|  |  |  |
| --- | --- | --- |
| **МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА** | | |
| **ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО ЕЛЕКТРОТЕХНИКА И ЕЛЕКТРОНИКА**  бул. Пещерско шосе № 26  4002 гр. Пловдив, България  тел. 032 / 643-657  info-1690174@edu.mon.bg |  | **VOCATIONAL SCHOOL OF ELECTRICAL ENGINEERING AND ELECTRONICS**  26 Peshtersko chaussee blvd.,  4002 Plovdiv, Bulgaria  Phone: 032 / 643-657  info-1690174@edu.mon.bg |

**професия код 481030 „Приложен програмист“**

**специалност код 4810301 „Приложно програмиране“**

**Курсов Проект**

**Тема:Изграждане на приложение „Каталог за игри“**

**Разработили**: …………… **Ръководител-консултант**: …………

/Илиян Куршумов, /инж. Ваня Менгова/

Ангел Масарлиев,

Божидар Иванов/

Клас: 12а

e-mail: ilijan.kurshumov@gmail.com

Пловдив

2025 година

# Съдържание

[Съдържание 1](#_Toc184753971)

[Увод 1](#_Toc184753972)

[Глава 1 - Технологии и Концепции 2](#_Toc184753973)

[1.1. Asp.Net Core 2](#_Toc184753974)

[1.2. MVC 2](#_Toc184753975)

[1.3. EntityFrameworkCore 3](#_Toc184753976)

[1.4. База данни 3](#_Toc184753977)

[1.5. Езици за програмиране 4](#_Toc184753978)

[Глава 2 – Структура на базата данни 5](#_Toc184753979)

[2.1. Структура на таблица и данни 5](#_Toc184753980)

[Глава 3 – Реализация на Програмата 6](#_Toc184753981)

[3.1. Създаване на база от данни 6](#_Toc184753982)

[3.2. Game контролер 8](#_Toc184753983)

[3.3. Javascript метод за търсачка 9](#_Toc184753984)

[3.4. Game Container 9](#_Toc184753985)

[Глава 4 - ръководство за потребителя 12](#_Toc184753986)

[4.1. Описание и ръководство за ползване на Game Catalog 12](#_Toc184753987)

[Заключение 13](#_Toc184753988)

[Използвани Източници 14](#_Toc184753989)

[Приложение 1 - GameController 15](#_Toc184753990)

[Приложение 2 22](#_Toc184753991)

[Приложение 3 26](#_Toc184753992)

[Приложение 4 28](#_Toc184753993)

[Приложение 5 30](#_Toc184753994)

[Приложение 6 33](#_Toc184753995)

# Увод

Проектът представлява платформа за управление на каталог за игри, създадена с цел да бъде удобна и функционална за потребителите. Основната цел е да се осигури интуитивен и ефективен начин за работа с информация за игри.

Приложението предоставя възможност на потребителите да разглеждат наличните игри, да добавят нови записи, да редактират информацията за вече съществуващи игри, както и да изтриват записи, ако е необходимо. В допълнение към тези основни функционалности е добавена опция за бързо и лесно търсене чрез **търсачка (search bar)**, която улеснява намирането на конкретни игри в каталога.

Тази платформа е разработена, за да предложи както удобство за крайния потребител, така и възможност за разширяване с допълнителни функционалности в бъдеще.

# Глава 1 - Технологии и Концепции

## 1.1. Asp.Net Core

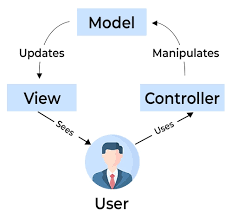
**ASP.NET Core** е модерна, платформа-независима и с отворен код уеб рамка, разработена от Microsoft. Тя позволява създаването на уеб приложения, услуги и API-та с висока производителност. Благодарение на своята модулна архитектура и платформена независимост, приложенията могат да работят на Windows, macOS и Linux.

Рамката е оптимизирана за облачни услуги, като Azure, и предлага мощни инструменти като Razor Pages и Model-View-Controller (MVC). Със своите висока скорост и гъвкавост, ASP.NET Core е идеален избор за съвременни, мащабируеми и сигурни уеб приложения.

## 1.2. MVC

MVC (Model-View-Controller) е архитектурен шаблон в програмирането, който разделя бизнес логиката на три свързани компонента: Model, View и Controller. Model представлява "ядрото" на приложението и съдържа всички данни, с които се работи. View е изходната част на софтуера, която визуализира обработените данни за потребителя. Controller извиква методи върху данните в модела, за да ги обработи и подготви за визуализация.

