POLITECHNIKA ŁÓDZKA

Wydział Elektrotechniki, Elektroniki,

Informatyki i Automatyki

|  |
| --- |
| Praca dyplomowa inżynierska **Opracowanie aplikacji wspierającej proces decyzyjny przy zakładach bukmacherskich z wykorzystaniem sztucznych sieci neuronowych** Artur Widziński |

# Nr albumu: 195085

Opiekun pracy:

**Dr inż. Wojciech Zabierowski**

## Łódź, 2018

**Streszczenie**

Tematem tej pracy inżynierskiej jest opracowanie aplikacji, która służyłaby jako pomoc przy podejmowaniu decyzji związanych z robieniem zakładów bukmacherskich. Istotnym elementem było wykorzystanie sztucznych sieci neuronowych.

Ta praca opisuje technologie użyte w tej aplikacji webowej oraz proces jej powstawania, której podstawą stał się pomysł serwisu bukmacherskiego.

W pierwszym rozdziale zostaje opisany cel aplikacji, czemu ma służyć. Wypisany jest tam również zbiór użytych technologii oraz ich zastosowanie w tej aplikacji. Dodatkowo został tam dodany słownik pojęć, którymi autor pracy będzie się często posługiwał. Ostatnią częścią tego fragmentu jest przedstawienie pierwszego etapu tworzenia serwisu bukmacherskiego, oraz problemy, które wtedy występowały.

Kolejny rozdział poświęcony jest opisowi aplikacji: został tam pokazany schemat bazy danych. Z następnej części można się dowiedzieć jak wyglądają i działają wszystkie widoki oraz funkcjonalności serwisu. W ostatnich podpunktach autor wyjaśnia się jak zostały zaimplementowane matematyczne algorytmy zawarte w tym projekcie.

Trzeci rozdział opisuje doświadczenie, które miało na celu sprawdzić skuteczność aplikacji. Autor zamieścił tam założenia eksperymentu, jego przebieg oraz skutki, które z nich wyniknęły. Na ich podstawie zostały sformułowane wnioski dotyczące efektywności aplikacji oraz osób biorących udział w jej testowaniu.

Ostatni rozdział zawiera podsumowanie pracy. Można tam znaleźć wnioski autora po ukończeniu tego projektu oraz propozycje jego rozwoju.

Słowa kluczowe: sieci neuronowe, serwis bukmacherski, Java

**Creating the application supporting decision-making   
processes at bookmaker's bets using artificial neural networks**

**Abstract**

The topic of the following Engineering Thesis was to design application, which would help making decisions regarding making bets. One of the most crucial elements of the project was using neural network.

This work describes the technology used in web application and the process of developing it, which was based on the idea of bookmaker service.

In the first chapter the author ellaborates on the purpose of the application. Within we can find the collection of the technical terminology and their usage in this work. What is more you can find there vocabulary of terms which will be frequently used by the author. The last part of this chapter is presentation how the initial creation stages were carried out along with the first problems encountered.

The next chapter destination is description of application: you can find where database schema. From the next part you can learn how looks and works all the features that project is providing. From the last parts of this chapter you can learn how was implemented math algorithms.

Third unit describes experiment, which was invented for testing the application's efficiency. The author put there assumptions, process and results of this project. Based on application effects there were phrased conclusions about accuracy and people taking part In testing it.

The last chapter contains work summary. You can find there author’s conclusions after finishing the project and proposition of how to elaborate this service.

Key words: neural networks, bookmaker service, java

**Spis treści**

1. Wstęp 4

1.1. Użyte technologie i frameworki 5

1.2. Słownik pojęć 8

1.3. Początki projektu 9

2. Aplikacja webowa 11

2.1. Wzorzec MVC 11

2.2. Opis serwisu 12

2.3. Schemat bazy danych 14

2.4. Wygląd i działanie aplikacji 17

2.4.1. Główne menu 17

2.4.2. Panel użytkownika 21

2.4.3. Przygotowywanie meczy 23

2.4.4. Tworzenie kuponów 29

2.4.5. Rozgrywanie meczy 34

2.4.6. Statystyki 37

2.5. Algorytm obliczania kursów bukmacherskich 43

2.6. Algorytm symulujący rozgrywanie meczy 48

3. Doświadczenie 52

4. Podsumowanie i wnioski 56

5. Spis rysunków 57

6. Spis tabel 59

7. Spis równań 60

8. Bibliografia 61

1. Wstęp

Obecnie coraz większe pole w obszarach informatycznych wypełniają projekty wykorzystujące sztuczną inteligencję. Społeczeństwo nawet nie zdaje sobie sprawę, w jakich podstawowych czynnościach codziennego życia jest wykorzystywana AI (ang. Artificial Intelligence). Takim przykładowym urządzeniem jest GPS, który używa algorytmów sztucznej inteligencji do znalezienia najbardziej optymalnej trasy między punktem A i B.

Dlatego też, w niniejszej pracy zdecydowano się zaimplementować aplikację webową, która wykorzystywałaby sieć neuronową. Jako temat aplikacji wybrano serwis bukmacherski.

W tym programie sztuczna inteligencja służyła do symulowania wyników meczów piłki nożnej. Zaprojektowana w ten sposób aplikacja mogła się stać przydatny dla ludzi, którzy dopiero zaznajamiają się ze sposobem funkcjonowania tego typu stron, tj. pozwolić im na ocenienie stopnia ryzyka, które wiąże się z postawieniem zakładu na dany klub. Do tego celu w projekcie został dodany symulator meczów, za pomocą którego użytkownik może przetestować swoje umiejętności w przewidywaniu rezultatów gier.

Głównym celem opisanych tu działań jest zaimplementowanie takiej aplikacji, która w sposób możliwie dokładny byłaby w stanie przewidzieć wynik danego meczu.

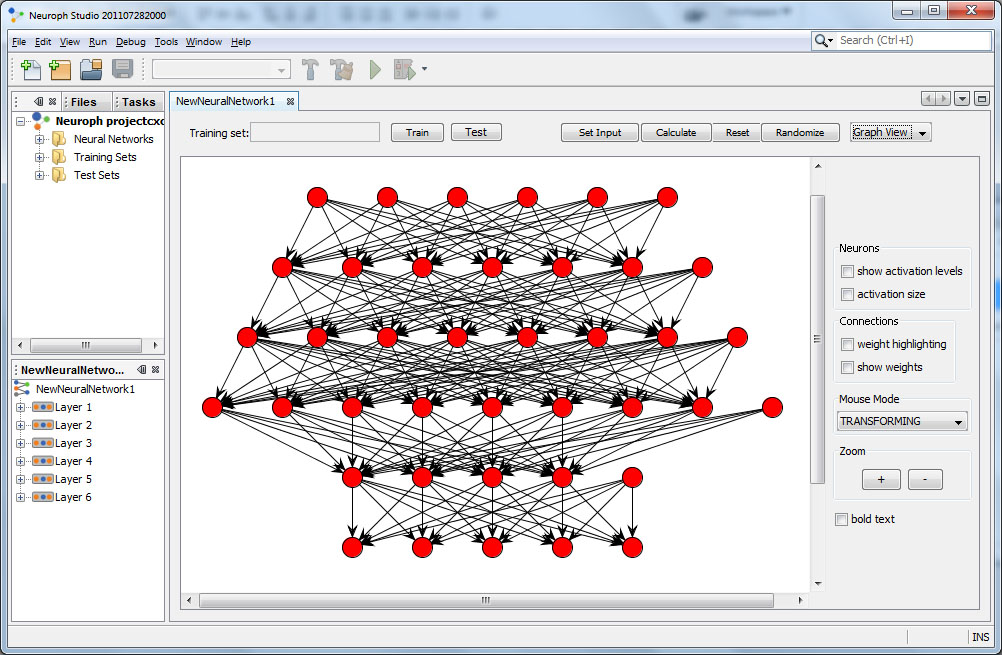
Wykorzystanie zakładów bukmacherskich jako przykładu aplikacji webowej może wydać się ryzykowną decyzją, szczególnie jest zostanie to rozpatrzone w kontekście hazardu. Jednakże cały proces decyzyjny towarzyszący tego typu sytuacjom umożliwia wykorzystanie coraz popularniejszej oraz częściej wykorzystywanej technologii Spring. Co więcej, możliwe jest połączenie jej z tematem sieci neuronowych.

Projekt ten miał za zadanie odpowiedzieć na pytanie „Czy możliwe jest przewidzenie za pomocą sztucznej inteligencji rezultatu meczu?” Odpowiedź na to pytanie znajduje się w końcowym fragmencie pracy.

