Dokumentacja projektowa

Programowanie obiektowe i graficzne

Uproszczony sumulator piłkarski w C#.

Kierunek: Informatyka

Członkowie zespołu:
Jakub Licznar

Gliwice, 2024/2025

Spis treści

1 Wprowadzenie

1.1 Cel projektu

Celem projektu jest stworzenie aplikacji - prostego symulatora piłkarskiego. Wykorzystując C# jako jezyk obiektowy, dodając do tego graficzne GUI, w tym celu wykorzystano wzorzec MVC.

2 Założenia projektowe

2.1 Założenia techniczne i nietechniczne

1. Backend programowany w C#.
2. Frontend z wykorzystaniem wzroców MVC(HTML, CSS i JS).
3. Istniejąca baza daych do przechowywania obiektów klasy model.

2.2 Stos technologiczny

1. Microsoft Visual Studio 2022.
2. SQL Server Managment Studio 20.
3. MVC (Model-View-Controller).
4. FluentValidation
5. FluentValidation.AspNetCore
6. Microsoft.AspNet.WebPages
7. Microsoft.AspNetCore.Identity.EntityFramework.Core
8. Microsoft.EntityFrameworkCore
9. Microsoft.EntityFrameworkCore.SqlServer
10. Microsoft.VisualStrudnio.Web.CodeGeneration.Design
11. Microsoft Edge.

3 Struktura kodu

```
MyFootballGame/
 Program.cs
 appsettings.json
 Controllers
     HomeController.cs
     LeagueController.cs
     MatchController.cs
     PlayerController.cs
     SeasonController.cs
     TeamController.cs
 Models
     ErrorViewModel.cs
 Other
     Application
         Interfaces
             ILeagueService.cs
             IMatchService.cs
             IPlayerService.cs
             ISeasonService.cs
             ITeamService.cs
         Services
             LeagueService.cs
             MatchService.cs
             PlayerService.cs
             SeasonService.cs
             TeamService.cs
         ViewModel
             League
                 LeagueForListVm.cs
                 ListLeagueForListVm.cs
                 NewLeagueVm.cs
             Match
                 ListMatchForListVm.cs
                 MatchForListVm.cs
                 PlayAllMatchesOfSeasonVm.cs
             Player
                 ListPlayerForListVm.cs
                 PlayerForListVm.cs
             Season
```

```
                ListSeasonForListVm.cs
                NewSeasonVm.cs
                SeasonForListVm.cs
            Team
                ListTeamForListVm.cs
                TeamForListVm.cs
        DependencyInjection.cs
    Domain
        Interfaces
            ILeagueRepository.cs
            IMatchRepository.cs
            IPlayerRepository.cs
            ISeasonRepository.cs
            ITeamRepository.cs
        Model
            Common.cs
            Country.cs
            League.cs
            Match.cs
            Player.cs
            Season.cs
            Team.cs
    Infrastructure
        Repositories
            LeagueRepository.cs
            MatchRepository.cs
            PlayerRepository.cs
            SeasonRepository.cs
            TeamRepository.cs
        Context.cs
    Migrations
        20250402174651_InitialCreate.cs
        ContextModelSnapshot.cs
Views/
    Home
        Index.cshtml
        Privacy.cshtml
    League
        AddNewLeague.cshtml
        Index.cshtml
    Match
```

```
    Index.cshtml
    PlayWholeSeason.cshtml
Player
    Index.cshtml
Season
    AddNewSeason.cshtml
    Index.cshtml
Shared
    _Layout.cshtml
    _ValidationScriptsPartial.cshtml
    Error.cshtml
Team
    Index.cshtml
_ViewImports.cshtml
_ViewStart.cshtml
```

4 Kod poszczególnych elementów

4.1 Program.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.AspNetCore.Identity;
using MyFootballGame.Other.Application;
using MyFootballGame.Other.Application.ViewModel.League;
using MyFootballGame.Other.Domain.Interfaces;
using MyFootballGame.Other.Infrastructure;
using MyFootballGame.Other.Infrastructure.Repositories;
using FluentValidation;
using FluentValidation.AspNetCore;
using System.Web.WebPages;
using MyFootballGame.Other.Application.ViewModel.Season;
using MyFootballGame.Other.Application.ViewModel.Match;


var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews().
   AddFluentValidation();




builder.Services.AddTransient<IValidator<NewLeagueVm>,
   NewLeagueValidation>();
builder.Services.AddTransient<IValidator<NewSeasonVm>,
   NewSeasonValidation>();
builder.Services.AddTransient<IValidator<
   PlayAllMatchesOfSeasonVm>, PlayAllMatchesValidation
   >();

builder.Services.AddDbContext<Context>(options =>
    options.UseSqlServer("Server=.\\SQLEXPRESS;Database=
       FootballSim;Trusted_Connection=True;
       TrustServerCertificate=True;"));
builder.Services.AddApplication();

var app = builder.Build();

