МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО»

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

дисциплина «Избранные главы информатики»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

«**Разработка интерфейса приложения баз данных   
с использованием аутентификации и авторизации**»

Вариант № 16

Выполнил:

студент группы ИП-31, Самусев Р.А.

Принял:

доцент Асенчик О.Д.

Гомель 2018

**Цель работы:** Получить навыки создания типовых web-приложений для работы с информацией из реляционных баз данных, ознакомиться со способами аутентификации и авторизации в ASP.NET MVC Сore.

Github:

**Ход выполнения работы**

**ЗАДАНИЕ**:

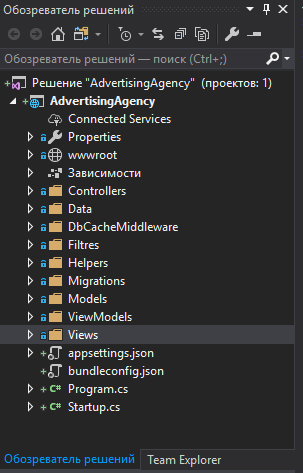
Используя разработанное ранее создать интерфейс Web-приложения для работы с данными, хранящимися в базе данных, на основе ASP.NET Core MVC Framework и Entity Framework Core.

Web- приложение должно удовлетворять следующим требованиям:

1. Любой пользователь для работы с приложением должен пройти аутентификацию.
2. Поддерживать реализацию не менее двух ролевых политик для доступа к классам и (или) методам контроллеров.
3. Управлять пользователями: просматривать, создавать, удалять и редактировать данные учетных записей.
4. Осуществлять ввод, редактирование, добавление и просмотр данных не менее чем из трех таблиц реляционной базы согласно варианту.
5. Вводимые и редактируемые данные должны проходить валидацию на стороне сервера и клиента.
6. Осуществлять разбиение таблично представленных данных, выбранных их реляционного источника, на страницы, их фильтрацию и сортировку.
7. Сохранять состояния элементов форм, используемых для фильтрации данных, и использовать их при повторном посещении соответствующей Web-страницы.
8. Использовать не менее одного разработанного html- хэлпера и не менее одного Tag-хэлпера. Например, хэлперы для разбиения на страницы, сортировки, генерации табличных данных и др.
9. Использовать не менее одного частичного представления (Partial View).
10. Иметь единое стилевое оформление, основанное на использовании одной или нескольких мастер-страниц, и систему навигации (строка меню, гиперссылки, кнопки), которая обеспечивает оптимальный путь перехода между двумя произвольно выбранными страницами в соответствии с логикой приложения.

**Решение:**

**Общий вид проекта**



**Исходные классы контроллеров:**

AccountController.cs

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using AdvertisingAgency.Models;

using AdvertisingAgency.ViewModels;

namespace AdvertisingAgency.Controllers

{

public class AccountController : Controller

{

private readonly UserManager<User> \_userManager;

private readonly SignInManager<User> \_signInManager;

private readonly RoleManager<IdentityRole> \_roleManager;

public AccountController(UserManager<User> userManager, SignInManager<User> signInManager, RoleManager<IdentityRole> roleManager)

{

\_userManager = userManager;

\_signInManager = signInManager;

\_roleManager = roleManager;

}

public IActionResult Register()

{

return View();

}

public IActionResult Login()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

[AllowAnonymous]

public async Task<IActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

User user = new User { Email = model.Email, UserName = model.Login };

// добавляем пользователя

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

if (result.Succeeded)

{

await \_signInManager.SignInAsync(user, true);

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

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

[AllowAnonymous]

public async Task<IActionResult> Login(LoginViewModel model)

{

if (ModelState.IsValid)

{

var result =

await \_signInManager.PasswordSignInAsync(model.Login, model.Password, model.RememberMe, false);

if (result.Succeeded)

{

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

}

else

{

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

}

}

else

{

return View(model);

}

return View(model);

}

public async Task<IActionResult> Logout()

{

await \_signInManager.SignOutAsync();

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

}

}

}

CustomerController.cs

using System;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Http;

using AdvertisingAgency.Models;

using Microsoft.Extensions.Caching.Memory;

using Newtonsoft.Json;

using AdvertisingAgency.Filtres;

using AdvertisingAgency.Data;

using AdvertisingAgency.ViewModels;

using System.Threading.Tasks;

namespace AdvertisingAgency.Controllers

{

[SetToSession]

public class CustomerController : Controller

{

public enum SortState

{

NameAsc, // по имени по возрастанию

NameDesc, // по имени по убыванию

AddressAsc, // по адресу по возрастанию

AddressDesc, // по адресу по убыванию

TelephoneAsc, // по телефону по возрастанию

TelephoneDesc // по телефону по убыванию

}

readonly adAgencyContext \_context;

public CustomerController(adAgencyContext context)

{

\_context = context;

}

public IActionResult Customer(string CustomerName, SortState sortCustomer = SortState.NameAsc, int page = 1)

{

int pageSize = 6;

IQueryable<Customers> customers = \_context.Customers;

if (!String.IsNullOrEmpty(CustomerName))

{

customers = \_context.Customers.Where(c => c.CustomerName.Contains(CustomerName));

}

if (HttpContext.Session.Get("customerSession") != null)

{

Customers customerSession = JsonConvert.DeserializeObject<Customers>(HttpContext.Session.GetString("customerSession"));

ViewBag.customerSession = customerSession;

}

if (HttpContext.Session.Get("customerSortSession") != null)

{

sortCustomer = JsonConvert.DeserializeObject<SortState>(HttpContext.Session.GetString("customerSortSession"));

}

ViewBag.CustomerNameSort = sortCustomer == SortState.NameAsc ? SortState.NameDesc : SortState.NameAsc;

ViewBag.CustomerAddressSort = sortCustomer == SortState.AddressAsc ? SortState.AddressDesc : SortState.AddressAsc;

ViewBag.CustomerTelephoneSort = sortCustomer == SortState.TelephoneAsc ? SortState.TelephoneDesc : SortState.TelephoneAsc;

switch (sortCustomer)

{

case SortState.NameAsc:

customers = customers.OrderBy(c => c.CustomerName);

break;

case SortState.NameDesc:

customers = customers.OrderByDescending(c => c.CustomerName);

break;

case SortState.AddressAsc:

customers = customers.OrderBy(c => c.CustomerAddress);

break;

case SortState.AddressDesc:

customers = customers.OrderByDescending(c => c.CustomerAddress);

break;

case SortState.TelephoneAsc:

customers = customers.OrderBy(c => c.CustomerTelephone);

break;

case SortState.TelephoneDesc:

customers = customers.OrderByDescending(c => c.CustomerTelephone);

break;

default:

customers = customers.OrderBy(c => c.CustomerName);

break;

}

int count = customers.Count();

customers = customers.Skip((page - 1) \* pageSize).Take(pageSize);

ViewBag.customers = customers.ToList();

PageViewModel pageViewModel = new PageViewModel(count, page, pageSize);

CustomerViewModel viewModel = new CustomerViewModel

{

PageViewModel = pageViewModel,

Customers = customers.ToList()

};

return View(viewModel);

}

[HttpPost]

public IActionResult AddCustomer(Customers customer)

{

\_context.Customers.Add(customer);

\_context.SaveChanges();

return RedirectToAction("Customer");

}

public async Task<IActionResult> Edit(string id)

{

Customers customer;

customer = await \_context.Customers.FindAsync(Convert.ToInt32(id));

if (customer == null)

{

return NotFound();

}

return View(customer);

}

[HttpPost]

public async Task<IActionResult> Edit(Customers customer)

{

if (ModelState.IsValid)

{

Customers cust = await \_context.Customers.FindAsync(Convert.ToInt32(customer.CustomersId));

if (cust != null)

{

cust.CustomerName = customer.CustomerName;

cust.CustomerAddress = customer.CustomerAddress;

cust.CustomerTelephone = customer.CustomerTelephone;

\_context.Customers.Update(cust);

await \_context.SaveChangesAsync();

}

}

return RedirectToAction("Customer");

}

[HttpPost]

public async Task<ActionResult> Delete(string id)

{

var cust = await \_context.Customers.FindAsync(Convert.ToInt32(id));

if (cust != null)

{

\_context.Customers.Remove(cust);

await \_context.SaveChangesAsync();

}

return RedirectToAction("Customer");

}

}

}

OrderController.cs

using System;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Http;

using AdvertisingAgency.Models;

using Microsoft.Extensions.Caching.Memory;

using AdvertisingAgency.Filtres;

using Newtonsoft.Json;

using AdvertisingAgency.Data;

using AdvertisingAgency.ViewModels;

using System.Threading.Tasks;

namespace AdvertisingAgency.Controllers

{

[SetToSession]

public class OrderController : Controller

{

public enum SortState

{

DateCustAsc, // по заказчикам по возрастанию

DateCustDesc, // по заказчикам по убыванию

DateAdvAsc, // по виду по возрастанию

DateAdvDesc, // по виду по убыванию

DateOrderAsc, // по дате договора по возрастанию

DateOrderDesc, // по дате договра по убыванию

PeriodAsc, // по периоду по возрастанию

PeriodDesc, // по периоду по убыванию

LocationAsc, // по локации по возрастанию

LocationDesc, // по локации по убыванию

PriceAsc, // по цене по убыванию

PriceDesc // по цене по убыванию

}

readonly adAgencyContext \_context;

public OrderController(adAgencyContext context)