* 1. Użyte technologie i frameworki

W niniejszym projekcie zostały użyte następujące technologie oraz frameworki:

* Spring Framework – pierwsze narzędzie do projektowania oraz implementowania aplikacji webowych, które zostało zaakceptowane przez większość programistów na świecie. Ten produkt jest ogólno dostępny od 2003 roku. Jego architektura pozwala na programowanie aplikacji nie tylko z użyciem języka programowania Java, można również używać platformy .NET. Jej głównym elementem jest kontener zarządzający komponentami. Kontener ten określa się IoC (ang. Inversion of Control), a wspomniany komponent „bean”. Podczas uruchamiania aplikacji kontener ma za zadanie tworzyć i wiązać ze sobą komponenty (ang. wiring) według konfiguracji, która jest zapisana w pliku XML. Takie rozwiązanie pozwala na zwolnienie logiki aplikacji z zadania tworzenia nowych obiektów. Mechanizmem odpowiedzialnym za sterowanie obiektami pomiędzy obiektami z nich korzystającymi w architekturze Spring jest tzw. wstrzykiwanie zależności (ang. Dependency Injection). Istnieje wiele sposobów na „wstrzyknięcie” jednego obiektu w drugi (np. poprzez konstruktor, interfejs czy metodę ustawiającą).
* Spring Boot – rozwiązanie oferowane przez twórców technologii Spring, bazuje na frameworku opisanym w poprzednim punkcie. Produkt ten powstał w 2013 roku i od tego momentu jest coraz częściej wykorzystywany. Tajemnicą tego sukcesu jest znaczące ułatwienie konfiguracji aplikacji. Nie trzeba ustawiać powiązań między komponentami w pliku XML, aby program działał. Ta zaleta sprawia, że implementacja aplikacji staje się o wiele szybsza i łatwiejsza. Dodatkowo Spring Boot posiada wszystkie komponenty potrzebne do uruchomienia aplikacji, np. serwer.
* MySQL – system do zarządzania relacyjnymi bazami danych. Większość dzisiejszych projektów napisanych za pomocą technologii Spring korzysta z baz danych Oracle, PostgreSQL lub właśnie MySQL. Baza Oracle nie posiada jednak licencji „Open-source”, co jest dużą wadą dla większości firm. Licencja „Open‑source” to legalne oprogramowanie, które pozwala na darmowe kopiowanie swoich produktów dla użytkowników. Dodatkowo daje im możliwość modyfikowania ich. W tym projekcie zdecydowano się na MySQL, ponieważ jest to baza, z którą autor miał już styczność, i wiedział jak połączyć swoją aplikację z tym systemem relacyjnych baz danych.
* Spring Security – framework umożliwiający zabezpieczenie aplikacji webowej poprzez sprawdzanie wierzytelności i autoryzacji użytkowników. Ten produkt powstał w 2003 roku, początkowo nosił nazwę „Acegi”. Jak większość produktów od twórców technologii Spring, swoją popularność zawdzięcza prostocie implementacji zabezpieczeń. Ten framework zapewnia również ochronę przed takimi atakami jak Session Fixation[[1]](#footnote-2), clickjacking[[2]](#footnote-3) czy CSRF[[3]](#footnote-4).
* Thymeleaf – szablony generujące kod HTML5, za pomocą których można w prosty sposób zaprezentować dane trzymane w bazie danych. Twórcy tego produktu zaprojektowali je głównie w celu integracji z technologią Spring. Aby móc go wykorzystać w plikach o rozszerzeniach .html bądź .jsp, wystarczy użyć odpowiednią przestrzeń nazw (zdefiniowana prefixem „th”). Daje wtedy dostęp do danych obiektów, które są przekazane przez kontroler do warstwy widokowej (wzorzec projektowy MVC zostanie opisany dokładniej w kolejnym rozdziale).
* Neuroph – biblioteka do implementacji sieci neuronowych w aplikacjach. Został on zaprojektowany przez twórców IDE Netbeans. Początkowo istniało osobne IDE do tworzenia różnych architektur sieci neuronowych, zwane Neuroph Studio. Posiada ono graficzny interfejs użytkownika, za pomocą którego można ustawić wszelkie parametry sieci.



Rysunek 1: Widok Neuroph Studio**[[4]](#footnote-5)**

* 1. Słownik pojęć

W tej pracy zostaną użyte terminy ściśle związane z opisywanym tematem. Ich znaczenie można znaleźć w poniższym spisie:

* Framework – szkielet aplikacji, na którym programuje się rozwiązania programistyczne.
* Uwierzytelnianie (inaczej autentykacja) – wyszukanie w systemie (bazie danych) użytkownika poprzez sprawdzenie poprawności danych osobowych (np. za pomocą hasła).
* Autoryzacja – operacja rozpatrująca możliwość udzielenia dostępu osobie o zdefiniowanej roli.
* API – interfejs do komunikacji pomiędzy aplikacjami. Pozwala ono na używanie udostępnionych metod od jednej z nich w drugiej.
* Zakłady bukmacherskie – zakłady losowe, w których przewiduje się rezultat spotkania sportowego.
* Bukmacher – osoba bądź instytucja prowadząca zakłady bukmacherskie.
* Kurs – współczynnik (mnożnik) określający możliwy zysk pieniężny. Przykładowo, kurs równy 2,00 oznacza, że za każde 10 złotych można otrzymać 20 złotych.
* Singiel (inaczej zdarzenie pojedyncze) - wydarzenie, które dotyczy pojedynczego meczu. W singlu zapisane są informacje o tym jaki rezultat użytkownik przypuszcza oraz kurs mu odpowiadający. Każdy zakład bukmacherski składa się z jednego lub wielu singli.
* Bundesliga - najwyższe w hierarchii rozgrywki piłkarskie mężczyzn w Niemczech.
* Skład wyjściowy (podstawowy) – zbiór piłkarzy w drużynie, którzy są od pierwszej minuty na boisku.
* Ławka rezerwowych – zbiór piłkarzy, którzy są przewidziani w składzie meczowym, ale nie ma ich na boisku od początku spotkania.
  1. Początki projektu

Ta aplikacja jako przykład danych wykorzystała rozgrywki Bundesligi począwszy od sezonu 2015/2016 (od sierpnia 2015 roku). Pierwszym zadaniem w tym projekcie było zebranie i zapisanie danych na temat tej ligi. Musiały się w nich znaleźć:

* Drużyny uczestniczące w tych rozgrywkach,
* Piłkarze reprezentujący dany klub oraz ich statystyki, między innymi:
  + Ilość rozegranych meczy,
  + Liczba strzelonych bramek / meczów bez straconej bramki (tzw. „czyste konta”),
  + Wartość rynkowa.
* Mecze rozgrywane w ramach Bundesligi, w tym:
  + Wynik meczu,
  + Zespoły w nim uczestniczące,
  + Piłkarze występujący w danym spotkaniu,
  + Zaistniałe wydarzenia wraz z zapisaną minutą spotkania (gol, zmiana).

Pierwszym pomysłem na zgromadzenie tych danych było znalezienie strony, która świadczyła API z pożądanymi informacjami. Jednakże każda witryna, która posiadała poszukiwane wiadomości oferowała je, ale tylko w zamian za zasoby finansowe. Z tego powodu postanowiono wybrać inny sposób stworzenia bazy danych.

Program WebHarvy[[5]](#footnote-6) pozwala na spisanie danych widocznych na dowolnej stronie (taka operacja określa się „scrappowaniem”) i zapisanie ich w dowolnym formacie (XML, JSON czy XML). Z pomocą wspomnianego produktu postanowiono zebrać informacje, które widnieją na jednej ze stron piłkarskich[[6]](#footnote-7). W pierwszych momentach użytkowania wyżej wymienionego programu taki sposób wydawał się dobrym rozwiązaniem. Udało się bowiem zapisać wszystkich piłkarzy w dość krótkim czasie. Jednak pobranie tą metodą wszystkich szczegółów dotyczących meczy pochłonęło zdecydowanie za dużo dni, przy miernych efektach.

Dlatego postanowiono powrócić do poprzedniego planu. Udało się zlokalizować jedną z mniej wymagających finansowo stron[[7]](#footnote-8), oferujące poszukiwane informacje. Po pokryciu wszystkich kosztów udało się zapisać pozostałe dane.

Kiedy już cała baza danych była dostępna, pojawił się kolejny problem. Okazało się, że dane zapisane z API nie posiadały znaków specjalnych. Dlatego np. piłkarz, który według zescrappowanych informacji nazywał się *„Jérôme Boateng”* był innym zawodnikiem niż *„Jerome Boateng”*. Wykrycie oraz poprawienie tych nieścisłości również zajęło trochę czasu.

Po zsynchronizowaniu danych przyszła kolej na zbudowanie webowej aplikacji. Zaprogramowanie serwisu bukmacherskiego odbyło się bez znacznych problemów.

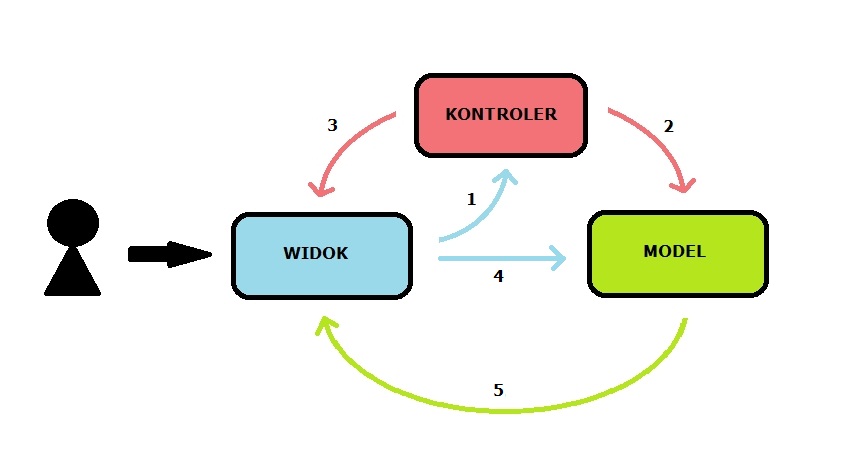
1. Aplikacja webowa

W tym rozdziale zostanie opisana aplikacja webowa, jej funkcjonalności dla klienta i administratora oraz sposób działania.

* 1. Wzorzec MVC

Projekt serwisu bukmacherskiego został zaprojektowany zgodnie z wzorcem projektowym Model-View-Controller (MVC). To rozwiązanie składa się z trzech osobnych części:

* Model – element logiczny aplikacji,
* Widok – miejsce prezentacji danych za pomocą graficznego interfejsu,
* Kontroler – element odpowiadający za wykonywanie operacji na danych wejściowych, które wywołuje użytkownik dzięki graficznemu interfejsowi, i następne przesłanie zaktualizowanych modelów do warstwy widokowej.



Rysunek : Schemat wzorca projektowego MVC[[8]](#footnote-9)

Taki wzorzec projektowy jest jedną z najczęściej wykorzystywanych koncepcji podczas tworzenia aplikacji webowych.

