*PROJEKT ZESPOŁOWY*

Projekt i implementacja aplikacji wspomagającej ligowe rozgrywki piłkarzyków.

*Artur Kucybała, Radosław Jagiełło, Mateusz Gwiazda*

*Grupa Z709*

**ETAP I**

**Analiza systemu**

1. Przedstawienie koncepcji systemu

1.1. Ogólny opis produktu

Aplikacja wspomagająca ligowe rozgrywki piłkarzyków powstała na potrzeby pracy inżynierskiej. Głównym celem wdrożenia oprogramowania jest udostępnienie światu funkcjonalnej aplikacji dzięki której każda grupa użytkowników będzie mogła w prostszy i bardziej wydajny sposób zarządzać prowadzonymi rozgrywkami w utworzonej przez tą grupę lidze. Aplikacja ta będzie informować każdego użytkownika o najważniejszych informacjach danej ligi, m.in. tabeli z punktacją, meczach odbytych oraz nadchodzących, profilach drużyn itp.

Aplikacja będzie przeznaczona przede wszystkim dla firm oraz innych zamkniętych grup. Z przeprowadzonych wstępnych analiz wynika, że w dużej części firm pracownicy w wolnym od pracy czasie prowadzą między sobą rozgrywki piłkarzyków lecz nie posiadają odpowiedniego narzędzia, które mogło by im pomóc stworzyć zamkniętą ligę dzięki, której mecze są z

Czas poświęcony na realizację projektu będzie także czasem edukowania i poznawania nowych możliwości platformy Visual Studio oraz wzorca MVC.

Diagram kontekstowy aplikacji wspomagającej zarządzanie ligą piłkarzyków.

***Administrator***

***Zarejestrowany użytkownik***

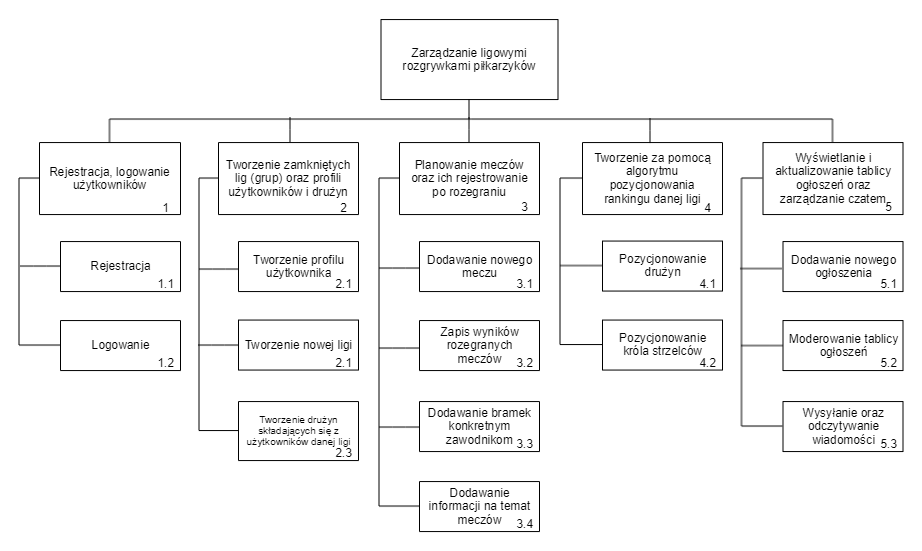
***Baza danych***

1.2. Charakterystyka użytkowników

* Użytkownik - jest to osoba która korzysta z aplikacji. Użytkownik posiada prywatne konto w systemie, ma możliwość zalogowania się oraz korzystania z wszystkich funkcjonalności systemu.
* Administrator jest to osoba administrująca i zarządzająca systemem, której zadania to przede wszystkim weryfikacja użytkowników, dokonywanie wszelkich aktualizacji oraz dbanie o bezpieczeństwo danych użytkowników.

2. Specyfikacja funkcjonalna

2.1. Diagram hierarchii funkcji (FHD)



2.2. Główne funkcje systemu

Użytkownik w kontekście aplikacji może:

* Tworzyć indywidualne konto użytkownika.
* Tworzyć nowe grupy w postaci własnej ligi.
* Aktualizować oraz sprawdzać status swojej ligi (będąc jej administratorem).
* Tworzyć zamknięte grupy użytkowników (nowe ligi).
* Administrować własną grupą użytkowników.
* Prowadzić rozmowę z innymi użytkownikami w postaci zamkniętego czatu.

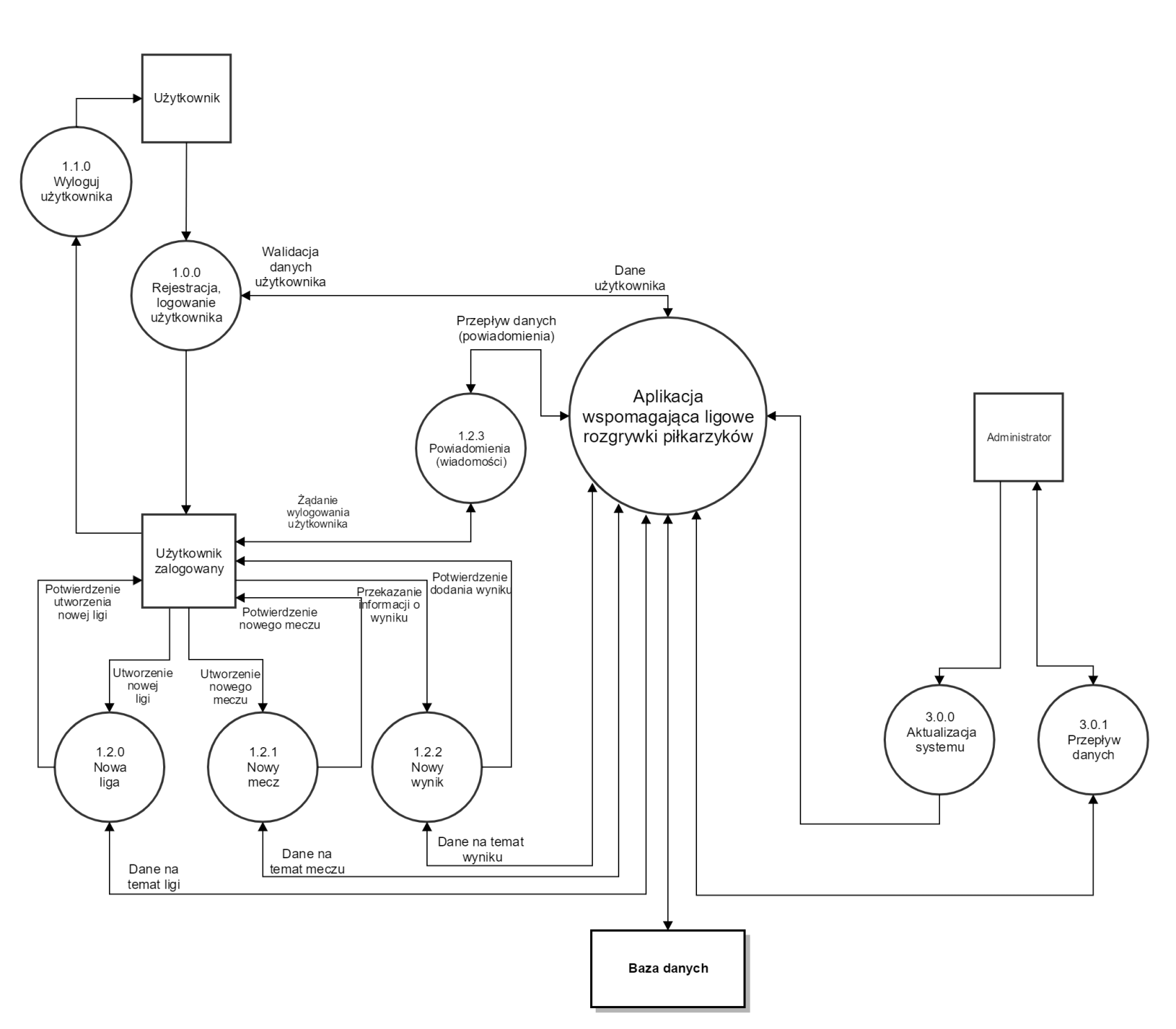
Administrator w kontekście aplikacji może:

* Dokonywać aktualizacji systemu.
* Zarządzać kontami użytkowników.
* Zarządzać bazą danych systemu.

Baza danych w kontekście aplikacji może:

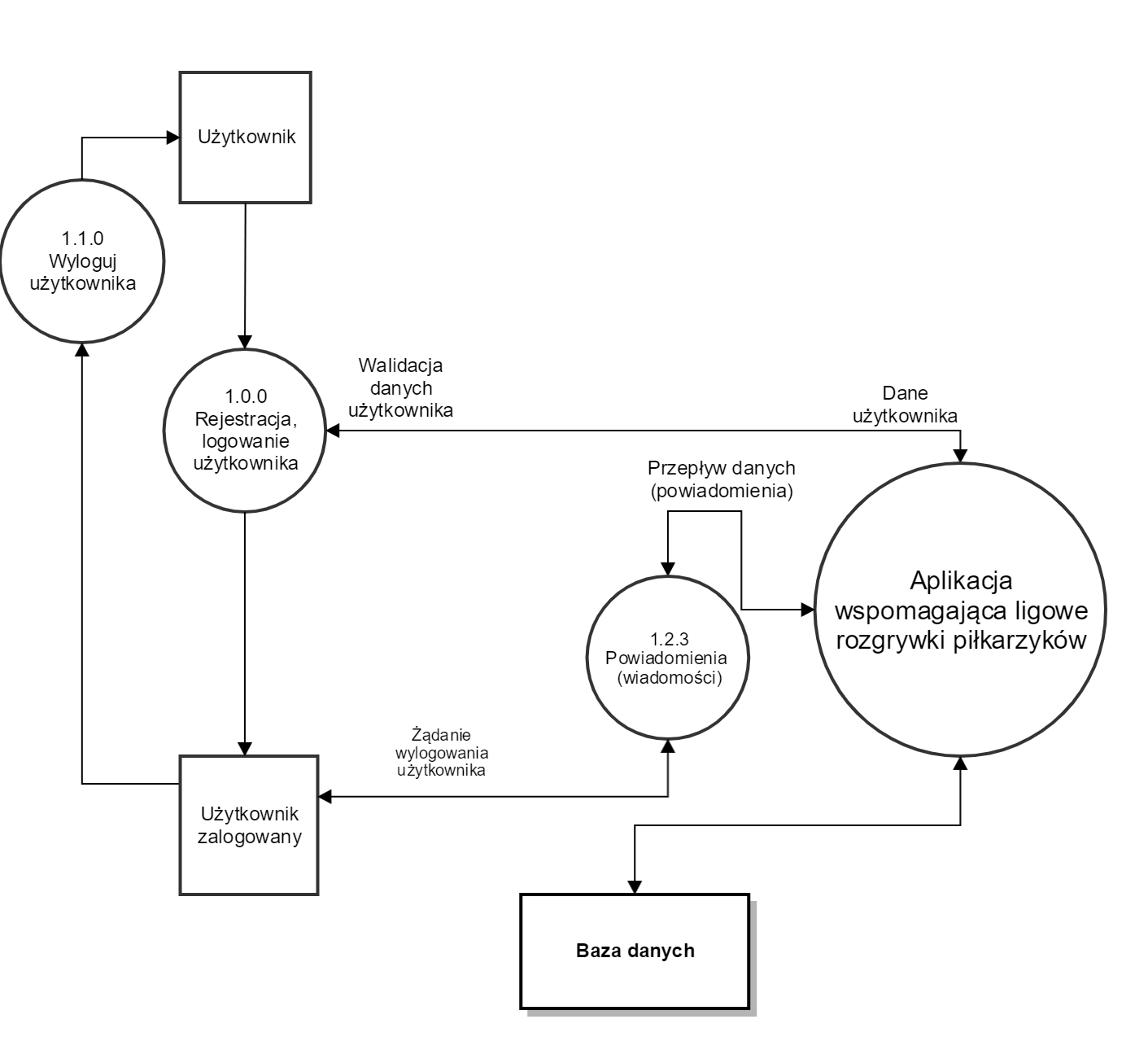
* Przyjmować oraz przechowywać dane użytkowników.
* Udostępniać przechowywane dane na żądanie systemu.

