# 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».

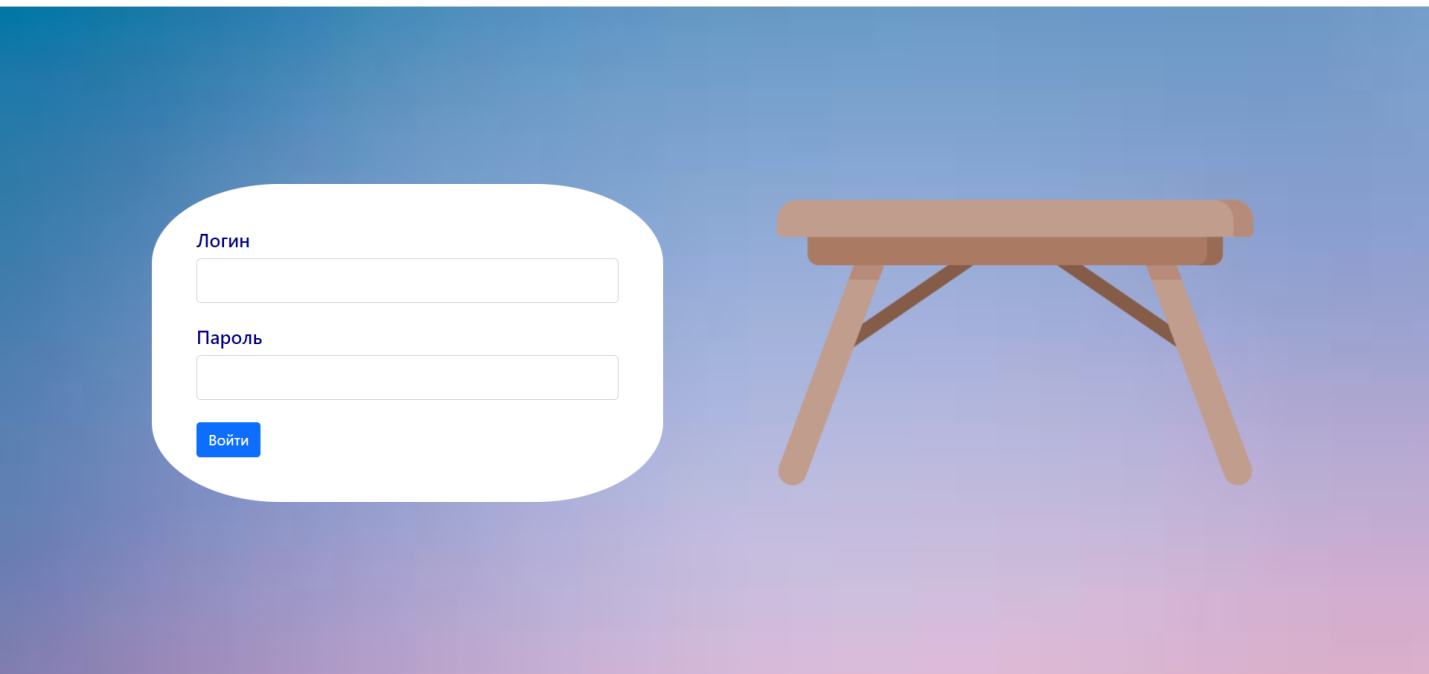


Рис. 1. Страница авторизации.