* 1. Opis serwisu

Strona bukmacherska jest aplikacją skierowaną dla ludzi interesujących się rozgrywkami sportowymi. Wszystkich użytkowników aplikacji można podzielić na kilka grup, którym przypisane są poniższe role:

* Administrator (inaczej opiekun serwisu),
* Zalogowany użytkownik,
* Niezalogowany użytkownik.

Administrator jest główną osobą, która czuwa nad poprawnością działania serwisu oraz wprowadza dane związane z nadchodzącymi meczami. Zalogowany użytkownik tworzy kupony, za pomocą których zakłada przewidywany rezultat meczu. Większość usług jest dostępnych dla wszystkich zalogowanych osób, jednakże jako użytkownik niezalogowany mamy również kilka możliwości.

Oto najważniejsze funkcjonalności przeznaczone dla administratora:

* Stworzenie meczu,
* Dodanie składu drużyny do meczu,
* Rozegranie meczu.

Z materialnego punktu spojrzenia udostępniono w aplikacji funkcjonalności odpowiadające za zarządzanie kuponami. Klient może sprawdzić jak wygląda jego sytuacja finansowa, a administrator może sprawdzić stan konta każdego zalogowanego użytkownika. Takie usługi są dostępne w tym celu:

* Doładowanie konta,
* Stworzenie kuponu,
* Sprawdzenie stanu konta.

Serwis bukmacherski jest stroną, która musi mieć dostęp do dużej ilości danych na temat rozgrywanych meczy. Jest to ważne z punktu widzenia klienta, który decyduje się, czy postawić zakład na daną drużynę. Chętnie sprawdzi np. jakie wyniki klub osiągał w przeciągu kilku ostatnich gier, jakich piłkarzy posiada czy też na którym miejscu w lidze się znajduje. Dlatego aplikacja oferuje dużą ilość różnych możliwości skierowanych głównie dla klienta. Oto niektóre z nich:

* Sprawdzenie tabeli ligi,
* Sprawdzenie danych drużyny,
* Sprawdzenie poprzednich wyników klubu,
* Zobaczenie detali meczu.
  1. Schemat bazy danych

W bazie danych tej aplikacji znajdują się tabele, w których aplikacja przechowuje dane potrzebne do poprawnego funkcjonowania, a także dodaje, modyfikuje i usuwa kolejne rekordy. Występują tam również tabele, z których aplikacja nie korzysta. Związane jest to z tym, że aplikacja będzie jeszcze rozwijana.

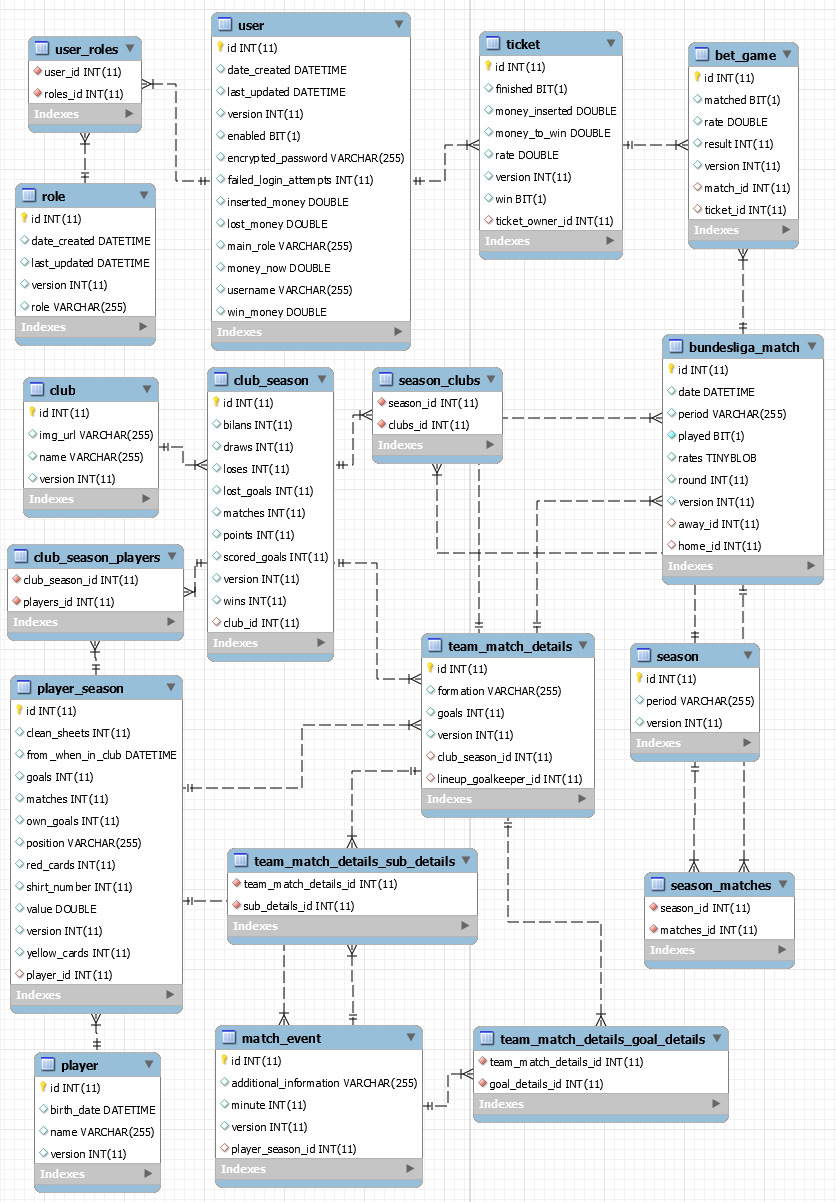
Podstawową tabelą w tej aplikacji jest tabela **USER**. Przechowuje ona dane użytkownika, takie jak nazwa użytkownika, zaszyfrowane hasło, ilość pieniędzy na koncie czy rolę użytkownika. Domyślnie po rejestracji każdy nowy klient należy do grupy użytkowników o roli *„user”*. Jedynym użytkownikiem, który nie należy do tej grupy jest administrator serwisu.

Każdy użytkownik ma przypisaną listę kuponów. Są wśród nich te, które zostały już rozstrzygnięte, oraz te, które istnieją w bazie przez krótki okres czasu i ich rezultat nie jest jeszcze znany. Wszystkie te zakłady przetrzymywane są w tabeli **TICKET**. Każdy kupon przechowuje informacje między innymi o tym ile użytkownik poświęcił na niego pieniędzy, ile może wygrać oraz czy dany kupon został już rozstrzygnięty. A do każdego z nich jest przypisanych kolekcja singli. Ten zbiór danych znajduje się w tabeli **BET\_GAME**.

Najbardziej zewnętrzną tabelą trzymającą informacje o rozgrywkach Bundesligi jest tabela **SEASON**. Każdy rekord przechowuje lata trwania, mecze rozegrane w tym okresie oraz kluby, które w nich uczestniczyły. Rekordy z tabeli **CLUB\_SEASON** przechowują dane o tym jak drużyny spisywały się w danym sezonie (ilość wygranych, przegranych, remisów, strzelonych i straconych bramek). Dodatkowo do każdej encji z tej tabeli przypisana jest lista zawodników, którzy wtedy należeli do danego klubu. Każdy z nich jest zapisany w tabeli **PLAYER\_SEASON**. Zawiera ona zarówno informacje personalne piłkarza (imię i nazwisko, data urodzenia) jak i dane piłkarskie (numer na koszulce, pozycja, wartość rynkowa, ilość meczy, strzelonych bramek oraz meczy bez straconej bramki bramkarzy).

Tabela **BUNDESLIGA\_MATCH** przechowuje encje wszystkich meczy, które się już odbyły oraz tych, które zostały już zaplanowane. Można tutaj znaleźć datę danej rozgrywki, wyliczone kursy na dany mecz oraz dane składów obu drużyn, które są trzymane w tabeli **TEAM\_MATCH\_DETAILS**. Każdy rekord zawiera informacje dotyczące piłkarzy składu wyjściowego oraz ławki rezerwowych, ilość strzelonych goli, a także zdarzenia w meczu dotyczące drużyny. Te ostatnie znajdziemy w tabeli **MATCH\_EVENT**. Jest tam zapisana minuta wydarzenia, piłkarz, którego ono dotyczy oraz dodatkowa krótka informacja, która dokładnie opisuje co to za rodzaj zdarzenia (np. gol, zmiana, samobój).

Bardziej szczegółowy schemat tej bazy danych przedstawia rysunek na następnej stronie.



