

Zastosowanie Programowania Obiektowego
Informatyka III Semestr (studia stacjonarne)

Piłka nożna - Katalog (Aplikacja PHP)

Skład zespołu:

Nr indeksu	Nazwisko	Imię
U-20019	Achtelik	Jakub
U-20041	Budnik	Oliwier



Politechnika Koszalińska
Koszalin 2023

Spis Treści

1	Koncepcja i funkcjonalność aplikacji	2
1.1	Założenia przyjęte dla tworzenia aplikacji:	2
1.1.1	Środowisko uruchomieniowe:	2
1.1.2	Cechy	3
1.1.3	Ograniczenia:	3
1.2	Narzędzia programistyczne:	4
1.3	Wykaz funkcjonalności aplikacji	5
1.4	Prototyp	6
2	Diagram przypadków użycia	8
2.1	Rysunek (diagram)	8
2.2	Opis	9
3	Diagram klas	10
3.1	Rysunek (diagram)	10
3.2	Opis przeznaczenia klas	10
3.3	Models	10
3.3.1	Aplikacja	10
3.3.2	BazaDanych	13
3.3.3	ZapytaniaSql	14
3.3.4	FiltrowanieSql	17
3.3.5	OperacjePilkarzy	20
3.3.6	WyswietlaniePilkarzy	21
3.3.7	PobieraczObrazowWikipedia	22
3.4	Controllers	24
3.4.1	Autoryzacja	24
3.4.2	FiltrowanieKontroler	25
3.4.3	KontrolerDanych	26
3.4.4	PilkarzPost	26
3.4.5	ZarządzaniePilkarzami	27
3.5	Views	28
3.5.1	StronaHtml	28
3.5.2	SzablonHtml	31
3.6	Helpers	37
3.6.1	BazaDanychHelper	37
3.6.2	FormularzHelper	38
3.7	Projekt	39
3.8	FileLoader	39
3.9	plik - index.php	40
4	Projekt Bazy Danych	41
4.1	Relacyjna baza danych SQL	41
4.2	Model bazy danych - Opis	41
4.2.1	Ustawienie uprawnień dla użytkownika	42
4.3	Rysunek	43

5	Opis działania i obsługi aplikacji	44
5.1	Przewodnik	44
5.1.1	Strona internetowa	44
5.2	Wyświetlanie oraz filtrowanie piłkarzy	45
5.3	Panel Logowania	46
5.4	Panel Administracyjny	47
5.5	Modyfikowanie danych na temat piłkarzy	47
5.5.1	Dodawanie piłkarza	47
5.5.2	Usuwanie piłkarza	48
5.5.3	Edycja piłkarza	49
5.6	Jak uruchomić projekt lokalnie - instrukcja	49
5.6.1	Uruchomienie kodu - PHP	49
5.6.2	Uruchomienie kontenera z bazą danych - Docker (Ubuntu 22.04)	50
5.6.3	Uruchomienie bazy danych - XAMPP (Windows 10)	52
5.6.4	Dodanie zmiennej środowiskowej PATH dla PHP - Windows 10	52
6	Wnioski	53
6.1	Podsumowanie	53
6.2	Podział pracy	53
6.2.1	Zadania wykonane przez U-20019	53
6.2.2	Zadania wykonane przez U-20041	53
7	Bibliografia - źródła	54

1 Koncepcja i funkcjonalność aplikacji

1.1 Założenia przyjęte dla tworzenia aplikacji:

Temat aplikacji: Baza danych (wprowadzanie, wyświetlanie, edycja, usuwanie, wyszukiwanie, sortowanie, baza danych, zawartość bazy danych dowolna - np. dane osobowe)

Nazwa aplikacji: Piłka nożna - Katalog (Aplikacja PHP)

Rodzaj aplikacji: Aplikacja internetowa

Architektura klient-serwer.

Aplikacja uruchamiana lokalnie na komputerach oraz zdalnie na serwerze VPS.

Aplikacja jest dostosowana również do urządzeń mobilnych.

Środowisko lokalne: Windows 10, Ubuntu 22.04 LTS

Środowisko produkcyjne (serwer): VPS Linux (Ubuntu 20.04 LTS):

1.1.1 Środowisko uruchomieniowe:

Dla Windows:

- XAMPP 8.2.4 (serwer HTTP Apache, Serwer bazy danych MariaDB, interpreter PHP)
- PHP 8.2.4 Development Server (serwer HTTP + interpreter PHP)

Dla Ubuntu:

- PHP 8.1.2-1ubuntu2.14 Development Server (serwer HTTP + interpreter PHP)
- Docker : obraz MySQL server version: 8.2.0 + phpMyAdmin

Dla VPS:

- Serwer HTTP Apache2 - apache2 2.4.41-4ubuntu3.14
- Serwer bazy danych MySQL - mysql-server ver 8.0.34-0ubuntu0.20.04.1
- Interpreter - PHP 8.1.2-1ubuntu2.14

1.1.2 Cechy

1. Zastosowanie statycznego typowania (zmiennych, funkcji, metod, pól klasy), podobnie jak w językach C/C++, Java, C#. Jest to bardziej przewidywalne i pozwala narzucić określony typ np. zwracanej zmiennej, aby uniknąć wielu błędów. Domyślnie PHP nie wymaga statycznego typowania.
2. Podział projektu na wiele plików według struktury MVC Model-View-Controller (pol. Model-Widok-Kontroler):
 - Model jest pewną reprezentacją problemu bądź logiki aplikacji.
 - Widok opisuje, jak wyświetlić pewną część modelu w ramach interfejsu użytkownika.
 - Kontroler przyjmuje dane wejściowe od użytkownika i reaguje na jego poczynania
3. Logika aplikacji będzie zawarta w sposób obiektowy w klasach, każda klasa to osobny plik.

1.1.3 Ograniczenia:

- PHP jest podatny na pewne rodzaje ataków, takich jak na przykład wstrzykiwanie SQL, dlatego bezpieczeństwo aplikacji nie jest na najwyższym możliwym poziomie i szczegółowa konfiguracja zabezpieczeń nie jest łatwa do wdrożenia w krótkim czasie
- PHP jest językiem interpretowanym dlatego wydajność w stosunku do języków kompilowanych jest niższa
- PHP nie posiada wszystkich elementów obiektowych znanych z innych języków
- Ograniczony czas, przez co nie można zawrzeć wszystkich celów w wzorcowy sposób zgodny w 100% z dokumentacją
- Ograniczenie aktualnej wiedzy, przez co niektóre elementy projektu mogą stanowić wyzwanie
- W naszej aplikacji proces logowania ogranicza się do jednego super użytkownika, którego poświadczenia są statycznie zapisane w pliku **KonfiguracjaApp.php** jako tzw. **plain text** (niezahaszowane hasła przechowywane jako zwykły, łatwy do odczytania i przechwycenia tekst), jest to zła praktyka jednak chcieliśmy uprościć tę część aplikacji ponieważ oddalała się ona dość znacząco od pierwotnego tematu, a wymagała dość dużo nadprogramowej pracy.

Plik konfiguracyjny **KonfiguracjaApp.php**:

```
1 <?php
2 // KONFIGURACJA APLIKACJI
3
4 define("UZYTKOWNIKADMIN", "admin");
5 define("HASLOADMIN", "admin");
6
7
8 define("TYTUL", "Pilka Nozna - Katalog");
9 define("AUTORZY", "Jakub Achtelik, Olivier Budnik");
10 define("ROK", "2023");
11 define("PRZEDMIOT", "III Semestr, Zastosowanie Programowania Obiektowego");
12 define("UCZLENIA", "Politechnika Koszalińska");
```

1.2 Narzędzia programistyczne:

Język: **PHP 8+ OOP**

Dodatkowe biblioteki: **mysqli** (łączenie się z bazą danych)

Dodatkowe technologie: **HTML, CSS, JavaScript, MySQL, FontAwesome**(ikonki)

Visual Studio Code + PHP Code Extensions: IDE (Zintegrowane środowisko programistyczne)

JetBrains PhpStorm: IDE (Zintegrowane środowisko programistyczne)

XAMPP: środowisko uruchomieniowe dla Windows

Docker: konteneryzacja bazy danych lokalnie

GIT – System Kontroli Wersji

phpMyAdmin - graficzna nakładka na serwer MySQL, ułatwiająca operacje na bazie danych

Brave, Google Chrome– przeglądarka internetowa posiadająca narzędzia Chrome DevTools

Microsoft Edge– przeglądarka internetowa oraz rozbudowany czytnik plików PDF

Pakiet **make** – automatyzacja poleceń w terminalu

FileZilla – klient FTP

PlantUML - narzędzie do tworzenia rysunków i schematów, z poziomu pików tekstowych

Trello – zarządzanie zadaniami w zespole

Figma – Prototypowanie wyglądu aplikacji

GIMP 2.10.34 – Prosta edycja oraz tworzenie grafiki rastrowej

1.3 Wykaz funkcjonalności aplikacji

Interfejs webowy, zarządzanie bazą danych z poziomu przeglądarki internetowej:

- edycje, usuwanie, dodawanie nowego piłkarza,
- sortowanie oraz wyświetlanie zdjęć,
- wyszukiwanie po nazwisku, imieniu itp.
- filtrowanie szczegółowe po np. kraju, pozycji itp.
- logowanie oraz autoryzacja użytkownika przeglądającego aplikację
- wyświetlanie zdjęcia piłkarza (pobierane z serwisu Wikipedia)

Użytkownik może za pomocą przeglądarki internetowej:

- połączyć się z serwerem na którym hostowana jest aplikacja
- zalogować się do panelu poprzez formularz logowania, uzyskać autoryzację
- Panel umożliwia przeglądanie katalogu piłkarzy w przystępnej formie oraz inne operacje (edycja, usuwaniem, filtrowanie itp.).
- Użytkownik końcowy (klient) nie musi posiadać znajomości obsługi relacyjnej bazy danych aby w intuicyjny sposób zarządzać aplikacją.

1.4 Prototyp

Prototyp graficzny interfejsu użytkownika. Wykonane w programie Figma.



Rysunek 1: Widok strony głównej



Rysunek 2: Widok ekranu logowania

The screenshot shows a web application titled 'Pilka Nożna Katalog'. At the top, there is a navigation bar with four buttons: 'Strona Główna', 'Dodaj', 'Szukaj', and 'Wyloguj'. The 'Dodaj' button is highlighted. Below the navigation bar, the main content area is titled 'Dodawanie'. It contains a form with six input fields, each with a label: 'imię', 'nazwisko', 'kraj', 'pozycja', 'wzrost', and 'dodaj'. The 'dodaj' field is a submit button.

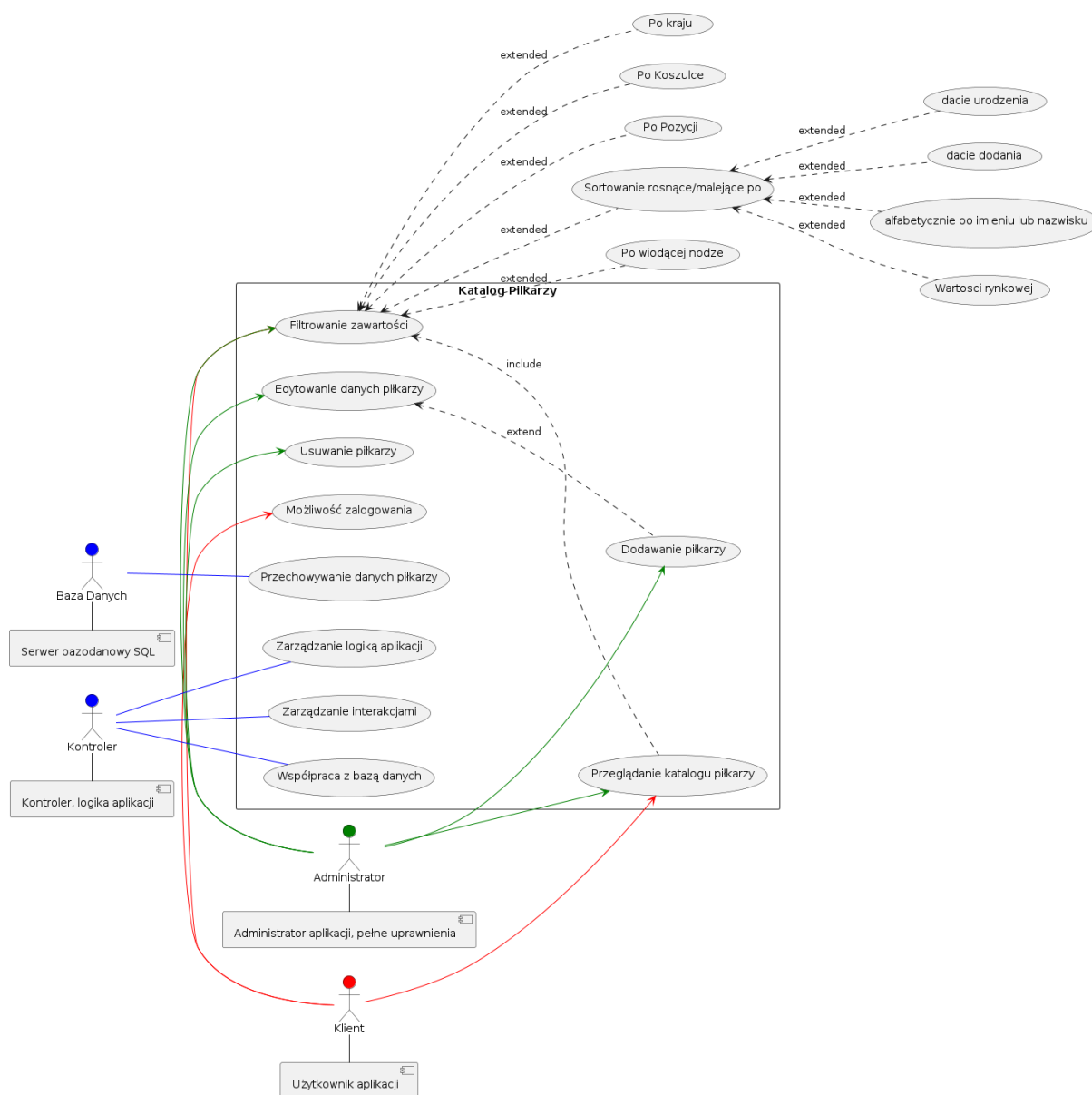
Rysunek 3: Widok strony głównej

The screenshot shows the same web application 'Pilka Nożna Katalog' with the same navigation bar. The 'Edycja' button is highlighted. The main content area is titled 'Edycja'. It contains a form with six input fields, each with a label: 'Lionel', 'Messi', 'Argentyna', 'napastnik', '1.62m', and 'edycja'. The 'edycja' field is a submit button.

