**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное

учреждение высшего образования

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

Направление «Информатика и вычислительная техника»

Отчет по лабораторной работе №2 по дисциплине

**«Современные платформы программирования»**

Разработка многостраничного web-сайта по шаблону MVC

Выполнил:

Студент группы 8ВМ82 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.К. Бокижонов

Проверил:

Доцент ОИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Н. Попов

**Задание**

Вариант 9. Реализовать веб-сайт со следующими страницами: административная страница для удаления файлов и их описаний с сайта; пользовательская страница для загрузки файлов и их описаний на сайт; пользовательская страница для поиска файла по ключевым словам описания и скачивания файла.

**Проектирование приложения**

**Архитектура приложения**

Проект разработан на C# ASP.NET Core 2.1 на базе шаблона проектирования MVC. Схема архитектуры MVC приложения представлена на рисунке (рис. 1)



Рис. 1 Схема работы MVC приложения

**Реализация программы:**

Стартовая страница приложения показана на рисунке 2а. Страница фильма предствалена на рисунке 2б. В верхней части находится панель управления, в которой отображаются различные функции в зависимости от авторизации и роли пользователя. В центральном окне отображается список фильмов.

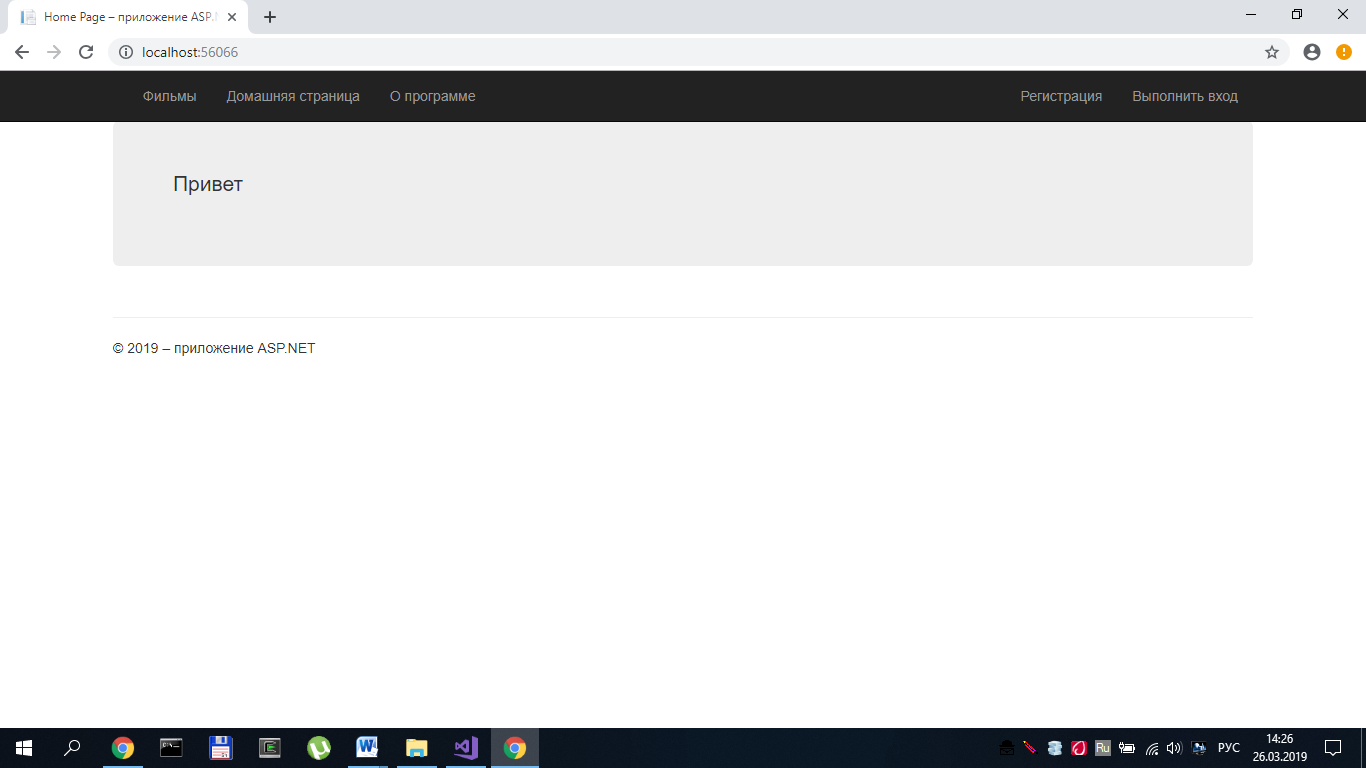


Рис. 2а Главное окно приложения

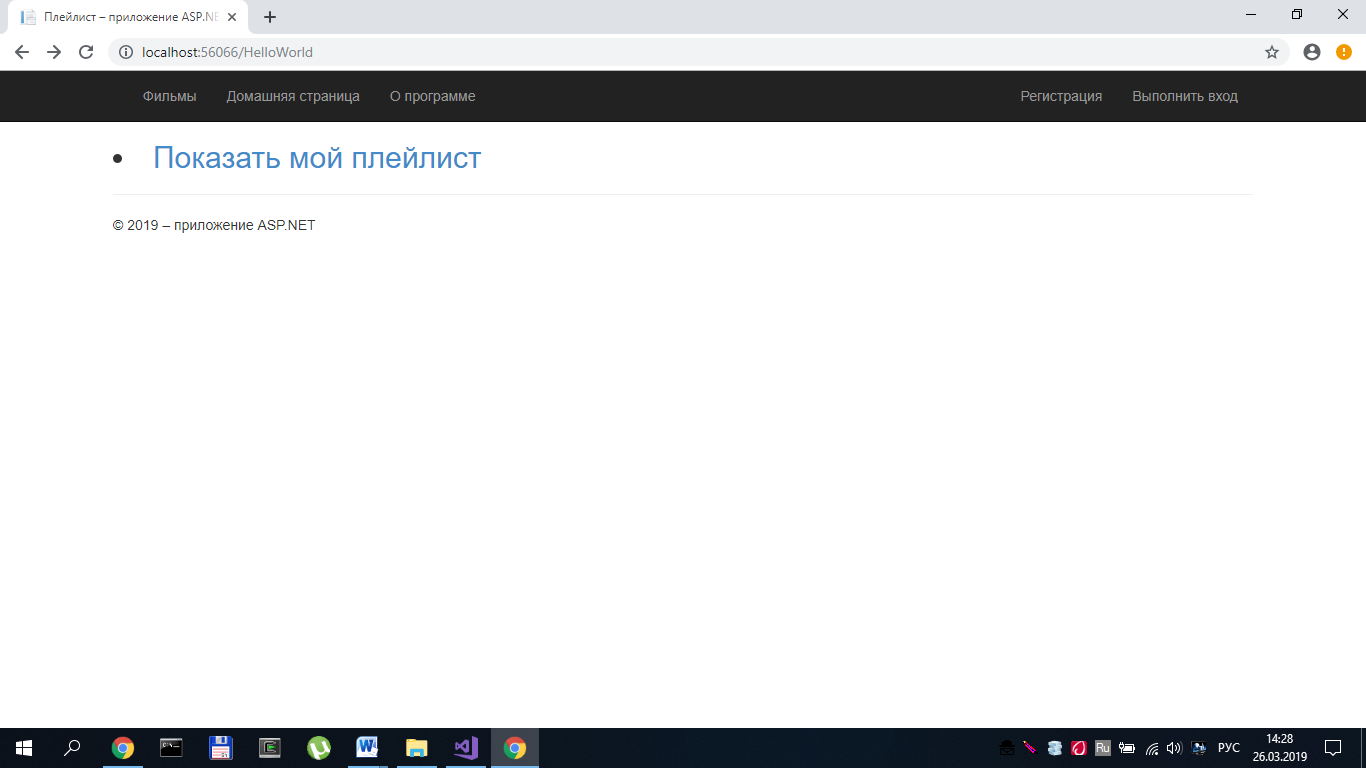
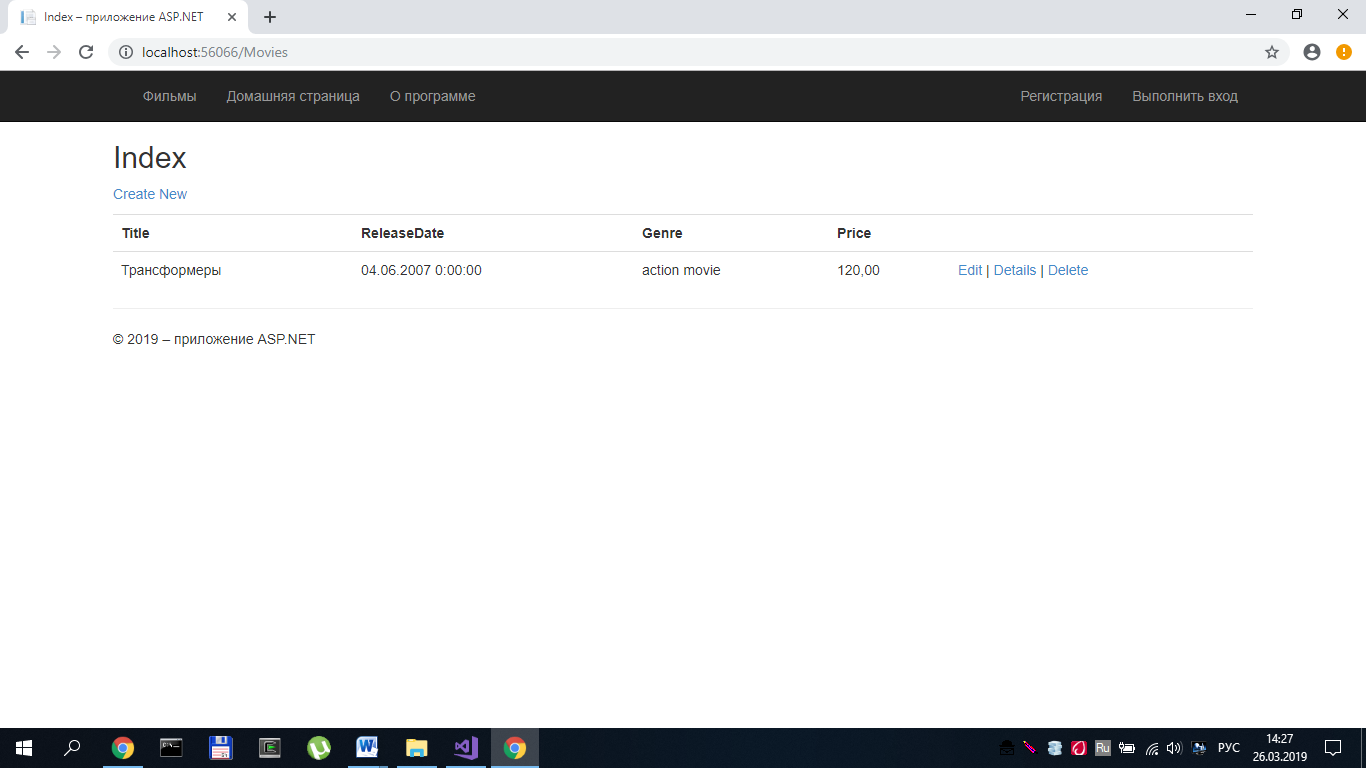


Рис. 2б Страница плейлиста

При попытке отправить составленный заказ (рис. 3) пользователю будет предложено зарегестрироваться или войти (рис. 4)