{

\_context = context;

}

public IActionResult Order(string DateOrder, SortState sortOrder = SortState.DateOrderAsc, int page = 1)

{

int pageSize = 5;

IQueryable<Orders> orders = \_context.Orders;

DateTime dateorder = Convert.ToDateTime(DateOrder);

if (DateOrder != null)

{

orders = orders.Where(o => o.DateOrder.Date.Equals(dateorder));

}

if (Request.Cookies["orderSession"] != null)

{

Orders orderSession = JsonConvert.DeserializeObject<Orders>(Request.Cookies["orderSession"].ToString());

ViewBag.orderSession = orderSession;

}

if (HttpContext.Session.Get("orderSortSession") != null)

{

sortOrder = JsonConvert.DeserializeObject<SortState>(HttpContext.Session.GetString("orderSortSession"));

}

ViewBag.CustOrderSort = sortOrder == SortState.DateCustAsc ? SortState.DateCustDesc : SortState.DateCustAsc;

ViewBag.AdvPeriodSort = sortOrder == SortState.DateAdvAsc ? SortState.DateAdvDesc : SortState.DateAdvAsc;

ViewBag.DateOrderSort = sortOrder == SortState.DateOrderAsc ? SortState.DateOrderDesc : SortState.DateOrderAsc;

ViewBag.OrderPeriodSort = sortOrder == SortState.PeriodAsc ? SortState.PeriodDesc : SortState.PeriodAsc;

ViewBag.OrderLocationSort = sortOrder == SortState.LocationAsc ? SortState.LocationDesc : SortState.LocationAsc;

ViewBag.OrderCostSort = sortOrder == SortState.PriceAsc ? SortState.PriceDesc : SortState.PriceAsc;

switch (sortOrder)

{

case SortState.DateCustAsc:

orders = orders.OrderBy(o => o.Customers.CustomerName);

break;

case SortState.DateCustDesc:

orders = orders.OrderByDescending(o => o.Customers.CustomerName);

break;

case SortState.DateAdvAsc:

orders = orders.OrderBy(o => o.Advedirsments.AdType);

break;

case SortState.DateAdvDesc:

orders = orders.OrderByDescending(o => o.Advedirsments.AdType);

break;

case SortState.DateOrderAsc:

orders = orders.OrderBy(o => o.DateOrder);

break;

case SortState.DateOrderDesc:

orders = orders.OrderByDescending(o => o.DateOrder);

break;

case SortState.PeriodAsc:

orders = orders.OrderBy(o => o.DateBegin);

break;

case SortState.PeriodDesc:

orders = orders.OrderByDescending(o => o.DateEnd);

break;

case SortState.LocationAsc:

orders = orders.OrderBy(o => o.Location);

break;

case SortState.LocationDesc:

orders = orders.OrderByDescending(o => o.Location);

break;

case SortState.PriceAsc:

orders = orders.OrderBy(o => o.OrderCost);

break;

case SortState.PriceDesc:

orders = orders.OrderByDescending(o => o.OrderCost);

break;

default:

orders = orders.OrderBy(o => o.DateOrder);

break;

}

int count = orders.Count();

orders = orders.Skip((page - 1) \* pageSize).Take(pageSize);

ViewBag.advedirsments = \_context.Advedirsments.ToList();

ViewBag.customers = \_context.Customers.ToList();

ViewBag.orders = orders.ToList();

PageViewModel pageViewModel = new PageViewModel(count, page, pageSize);

OrderViewModel viewModel = new OrderViewModel

{

PageViewModel = pageViewModel,

Orders = orders.ToList()

};

return View(viewModel);

}

[HttpPost]

public IActionResult AddOrder(Orders order)

{

\_context.Orders.Add(order);

\_context.SaveChanges();

return RedirectToAction("Order");

}

public async Task<IActionResult> Edit(string id)

{

ViewBag.advedirsments = \_context.Advedirsments.ToList();

ViewBag.customers = \_context.Customers.ToList();

Orders order;

order = await \_context.Orders.FindAsync(Convert.ToInt32(id));

if (order == null)

{

return NotFound();

}

return View(order);

}

[HttpPost]

public async Task<IActionResult> Edit(Orders order)

{

if (ModelState.IsValid)

{

Orders or = await \_context.Orders.FindAsync(Convert.ToInt32(order.OrdersId));

if (or != null)

{

or.AdvedirsmentsId = order.AdvedirsmentsId;

or.CustomersId = order.CustomersId;

or.DateOrder = order.DateOrder;

or.DateBegin = order.DateBegin;

or.DateEnd = order.DateEnd;

or.Location = order.Location;

or.OrderCost = order.OrderCost;

\_context.Orders.Update(or);

await \_context.SaveChangesAsync();

}

}

return RedirectToAction("Order");

}

[HttpPost]

public async Task<ActionResult> Delete(string id)

{

var or = await \_context.Orders.FindAsync(Convert.ToInt32(id));

var advedirsments = \_context.Advedirsments.Where(a => a.AdvedirsmentsId == or.AdvedirsmentsId);

var customers = \_context.Customers.Where(c => c.CustomersId == or.CustomersId);

if (or != null)

{

\_context.Orders.Remove(or);

await \_context.SaveChangesAsync();

}

return RedirectToAction("Order");

}

}

}

RolesController.cs

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Identity;

using AdvertisingAgency.Models;

using AdvertisingAgency.ViewModels;

using Microsoft.AspNetCore.Authorization;

namespace CustomIdentityApp.Controllers

{

[Authorize(Roles = "admin")]

public class RolesController : Controller

{

RoleManager<IdentityRole> \_roleManager;

UserManager<User> \_userManager;

public RolesController(RoleManager<IdentityRole> roleManager, UserManager<User> userManager)

{

\_roleManager = roleManager;

\_userManager = userManager;

}

public IActionResult Index() => View(\_roleManager.Roles.ToList());

public IActionResult Create() => View();

[HttpPost]

public async Task<IActionResult> Create(string name)

{

if (!string.IsNullOrEmpty(name))

{

IdentityResult result = await \_roleManager.CreateAsync(new IdentityRole(name));

if (result.Succeeded)

{

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(name);

}

[HttpPost]

public async Task<IActionResult> Delete(string id)

{

IdentityRole role = await \_roleManager.FindByIdAsync(id);

if (role != null)

{

IdentityResult result = await \_roleManager.DeleteAsync(role);

}

return RedirectToAction("Index");

}

public IActionResult UserList() => View(\_userManager.Users.ToList());

public async Task<IActionResult> Edit(string userId)

{

            // получаем пользователя

            User user = await \_userManager.FindByIdAsync(userId);

if (user != null)

{

                // получем список ролей пользователя

                var userRoles = await \_userManager.GetRolesAsync(user);

var allRoles = \_roleManager.Roles.ToList();

ChangeRoleViewModel model = new ChangeRoleViewModel

{

UserId = user.Id,

UserEmail = user.Email,

UserRoles = userRoles,

AllRoles = allRoles

};

return View(model);

}

return NotFound();

}

[HttpPost]

public async Task<IActionResult> Edit(string userId, List<string> roles)

{

            // получаем пользователя

            User user = await \_userManager.FindByIdAsync(userId);

if (user != null)

{

                // получем список ролей пользователя

                var userRoles = await \_userManager.GetRolesAsync(user);

                // получаем все роли

                var allRoles = \_roleManager.Roles.ToList();

                // получаем список ролей, которые были добавлены

                var addedRoles = roles.Except(userRoles);

                // получаем роли, которые были удалены

                var removedRoles = userRoles.Except(roles);

await \_userManager.AddToRolesAsync(user, addedRoles);

await \_userManager.RemoveFromRolesAsync(user, removedRoles);

return RedirectToAction("UserList");

}

return NotFound();

}

}

}

UserController.cs

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Identity;

using AdvertisingAgency.Models;

using AdvertisingAgency.ViewModels;

using Microsoft.AspNetCore.Authorization;

namespace AdvertisingAgency.Controllers

{

[Authorize(Roles = "admin")]

public class UsersController : Controller

{

UserManager<User> \_userManager;

RoleManager<IdentityRole> \_roleManager;

public UsersController(UserManager<User> userManager, RoleManager<IdentityRole> roleManager)

{

\_userManager = userManager;

\_roleManager = roleManager;

}

public IActionResult Index() => Redirect("/Users/UsersView");

public IActionResult UsersView() => View(\_userManager.Users.ToList());

public IActionResult Create() => View();