Rysunek 4: Widok strony głównej

2 Diagram przypadków użycia

2.1 Rysunek (diagram)



Rysunek 5: Diagram przypadków użycia

2.2 Opis

Aktorami osobowymi są: Administrator oraz Klient, są to odbiorcy aplikacji

Aktorami bezosobowymi, abstrakcyjnymi są:

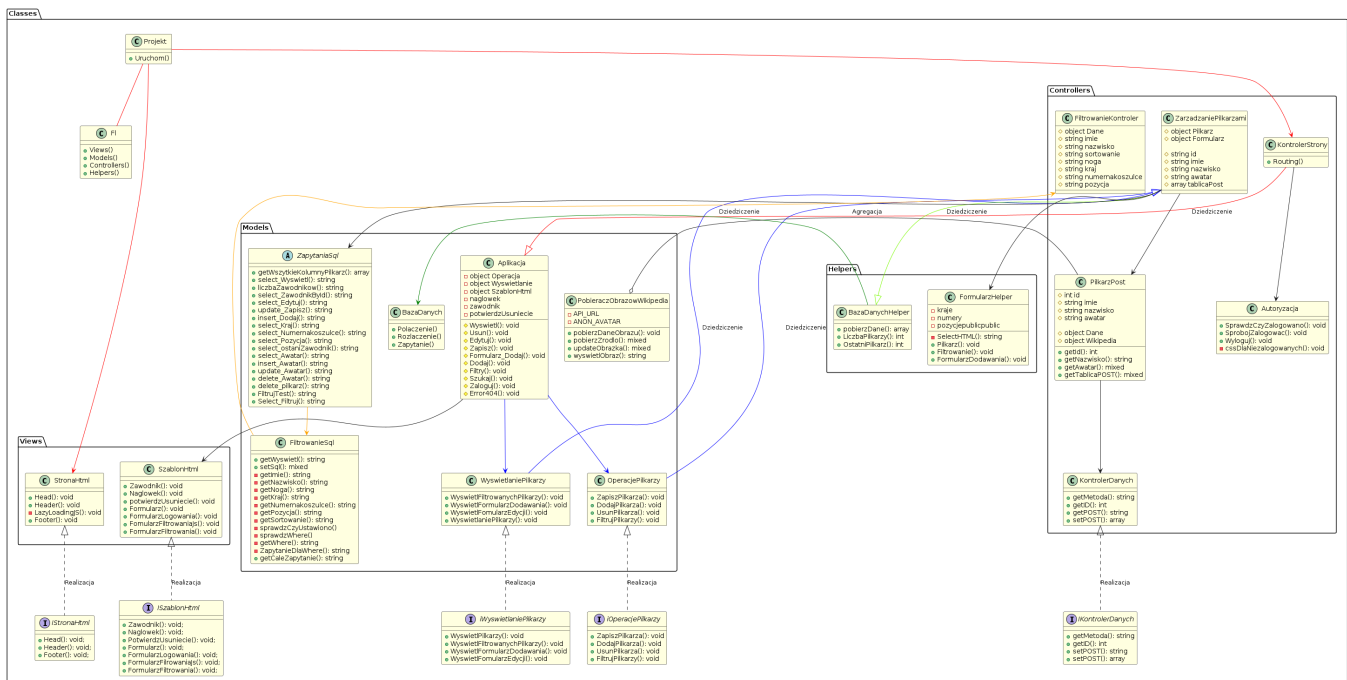
- Kontroler – reprezentuje logikę aplikacji
- BazaDanych – reprezentuje miejsce, gdzie informację są przechowywane

Podstawowe operacje aktorów:

- Klient -> (Przeglądanie katalogu piłkarzy)
- Klient -> (Filtrowanie zawartości)
- Klient -> (Możliwość zalogowania)
- Administrator -> (Przeglądanie katalogu piłkarzy)
- Administrator -> (Filtrowanie zawartości)
- Administrator -> (Dodawanie piłkarzy)
- Administrator -> (Edytowanie danych piłkarzy)
- Administrator -> (Usuwanie piłkarzy)
- BazaDanych – (Przechowywanie danych piłkarzy)
- Kontroler – (Zarządzanie interakcjami)
- Kontroler – (Współpraca z bazą danych)
- Kontroler – (Zarządzanie logiką aplikacji)

3 Diagram klas

3.1 Rysunek (diagram)



Rysunek 6: Diagram klas - Obraz w pełnej rozdzielczości <https://tiny.pl/cnmwh>

3.2 Opis przeznaczenia klas

Klasy są segregowane według modelu **MVC** i pomocniczych klas opisanych jako **Helpers**

3.3 Models

3.3.1 Aplikacja

Zależności: Models\OperacjePilkarzy, Models\WyświetlaniePilkarzy, Views\SzablonHtml.

Stanowi rdzeń całej struktury programu, zarządzając jego ogólnym działaniem, inicjalizacją, oraz kontrolą przepływu danych i interakcji między poszczególnymi komponentami. Jest to kluczowy element, który koordynuje i integruje funkcjonalność innych klas, tworząc spójną aplikację. Skupia się na zarządzaniu operacjami związanymi z piłkarzami w kontekście aplikacji. Posiada różne chronione metody, które współpracują z innymi klasami, takimi jak OperacjePilkarzy, WyświetlaniePilkarzy oraz SzablonHtml. Te metody wykonują różne zadania, takie jak wyświetlanie, usuwanie, edytowanie, dodawanie, filtrowanie oraz obsługę błędów w aplikacji związanych z piłkarzami. Klasa ta inicjuje obiekty innych klas w swoim konstruktorze i wykorzystuje je do odpowiedniego przetwarzania i

wyświetlania danych w interfejsie użytkownika.

Dostępne metody, pełnią następującą funkcjonalność:

- **Wyświetl()** - Wyświetla piłkarzy
- **Usun()** - Usuwa piłkarza
- **Edytuj()** - Wyświetla formularz do edycji
- **Zapisz()** - Zapisuje dane przesłane z formularza
- **Formularz_Dodaj()** - Wyświetla pusty formularz służący dodawaniu nowego użytkownika
- **Dodaj()** - Dodaje użytkownika do bazy danych na podstawie przesłanych z formularza danych
- **Filtruj()** - Wyświetla formularz, który służy do filtrowania
- **Szukaj()** - Wyświetla piłkarzy na podstawie wybranych parametrów
- **Zaloguj()** - Wyświetla formularz logowania
- **Error404()** - Wyświetla status błędu, jeżeli strona nie istnieje

```
1 <?php
2
3 namespace Pilkanozna\Models;
4
5
6 use Pilkanozna\Views\SzablonHtml;
7 use Pilkanozna\Models\WyświetlaniePiłkarzy;
8 use Pilkanozna\Models\OperacjePiłkarzy;
9
10
11 class Aplikacja
12 {
13
14     private object $Operacja;
15     private object $Wyświetlanie;
16
17     private object $SzablonHtml;
18
19     private $naglowek;
20     private $zawodnik;
21     private $potwierdzUsuniecie;
22
23
24
25     public function __construct()
26     {
27
28         $this->Operacja = new OperacjePiłkarzy();
29         $this->Wyświetlanie = new WyświetlaniePiłkarzy();
30         $this->SzablonHtml = new SzablonHtml();
31
32         $this->naglowek = [$this->SzablonHtml, 'Naglowek'];
33         $this->zawodnik = [$this->SzablonHtml, 'Zawodnik'];
34         $this->potwierdzUsuniecie = [$this->SzablonHtml, 'PotwierdzUsuniecie'];
35
36     }
37
38     protected function Wyświetl(): void
39     {
40         $this->Wyświetlanie->WyświetlPiłkarzy($this->naglowek, $this->zawodnik);
41     }
42 }
```

```
42
43
44     protected function Usun(): void
45     {
46         $this->Operacja->UsunPilkarza($this->naglowek,$this->potwierdzUsuniecie);
47     }
48
49     protected function Edytuj(): void
50     {
51         $this->SzablonHtml->Naglowek("EDYCJA");
52         $this->Wyswietalnie->WyswietlFomularzEdycji();
53     }
54
55     protected function Zapisz(): void
56     {
57         $this->Operacja->ZapiszPilkarza($this->naglowek);
58         header( "refresh:1;url=/" );
59         $this->Wyswietl();
60     }
61
62     protected function Formularz_Dodaj(): void
63     {
64         $this->SzablonHtml->Naglowek("Dodawanie");
65         $this->Wyswietalnie->WyswietlFormularzDodawania();
66     }
67
68     protected function Dodaj(): void
69     {
70
71         $this->Operacja->DodajPilkarza($this->naglowek);
72         header( "refresh:1;url=/" );
73         $this->Wyswietl();
74     }
75
76
77
78     private function Filtry(): void
79     {
80         $this->SzablonHtml->Naglowek("Filtry");
81         $this->Operacja->FiltrujPilkarzy();
82         $this->SzablonHtml->Naglowek("Wyniki wyszukiwania: <b id='szukanypilkarz'></b>");
83     }
84
85
86     protected function Szukaj(): void
87     {
88         $this->Filtry();
89         $this->Wyswietalnie->WyswietlFiltrowanychPilkarzy($this->naglowek, $this->zawodnik);
90     }
91
92
93
94     protected function Zaloguj(): void
95     {
96
97         $this->SzablonHtml->Naglowek("Zaloguj sie");
98         $this->SzablonHtml->FormularzLogowania();
99     }
100
101
102     protected function Error404(): void
103     {
104         $this->SzablonHtml->Naglowek("404 - Strona nie istnieje! ");
105     }
106
107
108 }
```

3.3.2 BazaDanych

Odpowiedzialna jest za obsługę połączenia z bazą danych, zarządzanie połączeniem oraz zapewnienie ogólnej komunikacji z nią. Jest kluczowym połączeniem między aplikacją a danymi przechowywanymi w systemie. Dostępne metody, pełnią następującą funkcjonalność:

- **Połączenie()** - Ustawienie połączenie z serwerem bazy danych
- **Rozłączenie()** - Kończy i przerywa połączenie
- **Zapytanie()** - Wykonuje polecenie SQL i pozwala przechwycić rezultat

Plik konfiguracyjny **KonfiguracjaDB.php**

```
1 <?php
2 // KONFIGURACJA DOSTĘPU DO SERWERA BAZY DANYCH MYSQL - port 3306 domyslny dla mysql
3 define("PORT", "3333");
4 define("HOST", "127.0.0.1");
5 define("UZYTKOWNIK", "projekt");
6 define("HASLO", "Pracownia107!");
7 define("BAZADANYCH", "pilkanozna");

1 <?php
2 namespace Pilkanozna\Models;
3
4 use mysqli;
5
6
7 class BazaDanych
8 {
9     private ?mysqli $polaczenie = null;
10
11     public function Polaczenie(): void
12     {
13         include_once './KonfiguracjaDB.php';
14         $this->polaczenie = new mysqli(HOST, UZYTKOWNIK, HASLO, BAZADANYCH, PORT);
15
16         if ($this->polaczenie->connect_error) {
17             echo "<h1 style='color: red;'>Bład polacznienia z baza danych: " . $this->polaczenie->connect_error .
18                 "</h1>";
19             exit();
20         }
21
22     protected function Rozlaczenie(): void
23     {
24         if ($this->polaczenie) {
25             $this->polaczenie->close();
26         }
27     }
28
29     protected function Zapytanie(string $sql)
30     {
31         if ($this->polaczenie === null) {
32             $this->Polaczenie();
33         }
34
35         $zapytanie = $this->polaczenie->query($sql);
36
37         if (!$zapytanie) {
38             echo "<p>Bład w zapytaniu: " . $this->polaczenie->error . "</p>";
39             exit();
40         }
41
42         return $zapytanie;
43     }
44 }
```