Рис. 3 Окно плейлисты

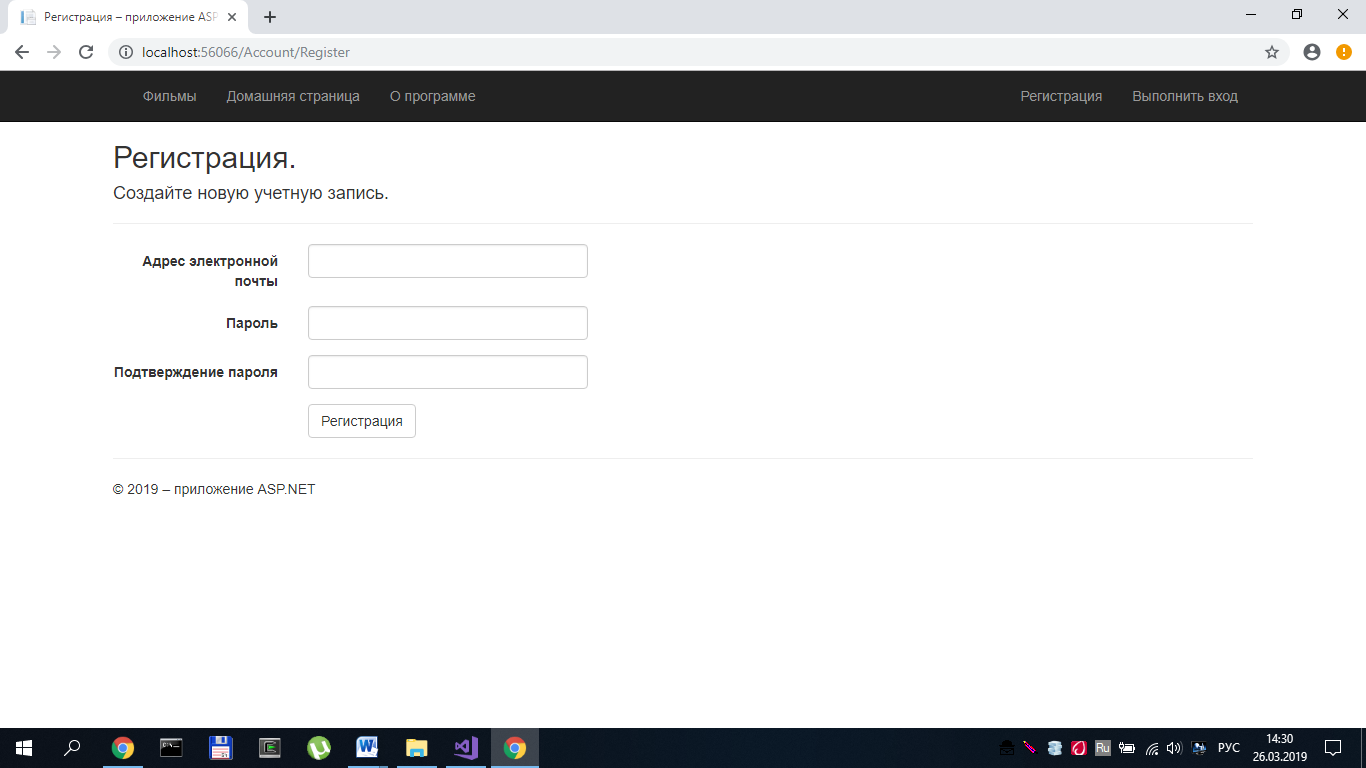


Рис. 4 Окно авторизации

После авторизации в качестве администратора пользователю становится доступен следующий функционал: Просмотр и редактирование данных о пользователях, возможность управлять ролями пользователей, просмотр и редактирование товаров, просмотр и удаление заказов пользователей.

Поскольку приложение было реализовано по шаблону MVC, код представлений находится в файлах view, код контроллеров в файлах controller, а модели данных в файлах models. Некоторые ключевые исходные коды приведены в приложении (Приложение А).

**Заключение**

Было реализовано интернет приложение для продажи фильмов на С#. Приложение полностью соответствует требованиям технического задания, а именно поддерживает все ключевые возможности классического интернет-магазина. Присутствует возможность авторизации в качестве обычного пользователя и администратора. В ходе выполнения лабораторной работы были получены навыки работы с ASP.NET Core и SQL Server, а также углублены знания по языку программирования C#.

