



Politechnika
Wrocławska

ZASTOSOWANIE BAZ DANYCH

Raport Projektu

Baza danych zawodników, drużyn, meczy NBA

Wykonał: Hubert Kowalczyk
Prowadzący: Dr. inż. Konrad Kluwak

Wrocław, 03.12.2023

1 Opis Projektu

Celem Projektu jest napisanie aplikacji webowej która, wyświetla drużyny, zawodników, mecze NBA. Do napisania aplikacji utworzono bazę danych MySQL oraz dodano do niej dane. Sama aplikacja webowa została napisana w języku C# w narzędziu EntityFramework 6 z szablonu MVC.

2 Baza Danych

Bazę danych została utworzona w msSQL. Składa się ona z czterech tabel:

- Teams - Tabela przechowująca Informacje o drużynie NBA jak:
 - Id drużyny - Klucz główny.
 - Nazwa - Nazwa Drużyny NBA.
 - City - Miasto, z którego drużyna się wywodzi.
 - Coach - Trener drużyny.
- Plyers - Table przechowująca podstawowe informacje o zawodniku NBA takie jak:
 - PlayerId - Klucz główny.
 - TeamID - Id drużyny, dla której zawodnik gra.
 - Name - Imie oraz nazwisko zawodnika.
 - Position - Pozycja zawodnika.
 - Height - Wzrost Zawodnika.
 - Weight - Waga zawodnika.
 - BirthDate - Data urodzenia zawodnika.
 - Nationality - Narodowość zawodnika.
- Matches - Przechowuje informacje o meczach ligi NBA takich jak:
 - MatchId - Klucz główny zawodnika.
 - Date - Data odbycia się meczu.
 - Location - Nazwa hali, na której odbywa się mecz.
 - HomeTeamId - Id drużyny, która jest gospodarzem.
 - AwayTeamId - Id drużyny, która jest gościem.
- PlayersInTeams - Przyporządkowuje zawodników do drużyn NBA:
 - PlayerName - Nazwa zawodnika.
 - Position - Pozycja zawodnika.
 - Height - Wzrost zawodnika.
 - Weight - Waga zawodnika.
 - TeamName - Nazwa Drużyny, dla której zawodnik gra.
 - City - Nazwa miasto z którego drużyna się wywodzi.

Do utworzenia tabel użyto poniższego kody msSQL

- Teams

```
1 CREATE TABLE Teams (  
2     TeamID INT PRIMARY KEY,  
3     Name NVARCHAR(100),  
4     City NVARCHAR(100),  
5     Coach NVARCHAR(100)  
6 );
```

- Players

```
1 CREATE TABLE Players (  
2     PlayerID INT PRIMARY KEY,  
3     TeamID INT,  
4     Name NVARCHAR(100),  
5     Position NVARCHAR(50),  
6     Height INT,  
7     Weight INT,  
8     BirthDate DATE,  
9     Nationality NVARCHAR(50),  
10    FOREIGN KEY (TeamID) REFERENCES Teams(TeamID)  
11 );
```

- Matches

```
1 CREATE TABLE Matches (  
2     MatchID INT PRIMARY KEY,  
3     Date DATE,  
4     Time TIME,  
5     Location NVARCHAR(100),  
6     HomeTeamID INT,  
7     AwayTeamID INT,  
8     FOREIGN KEY (HomeTeamID) REFERENCES Teams(TeamID),  
9     FOREIGN KEY (AwayTeamID) REFERENCES Teams(TeamID)  
10 );
```

- PlayersInTeams

```
1 SELECT  
2     p.Name AS PlayerName,  
3     p.Position,  
4     p.Height,  
5     p.Weight,  
6     t.Name AS TeamName,  
7     t.City  
8 INTO PlayersInTeams  
9 FROM  
10    Players p  
11 INNER JOIN  
12    Teams t ON p.TeamID = t.TeamID;
```

2.1 Dodanie danych

Dane uzyskano z strony internetowej: https://www.basketball-reference.com/leagues/NBA_2022_per_game.html. Strona ta pozwala na eksportowanie danych o meczach, zawodnikach w postaci pliku .csv. Drużyny wpisano ręcznie. Natomiast do dodania zawodników oraz meczy napisano dwa poniższe skrypty w języku python:

- Skrypt generujący zapytania insert do tabeli Players

```
1 def convert_height_to_cm(feet, inches):
2
3     return int(feet) * 30.48 + int(inches) * 2.54
4
5 def convert_weight_to_kg(pounds):
6
7     return int(pounds) * 0.453592
8
9 def convert_birth_date(date_string):
10
11     from datetime import datetime
12     return datetime.strptime(date_string, "%B %d %Y").strftime("%Y-%m-%d")
13
14 def generate_insert_queries(csv_data, team_id, start_player_id):
15
16     data_lines = csv_data.strip().split('\n')[1:]
17     queries = []
18     player_id = start_player_id
19
20     for line in data_lines:
21         player = line.split(',')
22         name = player[1].strip()
23         position = player[2].strip()
24         height_ft_in = player[3].split('-')
25         height_cm = round(convert_height_to_cm(height_ft_in[0], height_ft_in[1]))
26         weight_kg = round(convert_weight_to_kg(player[4]))
27         birth_date = convert_birth_date(player[5])
28         nationality = player[6].strip().upper()
29
30         query = f"INSERT INTO Players (PlayerID, TeamID, Name, Position, Height, Weight,
31             BirthDate, Nationality) VALUES ({player_id}, {team_id}, '{name}', '{position}', {
32             height_cm}, {weight_kg}, '{birth_date}', '{nationality}');"
33         queries.append(query)
34         player_id += 1
35
36     return queries
37
38 csv_data_example = """
39 No.,Player,Pos,Ht,Wt,Birth Date,,Exp,College
40 1,John Doe,SG,6-4,210,January 1 1990,us,5,Example College
41 """
42 team_id_example = 1
43 start_player_id_example = 100
44
45 insert_queries_example = generate_insert_queries(csv_data_example, team_id_example,
46     start_player_id_example)
47
48 for query in insert_queries_example:
49     print(query)
```