Rysunek 3: Schemat bazy danych

* 1. Wygląd i działanie aplikacji

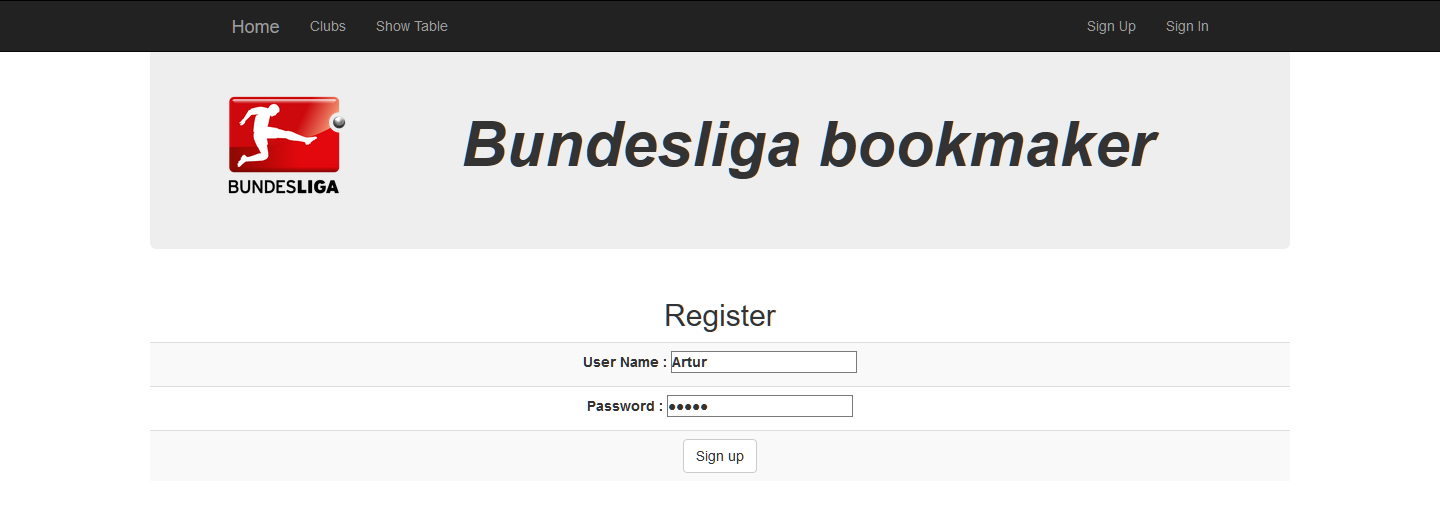
W tym podrozdziale zostaną przedstawione oraz omówione poszczególne strony aplikacji.

* + 1. Główne menu

Po uruchomieniu serwisu zostaje pokazana główna strona serwisu.

Rysunek : Strona główna

W tym momencie użytkownik nie jest zalogowany. Dla serwisu oznacza to, że dany użytkownik nie jest zautoryzowany, czyli nie ma dostępu do większości usług. Aby to zmienić, klient musi się zarejestrować, a następnie zalogować.

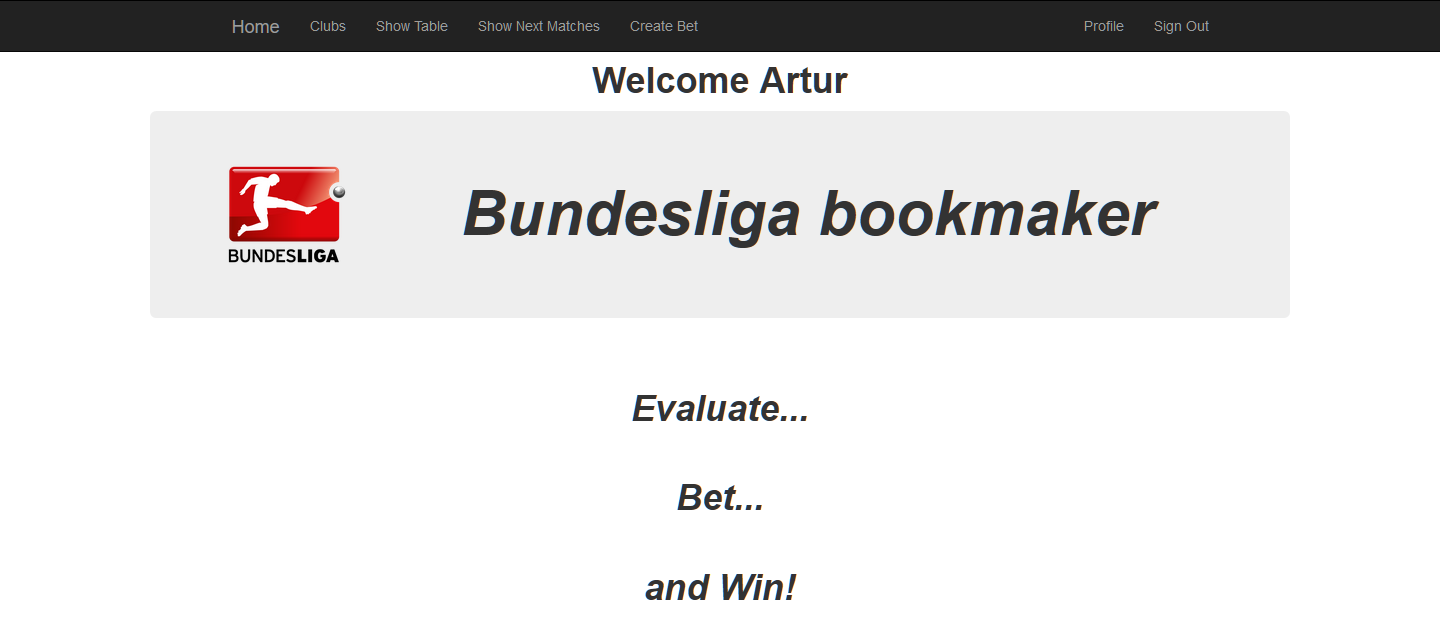


Rysunek : Ekran rejestracji

##### Logowanie.png

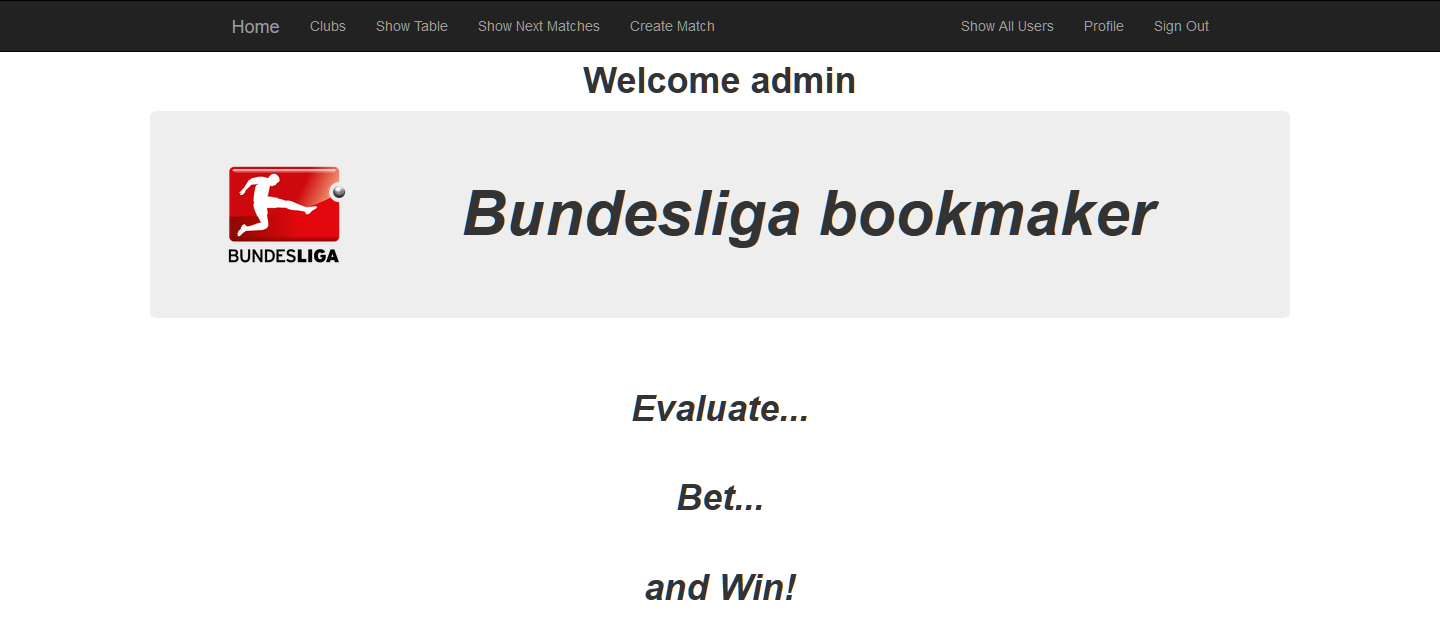
Rysunek : Ekran logowania

Po poprawnym zalogowaniu przed użytkownikiem ponownie pojawia się główna strona. Jednak tym razem system rozpoznaje użytkownika jako osobę uwierzytelnioną, dlatego udostępnia mu więcej możliwości.



Rysunek : Ekran główny zalogowanego użytkownika

Zmiany można zauważyć w górnym pasku menu. Uwierzytelniony użytkownik może teraz skorzystać z takich zakładek jak „Show next matches”, „Create bet” czy „Profile”. Nowy użytkownik domyślnie należy do grupy klientów z rolą *„user”*. Natomiast administrator systemu jest jedyną osobą z rolą *„admin”*. Jego ekran główny posiada kilka funkcji, do których zwykły użytkownik nie ma dostępu.



