного в проект с помощью менеджера пакетов NuGet. Макет отчета создан с помощью дизайнера отчетов FastReport Community и хранится в папке проекта.

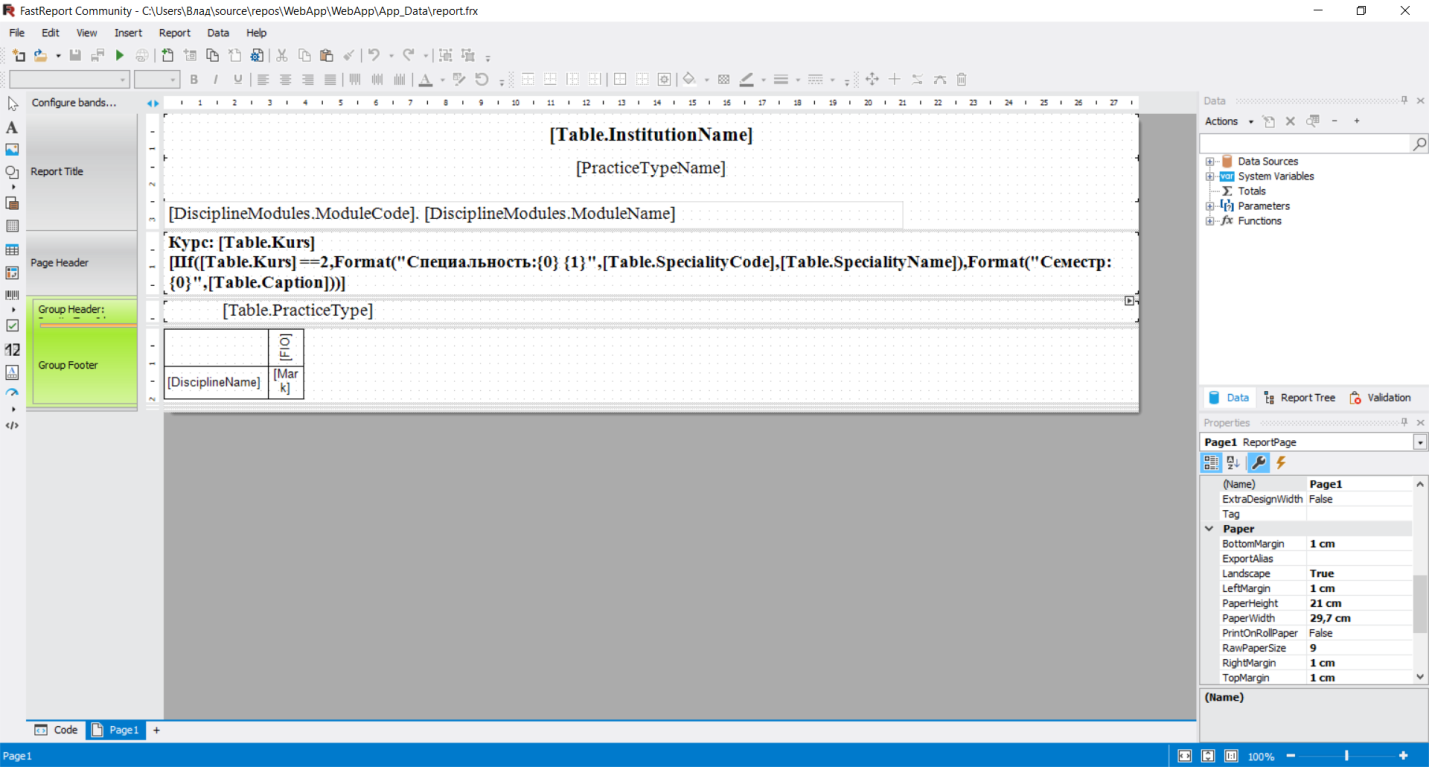


Рис. 6. Макет отчета успеваемости

Нажатие кнопки «Скачать» вызывает действие Export контроллера Examination с параметрами, выбранными на форме страницы успеваемости.

Действие Export передает в макет отчета параметры и строку подключения к базе данных, после чего отчет будет подготовлен и сохранен в формат pdf для скачивания.

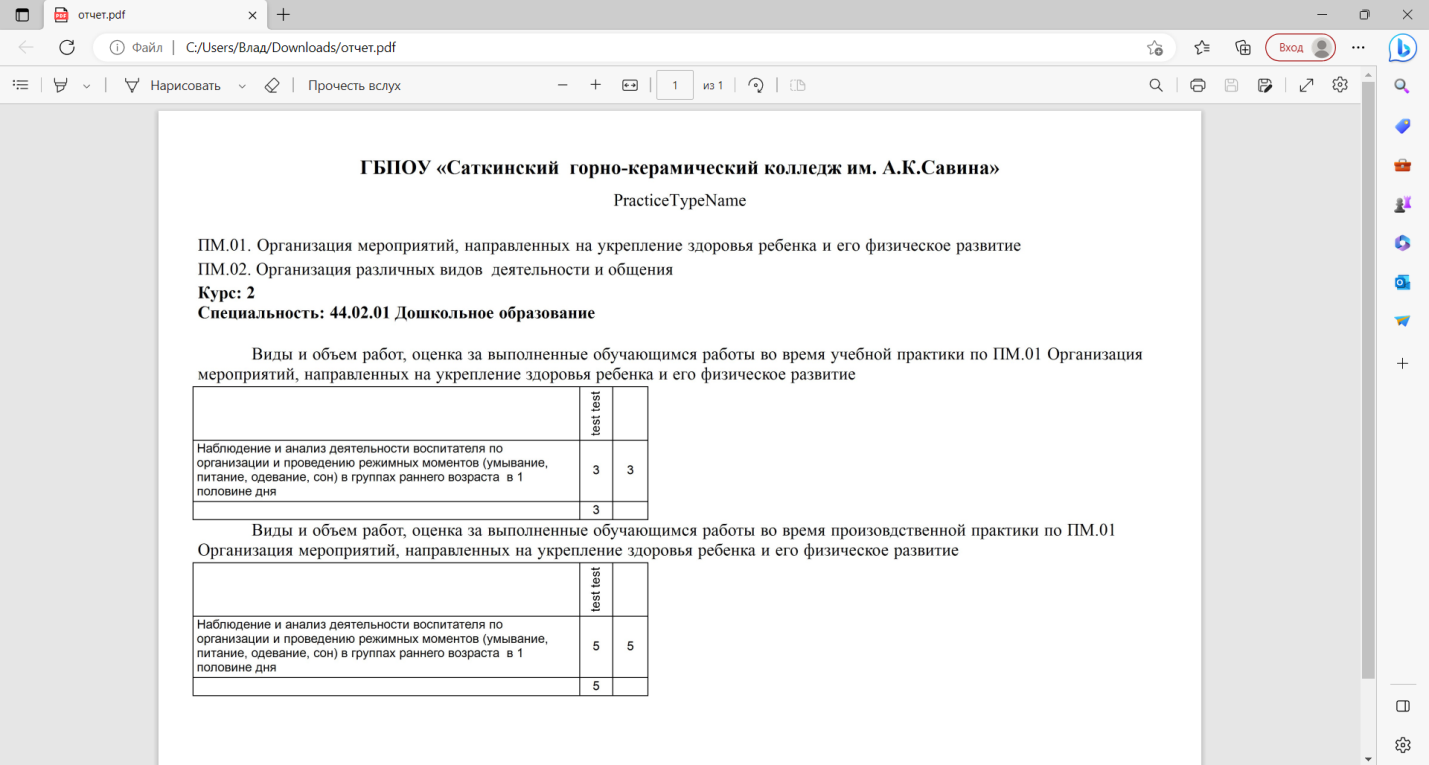


Рис. 7. Pdf-файл отчета успеваемости.

## Публикация и хостинг приложения

Публикация приложения для развертывания на сервере происходит с помощью функции «Опубликовать» Visual Studio. Профиль публикации зависи от целевой платформы развертывания приложения. Для демонстрации проекта проект будет опубликован как папка с файлами для загрузки на сервер хостинга. Для развертывания приложения необходимы следущие условия:

* Веб-сервер с поддержкой приложений на платформе .net 6.0 (пакет поддержки для ОС Windows и Linux доступен на сайте Microsoft);
* Microsoft SQL Server 2016 и выше;
* Настройка подключения к серверу в файле конфигурации приложения;
* Регистрация домена при необходимости работы по сети интернет.

