Лабораторная работа № 3.  
ASP.Net MVC

**Цель лабораторной работы**

Закрепление теоретических знаний по основам разработки приложений на C# с помощью технологии ASP.NET MVC.

**Постановка задачи**

ASP.NET — это платформа для разработки веб-приложений, созданная компанией Microsoft. Она позволяет разработчикам создавать динамические веб-сайты, веб-приложения и веб-сервисы. ASP.NET поддерживает языки .NET, такие как C#, и предоставляет мощные инструменты и библиотеки для упрощения разработки.

**Основные компоненты ASP.NET**

1. **ASP.NET Web Forms**: Позволяет создавать веб-приложения с использованием событийно-ориентированной модели, аналогичной Windows-приложениям. Подходит для быстрого создания приложений с минимальным количеством кода.
2. **ASP.NET MVC**: Архитектурный паттерн Model-View-Controller, который разделяет приложение на три основных компонента:
   * **Model**: Представляет данные и бизнес-логику.
   * **View**: Отвечает за отображение данных пользователю.
   * **Controller**: Обрабатывает пользовательские запросы, взаимодействует с моделью и выбирает представление для отображения.
3. **ASP.NET Web API**: Позволяет создавать RESTful сервисы, которые могут быть использованы для взаимодействия с клиентскими приложениями, такими как мобильные приложения и одностраничные приложения (SPA).
4. **ASP.NET Core**: Современная версия ASP.NET, которая является кросс-платформенной, высокопроизводительной и более легковесной. Поддерживает создание как веб-приложений, так и API.

Краткий пример реализации выбора города для человека в ASP.NET MVC:

public class City

{

public int Id { get; set; }

public string Name { get; set; }

}

public class Person

{

public int Id { get; set; }

public string Name { get; set; }

public int CityId { get; set; } // Внешний ключ

public City City { get; set; } // Навигационное свойство

}

В контроллере PersonsController необходимо передать города в представление Create:

[HttpGet]

public ActionResult Create()

{

ViewBag.Cities = new SelectList(\_db.Cities, "Id", "Name");

return View();

}

Представление Create.cshtml:

@using WebApplication1.Models

@model Person

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

<div class="form-group">

<**label** **asp-for**="Name">Имя: </**label**>

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

</div>

<div class="form-group">

<**select** **asp-for**="CityId" **asp-items**="ViewBag.Cities" class="form-control">

<**option** **value**="">-- Выберите город --</**option**>

</**select**>

</div>

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

</**form**>

**Задание на лабораторную работу**

1. Разработать ASP.NET MVC приложение для предметной области согласно варианту.
2. **Для оценки отлично** необходимо использовать оба класса из варианта.
3. Реализовать в сервисах CRUD (Create Read Update Delete) операции для классов из варианта.
4. Реализовать хранение данных в любой реляционной базе данных. Взаимодействие с БД должно осуществляться с помощью любой ORM системы.
5. Для успешной защиты:

* программа должна быть задокументирована с помощью комментариев,
* код программы не должен содержать не используемые блоки, должен быть «чистым» и стилистически верно оформлен,
* интерфейс программы должен быть гибким и удобным.