- Skrypt generujący zapytania insert do tabeli Matches

```

1
2
3 def generate_matches_insert_queries( location, home_team_id, teams, match_id):
4
5     matches_csv = """
6
7     """
8
9     from datetime import datetime
10
11
12     matches_data = [line.split(',') for line in matches_csv.strip().split('\n')[1:]]
13
14     insert_queries = []
15
16     for match in matches_data:
17
18         match_date = datetime.strptime(match[1].strip(), '%a %b %d %Y').strftime('%Y-%m-%d')
19         match_time = match[2].strip()
20
21
22         opponent_team = match[6].strip()
23         if '@' in match[5]:
24
25             home_team_id = home_team_id
26             away_team_id = teams.get(opponent_team, None)
27         else:
28
29             away_team_id = teams.get(opponent_team, None)
30
31         if home_team_id is None or away_team_id is None:
32             continue
33
34         match_time += "m";
35
36
37         query = f"INSERT INTO Matches (MatchID, Date, Time, Location, HomeTeamID, AwayTeamID)
38             VALUES ({match_id}, '{match_date}', '{match_time}', '{location}', {home_team_id
39             }, {away_team_id});"
40         insert_queries.append(query)
41         match_id += 1
42
43     return insert_queries
44
45 teams_dict = {
46     'Atlanta Hawks': 1, 'Boston Celtics': 2, 'Brooklyn Nets': 3, 'Charlotte Hornets': 4,
47     'Chicago Bulls': 5, 'Cleveland Cavaliers': 6, 'Dallas Mavericks': 7, 'Denver Nuggets': 8,
48     'Detroit Pistons': 9, 'Golden State Warriors': 10, 'Houston Rockets': 11, 'Indiana Pacers
49     ': 12,
50     'Los Angeles Clippers': 13, 'Los Angeles Lakers': 14, 'Memphis Grizzlies': 15, 'Miami
51     Heat': 16,
52     'Milwaukee Bucks': 17, 'Minnesota Timberwolves': 18, 'New Orleans Pelicans': 19, 'New
53     York Knicks': 20,
54     'Oklahoma City Thunder': 21, 'Orlando Magic': 22, 'Philadelphia 76ers': 23, 'Phoenix Suns
55     ': 24,
56     'Portland Trail Blazers': 25, 'Sacramento Kings': 26, 'San Antonio Spurs': 27, 'Toronto
57     Raptors': 28,
58     'Utah Jazz': 29, 'Washington Wizards': 30
59 }
60
61 insert_queries_matches = generate_matches_insert_queries( "Capital One Arena", 30, teams_dict
62 ,2321)
63
64 for query in insert_queries_matches:
65     print(query)

```

3 Aplikacja Webowa EF

Do napisania aplikacji został użyty EntityFramework. Wersja platformy .Net 6.0. Ponadto za pomocą menedżera pakietów NuGet dodano do projektu poniższe pakiety:

- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools

Aplikacja jest projektem MVC - Model-View-Controller (MVC) to wzorec projektowy powszechnie stosowany w tworzeniu aplikacji webowych, w tym tych wykorzystujących Entity Framework w .NET. MVC rozdziela aplikację na trzy główne komponenty:

- Model: Reprezentuje strukturę danych, logikę biznesową oraz reguły systemu. W przypadku wykorzystania Entity Framework, modele są zwykle klasami C#, które reprezentują tabele bazy danych. Każda klasa odpowiada tabeli, a jej właściwości odpowiadają kolumnom w tabeli. Entity Framework służy do mapowania tych klas na bazę danych (tzw. ORM - Object-Relational Mapping), co pozwala na wykonywanie operacji CRUD (Create, Read, Update, Delete) na bazie danych w sposób zorientowany obiektowo.
- View: To część, która odpowiada za prezentację danych użytkownikowi - to, co użytkownik widzi i z czym wchodzi w interakcję. Widoki w MVC są zwykle plikami Razor (z rozszerzeniem .cshtml), które dynamicznie generują HTML na podstawie danych przekazywanych z kontrolera. Widoki mogą używać modeli do wyświetlania danych w przeglądarce.
- Controller: Odpowiada za odbieranie żądań od użytkownika, wykonywanie odpowiednich operacji na modelach, a następnie przekazywanie danych do widoków. Kontrolery są klasami C#, które zawierają metody odpowiadające na różne żądania HTTP (np. GET, POST). W kontekście Entity Framework, kontrolery często wykorzystują instancje kontekstu bazy danych (zdefiniowane przez klasy dziedziczące po DbContext) do manipulowania danymi i odzwierciedlenia zmian w bazie danych.