Приложение

Общий макет страниц приложения \_Layout.cshtml

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

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

<title>@ViewData["Title"]</title>

<**link** rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<**link** rel="stylesheet" **href**="~/css/site.css" **asp-append-version**="true" />

<**link** rel="stylesheet" **href**="~/WebApp.styles.css" **asp-append-version**="true" />

</head>

<body>

@if (@Context.Session.TryGetValue("userId", out \_))

{

<nav class="navbar navbar-expand-lg navbar-dark menu">

<div class="container-fluid">

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<label class="navbar-brand mt-2 mt-lg-0">

@Context.Session.GetString("userName")

</label>

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

@switch (Context.Session.GetInt32("userRole"))

{

case 1:

{

<li class="nav-item">

<a class="@string.Format("nav-link {0}", ViewContext.RouteData.Values["controller"].ToString().Equals("Administration") ? "active" : "")" href="@Url.Action("","Administration")">Администрирование</a>

</li>

<li class="nav-item">

<a class="@string.Format("nav-link {0}", ViewContext.RouteData.Values["controller"].ToString().Equals("Examination") ? "active" : "")" href="@Url.Action("","Examination")">Отчеты</a>

</li>

}

break;

default:

{

<li class="nav-item">

<a class="@string.Format("nav-link {0}", ViewContext.RouteData.Values["controller"].ToString().Equals("Examination") ? "active" : "")" href="@Url.Action("","Examination")">@string.Format("{0}", Context.Session.GetInt32("userRole") == 1 ? "Отчеты" : "Успеваемость")</a>

</li>

}

break;

}

</ul>

</div>

<div class="d-flex align-items-center">

<div class="dropdown">

<a class="dropdown-item" href="@Url.Action("LogOut", "Home")">

<img width="20" height="20" src="/images/exit\_icon.png" />

</a>

</div>

</div>

</div>

</nav>

}

@RenderBody()

<**script** src="~/lib/jquery/dist/jquery.min.js"></**script**>

<**script** src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></**script**>

<**script** **src**="~/js/site.js" **asp-append-version**="true"></**script**>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

Функции javascript из файла site.js

//заполнение формы добавления/редактирования учебных заведений

function readyInstForm(name, id) {

$("#instId").val(id);

$("#instName").val(name);

if (id != -1) {

$("#instModalSubmit").text("Сохранить");

}

else {

$("#instModalSubmit").text("Добавить");

}

}

//заполнение формы добавления/редактирования направлений/специальностей

function readySpecForm(name, code, id, instId) {

$("#specId").val(id);

$("#specInstId").val(instId);

$("#specCode").val(code);

$("#specName").val(name);

if (id != -1) {

$("#specModalSubmit").text("Сохранить");

}

else {

$("#specModalSubmit").text("Добавить");

}

}

//установка значения поля

function setField(name, value) {

$(name).val(value);

}

//заполнение формы удаления записи

function setRecordForDeletion(type, id) {

$("#recordType").val(type);

$("#recordId").val(id);

}

//заполнение формы добавления/редактирования модулей дисциплин

function readyModuleForm(id, spec, code, name) {

$("#moduleId").val(id);

$("#specId").val(spec);

$("#moduleCode").val(code);

$("#moduleName").val(name);

if (id != -1) {

$("#moduleModalSubmit").text("Сохранить");

}

else {

$("#moduleModalSubmit").text("Добавить");

}

}

//заполнение формы добавления/редактирования дисциплин

function readyDiscForm(id, module, name, sem) {

$("#discId").val(id);

$("#mId").val(module);

$("#discName").val(name);

$("#semester").val(sem).change();

if (id != -1) {

$("#practiceSelect").hide();

$("#semesterSelect").hide();

$("#studyPractice").removeAttr("required");

$("#workPractice").removeAttr("required");

$("#discModalSubmit").text("Сохранить");

}

else {

$("#practiceSelect").show();

$("#semesterSelect").show();

$("#studyPractice").attr("required", true);

$("#workPractice").attr("required", true);

$("#discModalSubmit").text("Добавить");

}

}

//изменение выбранного семестра на форме добавления/редактирования дисциплин

function semSelectChanged(v) {

if (v == 7) {

$("#practiceSelect").hide();

$("#studyPractice").prop("checked", false);

$("#workPractice").prop("checked", false);

$("#studyPractice").removeAttr("required");

$("#workPractice").removeAttr("required");

}

else {

$("#practiceSelect").show();

$("#studyPractice").attr("required", true);

$("#workPractice").attr("required", true);

}

}

//отметка видов практики на форме добавления/редактирования дисциплин

function checkBoxClick(x) {

if (x.checked) {

$("#studyPractice").removeAttr("required");

$("#workPractice").removeAttr("required");

}

else {

if (!$("#studyPractice").is(':checked') && !$("#workPractice").is(':checked')) {

$("#studyPractice").attr("required", true);

$("#workPractice").attr("required", true);

}

}

}

//заполнение формы добавления/редактирования студентов

function readyStudentForm(id, spec, lastName, firstName, middleName, kurs) {

$("#studId").val(id);

$("#studSpec").val(spec);

$("#lastName").val(lastName);

$("#firstName").val(firstName);

$("#middleName").val(middleName);

$("#kurs").val(kurs).change();

if (id != -1) {

$("#studentModalSubmit").text("Сохранить");

}

else {

$("#studentModalSubmit").text("Добавить");

}

}

//Заполнение списков специальностей и семестров на форме формирования страницы успевамости

function selectChanged(target, action, value) {

//вызов действия контроллера

$.ajax({

type: "GET",

url: "/Examination/" + action,

data: { Id: value },

//обработка результата

success: function (data) {

var s = '<option value=""></option>';

for (var i = 0; i < data.length; i++) {

s += '<option value="' + data[i].id + '">' + data[i].name + '</option>';

}

$("#" + target).html(s);

$("#" + target).change();

}

});

}

//функция перехода в режим редактирования оценки

function beginEdit(id) {

$("#edit" + id + " > input").val($("#label" + id).text());

$("#label" + id).hide();

$("#edit" + id).show();

$("#edit" + id + " > input").focus();

}

//функция сохранения изменений оценки и обновления таблиц успеваемости

function endEdit(id) {

//проверка на числовое или пустое значение

if (!isNaN($("#edit" + id + " > input").val()) || $("#edit" + id + " > input").val() == null) {

//вызов действия контроллера

$.ajax({

type: "POST",

url: "/Examination/SaveMark",

data: { Id: id, Mark: $("#edit" + id + " > input").val() },

//обработка результата

success: function (data) {

$("#label" + id).text($("#edit" + id + " > input").val());

//обновление ячейки среднего балла по дисциплине

if (data.avgD > 0) {

$("#avgD" + data.disc).text(Math.round(data.avgD));

}

else {

$("#avgD" + data.disc).text("");

}

//обновление ячейки среднего балла по студенту

if (data.avgS > 0) {

$("#avgS" + data.id + "-avgDM" + data.dm + "-avgPr" + data.pr).text(Math.round(data.avgS));

}

else {

$("#avgS" + data.id + "-avgDM" + data.dm + "-avgPr" + data.pr).text("");

}

},

//обработка ошибки при вызове

fail: function () {

alert('Ошибка!');

}

});

}

else {

alert('Введенное значение не является допустимым!');

$("#edit" + id + " > input").val('');

}

$("#label" + id).show();

$("#edit" + id).hide();

}

//функция добавления студента в таблице успеваемости

function addStudent() {

//вызов действия контроллера

$.ajax({

type: "POST",

url: "/Examination/AddStudent",

data: { lastName: $("#lastName").val(), firstName: $("#firstName").val(), middleName: $("#middleName").val(), kurs: $("#kurs").val(), spec: $("#studSpec").val() },

//обработка результата

success: function (data) {

if (data != null) {

//добавление столбца с ФИО студента во все таблицы

$("table").each(function () {

$("<th>" + data.fio + "</th>").insertBefore($("#" + this.id + " > thead > tr > th.last"));

})

data.exams.forEach(ex => {

//добавление ячейки среднего балла по студенту

if ($("#avgS" + data.id + "-avgDM" + ex.moduleId + "-avgPr" + ex.practiceTypeId).length == 0) {

$("<td><div id=\"avgS" + data.id + "-avgDM" + ex.moduleId + "-avgPr" + ex.practiceTypeId + "\" class=\"p-1\"></div></td>").insertBefore($("#DM" + ex.moduleId + "-Pr" + ex.practiceTypeId + " > tfoot > tr > td.last"));

}

//добавление ячейки с оценкой

$("<td id=\"ex" + ex.id + "\"></td>").insertBefore($("#DM" + ex.moduleId + "-Pr" + ex.practiceTypeId + " > tbody > tr#D" + ex.disciplineId + " > td.last"));