3.3.3 ZapytaniaSql

Definiuje i obsługuje generowanie zapytań SQL do bazy danych. Odpowiada za tworzenie struktur zapytań, umożliwiając aplikacji komunikację z bazą w celu pobierania, aktualizacji, usuwania i wstawiania danych. Metody zawarte w klasie odpowiadają operacją, jakie wykonuje dane polecenie zawarte w metodzie.

```
1 <?php
2 namespace Pilkanozna\Models;
3
4 use Pilkanozna\Models\FiltrowanieSql;
5
6 abstract class ZapytaniaSql
7 {
8
9     public static function getWszystkieKolumnyPilkarz(): array
10     {
11         $kolumny = array();
12         $kolumny = [
13             "id", "imie", "nazwisko", "wzrost",
14             "data_urodzenia", "wiodaca_noga",
15             "wartosc_rynkowa", "ilosc_strzelonych_goli",
16             "fk_kraj", "fk_numernakoszulce", "fk_pozycja",
17             "pk_kraj", "pk_numernakoszulce", "pk_pozycja"
18         ];
19
20     }
21
22     public static function select_Wyswietl(): string
23     {
24         return <<<SQL
25         SELECT PK_pilkarz as 'id', imie, nazwisko, wzrost, data_urodzenia, wiodaca_noga, wartosc_rynkowa,
26         ilosc_strzelonych_goli,
27         krajpilkarza.nazwa as 'pilkarzkraj', numernakoszulce.numer, pozycja.nazwa as 'pozycja',
28         awatar.link as 'link'
29         FROM pilkarz
30         join krajpilkarza on FK_kraj=PK_kraj
31         join numernakoszulce on FK_numernakoszulce=PK_numernakoszulce
32         join pozycja on FK_pozycja=PK_pozycja
33         join awatar on Fk_pilkarz=Pilkarz
34         order by pk_pilkarz DESC
35         SQL;
36     }
37
38     public static function liczbaZawodnikow(): string
39     {
40         return <<<SQL
41         SELECT count(*) as 'liczba_pilkarzy'
42         FROM pilkarz
43         SQL;
44     }
45
46     public static function select_ZawodnikById($id): string
47     {
48         return <<<SQL
49         SELECT imie, nazwisko
50         FROM pilkarz
51         WHERE pk_pilkarz = $id
52         SQL;
53     }
54
55     public static function select_Edytuj(string $id): string
56     {
57         return <<<SQL
58         SELECT PK_pilkarz as 'id', imie, nazwisko, wzrost, data_urodzenia, wiodaca_noga, wartosc_rynkowa,
59         ilosc_strzelonych_goli, krajpilkarza.nazwa as 'fk_kraj', numernakoszulce.numer as
60         'fk_numernakoszulce', pozycja.nazwa as 'fk_pozycja',
61         krajpilkarza.pk_kraj as 'pk_kraj',
```



```
63     numernakoszulce.pk_numernakoszulce as 'pk_numernakoszulce',
64     pozycja.pk_pozycja as 'pk_pozycja'
65 FROM pilkarz
66 join krajpilkarza on FK_kraj=PK_kraj
67 join numernakoszulce on FK_numernakoszulce=PK_numernakoszulce
68 join pozycja on FK_pozycja=PK_pozycja
69 where 'pk_pilkarz' = $id
70 SQL;
71 }
72
73 public static function update_Zapisz(int $id,array $setPOST): string
74 {
75     return <<<SQL
76     UPDATE pilkarz SET
77     imie = "${setPOST["imie"]}",
78     nazwisko = "${setPOST["nazwisko"]}",
79     wzrost = ${setPOST["wzrost"]},
80     data_urodzenia = "${setPOST['data_urodzenia']}",
81     wiodaca_noga = "${setPOST["wiodaca_noga"]}",
82     wartosc_rynkowa = ${setPOST["wartosc_rynkowa"]},
83     ilosc_strzelonych_goli = ${setPOST["ilosc_strzelonych_goli"]},
84     fk_kraj = ${setPOST["fk_kraj"]},
85     fk_numernakoszulce = ${setPOST["fk_numernakoszulce"]},
86     fk_pozycja = ${setPOST["fk_pozycja"]}
87     WHERE PK_pilkarz = $id
88     SQL;
89
90 }
91
92 public static function insert_Dodaj(array $setPOST)
93 {
94     return <<<SQL
95     INSERT INTO pilkarz
96     (
97         pk_pilkarz,
98         imie,nazwisko,
99         wzrost,data_urodzenia,
100         wiodaca_noga,
101         wartosc_rynkowa,
102         ilosc_strzelonych_goli,
103         fk_kraj,
104         fk_numernakoszulce,
105         fk_pozycja
106     )
107     VALUES
108     (
109         NULL,
110         "${setPOST["imie"]}",
111         "${setPOST["nazwisko"]}",
112         ${setPOST["wzrost"]},
113         "${setPOST['data_urodzenia']}",
114         "${setPOST['wiodaca_noga']}",
115         ${setPOST["wartosc_rynkowa"]},
116         ${setPOST["ilosc_strzelonych_goli"]},
117         ${setPOST["fk_kraj"]},
118         ${setPOST["fk_numernakoszulce"]},
119         ${setPOST["fk_pozycja"]}
120     )
121     SQL;
122
123 }
124
125 public static function select_Kraj()
126 {
127     return <<<SQL
128     SELECT pk_kraj, nazwa
129     FROM 'krajpilkarza'
130     SQL;
131 }
132
133 public static function select_Numernakoszulce()
134 {
```

```
135         return <<<SQL
136         SELECT pk_numernakoszulce, numer
137         FROM numernakoszulce
138         SQL;
139     }
140
141     public static function select_Pozycja()
142     {
143         return <<<SQL
144         SELECT pk_pozycja, nazwa
145         FROM pozycja
146         SQL;
147     }
148
149     public static function select_ostaniZawodnik()
150     {
151         return <<<SQL
152         SELECT pk_pilkarz FROM 'pilkarz'
153         ORDER BY 'pilkarz'.'.pk_pilkarz'
154         DESC limit 1
155         SQL;
156     }
157
158
159     public static function select_Awatar()
160     {
161         return <<<SQL
162         SELECT link, fk_pilkarz
163         FROM awatar
164         SQL;
165     }
166
167     public static function insert_Awatar($link, $liczba)
168     {
169         return <<<SQL
170         INSERT INTO 'awatar' ('pk_awatar', 'link', 'fk_pilkarz')
171         VALUES (NULL, '$link', '$liczba')
172         ;
173         SQL;
174     }
175
176
177     public static function update_Awatar($link, $pk_pilkarz)
178     {
179         return <<<SQL
180         UPDATE awatar
181         SET link = "$link"
182         WHERE awatar.fk_pilkarz = $pk_pilkarz;
183         SQL;
184     }
185
186
187     public static function delete_Awatar($pk_pilkarz)
188     {
189         return <<<SQL
190         DELETE FROM 'awatar'
191         WHERE 'awatar'.'.fk_pilkarz' = $pk_pilkarz
192         ;
193         SQL;
194     }
195
196
197
198     public static function delete_pilkarz($pk_pilkarz)
199     {
200         return <<<SQL
201         DELETE FROM pilkarz WHERE
202         PK_pilkarz = $pk_pilkarz
203         ;
204         SQL;
205     }
206 }
```

```
207 public static function FiltrujTest()
208 {
209     $test = new FiltrowanieSql();
210     echo $test->getCaleZapytanie();
211 }
212
213
214 public static function Select_Filtruj()
215 {
216     $filtr = new FiltrowanieSql();
217     return $filtr->getCaleZapytanie();
218 }
219
220
221
222
223 }
224 }
```

3.3.4 FiltrowanieSql

Jest odpowiedzialna za filtrowanie danych otrzymanych z bazy danych. Zapewnia mechanizmy filtrowania danych, aby zwracać jedynie pożądane informacje w spójny sposób. Wynikiem końcowym metody *getCaleZapytanie()* jest jedno poprawne zapytanie SQL przekazane do wykonania do bazy danych.

```
1 <?php
2 namespace Pilkanozna\Models;
3
4 use Pilkanozna\Controllers\FiltrowanieKontroler;
5
6 class FiltrowanieSql extends FiltrowanieKontroler
7 {
8
9     public static function getWyswietl(): string
10     {
11         return <<<SQL
12         SELECT PK_pilkarz as 'id', imie, nazwisko, wzrost, data_urodzenia, wiodaca_noga, wartosc_rynkowa,
13         ilosc_strzelonych_goli,
14         krajpilkarza.nazwa as 'pilkarzkraj', numernakoszulce.numer, pozycja.nazwa as 'pozycja',
15         awatar.link as 'link'
16         FROM pilkarz
17         join krajpilkarza on FK_kraj=PK_kraj
18         join numernakoszulce on FK_numernakoszulce=PK_numernakoszulce
19         join pozycja on FK_pozycja=PK_pozycja
20         join awatar on Fk_pilkarz=Pk_pilkarz
21         SQL;
22     }
23
24     public function setSql($kolumna,$parametr)
25     {
26         $sql = " $kolumna='$parametr' ";
27         $sql = $this->sprawdzCzyUstawiono($parametr,$sql);
28
29         if($sql == "") $sql = " $kolumna IS NOT NULL ";
30
31         return $sql;
32     }
33
34
35     private function getImie(): string
36     {
37
38         return $this->setSql("imie",$this->imie);
39     }
40
41     private function getNazwisko(): string
42     {
43         return $this->setSql("nazwisko",$this->nazwisko);
44     }
45 }
```

```
44     }
45
46
47
48     private function getNoga(): string
49     {
50         return $this->setSql("pilkarz.wiodaca_noga",$this->noga);
51     }
52
53     private function getKraj(): string
54     {
55         return $this->setSql("pilkarz.fk_kraj",$this->kraj);
56     }
57
58     private function getNumernakoszulce(): string
59     {
60         return $this->setSql("pilkarz.fk_numernakoszulce",$this->numernakoszulce);
61     }
62
63
64     private function getPozycja(): string
65     {
66         return $this->setSql("pilkarz.fk_pozycja",$this->pozycja);
67     }
68
69
70     private function getSortowanie(): mixed
71     {
72         $SqlSortowanie = " ORDER BY ";
73         switch($this->sortowanie)
74         {
75
76             case 'najnowsze':
77                 $SqlSortowanie .= "pk_pilkarz ASC";
78                 break;
79
80             case 'najstarsze':
81                 $SqlSortowanie .= "pk_pilkarz DESC";
82                 break;
83
84             case 'a-z':
85                 $SqlSortowanie .= "nazwisko ASC";
86                 break;
87
88             case 'z-a':
89                 $SqlSortowanie .= "nazwisko DESC";
90                 break;
91
92             case "wzrost-desc":
93                 $SqlSortowanie .= "wzrost DESC";
94                 break;
95
96             case "wzrost-asc":
97                 $SqlSortowanie .= "wzrost ASC";
98
99                 break;
100
101             case "dataurodzania-desc":
102                 $SqlSortowanie .= "data_urodzenia DESC";
103                 break;
104
105             case "dataurodzania-asc":
106                 $SqlSortowanie .= "data_urodzenia ASC";
107                 break;
108
109
110             case "wartosc-desc":
111                 $SqlSortowanie .= "wartosc_rynkowa DESC";
112                 break;
113
114             case "wartosc-asc":
115                 $SqlSortowanie .= "wartosc_rynkowa ASC";
```

```
116         break;
117     }
118
119     $SqlSortowanie = $this->sprawdzCzyUstawiono($this->sortowanie,$SqlSortowanie);
120
121     return $SqlSortowanie;
122 }
123
124
125 public function sprawdzCzyUstawiono($parametr, $sql): mixed
126 {
127     return ($parametr != "") ? $sql : "";
128 }
129
130
131
132 public function sprawdzWhere(): bool
133 {
134     if(
135         $this->imie OR
136         $this->nazwisko OR
137         $this->noga OR
138         $this->kraj OR
139         $this->numernakoszulce OR
140         $this->pozycja
141     )
142     {
143         return true;
144     }
145     else{
146         return false;
147     }
148 }
149
150 public function getWhere(): mixed
151 {
152     if($this->sprawdzWhere()) return " WHERE ";
153     else return false;
154 }
155
156 public function ZapytanieDlaWhere(): string
157 {
158     $Imie = $this->getImie();
159     $Nazwisko = $this->getNazwisko();
160     $Noga = $this->getNoga();
161     $Kraj = $this->getKraj();
162
163     $Numer = $this->getNumernakoszulce();
164     $Pozycja = $this->getPozycja();
165
166
167     $Sql = $Imie . " AND " . $Nazwisko .
168     " AND " . $Noga . " AND " . $Kraj
169     . " AND " . $Numer . " AND " . $Pozycja
170     ;
171
172
173     return $Sql;
174 }
175
176
177
178 public function getCaleZapytanie(): string
179 {
180     $Poczatek = $this->getWyswietl();
181     $Sortowanie = $this->getSortowanie();
182     $Where = $this->getWhere();
183
184     $ZapytaniaDlaWhere = $this->ZapytanieDlaWhere();
185
186     $Sql = $Poczatek;
187 }
```

```
188         if(!$Where) return $Sql . $Sortowanie;
189
190         $Sql .= $Where . $ZapytaniaDlaWhere . $Sortowanie;
191         return $Sql;
192     }
193 }
194 }
```

3.3.5 OperacjePilkarzy

Zależności: Controllers\ZarządzaniePilkarzami

Dostarcza zestaw operacji związanych z zarządzaniem danymi dotyczącymi piłkarzy w aplikacji. Obejmuje dodawanie, usuwanie, edycję danych piłkarzy oraz inne operacje z nimi związane.

Dostępne metody pełnią następującą funkcjonalność:

- **DodajPilkarza()** - Wykonuje zapytanie do bazy danych i dodaje piłkarza
- **ZapiszPilkarza()** - Wykonuje zapytanie do bazy danych i aktualizuje dane na temat piłkarza
- **FiltrujPilkarzy()** - Wykonuje zapytanie do bazy danych, zwraca przefiltrowane wyniki
- **UsunPilkarza()** - Wykonuje zapytanie do bazy danych i usuwa piłkarza

```
1  <?php
2
3  namespace Pilkanozna\Models;
4
5  use Pilkanozna\Controllers\ZarządzaniePilkarzami;
6
7  interface iOperacjePilkarzy
8  {
9      public function ZapiszPilkarza(mixed $Naglowek): void;
10     public function DodajPilkarza(mixed $Naglowek): void;
11     public function UsunPilkarza(mixed $Naglowek, mixed $SzablonPotwierdzenia): void;
12     public function FiltrujPilkarzy(): void;
13 }
14
15 class OperacjePilkarzy extends ZarządzaniePilkarzami implements iOperacjePilkarzy
16 {
17     public function ZapiszPilkarza(mixed $Naglowek): void
18     {
19         $Naglowek("Zapisano <b>{$this->imie} {$this->nazwisko}</b>!");
20
21         $this->Zapytanie(
22             ZapytaniaSql::update_Zapisz($this->id,$this->tablicaPost)
23         );
24
25         $this->Zapytanie(
26             ZapytaniaSql::update_Awatar($this->awatar,$this->id)
27         );
28     }
29
30     public function DodajPilkarza(mixed $Naglowek): void
31     {
32         $Naglowek("Dodano <b>{$this->imie} {$this->nazwisko}</b>!");
33
34         $this->Zapytanie
35         (
36             ZapytaniaSql::insert_Dodaj(
37                 $this->Pilkarz->getTablicaPOST(ZapytaniaSql::getWszytkieKolumnyPilkarz())
38             )
39         )
40     }
41 }
```

```
39     );
40
41     $liczba = $this->OstatniPilkarz();
42
43     $this->Zapytanie(
44         ZapytaniaSql::insert_Awatar($this->awatar, $liczba)
45     );
46 }
47
48 public function UsunPilkarza(mixed $Naglowek, mixed $SzablonPotwierdzenia): void
49 {
50     $imie = "";
51     $nazwisko = "";
52
53     $wynik = $this->Zapytanie(
54         ZapytaniaSql::select_ZawodnikById($this->id)
55     );
56
57     while ($wiersz = $wynik->fetch_assoc()){
58         $imie = $wiersz['imie'];
59         $nazwisko = $wiersz['nazwisko'];
60     }
61
62     $potwierdzenie = (isset($_GET['potwierdzenie'])) ? $_GET['potwierdzenie'] : null;
63
64     if($potwierdzenie == "tak"){
65
66         $Naglowek("Usunieto pilkarza <b>$imie $nazwisko</b>!");
67
68         $this->Zapytanie(ZapytaniaSql::delete_pilkarz($this->id));
69         $this->Zapytanie(ZapytaniaSql::delete_Awatar($this->id));
70
71         header( "refresh:1;url=/" );
72
73     }
74     else{
75         $SzablonPotwierdzenia($this->id,$imie,$nazwisko);
76     }
77 }
78
79 public function FiltrujPilkarzy(): void
80 {
81     $this->Formularz->Filtrowanie([$this, 'pobierzDane']);
82 }
83
84 }
```

3.3.6 WyświetlaniePilkarzy

Klasa koncentruje się na prezentacji danych dotyczących piłkarzy. Zapewnia funkcje prezentacji, sortowania, wyświetlania informacji o piłkarzach, dostosowanej do interfejsu użytkownika.