// Configure the HTTP request pipeline.
```

```
33 if (!app.Environment.IsDevelopment())
34 {
35     app.UseExceptionHandler("/Home/Error");
36     app.UseHsts();
37 }
38
39 app.UseHttpsRedirection();
40 app.UseStaticFiles();
41
42 app.UseRouting();
43
44 app.UseAuthorization();
45
46 app.MapControllerRoute(
47     name: "default",
48     pattern: "{controller=Home}/{action=Index}/{id?}");
49
50 app.Run();
```

4.2 Controller

```
1  using Microsoft.AspNetCore.Mvc;
2  using MyFootballGame.Other.Application.Interfaces;
3  using MyFootballGame.Other.Application.Services;
4  using MyFootballGame.Other.Application.ViewModel.Match;
5  using System.CodeDom;
6
7  namespace MyFootballGame.Controllers
8  {
9      public class MatchController : Controller
10     {
11         private readonly IMatchService _matchService;
12         public MatchController(IMatchService
               matchService)
13         {
14             _matchService = matchService;
15         }
16         [HttpGet]
17         public IActionResult Index()
18         {
19             var model = _matchService.GetAllMatches(10,
                   1, "");
```

```
20              return View(model);
21          }
22          [HttpPost]
23          public IActionResult Index(int pageSize, int
                pageNum, string searchString)
24          {
25              if (pageNum == null)
26              {
27                  pageNum = 1;
28              }
29              if (searchString == null)
30              {
31                  searchString = String.Empty;
32              }
33              var model = _matchService.GetAllMatches(
                    pageSize, pageNum, searchString);
34              return View(model);
35          }
36          [HttpGet]
37          public IActionResult PlayWholeSeason()
38          {
39              return View();
40          }
41          [HttpPost]
42          public IActionResult PlayWholeSeason(
                PlayAllMatchesOfSeasonVm model)
43          {
44              var id = _matchService.PlayWholeSeason(model
                    );
45              return RedirectToAction("Index");
46          }
47      }
48  }
```

4.3 Repository Interface

```
1   using MyFootballGame.Other.Domain.Model;
2
3   namespace MyFootballGame.Other.Domain.Interfaces
4   {
5       public interface IMatchRepository
6       {
```

```
7           //Play one match
8           int AddMatch(int seasonId, int hostId, int
               guestId);
9
10          //Play all matches of a season
11          void PlayWholeSeasonByLeagueAndSeasonId(int
               leaguesId, int seasonId);
12
13          //Update match
14          void UpdateMatch(Match match);
15
16          //Delete match(make inactive)
17          void DeleteMatch(int matchId);
18
19          //Get match by id
20          Match GetMatchById(int id);
21          //Get all matches
22          IQueryable<Match> GetAllMatches();
23
24          //Get all matches of a season
25          IQueryable<Match> GetMatchesBySeason(int
               seasonId);
26
27          //Display match by id
28          void DisplayMatchById(int matchId);
29
30          //Clear statistics by league id
31          public void ClearStatisticsByLeagueId(int
               leagueId);
32
33      }
34  }
```

4.4 Repository

```
1  using Microsoft.EntityFrameworkCore;
2  using MyFootballGame.Other.Domain.Interfaces;
3  using MyFootballGame.Other.Domain.Model;
4  using MyFootballGame.Other.Infrastructure;
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
```