[HttpPost]

public async Task<IActionResult> Create(CreateUserViewModel model)

{

if (ModelState.IsValid)

{

User user = new User { Email = model.Email, UserName = model.Email};

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

if (result.Succeeded)

{

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(model);

}

public async Task<IActionResult> Edit(string id)

{

User user = await \_userManager.FindByIdAsync(id);

if (user == null)

{

return NotFound();

}

EditUserViewModel model = new EditUserViewModel { Id = user.Id, Email = user.Email, Login = user.UserName };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Edit(EditUserViewModel model)

{

if (ModelState.IsValid)

{

User user = await \_userManager.FindByIdAsync(model.Id);

if (user != null)

{

user.Email = model.Email;

user.UserName = model.Login;

var result = await \_userManager.UpdateAsync(user);

if (result.Succeeded)

{

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

}

return View(model);

}

[HttpPost]

public async Task<ActionResult> Delete(string id)

{

User user = await \_userManager.FindByIdAsync(id);

if (user != null)

{

IdentityResult result = await \_userManager.DeleteAsync(user);

}

return RedirectToAction("Index");

}

}

}

HomeController.cs

using System;

using System.Diagnostics;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using AdvertisingAgency.Models;

using Newtonsoft.Json;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.Caching.Memory;

using AdvertisingAgency.Filtres;

using AdvertisingAgency.Data;

using AdvertisingAgency.ViewModels;

using System.Threading.Tasks;

namespace AdvertisingAgency.Controllers

{

[SetToSession]

public class HomeController : Controller

{

public enum SortState

{

TypeAsc, // по типу по возрастанию

TypeDesc, // по типу по убыванию

DescriptAsc, // по описанию по возрастанию

DescriptDesc // по описанию по убыванию

}

readonly adAgencyContext \_context;

public HomeController(adAgencyContext context)

{

\_context = context;

}

public IActionResult Index(string Adtype, SortState sortType = SortState.TypeAsc, int page = 1)

{

int pageSize = 3;

IQueryable<Advedirsments> advedirsments = \_context.Advedirsments;

if (!String.IsNullOrEmpty(Adtype))

{

advedirsments = advedirsments.Where(a => a.AdType.Contains(Adtype));

}

if (HttpContext.Session.Get("adSession") != null)

{

Advedirsments adSession = JsonConvert.DeserializeObject<Advedirsments>(HttpContext.Session.GetString("adSession"));

ViewBag.adSession = adSession;

}

if (HttpContext.Session.Get("sortTypeSession") != null)

{

sortType = JsonConvert.DeserializeObject<SortState>(HttpContext.Session.GetString("sortTypeSession"));

}

ViewBag.AdTypeSort = sortType == SortState.TypeAsc ? SortState.TypeDesc : SortState.TypeAsc;

ViewBag.DesriptionSort = sortType == SortState.DescriptAsc ? SortState.DescriptDesc : SortState.DescriptAsc;

switch (sortType)

{

case SortState.TypeAsc:

advedirsments = advedirsments.OrderBy(t => t.AdType);

break;

case SortState.TypeDesc:

advedirsments = advedirsments.OrderByDescending(t => t.AdType);

break;

case SortState.DescriptAsc:

advedirsments = advedirsments.OrderBy(d => d.AdDescription);

break;

case SortState.DescriptDesc:

advedirsments = advedirsments.OrderByDescending(d => d.AdDescription);

break;

default:

advedirsments = advedirsments.OrderBy(t => t.AdType);

break;

}

int count = advedirsments.Count();

advedirsments = advedirsments.Skip((page - 1) \* pageSize).Take(pageSize);

ViewBag.advedirsments = advedirsments.ToList();

PageViewModel pageViewModel = new PageViewModel(count, page, pageSize);

IndexViewModel viewModel = new IndexViewModel

{

PageViewModel = pageViewModel,

Advedirsments = advedirsments.ToList()

};

return View(viewModel);

}

[HttpPost]

public IActionResult AddType(Advedirsments advedirsment)

{

\_context.Advedirsments.Add(advedirsment);

\_context.SaveChanges();

return RedirectToAction("Index");

}

[HttpPost]

public async Task<ActionResult> Delete(string id)

{

var adv = await \_context.Advedirsments.FindAsync(Convert.ToInt32(id));

if (adv != null)

{

\_context.Advedirsments.Remove(adv);

await \_context.SaveChangesAsync();

}

return RedirectToAction("Index");

}

public async Task<IActionResult> Edit(string id)

{

Advedirsments advedirsment;

advedirsment = await \_context.Advedirsments.FindAsync(Convert.ToInt32(id));

if (advedirsment == null)

{

return NotFound();

}

return View(advedirsment);

}

[HttpPost]

public async Task<IActionResult> Edit(Advedirsments advedirsment)

{

if (ModelState.IsValid)

{

Advedirsments adv = await \_context.Advedirsments.FindAsync(Convert.ToInt32(advedirsment.AdvedirsmentsId));

if (adv != null)

{

adv.AdType = advedirsment.AdType;

adv.AdDescription = advedirsment.AdDescription;

\_context.Advedirsments.Update(adv);

await \_context.SaveChangesAsync();

}

}

return RedirectToAction("Index");

}

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

**Исходные классы разработанных хэлперов**

TableHelper.cs

using Microsoft.AspNetCore.Html;

using Microsoft.AspNetCore.Mvc.Rendering;

using AdvertisingAgency.Models;

using System.Collections.Generic;

namespace AdvertisingAgency.Helpers

{

public static class TableHelper

{

public static HtmlString CreateCustomerTable(this IHtmlHelper html, IEnumerable<Customers> customers)

{

string result = "";

foreach (Customers item in customers)

{

result += "<tr>";

result += $"<td>{item.CustomerName}</td>";

result += $"<td>{item.CustomerAddress}</td>";

result += $"<td>{item.CustomerTelephone}</td>";

result += $"<td class=\"act\"><form action=\"/Customer/Delete/ " + item.CustomersId +

"\" method=\"post\"><a class=\"btn btn-sm btn-primary\" href=\"/Customer/Edit/" + item.CustomersId +

"\">Изменить</a><button type = \"submit\" class=\"btn btn-sm btn-danger\">Удалить</button></form></td>";

result += "</tr>";

}

return new HtmlString(result);

}

}

}

FilterTagHelper.cs

using Microsoft.AspNetCore.Razor.TagHelpers;

namespace AdvertisingAgency.Helpers

{

public class FilterTagHelper : TagHelper

{

public override void Process(TagHelperContext context, TagHelperOutput output)

{

output.TagName = "input";

output.Attributes.Add("type", "submit");

output.Attributes.Add("value", "Filter");

}

}

}

**Исходные коды представлений:**

Login.cshtml

@model AdvertisingAgency.ViewModels.LoginViewModel

@{

ViewData["Title"] = "Login";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Вход в приложение</h2>

<**form** method="post" **asp-controller**="Account" **asp-action**="Login">

<**div** **asp-validation-summary**="ModelOnly"></**div**>

<div class="form-group">

<**label** **asp-for**="Login"></**label**><br />

<**input** **asp-for**="Login" class="form-control"/>

<**span** **asp-validation-for**="Login"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Password"></**label**><br />

<**input** **asp-for**="Password" class="form-control" />

<**span** **asp-validation-for**="Password"></**span**>

</div>

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

</**form**>

Register.cshtml

@model AdvertisingAgency.ViewModels.RegisterViewModel

@{

ViewData["Title"] = "Регистрация";

}

<h2>@ViewData["Title"]</h2>

<div class="row">

<div class="col-md-4">

<**form** method="post" **asp-controller**="Account" **asp-action**="Register">

<h4>Регистрация нового пользователя</h4>

<**div** **asp-validation-summary**="All" class="text-danger"></**div**>

<div class="form-group">

<**label** **asp-for**="Email"></**label**>

<**input** **asp-for**="Email" class="form-control" />

<**span** **asp-validation-for**="Email" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Login"></**label**>

<**input** **asp-for**="Login" class="form-control" />

<**span** **asp-validation-for**="Login" class="text-danger"></**span**>

</div>

@Html.Partial("Partial", Model)

<div class="form-group">

<**label** **asp-for**="ConfirmPassword"></**label**>

<**input** **asp-for**="ConfirmPassword" class="form-control" />

<**span** **asp-validation-for**="ConfirmPassword" class="text-danger"></**span**>

</div>

<button type="submit" class="btn btn-default">Зарегистрироваться</button>

</**form**>

</div>

</div>

@section Scripts {

@await Html.PartialAsync("\_ValidationScriptsPartial")

}

Customer. cshtml

@using Microsoft.AspNetCore.Identity

@using AdvertisingAgency.ViewModels

@using AdvertisingAgency.Helpers

@model CustomerViewModel

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

@addTagHelper "\*, AdvertisingAgency"