**Приложение** А

**Index.cshtml**

@model Store.Models.MovieGenreViewModel

@{

ViewData["Title"] = "Index";

}

<h2>Catalog</h2>

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

{

<p>

<a asp-action="Create">Create New</a>

</p>

}

<form asp-controller="Movies" asp-action="Index" method="get">

<p>

<select asp-for="MovieGenre" asp-items="Model.Genres">

<option value="">All</option>

</select>

Title: <input asp-for="SearchString">

<input type="submit" value="Filter" />

</p>

</form>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Movies[0].Title)

</th>

<th>

@Html.DisplayNameFor(model => model.Movies[0].ReleaseDate)

</th>

<th>

@Html.DisplayNameFor(model => model.Movies[0].Genre)

</th>

<th>

@Html.DisplayNameFor(model => model.Movies[0].Price)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model.Movies)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Title)

</td>

<td>

@Html.DisplayFor(modelItem => item.ReleaseDate)

</td>

<td>

@Html.DisplayFor(modelItem => item.Genre)

</td>

<td>

@Html.DisplayFor(modelItem => item.Price)

</td>

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

{

<td>

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

<a class="btn btn-sm btn-primary" asp-action="Details" asp-route-id="@item.ID">Details</a>

<a class="btn btn-sm btn-danger" asp-action="Delete" asp-route-id="@item.ID">Delete</a>

</td>

}

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

{

<td>

<a class="btn btn-sm btn-primary" asp-controller="ShoppingCart" asp-action ="AddToShoppingCart" asp-route-movieId="@item.ID">Buy</a>

<a class="btn btn-sm btn-primary" asp-action="Details" asp-route-id="@item.ID">Details</a>

</td>

}

</tr>

}

</tbody>

</table>

**MoviesController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using Store.Models;

using Microsoft.AspNetCore.Authorization;

using System.Security.Claims;

using System.IO;

using System.Net.Http.Headers;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using NLog;

namespace Store.Controllers

{

public class MoviesController : Controller

{

private IHostingEnvironment \_environment;

private static Logger logger = LogManager.GetCurrentClassLogger();

private readonly StoreContext \_context;

public MoviesController(StoreContext context, IHostingEnvironment environment)

{

\_context = context;

\_environment = environment;

}

// GET: Movies

public async Task<IActionResult> Index(string movieGenre, string searchString)

{

// Use LINQ to get list of genres.

IQueryable<string> genreQuery = from m in \_context.Movie

orderby m.Genre

select m.Genre;

var movies = from m in \_context.Movie

select m;

if (!String.IsNullOrEmpty(searchString))

{

movies = movies.Where(s => s.Title.Contains(searchString));

}

if (!String.IsNullOrEmpty(movieGenre))

{

movies = movies.Where(x => x.Genre == movieGenre);

}

var movieGenreVM = new MovieGenreViewModel();

movieGenreVM.Genres = new SelectList(await genreQuery.Distinct().ToListAsync());

movieGenreVM.Movies = await movies.ToListAsync();

movieGenreVM.SearchString = searchString;

return View(movieGenreVM);

}

// GET: Movies/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var movie = await \_context.Movie

.FirstOrDefaultAsync(m => m.ID == id);

if (movie == null)

{

return NotFound();

}

logger.Debug("Show the movie DETAILS. ID: " + id);

return View(movie);

}

[Authorize(Roles = "admin")]

// GET: Movies/Create

public IActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("ID,Title,ReleaseDate,Genre,Price,Description, ImageName")] Movie movie)

{

if (ModelState.IsValid)

{

//add picture

var files = HttpContext.Request.Form.Files;

foreach (var Image in files)

{

if (Image != null && Image.Length > 0)

{

var file = Image;

var uploads = Path.Combine(\_environment.WebRootPath, "images");

if (file.Length > 0)

{

var fileName = ContentDispositionHeaderValue.Parse

(file.ContentDisposition).FileName.Trim('"');