```
8  using System.Text;
9  using System.Threading.Tasks;
10
11 namespace MyFootballGame.Other.Infrastructure.
       Repositories
12 {
13     public class MatchRepository : IMatchRepository
14     {
15         private readonly Context _context;
16         public MatchRepository(Context context)
17         {
18             _context = context;
19         }
20
21         public int AddMatch(int seasonId, int hostTeamId
               , int guestTeamId)
22         {
23             var hostTeam = _context.Teams.Find(
                   hostTeamId);
24             var guestTeam = _context.Teams.Find(
                   guestTeamId);
25
26             Random rand = new Random();
27             int hostScore = 0;
28             int guestScore = 0;
29
30             const int homeAdvantage = 3; // Przewaga
                   gospodarzy
31
32             int adjustedHostSkill = hostTeam.TeamSkill +
                    homeAdvantage;
33             int adjustedGuestSkill = guestTeam.TeamSkill
                   ;
34
35             int advantage = adjustedHostSkill -
                   adjustedGuestSkill;
36
37             for (int minute = 0; minute < 90; minute++)
38             {
39                 if (rand.NextDouble() < 0.03 + advantage
                        / 100) // Szansa na gol gospodarzy
40                 {
41                     hostScore++;
```

```
42                  }
43                  if (rand.NextDouble() < 0.03 - advantage
                        / 100) // Szansa na gol gosci
44                  {
45                      guestScore++;
46                  }
47              }
48
49              // Aktualizacja statystyk zespolow
50              if (hostScore > guestScore)
51              {
52                  hostTeam.Wins++;
53                  guestTeam.Losses++;
54              }
55              else if (hostScore < guestScore)
56              {
57                  hostTeam.Losses++;
58                  guestTeam.Wins++;
59              }
60              else
61              {
62                  hostTeam.Draws++;
63                  guestTeam.Draws++;
64              }
65
66              hostTeam.Points = hostTeam.Wins * 3 +
                    hostTeam.Draws;
67              guestTeam.Points = guestTeam.Wins * 3 +
                    guestTeam.Draws;
68
69              var newMatch = new Match
70              {
71                  SeasonId = seasonId,
72                  HostTeamId = hostTeamId,
73                  GuestTeamId = guestTeamId,
74                  HostScore = hostScore,
75                  GuestScore = guestScore,
76                  HostTeam = hostTeam,
77                  GuestTeam = guestTeam,
78                  CreatedTime = DateTime.Now,
79                  Status = CommonStatusEnum.Active
80              };
81
```

```
82              _context.Matches.Add(newMatch);
83              _context.SaveChanges();
84
85              return newMatch.Id;
86
87          }
88
89          public void DeleteMatch(int matchId)
90          {
91              _context.Matches.Where(m => m.Id == matchId)
                    .FirstOrDefault().Status =
                    CommonStatusEnum.Inactive;
92          }
93
94          public IQueryable<Match> GetAllMatches()
95          {
96              var matches = _context.Matches
97                  .Include(m => m.HostTeam)
98                  .Include(m => m.GuestTeam)
99                  .Where(m => m.Status == CommonStatusEnum
                        .Active);
100             return matches;
101         }
102
103         public Match GetMatchById(int id)
104         {
105             var match = _context.Matches.Find(id);
106             return match;
107         }
108
109         public IQueryable<Match> GetMatchesBySeason(int
                seasonId)
110         {
111
112             var matches = _context.Matches.Where(m => m.
                    SeasonId == seasonId);
113             return matches;
114         }
115
116         public void PlayWholeSeasonByLeagueAndSeasonId(
                int leagueId, int seasonId)
117         {
```

```
118             var teams = _context.Teams.Where(t => t.
                    LeagueId == leagueId).ToList();
119             foreach (var teamOne in teams)
120             {
121                 foreach (var teamTwo in teams)
122                 {
123                     if (teamOne != teamTwo)
124                     {
125                         var newmatch = AddMatch(seasonId
                                , teamOne.Id, teamTwo.Id);
126                         DisplayMatchById(newmatch);
127                     }
128                 }
129             }
130         }
131 
132         public void UpdateMatch(Match match)
133         {
134             throw new NotImplementedException();
135         }
136 
137         public void DisplayMatchById(int matchId)
138         {
139             var match = GetMatchById(matchId);
140             Console.WriteLine($"{match.HostTeam.Name} {
                    match.HostScore} - {match.GuestScore} {
                    match.GuestTeam.Name}");
141         }
142         public void ClearStatisticsByLeagueId(int
                leagueId)
143         {
144             var teams = _context.Teams.Where(t => t.
                    LeagueId == leagueId).ToList();
145             foreach (var team in teams)
146             {
147                 team.Wins = 0;
148                 team.Draws = 0;
149                 team.Losses = 0;
150                 team.Points = 0;
151                 team.ModifyTime = DateTime.Now;
152             }
153             _context.SaveChanges();
154         }
```