2.3 Diagram poziomu zerowego aplikacji wspomagającej ligowe rozgrywki piłkarzyków



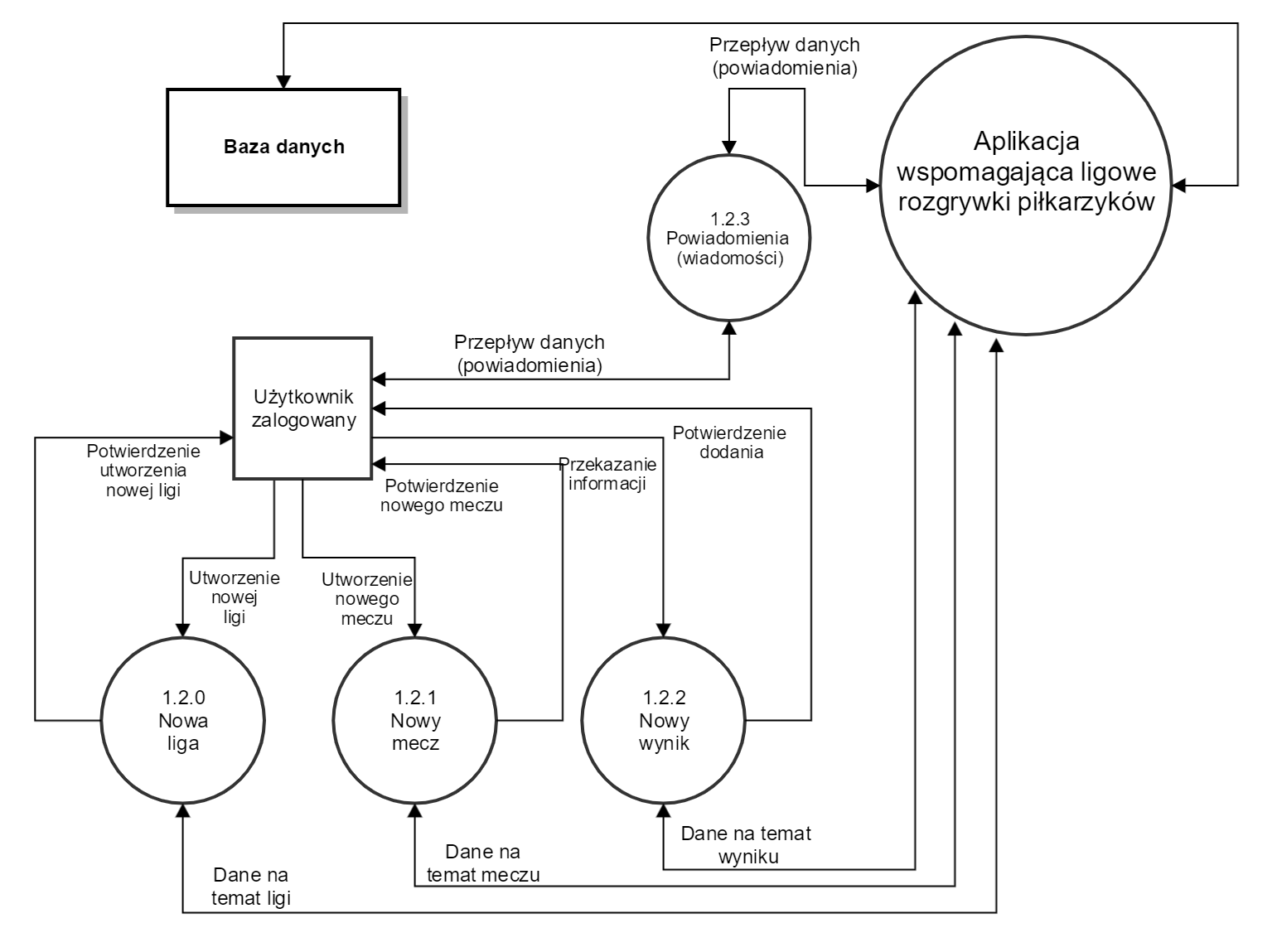
|  |  |  |
| --- | --- | --- |
| **Nazwa elementu** | **Typ elementu** | **Opis** |
| Użytkownik | Terminator | Osoba, która używa aplikacji wspomagającej ligowe rozgrywki piłkarzyków |
| Administrator | Terminator | Osoba zarządzająca całym systemem |
| 1.0.0  Rejestracja, logowanie użytkownika | Proces | Proces, w którym użytkownik rejestruje się i loguje w aplikacji |
| 2.0.0  Rejestracja, logowanie administratora | Proces | Proces w którym administrator rejestruje się i loguje w aplikacji |
| 1.1.0  Wyloguj użytkownika | Proces | Proces, w którym użytkownik zostaje wylogowany z aplikacji |
| 2.1.0  Wyloguj administratora | Proces | Proces w którym administrator zostaje wylogowany z aplikacji |
| 1.2.0  Nowa liga | Proces | Proces w którym zalogowany użytkownik tworzy na swoim profilu nową ligę piłkarzyków  (staje się on automatycznie administratorem danej grupy) |
| 1.2.1  Nowy mecz | Proces | Proces w którym zalogowany użytkownik tworzy na swoim profilu nowy mecz |
| 1.2.2  Nowy wynik | Proces | Proces, w którym użytkownik dodaje wynik odbytego meczu w lidze, której jest administratorem |
| 1.2.3  Powiadomienia  (wiadomości) | Proces | Proces, w którym użytkownik wysyła wiadomość tekstową do innego użytkownika będącego członkiem tej samej grupy. Jest to również proces, w którym system wysyła powiadomienia do użytkownika w postaci tekstowej oraz w formie tablicy ogłoszeń |
| 3.0.0  Aktualizacja  systemu | Proces | Proces, w którym terminator "Administrator" dokonuje zmian w aplikacji |
| 3.0.1  Przepływ danych | Proces | Proces, w którym terminator "Administrator" uzyskuje żądane dane z systemu oraz takie dane wysyła w postaci komunikatów. |
| Dane użytkownika | Przepływ | Przepływ danych zarejestrowanego użytkownika |
| Żądanie wylogowania użytkownika | Przepływ | Przepływ danych z żądaniem wylogowania użytkownika |
| Utworzenie nowej ligi | Przepływ | Przepływ zawierający informacje o nowej lidze stworzonej przez użytkownika |
| Potwierdzenie nowej ligi | Przepływ | Przepływ danych zawierający informację o dodaniu w aplikacji nowej ligi |
| Utworzenie nowego meczu | Przepływ | Przepływ danych zawierający informację o nowo utworzonym meczu |
| Potwierdzenie nowego meczu | Przepływ | Przepływ danych zawierający informację o przyjęciu przez aplikację nowego meczu |
| Przekazanie informacji o wyniku | Przepływ | Przepływ danych zawierający informacje o wyniku odbytego meczu (oraz innych informacji na temat odbytego meczu) |
| Potwierdzenie dodania wyniku | Przepływ | Przepływ danych zawierający informację o przyjęciu przez aplikację danych na temat odbytego meczu |
| Przepływ danych (powiadomienia) | Przepływ | Przepływ danych służący do komunikacji pomiędzy użytkownikami tej samej grupy oraz pomiędzy aplikacją a użytkownikiem |
| Dane na temat ligi | Przepływ | Przepływ danych zawierający informację o lidze |
| Dane na temat meczu | Przepływ | Przepływ danych zawierający informacje na temat meczu |
| Dane na temat wyniku | Przepływ | Przepływ danych zawierający informacje na temat wyniku oraz innych informacji, o których wiadomo po jego odbyciu |
| Baza danych | Magazyn danych | Magazyn danych na temat całości aplikacji |

3.2.1. Diagram rejestracji, logowania i wylogowania użytkownika.



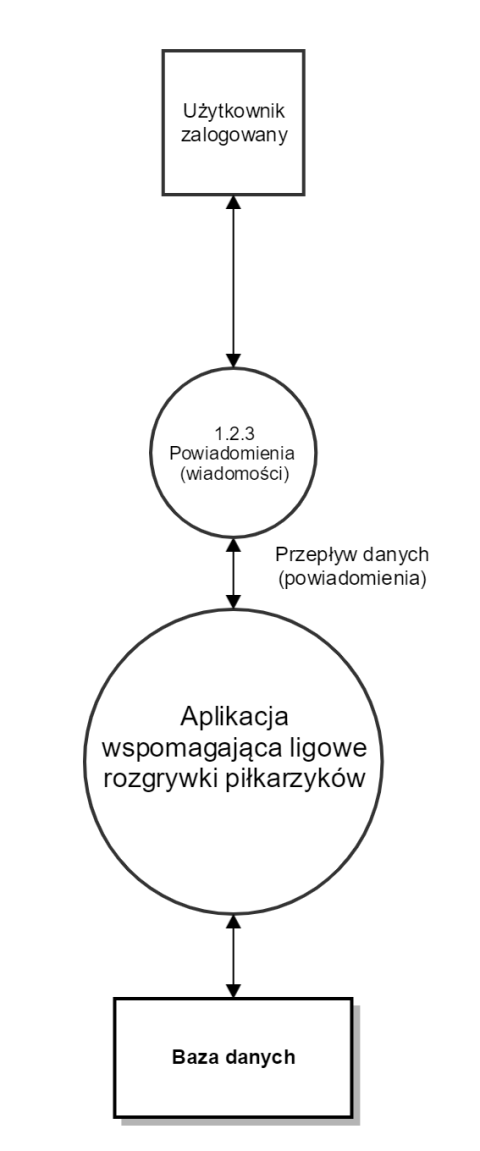
|  |  |  |
| --- | --- | --- |
| **Nazwa elementu** | **Typ elementu** | **Opis** |
| Użytkownik | Terminator | Osoba, która używa aplikacji wspomagającej ligowe rozgrywki piłkarzyków |
| 1.0.0  Rejestracja, logowanie użytkownika | Proces | Proces, w którym użytkownik rejestruje się i loguje w aplikacji |
| 1.1.0  Wyloguj użytkownika | Proces | Proces, w którym użytkownik zostaje wylogowany z aplikacji |
| Walidacja danych użytkownika | Przepływ | Przepływ danych potwierdzających możliwość rejestracji/logowania użytkownika |
| Dane użytkownika | Przepływ | Przepływ danych zarejestrowanego użytkownika |
| Walidacja danych użytkownika | Przepływ | Przepływ danych potwierdzających możliwość rejestracji/logowania użytkownika |