$.ajax({

type: "GET",

url: "/Examination/Mark",

data: { id: ex.id },

success: function (response) {

$("#ex"+ex.id).html(response);

}

});

});

}

},

//обработка ошибки при вызове

fail: function () {

alert('Ошибка!');

}

});

Контроллер Home

public class HomeController : Controller

{

private readonly DContext \_context;

public HomeController(DContext context)

{

\_context = context;

}

public IActionResult Index()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Index(string login, string password)

{

try

{

var user = await \_context.Users.FirstOrDefaultAsync(u => u.Login.Equals(login) && u.Password == Models.Helper.CreateMD5(password));

if (user != null)

{

HttpContext.Session.SetInt32("userId", user.Id);

HttpContext.Session.SetInt32("userRole", user.RoleId);

HttpContext.Session.SetString("userName", string.Format("{0} {1} {2}", user.LastName, user.FirstName, user.MiddleName));

switch (user.RoleId)

{

case 1:

return RedirectToAction("", "Administration");

case 2:

return RedirectToAction("", "Examination");

}

}

else

{

ViewBag.Error = true;

ViewBag.ErrorMessage = "Неверный логин или пароль";

}

}

catch(Exception ex)

{

ViewBag.ErrorMessage = ex.Message;

ViewBag.Error = true;

}

return View();

}

public IActionResult LogOut()

{

HttpContext.Session.Clear();

return RedirectToAction("Index");

}

}

Представление Index контроллера Home

@{

ViewData["Title"] = "Вход в систему";

}

<section class="vh-100">

<div class="container py-5 h-100">

<div class="row d-flex align-items-center justify-content-center h-100">

<div class="col-md-7 col-lg-6 col-xl-5 bg-white p-5 rounded-x">

<**form** method="post">

@if (ViewBag.Error != null && ViewBag.Error)

{

<div class="form-outline mb-4">

<label class="form-label w-100 text-center h4 text-danger h3">@ViewBag.ErrorMessage</label>

</div>

}

<div class="form-outline mb-4">

<label class="form-label h5 text-menu-color">Логин</label>

<input type="text" name="login" class="@string.Format("form-control form-control-lg {0}", ViewBag.Error != null ? ViewBag.Error ? "border border-danger" : "" : "")" required />

</div>

<div class="form-outline mb-4">

<label class="form-label h5 text-menu-color">Пароль</label>

<input name="password" type="password" class="@string.Format("form-control form-control-lg {0}", ViewBag.Error != null ? ViewBag.Error ? "border border-danger" : "" : "")" required />

</div>

<button type="submit" class="btn btn-primary btn-block">Войти</button>

</**form**>

</div>

<div class="col-md-7 col-lg-5 col-xl-5 offset-xl-1 ">

<img src="/images/desk.png"

class="img-fluid">

</div>

</div>

</div>

</section>

Контроллер Administration

public class AdministrationController : Controller

{

private readonly DContext \_context;

private readonly string pageName = "Administration";

public AdministrationController(DContext context)

{

\_context = context;

}

public async Task<IActionResult> Index()

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

//список пользователей

var users = await \_context.Users.Include(u => u.InstitutionAssignments).ThenInclude(u => u.Institution).Include(u => u.Role).ToListAsync();

//список учебных заведений

var institutions = await \_context.Institutions.Include(i => i.Specialities).ToListAsync();

//анонимный класс для передачи в представление без строго типизированной модели

var model = new { Users = users, Institutions = institutions };

//сообщение об ошибке при добавлении пользователя

ViewBag.NewUserErrorMessage = TempData["NewUserErrorMessage"]?.ToString();

//сообщение об ошибке при добавлении учебного заведения

ViewBag.InstitutionErrorMessage = TempData["InstitutionErrorMessage"]?.ToString();

//сообщение об ошибке при добавлении направления/специальности учебному заведения

ViewBag.SpecialityErrorMessage = TempData["SpecialityErrorMessage"]?.ToString();

ViewBag.InstitutionId = TempData["InstitutionId"]?.ToString();

ViewBag.KeepSectionOpen = TempData["KeepSectionOpen"]?.ToString();

//список учебных заведений для формы завкрепления уч.з. за пользователем

ViewBag.Institutions = new SelectList(\_context.Institutions.ToList(), "Id", "InstitutionName");

//список ролей для формы создания пользователя

ViewBag.Roles = new SelectList(\_context.Roles.ToList(), "Id", "RoleName");

return View(model);

}

[HttpPost]

public async Task<IActionResult> AddInstitutionAssignment(int id, int institutionId)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

//проверка наличия уч.з. в закрепленных за пользователем перед добавлением

if (\_context.InstitutionAssignments.FirstOrDefault(ia => ia.UserId == id && ia.InstitutionId == institutionId) == null)

{

\_context.InstitutionAssignments.Add(new InstitutionAssignment() { UserId = id, InstitutionId = institutionId });

await \_context.SaveChangesAsync();

}

return RedirectToAction("");

}

[HttpPost]

public async Task<IActionResult> Delete(string type, int id)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

//поиск типа данных

var t = Type.GetType(string.Format("WebApp.Models.{0}", type));

if (t != null)

{

//поиск записи с указанным типом и идентификатором

var x = await \_context.FindAsync(t, id);

if (x != null)

\_context.Remove(x);

await \_context.SaveChangesAsync();

}

//возврат к предыдущей странице

return Redirect(Request.Headers["Referer"].ToString());

}

[HttpPost]

public async Task<IActionResult> ChangePassword(int id, string password)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

var user = await \_context.Users.FindAsync(id);

if (user != null)

{

user.Password = Helper.CreateMD5(password);

await \_context.SaveChangesAsync();

}

return RedirectToAction("");

}

[HttpPost]

public async Task<IActionResult> EditInstitution(int id, string name)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

//проверка наличия учебного заведения с указанным названием

if (await \_context.Institutions.FirstOrDefaultAsync(x => x.InstitutionName.Equals(name)) != null)

TempData["InstitutionErrorMessage"] = "Учебное заведение с таким названием уже существует";

else

{

if (id == -1)//новое учебное заведение

await \_context.Institutions.AddAsync(new Institution() { InstitutionName = name });

else

{

var inst = await \_context.Institutions.FindAsync(id);

if (inst != null)

inst.InstitutionName = name;

}

}

await \_context.SaveChangesAsync();

return RedirectToAction("");

}

[HttpPost]

public async Task<IActionResult> EditInstitution(int id, string name)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

//проверка наличия учебного заведения с указанным названием

if (await \_context.Institutions.FirstOrDefaultAsync(x => x.InstitutionName.Equals(name)) != null)

TempData["InstitutionErrorMessage"] = "Учебное заведение с таким названием уже существует";

else

{

if (id == -1)//новое учебное заведение

await \_context.Institutions.AddAsync(new Institution() { InstitutionName = name });

else

{

var inst = await \_context.Institutions.FindAsync(id);

if (inst != null)

inst.InstitutionName = name;

}

}

await \_context.SaveChangesAsync();

return RedirectToAction("");

}

[HttpPost]

public async Task<IActionResult> EditSpeciality(int id, int instId, string code, string name)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

TempData["KeepSectionOpen"] = instId.ToString();

//проверка ниличия направления/специальности перед добавлением

if (await \_context.Specialities.FirstOrDefaultAsync(x => (x.SpecialityCode.Equals(code) || x.SpecialityName.Equals(name)) && x.InstitutionId.Equals(instId)) != null)

{

TempData["InstitutionId"] = instId.ToString();

TempData["SpecialityErrorMessage"] = "Направление/специальность с таким кодом и названием уже существует";

}

else

{

if (id == -1)//новое направление/специальность

await \_context.Specialities.AddAsync(new Speciality() { InstitutionId = instId, SpecialityCode = code, SpecialityName = name });

else

{

var spec = await \_context.Specialities.FindAsync(id);

if (spec != null)

{

spec.SpecialityName = name;

spec.SpecialityCode = code;

}

}

}

await \_context.SaveChangesAsync();

return RedirectToAction("");

}

[HttpPost]

public async Task<IActionResult> AddUser(string login, string password, string lastName, string firstName, string middleName, int role)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

//проверка занятости логина или начилия пользователя с указнными ФИО

if (\_context.Users.FirstOrDefault(u => u.Login.Equals(login) || (u.LastName.Equals(lastName) && u.FirstName.Equals(firstName) && ((string.IsNullOrEmpty(u.MiddleName) && string.IsNullOrEmpty(middleName)) || u.MiddleName.Equals(middleName)))) != null)