```
155     }
156 }
```

4.5 Service Interface

```
1  using MyFootballGame.Other.Application.ViewModel.League;
2  using MyFootballGame.Other.Application.ViewModel.Match;
3
4  namespace MyFootballGame.Other.Application.Interfaces
5  {
6      public interface IMatchService
7      {
8          ListMatchForListVm GetAllMatches(int pageSize,
               int pageNum, string searchString);
9          int PlayWholeSeason(PlayAllMatchesOfSeasonVm
               allMatchesVm);
10
11     }
12 }
```

4.6 Service

```
1  using MyFootballGame.Other.Application.Interfaces;
2  using MyFootballGame.Other.Application.ViewModel.Match;
3  using MyFootballGame.Other.Application.ViewModel.Player;
4  using MyFootballGame.Other.Domain.Interfaces;
5  using MyFootballGame.Other.Infrastructure.Repositories;
6
7  namespace MyFootballGame.Other.Application.Services
8  {
9      public class MatchService : IMatchService
10     {
11         private readonly IMatchRepository
               _matchRepository;
12         private readonly ISeasonRepository
               _seasonRepository;
13         public MatchService(IMatchRepository
               matchRepository, ISeasonRepository
               seasonRepository)
14         {
15             _matchRepository = matchRepository;
```

```
16              _seasonRepository = seasonRepository;
17          }
18
19          public ListMatchForListVm GetAllMatches(int
                pageSize, int pageNum, string searchString)
20          {
21              var matches = _matchRepository.GetAllMatches
                    ().Where(m => m.HostTeam.Name.Contains(
                    searchString));
22              if (pageNum < 1)
23              {
24                  pageNum = 1;
25              }
26              var matchesToShow = matches.Skip(pageSize *
                    (pageNum - 1)).Take(pageSize).ToList();
27              ListMatchForListVm result = new
                    ListMatchForListVm()
28              {
29                  PageSize = pageSize,
30                  CurrentPage = pageNum,
31                  SearchString = searchString,
32                  Matches = new List<MatchForListVm>(),
33                  Count = matches.Count()
34              };
35              foreach (var match in matchesToShow)
36              {
37                  var matchVm = new MatchForListVm()
38                  {
39                      Id = match.Id,
40                      HostTeamName = match.HostTeam.Name,
41                      HostTeamScore = match.HostScore,
42                      GuestTeamName = match.GuestTeam.Name
                            ,
43                      GuestTeamScore = match.GuestScore,
44
45                  };
46                  result.Matches.Add(matchVm);
47              }
48              return result;
49          }
50
51          public int PlayWholeSeason(
                PlayAllMatchesOfSeasonVm allMatchesVm)
```

```
52          {
53              _matchRepository.ClearStatisticsByLeagueId(
                    allMatchesVm.Id);
54              var activeSeason = _seasonRepository.
                    GetActiveSeasonByLeagueId(allMatchesVm.Id
                    );
55              _matchRepository.
                    PlayWholeSeasonByLeagueAndSeasonId(
                    allMatchesVm.Id, activeSeason.Id);
56              _seasonRepository.ChooseSeasonWinner(
                    allMatchesVm.Id);
57              _seasonRepository.GenerateNewSeason(
                    allMatchesVm.Id);
58
59              return allMatchesVm.Id;
60          }
61      }
62  }
```

4.7 Viewmodel

```
1   namespace MyFootballGame.Other.Application.ViewModel.
        Match
2   {
3       public class MatchForListVm
4       {
5           public int Id { get; set; }
6           public string HostTeamName { get; set; }
7           public int HostTeamScore { get; set; }
8           public int GuestTeamScore   { get; set; }
9           public string GuestTeamName { get; set; }
10
11      }
12  }
```

```
1   using MyFootballGame.Other.Application.ViewModel.Player;
2
3   namespace MyFootballGame.Other.Application.ViewModel.
        Match
4   {
5       public class ListMatchForListVm
6       {
```