3.2.3. Diagram tworzenia nowej ligi, meczu oraz wyników



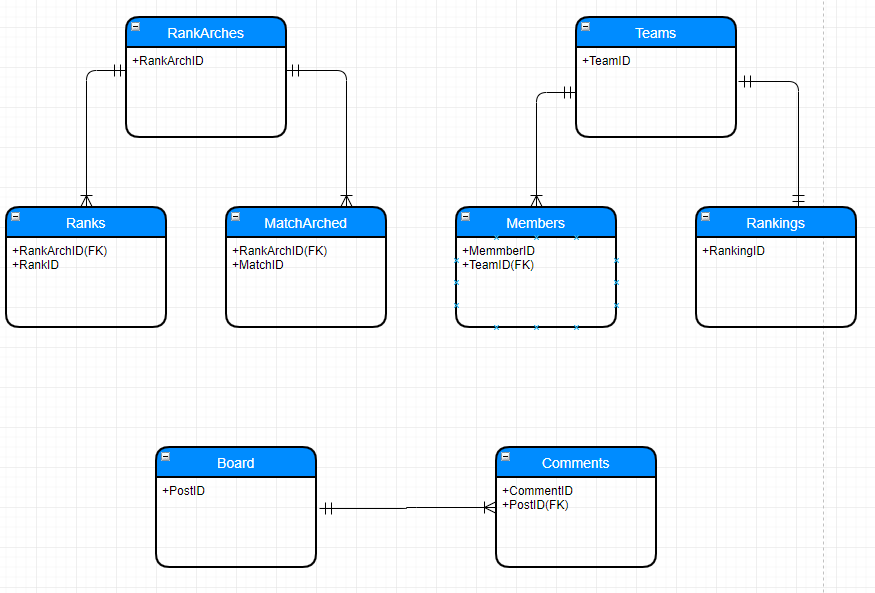
|  |  |  |
| --- | --- | --- |
| **Nazwa elementu** | **Typ elementu** | **Opis** |
| 1.2.0  Nowa liga | Proces | Proces w którym zalogowany użytkownik tworzy na swoim profilu nową ligę piłkarzyków  (staje się on automatycznie administratorem danej grupy) |
| 1.2.1  Nowy mecz | Proces | Proces w którym zalogowany użytkownik tworzy na swoim profilu nowy mecz |
| 1.2.2  Nowy wynik | Proces | Proces, w którym użytkownik dodaje wynik odbytego meczu w lidze, której jest administratorem |
| Baza Danych | Terminator | Baza przechowująca wszystkie informacje o użytkownikach i przekazująca te dane na żądanie systemu lub administratora. |
| Utworzenie nowej ligi | Przepływ | Przepływ zawierający informacje o nowej lidze stworzonej przez użytkownika |
| Potwierdzenie nowej ligi | Przepływ | Przepływ danych zawierający informację o dodaniu w aplikacji nowej ligi |
| Utworzenie nowego meczu | Przepływ | Przepływ danych zawierający informację o nowo utworzonym meczu |
| Potwierdzenie nowego meczu | Przepływ | Przepływ danych zawierający informację o przyjęciu przez aplikację nowego meczu |
| Przekazanie informacji o wyniku | Przepływ | Przepływ danych zawierający informacje o wyniku odbytego meczu (oraz innych informacji na temat odbytego meczu) |
| Potwierdzenie dodania wyniku | Przepływ | Przepływ danych zawierający informację o przyjęciu przez aplikację danych na temat odbytego meczu |
| Przepływ danych (powiadomienia) | Przepływ | Przepływ danych służący do komunikacji pomiędzy użytkownikami tej samej grupy oraz pomiędzy aplikacją a użytkownikiem |
| Dane na temat ligi | Przepływ | Przepływ danych zawierający informację o lidze |
| Dane na temat meczu | Przepływ | Przepływ danych zawierający informacje na temat meczu |
| Dane na temat wyniku | Przepływ | Przepływ danych zawierający informacje na temat wyniku oraz innych informacji, o których wiadomo po jego odbyciu |
| Baza danych | Magazyn danych | Magazyn danych na temat całości aplikacji |

3.2.4. Diagram przepływu powiadomień (wiadomości)

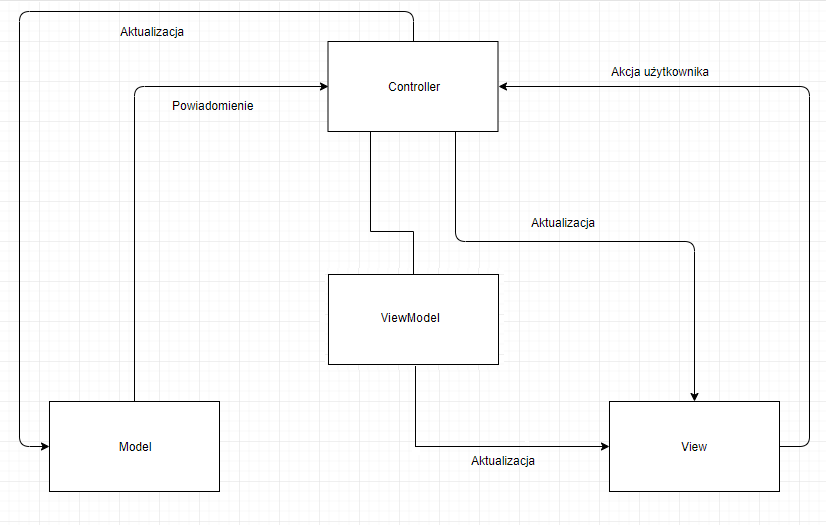


|  |  |  |
| --- | --- | --- |
| **Nazwa elementu** | **Typ elementu** | **Opis** |
| Użytkownik | Terminator | Osoba, która używa aplikacji wspomagającej ligowe rozgrywki piłkarzyków |
| 1.2.3  Powiadomienia  (wiadomości) | Proces | Proces, w którym użytkownik wysyła wiadomość tekstową do innego użytkownika będącego członkiem tej samej grupy. Jest to również proces, w którym system wysyła powiadomienia do użytkownika w postaci tekstowej oraz w formie tablicy ogłoszeń |
| Baza danych | Magazyn danych | Magazyn danych na temat całości aplikacji |

3. Diagram ERD (Entity Relationship Diagrams).



4. Model architektury systemu



**Architektura MVC**

MVC (Model-view-controller) jest tzw. wzorcem architektonicznym, stosowanym przy tworzeniu nowoczesnych systemów informatycznych.

Główną koncepcją MVC jest wymuszenie podziału aplikacji na 3 niezależne warstwy reprezentujące kolejno:

* (Model) Model - reprezentuje naszą logikę biznesową. Tutaj znajdują się wszelkie obiekty, które służą do wykonywania wszelkich operacji związanych z implementacją funkcjonalności naszej aplikacji
* (View) Widok - warstwa prezentacji. Widok odpowiedzialny jest za prezentację użytkownikowi wyników działania logiki biznesowej
* (Controller) Kontroler - obsługuje żądania użytkownika. Wszelkie żądania deleguje do odpowiednich metod Modelu.

Podział na warstwy służy uporządkowaniu architektury systemu. Dzięki temu, że każda logiczna część jest od siebie oddzielona, zmiana w jednym miejscu, nie powoduje konieczności wykonywania lawinowej ilości zmian w innych miejscach systemu.

Standardowy wzorzec MVC został rozszerzony o ViewModel – klasę modelu na potrzeby widoku.  
ViewModel zawiera:

* Kluczowe dla warstwy prezentacji dane
* Listę elementów zależnych - np. kontekstów które możemy wykorzystać w Tasku
* Informacje/dane dodatkowe przydatne w generowaniu widoku - np. informacje o stronicowaniu

Tworzony system jest aplikacją internetową. Do tego typu aplikacji, wzorzec MVC jest najkorzystniejszym wyborem. Posiada takie zalety jak:

* Podział na moduły porządkujące kod aplikacji,
* Oddzielenie logiki biznesowej od widoku,
* Brak zależności modelu od widoku,
* Ułatwia odnalezienie konkretnej części kodu,
* Łatwiejsza rozbudowa poprzez modułową budowę