@{

ViewData["Title"] = "Customers";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<img src="~/images/images.jpg" />

<h1>Заказчики</h1>

@using (Html.BeginForm("Customer", "Customer", FormMethod.Get))

{

<p>

<label>Имя заказчика</label>

@Html.TextBox("CustomerName")

<**filter**></**filter**>

</p>

}

<table>

<tr>

<th>

<**a** **asp-action**="Customer" **asp-route-sortCustomer**="@ViewBag.CustomerNameSort">

Имя

</**a**>

</th>

<th>

<**a** **asp-action**="Customer" **asp-route-sortCustomer**="@ViewBag.CustomerAddressSort">

Адрес

</**a**>

</th>

<th>

<**a** **asp-action**="Customer" **asp-route-sortCustomer**="@ViewBag.CustomerTelephoneSort">

Телефон

</**a**>

</th>

<th class="act">

<span>Действие</span>

</th>

</tr>

@TableHelper.CreateCustomerTable(Html, Model.Customers)

</table>

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Customer"

**asp-route-page**="@(Model.PageViewModel.PageNumber - 1)"

class="btn btn-default btn">

<i class="glyphicon glyphicon-chevron-left"></i>

Назад

</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Customer"

**asp-route-page**="@(Model.PageViewModel.PageNumber + 1)"

class="btn btn-default btn">

Вперед

<i class="glyphicon glyphicon-chevron-right"></i>

</**a**>

}

@{

AdvertisingAgency.Models.Customers customerSession =

(AdvertisingAgency.Models.Customers)ViewBag.customerSession;

}

@using (Html.BeginForm("AddCustomer", "Customer", FormMethod.Post))

{

<p class="add"> Добавить </p>

<p>

<label>Имя заказчика</label>

@{

if (customerSession != null)

{

@Html.TextBox("CustomerName", customerSession.CustomerName)

}

else

{

@Html.TextBox("CustomerName")

}

}

</p>

<p>

<label>Адрес</label>

@{

if (customerSession != null)

{

@Html.TextBox("CustomerAddress", customerSession.CustomerAddress)

}

else

{

@Html.TextBox("CustomerAddress")

}

}

</p>

<p>

<label>Телефон</label>

@{

if (customerSession != null)

{

@Html.TextBox("CustomerTelephone", customerSession.CustomerTelephone)

}

else

{

@Html.TextBox("CustomerTelephone")

}

}

</p>

<p>

<input type="submit" value="Send">

</p>

}

/Cutomer/Edit.cshtml

@model AdvertisingAgency.Models.Customers

@{

ViewData["Title"] = "Edit";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Изменить</h2>

@using (Html.BeginForm("Edit", "Customer", FormMethod.Post))

{

<**div** **asp-validation-summary**="ModelOnly"></**div**>

<div class="form-group">

<**input** **type**="hidden" **asp-for**="CustomersId" />

</div>

<div class="form-group">

<**label** **asp-for**="CustomerName" class="control-label">Имя заказчика</**label**>

<**input** **type**="text" **asp-for**="CustomerName" class="form-control" />

</div>

<div class="form-group">

<**label** **asp-for**="CustomerAddress" class="control-label">Адрес</**label**>

<**input** **type**="text" **asp-for**="CustomerAddress" class="form-control" />

</div>

<div class="form-group">

<**label** **asp-for**="CustomerTelephone" class="control-label">Телефон</**label**>

<**input** **type**="text" **asp-for**="CustomerTelephone" class="form-control" />

</div>

<div class="form-group">

<input type="submit" value="Сохранить" class="btn btn-success" />

</div>

}

/Home/Edit.cshtml

@model AdvertisingAgency.Models.Advedirsments

@{

ViewData["Title"] = "Edit";

}

<h2>Изменить</h2>

@using (Html.BeginForm("Edit", "Home", FormMethod.Post))

{

<**div** **asp-validation-summary**="ModelOnly"></**div**>

<div class="form-group">

<**input** **type**="hidden" **asp-for**="AdvedirsmentsId" />

</div>

<div class="form-group">

<**label** **asp-for**="AdType" class="control-label">Вид рекламы</**label**>

<**input** **type**="text" **asp-for**="AdType" class="form-control" />

</div>

<div class="form-group">

<**label** **asp-for**="AdDescription" class="control-label">Описание</**label**>

<**input** **type**="text" **asp-for**="AdDescription" class="form-control" />

</div>

<div class="form-group">

<input type="submit" value="Сохранить" class="btn btn-success" />

</div>

}

Index.cshtml

@using AdvertisingAgency.ViewModels

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

@addTagHelper "\*, AdvertisingAgency"

@model IndexViewModel

@{

ViewData["Title"] = "Advedirsments";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<img src="~/images/images.jpg" />

<h1>Виды рекламы</h1>

<p></p>

@using (Html.BeginForm("Index", "Home", FormMethod.Get))

{

<p>

<label>Тип Рекламы</label>

@Html.TextBox("AdType")

<**filter**></**filter**>

</p>

}

<table border="1">

<tr>

<th>

<**a** **asp-action**="Index" **asp-route-sortType**="@ViewBag.AdTypeSort">

Тип

</**a**>

</th>

<th>

<**a** **asp-action**="Index" **asp-route-sortType**="@ViewBag.DesriptionSort">

Описание

</**a**>

</th>

<th class="act">

<span>Действие</span>

</th>

</tr>

@foreach (var item in Model.Advedirsments)

{

<tr>

<td>@item.AdType</td>

<td>@item.AdDescription</td>

<td class="act">

<**form** **asp-action**="Delete" **asp-route-id**="@item.AdvedirsmentsId" method="post">

<**a** class="btn btn-sm btn-primary" **asp-action**="Edit" **asp-route-id**="@item.AdvedirsmentsId">Изменить</**a**>

<button type="submit" class="btn btn-sm btn-danger">

Удалить

</button>

</**form**>

</td>

</tr>

}

</table>

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber - 1)"

class="btn btn-default btn">

<i class="glyphicon glyphicon-chevron-left"></i>

Назад

</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber + 1)"

class="btn btn-default btn">

Вперед

<i class="glyphicon glyphicon-chevron-right"></i>

</**a**>

}

@using (Html.BeginForm("AddType", "Home", FormMethod.Post))

{

<p class="add"> Добавить </p>

<p>

<label>Тип</label>

@{

Advedirsments adSession = (Advedirsments)ViewBag.adSesion;

if (adSession != null)

{

@Html.TextBox("AdType", adSession.AdType);

}

else

{

@Html.TextBox("AdType");

}

}

</p>

<p>

<label>Описание</label>

@{

if (adSession != null)

{

@Html.TextBox("AdDescription", adSession.AdDescription);

}

else

{

@Html.TextBox("AdDescription");

}

}

</p>

<p>

<input type="submit" value="Send">

</p>

}

/Order/Edit.cshtml

@model AdvertisingAgency.Models.Orders

@{

ViewData["Title"] = "Edit";

}

<h2>Изменить</h2>

@using (Html.BeginForm("Edit", "Order", FormMethod.Post))

{

<**div** **asp-validation-summary**="ModelOnly"></**div**>

<div class="form-group">

<**input** **type**="hidden" **asp-for**="OrdersId" />

</div>

<div class="form-group">

<select name="CustomersId">

{

@{

List<Customers> customers = (List<Customers>)ViewBag.customers;

foreach (var item in customers)

{

if (Model.CustomersId == item.CustomersId)

{

<**option** **value**=@item.CustomerName selected>@item.CustomerName</**option**>

}

else

{

<**option** **value**=@item.CustomerName>@item.CustomerName</**option**>

}

}

}

}

</select>

</div>

<div class="form-group">

<select name="AdvedirsmentsId">

{

@{

List<Advedirsments> advedirsments = (List<Advedirsments>)ViewBag.advedirsments;

foreach (var item in advedirsments)

{

if (Model.AdvedirsmentsId == item.AdvedirsmentsId)

{

<**option** **value**=@item.AdvedirsmentsId selected>@item.AdvedirsmentsId</**option**>

}

else

{

<**option** **value**=@item.AdvedirsmentsId>@item.AdvedirsmentsId</**option**>

}

}

}

}

</select>

</div>

<div class="form-group">

<**label** **asp-for**="DateOrder" class="control-label">Дата заказа</**label**>

@{

DateTime tmp = Convert.ToDateTime(Model.DateOrder);

string n = tmp.ToString("yyyy-MM-dd");

}

<**input** **type**="date" **asp-for**="DateOrder" class="form-control" **value**="@n" />

</div>

<div class="form-group">

<**label** **asp-for**="DateBegin" class="control-label">Дата начала работы</**label**>

@{

tmp = Convert.ToDateTime(Model.DateBegin);

n = tmp.ToString("yyyy-MM-dd");

}

<**input** **type**="date" **asp-for**="DateBegin" class="form-control" **value**="@n" />