```
7           public List<MatchForListVm> Matches { get; set;
                }
8           public int PageSize { get; set; }
9           public int CurrentPage { get; set; }
10          public string SearchString { get; set; }
11          public int Count { get; set; }
12      }
13  }
```

4.8 DependencyInjection.cs

```
1   using MyFootballGame.Other.Application.Interfaces;
2   using MyFootballGame.Other.Application.Services;
3   using MyFootballGame.Other.Domain.Interfaces;
4   using MyFootballGame.Other.Infrastructure.Repositories;
5
6   namespace MyFootballGame.Other.Application
7   {
8       public static class DependencyInjection
9       {
10          public static IServiceCollection AddApplication(
                this IServiceCollection services)
11          {
12              services.AddTransient<ILeagueService,
                    LeagueService>();
13              services.AddTransient<ISeasonService,
                    SeasonService>();
14              services.AddTransient<ITeamService,
                    TeamService>();
15              services.AddTransient<IPlayerService,
                    PlayerService>();
16              services.AddTransient<ILeagueRepository,
                    LeagueRepository>();
17              services.AddTransient<ISeasonRepository,
                    SeasonRepository>();
18              services.AddTransient<ITeamRepository,
                    TeamRepository>();
19              services.AddTransient<IPlayerRepository,
                    PlayerRepository>();
20              services.AddTransient<IMatchService,
                    MatchService>();
```

```
21              services.AddTransient<IMatchRepository,
                    MatchRepository>();
22              return services;
23          }
24      }
25  }
```

4.9 Model

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace MyFootballGame.Other.Domain.Model
8   {
9       public class Common
10      {
11          public int Id { get; set; }
12          public DateTime CreatedTime { get; set; }
13          public DateTime ModifyTime { get; set; }
14          public CommonStatusEnum Status { get; set; }
15
16      }
17      public enum CommonStatusEnum
18      {
19          Inactive,
20          Active,
21      }
22  }
```

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace MyFootballGame.Other.Domain.Model
8   {
9       public class Match : Common
10      {
```

```
        public int SeasonId { get; set; }
        public int HostTeamId { get; set; }
        public int GuestTeamId { get; set; }

        public int HostScore { get; set; }
        public int GuestScore { get; set; }

        public Season Season { get; set; }
        public Team HostTeam { get; set; }
        public Team GuestTeam { get; set; }
    }
}
```

4.10 Context

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using MyFootballGame.Other.Domain.Model;

namespace MyFootballGame.Other.Infrastructure
{
    public class Context : IdentityDbContext
    {
        public Context(DbContextOptions<Context> options
            ) : base(options)
        {
        }
        public DbSet<League> Leagues { get; set; }
        public DbSet<Season> Seasons { get; set; }
        public DbSet<Match> Matches { get; set; }
        public DbSet<Team> Teams { get; set; }
        public DbSet<Country> Countries { get; set; }
        public DbSet<Player> Players { get; set; }

        protected override void OnConfiguring(
            DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("Server=.\\
                SQLEXPRESS;Database=FootballSim;
                Trusted_Connection=True;
                TrustServerCertificate=True;");
```

```
23          }
24
25          protected override void OnModelCreating(
                ModelBuilder modelBuilder)
26          {
27              base.OnModelCreating(modelBuilder);
28
29              // Match Relations
30              modelBuilder.Entity<Match>()
31                  .HasOne(m => m.HostTeam)
32                  .WithMany(t => t.HomeMatches)
33                  .HasForeignKey(m => m.HostTeamId)
34                  .OnDelete(DeleteBehavior.Restrict);
35
36              modelBuilder.Entity<Match>()
37                  .HasOne(m => m.GuestTeam)
38                  .WithMany(t => t.AwayMatches)
39                  .HasForeignKey(m => m.GuestTeamId)
40                  .OnDelete(DeleteBehavior.Restrict);
41
42              // SeasonWinner relation
43              modelBuilder.Entity<Season>()
44                  .HasOne(s => s.SeasonWinner)
45                  .WithMany()
46                  .HasForeignKey(t => t.SeasonWinnerId)
47                  .OnDelete(DeleteBehavior.Restrict);
48
49              // Configure primary key for
                    IdentityUserLogin<string>
50              modelBuilder.Entity<IdentityUserLogin<string
                    >>()
51                  .HasKey(l => new { l.LoginProvider, l.
                        ProviderKey });
52          }
53      }
54  }
```