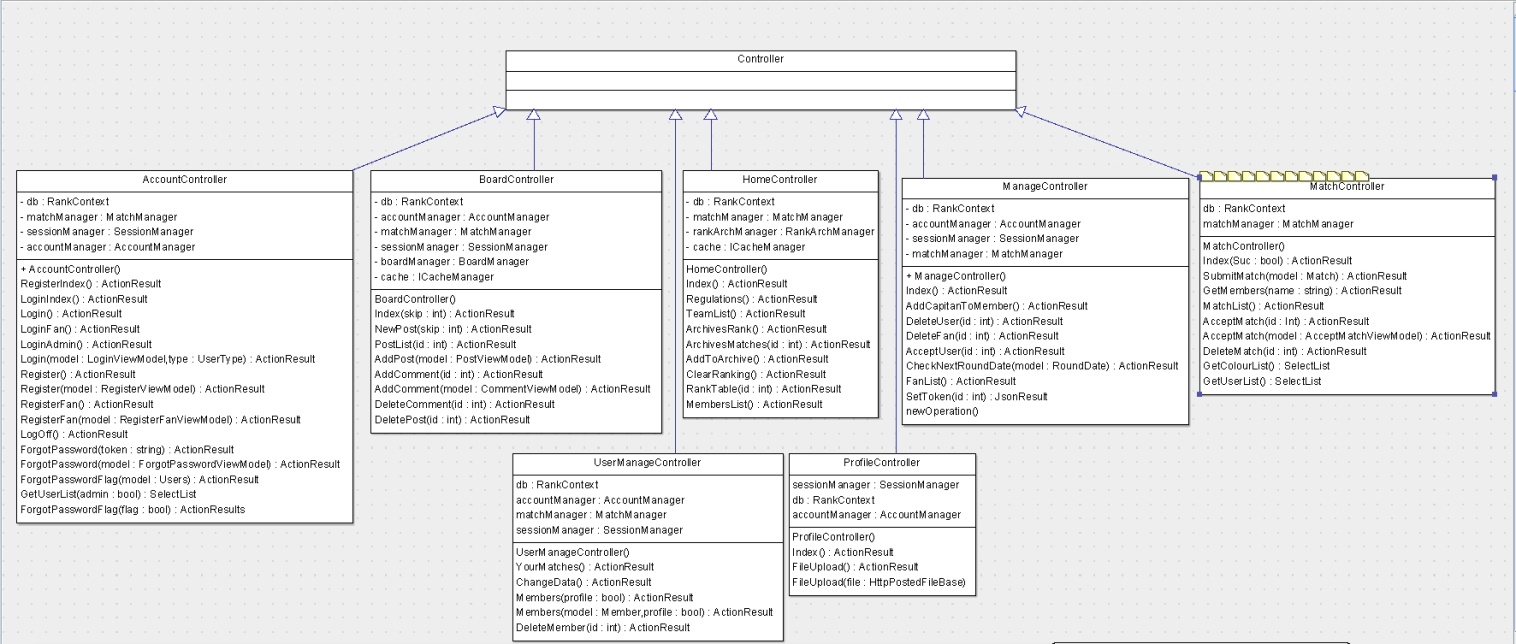
**ETAP II**

**Projekt systemu**

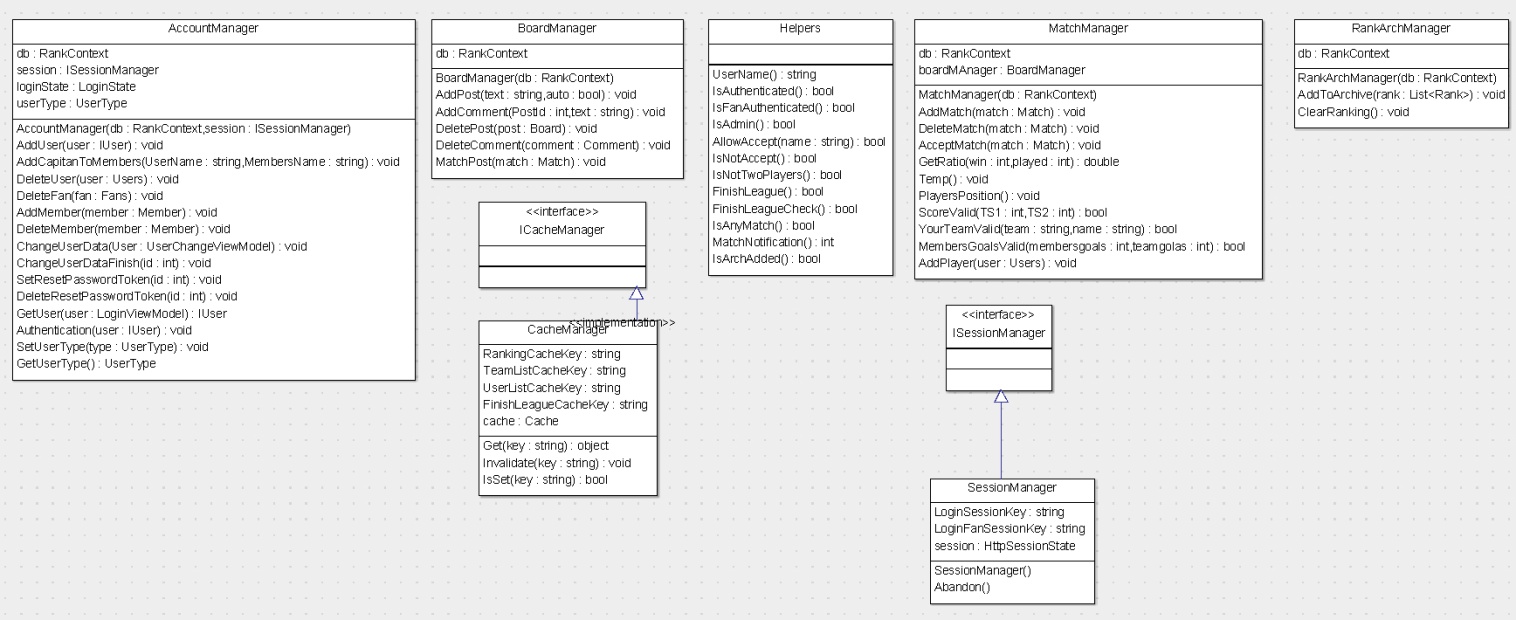
1. Projekt architektury systemu

1.1. Projekt diagramu klas

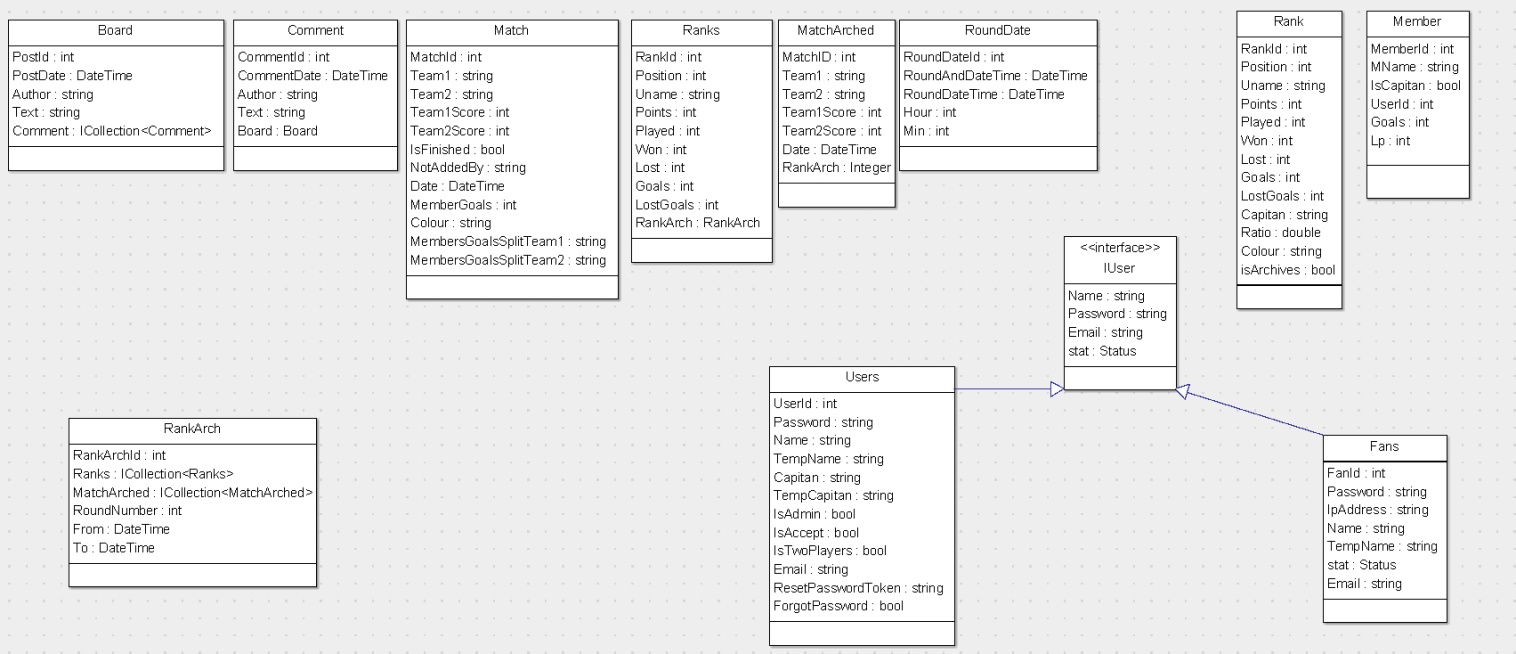
1.1.1. Warstwa konrolera



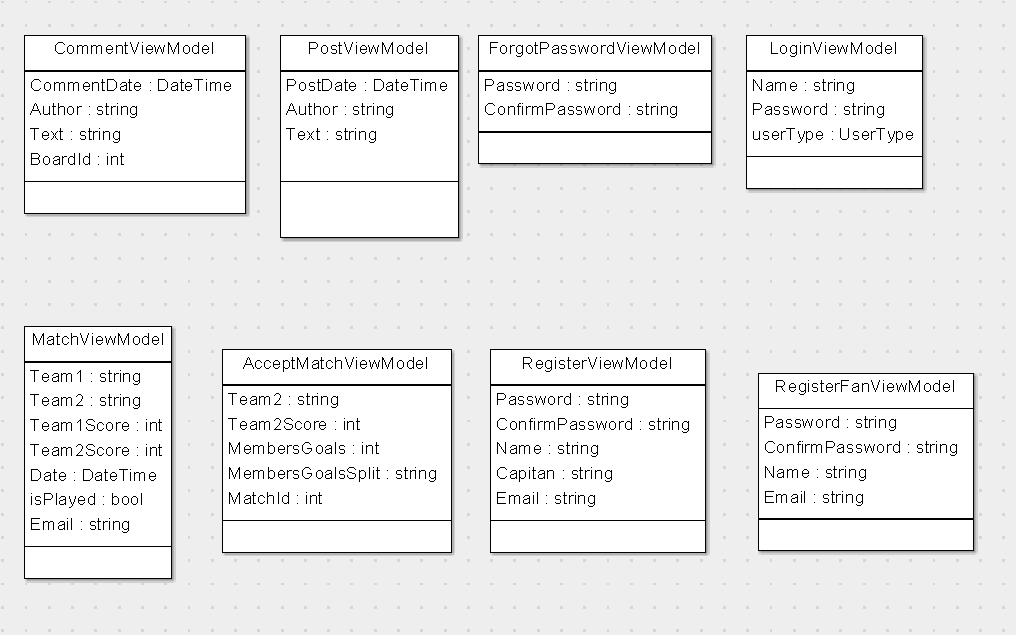
1.1.2. Warstwa infrastruktury



1.1.3. Warstwa modelu

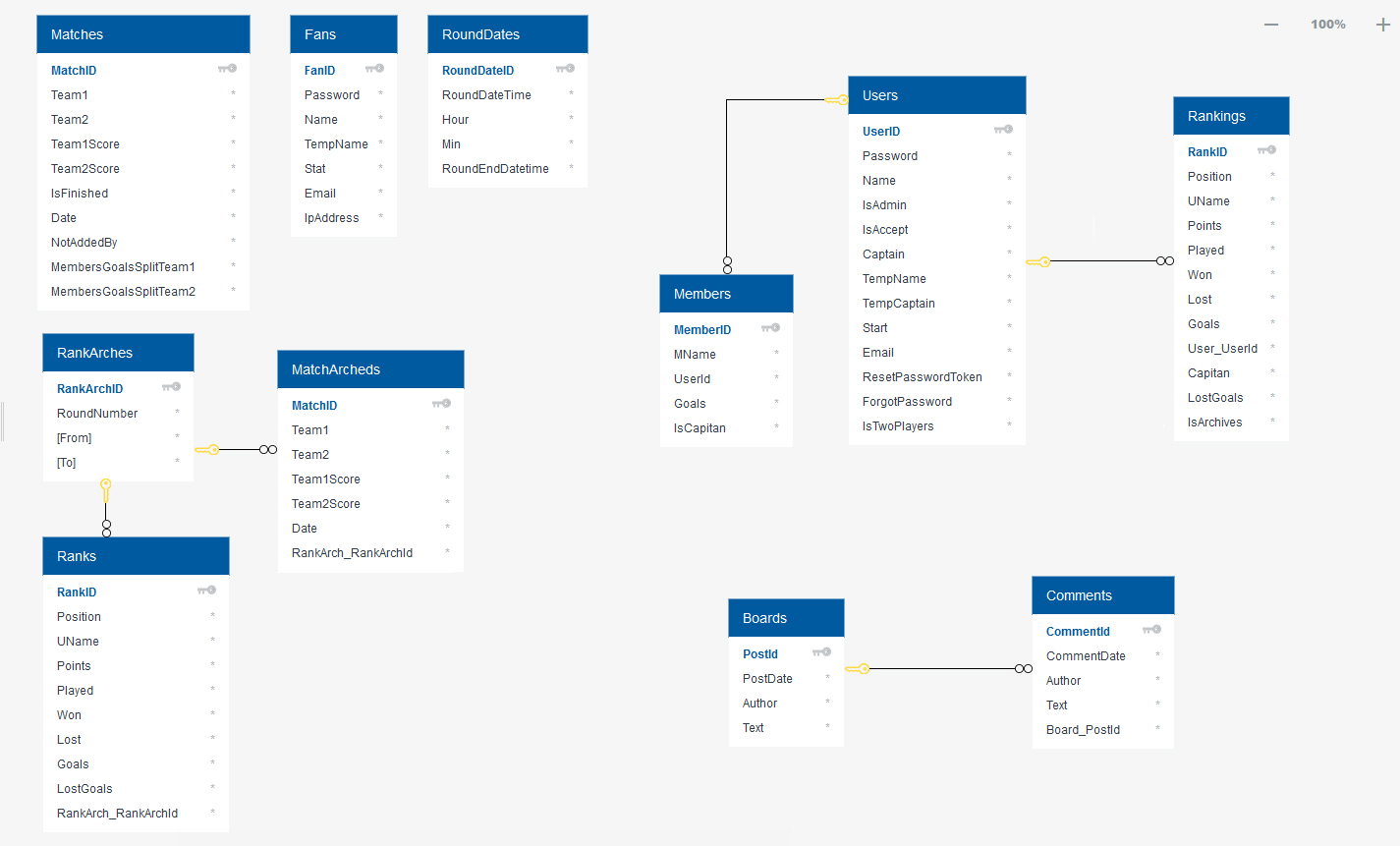


1.1.3. Warstwa widok-model



2. Projekt bazy danych

2.1. Diagram tabel i relacji



2.2. Opis pól i ich atrybutów

Tabela : Matches

|  |  |
| --- | --- |
| Nazwa pola: | Typ: |
| MatchId | int |
| Team1 | string |
| Team2 | string |
| Team1Score | int |
| Team2Score | int |
| IsFinished | bool |
| Date | DateTime |
| NotAddedBy | string |
| MembersGoalsSplitTeam1 | string |
| MembersGoalsSplitTeam2 | string |

Tabela : Fans

|  |  |
| --- | --- |
| Nazwa pola : | Typ: |
| FanId | int |
| Password | string |
| Name | string |
| TempName | string |
| Stat | Status |
| Email | string |
| IpAddress | string |

Tabela : RoundDates

|  |  |
| --- | --- |
| Nazwa pola : | Typ: |
| RoundDateId | int |
| RoundDateTime | DateTime |
| Hour | int |
| Min | int |
| RoundEndDateTime | DateTime |

Tabela : RankArches

|  |  |
| --- | --- |
| Nazwa pola : | Typ: |
| RankArchId | int |
| RoundNumber | int |
| [From] | DateTime |
| [To] | DateTime |

Tabela : MatchArcheds

|  |  |
| --- | --- |
| Nazwa pola : | Typ: |
| MatchID | Int |
| Team1 | String |
| Team2 | String |
| Team1Score | Int |
| Team2Score | Int |
| Date | DateTime |
| RankArch\_RankArchId | RankArch |

Tabela : Ranks

|  |  |
| --- | --- |
| Nazwa pola : | Typ: |
| RankID | Int |
| Position | Int |
| UName | String |
| Points | int |
| Played | int |
| Won | int |
| Loas | int |
| Goals | int |
| LostGoals | int |
| RankArch\_RankArchId | RankArch |