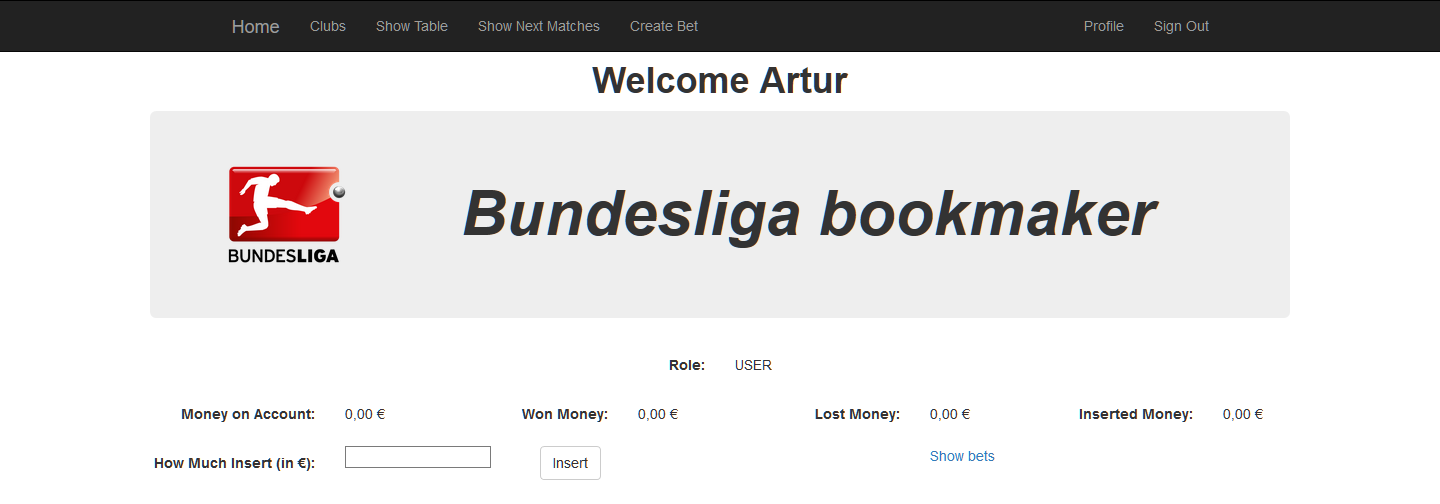
Rysunek : Ekran główny administratora

Na rysunku nr 8 można zauważyć, że administrator systemu ma kilka dodatkowych funkcji, np. „Create match” czy też „Show all users”. Taką różnicę w dostępności usług w serwisie uzyskano dzięki autoryzacji. Proces ten pozwala na rozpoznanie roli danego użytkownika i na tej podstawie decyduje, z których możliwości strony może on skorzystać.

Uwierzytelnianie i autoryzacje użytkowników można uzyskać dzięki bibliotece Spring Security. W projekcie serwisu bukmacherskiego te dwa procesy są wykorzystywane na każdej stronie. Ten framework zabezpiecza serwis również przed wchodzeniem na strony niedostępne dla użytkownika o niższej w hierarchii serwisu roli. Strona bukmacherska posiada ochronę w sytuacji, w której jeden użytkownik próbuje wcielić się w drugiego.

* + 1. Panel użytkownika

Zalogowany klient ma dostęp do wglądu w swój profil. Po naciśnięciu przycisku „Profile” GUI (graficzny interfejs użytkownika) przekazuje do kontrolera żądanie wykonania usługi. W tym celu kontroler pobiera z bazy danych informacje o tym użytkowniku, a następnie przekazuje je do warstwy widokowej, która wyświetla wszystko w określony sposób (Rysunek 9). W taki sposób wygląda komunikacja klient-serwer w tej aplikacji.



Rysunek : Profil użytkownika

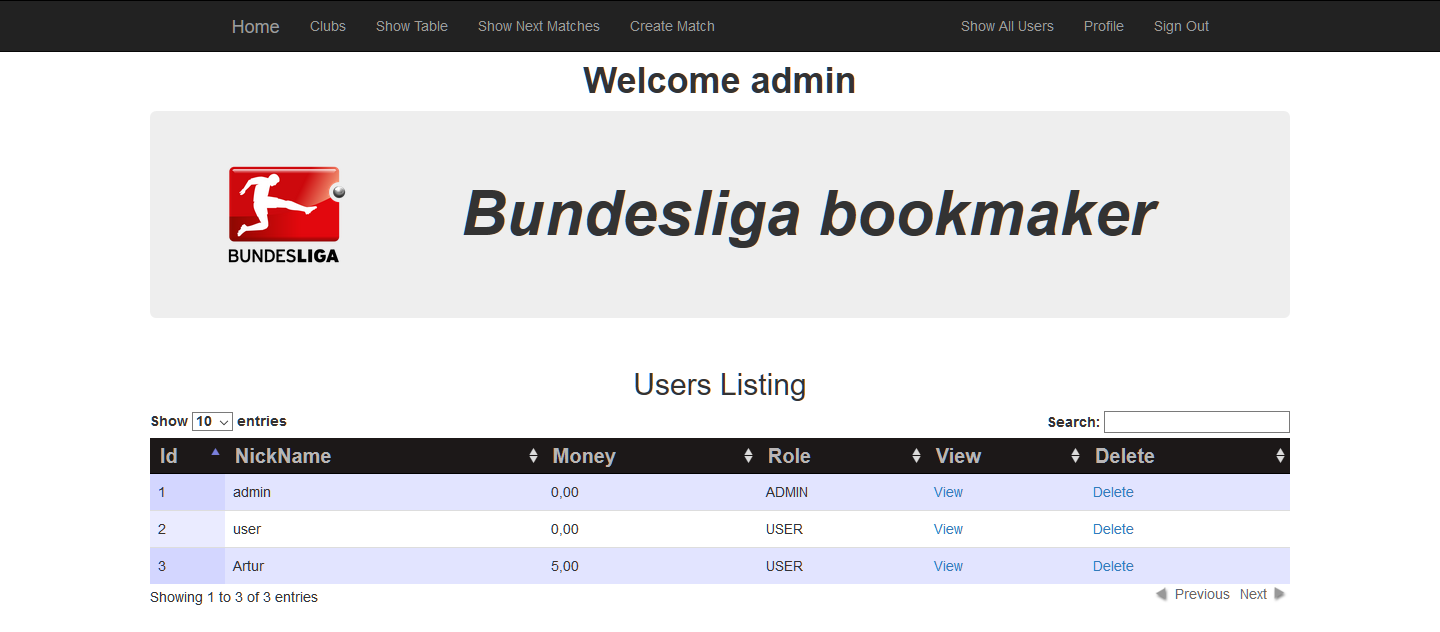
Tutaj znajdują się informacje o danym kliencie. Może on zobaczyć do jakiej grupy użytkowników należy (czyli jakie ma uprawnienia), sprawdzić stan swojego konta pieniężnego oraz zobaczyć historię własnych kuponów.

W tym serwisie bukmacherskim nie używa się własnych pieniędzy, zgodnie z tym, co autor wcześniej napisał. Ta aplikacja jest dla klientów sprawdzianem swoich umiejętności, próbą generalną przed postawieniem pierwszego zakładu bukmacherskiego za pomocą finansów z własnego konta bankowego. Aby użytkownik mógł doładować swoje konto, wystarczy że wpiszę w pole tekstowe liczbę, i naciśnie przycisk *„Insert”*. Po tej operacji profil klienta zostanie zaktualizowany, zwiększy się ilość pieniędzy wyświetlana przy tekstach *„Money on Account”* oraz *„Inserted Money”*.

Nie każdego stać, żeby na każdy kupon poświęcać duże kwoty pieniężne. Dlatego ciągłe doładowywanie konta tysiącami złotych może odebrać realizm z zakładów bukmacherskich. Takie zachowanie będzie jednak złe tylko dla klienta. Jeśli nawet w tym serwisie uda mu się zarobić więcej pieniędzy niż wpłacił za pomocą jednego kuponu o wpisowej kwocie np. milion złotych, to po rozpoczęciu przygody z realnymi zakładami szybko zobaczy, że takiego samego zysku prawie na pewno nie osiągnie.

Dodatkowymi polami widocznymi w profilu klienta są *„Won Money”* oraz *„Lost Money”*. Te wartości informują o ilości wygranych i przegranych pieniędzy. Zestawienie obok siebie tych dwóch kwot jest celowe. Szybkie porównanie tych wartości pozwala na postawienie dosyć trafnej opinii na temat własnych umiejętności w zakładach bukmacherskich.

Jako administrator mamy dostęp do profili wszystkich użytkowników.

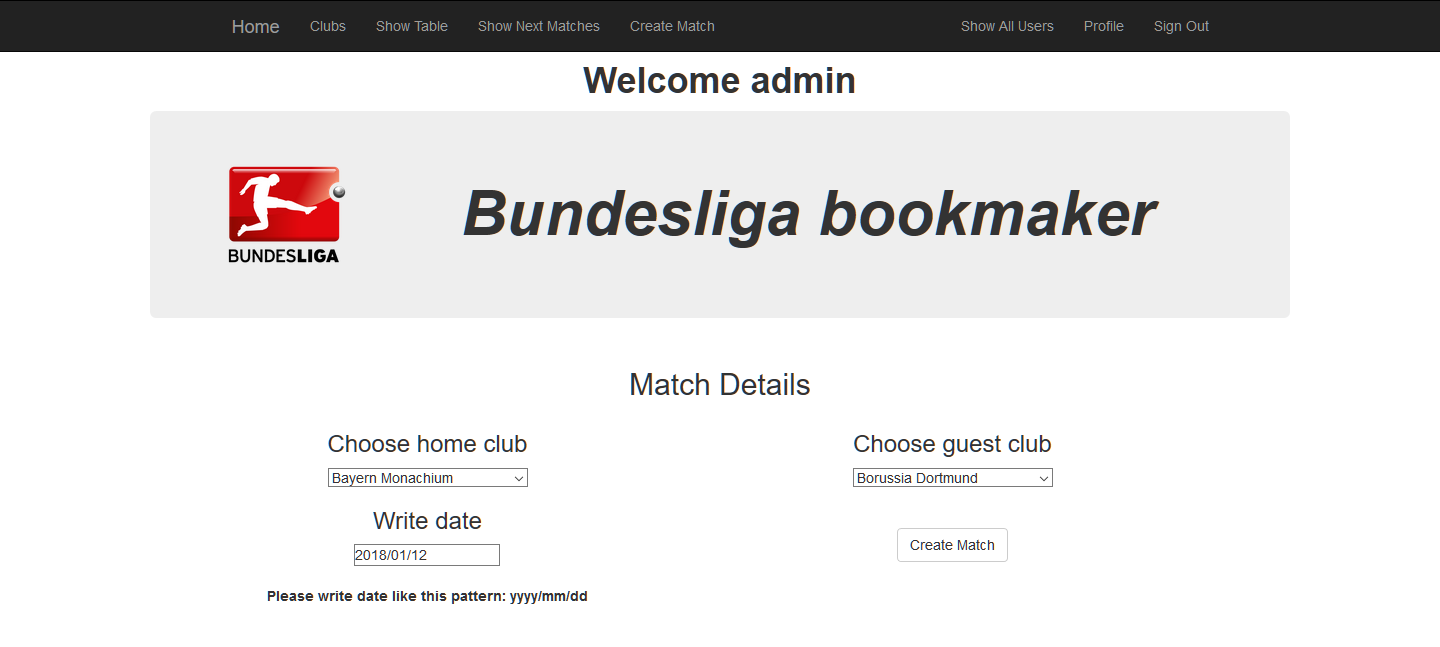


Rysunek : Widok wszystkich użytkowników

W tym panelu można zobaczyć ilość aktualnych pieniędzy na koncie klientów. Istnieje tutaj możliwość usunięcia użytkownika, a także sprawdzenia tych samych danych, które były pokazane na poprzednim rysunku (Rysunek 9). Jedyna różnica w wyświetleniu tej strony polega na tym, iż administrator nie może doładować konta żadnego z użytkowników.