System.Console.WriteLine(fileName);

using (var fileStream = new FileStream(Path.Combine(uploads, file.FileName), FileMode.Create))

{

await file.CopyToAsync(fileStream);

movie.ImageName = file.FileName;

}

}

}

}

//add picture end

\_context.Add(movie);

await \_context.SaveChangesAsync();

logger.Debug("CREATE movie. Title: " + movie.Title);

return RedirectToAction(nameof(Index));

}

return View(movie);

}

[Authorize(Roles = "admin")]

// GET: Movies/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var movie = await \_context.Movie.FindAsync(id);

if (movie == null)

{

return NotFound();

}

return View(movie);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("ID,Title,ReleaseDate,Genre,Price,Description, ImageName")] Movie movie)

{

if (id != movie.ID)

{

return NotFound();

}

if (ModelState.IsValid)

{

//add picture

var files = HttpContext.Request.Form.Files;

foreach (var Image in files)

{

if (Image != null && Image.Length > 0)

{

var file = Image;

var uploads = Path.Combine(\_environment.WebRootPath, "images");

if (file.Length > 0)

{

var fileName = ContentDispositionHeaderValue.Parse

(file.ContentDisposition).FileName.Trim('"');

System.Console.WriteLine(fileName);

using (var fileStream = new FileStream(Path.Combine(uploads, file.FileName), FileMode.Create))

{

await file.CopyToAsync(fileStream);

movie.ImageName = file.FileName;

}

}

}

}

//add picture end

try

{

\_context.Update(movie);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!MovieExists(movie.ID))

{

return NotFound();

}

else

{

throw;

}

}

logger.Debug("EDIT movie. Title: " + movie.Title);

return RedirectToAction(nameof(Index));

}

return View(movie);

}

[Authorize(Roles = "admin")]

// GET: Movies/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var movie = await \_context.Movie

.FirstOrDefaultAsync(m => m.ID == id);

if (movie == null)

{

return NotFound();

}

return View(movie);

}

// POST: Movies/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var movie = await \_context.Movie.FindAsync(id);

\_context.Movie.Remove(movie);

await \_context.SaveChangesAsync();

logger.Debug("DELETE movie. Title: " + movie.Title);

return RedirectToAction(nameof(Index));

}

private bool MovieExists(int id)

{

return \_context.Movie.Any(e => e.ID == id);

}

}

}

**ShoppingCartController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Store.Models;

using NLog;

// For more information on enabling MVC for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace Store.Controllers

{

public class ShoppingCartController : Controller

{

private readonly ShoppingCart \_shoppingCart;

private readonly StoreContext \_appDbContext;

private static Logger logger = LogManager.GetCurrentClassLogger();

public ShoppingCartController(StoreContext movieRepository, ShoppingCart shoppingCart)

{

\_appDbContext = movieRepository;

\_shoppingCart = shoppingCart;

}

public ViewResult Index()

{

var items = \_shoppingCart.GetShoppingCartItems();

\_shoppingCart.ShoppingCartItems = items;

var shoppingCartViewModel = new ShoppingCartViewModel

{

ShoppingCart = \_shoppingCart,

ShoppingCartTotal = \_shoppingCart.GetShoppingCartTotal()

};

return View(shoppingCartViewModel);

}

public RedirectToActionResult AddToShoppingCart(int movieId)

{

var selectedMovie = \_appDbContext.Movie.FirstOrDefault(p => p.ID == movieId);

if (selectedMovie != null)

{

\_shoppingCart.AddToCart(selectedMovie, 1);

}

logger.Debug("ADD to cart movie. ID:" + movieId);

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

}

public RedirectToActionResult RemoveFromShoppingCart(int movieId)

{

var selectedMovie = \_appDbContext.Movie.FirstOrDefault(p => p.ID == movieId);

if (selectedMovie != null)

{

\_shoppingCart.RemoveFromCart(selectedMovie);

}

logger.Debug("REMOVE from cart movie. ID:" + movieId);

return RedirectToAction("Index");

}

}

}