## 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.

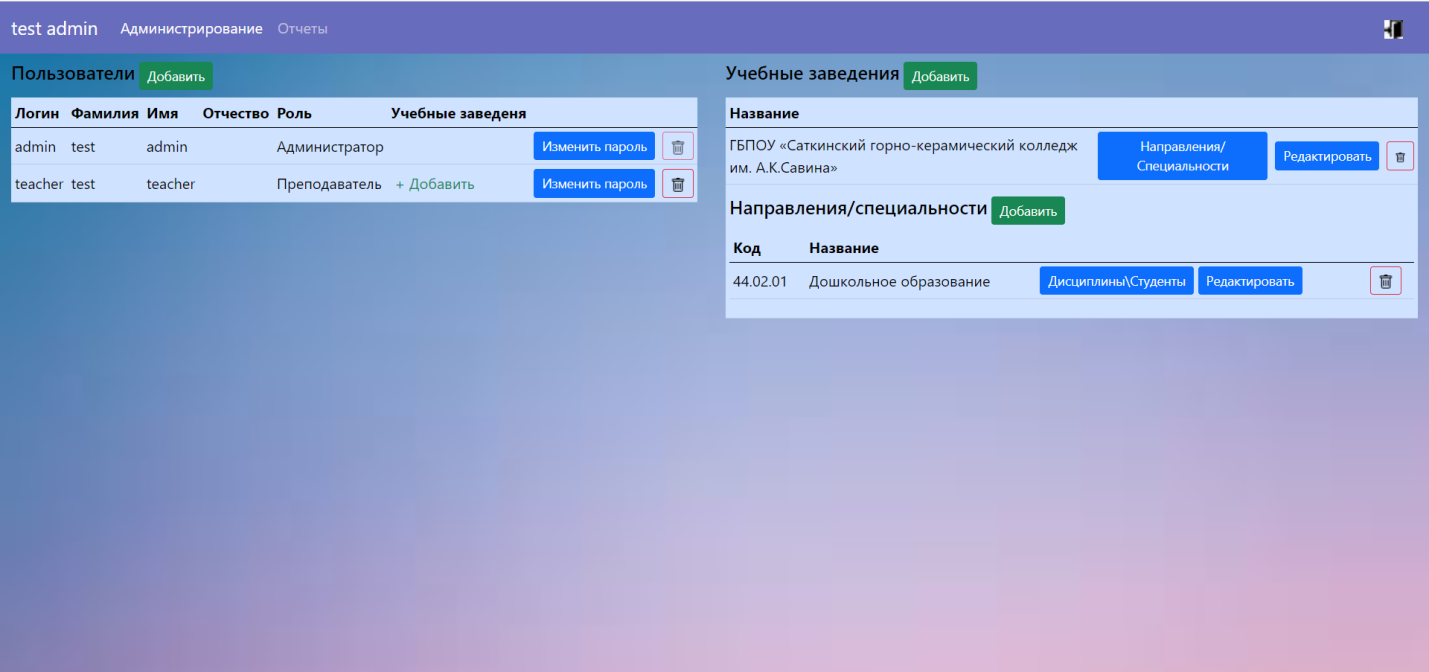


Рис. 2. Страница администрирования

## 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.

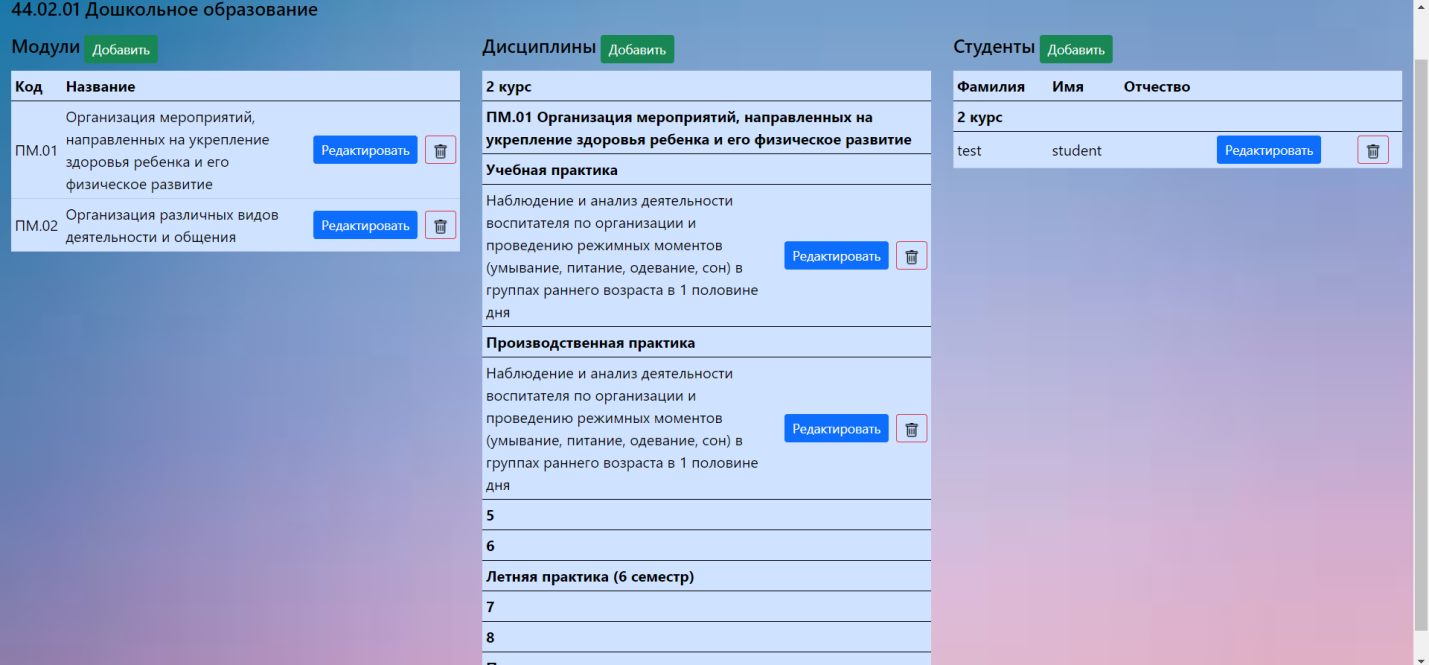


Рис. 3. Страница специальности.

## 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.

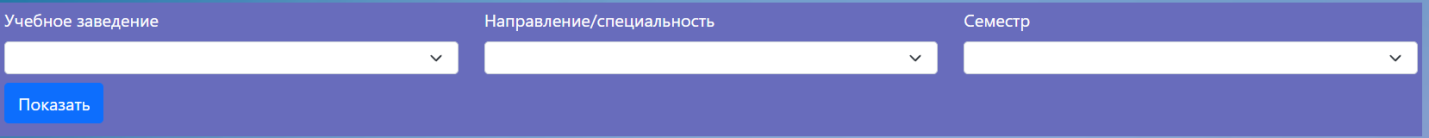


Рис. 4. Форма выбора параметров.

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.

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!');

}

});