* + 1. Przygotowywanie meczy

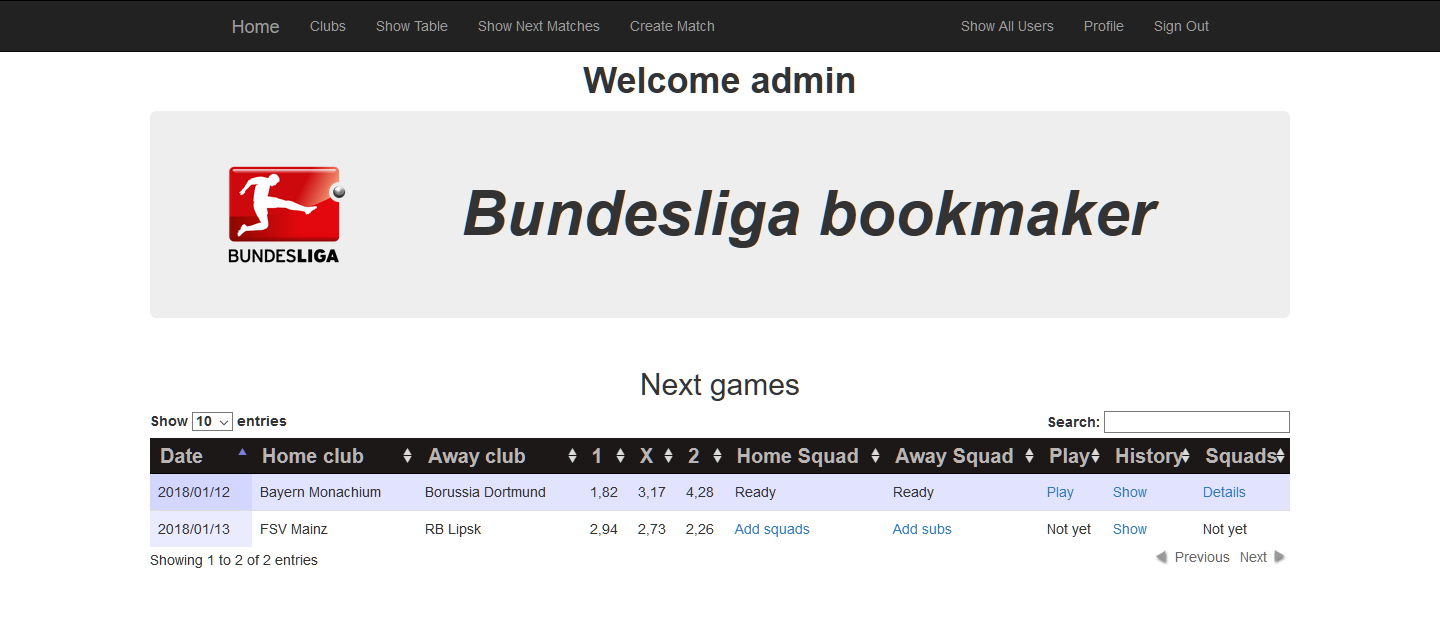
Kiedy po raz pierwszy zostanie naciśnięty przycisk „Show next matches”, pojawi się nowe okno, na którym jest napisana informacja o braku w bazie danych meczy nierozegranych. Aby to zmienić, musi zostać ogłoszona następna gra. W tym serwisie bukmacherskim tylko administrator posiada taką możliwość. Po wybraniu przycisku „Create match” z górnego menu strona wyświetla taką zawartość, jak w rysunku poniżej (Rysunek 11).



Rysunek : Tworzenie meczu

W tym panelu opiekun serwisu wybiera które drużyny zagrają ze sobą w tym meczu. Nie jest obojętne który klub zostanie wybrany pod polem *„Choose home club”* i *„Choose away club”*, ponieważ algorytmy liczące kursy na dany mecz i symulujące daną grę operują między innymi na tym, która drużyna gra na swoim stadionie (ich szczegółowy opis zostanie opisany w późniejszej części pracy).

Po wpisaniu daty i naciśnięciu przycisku *„Create match”* aplikacja sprawdzi poprawność przesłanych danych (np. czy data jest podana w odpowiednim formacie, bądź klub nie gra sam ze sobą), następnie algorytm wyliczy kursy na ten mecz, a na końcu serwis przekieruje użytkownika do strony z nierozegranymi meczami (Rysunek 12).

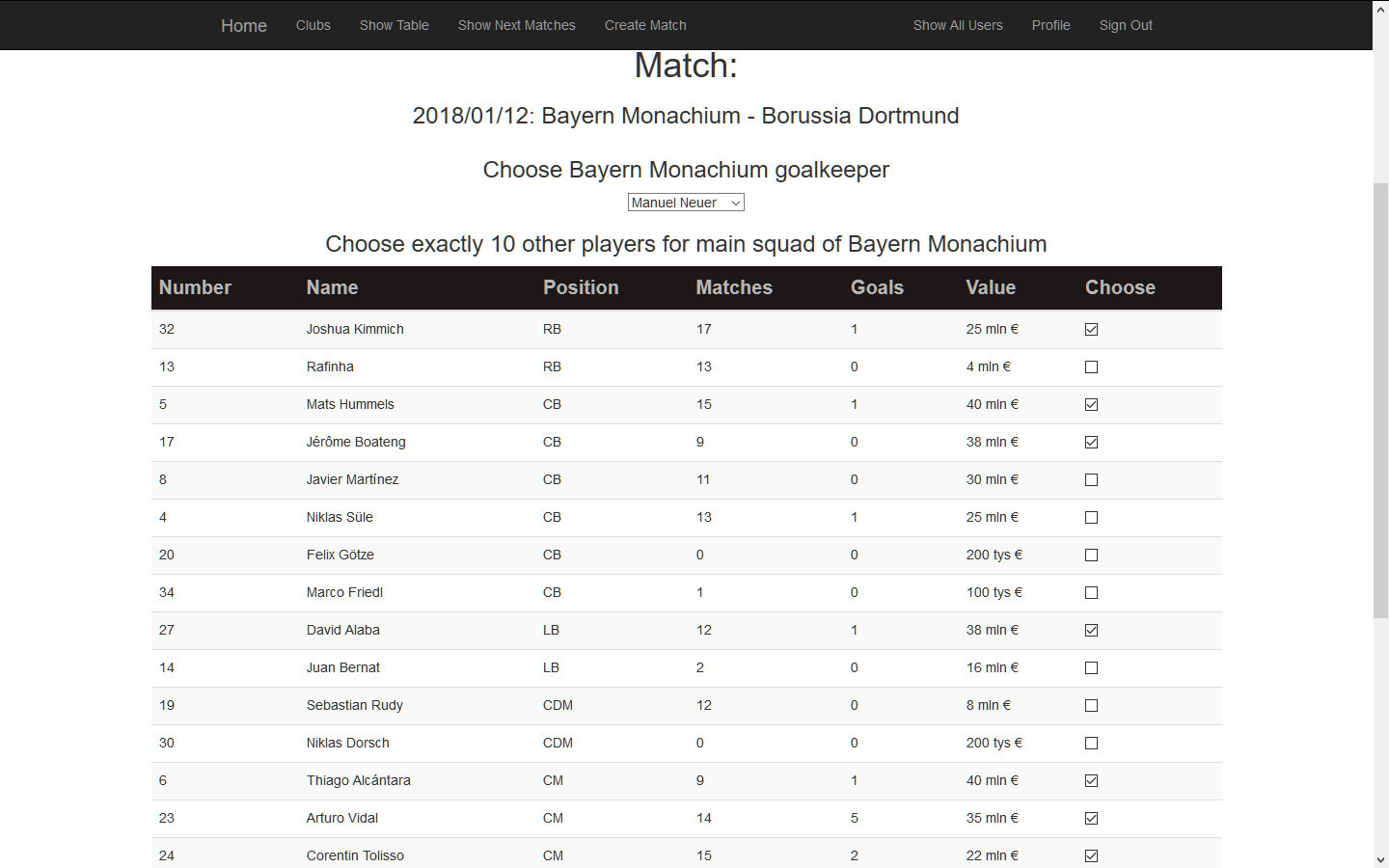


Rysunek : Następne mecze

Na tej stronie widać tabelę z dużą ilością kolumn. Pierwsze trzy z lewej zostały wybrane w poprzednim kroku (Rysunek 11). Pozostałe z nich:

* 1 – kolumna pokazująca kurs na wygranie drużyny gospodarzy,
* X – kolumna pokazująca kurs na remis w tym spotkaniu,
* 2 – kolumna pokazująca kurs na wygranie drużyny gości,
* Home Squad – kolumna widoczna tylko dla administratora, znajduje się tutaj informacja, czy skład wyjściowy i rezerwowy drużyny gospodarzy został już dodany i zapisany w bazie danych. Może tutaj się znaleźć jedna z opcji:
  + Add squads – link do dodania piłkarzy składu wyjściowego,
  + Add subs – link do dodania piłkarzy z ławki rezerwowych,
  + Ready – informacja potwierdzająca zapisanie wszystkich wymaganych informacji na temat drużyny gospodarzy w tym meczu.
* Away Squad – analogicznie do kolumny „Home Squad”, dotyczy drużyny gości,
* Play – kolumna widoczna tylko dla administratora, mówiąca czy można już zasymulować wynik tego meczu. Ta kolumna może zawierać:
  + Not yet – co najmniej jedna z dwóch drużyn nie jest jeszcze gotowa do rozegrania meczu (nie ma w kolumnie „Home/Away Squad” napisu „Ready”),
  + Play – link do strony rozgrywającej ten mecz.
* History – kolumna zawierająca link, który prowadzi do strony wyświetlającej historię meczy rozgrywanych pomiędzy tymi dwoma drużynami,
* Squads – kolumna pokazująca dostępność zobaczenia składów wyjściowych oraz ławek rezerwowych. Zawiera ona pole:
  + Not yet – tak samo jak pole „Not yet” w kolumnie „Play”,
  + Details – link do strony pokazującej składy obu drużyn na to spotkanie.