</div>

<div class="form-group">

<**label** **asp-for**="DateEnd" class="control-label">Дата окончания работы</**label**>

@{

tmp = Convert.ToDateTime(Model.DateEnd);

n = tmp.ToString("yyyy-MM-dd");

}

<**input** **type**="date" **asp-for**="DateEnd" class="form-control" **value**="@n" />

</div>

<div class="form-group">

<**label** **asp-for**="Location" class="control-label">Место расположения</**label**>

<**input** **type**="text" **asp-for**="Location" class="form-control" />

</div>

<div class="form-group">

<**label** **asp-for**="OrderCost" class="control-label">Цена услуги</**label**>

<**input** **type**="text" **asp-for**="OrderCost" class="form-control" />

</div>

<div class="form-group">

<input type="submit" value="Сохранить" class="btn btn-success" />

</div>

}

Order.cshtml

@using AdvertisingAgency.ViewModels

@model OrderViewModel

@inject SignInManager<AdvertisingAgency.Models.User> SignInManager

@inject UserManager<AdvertisingAgency.Models.User> UserManager

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

@addTagHelper "\*, AdvertisingAgency"

@{

ViewData["Title"] = "Orders";

}

<img src="~/images/images.jpg" />

<h1>Заказы</h1>

@using (Html.BeginForm("Order", "Order", FormMethod.Get))

{

<p>

<label>Дата Договора</label>

@Html.TextBox("DateOrder")

<**filter**></**filter**>

</p>

}

<table>

<tr>

<th>

<**a** **asp-action**="Order" **asp-route-sortOrder**="@ViewBag.CustOrderSort">

Заказчик

</**a**>

</th>

<th>

<**a** **asp-action**="Order" **asp-route-sortOrder**="@ViewBag.AdvPeriodSort">

Вид рекламы

</**a**>

</th>

<th>

<**a** **asp-action**="Order" **asp-route-sortOrder**="@ViewBag.DateOrderSort">

Дата заключения договора

</**a**>

</th>

<th>

<**a** **asp-action**="Order" **asp-route-sortOrder**="@ViewBag.OrderPeriodSort">

Срок действия

</**a**>

</th>

<th>

<**a** **asp-action**="Order" **asp-route-sortOrder**="@ViewBag.OrderLocationSort">

Место расположения

</**a**>

</th>

<th>

<**a** **asp-action**="Order" **asp-route-sortOrder**="@ViewBag.OrderCostSort">

Цена услуги

</**a**>

</th>

<th class="act">

<span>Действие</span>

</th>

</tr>

@foreach (var item in Model.Orders)

{

List<Advedirsments> advedirsments = (List<Advedirsments>)ViewBag.advedirsments;

string adv = "";

foreach (var ad in advedirsments)

{

if (item.AdvedirsmentsId == ad.AdvedirsmentsId)

{

adv = ad.AdType;

}

}

List<Customers> customers = (List<Customers>)ViewBag.customers;

string cust = "";

foreach (var c in customers)

{

if (item.CustomersId == c.CustomersId)

{

cust = c.CustomerName;

}

}

<tr>

<td>@cust</td>

<td>@adv</td>

<td>@item.DateOrder.ToShortDateString()</td>

<td>@item.DateBegin.ToShortDateString() - @item.DateEnd.ToShortDateString()</td>

<td>@item.Location</td>

<td>@item.OrderCost</td>

<td class="act">

<**form** **asp-action**="Delete" **asp-route-id**="@item.OrdersId" method="post">

<**a** class="btn btn-sm btn-primary" **asp-action**="Edit" **asp-route-id**="@item.OrdersId">Изменить</**a**>

<button type="submit" class="btn btn-sm btn-danger">

Удалить

</button>

</**form**>

</td>

</tr>

}

</table>

@if (Model.PageViewModel.HasPreviousPage)

{

<**a** **asp-action**="Order"

**asp-route-page**="@(Model.PageViewModel.PageNumber - 1)"

class="btn btn-default btn">

<i class="glyphicon glyphicon-chevron-left"></i>

Назад

</**a**>

}

@if (Model.PageViewModel.HasNextPage)

{

<**a** **asp-action**="Order"

**asp-route-page**="@(Model.PageViewModel.PageNumber + 1)"

class="btn btn-default btn">

Вперед

<i class="glyphicon glyphicon-chevron-right"></i>

</**a**>

}

@{

AdvertisingAgency.Models.Orders orderSession =

(AdvertisingAgency.Models.Orders)ViewBag.orderSession;

}

@if (SignInManager.IsSignedIn(User))

{

@using (Html.BeginForm("AddOrder", "Order", FormMethod.Post))

{

<**div** **asp-validation-summary**="ModelOnly"></**div**>

<p>

<label>Заказчик</label>

<**select** **asp-for**="orders.CustomersId">

{

@{

List<Customers> customers = (List<Customers>)ViewBag.customers;

foreach (var item in customers)

{

if (orderSession != null && orderSession.OrdersId == item.CustomersId)

{

<**option** asp-for="orders.CustomersId" **value**=@item.CustomersId selected>@item.CustomerName</**option**>

}

else

{

<**option** asp-for="orders.CustomersId" **value**=@item.CustomersId>@item.CustomerName</**option**>

}

}

}

}

</**select**>

@Html.DropDownList("AdvedirsmentsId", new SelectList(Model.orders.Advedirsments.AdType, "Id", "Type.Name"))

</p>

<p>

<**label** **asp-for**="orders.AdvedirsmentsId">Вид рекламы</**label**>

<**select** **asp-for**="orders.AdvedirsmentsId">

{

@{

List<Advedirsments> advedirsments = (List<Advedirsments>)ViewBag.advedirsments;

foreach (var item in advedirsments)

{

if (orderSession != null && orderSession.AdvedirsmentsId == item.AdvedirsmentsId)

{

<**option** asp-for="orders.AdvedirsmentsId" **value**=@item.AdvedirsmentsId selected>@item.AdType</**option**>

}

else

{

<**option** asp-for="orders.AdvedirsmentsId" **value**=@item.AdvedirsmentsId>@item.AdType</**option**>

}

}

}

}

</**select**>

</p>

<p>

<label>Дата заказа</label>

@{

if (orderSession != null)

{

@Html.TextBox("DateOrder", orderSession.DateOrder);

}

else

{

@Html.TextBox("DateOrder");

}

}

</p>

<p>

<label>Дата начала</label>

@{

if (orderSession != null)

{

@Html.TextBox("DateBegin", orderSession.DateBegin);

}

else

{

@Html.TextBox("DateBegin");

}

}

</p>

<p>

<label>Конец</label>

@{

if (orderSession != null)

{

@Html.TextBox("DateEnd", orderSession.DateEnd);

}

else

{

@Html.TextBox("DateEnd");

}

}

</p>

<p>

<label>Место расположения</label>

@{

if (orderSession != null)

{

@Html.TextBox("Location", orderSession.Location);

}

else

{

@Html.TextBox("Location");

}

}

</p>

<p>

<label>Цена услуги</label>

@{

if (orderSession != null)

{

@Html.TextBox("OrderCost", orderSession.OrderCost);

}

else

{

@Html.TextBox("OrderCost");

}

}

</p>

<p>

<input type="submit" value="Send">

</p>

}

}

/Roles/Create.cshtml

@model string

@{

ViewBag.Title = "Create";

}

<**div** **asp-validation-summary**="All" class="text-danger"></**div**>

<**form** **asp-action**="Create" method="post">

<div class="form-group">

<label for="name">Новая роль</label>

<input name="name" class="form-control" />

</div>

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

</**form**>

/Roles/Edit.cshtml

@using Microsoft.AspNetCore.Identity

@model AdvertisingAgency.ViewModels.ChangeRoleViewModel

@{

ViewBag.Title = "Edit";

}

<h2>Изменение роли для пользователя @Model.UserEmail</h2>

<**form** **asp-action**="Edit" method="post">

<input type="hidden" name="userId" value="@Model.UserId" />

<div class="form-group">

@foreach (IdentityRole role in Model.AllRoles)

{

<input type="checkbox" name="roles" value="@role.Name"

@(Model.UserRoles.Contains(role.Name) ? "checked=\"checked\"" : "") />@role.Name <br />

}

</div>

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

</**form**>

/Roles/Index.cshtml

@model IEnumerable<Microsoft.AspNetCore.Identity.IdentityRole>

@{

ViewBag.Titile = "Index";

}

<h2>Список ролей</h2>

<table>

@foreach (var role in Model)

{

<tr>

<td>@role.Name</td>

<td>

<**form** **asp-action**="Delete" **asp-route-id**="@role.Id" method="post">

<button type="submit" class="btn btn-sm btn-danger">

Удалить

</button>

</**form**>

</td>

</tr>

}

</table>

<p></p>

<**a** class="btn btn-info" **asp-action**="Create">Добавить роль</**a**>

<**a** class="btn btn-info" **asp-action**="UserList">Список пользователей</**a**>