Tabela : Members

|  |  |
| --- | --- |
| Nazwa pola : | Typ: |
| MemberID | Int |
| MName | String |
| UserId | Int |
| Goals | Int |
| IsCapitan | bool |

Tabela : Users

|  |  |
| --- | --- |
| Nazwa pola : | Typ: |
| UserID | Int |
| Password | String |
| Name | String |
| IsAdmin | bool |
| IsAccept | bool |
| Capitan | String |
| TempName | String |
| TempCapitan | String |
| Stat | Status |
| Email | String |
| ResetPasswordToken | String |
| ForgotPassword | Bool |
| IsTwoPlayers | bool |

Tabela : Rankings

|  |  |
| --- | --- |
| Nazwa pola : | Typ: |
| RankID | Int |
| Position | Int |
| UName | string |
| Points | int |
| Played | int |
| Won | int |
| Lost | int |
| Goals | int |
| User\_UserId | int |
| Capitan | string |
| LostGoals | int |
| IsArchives | bool |

Tabela : Boards

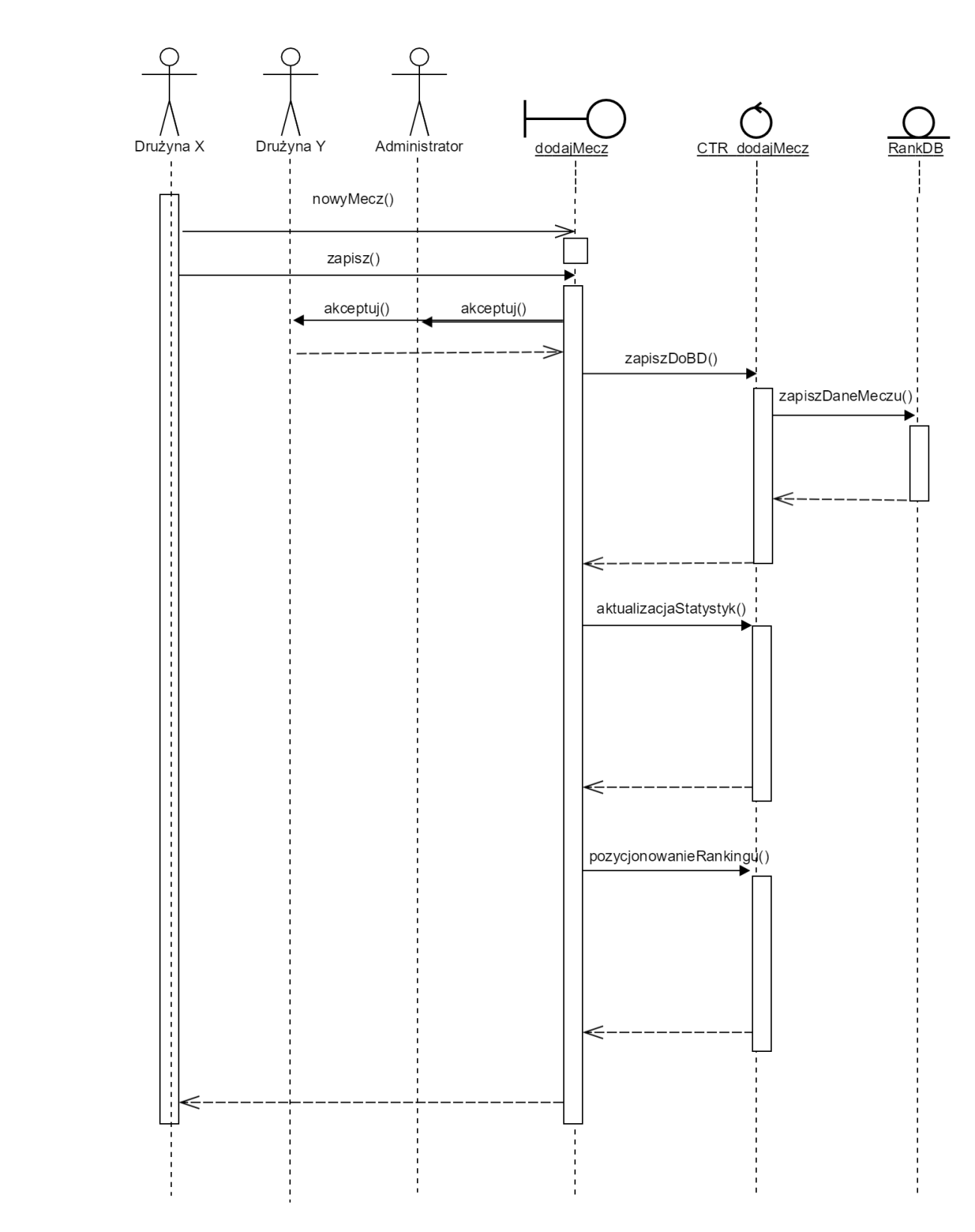
|  |  |
| --- | --- |
| Nazwa pola : | Typ: |
| PostId | Int |
| PostDate | DateTime |
| Author | String |
| Text | string |

Tabela : Comments

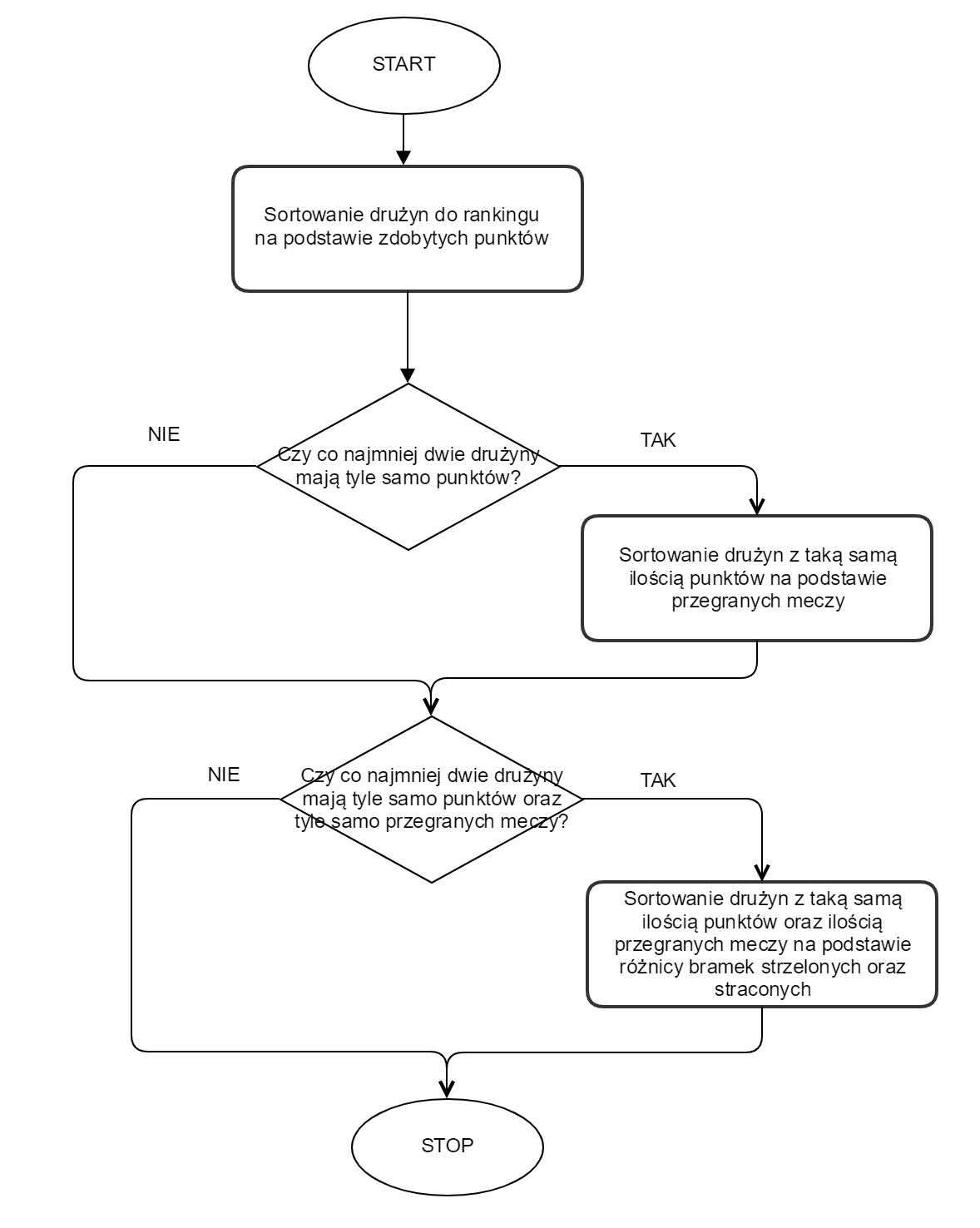
|  |  |
| --- | --- |
| Nazwa pola : | Typ: |
| CommentId | Int |
| CommentDate | DateTime |
| Author | String |
| Text | String |
| Board\_PostId | Board |

3. Projekt algorytmów

3.1. Zapis meczu - diagram sekwencji



3.2. Sortowanie drużyn do rankingu (diagram blokowy)



4. Projekt interfejsu użytkownika

Interfejs użytkownika definiuje przestrzeń, dzięki której użytkownik może podejmować interakcję z systemem. Z perspektywy użytkownika jest to bardzo ważna część systemu odpowiadająca przede wszystkim za jego postrzeganie oraz bezproblemowe korzystanie.

4.1. Makiety okien aplikacji

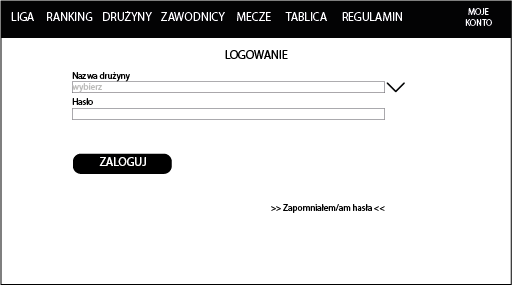
4.1.1. Rejestracja drużyny

Proces rejestracji drużyny jest pierwszym krokiem, który powinien wykonać nowy użytkownik chcący wziąć udział w rozgrywkach.



4.1.2. Formularz logowania drużyny

Następnie użytkownik po przejściu procesu rejestracji powinien się zalogować aby w pełni korzystać z możliwości aplikacji.



4.1.3. Drużyny

Poniżej widzimy makietę przedstawiającą zarejestrowane drużyny w systemie. Widok ten jest przypisany tylko do konta administratora, który może zarządzać wszystkimi drużynami oraz rundami rozgrywek.



4.1.4. Mecze

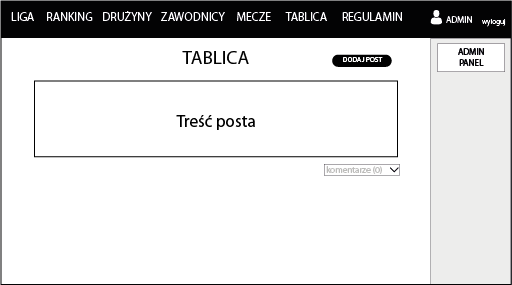
Makieta ta przedstawia widok dodawania odbytego meczu. Mecz może dodać zarówno administrator jak i użytkownik należący do drużyny w nim uczestniczącej.



4.1.5. Ranking



4.1.6. Tablica informacyjna dla wszystkich użytkowników



**ETAP III**

**Implementacja systemu**

Implementacja bazy danych

Przedstawienie sposobu implementacji wybranych elementów baz danych struktury oraz logiki.