Kolejnym etapem w przygotowywaniu meczu jest dodanie piłkarzy, którzy zagrają od początku spotkania na boisku. Do tego używa się przycisku *„Add Squad”*.



Rysunek : Wybieranie graczy składu wyjściowego

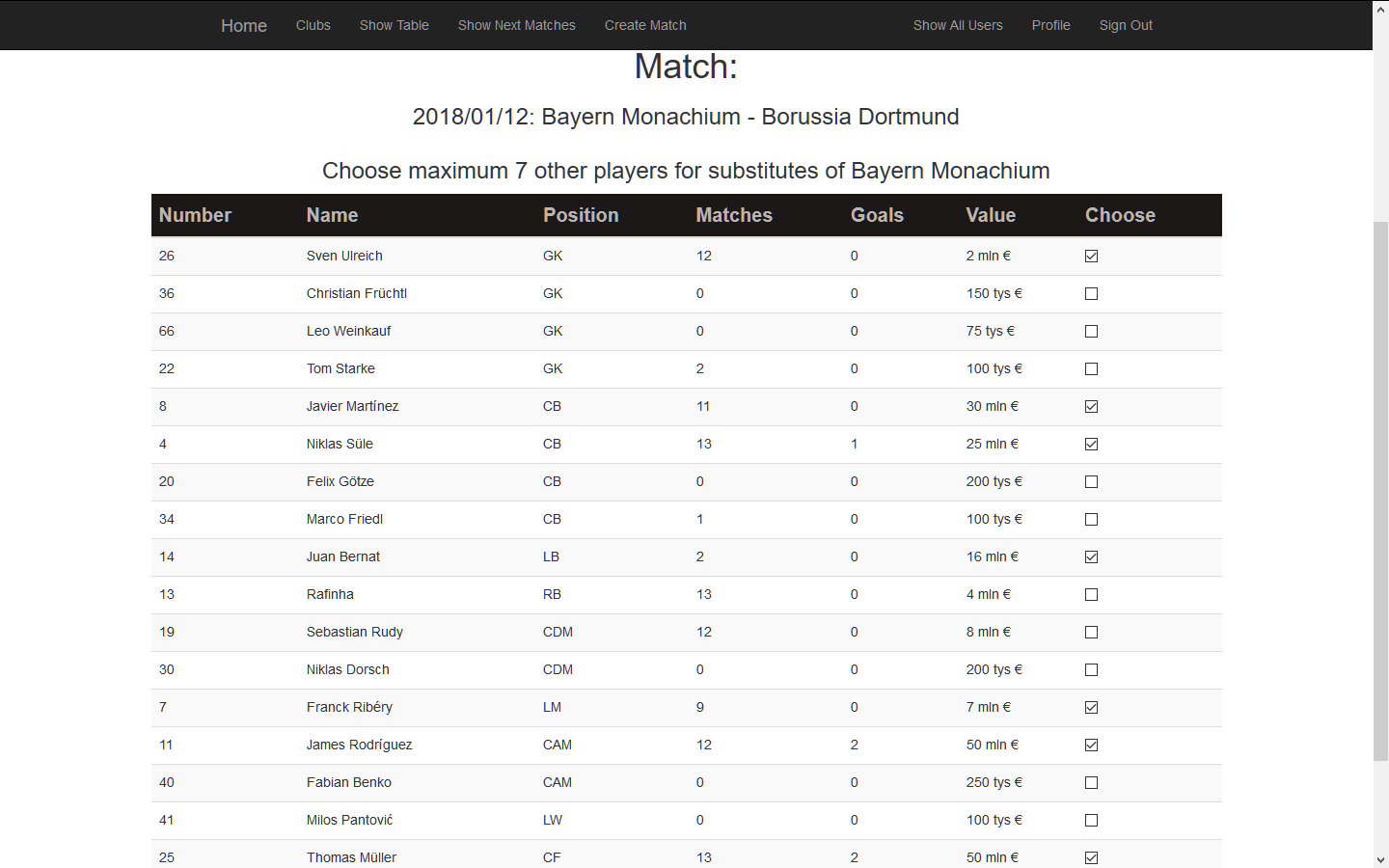
Na ekranie pojawili się gracze z drużyny, którą opiekun wybrał w poprzednim oknie. Aplikacja wyciągając ich z bazy danych podzieliła ich na dwie grupy: piłkarze grający na pozycji bramkarza i pozostali zawodnicy. Ten zabieg zapewnił to, iż w każdym podstawowym składzie będzie co najmniej jeden bramkarz. Z listy rozwijanej administrator wybiera zawodnika z tej pozycji. Niżej znajdują się wszyscy pozostali piłkarze tej drużyny. W tej tabeli znajdują się kolejno numer na koszulce, imię i nazwisko zawodnika, podstawowa pozycja na boisku, ilość rozegranych meczy w sezonie, ilość goli w sezonie oraz wartość rynkowa tego gracza. Pozycja każdego z nich jest zapisana w skrócie, którego rozwinięcie można znaleźć w poniższej tabeli.

Tabela : Rozwinięcie skrótów pozycji na boisku

|  |  |  |  |
| --- | --- | --- | --- |
| Skrót | Angielskie rozwinięcie | Polskie znaczenie | Formacja |
| GK | Goalkeeper | Bramkarz | Bramkarz |
| RB | Right Back | Prawy obrońca | Obrońca |
| CB | Center Back | Środkowy obrońca | Obrońca |
| LB | Left Back | Lewy obrońca | Obrońca |
| CDM | Center Defensive Midfield | Defensywny pomocnik | Pomocnik |
| RM | Right Midfield | Prawy pomocnik | Pomocnik |
| CM | Center Midfield | Środkowy pomocnik | Pomocnik |
| LM | Left Midfield | Lewy pomocnik | Pomocnik |
| CAM | Center Attacking Midfield | Ofensywny pomocnik | Pomocnik |
| RW | Right Wing | Prawy skrzydłowy | Napastnik |
| LW | Left Wing | Lewy skrzydłowy | Napastnik |
| CF | Center Forward | Cofnięty napastnik | Napastnik |
| ST | Striker | Środkowy napastnik | Napastnik |

Spośród wszystkich wypisanych piłkarzy administrator wybiera dziesięciu, którzy rozpoczną mecz zaznaczając okienko po prawej stronie, a na końcu wybiera przycisk *„Choose”*. Wtedy aplikacja zapisuje wybranego bramkarza oraz zaznaczonych piłkarzy w bazie, przekierowywuje do poprzedniej strony, a tam, gdzie wcześniej opiekun kliknął przycisk *„Add Squad”* zmieniła się pozycja na *„Add subs”*.