4.11 View

```
1  @model MyFootballGame.Other.Application.ViewModel.Match.
       ListMatchForListVm;
2
```

```
@{
    ViewData["Title"] = "Index";
}

<h1>AllMatches</h1>

<p>
    <a asp-action="PlayWholeSeason">Play whole season</a> <!DO IMPLEMENTACJI!!!!!>
</p>
<form asp-action="Index" asp-controller="Match" method="post">
    <div class="row">
        <input type="text" asp-for="SearchString" name="searchString" id="searchString" />
        <input type="submit" value="Search: " style="background-color: white; color: blue" />
    </div>

    <div class="row">
        <table class="table">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        HostTeamName
                    </th>
                    <th>
                        HostTeamScore
                    </th>
                    <th>
                        GuestTeamScore
                    </th>
                    <th>
                        GuestTeamName Kocham Laure
                    </th>
                    <th>
                        Operations
                    </th>
                    <th></th>
                </tr>
```

```
42              </thead>
43              <tbody>
44                  @foreach (var item in Model.Matches)
45                  {
46                      <tr>
47                          <td>
48                              @Html.DisplayFor(modelItem
                                    => item.Id)
49                          </td>
50                          <td>
51                              @Html.DisplayFor(modelItem
                                    => item.HostTeamName)
52                          </td>
53                          <td>
54                              @Html.DisplayFor(modelItem
                                    => item.HostTeamScore)
55                          </td>
56                          <td>
57                              @Html.DisplayFor(modelItem
                                    => item.GuestTeamScore)
58                          </td>
59                          <td>
60                              @Html.DisplayFor(modelItem
                                    => item.GuestTeamName)
61                          </td>
62                          <td>
63                              @Html.ActionLink("Edit", "
                                    Edit", new { id = item.Id
                                     }) |
64                              @Html.ActionLink("Details",
                                    "Details", new { id =
                                    item.Id }) |
65                              @Html.ActionLink("Delete", "
                                    Delete", new { id = item.
                                    Id })
66                          </td>
67                      </tr>
68                  }
69              </tbody>
70          </table>
71      </div>
72      <div class="row">
73          <table>
```

```
74              <tr>
75                  @for (int i = 1; i <= Math.Ceiling(Model
                        .Count / (double)Model.PageSize); i
                        ++)
76                  {
77                      <td>
78                          @if (i != Model.CurrentPage)
79                          {
80                              <a href="javascript:
                                    PagerClick(@i)">@i</a>
81                          }
82                          else
83                          {
84                              <span>@i</span>
85                          }
86                      </td>
87                  }
88              </tr>
89          </table>
90          <input type="hidden" name="pageNum" id="pageNum"
                />
91          <input type="hidden" name="pageSize" id="
                pageSize" value="10" />
92      </div>
93  </form>
94
95  @section Scripts
96  {
97      <script type="text/javascript">
98          function PagerClick(index)
99          {
100             document.getElementById("pageNum").value =
                    index;
101             document.forms[0].submit();
102         }
103     </script>
104 }
```

```
1   @model MyFootballGame.Other.Application.ViewModel.Match.
        PlayAllMatchesOfSeasonVm
2
3   @{
4       ViewData["Title"] = "PlayWholeSeason";
```

```
}

<h1>PlayWholeSeason</h1>

<h4>PlayAllMatchesOfSeasonVm</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="PlayWholeSeason">
            <div asp-validation-summary="ModelOnly"
                class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Id" class="control-label
                    "></label>
                <input asp-for="Id" class="form-control"
                     />
                <span asp-validation-for="Id" class="
                    text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create"
                    class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("
        _ValidationScriptsPartial");}
}
```

5 Wnioski

Powyżej zzaprezentowano strukturę oraz kod źródłowy poszczególnych elementów projektu. Aplikacja nie posiada zbyt dużej ilości funkcjonalności, co pozwala na jej łatwy dalszy rozwój np. zwiększenie ilości lig, dodanie systemu starzenia się zawodników czy transferów. Użycie wzroca MVC znacząco ułatwia proces rozwijania aplikacji. Takie rozwiązanie jest bardzo dobre dla początkujących programistów, gdyż podczas pracy z nim jesteśmy zmuszeni do poznania podstaw nie tylko języka C#, ale także np. CSs i HTML.