W naszym projekcie korzystamy z narzędzia do mapowania obiektowo – relacyjnego czyli Entity Framework oraz z DAL czyli warstwy dostępu do danych. Wykorzystujemy podejście code first czyli na podstawie encji oraz modeli tworzymy logikę odpowiedzialną za wykonywanie operacji na tym modelu, w tym wypadku będzie to zapisywanie oraz wczytywanie wyników z relacyjnej bazy danych.

warstwa dostępu do danych

namespace Ranking.DAL

{

public class RankContext : DbContext

{

public RankContext()

: base("RankContext")

{ }

public DbSet<Match> Match { get; set; }

public DbSet<Rank> Rank { get; set; }

public DbSet<Users> Users { get; set; }

public DbSet<RankArch> RankArch { get; set; }

public DbSet<Ranks> Ranks { get; set; }

public DbSet<RoundDate> RoundDate { get; set; }

public DbSet<Board> Board { get; set; }

public DbSet<Comment> Comment { get; set; }

public DbSet<Fans> Fans { get; set; }

public DbSet<Member> Member { get; set; }

public DbSet<MatchArched> MatchArched { get; set; }

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();

}

}

}

modele danych

namespace Ranking.Models

{

public class Board

{

[Key]

public int PostId { get; set; }

public DateTime PostDate { get; set; }

public string Author { get; set; }

public string Text { get; set; }

public virtual ICollection<Comment> Comment { get; set; }

}

public class Comment

{

[Key]

public int CommentId { get; set; }

public DateTime CommentDate { get; set; }

public string Author { get; set; }

public string Text { get; set; }

public Board Board { get; set; }

}

public class Match

{

[Key]

public int MatchId { get; set; }

[Required]

public string Team1 { get; set; }

[Required]

public string Team2 { get; set; }

[Required(ErrorMessage = "Wpisz wynik")]

[Range(0,10, ErrorMessage = "Wynik musi być w zakresie 0 - 10")]

public int Team1Score { get; set; }

[Required(ErrorMessage = "Wpisz wynik")]

[Range(0, 10, ErrorMessage = "Wynik musi być w zakresie 0 - 10")]

public int Team2Score { get; set; }

[DefaultValue(false)]

public bool IsFinished { get; set; }

public string NotAddedBy { get; set; }

[Required]

public DateTime Date { get; set; }

[NotMapped]

[DefaultValue(0)]

public int MemberGoals { get; set; }

[Required(ErrorMessage = "Wybierz kolor drużyny")]

public string Colour { get; set; }

public string MembersGoalsSplitTeam1 { get; set; }

public string MembersGoalsSplitTeam2 { get; set; }

}

public class Ranks

{

[Key]

public int RankId { get; set; }

public int Position { get; set; }

public string Uname { get; set; }

[DefaultValue(0)]

public int Points { get; set; }

[DefaultValue(0)]

public int Played { get; set; }

[DefaultValue(0)]

public int Won { get; set; }

[DefaultValue(0)]

public int Lost { get; set; }

[DefaultValue(0)]

public int Goals { get; set; }

[DefaultValue(0)]

public int LostGoals { get; set; }

public RankArch RankArch { get; set; }

}

public class MatchArched

{

[Key]

public int MatchID { get; set; }

public string Team1 { get; set; }

public string Team2 { get; set; }

[DefaultValue(0)]

public int Team1Score { get; set; }

[DefaultValue(0)]

public int Team2Score { get; set; }

public DateTime Date { get; set; }

public RankArch RankArch { get; set; }

}

public class RankArch

{

[Key]

public int RankArchId { get; set; }

public virtual ICollection<Ranks> Ranks { get; set; }

public virtual ICollection<MatchArched> MatchArched { get; set; }

public int RoundNumber { get; set; }

public DateTime From { get; set; }

public DateTime To { get; set; }

}

public class RoundDate

{

[Key]

public int RoundDateId { get; set; }

[DataType(DataType.DateTime)]

public DateTime RoundEndDatetime { get; set; }

[DataType(DataType.DateTime)]

public DateTime RoundDatetime { get; set; }

[Range(0, 23, ErrorMessage = "Nieprawidłowa godzina")]

public int Hour { get; set; }

[Range(0, 59, ErrorMessage = "Nieprawidłowa minuta")]

public int Min { get; set; }

}

public class Rank

{

[Key]

public int RankId { get; set; }

public int Position { get; set; }

public string Uname { get; set; }

[DefaultValue(0)]

public int Points { get; set; }

[DefaultValue(0)]

public int Played { get; set; }

[DefaultValue(0)]

public int Won { get; set; }

[DefaultValue(0)]

public int Lost { get; set; }

[DefaultValue(0)]

public int Goals { get; set; }

[DefaultValue(0)]

public int LostGoals { get; set; }

public string Captain { get; set; }

[NotMapped]

[DefaultValue(0)]

public double Ratio { get; set; }

[NotMapped]

public string Colour { get; set; }

[DefaultValue(false)]

public bool IsArchives { get; set; }

}

public class Users : IUser

{

[Key]

public int UserId { get; set; }

[DataType(DataType.Password)]

public string Password { get; set; }

public string Name { get; set; }

public string TempName { get; set; }

public string Captain { get; set; }

public string TempCaptain { get; set; }

[DefaultValue(false)]

public bool IsAdmin { get; set; }

[DefaultValue(false)]

public bool IsAccept { get; set; }

[DefaultValue(false)]

public bool IsTwoPlayers { get; set; }

[DefaultValue(0)]

public Status stat { get; set; }

public string Email { get; set; }

public string ResetPasswordToken { get; set; }

[DefaultValue(false)]

public bool ForgotPassword { get; set; }

public virtual ICollection<Member> Members { get; set; }

}

public class Member

{

public int MemberId { get; set; }

[Required]

[StringLength(30, ErrorMessage = "{0} musi mieć co najmniej {2} i nie więcej niż {1} znaków.", MinimumLength = 1)]

public string MName { get; set; }

[DefaultValue(false)]

public bool IsCaptain { get; set; }

public int UserId { get; set; }

[DefaultValue(0)]

public int Goals { get; set; }

[NotMapped]

public int Lp { get; set; }

public virtual Users Users { get; set; }

}

public class Fans : IUser

{

[Key]

public int FanId { get; set; }

[DataType(DataType.Password)]

public string Password { get; set; }

public string IpAddress { get; set; }

public string Name { get; set; }

public string TempName { get; set; }

[DefaultValue(0)]

public Status stat { get; set; }

public string Email { get; set; }

}

public enum Status

{

Registration,

Modification

}

public interface IUser

{

string Name { get; set; }

string Password { get; set; }

string Email { get; set; }

Status stat { get; set; }

}

}

Implementacja warstwy logicznej

przedstawienie rozwiązań programistycznych dla wybranych problemów(struktur, algorytmów) opisanie sposobu realizacji wybranego algorytmu.

**Logownie do witryny**

Logowanie do witryny odbywa się przy pomocy unikalnego klucza w sesji

public class SessionManager : ISessionManager

{

public const string LoginSessionKey = "LoginSessionKey";

public const string LoginFanSessionKey = "LoginFanSessionKey";

private HttpSessionState session;

public SessionManager()

{

session = HttpContext.Current.Session;

session.Timeout = 60;

}

public void Abandon()

{

session.Abandon();

}

public T Get<T>(string key)

{

return (T)session[key];

}

public void Set<T>(string name, T value)

{

session[name] = value;

}

public T TryGet<T>(string key)

{

try

{

return (T)session[key];

}

catch(NullReferenceException)

{

return default(T);

}

}

}

Hasła przechowywane są jako hashe szyfrowane dzięki zestawowi kryptograficznemu SHA256

var sha256 = Crypto.SHA256(model.Password);

var user = new LoginViewModel() { Name = model.Name, Password = sha256 };

if (accountManager.Login(user))

{

accountManager.SetLoginState(true);

accountManager.Authentication(accountManager.GetUser(user));

return RedirectToAction("Index", "Home");

}

Metody logujące użytkownika

public bool Login(LoginViewModel user)

{

IUser userT;

if(userType == UserType.Fan)

{

userT = db.Fans.FirstOrDefault(u => u.Name == user.Name && u.Password == user.Password);

if (userT == null)

return false;

}

else if(userType == UserType.Team || userType == UserType.Admin)

{

userT = db.Users.FirstOrDefault(u => u.Name == user.Name && u.Password == user.Password);

if (userT == null)

return false;

}

return true;

}

public void SetLoginState(bool isLogin)

{

loginState = isLogin ? LoginState.Log\_on : LoginState.Log\_off;

}

public void Authentication(IUser user)

{

if(loginState == LoginState.Log\_on)

{

if (user is Fans)

session.Set<Fans>(SessionManager.LoginFanSessionKey, user as Fans);

else

session.Set<Users>(SessionManager.LoginSessionKey, user as Users);

}

else if(loginState == LoginState.Log\_off)

{

session.Set<Fans>(SessionManager.LoginFanSessionKey, null);

session.Set<Users>(SessionManager.LoginSessionKey, null);

}

}

public static bool IsAuthenticated()

{

ISessionManager session = new SessionManager();

if (session.Get<Users>(SessionManager.LoginSessionKey) == null)

return false;

return true;

}

public static bool IsFanAuthenticated()

{

ISessionManager session = new SessionManager();

if (session.Get<Fans>(SessionManager.LoginFanSessionKey) == null)

return false;

return true;

}

**Algorytm pozycjonowania rankingu**

Algorytm bierze pod uwagę 4 kryteria: ilość punktów, ilość przegranych meczów, ilość straconych i strzelonych bramek.  
Po wbudowanym sortowaniu na podstawie ilości punktów, każdy rekord porównywany jest z każdym innym i zamieniany miejscami, jeżeli nie spełnienia chociaż jednego z kryteriów.

public void PlayersPosition()

{

var rank = db.Rank.ToList();

rank.Sort((x, y) => x.Points.CompareTo(y.Points));

int p = rank.Count;

foreach(var r in rank)

{

r.Position = p;

p--;

}

for(int i = 0; i<rank.Count; i++)

{

for (int j = 0; j < rank.Count; j++)

{

if(rank[j].Points == rank[i].Points && i != j)

{

if(rank[j].Position < rank[i].Position && rank[j].Lost > rank[i].Lost)

{

int pos = rank[j].Position;

rank[j].Position = rank[i].Position;

rank[i].Position = pos;

}

if(rank[j].Position < rank[i].Position && rank[j].LostGoals > rank[i].LostGoals && rank[j].Lost == rank[i].Lost)

{

int pos = rank[j].Position;

rank[j].Position = rank[i].Position;

rank[i].Position = pos;

}

if (rank[j].Position < rank[i].Position && rank[j].LostGoals == rank[i].LostGoals && rank[j].Lost == rank[i].Lost && rank[j].Goals < rank[i].Goals)

{

int pos = rank[j].Position;

rank[j].Position = rank[i].Position;

rank[i].Position = pos;

}

}

}

}

db.SaveChanges();

}

**Dodawanie oraz akceptacja meczu**

Dodawanie meczu

Dodany mecz wraz z wynikiem i przypisanymi bramkami do zawodników przechowywany jest w bazie z flagą akceptacji ustawioną na false.

public void AddMatch(Match match)

{

match.NotAddedBy = Helpers.UserName() == match.Team1 ? match.Team2 : match.Team1;

match.Date = DateTime.Now;

if(match.Colour == "niebieska")

{

string tempName = "";

string tempMembersGoalsSplitTeam1 = "";

int tempScore;

tempName = match.Team1;

tempScore = match.Team1Score;

tempMembersGoalsSplitTeam1 = match.MembersGoalsSplitTeam1;

match.Team1 = match.Team2;

match.Team1Score = match.Team2Score;

match.Team2 = tempName;

match.Team2Score = tempScore;

match.MembersGoalsSplitTeam1 = match.MembersGoalsSplitTeam2;

match.MembersGoalsSplitTeam2 = tempMembersGoalsSplitTeam1;

}

db.Match.Add(match);

db.SaveChanges();

}

Akceptacja meczu

Podczas akceptacji, statystyki w rankingu i profilach drużyn oraz zawodników są aktualizowane. Po wykonaniu metedy, flaga akceptacji ustawiana jest na true, po czym wywoływany jest algorytm pozycjonowania rankingu.

public void AcceptMatch(Match match)

{

var rank1 = db.Rank.Where(r => r.Uname == match.Team1).SingleOrDefault();

var rank2 = db.Rank.Where(r => r.Uname == match.Team2).SingleOrDefault();

db.Match.Find(match.MatchId).IsFinished = true;

db.Match.Find(match.MatchId).Colour = "czerowny";

if(match.Team1Score > match.Team2Score)