Тази структура осигурява ясно разделение на отговорностите, прави приложенията по-лесни за разработка и поддръжка и е основна част от работната рамка ASP.NET Core(Фигура 1).



Фигура 1. Визуализация на MVC

## 1.3. EntityFrameworkCore

С **Entity Framework Core (EF Core)** е лек и разширяем ORM (Object-Relational Mapping) инструмент, който позволява на разработчиците да работят с бази данни чрез .NET обекти, вместо да пишат SQL код. EF Core автоматично преобразува операциите като създаване, четене, актуализиране и изтриване на данни в SQL команди.

Той поддържа различни бази данни (SQL Server, MySQL, PostgreSQL и други) и позволява използването на LINQ за създаване на заявки. EF Core включва функции за миграции, които улесняват управлението на промените в базата данни, и предлага поддръжка за различни типове взаимоотношения между таблици. Това прави работата с бази данни по-бърза, лесна и ефективна.

## 1.4. База данни

База данни е организирана колекция от данни, които се съхраняват и управляват по структуриран начин, позволявайки лесен достъп, манипулация и обновяване. Базите данни се използват за много видове проекти, в нашия случай – за управление на каталог за игри. Например, в базата данни на каталога се съхраняват информация за игрите като имена, жанрове, платформи, дати на издаване и оценки. Това позволява на потребителите лесно да разглеждат, добавят, редактират или изтриват игри, както и да търсят конкретни заглавия в каталога.

SQL (Structured Query Language) е език за управление на релационни бази данни, който се използва за създаване, четене, актуализиране и изтриване на данни. В контекста на каталог за игри, SQL може да се използва за извличане на информация за конкретна игра, добавяне на нови записи или актуализиране на съществуващи.

## 1.5. Езици за програмиране

* + 1. **C#**: Езикът за програмиране, използван в сърцето на проекта, чрез който се реализира логиката на приложението, като работа с базата данни и обработка на потребителски заявки.
    2. **HTML**: Използва се за структурата на уеб страниците, създавайки основните елементи като заглавия, параграфи и формуляри, които потребителите взаимодействат.
    3. **CSS**: Отговаря за стилизирането и оформлението на уеб страниците, осигурявайки визуално привлекателен и подреден изглед на интерфейса.
    4. **JavaScript**: Използва се за добавяне на динамичност и интерактивност към сайта, като например обработка на събития от потребителския интерфейс и взаимодействие с данни в реално време.

# Глава 2 – Структура на базата данни

## 2.1. Структура на таблица и данни

В нашия проект се използва релационна SQL база от данни, във вид на таблица. Присъстващи колони в нея са: Name, Manufacturer, Description, Instruction, Rating, ImageUrl (Фигура 2).

|  |  |
| --- | --- |
| **Column** | **Type** |
| Name | [string] |
| Manufacturer | [string] |
| Description | [string] |
| Instruction | [string] |
| Rating | [number] |
| ImageUrl | [string] |

Фигура 2. Колоните и техните типове

## 

# Глава 3 – Реализация на Програмата

## 3.1. Създаване на база от данни

За създаване на базата от данни първо трябва да направим **model** с колоните, които искаме да има нашата таблица и техните типове на данните.

Модел Game:

public int Id { get; set; }

[Required]

[StringLength(100)]

public string Name { get; set; } = string.Empty;

[Required]

[StringLength(100)]

public string Manufacturer { get; set; } = string.Empty;

public string Description { get; set; } = string.Empty;

public string Instructions { get; set; } = string.Empty;

[Range(0, 5)]

public double Rating { get; set; }

public string ImageUrl { get; set; } = string.Empty;

След това дефинираме ApplicationDbContext, който служи като контекст за базата данни в приложението. Той наследява DbContext от Entity Framework Core и конфигурира връзката към базата данни чрез конструктора. Чрез свойството DbSet<Game> контекстът свързва модела Game.

ApplicationDbContext:

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options) { }

public DbSet<Game> Games { get; set; }

}

След това в appsetting.json файла променяме връзката, така че да се свърже с базата данни. В program.cs конфигурираме новия контекст на базата и в в конзолата създаваме миграция и обновяваме базата.

## 3.2. Game контролер

GameController е контролер в ASP.NET Core MVC приложение, който управлява CRUD операциите за игри в каталога, използвайки базата данни чрез ApplicationDbContext. Той съдържа следните основни действия: Преглед на игрите, създаване на нова игра, редактиране на вече създадена игра, преглед на игра, изтриване на игра и търсене на игра чрез търсачка.