3.1 Połączenie z bazą Danych

Pierwszym Krokiem do utworzenia bazy danych było napisanie Modelu, który byłby w stanie połączyć się z bazą danych. Do tego celu napisano klasę MVCDBcontext.cs. Kod tej klasy został przedstawiony poniżej.

```
1  using System;
2  using System.Collections.Generic;
3  using Microsoft.EntityFrameworkCore;
4  using Microsoft.EntityFrameworkCore.Metadata;
5  using static NBA_GAMES_SCHEDULE.Models.UserMessage;
6
7  namespace NBA_GAMES_SCHEDULE.Models
8  {
9      public partial class MVCDBcontext : DbContext
10     {
11         public MVCDBcontext()
12         {
13         }
14
15         public DbSet<UserMessage> UserMessages { get; set; }
16
17         public MVCDBcontext(DbContextOptions<MVCDBcontext> options)
18             : base(options)
19         {
20         }
21
22         public virtual DbSet<Match> Matches { get; set; } = null!;
23         public virtual DbSet<Player> Players { get; set; } = null!;
24         public virtual DbSet<Team> Teams { get; set; } = null!;
25
26         protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
27         {
28             if (!optionsBuilder.IsConfigured)
29             {
30                 optionsBuilder.UseSqlServer("Server=DESKTOP-HR5LKTM;Database=NBA_Player_Stats;
31                                     Trusted_connection=true");
32             }
33
34             protected override void OnModelCreating(ModelBuilder modelBuilder)
35             {
36
37                 modelBuilder.Entity<Team>(entity =>
```

```

38     {
39         entity.Property(e => e.TeamId)
40             .ValueGeneratedNever()
41             .HasColumnName("TeamID");
42
43         entity.Property(e => e.City).HasMaxLength(100);
44
45         entity.Property(e => e.Coach).HasMaxLength(100);
46
47         entity.Property(e => e.Name).HasMaxLength(100);
48
49     };
50
51 });
52
53
54 modelBuilder.Entity<Player>(entity =>
55 {
56     entity.Property(e => e.PlayerId)
57         .ValueGeneratedOnAdd()
58         .HasColumnName("PlayerID");
59
60     entity.Property(e => e.Name).HasMaxLength(100);
61
62     entity.HasOne(d => d.Team)
63         .WithMany(p => p.Players)
64         .HasForeignKey(d => d.TeamId)
65         .OnDelete(DeleteBehavior.ClientSetNull);
66
67 });
68
69
70
71 modelBuilder.Entity<Match>(entity =>
72 {
73
74
75     entity.Property(e => e.MatchId)
76         .ValueGeneratedOnAdd()
77         .HasColumnName("MatchID");
78
79     entity.Property(e => e.Date)
80         .HasColumnType("date")
81         .HasColumnName("Date");
82
83     entity.Property(e => e.Time)
84         .HasColumnType("time")
85         .HasColumnName("Time");
86
87     entity.Property(e => e.Location)
88         .HasMaxLength(100)
89         .HasColumnName("Location");
90
91     entity.Property(e => e.HomeTeamId)
92         .HasColumnName("HomeTeamId");
93
94     entity.Property(e => e.AwayTeamId)
95         .HasColumnName("AwayTeamId");
96
97     entity.HasOne(m => m.HomeTeam)
98         .WithMany(t => t.HomeMatches)
99         .HasForeignKey(m => m.HomeTeamId)
100        .OnDelete(DeleteBehavior.Restrict);
101
102     entity.HasOne(m => m.AwayTeam)
103         .WithMany(t => t.AwayMatches)
104         .HasForeignKey(m => m.AwayTeamId)
105         .OnDelete(DeleteBehavior.Restrict);
106 });
107
108
109
110
111
112
113
114
115
116

```

```
117
118
119
120
121
122
123         OnModelCreatingPartial(modelBuilder);
124
125     }
126
127     partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
128 }
129 }
```

Następnie do pliku appsettings.json dodano linijkę, która określa bazę danych, z którą nawiązane zostanie połączenie.

```
1 "ConnectionStrings": {
2   "MvcConnectionString": "Server=DESKTOP-HR5LKTM;Database=NBA_Player_Stats;Trusted_connection=true"
3 }
```

Oraz w pliku Program.cs (plik main aplikacji) dodany został kod, który łączy aplikację z bazą danych.

```
1 builder.Services.AddDbContext<MVCDbContext>(options =>
2     options.UseSqlServer(builder.Configuration.GetConnectionString("MvcConnectionString"));
```