{

TempData["NewUserErrorMessage"] = "Пользователь с таким логином или ФИО уже существует!";

}

else

{

var newUser = new User() { Login = login, LastName = lastName, FirstName = firstName, MiddleName = middleName, RoleId = role, Password = Helper.CreateMD5(password) };

\_context.Add(newUser);

await \_context.SaveChangesAsync();

}

return RedirectToAction("");

}

public async Task<IActionResult> Speciality(int id)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

//список студентов

var students = await \_context.Students.Where(s => s.SpecialityId == id).ToListAsync();

//список модулей дисциплин

var modules = await \_context.DisciplineModules.ToListAsync();

//список дисциплин

var disciplines = await \_context.Disciplines.Where(d => d.Module.SpecialityId == id).Include(d => d.PracticeType).ToListAsync();

//Направление/специальность для отображения

var sp = await \_context.Specialities.FirstOrDefaultAsync(x => x.Id == id);

ViewBag.Speciality = string.Format("{0} {1}", sp?.SpecialityCode, sp?.SpecialityName);

ViewBag.SpecialityId = sp?.Id;

//Семестры для таблицы

ViewBag.Semesters = await \_context.Semesters.OrderBy(s => s.Id).ToListAsync();

//Ошибка добавления модуля

ViewBag.ModuleError = TempData["ModuleError"]?.ToString();

//Ошибка добавления дисциплины

ViewBag.DiscError = TempData["DiscError"]?.ToString();

//Ошибка добавления студента

ViewBag.StudentError = TempData["StudentError"]?.ToString();

//Список семестров для формы добавления дисциплиеы

ViewBag.SemestersSelect = new SelectList(await \_context.Semesters.ToListAsync(), "Id", "Caption");

//Список модулей для формы добавления дисциплиеы

ViewBag.ModulesSelect = new SelectList(await \_context.DisciplineModules.Where(dm => dm.SpecialityId == id).ToListAsync(), "Id", "ModuleName");

//анонимный класс для передачи в представление без строго типизированной модели

var model = new { Disciplines = disciplines, Modules = modules, Students = students };

return View(model);

}

[HttpPost]

public async Task<IActionResult> EditModule(int id, int spec, string code, string name)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

//проверка наличия модуля с кодом и названием

if (await \_context.DisciplineModules.FirstOrDefaultAsync(x => x.SpecialityId == spec && (x.ModuleCode.Equals(code) || x.ModuleName.Equals(name))) != null)

TempData["ModuleError"] = "Модуль с таким кодом или названием существует";

else

{

if (id == -1)//новый модуль

await \_context.DisciplineModules.AddAsync(new DisciplineModule() { SpecialityId = spec, ModuleCode = code, ModuleName = name });

else

{

var dm = await \_context.DisciplineModules.FindAsync(id);

if (dm != null)

{

dm.ModuleCode = code;

dm.ModuleName = name;

}

}

}

await \_context.SaveChangesAsync();

return Redirect(Request.Headers["Referer"].ToString());

}

[HttpPost]

public async Task<IActionResult> EditDiscipline(int id, int module, string name, int semester, bool studyPractice, bool workPractice)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

string errorMessage = "";

//проверка наличия дисциплины

if (id == -1)//новая дисциплина

{

var students = await \_context.Students.Where(s => s.SpecialityId == \_context.DisciplineModules.Find(module).SpecialityId).ToListAsync();

if (studyPractice)//учебная практика

{

if (await \_context.Disciplines.FirstOrDefaultAsync(x => x.ModuleId == module && x.DisciplineName.Equals(name) && x.Semester.Equals(semester) && x.PracticeTypeId == 1) != null)

errorMessage += "Дисциплина с таким названием и видом практики \"учебная\" в данном семестре существует\r\n";

else

{

var d = new Discipline() { ModuleId = module, DisciplineName = name, Semester = semester, PracticeTypeId = 1 };

foreach (var student in students)

{

d.Examinations.Add(new Models.Examination() { Student = student });

}

await \_context.Disciplines.AddAsync(d);

}

}

if (workPractice)//производственная практика

{

if (await \_context.Disciplines.FirstOrDefaultAsync(x => x.ModuleId == module && x.DisciplineName.Equals(name) && x.Semester.Equals(semester) && x.PracticeTypeId == 2) != null)

errorMessage += "Дисциплина с таким названием и видом практики \"производственная\" в данном семестре существует\r\n";

else

{

var d = new Discipline() { ModuleId = module, DisciplineName = name, Semester = semester, PracticeTypeId = 2 };

foreach (var student in students)

{

d.Examinations.Add(new Models.Examination() { Student = student });

}

await \_context.Disciplines.AddAsync(d);

}

}

if (!studyPractice && !workPractice)

{

if (await \_context.Disciplines.FirstOrDefaultAsync(x => x.ModuleId == module && x.DisciplineName.Equals(name) && x.Semester.Equals(semester)) != null)

errorMessage += "Дисциплина с таким названием в данном семестре существует\r\n";

else

{

var d = new Discipline() { ModuleId = module, DisciplineName = name, Semester = semester };

foreach (var student in students)

{

d.Examinations.Add(new Models.Examination() { Student = student });

}

await \_context.Disciplines.AddAsync(d);

}

}

}

else

{

var disc = await \_context.Disciplines.FindAsync(id);

if (disc != null)

{

if (await \_context.Disciplines.FirstOrDefaultAsync(x => x.ModuleId == module && x.DisciplineName.Equals(name) && x.Semester.Equals(semester) && x.PracticeTypeId == disc.PracticeTypeId) != null)

errorMessage += "Дисциплина с таким названием и видом практики в данном семестре существует";

else

{

disc.ModuleId = module;

disc.DisciplineName = name;

}

}

}

TempData["DiscError"] = errorMessage;

await \_context.SaveChangesAsync();

return Redirect(Request.Headers["Referer"].ToString());

}

[HttpPost]

public async Task<IActionResult> EditStudent(int id, int spec, string lastName, string firstName, string middleName, int kurs)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

//Проверка наличия студента

if (await \_context.Students.FirstOrDefaultAsync(s => s.SpecialityId.Equals(spec) && s.LastName.Equals(lastName) && s.FirstName.Equals(firstName) && ((string.IsNullOrEmpty(s.MiddleName) && string.IsNullOrEmpty(middleName)) || s.MiddleName.Equals(middleName)) && (s.Kurs.Equals(kurs) || id != -1)) != null)

TempData["StudentError"] = "Студент с такими ФИО на указаном курсе уже существует";

else

{

if (id == -1)//новый студент

{

var newStudent = new Student() { SpecialityId = spec, LastName = lastName, FirstName = firstName, MiddleName = middleName, Kurs = kurs };

var disc = await \_context.Disciplines.Include(d => d.Module).Where(d => d.Module.SpecialityId == spec).ToListAsync();

foreach (var d in disc)

{

newStudent.Examinations.Add(new Models.Examination() { Discipline = d });

}

await \_context.Students.AddAsync(newStudent);

}

else

{

var stud = await \_context.Students.FindAsync(id);

if (stud != null)

{

stud.LastName = lastName;

stud.FirstName = firstName;

stud.MiddleName = middleName;

stud.Kurs = kurs;

}

}

}

await \_context.SaveChangesAsync();

return Redirect(Request.Headers["Referer"].ToString());

}

}

Представление Index контроллера Administration

@{

ViewData["Title"] = "Администрирование";

}

<div class="container-fluid">

<div class="row">

@{

await Html.RenderPartialAsync("Users", Model.Users, ViewData);

await Html.RenderPartialAsync("Institutions", Model.Institutions, ViewData);

}

</div>

</div>

<div>

@{

await Html.RenderPartialAsync("DeleteForm");

}

<div class="modal fade" id="isntitutionAddAssignment" tabindex="-1" role="dialog" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<**form** action="@Url.Action("AddInstitutionAssignment","Administration")" method="post">

<div class="modal-header">

<h5 class="modal-title">Добавить учебное заведение</h5>

<button type="button" class="close" data-bs-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<input id="idField" name="id" type="hidden" value="" required />

<**select** **name**="InstitutionId" class="form-select" **asp-items**="ViewBag.Institutions"></**select**>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Отмена</button>

<button type="submit" class="btn btn-primary">Добавить</button>

</div>

</**form**>

</div>

</div>

</div>

<div class="modal fade" id="changePassword" tabindex="-1" role="dialog" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<**form** action="@Url.Action("ChangePassword","Administration")" method="post">