Method за търсачка:

[HttpGet]

public IActionResult Search(string searchQuery)

{

var games = \_context.Games.AsQueryable();

if (!string.IsNullOrEmpty(searchQuery))

{

games = games.Where(g =>

g.Name.Contains(searchQuery) ||

g.Manufacturer.Contains(searchQuery));

}

return PartialView("\_GameList", games.ToList());

}

## 3.3. Javascript метод за търсачка

Този JavaScript код добавя динамично търсене на игри. Когато потребителят въвежда текст в полето с ID searchInput, се изпраща заявка към /Game/Search с текущия текст като параметър searchQuery. След като сървърът върне новия HTML с резултатите от търсенето, съдържанието на елемента с ID gameList се обновява, показвайки актуализирания списък с игри. В случай на грешка се извежда съобщение в конзолата. Това позволява на потребителите да виждат резултатите в реално време без презареждане на страницата.

Код:

document.getElementById('searchInput').addEventListener('input', function () {

const query = this.value;

fetch(`/Game/Search?searchQuery=${query}`)

.then(response => response.text())

.then(html => {

document.getElementById('gameList').innerHTML = html;

})

.catch(error => console.error('Error:', error));

});

## 3.4. Game Container

Този HTML и CSS код показва игрите в отделни кутии вместо в таблица, като всяка кутия съдържа снимка, име, производител и бутони за действия като "Детайли" и "Редактиране". Дизайнът използва стилове за закръглени ръбове, сенки и анимации при задържане с мишката, което прави интерфейса по-модерен и привлекателен. Това подобрява визуалното представяне на каталога и улеснява потребителите при разглеждането на игрите.

Код:

HTML:

<div class="game-container">

@foreach (var game in Model)

{

<div class="game-box">

<img src="@game.ImageUrl" alt="@game.Name" />

<h3>@game.Name</h3>

<p>@game.Manufacturer</p>

<button onclick="location.href='@Url.Action("Details", new { id = game.Id })'">Детайли</button>

<button onclick="location.href='@Url.Action("Edit", new { id = game.Id })'">Редактиране</button>

</div>

}

</div>

CSS:

.game-container {

display: flex;

flex-wrap: wrap;

justify-content: center;

padding: 20px;

}

.game-box {

background: rgba(255, 255, 255, 0.1);

border-radius: 10px;

padding: 20px;

margin: 15px;

width: 250px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2), 0 0 15px rgba(138, 43, 226, 0.8);

transition: transform 0.3s ease, box-shadow 0.3s ease;

}

# Глава 4 - ръководство за потребителя

## 4.1. Описание и ръководство за ползване на Game Catalog

Приложението Game Catalog позволява управление на каталог с игри чрез интуитивен интерфейс. На началната страница се показва списък с игри във вид на кутии, съдържащи снимка, име, производител и бутони за действия като преглед на детайли или редактиране. В горната част е разположено поле за търсене, което обновява резултатите в реално време при въвеждане на текст. Потребителите могат да добавят нови игри чрез специален бутон, който отваря формуляр за попълване на информация като име, производител, описание, рейтинг и URL на изображение. Всяка игра може да бъде прегледана детайлно, редактирана или изтрита от каталога

# Заключение

Проектът "Каталог за игри" представлява ефективно решение за управление на игри, използващо ASP.NET Core MVC и SQL база данни. Чрез предоставените функционалности за добавяне, редактиране, преглеждане на детайли, изтриване, преглеждане на всички игри и търсене, потребителите и администраторите могат лесно да взаимодействат с каталога, като добавят нови записи, актуализират информация, или търсят игри.

Бъдещо развитие на проекта включва добавяне на нови функции като система за потребителски рецензии и оценки на игрите и възможност за създаване на списъци с любими игри.

# Използвани Източници

Freeman, А. (2024). Pro ASP.NET Core 6. Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. София. APress

Microsoft Learn Challenge. Първи стъпки с ASP.NET MVC. Достъпно към 09.12.2024г. от <https://learn.microsoft.com/bg-bg/aspnet/mvc/overview/getting-started/introduction/>

Microsoft Access SQL: основни понятия, речник и синтакс Достъпно към 05.12.2024г от: <https://shorturl.at/Ur7u2>

# Приложение 1 - GameController

public class GameController : Controller

{

private readonly ApplicationDbContext \_context;

public GameController(ApplicationDbContext context)

{

\_context = context;

}

// GET: Game

public IActionResult Index(string searchQuery)

{

var games = \_context.Games.AsQueryable();

if (!string.IsNullOrEmpty(searchQuery))

{

games = games.Where(g =>

g.Name.Contains(searchQuery) ||

g.Manufacturer.Contains(searchQuery));

}

return View(games.ToList());