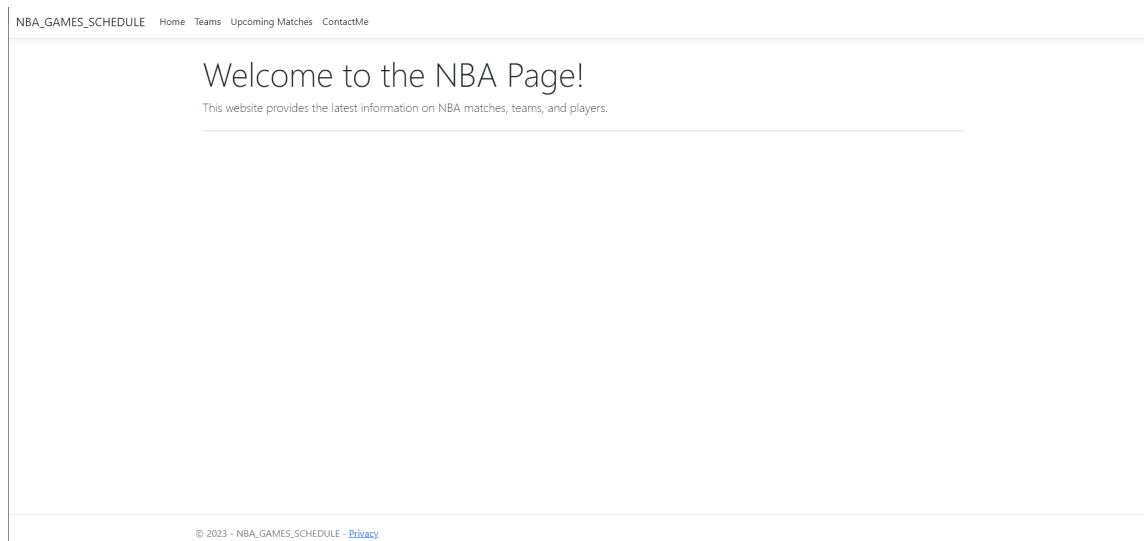
4 Opis poszczególnych widoków

Aplikacja webowa składa się z kilku podstron są to

- Home
- Teams
- Upcoming Matches

4.1 Home

Pierwszą opisaną stroną będzie strona Home. Ukazuje się ona po uruchomieniu aplikacji. W lewym górnym rogu znajdują się odnośniki do kolejnych podstron. Oprócz tego na środku znajduje się krótki opis aplikacji. Poniżej zamieszczono obraz, na którym znajduje się podgląd tego widoku.



Rysunek 1: Podgląd widoku Home

W celu napisania go skorzystano z poniższego kodu:

```
1  @ {
2      ViewData["Title"] = "Home Page";
3  }
4
5  <div class="container">
6      <div class="jumbotron mt-4">
7          <h1 class="display-4">Welcome to the NBA Page!</h1>
8          <p class="lead">This website provides the latest information on NBA matches, teams, and
9              players.</p>
10         <hr class="my-4">
11     </div>
12 </div>
```

Oraz korzysta on z kontrolera którego kod podano poniżej:

```
1  using Microsoft.AspNetCore.Mvc;
2  using Microsoft.EntityFrameworkCore;
3  using NBA_GAMES_SCHEDULE.Models;
4  using System.Diagnostics;
5
6
7  namespace NBA_GAMES_SCHEDULE.Controllers
8  {
9
10     public class HomeController : Controller
11     {
12         private readonly ILogger<HomeController> _logger;
13
14         public HomeController(ILogger<HomeController> logger)
15         {
16             _logger = logger;
17         }
18
19         public IActionResult Index()
20         {
21             return View();
22         }
23
24         public IActionResult Privacy()
25         {
26             return View();
27         }
28
29         [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
30         public IActionResult Error()
31         {
```

```

32         return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.
           TraceIdentifier });
33     }
34
35     public IActionResult ContactMe()
36     {
37         return View();
38     }
39
40
41
42 }
43 }

```

4.2 Teams

Po naciśnięciu odnośnika Teams aplikacja przechodzi do widoku Teams. Wyświetla on informacje o drużynie w postaci kafelka. Poniżej zamieszczono obraz, na którym zaprezentowany został ten widok:



Rysunek 2: Podgląd widoku Teams

W celu reprezentacji drużyny napisano model Team.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations.Schema;
4
5 namespace NBA_GAMES_SCHEDULE.Models
6 {
7     [Table("Teams")]
8     public partial class Team
9     {
10
11
12
13         public int TeamId { get; set; }
14         public string Name { get; set; }
15         public string City { get; set; }
16         public string Coach { get; set; }

```

```

17
18     public ICollection<Player> Players { get; set; }
19
20     public ICollection<Match> HomeMatches { get; set; }
21     public ICollection<Match> AwayMatches { get; set; }
22
23
24     [NotMapped]
25     public string ImagePath => $"/images/teams/{TeamId}.png";
26
27 }
28
29
30 }

```

Oraz Kontroler:

```

1  using Microsoft.AspNetCore.Mvc;
2  using System.Linq;
3  using NBA_GAMES_SCHEDULE.Models;
4  using Microsoft.EntityFrameworkCore;
5
6  public class TeamsController : Controller
7  {
8      private readonly MVCDContext _context;
9
10     public TeamsController(MVCDContext context)
11     {
12         _context = context;
13     }
14
15     public IActionResult Index()
16     {
17         var teams = _context.Teams.ToList();
18         return View(teams);
19     }
20
21
22
23     public async Task<IActionResult> Details(int? id)
24     {
25         if (id == null)
26         {
27             return NotFound();
28         }
29
30         var team = await _context.Teams
31             .Include(t => t.Players)
32             .FirstOrDefaultAsync(m => m.TeamId == id);
33
34
35
36         if (team == null)
37         {
38             return NotFound();
39         }
40
41         return View(team);
42
43     }
44
45
46     public async Task<IActionResult> TeamMatches(int? id)
47     {
48         if (id == null)
49         {
50             return NotFound();
51         }
52
53         var team = await _context.Teams
54             .Include(t => t.HomeMatches)
55             .FirstOrDefaultAsync(m => m.TeamId == id);
56
57
58
59         if (team == null)
60         {
61             return NotFound();
62         }
63

```

```

64         return View(team);
65
66     }
67
68
69
70
71
72
73     }

```

sam front-end został napisany w widoku Teams/Index.cshtml