<div class="modal-header">

<h5 class="modal-title">Смена пароля</h5>

<button type="button" class="close" data-bs-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<input id="userIdField" name="id" type="hidden" value="" required />

<input name="password" class="form-control" type="password" required />

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Отмена</button>

<button type="submit" class="btn btn-primary">Сохранить</button>

</div>

</**form**>

</div>

</div>

</div>

<div class="modal fade" id="newUser" tabindex="-1" role="dialog" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<**form** action="@Url.Action("AddUser","Administration")" method="post">

<div class="modal-header">

<h5 class="modal-title">Новый пользователь</h5>

<button type="button" class="close" data-bs-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body bg-pr">

<div class="form-group">

<label class="control-label">Логин</label>

<input type="text" name="login" class="form-control" required />

</div>

<div class="form-group">

<label class="control-label">Пароль</label>

<input type="password" name="password" class="form-control" required />

</div>

<div class="form-group">

<label class="control-label">Фамилия</label>

<input type="text" name="lastName" class="form-control" />

</div>

<div class="form-group">

<label class="control-label">Имя</label>

<input type="text" name="firstName" class="form-control" required />

</div>

<div class="form-group">

<label class="control-label">Отчество</label>

<input type="text" name="middleName" class="form-control" />

</div>

<div class="form-group">

<label class="control-label">Роль</label>

<**select** **name**="role" class="form-select" **asp-items**="ViewBag.Roles" required>

</**select**>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Отмена</button>

<button type="submit" class="btn btn-primary">Добавить</button>

</div>

</**form**>

</div>

</div>

</div>

<div class="modal fade" id="instModal" tabindex="-1" role="dialog" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<**form** action="@Url.Action("EditInstitution", "Administration")" method="post">

<div class="modal-header">

<h5 class="modal-title">Учебное заведение</h5>

<button type="button" class="close" data-bs-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<input id="instId" name="id" type="hidden" value="" required />

<input type="text" id="instName" name="name" class="form-control" required />

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Отмена</button>

<button id="instModalSubmit" type="submit" class="btn btn-primary"></button>

</div>

</**form**>

</div>

</div>

</div>

<div class="modal fade" id="specModal" tabindex="-1" role="dialog" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<**form** action="@Url.Action("EditSpeciality", "Administration")" method="post">

<div class="modal-header">

<h5 class="modal-title">Направление/специальность</h5>

<button type="button" class="close" data-bs-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<input id="specId" name="id" type="hidden" value="" required />

<input id="specInstId" name="instId" type="hidden" value="" required />

<div class="form-group">

<label class="control-label">Код</label>

<input type="text" id="specCode" name="code" class="form-control" required />

</div>

<div class="form-group">

<label class="control-label">Название</label>

<input type="text" id="specName" name="name" class="form-control" required />

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Отмена</button>

<button id="specModalSubmit" type="submit" class="btn btn-primary"></button>

</div>

</**form**>

</div>

</div>

</div>

</div>

Частичное представление Users

@model IEnumerable<WebApp.Models.User>

<div class="col">

<table class="table table-sm table-primary w-auto align-middle caption-top">

<caption>

<span class="h5 text-black">Пользователи</span>

<button type="button" class="btn btn-success btn-sm" data-bs-toggle="modal" data-bs-target="#newUser">Добавить</button>

@if (ViewBag.NewUserErrorMessage != null)

{

<span class="h5 text-danger">@ViewBag.NewUserErrorMessage</span>

ViewBag.NewUserErrorMessage = null;

}

</caption>

<thead>

<tr>

<th scope="col">Логин</th>

<th scope="col">Фамилия</th>

<th scope="col">Имя</th>

<th scope="col">Отчество</th>

<th scope="col">Роль</th>

<th scope="col">Учебные заведеня</th>

<th scope="col" colspan="2"></th>

</tr>

</thead>

<tbody>

@foreach (var user in Model.OrderBy(x => x.Id))

{

<tr>

<td>

@user.Login

</td>

<td>

@user.LastName

</td>

<td>

@user.FirstName

</td>

<td>

@user.MiddleName

</td>

<td>

@user.Role.RoleName

</td>

<td>

@if (user.RoleId == 2)

{

<table class="table table-sm table-borderless p-0 m-0 align-middle w-auto">

@foreach (InstitutionAssignment assignment in user.InstitutionAssignments.OrderBy(x => x.Id))

{

<tr>

<td>

@assignment.Institution.InstitutionName

</td>

<td>

<button type="button" class="btn btn-sm btn-link" data-bs-toggle="modal" data-bs-target="#deleteConfirmation" onclick="setRecordForDeletion('InstitutionAssignment', @assignment.Id)"><img src="/images/del.jpg" /></button>

</td>

</tr>

}

<tr>

<td colspan="2">

<a class="link-success cursor-pointer text-decoration-none" data-bs-toggle="modal" data-bs-target="#isntitutionAddAssignment" onclick="setField('#idField',@user.Id)">+ Добавить</a>

</td>

</tr>

</table>

}

</td>

<td>

<button type="button" class="btn btn-primary btn-sm" data-bs-toggle="modal" data-bs-target="#changePassword" onclick="setField('#userIdField',@user.Id)">Изменить пароль</button>

</td>

<td>

<button type="button" class="btn btn-outline-danger btn-sm" data-bs-toggle="modal" data-bs-target="#deleteConfirmation" onclick="setRecordForDeletion('User', @user.Id)" @string.Format("{0}", Context.Session.GetInt32("userId")==user.Id ? "disabled" : "")><img src="/images/trash.png" /></button>

</td>

</tr>

}

</tbody>

</table>

</div>

Частичное представление Institutions

@model IEnumerable<WebApp.Models.Institution>

<div class="col">

<table class="table table-sm table-primary w-auto align-middle caption-top">

<caption>

<span class="h5 text-black">Учебные заведения</span>

<button type="button" class="btn btn-success btn-sm" data-bs-toggle="modal" data-bs-target="#instModal" onclick="readyInstForm('', -1)">Добавить</button>

@if (ViewBag.InstitutionErrorMessage != null)

{

<span class="h5 text-danger">@ViewBag.InstitutionErrorMessage</span>

ViewBag.InstitutionErrorMessage = null;

}

</caption>

<thead>

<tr>

<th scope="col">Название</th>

<th scope="col" colspan="4"></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model.OrderBy(x => x.Id))

{

<tr>

<td>

@item.InstitutionName

</td>

<td>

<button type="button" class="btn btn-primary btn-sm" data-bs-toggle="collapse" data-bs-target="@string.Format("#SpecView{0}", item.Id)" aria-expanded="false" aria-controls="@string.Format("SpecView{0}", item.Id)">Направления/Специальности</button>

</td>

<td>

<button type="button" class="btn btn-primary btn-sm" data-bs-toggle="modal" data-bs-target="#instModal" onclick="readyForm('@item.InstitutionName', @item.Id)">Редактировать</button>

</td>

<td>

<button type="button" class="btn btn-outline-danger btn-sm" data-bs-toggle="modal" data-bs-target="#deleteConfirmation" onclick="setRecordForDeletion('Institution', @item.Id)"><img class="img-fluid" src="/images/trash.png" /></button>

</td>

</tr>

<tr id="@string.Format("SpecView{0}", item.Id)" class="@string.Format("collapse {0}", ViewBag.KeepSectionOpen != null ? ViewBag.KeepSectionOpen.ToString()==item.Id.ToString() ? "show" : "" : "")">

<td colspan="4">

<table class="table table-sm table-primary w-100 align-middle caption-top">

<caption>

<span class="h5 text-black">Направления/специальности</span>

<button type="button" class="btn btn-success btn-sm" data-bs-toggle="modal" data-bs-target="#specModal" onclick="readySpecForm('', '', -1, @item.Id)">Добавить</button>

@if (ViewBag.SpecialityErrorMessage != null && ViewBag.InstitutionId == item.Id.ToString())

{

<span class="h5 text-danger">@ViewBag.SpecialityErrorMessage</span>

ViewBag.SpecialityErrorMessage = null;

}

</caption>

<thead>

<tr>

<th scope="col">Код</th>

<th scope="col">Название</th>

<th scope="col" colspan="2"></th>

</tr>

</thead>

<tbody>

@foreach (Speciality spec in item.Specialities.OrderBy(x => x.Id))

{

<tr>

<td>@spec.SpecialityCode</td>

<td>@spec.SpecialityName</td>

<td>

<a class="btn btn-primary btn-sm" href="@Url.Action("Plan", "Administration", new { Id = spec.Id })" role="button">Дисциплины\Студенты</a>