Dostępne metody pełnią następującą funkcjonalność:

- **WyświetlaniePilkarzy()** - Wykonuje zapytanie do bazy danych i wyświetla piłkarzy
- **WyświetlFiltrowanychPilkarzy()** - Wykonuje zapytanie do bazy danych i wyświetla piłkarzy według zastosowanych filtrów
- **WyświetlFormularzDodawania()** - Wyświetla formularz dodawania
- **WyświetlFormularzEdycji()** - Wykonuje zapytanie do bazy danych i wyświetla formularz do edycji

```
1 <?php
2 namespace Pilkanozna\Models;
```

```
3
4 use Pilkanozna\Controllers\ZarzadzaniePilkarzami;
5
6 interface iWyswietlaniePilkarzy
7 {
8     public function WyswietlPilkarzy(mixed $Naglowek, mixed $SzablonZawodnik): void;
9     public function WyswietlFiltrowanychPilkarzy(mixed $Naglowek, $SzablonZawodnik): void;
10    public function WyswietlFormularzDodawania(): void;
11    public function WyswietlFomularzEdycji(): void;
12 }
13
14 class WyswietlaniePilkarzy extends ZarzadzaniePilkarzami implements iWyswietlaniePilkarzy
15 {
16     public function WyswietlPilkarzy(mixed $Naglowek, mixed $SzablonZawodnik): void
17     {
18         $Naglowek("ZAWODNICY ({$this->LiczbaPilkarzy()})");
19
20         $wynik = $this->Zapytanie(
21             ZapytaniaSql::select_Wyswietl()
22         );
23
24         if ($wynik->num_rows > 0)
25             while ($wiersz = $wynik->fetch_assoc()) $SzablonZawodnik($wiersz);
26         else $Naglowek("Brak ipikarzy");
27     }
28
29     public function WyswietlFiltrowanychPilkarzy(mixed $Naglowek, $SzablonZawodnik): void
30     {
31         $sql = ZapytaniaSql::Select_Filtruj();
32
33         $wynik = $this->Zapytanie($sql);
34
35         if ($wynik->num_rows > 0)
36             while ($wiersz = $wynik->fetch_assoc()) $SzablonZawodnik($wiersz);
37         else
38             $Naglowek("BRAK");
39     }
40
41
42     public function WyswietlFormularzDodawania(): void
43     {
44         $pusty_formularz = (array)
45         $pusty_formularz = $this->tablicaPost;
46
47         $this->Formularz->FormularzDodawania($pusty_formularz, [$this, 'pobierzDane']);
48     }
49
50     public function WyswietlFomularzEdycji(): void
51     {
52         $wynik = $this->Zapytanie(
53             ZapytaniaSql::select_Edytuj($this->id)
54         );
55
56         while ($wiersz = (array) $wynik->fetch_assoc())
57             $this->Formularz->FormularzZapisu($wiersz, $this->id, [$this, 'pobierzDane']);
58     }
59 }
60 }
```

3.3.7 PobieraczObrazowWikipedia

Klasa za pomocą otwartego API Wikipedia, pobiera adres URL do głównego zdjęcia piłkarza z artykułu na Wikipedii. Zdjęcie jest pobierane podczas dodawania nowego piłkarza bądź jego edycji i zapisywana w bazie danych w tabeli **awatar** w postaci linku.

Dostępne metody pełnią następującą funkcjonalność:

- **pobierzDaneObrazu()** - pobiera jako format JSON dane z adresu <https://pl.wikipedia.org/w/api.php?>

action=query&prop=pageimages&format=json&piprop=original&titles=imie_nazwisko

- **pobierzZrodlo()** - pobiera z tego formatu JSON, link do obrazu
- **updateObrazka()** - w przypadku braku zwraca informacje na temat ścieżki domyślnego obrazu anonimowego użytkownika lub linku do grafiki z serwisu wikipedia
- **wyswietlObraz()** - zwraca tag img, który następnie może zostać wyświetlony.

```
1 <?php
2 namespace Pilkanozna\Models;
3
4 class PobieraczObrazowWikipedia
5 {
6     const API_URL =
7         "https://pl.wikipedia.org/w/api.php?action=query&prop=pageimages&format=json&piprop=original&titles=";
8     const ANON_AVATAR = "public/user.png";
9
10    private string $szukaneHaslo;
11    private string | array $daneJson;
12
13    public function __construct($imie, $nazwisko) {
14        $this->szukaneHaslo = urlencode("{ $imie }_ { $nazwisko }");
15    }
16
17    public function pobierzDaneObrazu(): void {
18        $daneWikipedia = file_get_contents(self::API_URL . $this->szukaneHaslo);
19        $this->daneJson = json_decode($daneWikipedia, true);
20    }
21
22    public function pobierzZrodlo(): mixed {
23        if (isset($this->daneJson['query']['pages'])) {
24            $dane = $this->daneJson;
25            $pierwszyKluczStrony = key($dane['query']['pages']);
26            if (isset($dane['query']['pages'][$pierwszyKluczStrony]['original']['source'])) {
27                return $dane['query']['pages'][$pierwszyKluczStrony]['original']['source'];
28            }
29        }
30        return self::ANON_AVATAR;
31    }
32
33    public function updateObrazka(): mixed {
34        $this->pobierzDaneObrazu();
35        return $this->pobierzZrodlo();
36    }
37
38    public function wyswietlObraz(): string {
39        $zrodloObrazu = $this->pobierzZrodlo();
40        return <<<HTML
41
42            <img
43                data-src=' $zrodloObrazu '
44                class='lazyload'
45                width="150px"
46                style="border-radius: 20px;"
47            />
48
49        HTML;
50    }
51 }
52 }
```

3.4 Controllers

3.4.1 Autoryzacja

Obsługuje mechanizm uwierzytelniania użytkowników. Wykorzystuje wbudowany mechanizm sesji w PHP do uwierzytelniania, zapewniając kontrolę dostępu i identyfikację użytkowników w aplikacji.

```
1 <?php
2 namespace Pilkanozna\Controllers;
3
4 include_once './KonfiguracjaApp.php';
5
6
7 interface iAutoryzacja
8 {
9     public function SprawdzCzyZalogowano(): void;
10    public function SprobojZalogowac(): void;
11    public function Wyloguj(): void;
12 }
13
14
15 class Autoryzacja implements iAutoryzacja
16 {
17     private string $login = UZYTKOWNIKADMIN;
18     private string $haslo = HASLOADMIN;
19
20     public function __construct() {
21         session_start();
22         $this->cssDlaNiezalogowanych();
23     }
24
25
26     public function SprawdzCzyZalogowano(): void
27     {
28         if (!isset($_SESSION['logged_in']) || $_SESSION['logged_in'] !== true) {
29             header('Location: /zaloguj');
30             exit();
31         }
32     }
33
34     public function SprobojZalogowac(): void
35     {
36
37         if ($_SERVER['REQUEST_METHOD'] === 'POST')
38         {
39
40             $uzytkownik = $this->login;
41             $haslo = $this->haslo;
42
43             if (isset($_POST['uzytkownik']) && isset($_POST['haslo']))
44             {
45                 if ($_POST['uzytkownik'] === $uzytkownik && $_POST['haslo'] === $haslo)
46                 {
47
48                     $_SESSION['logged_in'] = true;
49                     header('Location: /');
50                     exit();
51
52                 }
53
54                 else{
55                     echo "<i style='color: red;'>Niepoprawne dane logowania!</i> ";
56
57                 }
58             }
59         }
60
61     }
62
63     public function Wyloguj(): void
```

```
64 {
65     unset($_SESSION['logged_in']);
66     header('Location: /zaloguj');
67     exit();
68 }
69
70
71
72 private function cssDlaNiezalogowanych(): void
73 {
74     if (!isset($_SESSION['logged_in']) || $_SESSION['logged_in'] !== true) {
75         echo <<<HTML
76         <style>
77         .btn_admin, .card .fakeBtn{
78             display: none;
79         }
80         </style>
81         <script>
82             const btn_text = document.querySelector(".btn-login span");
83             const btn_a = document.querySelector(".btn-login");
84
85             btn_text.textContent = "Zaloguj";
86             btn_a.href = "/zaloguj";
87
88         </script>
89         HTML;
90     }
91 }
92 }
93
94 ?>
```

3.4.2 FiltrowanieKontroler

Odpowiada za filtrowanie danych otrzymywanych od użytkownika. Jest odpowiedzialna za proces sprawdzania, walidacji i filtrowania danych wejściowych (takich jak parametry GET z adresu URL oraz dane POST z formularzy), aby zapewnić bezpieczeństwo i poprawność przetwarzanych informacji.

```
1 <?php
2 namespace Pilkanozna\Controllers;
3 use Pilkanozna\Controllers\KontrolerDanych;
4
5 class FiltrowanieKontroler
6 {
7     protected object $Dane;
8     protected string $imie;
9     protected string $nazwisko;
10
11     protected string $sortowanie;
12
13     protected string $noga;
14     protected string $kraj;
15     protected string $numernakoszulce;
16     protected string $pozycja;
17
18
19     public function __construct() {
20         $this->Dane = new KontrolerDanych();
21         $this->imie = $this->Dane->getMetoda('imie');
22         $this->nazwisko = $this->Dane->getMetoda('nazwisko');
23         $this->sortowanie = $this->Dane->getMetoda('sortuj');
24         $this->noga = $this->Dane->getMetoda('wiodaca_noga');
25         $this->kraj = $this->Dane->getMetoda('fk_kraj');
26
27
28         $this->numernakoszulce = $this->Dane->getMetoda('fk_numernakoszulce');
29         $this->pozycja = $this->Dane->getMetoda('fk_pozycja');
```

```
31     }  
32 }
```

3.4.3 KontrolerDanych

Ta klasa ma na celu odbieranie danych od użytkownika z poziomu aplikacji, korzystając z metod GET (parametry z linku) i POST (dane przesyłane z formularzy). Zarządza odbiorem, przetwarzaniem i kontrolą danych, zapewniając ich poprawność i integrację z aplikacją.

```
1  <?php  
2  
3  namespace Pilkanozna\Controllers;  
4  
5  
6  interface IKontrolerDanych  
7  {  
8      public static function getMetoda(string $slowo): string; // POBIERA POLE Z FORMULARZA METODA GET  
9      public static function getID(): int; // POBIERA ID Z LINKU  
10     public function getPOST(string $nazwa): string; // POBIERA POLE Z FORMULARZA METODA POST  
11     public function setPOST(array $lista): array; // POBIERA I USTAWIA POLA Z FORMULARZA DO NOWEJ TABLICZY  
12  
13 }  
14  
15  
16 class KontrolerDanych implements IKontrolerDanych  
17 {  
18  
19     public static function getMetoda(string $slowo): string  
20     {  
21         return (isset($_GET[$slowo])) ? $_GET[$slowo] : "";  
22     }  
23  
24  
25     public static function getID(): int  
26     {  
27         return (isset($_GET['id'])) ? $_GET['id'] : 0;  
28     }  
29  
30  
31     public function getPOST(string $nazwa): string  
32     {  
33         return (isset($_POST[$nazwa])) ? $_POST[$nazwa] : "";  
34     }  
35  
36  
37     public function setPOST(array $lista): array  
38     {  
39         $setPOST = array();  
40  
41         foreach($lista as $element)  
42             $setPOST[$element] = $this->getPOST($element);  
43  
44         return $setPOST;  
45     }  
46  
47 }  
48 }
```

3.4.4 PilkarzPost

Zajmuje się operacjami na danych związanych z piłkarzami, odbierając i przetwarzając informacje przesłane od użytkownika z formularzy POST. Realizuje operacje zapisu danych związanego z piłkarzami do systemu.

```
1 <?php
2
3 namespace Pilkanozna\Controllers;
4 use Pilkanozna\Models\PobieraczObrazowWikipedia;
5 use Pilkanozna\Controllers\KontrolerDanych;
6
7 class PilkarzPost
8 {
9     {
10         private int $id;
11         private string $imie;
12         private string $nazwisko;
13         private string $awatar;
14
15         private object $Dane;
16         private object $Wikipedia;
17
18         public function __construct() {
19             $this->Dane = new KontrolerDanych();
20
21             $this->id = $this->Dane->getID();
22             $this->imie = $this->Dane->getPost("imie");
23             $this->nazwisko = $this->Dane->getPost("nazwisko");
24
25             $this->Wikipedia = new PobieraczObrazowWikipedia($this->imie, $this->nazwisko);
26             $this->awatar = $this->Wikipedia->updateObrazka();
27         }
28
29         public function getId(): int { return $this->id; }
30
31         public function getImie(): string { return $this->imie; }
32
33         public function getNazwisko(): string { return $this->nazwisko; }
34
35         public function getAwatar(): mixed { return $this->awatar; }
36
37         public function getTablicaPOST($lista): mixed
38         {
39             $tablica = $this->Dane->setPOST($lista);
40             return $tablica;
41         }
42     }
43 }
44 }
```