```

1  @model IEnumerable<NBA_GAMES_SCHEDULE.Models.Team>
2  <h2>Teams </h2>
3
4  <div class="team-logos">
5      @foreach (var team in Model)
6      {
7          <div class="team-card">
8              <a asp-action="Details" asp-controller="Teams" asp-route-id="@team.TeamId">
9                  
10             </a>
11             <div>
12                 <h3>@team.Name</h3>
13                 <p>@team.City</p>
14                 <p>Coach: @team.Coach</p>
15             </div>
16         </div>
17     }
18 </div>

```

4.2.1 Teams/Details

Po naciśnięciu loga drużyny użytkownik przeniesiony zostanie do widoku Teams/Details Wyświetla on tabele zawodników danej drużyny. Przykładowy Widok został pokazany poniżej:



Coach:

Quin Snyder

City:

Atlanta

See Matches

Player	Pos	Ht	Wt	Birth Date
Dejounte Murray	SG	193	82	09.19.1996
Bogdan Bogdanović	SG	196	100	08.18.1992
Jalen Johnson	SF	206	100	12.18.2001
DeAndre Hunter	SF	203	102	12.02.1997
Saddiq Bey	SF	201	98	04.09.1999
Clint Capela	C	208	109	05.18.1994
Onyeka Okongwu	C	203	107	12.11.2000
Trae Young	PG	185	74	09.19.1998
AJ Griffin	SF	198	101	08.25.2003
Wesley Matthews	SG	193	100	10.14.1986
Garrison Mathews	SG	196	98	10.24.1996
Bruno Fernando	C	206	109	08.15.1998
Trent Forrest (TW)	SG	193	95	06.12.1998
Kobe Bufkin	SG	193	88	09.21.2003
Mouhamed Gueye	PF	211	95	11.09.2002
Miles Norris (TW)	PF	208	100	04.15.2000
Seth Lundy (TW)	SF	198	100	04.02.2000
Patty Mills	PG	188	82	08.11.1988
Jarkel Joiner	G	185	82	05.20.1999

Rysunek 3: Podgląd widoku Teams/Details

Korzysta on z tego samego z Modelu Player.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations.Schema;
4
5 namespace NBA_GAMES_SCHEDULE.Models
6 {
7     [Table("Players")]
8     public partial class Player
9     {
10         public int PlayerId { get; set; }
11         public int? TeamId { get; set; }
12         public string? Name { get; set; }
13         public string? Position { get; set; }
14         public int? Height { get; set; }
15         public int? Weight { get; set; }
16         public DateTime? BirthDate { get; set; }
17         public string? Nationality { get; set; }
18
19         public Team Team { get; set; }
20     }
21 }
```

Same dane zostały wybrane za pomocą funkcji Details znajdującej się w TeamsController.

```
1
2
3 public async Task<IActionResult> Details(int? id)
4 {
5     if (id == null)
6     {
7         return NotFound();
8     }
9
10    var team = await _context.Teams
11        .Include(t => t.Players)
12        .FirstOrDefaultAsync(m => m.TeamId == id);
13
14
15
16    if (team == null)
17    {
18        return NotFound();
19    }
20
21    return View(team);
22
23
24 }
```

Front-end bez css został napisany w poniższy sposób

```
1 @model NBA_GAMES_SCHEDULE.Models.Team
2
3
4
5 <div class="team-container">
6     <div class="team-logo">
7         
8     </div>
9     @<div class="team-info">
10
11
12         <p><strong>Coach:</strong> @Model.Coach</p>
13         <p><strong>City:</strong> @Model.City</p>
14     </div>
15 }
16 </div>
17
18 <a href="@Url.Action("TeamMatches", "Teams", new { id = Model.TeamId })" class="btn btn-primary">See
    Matches </a>
19
20 <table class="player-table">
21     <thead>
22         <tr>
23
24             <th>Player</th>
25             <th>Pos</th>
26             <th>Ht</th>
```
























```

27         <th>Wt</th>
28         <th>Birth Date</th>
29
30     </tr>
31 </thead>
32 <tbody>
33     @foreach (var player in Model.Players)
34     {
35         <tr>
36
37             <td>@player.Name</td>
38             <td>@player.Position</td>
39             <td>@player.Height</td>
40             <td>@player.Weight</td>
41             <td>@player.BirthDate?.ToString("MM/dd/yyyy")</td>
42
43         </tr>
44     }
45 </tbody>
46 </table>

```

4.2.2 Teams/Matches

Po naciśnięciu przycisku seeMatches ukazują się kolejny widok Team/Matches. Wyświetla on w postaci tabeli wszystkie mecze danej drużyny w trakcie najbliższego sezonu NBA. Widok Prezentuje się tak:

<div>  <div> Coach: Quin Snyder City: Atlanta </div> </div>					
Date	Time	Location	Home Team	Away Team	Away Team Logo
25.10.2023	07:00	State Farm Arena	Atlanta Hawks	Charlotte Hornets	
27.10.2023	07:30	State Farm Arena	Atlanta Hawks	New York Knicks	
29.10.2023	19:00	State Farm Arena	Atlanta Hawks	Milwaukee Bucks	
30.10.2023	19:30	State Farm Arena	Atlanta Hawks	Minnesota Timberwolves	
01.11.2023	19:30	State Farm Arena	Atlanta Hawks	Washington Wizards	
04.11.2023	19:00	State Farm Arena	Atlanta Hawks	New Orleans Pelicans	
06.11.2023	20:00	State Farm Arena	Atlanta Hawks	Oklahoma City Thunder	
09.11.2023	21:30	State Farm Arena	Atlanta Hawks	Orlando Magic	
11.11.2023	19:30	State Farm Arena	Atlanta Hawks	Miami Heat	
14.11.2023	19:00	State Farm Arena	Atlanta Hawks	Detroit Pistons	
15.11.2023	19:30	State Farm Arena	Atlanta Hawks	New York Knicks	
17.11.2023	19:30	State Farm Arena	Atlanta Hawks	Philadelphia 76ers	
21.11.2023	19:30	State Farm Arena	Atlanta Hawks	Indiana Pacers	
22.11.2023	19:30	State Farm Arena	Atlanta Hawks	Brooklyn Nets	
25.11.2023	19:00	State Farm Arena	Atlanta Hawks	Washington Wizards	
26.11.2023	18:00	State Farm Arena	Atlanta Hawks	Boston Celtics	
28.11.2023	19:30	State Farm Arena	Atlanta Hawks	Cleveland Cavaliers	
30.11.2023	20:00	State Farm Arena	Atlanta Hawks	San Antonio Spurs	
02.12.2023	20:00	State Farm Arena	Atlanta Hawks	Milwaukee Bucks	
11.12.2023	19:30	State Farm Arena	Atlanta Hawks	Denver Nuggets	
13.12.2023	19:30	State Farm Arena	Atlanta Hawks	Toronto Raptors	
15.12.2023	19:30	State Farm Arena	Atlanta Hawks	Toronto Raptors	

Rysunek 4: Podgląd widoku Teams
Matches

Korzysta on z modelu Match.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.ComponentModel.DataAnnotations.Schema;
5
6 namespace NBA_GAMES_SCHEDULE.Models
7 {
8     [Table("Matches")]
9     public partial class Match
10    {
11        public int MatchId { get; set; }
12        public DateTime? Date { get; set; }
13        public TimeSpan? Time { get; set; }
14        public string? Location { get; set; }
15        public int? HomeTeamId { get; set; }
16        [ForeignKey("HomeTeamId")]
17        public virtual Team HomeTeam { get; set; }
18
19        public int? AwayTeamId { get; set; }
20        [ForeignKey("AwayTeamId")]
21        public virtual Team AwayTeam { get; set; }
22
23
24
25
26
27    }
28 }
```

Same dane zostały wybrane za pomocą funkcji TeamMatches.cs znajdującej się w TeamsController.
























```
1
2
3 public async Task<IActionResult> TeamMatches(int? id)
4 {
5     if (id == null)
6     {
7         return NotFound();
8     }
9
10    var team = await _context.Teams
11        .Include(t => t.HomeMatches)
12        .FirstOrDefaultAsync(m => m.TeamId == id);
13
14
15
16    if (team == null)
17    {
18        return NotFound();
19    }
20
21    return View(team);
22
23
24 }
```

Front-End został napisany w poniższy sposób