<button type="button" class="btn btn-primary btn-sm" data-bs-toggle="modal" data-bs-target="#specModal" onclick="readySpecForm('@spec.SpecialityName', '@spec.SpecialityCode', @spec.Id, @item.Id)">Редактировать</button>

</td>

<td>

<button type="button" class="btn btn-outline-danger btn-sm" data-bs-toggle="modal" data-bs-target="#deleteConfirmation" onclick="setRecordForDeletion('Speciality', @spec.Id)"><img src="/images/trash.png" /></button>

</td>

</tr>

}

</tbody>

</table>

</td>

</tr>

}

</tbody>

</table>

</div>

Представление Speciality контроллера Administration

@{

ViewData["Title"] = ViewBag.Speciality;

}

<div class="container-fluid">

<h5 class="text-black">@ViewBag.Speciality</h5>

<div class="row">

<div class="col">

@{

await Html.RenderPartialAsync("Modules", Model.Modules, ViewData);

}

</div>

<div class="col">

@{

await Html.RenderPartialAsync("Disciplines", Model.Disciplines, ViewData);

}

</div>

<div class="col">

@{

await Html.RenderPartialAsync("Students", Model.Students, ViewData);

}

</div>

</div>

@{

await Html.RenderPartialAsync("DeleteForm");

}

<div class="modal fade" id="moduleModal" tabindex="-1" role="dialog" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<**form** action="@Url.Action("EditModule", "Administration")" method="post">

<div class="modal-header">

<h5 class="modal-title">Модуль</h5>

<button type="button" class="close" data-bs-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<input id="moduleId" name="id" type="hidden" value="" required />

<input id="specId" name="spec" type="hidden" value="" required />

<div class="form-group">

<label class="form-label">Код</label>

<input type="text" id="moduleCode" name="code" class="form-control" required />

</div>

<div class="form-group">

<label class="form-label">Название</label>

<input type="text" id="moduleName" name="name" class="form-control" required />

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Отмена</button>

<button id="moduleModalSubmit" type="submit" class="btn btn-primary"></button>

</div>

</**form**>

</div>

</div>

</div>

<div class="modal fade" id="discModal" tabindex="-1" role="dialog" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<**form** action="@Url.Action("EditDiscipline", "Administration")" method="post">

<div class="modal-header">

<h5 class="modal-title">Дисциплина</h5>

<button type="button" class="close" data-bs-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<input id="discId" name="id" type="hidden" value="" required />

<div class="form-group">

<label class="form-label">Модуль</label>

<**select** id="mId" class="form-select" **name**="module" value="" **asp-items**="ViewBag.ModulesSelect" required>

</**select**>

</div>

<div class="form-group">

<label class="form-label">Название</label>

<input type="text" id="discName" name="name" class="form-control" required />

</div>

<div id="semesterSelect" class="form-group">

<label class="form-label">Семестр</label>

<**select** id="semester" **name**="semester" class="form-select" **asp-items**="ViewBag.SemestersSelect" onchange="semSelectChanged(this.value)" required >

</**select**>

</div>

<div id="practiceSelect" class="form-group">

<label class="form-label" for="studyPractice">Учебная практика</label>

<input type="checkbox" id="studyPractice" name="studyPractice" value="true" class="form-check-input" onclick="checkBoxClick(this)" required />

<label class="form-label" for="workPractice">Производственная практика</label>

<input type="checkbox" id="workPractice" name="workPractice" value="true" class="form-check-input" onclick="checkBoxClick(this)" required />

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Отмена</button>

<button id="discModalSubmit" type="submit" class="btn btn-primary"></button>

</div>

</**form**>

</div>

</div>

</div>

<div class="modal fade" id="studentModal" tabindex="-1" role="dialog" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<**form** action="@Url.Action("EditStudent", "Administration")" method="post">

<div class="modal-header">

<h5 class="modal-title">Студент</h5>

<button type="button" class="close" data-bs-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<input id="studId" name="id" type="hidden" value="" required />

<input id="studSpec" name="spec" type="hidden" value="" required />

<div class="form-group">

<label class="form-label">Фамилия</label>

<input type="text" id="lastName" name="lastName" class="form-control" required />

</div>

<div class="form-group">

<label class="form-label">Имя</label>

<input type="text" id="firstName" name="firstName" class="form-control" required />

</div>

<div class="form-group">

<label class="form-label">Отчество</label>

<input type="text" id="middleName" name="middleName" class="form-control" />

</div>

<div class="form-group">

<label class="form-label">Курс</label>

<select class="form-select" id="kurs" name="kurs" required>

<**option** **value**="2">2 курс</**option**>

<**option** **value**="3">3 курс</**option**>

<**option** **value**="4">4 курс</**option**>

</select>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Отмена</button>

<button id="studentModalSubmit" type="submit" class="btn btn-primary"></button>

</div>

</**form**>

</div>

</div>

</div>

</div>

Частичное представление Modules

model IEnumerable<WebApp.Models.DisciplineModule>

<table class="table table-sm table-primary w-100 align-middle caption-top">

<caption>

<span class="h5 text-black">Модули</span>

<button type="button" class="btn btn-success btn-sm" data-bs-toggle="modal" data-bs-target="#moduleModal" onclick="readyModuleForm(-1, @ViewBag.SpecialityId, '', '')">Добавить</button>

@if (ViewBag.ModuleError != null)

{

<span class="h5 text-danger">@ViewBag.ModuleError</span>

ViewBag.ModuleError = null;

}

</caption>

<thead>

<tr>

<th>Код</th>

<th>Название</th>

<th colspan="2"></th>

</tr>

</thead>

<tbody>

@foreach (DisciplineModule module in Model)

{

<tr>

<td>@module.ModuleCode</td>

<td>@module.ModuleName</td>

<td>

<button type="button" class="btn btn-primary btn-sm" data-bs-toggle="modal" data-bs-target="#moduleModal" onclick="readyModuleForm(@module.Id, @module.SpecialityId, '@module.ModuleCode', '@module.ModuleName')">Редактировать</button>

</td>

<td>

<button type="button" class="btn btn-outline-danger btn-sm" data-bs-toggle="modal" data-bs-target="#deleteConfirmation" onclick="setRecordForDeletion('DisciplineModule', @module.Id)"><img src="/images/trash.png" /></button>

</td>

</tr>

}

</tbody>

</table>

Частичное представление Disciplines

@model IEnumerable<WebApp.Models.Discipline>

<table class="table table-sm table-primary w-auto align-middle caption-top">

<caption>

<span class="h5 text-black">Дисциплины</span>

<button type="button" class="btn btn-success btn-sm" data-bs-toggle="modal" data-bs-target="#discModal" onclick="readyDiscForm(-1, '', '', '')">Добавить</button>

@if (ViewBag.DiscError != null)

{

<span class="h5 text-danger">@ViewBag.DiscError</span>

ViewBag.DiscError = null;

}

</caption>

@foreach (Semester semester in ViewBag.Semesters)

{

<thead>

<tr>

<th colspan="3">

@string.Format("{0}{1}",semester.Caption, int.TryParse(semester.Caption, out int \_) ? " семестр" : "")

</th>

</tr>

</thead>

@foreach (var module in Model.Where(d => d.Semester == semester.Id).GroupBy(d => d.Module).OrderBy(m => m.Key.Id))

{

<thead>

<tr>

<th colspan="3">

@string.Format("{0} {1}", module.Key.ModuleCode, module.Key.ModuleName)

</th>

</tr>

</thead>

@foreach (var practiceType in module.GroupBy(d => d.PracticeType))

{

@if (practiceType.Key != null)

{

<thead>

<tr>

<th colspan="3">

@practiceType.Key.PracticeTypeName

</th>

</tr>

</thead>

}

<tbody>

@foreach (var disc in practiceType.OrderBy(d => d.Id))

{

<tr>

<td>

@disc.DisciplineName

</td>

<td>

<button type="button" class="btn btn-primary btn-sm" data-bs-toggle="modal" data-bs-target="#discModal" onclick="readyDiscForm(@disc.Id, @disc.ModuleId, '@disc.DisciplineName', @disc.Semester)">Редактировать</button>

</td>

<td>

<button type="button" class="btn btn-outline-danger btn-sm" data-bs-toggle="modal" data-bs-target="#deleteConfirmation" onclick="setRecordForDeletion('Discipline', @disc.Id)"><img src="/images/trash.png" /></button>