3.4.5 ZarządzaniePiłkarzami

Jest klasą, która wykorzystuje klasę pomocniczą BazaDanychHelper do operacji na bazie danych w kontekście informacji o piłkarzach. Ta klasa definiuje podstawowe funkcje i metody obsługi informacji o piłkarzach, wykorzystując metody HTTP - GET i POST do pobierania oraz przesyłania danych.

```
1 <?php
2
3 namespace Pilkanozna\Controllers;
4
5 use Pilkanozna\Models\ZapytaniaSql;
6 use Pilkanozna\Helpers\BazaDanychHelper;
7 use Pilkanozna\Controllers\PilkarzPost;
8 use Pilkanozna\Helpers\FormularzHelper;
9
10
11 class ZarzadzaniePilkarzami extends BazaDanychHelper
12 {
13     protected object $Pilkarz;
14     protected object $Formularz;
15
16     protected string $id;
```

```
17     protected string $imie;  
18     protected string $nazwisko;  
19     protected string $awatar;  
20     protected array $tablicaPost;  
21  
22  
23     public function __construct() {  
24         $this->Pilkarz = new PilkarzPost();  
25         $this->Formularz = new FormularzHelper();  
26  
27         $this->id = $this->Pilkarz->getId();  
28         $this->imie = $this->Pilkarz->getImie();  
29         $this->nazwisko = $this->Pilkarz->getNazwisko();  
30         $this->awatar = $this->Pilkarz->getAwatar();  
31         $this->tablicaPost = $this->Pilkarz->getTablicaPOST(ZapytaniaSql::getWszytkieKolumnyPilkarz());  
32     }  
33 }
```

3.5 Views

Views (pol. Widoki), ta sekcja zawiera metody, które reprezentują warstwę graficzną aplikacji w HTML, CSS i JavaScript. Odpowiada za poprawne wyświetlanie informacji, które zostały uzyskane z bazy danych.

3.5.1 StronaHtml

Zawiera szkielet strony HTML, taki jak sekcja <head> <body> <footer> czy <header>. Dzięki temu rozwiązaniu można ustawić Podstawowe dane takie jak tytuł strony, autorów, w kilku miejscach na stronie, wczytywana z pliku konfiguracyjnego **KonfiguracjaApp.php**

```
1  <?php  
2  namespace Pilkanozna\Views;  
3  
4  include_once './KonfiguracjaApp.php';  
5  
6  interface IStronaHtml  
7  {  
8      public function Head(): void;  
9      public function Header(): void;  
10     public function Footer(): void;  
11 }  
12  
13  
14 class StronaHtml implements IStronaHtml  
15 {  
16  
17     private string $tytul = TYTUL;  
18     private string $autorzy = AUTORZY;  
19     private string $rok = ROK;  
20     private string $przedmiot = PRZEDMIOT;  
21     private string $uczelnia = UCZLENIA;  
22  
23     public function Head(): void  
24     {  
25         echo <<<HTML  
26  
27         <html lang="pl">  
28         <head>  
29             <meta charset="UTF-8">  
30             <meta http-equiv="X-UA-Compatible" content="IE=edge">  
31             <meta name="viewport" content="width=device-width, initial-scale=1.0">  
32             <meta name="author" content="Adrian">  
33             <meta name="copyright" content="Adrian">  
34             <meta name="description" content="Prosta aplikacja OOP w języku PHP">  
35             <meta name="keywords" content="php,oop,mysql">  
36     }
```

```
37     <title> {$this->tytul} </title>
38
39     <link rel="stylesheet" href="public/style.css">
40
41
42     <!-- IKONKI -->
43     <link rel="icon" href="public/logo.webp">
44     <link rel="stylesheet"
45         href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
46     <script
47         src="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/js/fontawesome.min.js"></script>
48     <!-- IKONI::KONIEC -->
49
50 </head>
51 <body>
52
53 HTML;
54 }
55
56 public function Header(): void
57 {
58     echo <<<HTML
59     <header>
60         <div class="alert alert__nazwa">
61             <a href="/">
62                 
67             <a href="/">{$this->tytul} </a>
68         </div>
69
70         <div class="menu alert">
71
72             <div class="menu__item">
73                 <a class="fakeBtn" href="/">
74                     <i class="fa-solid fa-house"></i>
75                     <span>Strona główna</span>
76                 </a>
77             </div>
78
79             <div class="menu__item">
80                 <a class="fakeBtn btn_admin" href="/formularz_dodaj">
81                     <i class="fa-solid fa-user-plus"></i>
82                     <span>Dodaj</span>
83                 </a>
84             </div>
85
86             <div class="menu__item">
87                 <a class="fakeBtn btn_login" href="/wyloguj">
88                     <i class="fas fa-sign-out-alt"></i>
89                     <span>Wyloguj</span>
90                 </a>
91             </div>
92
93             <div class="menu__item">
94                 <a href="/szukaj?imie=&nazwisko=" class="gold">
95                     <i class="fa-solid fa-filter"></i>
96                     Filtry
97                 </a>
98             </div>
99
100         </div>
101     </header>
102     <main>
103
104
105 HTML;
```

```
107 }
108
109
110 private function LazyLoadigJS(): string
111 {
112
113     return <<<HTML
114         <script>
115             // Tworzenie obserwatora dla przewijania strony
116             let obserwator = new IntersectionObserver((wpisy, obserwator) => {
117                 wpisy.forEach(function (wpis) {
118                     if (wpis.intersectionRatio > 0 || wpis.isIntersecting) {
119                         const obraz = wpis.target;
120                         obserwator.unobserve(obraz);
121
122                         // Sprawdzenie, czy obraz już ma atrybut 'src'
123                         if (obraz.hasAttribute('src')) {
124                             return;
125                         }
126
127                         // Pobranie adresu źródłowego z atrybutu 'data-src'
128                         const adresZrodlowy = obraz.getAttribute('data-src');
129                         obraz.setAttribute('src', adresZrodlowy);
130
131                         obraz.onload = () => {
132                             // Dodatkowe działania, które można wykonać po załadowaniu obrazu
133                         }
134
135                         obserwator.unobserve(obraz);
136                     }
137                 });
138             });
139
140             // Obserwowanie wszystkich elementów z klasą 'lazyload'
141             document.querySelectorAll('.lazyload').forEach((element) => {
142                 obserwator.observe(element);
143             });
144         </script>
145
146     HTML;
147 }
148
149
150
151 public function Footer(): void
152 {
153
154     $js = $this->LazyLoadigJS();
155
156     echo <<<HTML
157         </main>
158         <footer class='footer'>
159             &COPY; {$this->rok} {$this->autorzy} <br>
160             {$this->przedmiot} <br>
161             {$this->uczelnia}
162
163         </footer>
164         $js
165     </body>
166     </html>
167     HTML;
168 }
169
170 }
```


3.5.2 SzablonHtml

Klasa zawiera komponenty HTML, które służą do wyświetlania elementów, takich jak card (karta z informacją o piłkarzu), formularze, panel logowania.

```
1  <?php
2
3  namespace Pilkanozna\Views;
4
5  use Pilkanozna\Models\PobieraczObrazowWikipedia;
6
7  interface ISzablonHtml
8  {
9      public function Zawodnik(array $wiersz): void;
10     public function Naglowek(string $napis): void;
11     public function PotwierdzUsuniecie($id,$imie,$nazwisko): void;
12     public static function Formularz(array $wiersz, string $adres, $kraje, $numernakoszulce, $pozycja,
        $napisprzycisk): void;
13     public static function FormularzLogowania(): void;
14     public static function FormularzFilrowaniaJs(): void;
15     public static function FormularzFiltrowania($kraje,$numernakoszulce,$pozycja): void;
16 }
17
18
19 class SzablonHtml implements ISzablonHtml
20 {
21
22     public function Zawodnik(array $wiersz): void
23     {
24
25         $id = $wiersz['id'];
26         $awatar = $wiersz['link'];
27
28         echo <<<HTML
29         <div class='card'>
30             <ul>
31                 <div class='card__avatar'>
32                     
33                 </div>
34                 <h2>{$wiersz['nazwisko']}</h2>
35                 <h3>{$wiersz['imie']}</h3>
36                 <div class='fakeBtn_wrapper'>
37                     <a class="fakeBtn" href="/usun?id=$id" name='delete'>
38                         <i class="fa-solid fa-trash"></i>
39                         <span>nUsu</span>
40                     </a>
41                     <a class="fakeBtn" href="/edytuj?id=$id" name='edit'>
42                         <i class="fa-solid fa-pen-to-square"></i>
43                         <span>Edytuj</span>
44                     </a>
45                 </div>
46                 <li> wzrost: <span>{$wiersz['wzrost']} m</span></li>
47                 <li> data urodzenia: <span>{$wiersz['data_urodzenia']}</span></li>
48                 <li> awiodca noga: <span>{$wiersz['wiodaca_noga']}</span></li>
49                 <li> wartosc rynkowa: <span>{$wiersz['wartosc_rynkowa']}</span> (mln USD)</li>
50                 <li> ilosc strzelonych goli: <span>{$wiersz['ilosc_strzelonych_goli']}</span></li>
51                 <li> kraj: <span>{$wiersz['pilkarzkraj']}</span></li>
52                 <li> numer na koszulce: <span>{$wiersz['numer']}</span></li>
53                 <li> pozycja: <span>{$wiersz['pozycja']}</span></li>
54             </ul>
55         </div>
56         HTML;
57     }
58
59
60     public function Naglowek(string $napis): void
61     {
62         echo <<<HTML
63         <div class="alert alert__header" >$napis</div>
64         HTML;
65     }
```

```
66
67 public function PotwierdzUsuniecie($id,$imie,$nazwisko): void
68 {
69     echo <<<HTML
70         <div class="alert" >
71             <p>Czy na pewno jesteś pewien że chcesz usunąć zawodnika <b>$imie $nazwisko</b>?</p>
72             <a href="/usun?id=$id&potwierdzenie=tak">
73                 <button class="fakeBtn">
74                     <i class="fa-solid fa-check"></i>
75                     Tak
76                 </button>
77             </a>
78             <a href="/">
79                 <button class="fakeBtn">
80                     <i class="fa-solid fa-xmark"></i>
81                     Anuluj
82                 </button>
83             </a>
84         </div>
85     HTML;
86 }
87
88
89 public static function Formularz(array $wiersz, string $adres, $kraje, $numernakoszulce, $pozycja,
90     $napisprzycisk): void
91 {
92
93     $imie = $wiersz['imie'];
94     $nazwisko = $wiersz['nazwisko'];
95
96     $pobieraczObrazow = new PobieraczObrazowWikipedia($imie, $nazwisko);
97     $pobieraczObrazow->pobierzDaneObrazu();
98
99
100     $awatar = $pobieraczObrazow->wyswietlObraz();
101
102
103     $id = $wiersz['id'];
104     echo <<<HTML
105         <form action="$adres" method="POST">
106             <table>
107
108                 <tr>
109                     <td colspan="2">
110
111                         <center style="margin: 20px">
112                             $awatar
113                         </center>
114
115                     </td>
116                 </tr>
117                 <tr style="display: none;">
118                     <td><label>ID</label></td>
119                     <td><input type="text" value="$id" name="id"></td>
120                 </tr>
121                 <tr>
122                     <td><label>Imie</label></td>
123                     <td><input type="text" value="{ $wiersz['imie']}" name="imie" required></td>
124                 </tr>
125                 <tr>
126                     <td><label>Nazwisko</label></td>
127                     <td><input type="text" value="{ $wiersz['nazwisko']}" name="nazwisko" required></td>
128                 </tr>
129                 <tr>
130                     <td><label>Wzrost</label></td>
131                     <td><input type="number" value="{ $wiersz['wzrost']}" name="wzrost" step="0.01" min="0"
132                         placeholder="np. 1.77" required></td>
133                 </tr>
134                 <tr>
135                     <td><label>Data urodzenia</label></td>
136                     <td><input type="date" value="{ $wiersz['data_urodzenia']}" name="data_urodzenia">
```