**Варианты заданий на лабораторную работу**

1. **Студент и Экзамен**
   * **Студент**: ФИО, Курс, Номер группы, Возраст
   * **Экзамен**: Предмет, Оценка, Студент
2. **Библиотека и Книга**
   * **Библиотека**: Название, Адрес, Телефон
   * **Книга**: Название, Автор, Жанр, ISBN, Библиотека
3. **Интернет-магазин и Продукт**
   * **Интернет-магазин**: Название, Адрес, Телефон
   * **Продукт**: Название, Цена, Описание, Категория, Интернет-магазин
4. **Блог и Пост**
   * **Блог**: Название, Автор, Дата создания
   * **Пост**: Заголовок, Содержание, Дата публикации, Блог
5. **Задачи и Проект**
   * **Проект**: Название, Описание, Дата начала, Дата окончания
   * **Задача**: Название, Описание, Статус, Проект
6. **Клиент и Заказ**
   * **Клиент**: ФИО, Email, Телефон
   * **Заказ**: Дата, Сумма, Статус, Клиент
7. **Сотрудник и Отдел**
   * **Сотрудник**: ФИО, Должность, Возраст
   * **Отдел**: Название, Местоположение, Сотрудники
8. **Фильм и Рецензия**
   * **Фильм**: Название, Жанр, Год выпуска
   * **Рецензия**: Оценка, Содержание, Фильм
9. **Товары и Корзина**
   * **Товар**: Название, Цена, Описание
   * **Корзина**: Пользователь, Товары
10. **Университет и Факультет**
    * **Университет**: Название, Город, Количество студентов
    * **Факультет**: Название, Декан, Университет
11. **Спортсмен и Команда**
    * **Спортсмен**: ФИО, Возраст, Спорт
    * **Команда**: Название, Тренер, Спортсмены
12. **Ресторан и Меню**
    * **Ресторан**: Название, Адрес, Телефон
    * **Меню**: Название блюда, Цена, Ресторан
13. **Курс и Урок**
    * **Курс**: Название, Описание, Дата начала
    * **Урок**: Название, Дата, Курс
14. **Пользователь и Сообщение**
    * **Пользователь**: Логин, Email, Дата регистрации
    * **Сообщение**: Текст, Дата, Пользователь
15. **Автомобиль и Владелец**
    * **Автомобиль**: Марка, Модель, Год выпуска
    * **Владелец**: ФИО, Адрес, Автомобили
16. **Книга и Автор**
    * **Книга**: Название, Жанр, Год выпуска
    * **Автор**: ФИО, Биография, Книги
17. **Турист и Путешествие**
    * **Турист**: ФИО, Email, Телефон
    * **Путешествие**: Направление, Дата начала, Дата окончания, Турист
18. **Сайт и Комментарий**
    * **Сайт**: Название, URL, Дата создания
    * **Комментарий**: Текст, Дата, Сайт
19. **Группа и Участник**
    * **Группа**: Название, Описание, Дата создания
    * **Участник**: ФИО, Роль, Группа
20. **Магазин и Отзыв**
    * **Магазин**: Название, Адрес, Телефон
    * **Отзыв**: Оценка, Текст, Магазин

**Пример разработки приложения**

Рассмотрим пример приложения для управления товарами.

**Шаг 1: Установка**

Необходимо открыть Visual Studio Installer, выбрать «ASP.NET и разработка веб-приложений» (рисунок 1) и дождаться окончания установки.

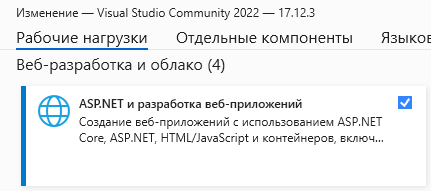


Рисунок 1 – Установка

**Шаг 2: Создание проекта**

При создании проекта необходимо выбрать «Пустой шаблон ASP.NET Core (Майкрософт)» (рисунок 2).

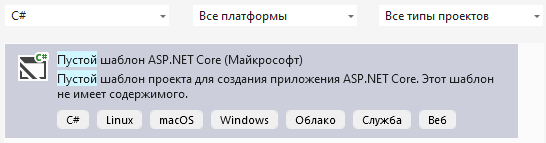


Рисунок 2 – Создание проекта

**Шаг 3: Установка пакетов**

Откройте консоль диспетчера пакетов (либо через NuGet) и выполните следующие команды для установки необходимых пакетов:

Install-Package Microsoft.EntityFrameworkCore.Sqlite

Install-Package Microsoft.EntityFrameworkCore.Tools

**Шаг 4: Иерархия классов и папок в решении**

Создать папки и классы согласно рисунку 3.

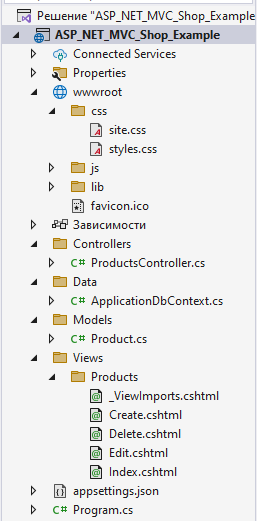


Рисунок 3 – Иерархия

**Шаг 5: Создание модели**

using System.ComponentModel.DataAnnotations;

namespace ASP\_NET\_MVC\_Shop\_Example.Models

{

public class Product

{

public int Id { get; set; }

[Required]