```
1 @model NBA_GAMES_SCHEDULE.Models.Team
2 @{
3     var teamsDict = new Dictionary<int, string>
4     {
5         {1, "Atlanta Hawks"}, {2, "Boston Celtics"}, {3, "Brooklyn Nets"}, {4, "Charlotte Hornets"},
6         {5, "Chicago Bulls"}, {6, "Cleveland Cavaliers"}, {7, "Dallas Mavericks"}, {8, "Denver Nuggets"},
7         {9, "Detroit Pistons"}, {10, "Golden State Warriors"}, {11, "Houston Rockets"}, {12, "Indiana
8         Pacers"},
9         {13, "Los Angeles Clippers"}, {14, "Los Angeles Lakers"}, {15, "Memphis Grizzlies"}, {16, "Miami
10        Heat"},
11        {17, "Milwaukee Bucks"}, {18, "Minnesota Timberwolves"}, {19, "New Orleans Pelicans"}, {20, "New
12        York Knicks"},
13        {21, "Oklahoma City Thunder"}, {22, "Orlando Magic"}, {23, "Philadelphia 76ers"}, {24, "Phoenix
14        Suns"},
15        {25, "Portland Trail Blazers"}, {26, "Sacramento Kings"}, {27, "San Antonio Spurs"}, {28, "Toronto
16        Raptors"},
17        {29, "Utah Jazz"}, {30, "Washington Wizards"}
18    };
19
20     string GetTeamNameById(int teamId)
21     {
22         if (teamsDict.TryGetValue(teamId, out string teamName))
23         {
24             return teamName;
25         }
26         return "Unknown Team";
27     }
28
29     <div class="team-container">
30         <div class="team-logo">
31             
32         </div>
33         <div class="team-info">
34             <p><strong>Coach:</strong> @Model.Coach</p>
35             <p><strong>City:</strong> @Model.City</p>
36         </div>
37     </div>
38
39     <table class="player-table">
40         <thead>
41             <tr>
42                 <th>Date</th>
43                 <th>Time</th>
44                 <th>Location</th>
45                 <th>Home Team</th>
46                 <th>Away Team</th>
47                 <th>Away Team Logo</th>
48             </tr>
49         </thead>
50         <tbody>
51             <tbody>
52                 @foreach (var match in Model.HomeMatches)
53                 {
54                     <tr>
55                         <td>@match.Date?.ToString("d")</td>
56                         <td>@match.Time?.ToString(@"hh\:mm")</td>
57                         <td>@match.Location</td>
58                         <td>@GetTeamNameById(match.HomeTeamId ?? 0)</td>
59                         <td>@GetTeamNameById(match.AwayTeamId ?? 0)</td>
60                         <td></td>
62                     </tr>
63                 }
64             </tbody>
65         </table>
```


5 Upcoming Matches

Sekcja ta wyświetla mecze z odbywające się w ciągu najbliższego tygodnia. Poniżej zamieszczono jej podgląd:

<div><div><div>Coach:</div><div>Quin Snyder</div><div>City:</div><div>Atlanta</div></div></div>					
Date	Time	Location	Home Team	Away Team	Away Team Logo
25.10.2023	07:00	State Farm Arena	Atlanta Hawks	Charlotte Hornets	
27.10.2023	07:30	State Farm Arena	Atlanta Hawks	New York Knicks	
29.10.2023	19:00	State Farm Arena	Atlanta Hawks	Milwaukee Bucks	
30.10.2023	19:30	State Farm Arena	Atlanta Hawks	Minnesota Timberwolves	
01.11.2023	19:30	State Farm Arena	Atlanta Hawks	Washington Wizards	
04.11.2023	19:00	State Farm Arena	Atlanta Hawks	New Orleans Pelicans	
06.11.2023	20:00	State Farm Arena	Atlanta Hawks	Oklahoma City Thunder	
09.11.2023	21:30	State Farm Arena	Atlanta Hawks	Orlando Magic	
11.11.2023	19:30	State Farm Arena	Atlanta Hawks	Miami Heat	
14.11.2023	19:00	State Farm Arena	Atlanta Hawks	Detroit Pistons	
15.11.2023	19:30	State Farm Arena	Atlanta Hawks	New York Knicks	
17.11.2023	19:30	State Farm Arena	Atlanta Hawks	Philadelphia 76ers	
21.11.2023	19:30	State Farm Arena	Atlanta Hawks	Indiana Pacers	
22.11.2023	19:30	State Farm Arena	Atlanta Hawks	Brooklyn Nets	
25.11.2023	19:00	State Farm Arena	Atlanta Hawks	Washington Wizards	
26.11.2023	18:00	State Farm Arena	Atlanta Hawks	Boston Celtics	
28.11.2023	19:30	State Farm Arena	Atlanta Hawks	Cleveland Cavaliers	
30.11.2023	20:00	State Farm Arena	Atlanta Hawks	San Antonio Spurs	
02.12.2023	20:00	State Farm Arena	Atlanta Hawks	Milwaukee Bucks	
11.12.2023	19:30	State Farm Arena	Atlanta Hawks	Denver Nuggets	
13.12.2023	19:30	State Farm Arena	Atlanta Hawks	Toronto Raptors	
15.12.2023	19:30	State Farm Arena	Atlanta Hawks	Toronto Raptors	

Rysunek 5: Podgląd widoku UpcomigMatches

Korzysta ona z modelu Match oraz kontrolera MatchesController.cs