Zastosowania programowania obiektowego (zajęcia projektowe)

```
201     </button>
202 </form>
203
204 <script>
205     document.querySelector(".menu").style.display = "none";
206 </script>
207 HTML;
208 }
209
210
211 public static function FormularzFilrowaniaJs(): void
212 {
213     echo <<<HTML
214         <script>
215             function setupCheckboxAndSelect(checkboxId, selectId) {
216                 // Pobierz checkbox i pole wyboru na podstawie przekazanych identyfikatorów
217                 var checkbox = document.getElementById(checkboxId);
218                 var select = document.getElementById(selectId);
219
220                 // Ustaw początkowy stan (domyślnie łąwyaczone)
221                 select.disabled = true;
222
223                 // Nasluchuj zdarzenia zmiany stanu checkboxa
224                 checkbox.addEventListener("change", function () {
225                     if (checkbox.checked) {
226                         select.disabled = false;
227                         checkbox.value="1";
228                     }
229                     else {
230                         select.disabled = true;
231                         checkbox.value="0";
232                     }
233                 });
234             }
235
236             function pobierzParametrGet(parametr) {
237                 const queryString = window.location.search.substr(1);
238
239                 const paryParametrow = queryString.split('&');
240
241                 for (const paraParametru of paryParametrow) {
242                     const [nazwa, wartosc] = paraParametru.split('=');
243                     if (nazwa === parametr) {
244                         return decodeURIComponent(wartosc);
245                     }
246                 }
247
248                 return null;
249             }
250
251             function setFormularzPole(getparametr, formularzid){
252
253                 var wartosc = pobierzParametrGet(getparametr);
254                 var pole = document.querySelector(formularzid);
255
256                 pole.value = wartosc;
257             }
258
259             function clickBox(id,id2){
260                 var box = document.getElementById(id);
261                 var wartosc = document.getElementById(id2);
262
263                 if(wartosc.value == "") console.log("puste");
264                 else box.click();
265             }
266
267             function policzElementyZKlasaCard() {
268                 var elementy = document.querySelectorAll('.card');
269             }
270
271
272 }
```

```
273         return elementy.length;
274     }
275
276
277     function WynikiWyszukiwania()
278     {
279         var imie = pobierzParametrGet("imie");
280         var nazwisko = pobierzParametrGet("nazwisko");
281
282         var ile = policzElementyZKlasaCard();
283
284
285         var pole = document.querySelector("#szukanypilkarz");
286         var wartosc = imie + " " + nazwisko + " (" + ile + ")";
287         if(imie==null && nazwisko==null) wartosc=" ";
288
289
290         pole.innerHTML = wartosc;
291
292     }
293
294
295
296     window.addEventListener("load", function () {
297         setupCheckboxAndSelect("noga_check", "noga");
298         setupCheckboxAndSelect("kraj_check", "fk_kraj");
299         setupCheckboxAndSelect("numernakoszulce_check", "fk_numernakoszulce");
300         setupCheckboxAndSelect("pozycja_check", "fk_pozycja");
301
302         setFormularzPole("sortuj", "#sortuj");
303         setFormularzPole("imie", "#imie");
304         setFormularzPole("nazwisko", "#nazwisko");
305
306         setFormularzPole("wiodaca_noga", "#noga");
307         clickBox("noga_check", "noga");
308
309         setFormularzPole("fk_kraj", "#fk_kraj");
310         clickBox("kraj_check", "fk_kraj");
311
312         setFormularzPole("fk_pozycja", "#fk_pozycja");
313         clickBox("pozycja_check", "fk_pozycja");
314
315         setFormularzPole("fk_numernakoszulce", "#fk_numernakoszulce");
316         clickBox("numernakoszulce_check", "fk_numernakoszulce");
317
318         WynikiWyszukiwania()
319
320
321     });
322 }
323 </script>
324
325 HTML;
326
327 }
328
329 public static function FormularzFiltrowania($kraje,$numernakoszulce,$pozycja): void
330 {
331     echo <<<HTML
332
333     <form action="/szukaj" method="GET">
334         <table>
335
336             <tr>
337                 <td>
338                     <label>Imie</label>
339                 </td>
340                 <td>
341                     <input type="text" name="imie" id="imie">
342                 </td>
343             </tr>
344
```

```
345 <tr>
346 <td>
347 <label>Nazwisko</label>
348 </td>
349 <td>
350 <input type="text" name="nazwisko" id="nazwisko">
351 </td>
352 </tr>
353
354 <tr>
355 <td><label>Sortuj</label></td>
356 <td>
357 <select name="sortuj" id="sortuj">
358 <option value="najnowsze">Najnowsze wpisy</option>
359 <option value="najstarsze">Najstarsze wpisy</option>
360
361 <option value="a-z">Alfabetycznie A-Z (Nazwisko)</option>
362 <option value="z-a">Alfabetycznie Z-A (Nazwisko)</option>
363
364 <option value="wzrost-asc">♣Rosnco: Wzrost</option>
365 <option value="wzrost-desc">♣Malejco: Wzrost</option>
366
367 <option value="dataurodzania-asc">♣Rosnco: Data urodzenia</option>
368 <option value="dataurodzania-desc">♣Malejco: Data urodzenia</option>
369
370 <option value="wartosc-asc">♣Malejco: śWartoc rynkowa</option>
371 <option value="wartosc-desc">♣Malejco: śWartoc rynkowa</option>
372
373 </select>
374 </td>
375 </tr>
376
377
378 <tr>
379 <td>
380 <input type="checkbox" id="noga_check" value="1">
381 <label>♣Wiodca noga</label>
382 </td>
383 <td>
384 <select name="wiodaca_noga" id="noga" disabled>
385 <option value="LEWA">LEWA</option>
386 <option value="PRAWA">PRAWA</option>
387 <option value="OBU-ŻNONY">OBU-ŻNONY</option>
388 </select>
389 </td>
390 </tr>
391
392 <tr>
393 <td>
394 <input type="checkbox" id="kraj_check" value="1">
395 <label for="kraj">Wybierz kraj:</label>
396 </td>
397 <td>$kraje</td>
398 </tr>
399
400 <tr>
401 <td>
402 <input type="checkbox" id="numernakoszulce_check" value="1">
403 <label for="numrnakoszulce">Numer na koszulce</label>
404 </td>
405 <td>$numernakoszulce</td>
406 </tr>
407
408 <tr>
409 <td>
410 <input type="checkbox" id="pozycja_check" value="1">
411 <label for="pozycja">Pozycja</label>
412 </td>
413 <td>$pozycja</td>
414 </tr>
415
416 <tr>
417 <td><br>
418 <button class="fakeBtn">
419 <i class="fa-solid fa-magnifying-glass"></i>
420 <span>Szukaj</span>
421 </button>
422 </td>
423 <td></td>
424 </tr>
```

```
417         </button>
418     </td>
419 </tr>
420 </table>
421 </form>
422 HTML;
423 }
424
425
426
427 }
```

3.6 Helpers

3.6.1 BazaDanychHelper

Służy do udostępniania metod wspomagających operacje bazodanowe, takie jak łączenie się z bazą danych, wykonywanie zapytań czy inne operacje związane z bazą danych.

```
1  <?php
2
3  namespace Pilkanozna\Helpers;
4
5  use Pilkanozna\Models\BazaDanych;
6  use Pilkanozna\Models\ZapytaniaSql;
7
8
9  class BazaDanychHelper extends BazaDanych
10 {
11
12     public function __construct()
13     {
14         $this->Polaczenie();
15     }
16
17     public function __destruct()
18     {
19         $this->Rozlaczenie();
20     }
21
22
23     public function pobierzDane($sql, $idkolumny, $nazwakolumny): array
24     {
25         $wynik = $this->Zapytanie($sql);
26         $dane = [];
27
28         while ($wiersz = $wynik->fetch_assoc()) {
29             $dane[] = [
30                 $idkolumny => $wiersz[$idkolumny],
31                 $nazwakolumny => $wiersz[$nazwakolumny],
32             ];
33         }
34
35         return $dane;
36     }
37
38     public function LiczbaPilkarzy(): int
39     {
40         $wynik = $this->Zapytanie(ZapytaniaSql::liczbaZawodnikow());
41         $wiersz = $wynik->fetch_assoc();
42         return $wiersz['liczba_pilkarzy'];
43     }
44
45     public function OstatniPilkarz(): int
46     {
47         $wynik = $this->Zapytanie(ZapytaniaSql::select_ostaniZawodnik());
48         $wiersz = $wynik->fetch_array();
```

```
49     return $wiersz[0];
50 }
51
52 }
```

3.6.2 FormularzHelper

Zawiera metody pomocnicze związane z tworzeniem, walidacją czy obsługą formularzy w aplikacji.

```
1  <?php
2  namespace Pilkanozna\Helpers;
3
4
5  use Pilkanozna\Models\ZapytaniaSql;
6  use Pilkanozna\Views\SzablonHtml;
7
8
9  class FormularzHelper
10 {
11     private $kraje;
12     private $numery;
13     private $pozycje;
14
15
16     public function __construct() {
17         $this->kraje = ZapytaniaSql::select_Kraj();
18         $this->numery = ZapytaniaSql::select_Numernakoszulce();
19         $this->pozycje = ZapytaniaSql::select_Pozycja();
20     }
21
22
23     private function SelectHTML(array $dane, $id, $nazwa, $fk): string
24     {
25         $html = "<select name='$fk' id='$fk' >";
26         foreach ($dane as $wiersz) {
27             $html .= "<option value='{ $wiersz[$id]} '>{ $wiersz[$nazwa]}</option>";
28         }
29         $html .= "</select>";
30
31         return $html;
32     }
33
34     public function Pilkarz($dane, $adres, $napisprzycisk, $pobierzDane): void
35     {
36         $kraje = $pobierzDane($this->kraje, 'pk_kraj', 'nazwa');
37         $numery = $pobierzDane($this->numery, 'pk_numernakoszulce', 'numer');
38         $pozycje = $pobierzDane($this->pozycje, 'pk_pozycja', 'nazwa');
39
40         $select_kraje_html = $this->SelectHTML($kraje, 'pk_kraj', 'nazwa', 'fk_kraj');
41         $select_numernakoszulce_html = $this->SelectHTML($numery, 'pk_numernakoszulce', 'numer',
42             'fk_numernakoszulce');
43         $select_pozycja_html = $this->SelectHTML($pozycje, 'pk_pozycja', 'nazwa', 'fk_pozycja');
44
45         SzablonHtml::Formularz(
46             $dane,
47             $adres,
48             $select_kraje_html,
49             $select_numernakoszulce_html,
50             $select_pozycja_html,
51             $napisprzycisk
52         );
53     }
54
55     public function Filtrowanie($pobierzDane): void
56     {
57         $kraje = $pobierzDane($this->kraje, 'pk_kraj', 'nazwa');
58         $numery = $pobierzDane($this->numery, 'pk_numernakoszulce', 'numer');
59         $pozycje = $pobierzDane($this->pozycje, 'pk_pozycja', 'nazwa');
```



```
60
61 $select_kraje_html = $this->SelectHTML($kraje, 'pk_kraj', 'nazwa', 'fk_kraj');
62 $select_numernakoszulce_html = $this->SelectHTML($numery, 'pk_numernakoszulce', 'numer',
    'fk_numernakoszulce');
63 $select_pozycja_html = $this->SelectHTML($pozycje, 'pk_pozycja', 'nazwa', 'fk_pozycja');
64
65 SzablonHtml::FormularzFiltrowania(
66     $select_kraje_html,
67     $select_numernakoszulce_html,
68     $select_pozycja_html,
69 );
70
71 SzablonHtml::FormularzFilrowaniaJs();
72
73 }
74
75
76 public function FormularzZapisu($wiersz, $id, $pobierzDane): void
77 {
78     $this->Pilkarz($wiersz, "/zapisz?id=$id", "Zapisz", $pobierzDane);
79 }
80
81 public function FormularzDodawania($wiersz, $pobierzDane): void
82 {
83     $this->Pilkarz($wiersz, "/dodaj", "Dodaj", $pobierzDane);
84 }
85 }
86 }
```

3.7 Projekt

Klasa ta jest ostatecznym elementem programu, posiadającym metodę *Uruchom()*, która integruje Routing, StronęHtml oraz klasę Aplikacja w celu właściwego uruchomienia programu z zachowaniem odpowiedniej kolejności działań. Metoda *Uruchom()* łączy elementy w logiczną sekwencję, umożliwiając kompleksowe działanie aplikacji poprzez współpracę wymienionych składników w spójny sposób.

```
1 <?php
2 // klasa do uruchamiania projektu
3 namespace Pilkanozna;
4
5 use Pilkanozna\Controllers\KontrolerStrony;
6 use Pilkanozna\Views\StronaHtml;
7
8 final class Projekt
9 {
10     public static function Uruchom(): void
11     {
12         $Aplikacja = new KontrolerStrony;
13         $Strona = new StronaHtml;
14
15         $Strona->Head();
16         $Strona->Header();
17         $Aplikacja->Routing();
18         $Strona->Footer();
19     }
20 }
21 }
```

3.8 FileLoader

Klasa ma jedynie jedno przeznaczenie: załączenie plików z klasami do projektu. Jej rola polega na importowaniu (łączeniu) różnych plików zawierających klasy, co umożliwia poprawne funkcjonowanie projektu poprzez dostęp do niezbędnych klas i ich metod.

```
1 <?php
2
3 class Fl
4 {
5     public static function Views($name) {
6         require_once "../classes/Views/$name.php";
7     }
8
9
10    public static function Models($name) {
11        require_once "../classes/Models/$name.php";
12    }
13
14
15    public static function Controllers($name) {
16        require_once "../classes/Controllers/$name.php";
17    }
18
19    public static function Helpers($name) {
20        require_once "../classes/Helpers/$name.php";
21    }
22 }
```

3.9 plik - index.php

Ten plik inicjuje metodę *Uruchom()* oraz wykorzystuje klasę FileLoader (skrót Fl) do załączenia plików. Jego głównym zadaniem jest uruchomienie metody *Uruchom()* oraz korzystanie z klasy FileLoader w celu załadowania plików do projektu.

```
1 <?php
2 use Pilkanozna\Projekt;
3
4 include_once '../classes/FileLoader.php';
5 include_once '../classes/Projekt.php';
6
7 // kolejnosc ladowania klas - nie zmieniac
8
9 Fl::Controllers("PilkarzPost");
10 Fl::Models("BazaDanych");
11 Fl::Helpers("BazaDanychHelper");
12 Fl::Controllers("FiltrowanieKontroler");
13 Fl::Models("FiltrowanieSql");
14 Fl::Models("ZapytaniaSql");
15 Fl::Helpers("FormularzHelper");
16
17 Fl::Controllers("ZarządzaniePilkarzami");
18 Fl::Models("WyswietlaniePilkarzy");
19 Fl::Models("OperacjePilkarzy");
20
21 Fl::Models("Aplikacja");
22
23 Fl::Controllers("KontrolerDanych");
24 Fl::Controllers("Autoryzacja");
25 Fl::Controllers("KontrolerStrony");
26
27 Fl::Models("PobieraczObrazowWikipedia");
28 Fl::Views("SzablonHtml");
29 Fl::Views("StronaHtml");
30
31
32 Projekt::Uruchom();
```

4 Projekt Bazy Danych

4.1 Relacyjna baza danych SQL

Struktura oparta jest na modelu relacyjnym. Serwer SQL jest platformą, która obsługuje taką bazę danych, umożliwiając przechowywanie, zarządzanie oraz wykonywanie zapytań do tych danych.

Modele relacyjne baz danych korzystają z relacji między tabelami, wykorzystując klucze główne i obce do ustanawiania powiązań. Dzięki językowi zapytań SQL (Structured Query Language) serwer SQL może zarządzać danymi, umożliwiając operacje takie jak wstawianie, odczyt, aktualizację i usuwanie informacji z tych tabel.