[StringLength(100)]

public string? Name { get; set; }

[Required]

[Range(0.01, double.MaxValue, ErrorMessage = "Цена должна быть не менее 0,01")]

public decimal Price { get; set; }

[StringLength(500)]

public string? Description { get; set; }

}

}

**Шаг 6: Создание контекста базы данных**

using ASP\_NET\_MVC\_Shop\_Example.Models;

using Microsoft.EntityFrameworkCore;

namespace ASP\_NET\_MVC\_Shop\_Example.Data

{

public class ApplicationDbContext : DbContext

{

public DbSet<Product> Products { get; set; }

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

Database.EnsureCreated();

}

}

}

**Шаг 7: Контроллер**

using ASP\_NET\_MVC\_Shop\_Example.Data;

using ASP\_NET\_MVC\_Shop\_Example.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

namespace ASP\_NET\_MVC\_Shop\_Example.Controllers

{

public class ProductsController : Controller

{

private readonly ApplicationDbContext \_context;

public ProductsController(ApplicationDbContext context)

{

\_context = context;

}

// GET: Products

public async Task<IActionResult> Index()

{

var products = await \_context.Products.ToListAsync();

return View(products);

}

// GET: Products/Create

public IActionResult Create()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Create(Product product)

{

if (ModelState.IsValid)

{

\_context.Products.Add(product);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(product);

}

// GET: Products/Edit/

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

{

if (id == null)

return NotFound();

var product = await \_context.Products.FindAsync(id);

if (product == null)

return NotFound();

return View(product);

}

// POST: Products/Edit/

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(Product product)

{

if (ModelState.IsValid)

{

\_context.Update(product);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(product);

}

// GET: Products/Delete/

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

{

if (id == null)

return NotFound();

var product = await \_context.Products.FindAsync(id);

if (product == null)

return NotFound();

return View(product);

}

// POST: Products/Delete/

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Delete(Product product)

{

\_context.Products.Remove(product);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

}

}

**Шаг 8: Создание представлений**

Представление для списка товаров (Index.cshtml)

@using ASP\_NET\_MVC\_Shop\_Example.Models

<head>

<**link** rel="stylesheet" href="~/css/styles.css" />

</head>

@model IEnumerable<Product>

<h2>Список товаров</h2>

<p>

<**a** class="button button-green" **asp-action**="Create">Добавить товар</**a**>

</p>

<table class="styled-table">

<thead>

<tr>

<th>Название</th>

<th>Цена</th>

<th>Описание</th>

<th></th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var product in Model)

{

<tr>

<td>@product.Name</td>

<td>@product.Price.ToString("C")</td>

<td>@product.Description</td>

<td>

<**a** class="button button-yellow" **asp-action**="Edit" **asp-route-id**="@product.Id">Редактировать</**a**>

</td>

<td>

<**a** class="button button-red" **asp-action**="Delete" **asp-route-id**="@product.Id">Удалить</**a**>

</td>

</tr>

}

</tbody>

</table>

Представление для добавления товара (Create.cshtml)

@using ASP\_NET\_MVC\_Shop\_Example.Models

<head>

<**link** rel="stylesheet" href="~/css/styles.css" />

</head>

@model Product

<div class="form-container">

<h2>Добавить товар</h2>

<**form** **asp-action**="create" **asp-controller**="products">

<p>

<**label** **asp-for**="Name">Название</**label**><br />

<**input** **type**="text" **asp-for**="Name" /><br />

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

</p>

<p>

<**label** **asp-for**="Price">Цена, руб</**label**><br />

<**input** **type**="number" step=".01" **asp-for**="Price" /><br />

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

</p>

<p>

<**label** **asp-for**="Description">Описание</**label**><br />

<**textarea** type="text" **asp-for**="Description"></**textarea**>

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

</p>

<p>

<input class="button button-green" type="submit" value="Добавить" />

</p>

</**form**>

</div>

Представление для редактирования товара (Edit.cshtml)

@using ASP\_NET\_MVC\_Shop\_Example.Models

<head>

<**link** rel="stylesheet" href="~/css/styles.css" />

</head>

@model Product

<div class="form-container">

<h2>Редактировать товар</h2>

<**form** class="form-container" **asp-action**="Edit" **asp-route-id**="@Model.Id">