{

rank1.Won += 1;

rank1.Points += 3;

rank1.Played += 1;

rank2.Lost += 1;

rank2.Played += 1;

}

else

{

rank2.Won += 1;

rank2.Points += 3;

rank2.Played += 1;

rank1.Lost += 1;

rank1.Played += 1;

}

rank1.Goals += match.Team1Score;

rank2.Goals += match.Team2Score;

var user1 = db.Users.Where(u => u.Name == match.Team1).SingleOrDefault();

var user2 = db.Users.Where(u => u.Name == match.Team2).SingleOrDefault();

string[] value1 = match.MembersGoalsSplitTeam1.ToString().TrimEnd().Split(' ');

string[] value2 = match.MembersGoalsSplitTeam2.ToString().TrimEnd().Split(' ');

if (match.MembersGoalsSplitTeam1 == "admin")

{

int count = user1.Members.Count();

int mod = match.Team1Score % count;

if (mod == 0)

foreach (var m in user1.Members)

m.Goals += match.Team1Score / count;

else

{

int result = match.Team1Score - mod;

foreach (var m in user1.Members)

m.Goals += result / count;

user1.Members.Where(m => m.IsCaptain == true).SingleOrDefault().Goals +=mod;

}

}

else

{

foreach(var v in value1)

{

string[] val = v.Split('|');

user1.Members.Where(m => m.MemberId == int.Parse(val[0])).SingleOrDefault().Goals += int.Parse(val[1]);

}

}

if (match.MembersGoalsSplitTeam2 == "admin")

{

int count = user2.Members.Count();

int mod = match.Team2Score % count;

if (mod == 0)

foreach (var m in user2.Members)

m.Goals += match.Team2Score / count;

else

{

int result = match.Team2Score - mod;

foreach (var m in user2.Members)

m.Goals += result / count;

user2.Members.Where(m => m.IsCaptain == true).SingleOrDefault().Goals += mod;

}

}

else

{

foreach (var v in value2)

{

string[] val = v.Split('|');

user2.Members.Where(m => m.MemberId == int.Parse(val[0])).SingleOrDefault().Goals += int.Parse(val[1]);

}

}

db.SaveChanges();

boardManager.MatchPost(match);

PlayersPosition();

}

Implementacja GUI

3. Projekt interfejsu użytkownika

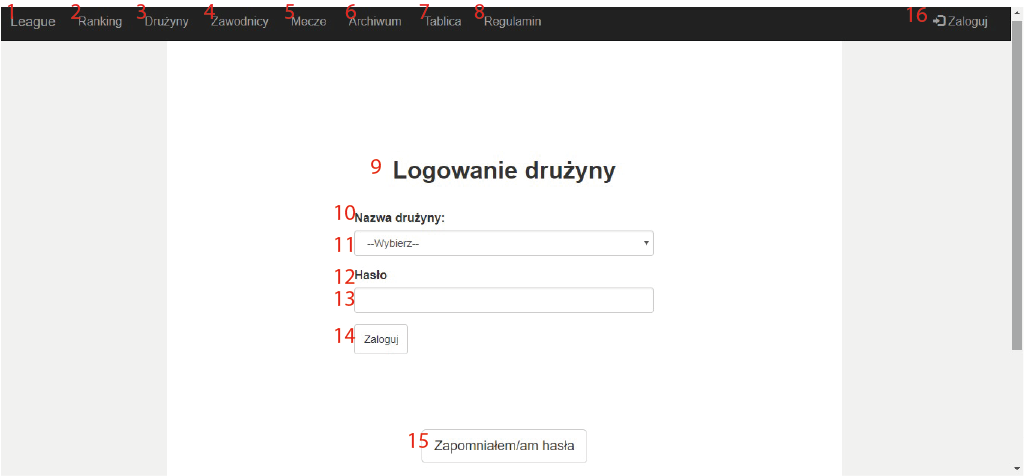
3.1. Okna aplikacji

3.1.1. Rejestracja drużyny



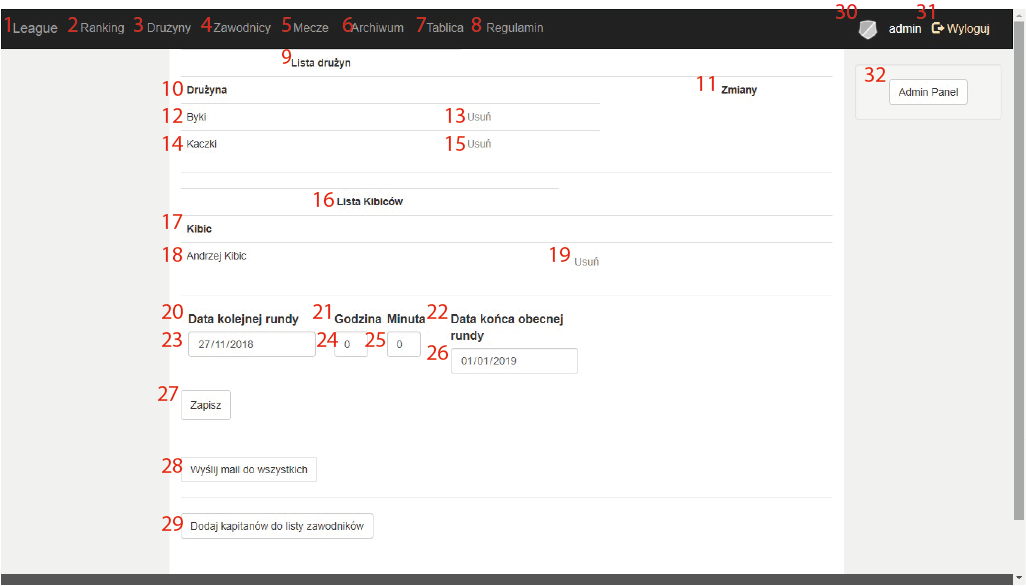
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr** | **Typ obiektu** | **Zdefiniowana treść** | **Kolor tekstu** | **Pole edytowalne** | **Funkcja** |
| 1 | var | League | biały | nie | Otwarcie strony głównej "League" po kliknięciu |
| 2 | var | Ranking | biały | nie | Otwarcie strony "Ranking" po kliknięciu |
| 3 | var | Drużyny | biały | nie | Otwarcie strony "Drużyny" po kliknięciu |
| 4 | var | Zawodnicy | biały | nie | Otwarcie strony "Zawodnicy" po kliknięciu |
| 5 | var | Mecze | biały | nie | Otwarcie strony "Mecze" po kliknięciu |
| 6 | var | Archiwum | biały | nie | Otwarcie strony "Archiwum" po kliknięciu |
| 7 | var | Tablica | biały | nie | Otwarcie strony "Tablica" po kliknięciu |
| 8 | var | Regulamin | biały | nie | Otwarcie strony "Regulamin" po kliknięciu |
| 9 | Div | Rejestracja drużyny | czarny | nie | Informacja dla użytkownika |
| 10 | Div | Adres e-mail | czarny | nie | Informacja dla użytkownika |
| 11 | String | brak | czarny | tak | Pole przeznaczone na wpisanie adresu e-mail |
| 12 | Div | Nazwa drużyny | czarny | nie | Informacja dla użytkownika |
| 13 | String | brak | czarny | tak | Pole przeznaczone na wpisanie nazwy drużyny |
| 14 | Div | Kapitan | czarny | nie | Informacja dla użytkownika |
| 15 | String | brak | czarny | tak | Pole przeznaczone na wpisanie nazwy kapitana |
| 16 | Div | Hasło | czarny | nie | Informacja dla użytkownika |
| 17 | String | brak | czarny | tak | Pole przeznaczone na wpisanie hasła |
| 18 | Div | Potwierdź hasło | czarny | nie | Informacja dla użytkownika |
| 19 | String | brak | czarny | tak | Pole przeznaczone na wpisanie hasła po raz drugi (potwierdzenie) |
| 20 | input | Zarejestruj | czarny | nie | Rozpoczęcie i akceptacja procesu rejestracji |

3.1.2. Formularz logowania drużyny



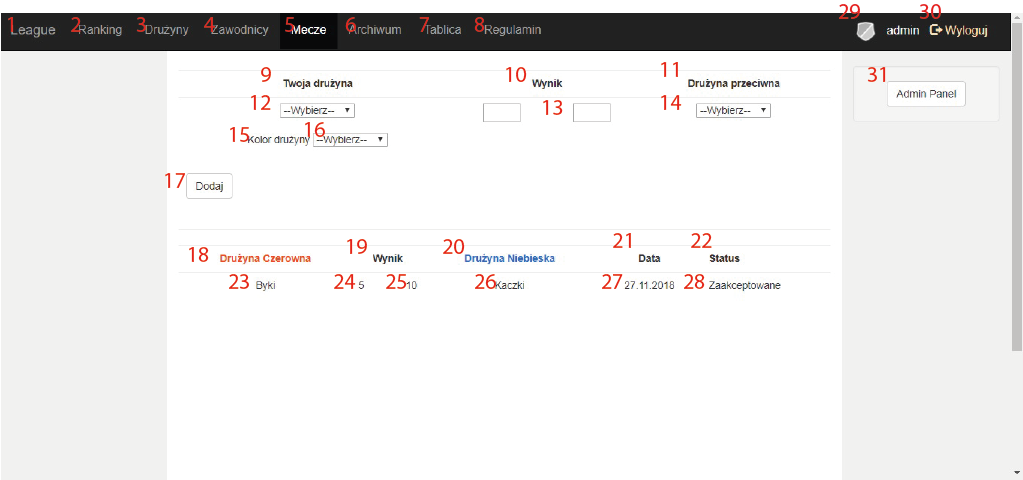
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr** | **Typ obiektu** | **Zdefiniowana treść** | **Kolor tekstu** | **Pole edytowalne** | **Funkcja** |
| 1 | var | League | biały | nie | Otwarcie strony głównej "League" po kliknięciu |
| 2 | var | Ranking | biały | nie | Otwarcie strony "Ranking" po kliknięciu |
| 3 | var | Drużyny | biały | nie | Otwarcie strony "Drużyny" po kliknięciu |
| 4 | var | Zawodnicy | biały | nie | Otwarcie strony "Zawodnicy" po kliknięciu |
| 5 | var | Mecze | biały | nie | Otwarcie strony "Mecze" po kliknięciu |
| 6 | var | Archiwum | biały | nie | Otwarcie strony "Archiwum" po kliknięciu |
| 7 | var | Tablica | biały | nie | Otwarcie strony "Tablica" po kliknięciu |
| 8 | var | Regulamin | biały | nie | Otwarcie strony "Regulamin" po kliknięciu |
| 9 | Div | Logowanie drużyny | czarny | nie | Informacja dla użytkownika |
| 10 | Div | Nazwa drużyny | czarny | nie | Informacja dla użytkownika |
| 11 | Div | brak | czarny | tak | Pole przeznaczone na wybranie drużyny |
| 12 | Div | Hasło | czarny | nie | Informacja dla użytkownika |
| 13 | String | brak | czarny | tak | Pole przeznaczone na wpisanie hasła |
| 14 | input | Zaloguj | czarny | nie | Rozpoczęcie i akceptacja procesu logowania |
| 15 | Div | Zapomniałem/am hasła | czarny | tak | Uruchamianie procesu odzyskiwania hasła |
| 16 | Div | Zaloguj | czarny | nie | Przejście do strony logowania |