Serwer SQL zapewnia efektywne zarządzanie bazą danych, a także zabezpieczenia, optymalizację oraz skalowalność. To narzędzie umożliwia obsługę wielu użytkowników jednocześnie, kontrolę nad dostępem do danych oraz wykonywanie złożonych operacji na danych przechowywanych w bazie relacyjnej

4.2 Model bazy danych - Opis

```
1 DROP DATABASE IF EXISTS pilkanozna;
2 CREATE DATABASE pilkanozna;
3 USE pilkanozna;
4
5 CREATE TABLE pilkarz (
6     pk_pilkarz int primary key auto_increment,
7     imie varchar(30) not null,
8     nazwisko varchar(30) not null,
9     wzrost float not null,
10    data_urodzenia date not null,
11    wiodaca_noga enum('LEWA', 'PRAWA', 'OBU-NOZNY'),
12    wartosc_rynkowa int not null,
13    ilosc_strzelonych_goli int not null,
14    fk_kraj int not null,
15    fk_numernakoszulce int not null,
16    fk_pozycja int not null
17 );
18
19 CREATE TABLE awatar (
20     pk_awatar int primary key auto_increment,
21     link varchar(200) NOT NULL,
22     fk_pilkarz int NOT NULL
23 )
24 ;
25
26 CREATE TABLE krajpilkarza (
27     pk_kraj int primary key auto_increment,
28     nazwa varchar(60) not null unique
29 );
30
31 CREATE TABLE numernakoszulce (
32     pk_numernakoszulce int primary key auto_increment,
33     numer int not null unique
34 );
35
36 CREATE TABLE pozycja (
37     pk_pozycja int primary key auto_increment,
38     nazwa varchar(30) not null unique
39 );
40
41 INSERT INTO pozycja VALUES
42 (1, 'Srodkowy napastnik'),
43 (2, 'Srodkowy obrońca'),
44 (3, 'Prawy napastnik'),
45 (4, 'Bramkarz'),
46 (5, 'Lewy napastnik'),
47 (6, 'Ofensywny pomocnik');
```

```
47 (7, 'Srodkowy pomocnik')
48 ;
49
50 INSERT INTO numernakoszulce
51 VALUES (1,7),(2,9),
52 (3,24),(4,15),(5,3),(6,99),(7,10),(8,30),
53 (9,18),(10,20),(11,19),(12,6),(13,33),
54 (14,5),(15,23);
55
56 INSERT INTO krajpilkarza
57 VALUES
58 (1, 'Portugalia'),
59 (2, 'Polska'),
60 (3, 'Hiszpania'),
61 (4, 'Dania'),
62 (5, 'Francja'),
63 (6, 'Wlochy'),
64 (7, 'Brazylia'),
65 (8, 'Argentyna'),
66 (9, 'Norwegia'),
67 (10, 'Niemcy'),
68 (11, 'Holandia'),
69 (12, 'Ukraina'),
70 (13, 'Anglia')
71 ;
```

Kod SQL definiuje strukturę bazy danych dla systemu zarządzania informacjami o piłkarzach.

1. Tworzenie bazy danych:

Kod rozpoczyna od próby usunięcia bazy danych **pilkanozna** (jeśli istnieje) i następnie tworzy nową bazę o tej samej nazwie. Następnie wybiera tę nowo utworzoną bazę jako aktywną dla dalszych operacji.

2. Definicje tabel:

- Tabela **piłkarz** przechowuje informacje o piłkarzach, takie jak imię, nazwisko, dane osobowe, umiejętności piłkarskie i pochodzenie.
- Tabela **awatar** zawiera linki do obrazów reprezentujących piłkarzy.
- Tabela **krajpilkarza** przechowuje informacje o krajach, do których należą piłkarze.
- Tabela **numernakoszulce** zawiera numery na koszulkach piłkarzy.
- Tabela **pozycja** zawiera różne pozycje, na których piłkarze mogą grać.

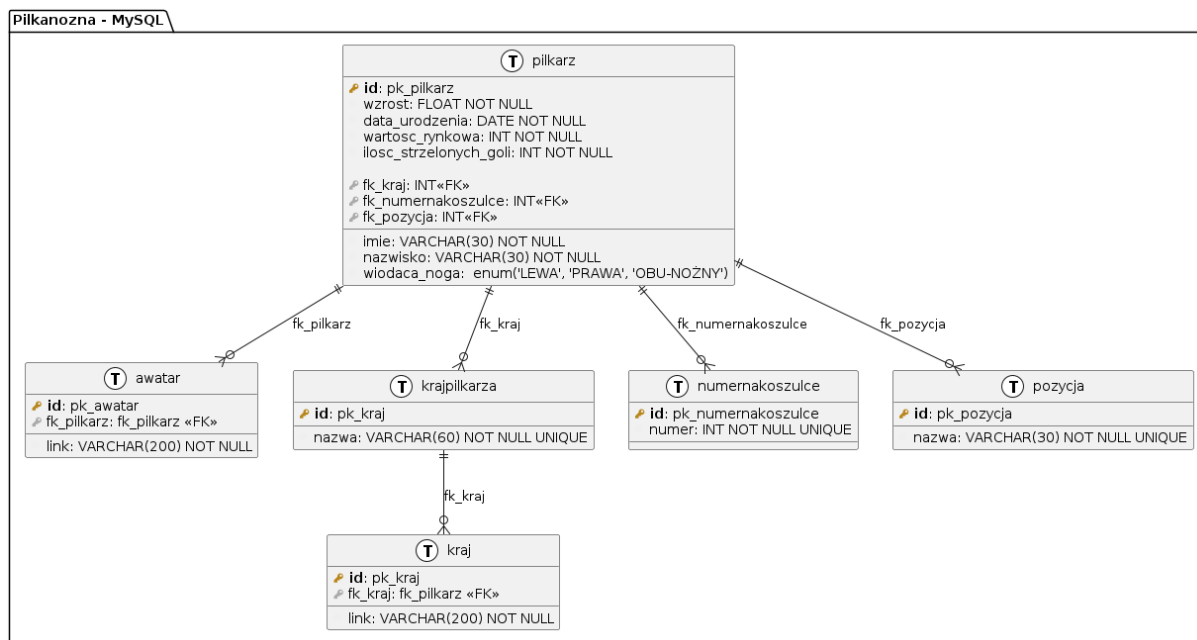
3. Wstawianie danych:

Dodaje przykładowe dane do tabeli **pozycja**, **numernakoszulce** i **krajpilkarza**.

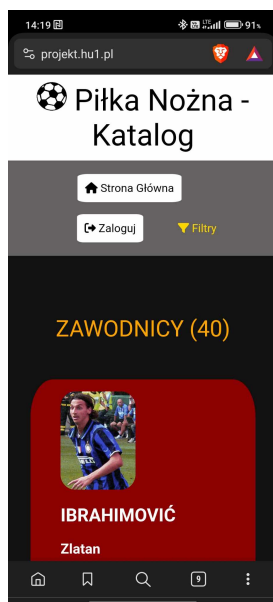
4.2.1 Ustawienie uprawnień dla użytkownika

```
1 CREATE USER "projekt"@"localhost" IDENTIFIED BY "Pracownia107!";
2 GRANT ALL PRIVILEGES ON pilkanozna.* TO "projekt"@"localhost";
```

4.3 Rysunek



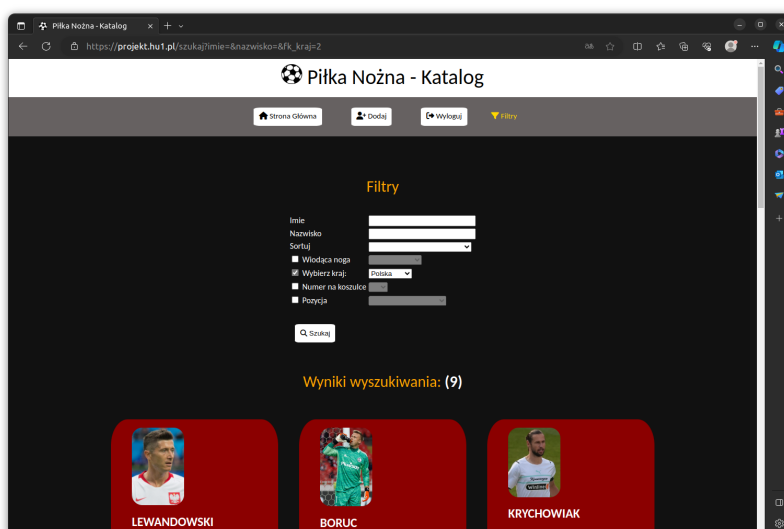
Rysunek 7: Diagram tabel bazy danych pilkanozna



Rysunek 9: Strona Główna - wersja mobilna, Android

5.2 Wyświetlanie oraz filtrowanie piłkarzy

- <https://projekt.hu1.pl/szukaj>



Rysunek 10: Panel filtrowania

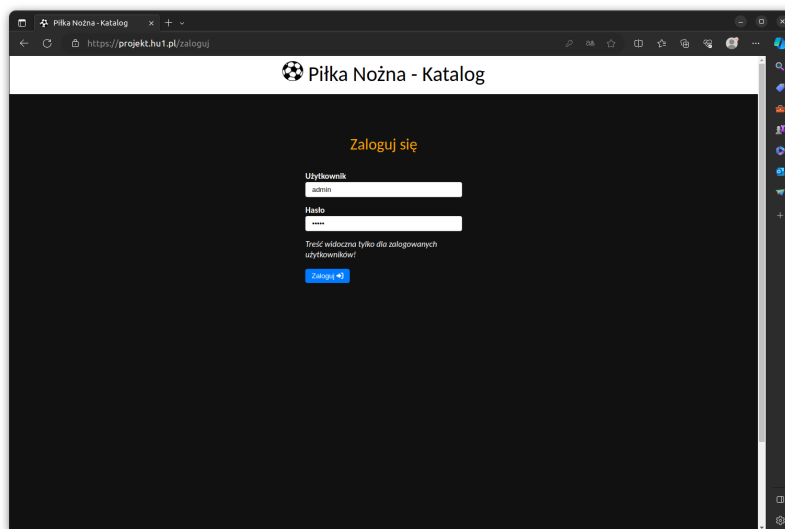
5.3 Panel Logowania

Panel logowania dostępny jest pod adresem:

- <https://projekt.hu1.pl/zaloguj>

Dane do logowania:

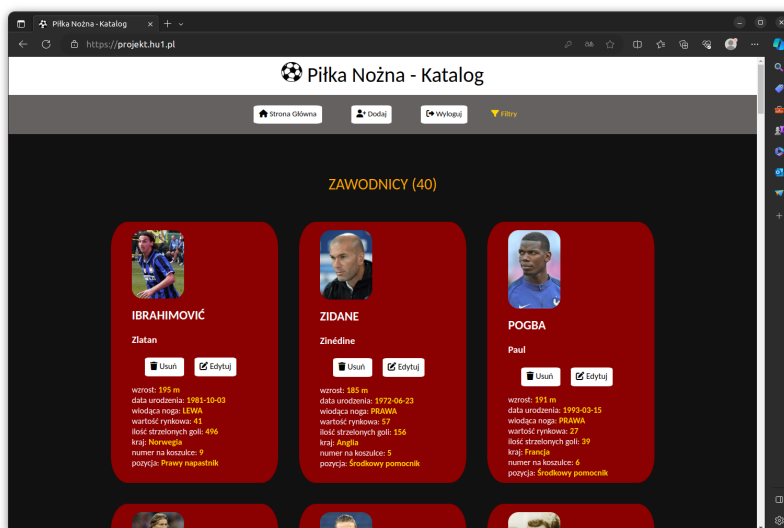
- Użytkownik: **admin**
- Hasło: **admin**



Rysunek 11: Panel logowania

5.4 Panel Administracyjny

Jako administrator użytkownik ma podniesione uprawnienia i dodatkową zakładkę *Dodaj* oraz przycisk *Edytuj* lub *Usuń* pod imieniem piłkarza.



Rysunek 12: Panel Administracyjny

5.5 Modyfikowanie danych na temat piłkarzy

5.5.1 Dodawanie piłkarza

- https://projekt.hu1.pl/formularz_dodaj

Zdjęcie piłkarza jest pobierane z serwisu Wikipedia na podstawie **Imienia** oraz **Nazwiska**, o ile zawodnik posiada tam swój artykuł wraz ze zdjęciem - co zazwyczaj ma miejsce w 90% przypadków. W sytuacji, gdy nie ma dostępnego zdjęcia piłkarza na stronie, automatycznie ustawiane jest domyślne zdjęcie anonimowego użytkownika.

The screenshot shows a web browser window with the title 'Piłka Nożna - Katalog'. The URL is 'https://projekt.hu1.pl/formularz_dodaj'. The page has a dark theme and a navigation bar with links: 'Strona Główna', 'Dodaj', 'Wyloguj', and 'Filtry'. The main content area is titled 'Dodawanie' and features a blue circular profile picture placeholder. Below the picture is a form with the following fields: 'Imię' (text input), 'Nazwisko' (text input), 'Wzrost' (text input with value 'np. 1.77'), 'Data urodzenia' (date picker with value 'nn/nn/yyyy'), 'Włosy' (dropdown menu with value 'Czarna'), 'Wartość rynkowa' (text input with value 'np. 25'), 'Ilość strzelonych goli' (text input with value 'np. 5'), 'Wybierz kraj' (dropdown menu with value 'Anglia'), 'Numer na koszulce' (text input with value '10'), and 'Pozycja' (dropdown menu with value 'Strzałowiec'). At the bottom of the form is a 'Dodaj' button.

Rysunek 13: Formularz, który umożliwia dodanie nowego Piłkarza

5.5.2 Usuwanie piłkarza

Przed usunięciem piłkarza pojawia się potwierdzenie. Można jeszcze anulować decyzję i zrezygnować z usunięcia lub potwierdzić.