<p>

<**label** **asp-for**="Name">Название</**label**><br />

<**input** **type**="text" **asp-for**="Name" /><br />

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

</p>

<p>

<**label** **asp-for**="Price">Цена, руб</**label**><br />

<**input** **type**="number" step=".01" **asp-for**="Price" />

<br />

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

</p>

<p>

<**label** **asp-for**="Description">Описание</**label**><br />

<**textarea** type="text" **asp-for**="Description"></**textarea**>

<br />

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

</p>

<input class="button button-green" type="submit" value="Сохранить" />

</**form**>

<**a** class="button button-blue" **asp-action**="Index">Назад к списку</**a**>

</div>

Представление для удаления товара (Delete.cshtml)

@using ASP\_NET\_MVC\_Shop\_Example.Models

<head>

<**link** rel="stylesheet" href="~/css/styles.css" />

</head>

@model Product

<div class="form-container">

<h2>Удалить товар</h2>

<h3>Вы уверены, что хотите удалить товар @Model.Name?</h3>

<**form** **asp-action**="Delete" **asp-route-id**="@Model.Id">

<input class="button button-red" type="submit" value="Удалить" />

</**form**>

<**a** class="button button-blue" **asp-action**="Index">Назад к списку</**a**>

</div>

Представление для подключения классов-помощников (без него проект запустится, но маршрутизация будет сломана и будут некорректные переходы по ссылкам)

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

**Шаг 9: Конфигурация проекта (appsettings.json)**

{

"ConnectionStrings": {

"DefaultConnection": "Data Source=products.db;"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**Шаг 10: Точка входа и настройка маршрутизации**

using ASP\_NET\_MVC\_Shop\_Example.Data;

using Microsoft.EntityFrameworkCore;

namespace ASP\_NET\_MVC\_Shop\_Example

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// получаем строку подключения из файла конфигурации

string connection = builder.Configuration.GetConnectionString("DefaultConnection");

// добавляем контекст ApplicationContext в качестве сервиса в приложение

builder.Services.AddDbContext<ApplicationDbContext>(options => options.UseSqlite(connection));

builder.Services.AddControllersWithViews();

var app = builder.Build();

app.UseHttpsRedirection();

app.UseStaticFiles(); // Это необходимо для передачи статических файлов, таких как CSS

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Products}/{action=Index}/{id?}");

app.Run();

}

}

}

**Шаг 11: Стилевой файл (styles.css – не обязателен, исключительно для декора)**

Если не добавлять, то в каждом представлении нужно удалить ссылку на стилевой файл.

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

/\* Цвет фона страницы \*/

margin: 0;

padding: 20px;

}

.field-validation-error {

color: red;

font-size: 12px;

}

input.input-validation-error {

border: 1px solid red;

background-color: rgb(255, 221, 221, 0.2);

}

input.valid, textarea.valid {

border: 1px solid green;

background-color: rgb(210, 255, 210, 0.2);

}

textarea {

resize: vertical;

}

.form-container {

background-color: #ffffff; /\* Цвет фона формы \*/

border-radius: 10px; /\* Закругленные углы \*/

padding: 20px; /\* Отступы внутри формы \*/

box-shadow: 0 0 15px rgba(0, 0, 0, 0.1); /\* Тень под формой \*/

max-width: 400px; /\* Максимальная ширина формы \*/

margin: auto; /\* Центрирование формы \*/

}

input,

textarea {

width: 100%; /\* Ширина полей ввода \*/

padding: 10px; /\* Отступы внутри полей ввода \*/

border: 1px solid #cccccc; /\* Граница полей ввода \*/

border-radius: 5px; /\* Закругленные углы полей ввода \*/

transition: border-color 0.3s; /\* Плавный переход цвета границы \*/

}

input:focus,

textarea:focus {

border-color: #007bff; /\* Цвет границы при фокусе \*/

outline: none; /\* Убираем обводку при фокусе \*/

box-shadow: 0 0 5px rgba(0, 123, 255, 0.5); /\* Тень при фокусе \*/

}

.styled-table {

width: 100%;

/\* Ширина таблицы \*/

border-collapse: collapse;

/\* Убираем двойные границы \*/

margin: 20px 0;