```
1  using Microsoft.AspNetCore.Mvc;
2  using Microsoft.EntityFrameworkCore;
3  using NBA_GAMES_SCHEDULE.Models;
4
5
6  namespace NBA_GAMES_SCHEDULE.Controllers
7  {
8      public class MatchesController : Controller
9      {
10         private readonly MVCDContext _context;
11
12         public MatchesController(MVCDContext context)
13         {
14             _context = context;
15         }
16
17         public IActionResult Index()
18         {
19             return View();
20         }
21
22         public async Task<IActionResult> UpcomingMatches()
23         {
24             var today = DateTime.Today;
25             var nextWeek = today.AddDays(7);
26
27             var upcomingMatches = await _context.Matches
28                 .Where(m => m.Date >= today && m.Date <= nextWeek)
29                 .OrderBy(m => m.Date)
30                 .ThenBy(m => m.Time)
31                 .ToListAsync();
32             Match previousMatch = null;
33             var filteredMatches = new List<Match>();
34
35             foreach (var match in upcomingMatches)
36             {
37                 if (previousMatch == null || (match.HomeTeamId != previousMatch.AwayTeamId || match.
38                     HomeTeamId != previousMatch.AwayTeamId))
39                 {
40                     filteredMatches.Add(match);
41                 }
42                 previousMatch = match;
43             }
44
45             return View(filteredMatches);
46         }
47
48         public IActionResult ContactMe()
49         {
50             return View();
51         }
52     }
53 }
54 }
```

oraz z widoku UpcomingMatches.cshtml:

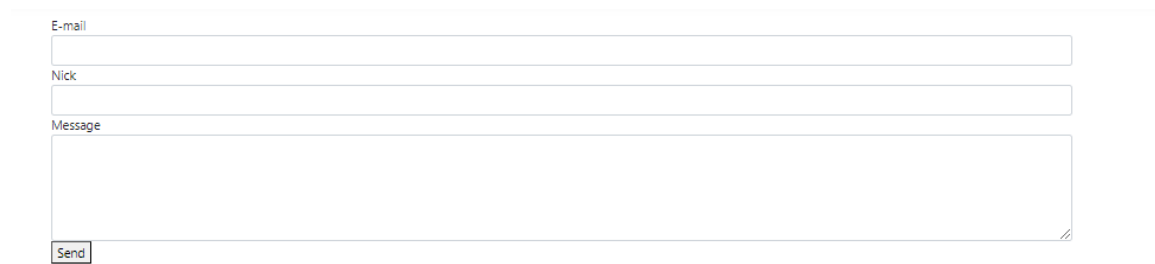
```
1  @model IEnumerable<NBA_GAMES_SCHEDULE.Models.Match>
2  @{
3      var teamsDict = new Dictionary<int, string>
4      {
5          {1, "Atlanta Hawks"}, {2, "Boston Celtics"}, {3, "Brooklyn Nets"}, {4, "Charlotte Hornets"},
6          {5, "Chicago Bulls"}, {6, "Cleveland Cavaliers"}, {7, "Dallas Mavericks"}, {8, "Denver Nuggets"},
7          {9, "Detroit Pistons"}, {10, "Golden State Warriors"}, {11, "Houston Rockets"}, {12, "Indiana
8          Pacers"},
9          {13, "Los Angeles Clippers"}, {14, "Los Angeles Lakers"}, {15, "Memphis Grizzlies"}, {16, "Miami
10         Heat"},
11         {17, "Milwaukee Bucks"}, {18, "Minnesota Timberwolves"}, {19, "New Orleans Pelicans"}, {20, "New
12         York Knicks"},
13         {21, "Oklahoma City Thunder"}, {22, "Orlando Magic"}, {23, "Philadelphia 76ers"}, {24, "Phoenix
14         Suns"},
15         {25, "Portland Trail Blazers"}, {26, "Sacramento Kings"}, {27, "San Antonio Spurs"}, {28, "Toronto
16         Raptors"},
17         {29, "Utah Jazz"}, {30, "Washington Wizards"}
18     };
19
20     string GetTeamNameById(int teamId)
21     {
22         if (teamsDict.TryGetValue(teamId, out string teamName))
23         {
24             return teamName;
25         }
26         return "Unknown Team";
27     }
28 }
29
30 <table class="match-table">
31     <thead>
32         <tr>
33             <th>Date</th>
34             <th>Time</th>
35             <th>Home Team</th>
36             <th>Home Team Logo</th>
37             <th>Away Team</th>
38             <th>Away Team Logo</th>
39             <th>Location</th>
40             <th>L</th>
41         </tr>
42     </thead>
43     <tbody>
44         @foreach (var match in Model)
45         {
46             <tr>
47                 <td>@match.Date?.ToString("yyyy-MM-dd")</td>
48                 <td>@match.Time?.ToString(@"hh:mm")</td>
49                 <td>@GetTeamNameById(match.HomeTeamId ?? 0)</td>
50                 <td></td>
52                 <td>@GetTeamNameById(match.AwayTeamId ?? 0)</td>
53                 <td></td>
55                 <td>@match.Location</td>
56                 <td></td>
57             </tr>
58         }
59     </tbody>
60 </table>
```

6 ContactMe

Dodatkowo stworzono widok, który umożliwia kontakt z twórcą. Prezentuje on formularz służący do przesłania wiadomości. Email, Nick, Treść wiadomości zapisywane są do bazy danych w tabeli UserMessages. W tym celu utworzono kolejną tabelę UserMessages. Poniżej zamieszczono kod sql, który ją tworzy:

```
1 CREATE TABLE UserMessages (
2     UserId INT PRIMARY KEY IDENTITY,
3     Email NVARCHAR(MAX),
4     Nick NVARCHAR(MAX),
5     Message NVARCHAR(MAX)
6 );
```

Obraz prezentujący podgląd formularza zamieszczono poniżej:



The screenshot shows a web form with three input fields: 'E-mail', 'Nick', and 'Message'. The 'Message' field is a larger text area. Below the fields is a 'Send' button.

Rysunek 6: Podgląd widoku UpcomigMatches

Napisano również model UserMessage.cs:

```
1  using  Microsoft.AspNetCore.Mvc;
2
3  namespace NBA_GAMES_SCHEDULE.Models
4  {
5      public class UserMessage
6      {
7          public int UserMessageId { get; set; }
8          public string Email { get; set; }
9          public string Nick { get; set; }
10         public string Message { get; set; }
11     }
12 }
13
14 }
```

Oraz Kontroler ContactMeController.cs

```
1  using  Microsoft.AspNetCore.Mvc;
2  using  Microsoft.EntityFrameworkCore;
3  using  NBA_GAMES_SCHEDULE.Models;
4  using  static NBA_GAMES_SCHEDULE.Models.UserMessage;
5
6  namespace NBA_GAMES_SCHEDULE.Controllers
7  {
8      public class ContactMeController : Controller
9      {
10         private readonly MVCDBcontext _context;
11
12         public ContactMeController(MVCDBcontext context)
13         {
14             _context = context;
15         }
16
17         public IActionResult ContactMe()
18         {
19             return View();
20         }
21
22         public IActionResult MessageSent()
23         {
24             return View();
25         }
26
27         [HttpPost]
28         public async Task<IActionResult> ContactMe(models.UserMessage userMessage)
29         {
30             if (ModelState.IsValid)
31             {
32
33                 _context.UserMessages.Add(userMessage);
34                 await _context.SaveChangesAsync();
35                 TempData["Message"] = "Twoja wiadomość została wysłana!";
36                 return RedirectToAction("MessageSent");
37             }
38             return View(userMessage);
39         }
40     }
41 }
```

7 Odnosiniki

Repozytorium github: https://github.com/PullPushSingar/Database_NBA_PROJECT_EF

Strona z której pozyskano dane do bazydanych: https://www.basketball-reference.com/leagues/NBA_2022_per_game.html

Wykonał: Hubert Kowalczyk