</td>

</tr>

}

</tbody>

}

}

}

</table>

Частичное представление Students

@model IEnumerable<WebApp.Models.Student>

<table class="table table-sm table-primary w-100 align-middle caption-top">

<caption>

<span class="h5 text-black">Студенты</span>

<button type="button" class="btn btn-success btn-sm" data-bs-toggle="modal" data-bs-target="#studentModal" onclick="readyStudentForm(-1, @ViewBag.SpecialityId, '', '', '', 0)">Добавить</button>

@if (ViewBag.StudentError != null)

{

<span class="h5 text-danger">@ViewBag.StudentError</span>

ViewBag.StudentError = null;

}

</caption>

<thead>

<tr>

<th>

Фамилия

</th>

<th>

Имя

</th>

<th>

Отчество

</th>

<th colspan="2"></th>

</tr>

</thead>

@foreach (var kurs in Model.GroupBy(s => s.Kurs).OrderBy(k => k.Key))

{

<thead>

<tr>

<th colspan="5">@string.Format("{0} курс", kurs.Key)</th>

</tr>

</thead>

<tbody>

@foreach (var student in kurs)

{

<tr>

<td>@student.LastName</td>

<td>@student.FirstName</td>

<td>@student.MiddleName</td>

<td>

<button type="button" class="btn btn-primary btn-sm" data-bs-toggle="modal" data-bs-target="#studentModal" onclick="readyStudentForm(@student.Id, @student.SpecialityId, '@student.LastName', '@student.FirstName', '@student.MiddleName', @student.Kurs)">Редактировать</button>

</td>

<td>

<button type="button" class="btn btn-outline-danger btn-sm" data-bs-toggle="modal" data-bs-target="#deleteConfirmation" onclick="setRecordForDeletion('Student', @student.Id)"><img src="/images/trash.png" /></button>

</td>

</tr>

}

</tbody>

}

</table>

Контроллер Examination

public class ExaminationController : Controller

{

private readonly string pageName = "Examination";

private readonly DContext \_context;

private readonly IWebHostEnvironment \_environment;

private readonly IConfiguration \_configuration;

public ExaminationController(DContext context, IWebHostEnvironment environment, IConfiguration config)

{

\_context = context;

\_environment = environment;

\_configuration = config;

}

public async Task<IActionResult> Index(int? inst, int? spec, int? sem)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return RedirectToAction("", redirect);

ViewBag.selectFormData = await \_context.Institutions.Include(i => i.InstitutionAssignments).ThenInclude(ia => ia.User).Where(i => i.InstitutionAssignments.FirstOrDefault(x => x.InstitutionId == i.Id && x.UserId == HttpContext.Session.GetInt32("userRole")) != null || HttpContext.Session.GetInt32("userRole")==1).ToListAsync();

ViewBag.ExportError = TempData["ExportError"]?.ToString();

if (inst.HasValue && spec.HasValue && sem.HasValue)

{

var intstitution = await \_context.Institutions.FindAsync(inst);

ViewBag.Institution = intstitution?.InstitutionName;

var speciality = await \_context.Specialities.FindAsync(spec);

ViewBag.Speciality = string.Format("{0} {1}", speciality?.SpecialityCode, speciality?.SpecialityName);

ViewBag.SpecialityId = speciality?.Id;

var semester = await \_context.Semesters.FindAsync(sem);

ViewBag.SemesterId = semester?.Id;

ViewBag.Semester = semester?.Id == 7 ? "8" : semester?.Caption;

var kurs = semester?.Kurs;

ViewBag.Kurs = kurs;

var marks = await \_context.Examinations.Include(e => e.MarkNavigation).Include(e => e.Student).Include(e => e.Discipline).ThenInclude(m => m.Module).Where(m => m.Discipline.Module.SpecialityId == spec && m.Discipline.Semester == sem && m.Student.Kurs == kurs).ToListAsync();

//Если студенты не были заполнены заранее

if(marks.Count == 0)

{

marks.AddRange(\_context.Disciplines.Include(d => d.Module).Where(d => d.Module.SpecialityId == spec && d.Semester == sem).Select(x => new Models.Examination() { Discipline = x }));

}

ViewBag.Header = string.Format("Отчет по {0} практике по профессиональным модулям:", marks.All(x => x.Discipline.PracticeTypeId.HasValue) ? string.Format("{0}{1}{2}", marks.Any(x => x.Discipline.PracticeTypeId == 1) ? "учебной" : "", marks.Any(x => x.Discipline.PracticeTypeId == 1) && marks.Any(x => x.Discipline.PracticeTypeId == 2) ? " и " : "", marks.Any(x => x.Discipline.PracticeTypeId == 2) ? "производственной " : "") : "преддипломной");

return View(marks);

}

return View();

}

/// <summary>

/// Спсок специальностей для ajax функции

/// </summary>

/// <param name="Id">Идентификатор учебного заведения, выбранный на форме</param>

/// <returns>список специальностей</returns>

public async Task<object?> Specialities(int Id)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return null;

var list = await \_context.Specialities.Where(s => s.InstitutionId == Id).Select(s => new { s.Id, Name = string.Format("{0} {1}", s.SpecialityCode, s.SpecialityName) }).ToListAsync();

return list;

}

/// <summary>

/// Спсок семестров для ajax функции

/// </summary>

/// <param name="Id">Идентификатор специальности, выбранный на форме</param>

/// <returns>список семестров</returns>

public async Task<object?> Semesters(int Id)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return null;

var list = await \_context.Disciplines.Include(d => d.SemesterNavigation).Include(d => d.Module).Where(d => d.Module.SpecialityId == Id).Select(d => new { d.SemesterNavigation.Id, Name = d.SemesterNavigation.Caption }).Distinct().ToListAsync();

return list;

}

[HttpPost]

public async Task<object?> SaveMark(int id, int? Mark)

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return null;

var mark = await \_context.Examinations.Include(m => m.Discipline).Where(e => e.Id == id).FirstOrDefaultAsync();

if (mark != null)

{

mark.Mark = Mark;

await \_context.SaveChangesAsync();

var response = new { id = mark.StudentId, disc = mark.DisciplineId, dm = mark.Discipline.ModuleId, pr = mark.Discipline.PracticeTypeId ?? 0, avgD = await \_context.Examinations.Where(e => e.DisciplineId == mark.DisciplineId).AverageAsync(m => m.Mark), avgS = await \_context.Examinations.Include(e => e.Discipline).Where(e => e.StudentId == mark.StudentId && e.Discipline.ModuleId == mark.Discipline.ModuleId && e.Discipline.PracticeTypeId.Equals(mark.Discipline.PracticeTypeId)).AverageAsync(m => m.Mark) };

return response;

}

else

throw new ArgumentNullException();

}

[HttpPost]

public async Task<object?> AddStudent(string lastName, string firstName, string? middleName, int spec, int kurs)

{

try

{

var access = Helper.ShowPage(HttpContext.Session, pageName, out string redirect);

if (!access)

return null;

if (await \_context.Students.FirstOrDefaultAsync(s => s.SpecialityId.Equals(spec) && s.LastName.Equals(lastName) && s.FirstName.Equals(firstName) && ((string.IsNullOrEmpty(s.MiddleName) && string.IsNullOrEmpty(middleName)) || s.MiddleName.Equals(middleName)) && (s.Kurs.Equals(kurs))) == null)

{

var newStudent = new Student() { SpecialityId = spec, LastName = lastName, FirstName = firstName, MiddleName = middleName, Kurs = kurs };

var disc = await \_context.Disciplines.Include(d => d.Module).Where(d => d.Module.SpecialityId == spec).ToListAsync();

foreach (var d in disc)

{

newStudent.Examinations.Add(new Models.Examination() { Discipline = d });

}

await \_context.Students.AddAsync(newStudent);

await \_context.SaveChangesAsync();

return new { id = newStudent.Id, fio = string.Format("{0} {1} {2}", newStudent.LastName, newStudent.FirstName, newStudent.MiddleName), exams = newStudent.Examinations.OrderBy(x => x.DisciplineId).Select(x => new { x.Id, DisciplineId = x.Discipline.Id, x.Discipline.ModuleId, PracticeTypeId = x.Discipline.PracticeTypeId ?? 0 }).ToList() };

}

else

return null;

}

catch(Exception ex)

{

return null;

}

}

public async Task<IActionResult> Mark(int id)

{

var mark = await \_context.Examinations.FindAsync(id);

var v = PartialView(mark);

return PartialView(mark);

}

[HttpPost]

public async Task<IActionResult> Export(int spec, int sem, string header)

{

try

{

FastReport.Report rep = new FastReport.Report();

rep.Load(Path.Combine(\_environment.ContentRootPath, "App\_Data", "report.frx"));

rep.Dictionary.Connections[0].ConnectionString = \_configuration.GetConnectionString("sql");

rep.SetParameterValue("sem", sem);

rep.SetParameterValue("spec", spec);

rep.SetParameterValue("PracticeTypeName", header);

rep.Prepare();