/\* Отступы вокруг таблицы \*/

border-radius: 10px;

/\* Закругленные углы \*/

overflow: hidden;

/\* Убираем выступающие углы \*/

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

/\* Тень под таблицей \*/

}

.styled-table thead {

background-color: #007bff;

/\* Цвет фона заголовка \*/

color: #ffffff;

/\* Цвет текста заголовка \*/

}

.styled-table th,

.styled-table td {

padding: 12px 15px;

/\* Отступы внутри ячеек \*/

text-align: left;

/\* Выравнивание текста \*/

}

.styled-table tbody tr {

background-color: #ffffff;

/\* Цвет фона для строк \*/

transition: background-color 0.3s;

/\* Плавный переход цвета фона \*/

}

.styled-table tbody tr:hover {

background-color: #f1f1f1;

/\* Цвет фона при наведении на строку \*/

}

.styled-table tbody td {

border-bottom: 1px solid #dddddd;

/\* Нижняя граница ячеек \*/

}

.button {

display: inline-block;

/\* Позволяет добавлять отступы и размеры \*/

padding: 10px 20px;

/\* Отступы внутри кнопки \*/

color: rgb(255, 255, 255);

font-weight: bold;

/\* Жирный текст \*/

/\* Цвет текста \*/

text-align: center;

/\* Центрирование текста \*/

text-decoration: none;

/\* Убираем подчеркивание \*/

border-radius: 5px;

/\* Закругленные углы \*/

border: none;

/\* Убираем рамку \*/

transition: background-color 0.3s ease;

/\* Плавный переход цвета фона \*/

}

.button-green {

background-color: rgb(0, 190, 0);

/\* Цвет фона кнопки \*/

}

.button-green:hover {

background-color: rgb(0, 225, 0);

/\* Цвет фона при наведении \*/

}

.button-green:active {

background-color: rgb(0, 255, 0);

/\* Цвет фона при нажатии \*/

}

.button-yellow {

background-color: rgb(190, 190, 0);

}

.button-yellow:hover {

background-color: rgb(225, 225, 0);

}

.button-yellow:active {

background-color: rgb(255, 255, 0);

}

.button-red {

background-color: rgb(190, 0, 0);

}

.button-red:hover {

background-color: rgb(225, 0, 0);

}

.button-red:active {

background-color: rgb(255, 0, 0);

}

.button-blue {

background-color: rgb(0, 100, 190);

}

.button-blue:hover {

background-color: rgb(0, 145, 225);

}

.button-blue:active {

background-color: rgb(0, 190, 255);

}

На рисунке 4 представлено главное окно программы с примером товаров.

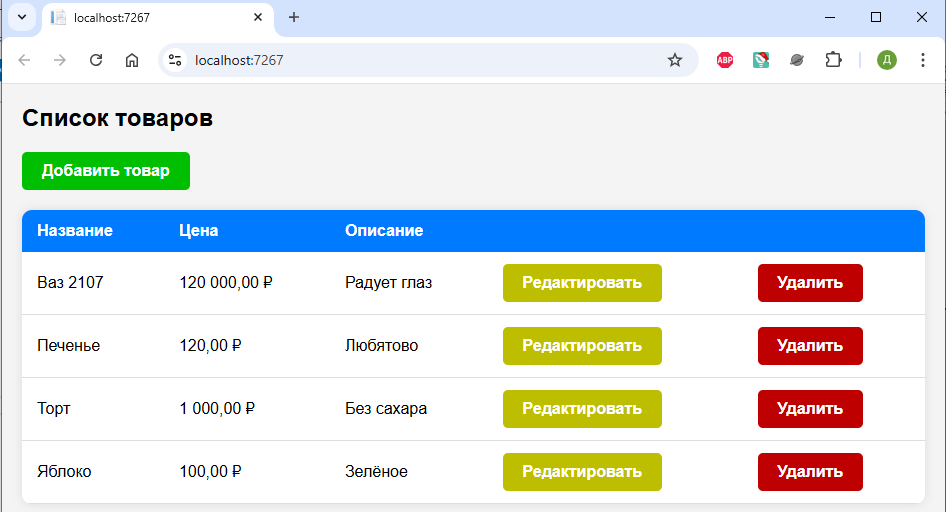


Рисунок 4 – Главное окно программы

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»