/Roles/UserList.cshtml

@model IEnumerable<AdvertisingAgency.Models.User>

@{

ViewBag.Title = "UserList";

}

<h2>Список пользователей</h2>

<table>

@foreach (var user in Model)

{

<tr>

<td>@user.Email</td>

<td >

<**a** class="btn btn-sm btn-primary" **asp-action**="Edit" **asp-route-userid**="@user.Id">Права доступа</**a**>

</td>

</tr>

}

</table>

/Users/Create.cshtml

@model AdvertisingAgency.ViewModels.RegisterViewModel

@{

ViewBag.Title = "Добавление пользователя";

}

<h2>Создать нового пользователя</h2>

<**form** **asp-action**="Create" **asp-controller**="Users" method="post">

<**div** **asp-validation-summary**="ModelOnly"></**div**>

<div class="form-group">

<**label** **asp-for**="Email"></**label**>

<**input** **asp-for**="Email" />

<**span** **asp-validation-for**="Email"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Login"></**label**>

<**input** **asp-for**="Login" />

<**span** **asp-validation-for**="Login"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Password"></**label**>

<**input** **asp-for**="Password" />

<**span** **asp-validation-for**="Password"></**span**>

</div>

<div class="form-group">

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

</div>

</**form**>

/Users/Edit.cshtml

@model AdvertisingAgency.ViewModels.EditUserViewModel

@{

ViewBag.Title = "Редактирование пользователя";

}

<**form** **asp-action**="Edit" **asp-controller**="Users" method="post">

<**div** **asp-validation-summary**="ModelOnly"></**div**>

<div class="form-group">

<**input** **type**="hidden" **asp-for**="Id" />

</div>

<div class="form-group">

<**label** **asp-for**="Email" class="control-label">Email</**label**>

<**input** **type**="text" **asp-for**="Email" class="form-control" />

</div>

<div class="form-group">

<**label** **asp-for**="Login" class="control-label">Login</**label**>

<**input** **type**="text" **asp-for**="Login" class="form-control" />

</div>

<div class="form-group">

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

</div></**form**>

/Users/UserList.cshtml

@model IEnumerable<AdvertisingAgency.Models.User>

@{

ViewBag.Title = "Список пользователей";

}

<h2>Пользователи</h2>

<**a** **asp-action**="Create">Добавить нового пользователя</**a**>

<table>

<tr>

<th>Email</th>

<th>Login</th>

<th>Действие</th>

</tr>

@foreach (var user in Model)

{

<tr>

<td>@user.Email</td>

<td>@user.UserName</td>

<td class="act">

<**form** **asp-action**="Delete" **asp-route-id**="@user.Id" method="post">

<**a** class="btn btn-sm btn-primary" **asp-action**="Edit" **asp-route-id**="@user.Id">Изменить</**a**>

<button type="submit" class="btn btn-sm btn-danger">

Удалить

</button>

</**form**>

</td>

</tr>

}

</table>

**Код частичного представления**

Partial.cshtml

@model AdvertisingAgency.ViewModels.RegisterViewModel

<div class="form-group">

<**label** **asp-for**="Password"></**label**>

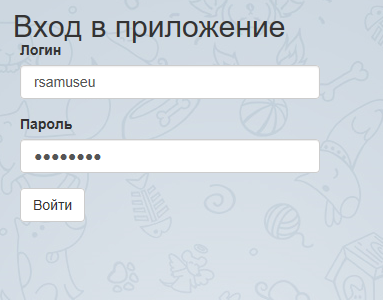
<**input** **asp-for**="Password" class="form-control" />

<**span** **asp-validation-for**="Password" class="text-danger"></**span**>

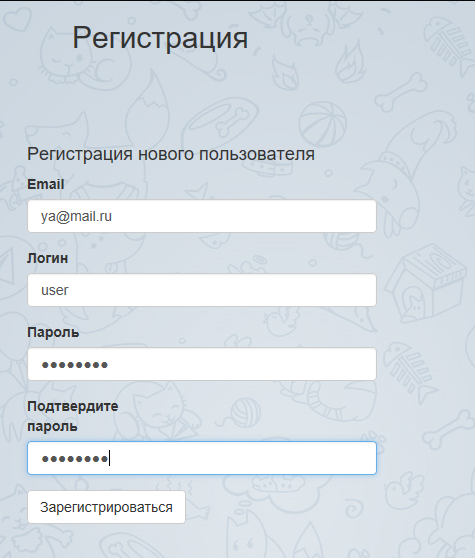
</div>

**Копии экранов браузера, демонстрирующие отображение представлений:**

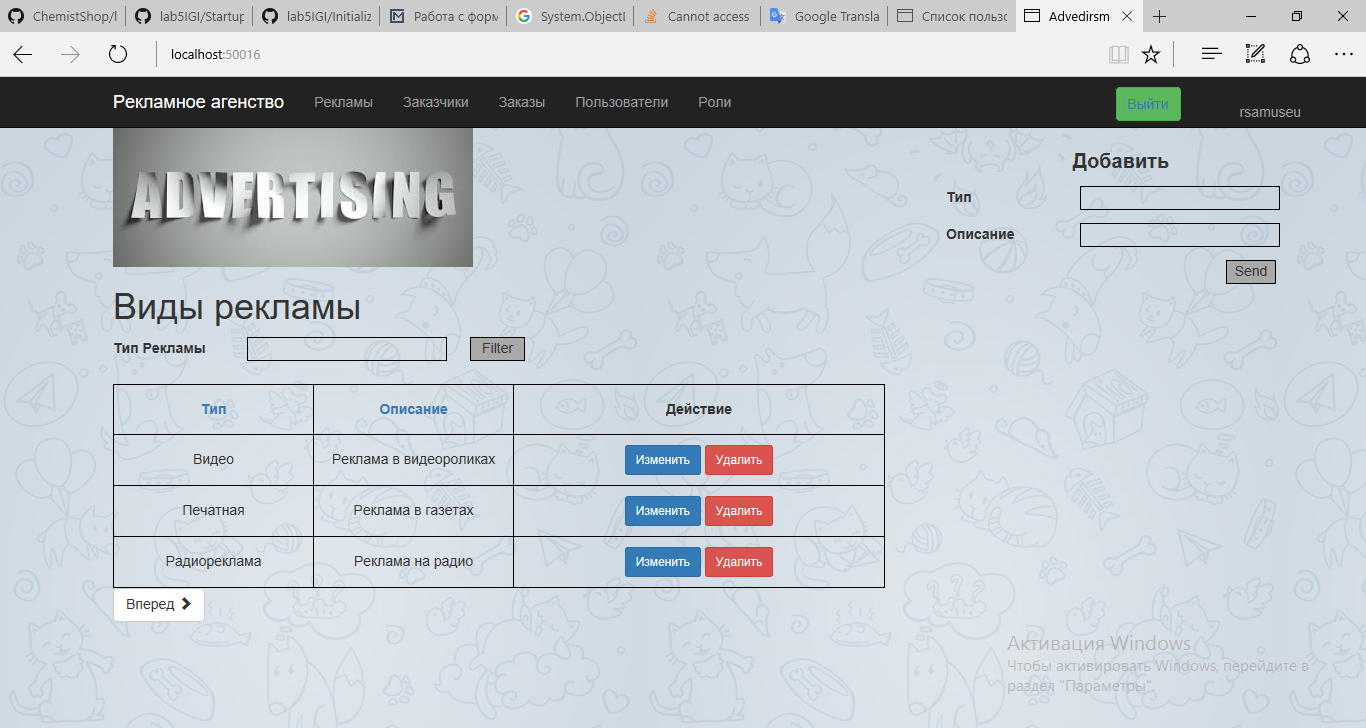
Вход в приложение(представление Login.cshtml):



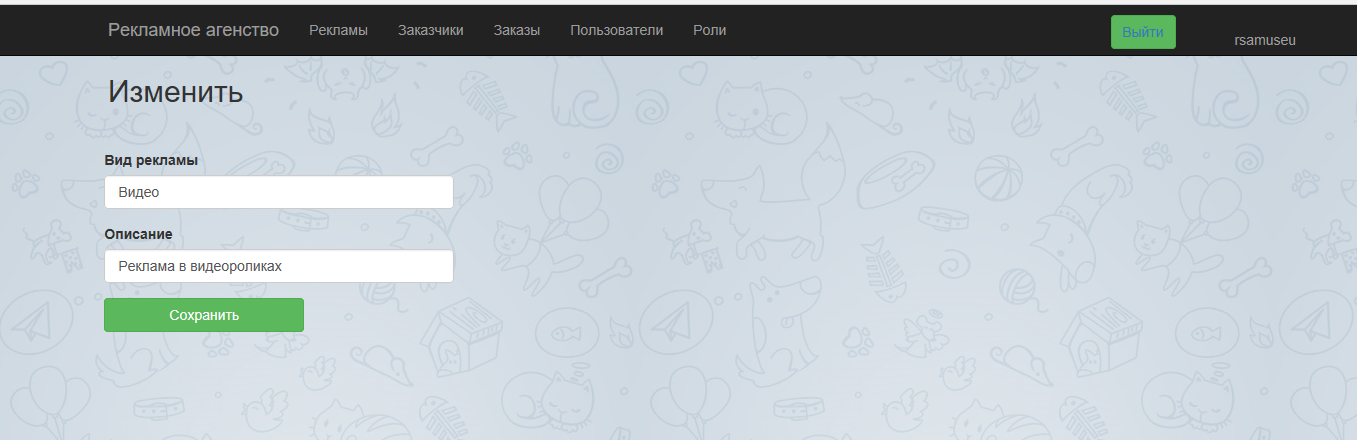
Регистрация(представление Register.cshtml):