}

// GET: Game/Details/5

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

{

if (id == null)

{

return NotFound();

}

var game = await \_context.Games

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

if (game == null)

{

return NotFound();

}

return View(game);

}

// GET: Game/Create

public IActionResult Create()

{

return View();

}

// POST: Game/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Id,Name,Manufacturer,Description,Instructions,Rating,ImageUrl")] Game game)

{

if (ModelState.IsValid)

{

\_context.Add(game);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(game);

}

// GET: Game/Edit/5

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

{

if (id == null)

{

return NotFound();

}

var game = await \_context.Games.FindAsync(id);

if (game == null)

{

return NotFound();

}

return View(game);

}

// POST: Game/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("Id,Name,Manufacturer,Description,Instructions,Rating,ImageUrl")] Game game)

{

if (id != game.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(game);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!GameExists(game.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(game);

}

// GET: Game/Delete/5

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

{

if (id == null)

{

return NotFound();

}

var game = await \_context.Games

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

if (game == null)

{

return NotFound();

}

return View(game);

}

// POST: Game/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var game = await \_context.Games.FindAsync(id);

if (game != null)

{

\_context.Games.Remove(game);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

//Searching loggic

[HttpGet]

public IActionResult Search(string searchQuery)

{

var games = \_context.Games.AsQueryable();

if (!string.IsNullOrEmpty(searchQuery))

{

games = games.Where(g =>

g.Name.Contains(searchQuery) ||

g.Manufacturer.Contains(searchQuery));

}

return PartialView("\_GameList", games.ToList());

}

private bool GameExists(int id)

{

return \_context.Games.Any(e => e.Id == id);

}

}

}

# Приложение 2

.Кодът за Game Container може лесно да се адаптира за приложение в уебсайт за онлайн магазин. В този случай всяка кутия може да представя продукт, съдържащ изображение, име, кратко описание, цена и бутон за добавяне в количката. Това позволява на потребителите лесно да преглеждат продукти и да взаимодействат с тях чрез модерен и удобен интерфейс.

HTML

<div class="game-container">

@foreach (var game in Model)

{

<div class="game-box">

<img src="@game.ImageUrl" alt="@game.Name" />

<h3>@game.Name</h3>

<p>@game.Manufacturer</p>

<div class="button-container">

<a asp-action="Details" asp-route-id="@game.Id" class="btn btn-primary">Details</a>

<a asp-action="Edit" asp-route-id="@game.Id" class="btn btn-secondary">Edit</a>

</div>

</div>

}

</div>

CSS

. .game-container {

display: flex;

flex-wrap: wrap;

justify-content: center;

margin-top: 20px;

}

.game-box {

background: rgba(255, 255, 255, 0.1);

border-radius: 10px;

padding: 20px;

margin: 15px;

width: 250px;

text-align: center;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

transition: transform 0.3s, box-shadow 0.3s;

}

.game-box:hover {

transform: scale(1.05);

box-shadow: 0 6px 12px rgba(0, 0, 0, 0.3);

}

.game-box img {

width: 100%;

border-radius: 5px;

margin-bottom: 15px;

}

.game-box h3 {

font-size: 1.2em;

margin-bottom: 10px;

color: white;

}

.game-box p {

color: #b0b0b0;

font-size: 1em;

margin-bottom: 15px;

}

.button-container a {

text-decoration: none;

padding: 8px 12px;

border-radius: 5px;

color: white;

margin: 0 5px;

transition: background-color 0.3s;

}

.button-container a.btn-primary {

background-color: #8a2be2;

}

.button-container a.btn-primary:hover {

background-color: #6a1bb8;

}

.button-container a.btn-secondary {

background-color: #5a5a5a;

}

.button-container a.btn-danger {

background-color: #d9534f;

}

# Приложение 3

Добавяме бутон за създаване на нови игри долу вдясно. Представяме го подобаващо за да бъде лесно разбираем за потребителя.

HTML

<button class="add-game-button">

<**a** **asp-action**="Create">+</**a**>

</button>

CSS

.add-game-button {

position: fixed;

bottom: 30px; /\* Adjust distance from the bottom of the screen \*/

right: 30px; /\* Adjust distance from the right of the screen \*/

width: 70px; /\* Size of the button \*/

height: 70px; /\* Size of the button \*/

background-color: #1D073C; /\* Background color \*/

border: none; /\* Remove border \*/

border-radius: 50%; /\* Make it a circle \*/

display: flex;

align-items: center;

justify-content: center;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1); /\* Optional: Add shadow for depth \*/

transition: all 0.3s ease

}

.add-game-button a {

font-size: 40px; /\* Make the "+" symbol bigger \*/

color: #A64EFB; /\* Set the text color to white \*/

text-decoration: none; /\* Remove underline from the link \*/

transition: all 0.3s ease; /\* Smooth transition for text color change \*/

}

/\* Hover effect \*/

.add-game-button:hover {

background-color: #A64EFB; /\* Change background to white on hover \*/

box-shadow: 0 6px 8px rgba(0, 0, 0, 0.2); /\* Slightly larger shadow on hover \*/