3.1.3. Drużyny (widok z panelu administratora)



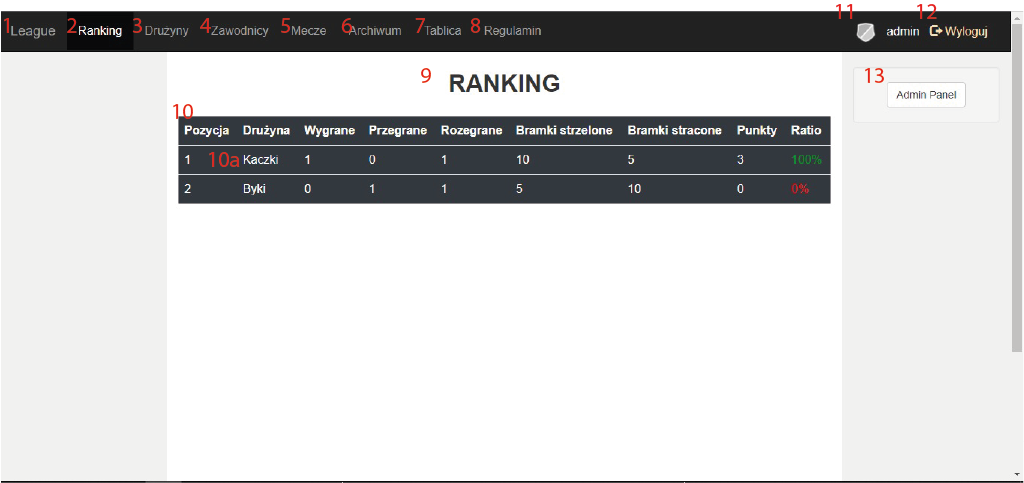
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr** | **Typ obiektu** | **Zdefiniowana treść** | **Kolor tekstu** | **Pole edytowalne** | **Funkcja** |
| 1 | var | League | biały | nie | Otwarcie strony głównej "League" po kliknięciu |
| 2 | var | Ranking | biały | nie | Otwarcie strony "Ranking" po kliknięciu |
| 3 | var | Drużyny | biały | nie | Otwarcie strony "Drużyny" po kliknięciu |
| 4 | var | Zawodnicy | biały | nie | Otwarcie strony "Zawodnicy" po kliknięciu |
| 5 | var | Mecze | biały | nie | Otwarcie strony "Mecze" po kliknięciu |
| 6 | var | Archiwum | biały | nie | Otwarcie strony "Archiwum" po kliknięciu |
| 7 | var | Tablica | biały | nie | Otwarcie strony "Tablica" po kliknięciu |
| 8 | var | Regulamin | biały | nie | Otwarcie strony "Regulamin" po kliknięciu |
| 9 | Div | Lista drużyn | czarny | nie | Informacja dla administratora |
| 10 | Div | Drużyna | czarny | nie | Informacja dla administratora |
| 11 | Div | Zmiany | czarny | nie | Informacja dla administratora |
| 12 | Div | "nazwa drużyny" | czarny | nie | Informacja dla administratora |
| 13 | Div | usuń | czarny | nie | Usuwanie drużyny |
| 14 | Div | "nazwa drużyny" | czarny | nie | Informacja dla administratora |
| 15 | Div | usuń | czarny | nie | Usuwanie drużyny |
| 16 | Div | Lista kiiców | czarny | nie | Informacja dla administratora |
| 17 | Div | Kibic | czarny | nie | Informacja dla administratora |
| 18 | Div | "nazwa kibica" | czarny | nie | Informacja dla administratora |
| 19 | Div | usuń | czarny | nie | Usuwanie kibica |
| 20 | Div | Data kolejnej rundy | czarny | nie | Informacja dla administratora |
| 21 | Div | Godzina / minuta | czarny | nie | Informacja dla administratora |
| 22 | Div | Data końca obecnej rundy | czarny | nie | Informacja dla administratora |
| 23 | String | Data kolejnej rundy | czarny | nie | Wprowadzanie daty kolejnej rundy |
| 24 | String | Godzina | czarny | nie | Wprowadzanie godziny |
| 25 | String | Minuta | czarny | nie | Wprowadzanie minuty |
| 26 | String | Data końca obecnej rundy | czarny | nie | Wprowadzanie daty końca obecnej rundy |
| 27 | Div | Zapisz | czarny | nie | Zapisanie zmian |
| 28 | Div | Wyślij mail do wszystkich | czarny | nie | Wysyłanie wiadomości informacyjnej w postaci emaila do wszystkich zawodników |
| 29 | Div | Dodaj kapitanów do listy zawodników | czarny | nie | Dodawanie kapitanów drużyn do listy zawodników |
| 30 | Div | "zdjęcie profilowe" | czarny | nie | Zdjęcie profilowe |
| 31 | Div | Wyloguj | czarny | nie | Wylogowanie użytkownika |
| 32 | Div | Admin panel | czarny | nie | Przejście do panelu administratora |

3.1.4. Mecze (dodawanie zakończonego meczu)



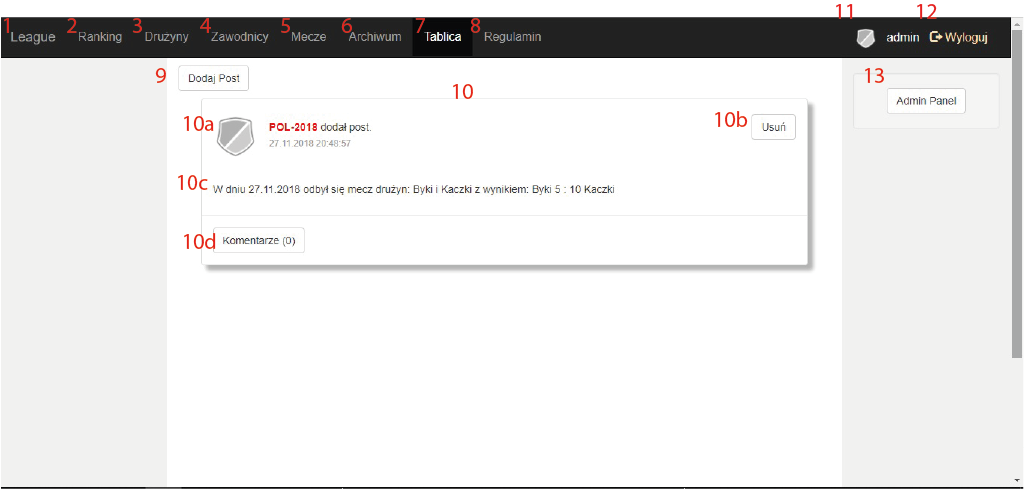
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr** | **Typ obiektu** | **Zdefiniowana treść** | **Kolor tekstu** | **Pole edytowalne** | **Funkcja** |
| 1 | var | League | biały | nie | Otwarcie strony głównej "League" po kliknięciu |
| 2 | var | Ranking | biały | nie | Otwarcie strony "Ranking" po kliknięciu |
| 3 | var | Drużyny | biały | nie | Otwarcie strony "Drużyny" po kliknięciu |
| 4 | var | Zawodnicy | biały | nie | Otwarcie strony "Zawodnicy" po kliknięciu |
| 5 | var | Mecze | biały | nie | Otwarcie strony "Mecze" po kliknięciu |
| 6 | var | Archiwum | biały | nie | Otwarcie strony "Archiwum" po kliknięciu |
| 7 | var | Tablica | biały | nie | Otwarcie strony "Tablica" po kliknięciu |
| 8 | var | Regulamin | biały | nie | Otwarcie strony "Regulamin" po kliknięciu |
| 9 | Div | Twoja drużyna | czarny | nie | Informacja dla użytkownika |
| 10 | Div | Wynik | czarny | nie | Informacja dla użytkownika |
| 11 | Div | Drużyna przeciwna | czarny | nie | Informacja dla użytkownika |
| 12 | String | brak | czarny | nie | Wybór drużyny |
| 13 | String | brak | czarny | nie | Wpisanie wyniku meczu |
| 14 | String | brak | czarny | nie | Wybór drużyny przeciwnej |
| 15 | Div | Kolor drużyny | czarny | nie | Informacja dla użytkownika |
| 16 | Div | brak | czarny | nie | Wybranie koloru drużyny |
| 17 | Div | Dodaj | czarny | nie | Dodanie meczu przez użytkownika |
| 18 | Div | Drużyna czerwona | czarny | nie | Informacja dla użytkownika |
| 19 | Div | Wynik | czarny | nie | Informacja dla użytkownika |
| 20 | Div | Drużyna niebieska | czarny | nie | Informacja dla użytkownika |
| 21 | Div | Data | czarny | nie | Data dodania meczu |
| 22 | Div | Status | czarny | nie | Status wprowadzonego meczu |
| 23 | String | brak | czarny | nie | Kolor drużyny 1szej |
| 24 | String | brak | czarny | nie | Ilość bramek drużyny czerwonej |
| 25 | String | brak | czarny | nie | Ilość bramek drużyny niebieskiej |
| 26 | String | brak | czarny | nie | Kolor drużyny 2giej |
| 27 | String | brak | czarny | nie | Data dodania meczu |
| 28 | String | brak | czarny | nie | Status meczu |
| 30 | Div | "zdjęcie profilowe" | czarny | nie | Zdjęcie profilowe |
| 31 | Div | Wyloguj | czarny | nie | Wylogowanie użytkownika |
| 32 | Div | Admin panel | czarny | nie | Przejście do panelu administratora |

3.1.5. Ranking



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr** | **Typ obiektu** | **Zdefiniowana treść** | **Kolor tekstu** | **Pole edytowalne** | **Funkcja** |
| 1 | var | League | biały | nie | Otwarcie strony głównej "League" po kliknięciu |
| 2 | var | Ranking | biały | nie | Otwarcie strony "Ranking" po kliknięciu |
| 3 | var | Drużyny | biały | nie | Otwarcie strony "Drużyny" po kliknięciu |
| 4 | var | Zawodnicy | biały | nie | Otwarcie strony "Zawodnicy" po kliknięciu |
| 5 | var | Mecze | biały | nie | Otwarcie strony "Mecze" po kliknięciu |
| 6 | var | Archiwum | biały | nie | Otwarcie strony "Archiwum" po kliknięciu |
| 7 | var | Tablica | biały | nie | Otwarcie strony "Tablica" po kliknięciu |
| 8 | var | Regulamin | biały | nie | Otwarcie strony "Regulamin" po kliknięciu |
| 9 | Div | RANKING | czarny | nie | Informacja dla użytkownika |
| 10 | Div | brak | czarny | nie | Ranking drużyn w postaci tabeli |
| 10a | Div | brak | czarny | nie | Dane statystyczne poszczególnych drużyn |
| 11 | Div | "zdjęcie profilowe" | czarny | nie | Zdjęcie profilowe |
| 12 | Div | Wyloguj | czarny | nie | Wylogowanie użytkownika |
| 13 | Div | Admin panel | czarny | nie | Przejście do panelu administratora |

3.1.6. Tablica



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr** | **Typ obiektu** | **Zdefiniowana treść** | **Kolor tekstu** | **Pole edytowalne** | **Funkcja** |
| 1 | var | League | biały | nie | Otwarcie strony głównej "League" po kliknięciu |
| 2 | var | Ranking | biały | nie | Otwarcie strony "Ranking" po kliknięciu |
| 3 | var | Drużyny | biały | nie | Otwarcie strony "Drużyny" po kliknięciu |
| 4 | var | Zawodnicy | biały | nie | Otwarcie strony "Zawodnicy" po kliknięciu |
| 5 | var | Mecze | biały | nie | Otwarcie strony "Mecze" po kliknięciu |
| 6 | var | Archiwum | biały | nie | Otwarcie strony "Archiwum" po kliknięciu |
| 7 | var | Tablica | biały | nie | Otwarcie strony "Tablica" po kliknięciu |
| 8 | var | Regulamin | biały | nie | Otwarcie strony "Regulamin" po kliknięciu |
| 9 | Div | Dodaj post | czarny | nie | Dodanie nowego posta przez administratora |
| 10 | Div | brak | czarny | nie | Ranking drużyn w postaci tabeli |
| 10a | Div | brak | czarny | nie | Dane użytkownika dodającego post |
| 10b | Div | Usuń | czarny | nie | Usuwanie posta |
| 10c | String | brak | czarny | nie | Szczegóły posta |
| 10d | Div | Komentarze | czarny | nie | Wyświetlanie komentarzy dotyczących posta |
| 11 | Div | "zdjęcie profilowe" | czarny | nie | Zdjęcie profilowe |
| 12 | Div | Wyloguj | czarny | nie | Wylogowanie użytkownika |
| 13 | Div | Admin panel | czarny | nie | Przejście do panelu administratora |