PDFSimpleExport pdf = new PDFSimpleExport();

var ms = new MemoryStream();

pdf.Export(rep, ms);

return File(ms.ToArray(), "application/pdf", "отчет.pdf");

}

catch(Exception ex)

{

TempData["ExportError"] = ex.Message;

return Redirect(Request.Headers["Referer"].ToString());

}

}

}

Представление Index контроллера Examination

@model IEnumerable<WebApp.Models.Examination>

@{

await Html.RenderPartialAsync("SelectForm", ViewBag.selectFormData, ViewData);

}

@if (Model != null)

{

<div id="content" class="container w-100 bg-light mt-2">

<p class="text-center h4 mb-4">@ViewBag.Institution</p>

<p class="text-center h5">@ViewBag.Header</p>

<p class="text-justify">

@foreach (var module in Model.Select(x => x.Discipline.Module).Distinct().OrderBy(x=> x.Id))

{

@Html.Raw(string.Format("{0}. {1}<br/>", module.ModuleCode, module.ModuleName));

}

<strong>@string.Format("Курс: {0}", ViewBag.Kurs)</strong><br/>

@if (ViewBag.Kurs > 2)

{

<strong>@string.Format("{0}{1}", int.TryParse(ViewBag.Semester.ToString(), out int \_) ? "Семестр: " : "", ViewBag.Semester)</strong>

<br />

}

<strong>@string.Format("Специальность: {0}", ViewBag.Speciality)</strong>

@if(ViewBag.ExportError != null)

{

<br/>

<strong class="text-danger">@ViewBag.ExportError</strong>

}

</p>

@{

int k = 1;

}

@foreach (var module in Model.GroupBy(x=> x.Discipline.Module))

{

@foreach (var practiceType in module.GroupBy(x=>x.Discipline.PracticeTypeId).OrderBy(x=>x.Key))

{

@if(practiceType.Key == null)

{

<p class="h5 text-justify">@string.Format("{0} {1}.", k, module.Key.ModuleCode, module.Key.ModuleName)</p>

}

else

{

<p class="h5 text-justify">@string.Format(@"{0}. Виды и объем работ, оценка за выполненные обучающимся работы во время {1} практики по {2} {3}",

k, practiceType.Key switch { 1 => "учебной", 2 => "производственной", \_ => "" }, module.Key.ModuleCode, module.Key.ModuleName)

</p>

}

ViewBag.Module = module.Key.Id;

await Html.RenderPartialAsync("Examination", practiceType, ViewData);

k++;

}

}

<**form** action="@Url.Action("Export", "Examination")" method="post">

<input type="hidden" name="spec" value="@ViewBag.SpecialityId" />

<input type="hidden" name="sem" value="@ViewBag.SemesterId" />

<input type="hidden" name="header" value="@ViewBag.Header" />

<button type="submit" class="btn btn-sm btn-primary">Скачать</button>

</**form**>

</div>

<div class="modal fade" id="studentModal" tabindex="-1" role="dialog" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<**form** action="javascript:void(0);" onsubmit="addStudent()">

<div class="modal-header">

<h5 class="modal-title">Студент</h5>

<button type="button" class="close" data-bs-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<input id="studSpec" name="spec" type="hidden" value="" required />

<div class="form-group">

<label class="form-label">Фамилия</label>

<input type="text" id="lastName" name="lastName" class="form-control" required />

</div>

<div class="form-group">

<label class="form-label">Имя</label>

<input type="text" id="firstName" name="firstName" class="form-control" required />

</div>

<div class="form-group">

<label class="form-label">Отчество</label>

<input type="text" id="middleName" name="middleName" class="form-control" />

</div>

<div class="form-group">

<input id="kurs" name="kurs" type="hidden" value="" required />

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Отмена</button>

<button id="studentModalSubmit" type="submit" data-bs-dismiss="modal" class="btn btn-primary">Добавить</button>

</div>

</**form**>

</div>

</div>

</div>

}

Частичное представление SelectForm

@model IEnumerable<WebApp.Models.Institution>

<div class="container mt-3 p-1 border-1 menu">

<**form** action="@Url.Action("", "Examination")">

<div class="row mb-2">

<div class="col">

<label class="form-label text-white" for="instSelect">Учебное заведение</label>

<**select** id="instSelect" class="form-select form-select-sm" **name**="inst" **asp-items**="@(new SelectList(Model.OrderBy(m=>m.Id), "Id", "InstitutionName"))" onchange="selectChanged('specSelect', 'Specialities', this.value)" required>

<**option** **value**="" selected></**option**>

</**select**>

</div>

<div class="col">

<label class="form-label text-white" for="specSelect">Направление/специальность</label>

<select id="specSelect" class="form-select form-select-sm" name="spec" onchange="selectChanged('semSelect', 'Semesters', this.value)" required>

<**option** **value**="" selected></**option**>

</select>

</div>

<div class="col">

<label class="form-label text-white" for="semSelect">Семестр</label>

<select id="semSelect" class="form-select form-select-sm" name="sem" required>

<**option** **value**="" selected></**option**>

</select>

</div>

</div>

<div class="row mb-2">

<div class="col">

<button type="submit" class="btn btn-primary btn-block">Показать</button>

</div>

</div>

</**form**>

</div>

Частичное представление Examination

@model IGrouping<int?, Examination>

<table id="@string.Format("DM{0}-Pr{1}",ViewBag.Module, Model.Key ?? 0)" class="table table-sm table-light table-bordered w-auto align-middle">

<thead>

<tr>

<th>

</th>

@foreach (var student in Model.Select(x => x.Student).Distinct().Where(s => s != null).OrderBy(s => s.Id))

{

<th>

@string.Format("{0} {1} {2}", student.LastName, student.FirstName, student.MiddleName)

</th>

}

@if(Context.Session.GetInt32("userRole") == 2)

{

<th class="last">

<button type="button" class="btn btn-sm btn-link" data-bs-toggle="modal" data-bs-target="#studentModal" onclick="readyStudentForm(-1, @ViewBag.SpecialityId, '', '', '', @ViewBag.Kurs)"><img src="/images/add.jpg" /></button>

</th>

}

<th></th>

</tr>

</thead>

<tbody>

@foreach (var disc in Model.GroupBy(x => x.Discipline).OrderBy(x => x.Key.Id))

{

var a = disc.Average(m => m.Mark);

<tr id="@string.Format("D{0}", disc.Key.Id)">

<td><strong>@disc.Key.DisciplineName</strong></td>

@foreach (var mark in disc.Where(x => x.Student != null).OrderBy(x => x.StudentId))

{

<td id="@string.Format("ex{0}",mark.Id)">

@if (Context.Session.GetInt32("userRole") == 1)

{

<div id="@string.Format("label{0}",@mark.Id)" class="cursor-pointer w-100 h-100 p-1">@mark.Mark</div>

}

else

{

<**partial** **name**="Mark" **for**="@mark"/>

}

</td>

}

@if (Context.Session.GetInt32("userRole") == 2)

{

<td class="last"></td>

}

<td><div id="@string.Format("avgD{0}", disc.Key.Id)" class="p-1">@string.Format("{0}", a.HasValue && a.Value > 0 ? Math.Round(a.Value) : "")</div></td>

</tr>

}

</tbody>

<tfoot>

<tr>

<td>

</td>

@foreach (var avg in Model.GroupBy(x => x.Student).Where(x => x.Key != null).OrderBy(x => x.Key.Id))

{

var a = avg.Average(d => d.Mark);

<td><div id="@string.Format("avgS{0}-avgDM{1}-avgPr{2}", avg.Key.Id, ViewBag.Module, Model.Key ?? 0)" class="p-1">@string.Format("{0}", a.HasValue && a.Value > 0 ? Math.Round(a.Value) : "")</div></td>

}

@if (Context.Session.GetInt32("userRole") == 2)

{

<td class="last"></td>

}

<td></td>

</tr>

</tfoot>

</table>

Частичное представление Mark

@model WebApp.Models.Examination

<div id="@string.Format("edit{0}",@Model.Id)" style="display : none;">

<input type="number" min="2" max="5" class="form-control m-0 p-0 input-sm" size="1" value="@Model.Id" onblur="endEdit(@Model.Id)" />

</div>

<div id="@string.Format("label{0}",@Model.Id)" class="cursor-pointer w-100 h-100 p-1" ondblclick="beginEdit(@Model.Id)">@Model.Mark</div>