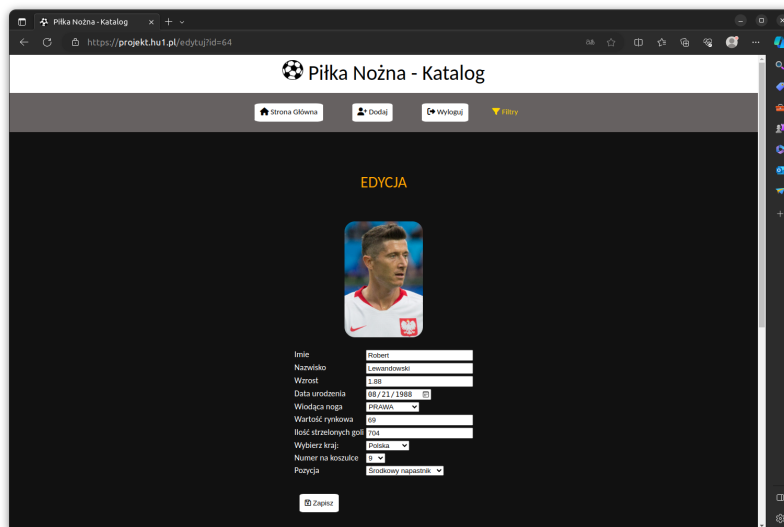
- np. <https://projekt.hu1.pl/usun?id=79>

The screenshot shows a web browser window with the title 'Piłka Nożna - Katalog'. The URL is 'https://projekt.hu1.pl/usun?id=126'. The page has a dark theme and a navigation bar with links: 'Strona Główna', 'Dodaj', 'Wyloguj', and 'Filtry'. The main content area displays a confirmation message in orange text: 'Czy na pewno jesteś pewien że chcesz usunąć zawodnika Zlatan Ibrahimović?'. Below the message are two buttons: '✓ Tak' and 'X Anuluj'.

Rysunek 14: Potwierdzenie usunięcia Piłkarza

5.5.3 Edycja piłkarza

- np. <https://projekt.hu1.pl/edytuj?id=79>



The screenshot shows a web browser window with the address <https://projekt.hu1.pl/edytuj?id=64>. The page title is 'Piłka Nożna - Katalog'. The main content area is titled 'EDYCJA' (Edit) and features a profile picture of a player. Below the picture is a form with the following fields:

Imię	Robert
Nazwisko	Lewandowski
Wzrost	1.88
Data urodzenia	08/21/1988
Wiedząca noga	Prawa
Wartość rynkowa	68
Rok aktywności	2018
Wybierz kraj	Polska
Numer na koszulce	11
Pozycja	Strzałowy napastnik

At the bottom of the form is a 'Zapisz' (Save) button.

Rysunek 15: Panel służący do edycji informacji na temat piłkarza

5.6 Jak uruchomić projekt lokalnie - instrukcja

5.6.1 Uruchomienie kodu - PHP

Aby lokalnie uruchomić projekt trzeba skorzystać z serwera deweloperskiego PHP

Polecenie:

```
1 php -S localhost:8080
```

```
> php -S localhost:8080
[Tue Nov 7 18:16:07 2023] PHP 8.1.2-1ubuntu2.14 Development Server (http://localhost:8080) started
[Tue Nov 7 18:16:13 2023] 127.0.0.1:55256 Accepted
[Tue Nov 7 18:16:13 2023] 127.0.0.1:55256 [200]: GET /
[Tue Nov 7 18:16:13 2023] 127.0.0.1:55256 Closing
[Tue Nov 7 18:16:13 2023] 127.0.0.1:59450 Accepted
[Tue Nov 7 18:16:13 2023] 127.0.0.1:59450 [200]: GET /public/style.css
[Tue Nov 7 18:16:13 2023] 127.0.0.1:59450 Closing
[Tue Nov 7 18:16:13 2023] 127.0.0.1:59454 Accepted
[Tue Nov 7 18:16:13 2023] 127.0.0.1:59454 [200]: GET /public/logo.webp
[Tue Nov 7 18:16:13 2023] 127.0.0.1:59454 Closing
[Tue Nov 7 18:16:13 2023] 127.0.0.1:59466 Accepted
[Tue Nov 7 18:16:13 2023] 127.0.0.1:59466 [200]: GET /public/user.png
[Tue Nov 7 18:16:13 2023] 127.0.0.1:59466 Closing
[Tue Nov 7 18:16:14 2023] 127.0.0.1:59478 Accepted
[Tue Nov 7 18:16:14 2023] 127.0.0.1:59478 [200]: GET /public/logo.webp
[Tue Nov 7 18:16:14 2023] 127.0.0.1:59478 Closing
```

Rysunek 16: Uruchomiony serwer deweloperski PHP w terminalu Ubuntu 22.04 LTS

```
PS C:\xampp\htdocs\Projekt> php -S localhost:8080
[Tue Nov 7 18:37:14 2023] PHP 8.2.4 Development Server (http://localhost:8080) started
[Tue Nov 7 18:37:17 2023] [::1]:51987 Accepted
[Tue Nov 7 18:37:17 2023] [::1]:51988 Accepted
[Tue Nov 7 18:37:19 2023] [::1]:51987 [200]: GET /
[Tue Nov 7 18:37:19 2023] [::1]:51987 Closing
[Tue Nov 7 18:37:19 2023] [::1]:51988 [200]: GET /public/style.css
[Tue Nov 7 18:37:19 2023] [::1]:51988 Closing
[Tue Nov 7 18:37:19 2023] [::1]:51992 Accepted
[Tue Nov 7 18:37:19 2023] [::1]:51992 [200]: GET /public/logo.webp
[Tue Nov 7 18:37:19 2023] [::1]:51992 Closing
[Tue Nov 7 18:37:19 2023] [::1]:51993 Accepted
[Tue Nov 7 18:37:19 2023] [::1]:51993 [200]: GET /public/user.png
[Tue Nov 7 18:37:19 2023] [::1]:51993 Closing
[Tue Nov 7 18:37:19 2023] [::1]:51994 Accepted
[Tue Nov 7 18:37:19 2023] [::1]:51994 [200]: GET /public/logo.webp
[Tue Nov 7 18:37:19 2023] [::1]:51994 Closing
```

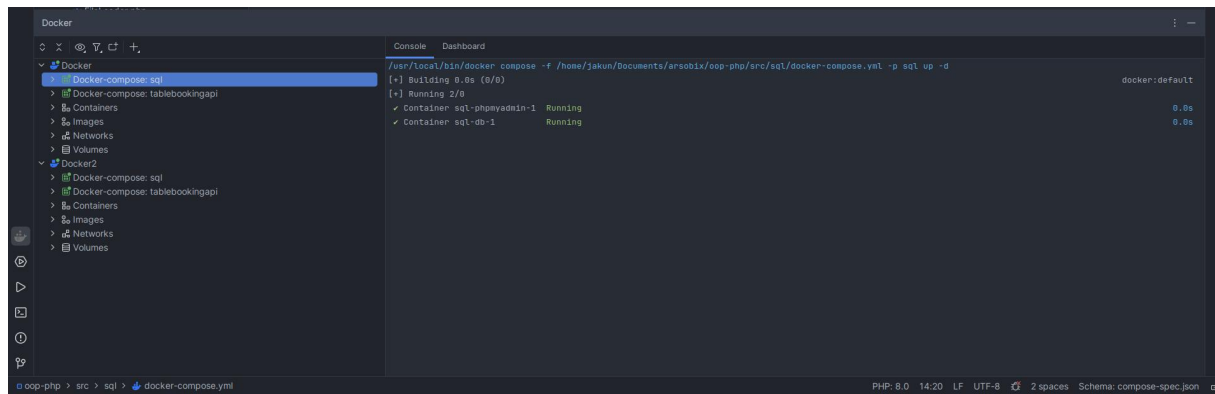
Rysunek 17: Uruchomiony serwer deweloperski PHP w PowerShell Windows 10

5.6.2 Uruchomienie kontenera z bazą danych - Docker (Ubuntu 22.04)

Instalacja docker na Ubuntu <https://docs.docker.com/desktop/install/ubuntu/>
docker-compose.yml - plik konfiguracyjny

```
1 version: '3.8'
2
3 services:
4
5   db:
6     image: mysql:latest
7     restart: always
8     environment:
9       MYSQL_ROOT_PASSWORD: xd
10      MYSQL_DATABASE: pilkanozna
11      MYSQL_USER: projekt
12      MYSQL_PASSWORD: Pracownia107!
13     ports:
14       - "3333:3306"
15   phpmyadmin:
16     image: phpmyadmin
17     restart: always
18     ports:
19       - 8099:80
20     environment:
21       - PMA_ARBITRARY=1
```

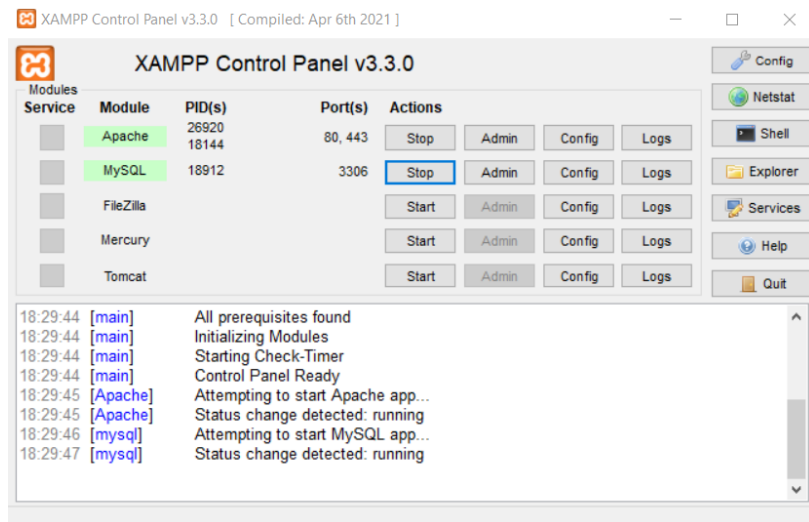
22 - PMA_HOST=db
23 - PMA_USER=root
24 - PMA_PASSWORD=xd
25 - PMA_PORTS=3307



Rysunek 18: Uruchomienie kontenera Docker z poziomu IDE PhpStorm

5.6.3 Uruchomienie bazy danych - XAMPP (Windows 10)

XAMPP(na Windows 10 służy do obsługi phpMyAdmin oraz uruchomienia serwera MySQL)

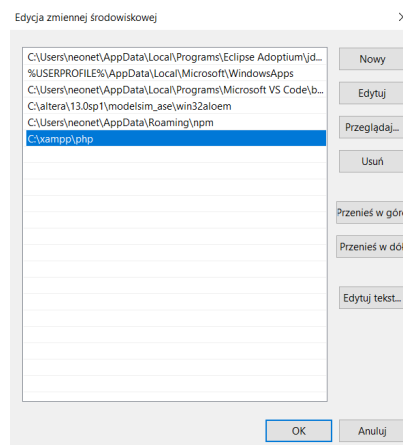


Rysunek 19: Panel XAMPP

5.6.4 Dodanie zmiennej środowiskowej PATH dla PHP - Windows 10

Aby korzystać z serwera deweloperskiego PHP należy dodać zmienną środowiskową PATH do systemu.

<https://dinocajic.medium.com/add-xampp-php-to-environment-variables-in-windows-10-af20a765b0ce>



Rysunek 20: Zmienna PATH

6 Wnioski

6.1 Podsumowanie

W trakcie realizacji projektu nad katalogiem piłkarzy opartym na OOP w PHP, skupiliśmy się na stworzeniu systemu, który integruje logiczne obiekty piłkarzy w spójny model danych. Implementacja strony internetowej za pomocą HTML, CSS i JavaScript pozwoliła nam na stworzenie atrakcyjnego i interaktywnego interfejsu dla użytkowników. Wykorzystanie architektury MVC umożliwiło nam lepszą strukturę projektu, dzięki czemu nasza aplikacja jest łatwiejsza w zarządzaniu, rozwijaniu i testowaniu.

6.2 Podział pracy

U-20019 - Jakub Achtelik

U-20041 - Oliwier Budnik

6.2.1 Zadania wykonane przez U-20019

- Zaprojektowanie oraz wykonanie bazy danych
- Konstrukcja zapytań oraz klas, które służą do łączenia z bazą danych
- Konfiguracja usług na serwerze VPS
- Implementacja połączenia API Wikipedia z naszą aplikacją
- Proste grafiki w programie GIMP
- Pobieranie danych od użytkownika za pomocą POST oraz GET

6.2.2 Zadania wykonane przez U-20041

- Prototyp wyglądu aplikacji
- Implementacja autoryzacji oraz formularz logowania
- Formularz edycji, dodawania i przekazanie tego do jednej tablicy
- Wyświetlanie zwróconych wyników z bazy danych w formie Szablonów HTML
- Szablon strony HTML oraz dostosowanie wyglądu w CSS
- Routing strony, czyli ustawienie klasy, która na podstawie /linku wykonuje określoną operację
- Implementacja logiki klasy Aplikacja

7 Bibliografia - źródła

Literatura

- [1] php.net - Oficjalna Dokumentacja PHP
<https://www.php.net/docs.php>
- [2] Medium - Informacje na temat MVC w PHP
<https://medium.com/@iamjoestack/how-to-build-a-custom-php-mvc-framework-e5a23da8f73d>
- [3] Konfiguracja Serwera SQL - Ubuntu VPS
<https://ubuntu.com/server/docs/databases-mysql>
- [4] Konfiguracja Serwera HTTP - Apache 2
<https://httpd.apache.org/docs/2.4/>
- [5] Tworzenie diagramów i schematów - PlantUML
<https://plantuml.com/>
- [6] Pobieranie obrazów z API Wikipedia
<https://stackoverflow.com/questions/8363531/accessing-main-picture-of-wikipedia-page-by-api>
- [7] Docker - Oficjalna Dokumentacja
<https://docs.docker.com/>
- [8] Obsługa programu GIMP
<https://www.gimp.org/docs/>
- [9] Budowanie strony przy użyciu HTML
<https://developer.mozilla.org/en-US/docs/Web/HTML>
- [10] Stylowanie HTML przy użyciu CSS
<https://developer.mozilla.org/en-US/docs/Web/CSS>
- [11] Polecenia stosowane w MySQL
<https://dev.mysql.com/doc/refman/8.2/en/>