}

.add-game-button:hover a {

color: #1D073C; /\* Change text color to the button's original color on hover \*/

}

# Приложение 4

За да добавим дадена игра първо ни трябва някакви характеристики за нея (име, снимка, описание, райтинг и др.). За това ипозлваме form, с който взимаме информацията подадена от потребителя.

HTML

<div class="row">

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

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

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

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

<input type="submit" value="Create" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

CSS

.form-control {

border-radius: 10px; /\* Add border radius to the input fields \*/

background: transparent;

width:400px;

padding: 10px; /\* Optional: Add padding inside the input fields \*/

border: 3px solid #1D073C; /\* Optional: Adjust the border color \*/

color: #A64EFB;

}

.form-group {

margin-bottom: 0.5rem; /\* Optional: Adjust space between form fields \*/

}

.form-control:focus {

background: transparent;

color: #A64EFB;

border: 1px solid #1D073C;

}

.btn-primary {

background: #1D073C;

width:400px;

border: 2px solid #A64EFB;

border-radius: 15px;

color: #A64EFB;

transition: all 0.3s ease; /\* Smooth transition for color changes \*/

font-size: 20px;

margin: 20px 0;

}

.btn-primary:hover {

background: #A64EFB;

border: solid 2px #A64EFB;

color: #1D073C;

}

# Приложение 5

Ако потребителя иска да направи промяна по дадена игра, той може да използва edit бутона, с който лесно може да редактира характеристиките, както и да изтрие играта.

HTML

<div class="row">

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

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

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

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

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" />

</div>

</**form**>

<!-- Delete Button Form -->

<**form** **asp-action**="Delete" **asp-route-id**="@Model.Id" method="post" onsubmit="return confirm('Are you sure you want to delete this game?');">

<div class="form-group">

<input type="submit" value="Delete" class="btn btn-danger" />

</div>

</**form**>

</div>

</div>

CSS

.form-control {

border-radius: 10px; /\* Add border radius to the input fields \*/

background: transparent;

width:400px;

padding: 10px; /\* Optional: Add padding inside the input fields \*/

border: 3px solid #1D073C; /\* Optional: Adjust the border color \*/

color: #A64EFB;

}

.form-group {

margin-bottom: 0.5rem; /\* Optional: Adjust space between form fields \*/

}

.form-control:focus {

background: transparent;

color: #A64EFB;

border: 1px solid #1D073C;

}

.btn-primary {

background: #1D073C;

width:400px;

border: 2px solid #A64EFB;

border-radius: 15px;

color: #A64EFB;

transition: all 0.3s ease; /\* Smooth transition for color changes \*/

font-size: 20px;

margin: 20px 0;

}

.btn-primary:hover {

background: #A64EFB;

border: solid 2px #A64EFB;

color: #1D073C;

}

.back-list a {

font-size: 20px;

color: #A64EFB;

text-decoration: none;

transition: all 0.3s ease; /\* Smooth transition for color changes \*/

}

.back-list a:hover {

color: #1D073C;

}

.btn-danger {

background: #1D073C;

width: 400px;

border: 2px solid #A64EFB;

border-radius: 15px;

color: #A64EFB;

transition: all 0.3s ease; /\* Smooth transition for color changes \*/

font-size: 15px;

}

.buttons{

display:flex;

justify-content:space-evenly;

}

.buttons button{

width:50%;

padding:12px 22px;

}

# Приложение 6

Ако потребителя иска да види пълните характеристики на играта, той може да използва details бутона, който предоставя информацията под формата на таблица.

HTML

<div>

<h4>Game</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Name)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Manufacturer)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Manufacturer)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Description)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Description)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Instructions)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Instructions)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Rating)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Rating)

</dd>

</dl>

</div>

CSS

dl.row {

background-color: mediumpurple;

padding: 20px;

border-radius: 8px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

}

/\* Label and definition styling \*/

dt {

font-weight: bold;

font-size: 1.3rem;

color: #1D073C;

border-right: 3px solid #1D073C;

border-bottom: 3px solid #1D073C;

border-radius:10px;

}

dd {

color: white;

font-weight:bold;

margin-bottom: 10px;

font-size: 1.2rem;

}