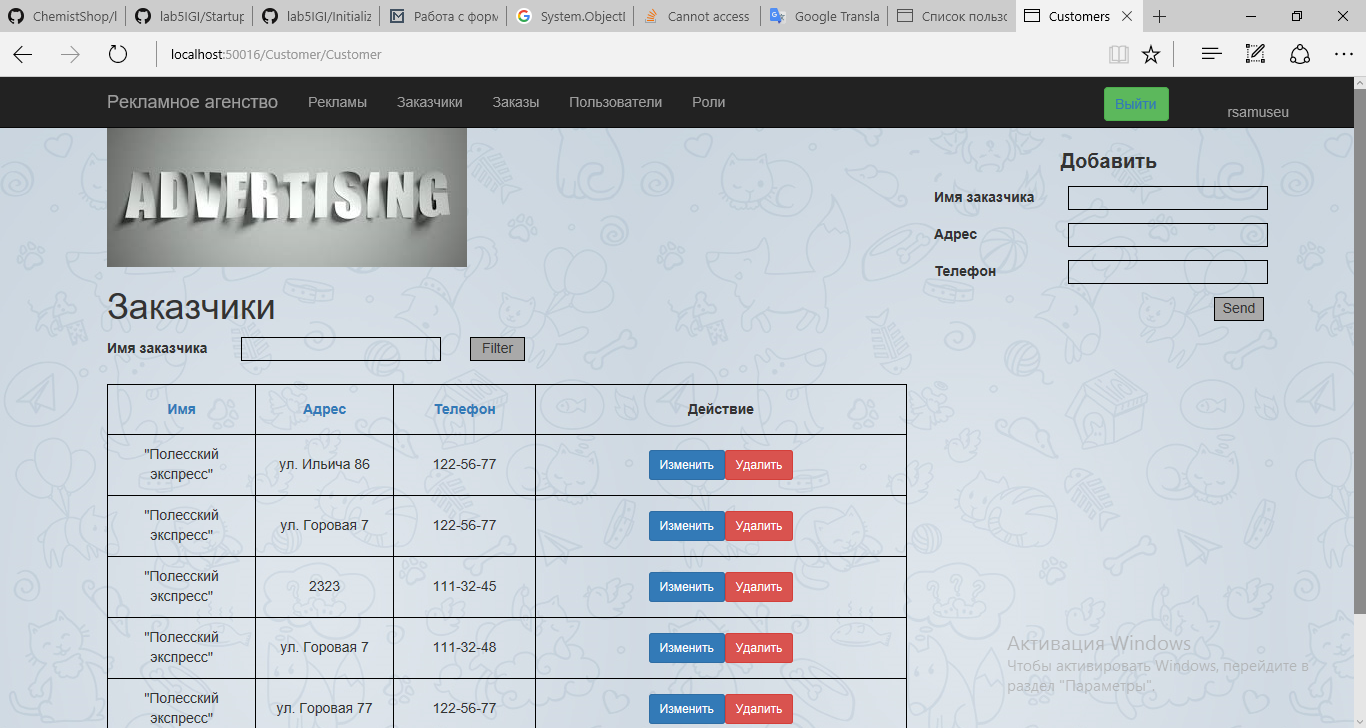
Представление /Home/Index.cshtml:



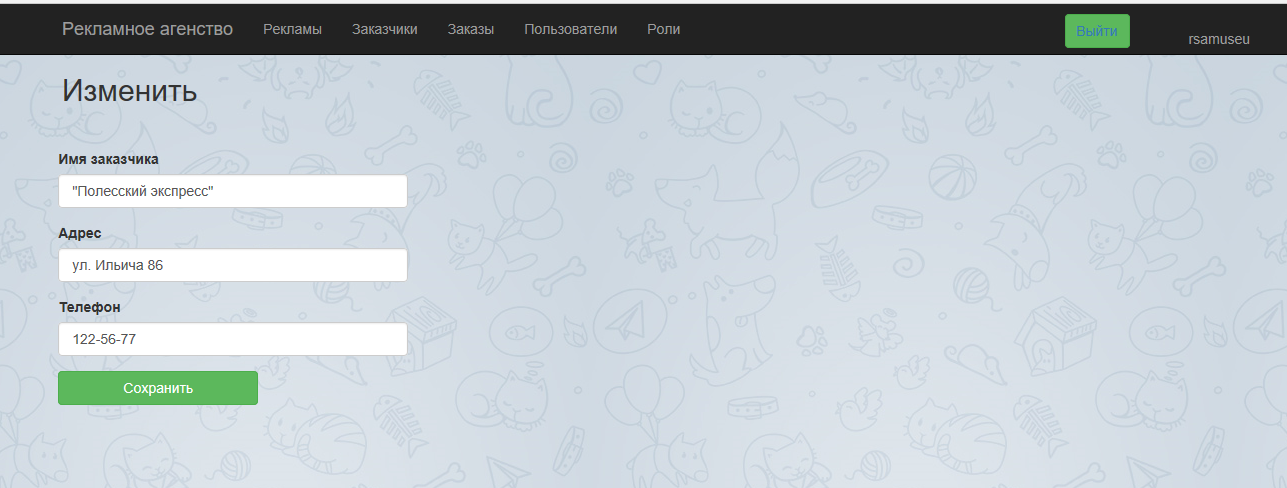
Представление /Home/Edit.cshtml:



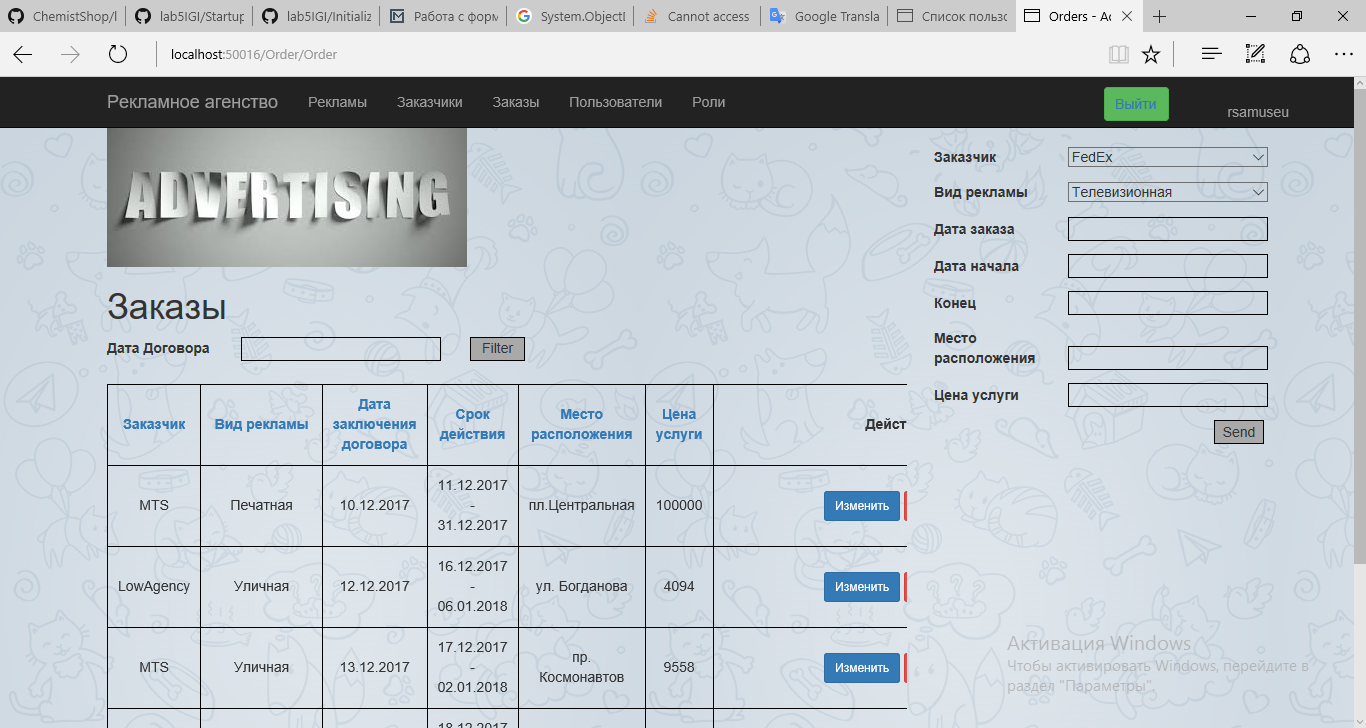
Представление /Customer/Customer.cshtml:



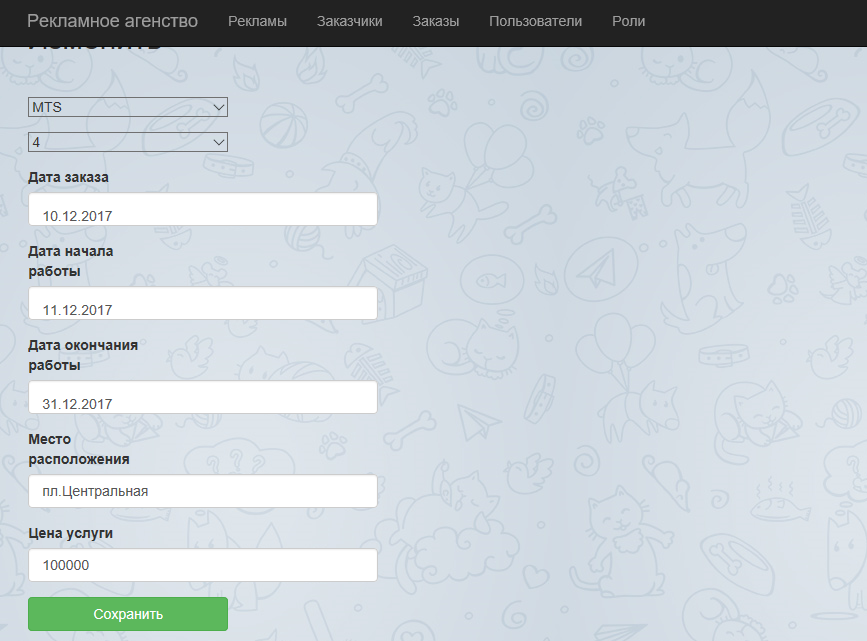
Представление /Customer/Edit.cshtml:



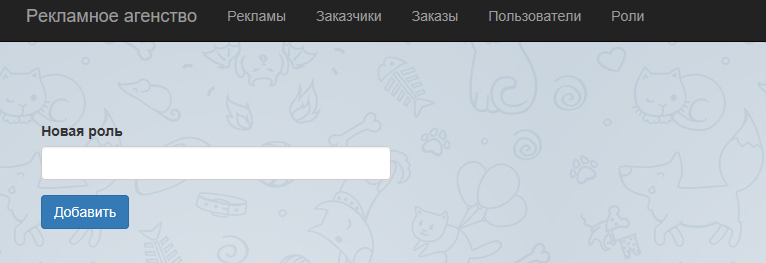
Представление /Order/Order.cshtml:



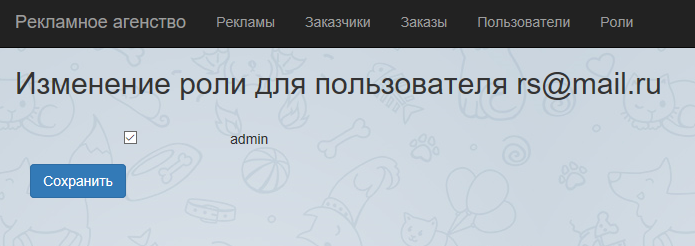
Представление /Order/Edit.cshtml:



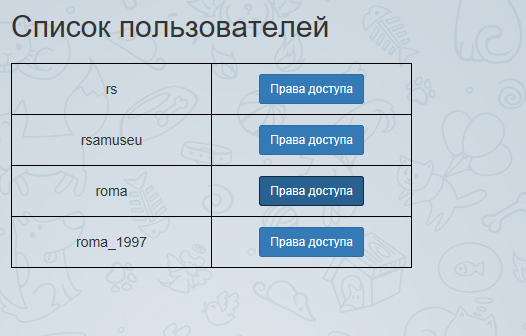
Представление /Roles/Create.cshtml:



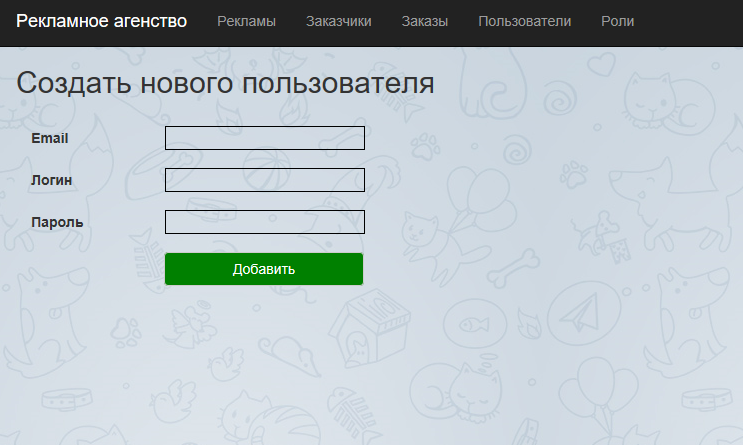
Представление /Roles/Edit.cshtml:



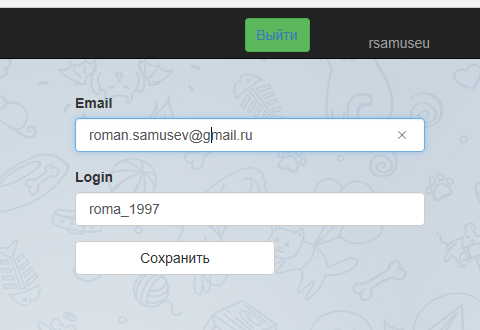
Представление /Roles/UserList.cshtml:



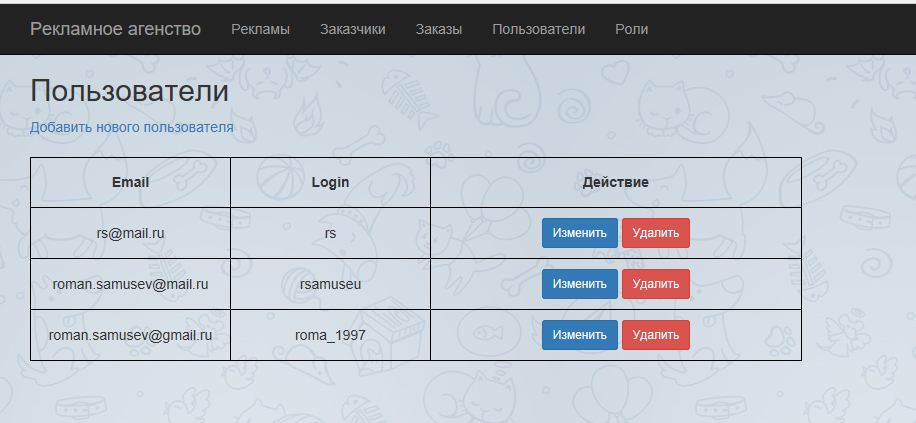
Представление /Users/Create.cshtml:



Представление /Users/Edit.cshtml:

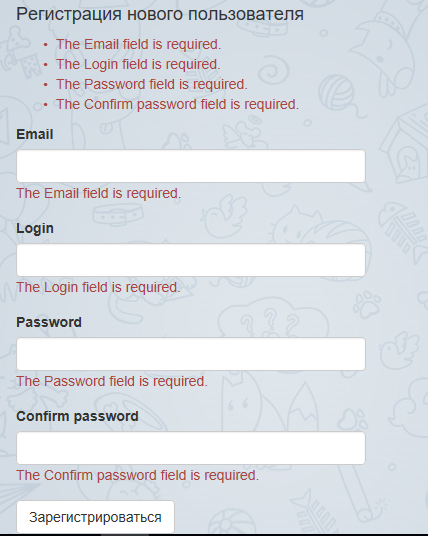


Представление /Users/UsersView.cshtml:

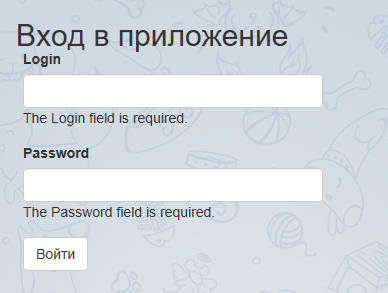


**Копии экранов браузера, демонстрирующие отображение не прошедших валидацию данных:**

Валидация данных при регистрации:



Валидация при входе:



**Вывод:** в ходе выполнения данной лабораторной работы были получены навыки в создании типовых web-приложений для работы с информацией из реляционных баз данных, способностями аутентификации и авторизации в ASP.NET MVC Сore, с администрированием пользователей в asp.net.