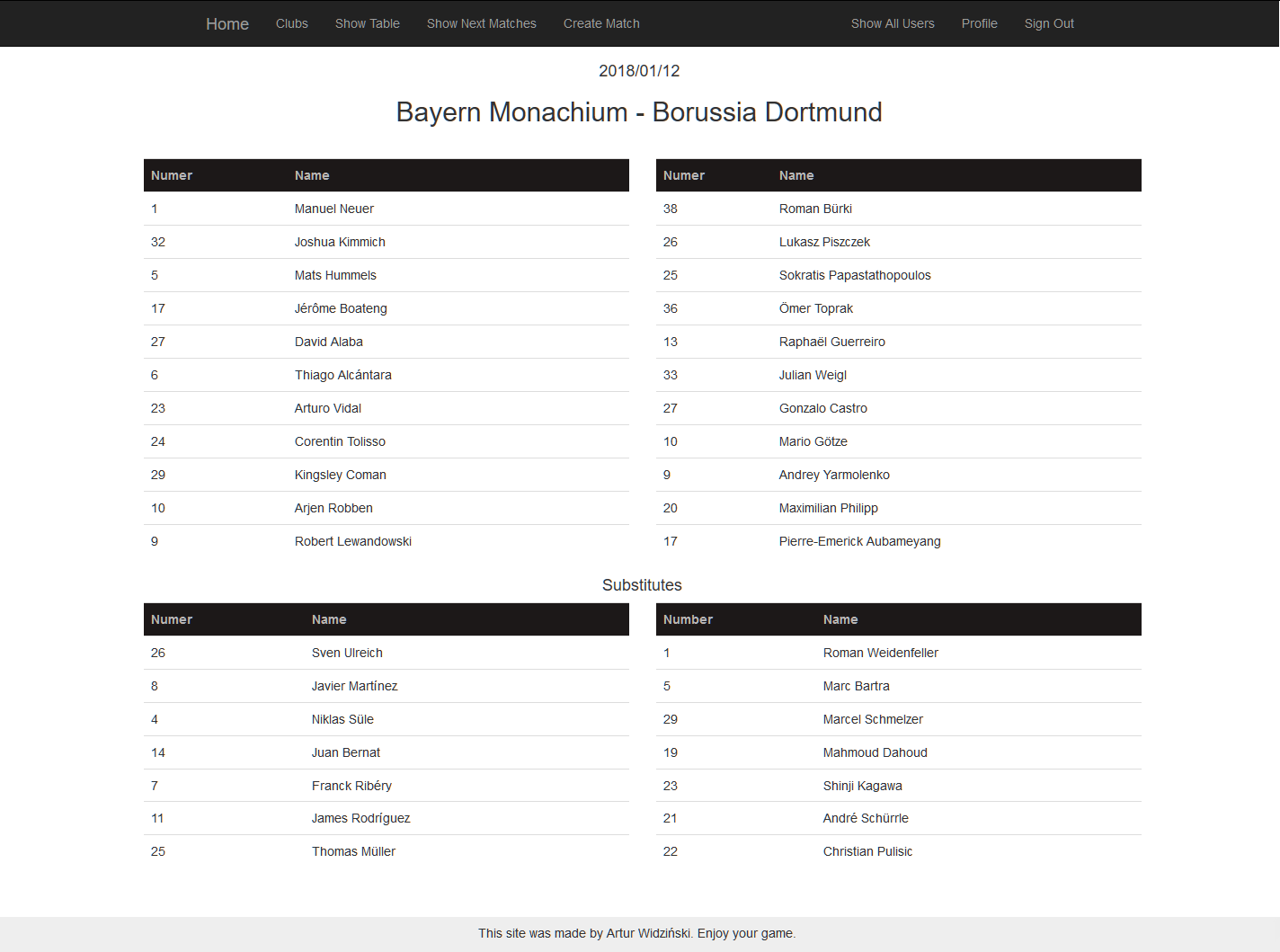
Następnym etapem przygotowania drużyn do meczu jest wybranie piłkarzy, którzy usiądą na ławce rezerwowych.



Rysunek : Wybieranie graczy do ławki rezerwowych

Ten etap wygląda prawie tak samo jak wybór graczy podstawowego składu, z jedną różnicą. Tutaj aplikacja pobierając piłkarzy z bazy danych odrzuciła wszystkich tych, którzy już są dodani do tego meczu, i nie robi podziału na bramkarzy i pozostałych zawodników. Stało się tak, ponieważ według zasad rozgrywek piłki nożnej w Niemczech nie ma ustalonej liczby ile maksymalnie bramkarzy może być wybranych do rezerwy. Jedyne ograniczenie jakie nakładają przepisy, to łączna liczba piłkarzy na ławce. Maksymalna ilość takich zawodników nie może być większa niż siedmiu.

Po wybraniu odpowiednich piłkarzy administrator zatwierdza swój wybór, a aplikacja go przenosi do strony z nierozegranymi meczami. W kolumnie, gdzie opiekun wcześniej nacisnął przycisk widnieje teraz napis *„Ready”*. Oznacza to, że ta drużyna jest już gotowa do rozegrania meczu. Po wykonaniu tych samych operacji dla drugiej drużyny w kolumnach „Play” oraz „Squads” aplikacja odblokuje możliwość rozegrania meczu oraz zobaczenia składów obu drużyn. Przebieg pierwszej z nich zostanie omówiony w jednym z kolejnych podrozdziałów. Widok składów obu drużyn w danym meczu wygląda następująco:

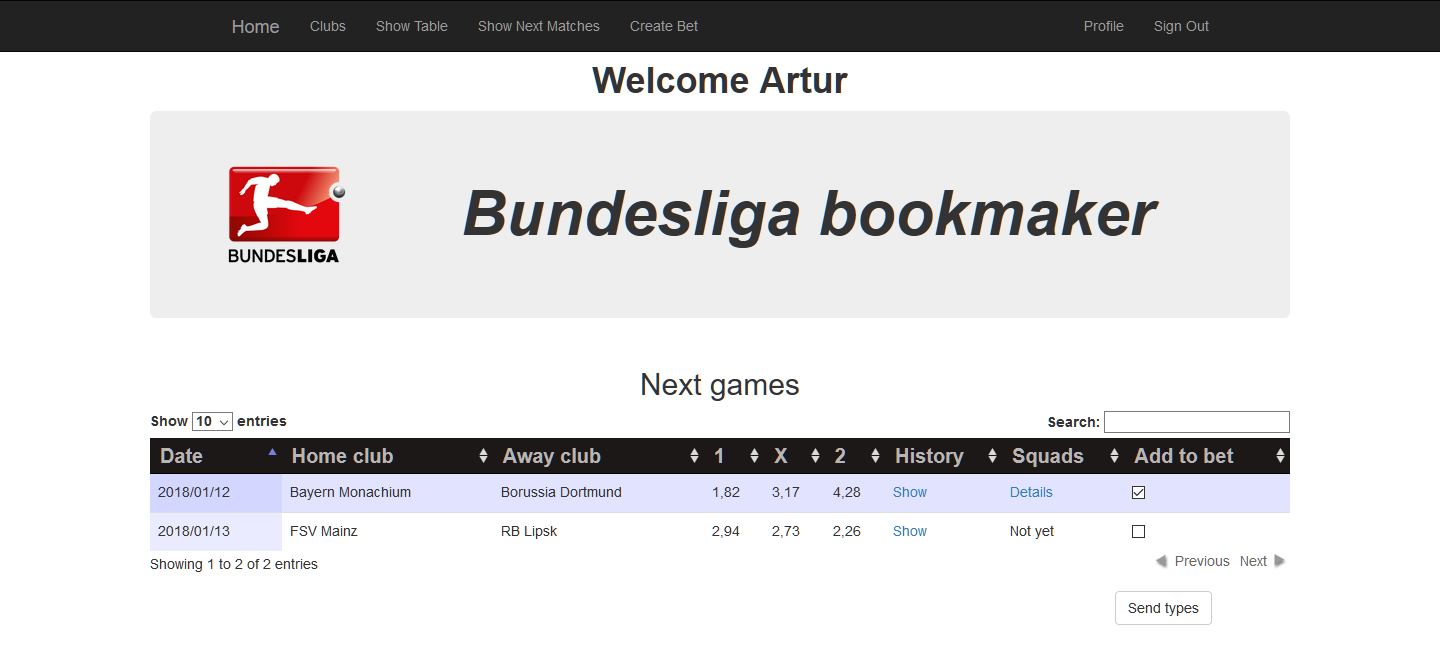


Rysunek : Składy meczowe

W tym momencie mecz jest gotowy do rozegrania.

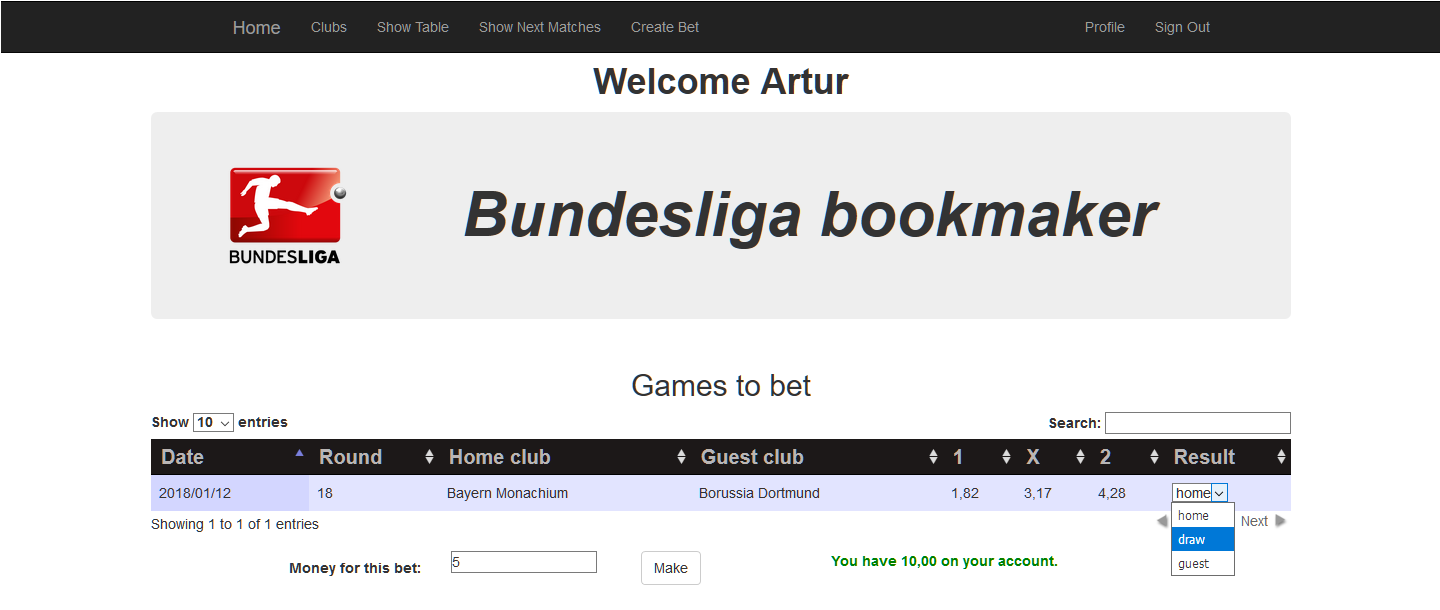
* + 1. Tworzenie kuponów

Klient może tworzyć kupony tylko wtedy, kiedy znane są mecze, które się jeszcze nie odbyły. Aby rozpocząć ten proces, użytkownik musi wybrać z górnego menu przycisk „Create bet”.



Rysunek : Wybieranie meczy do kuponu

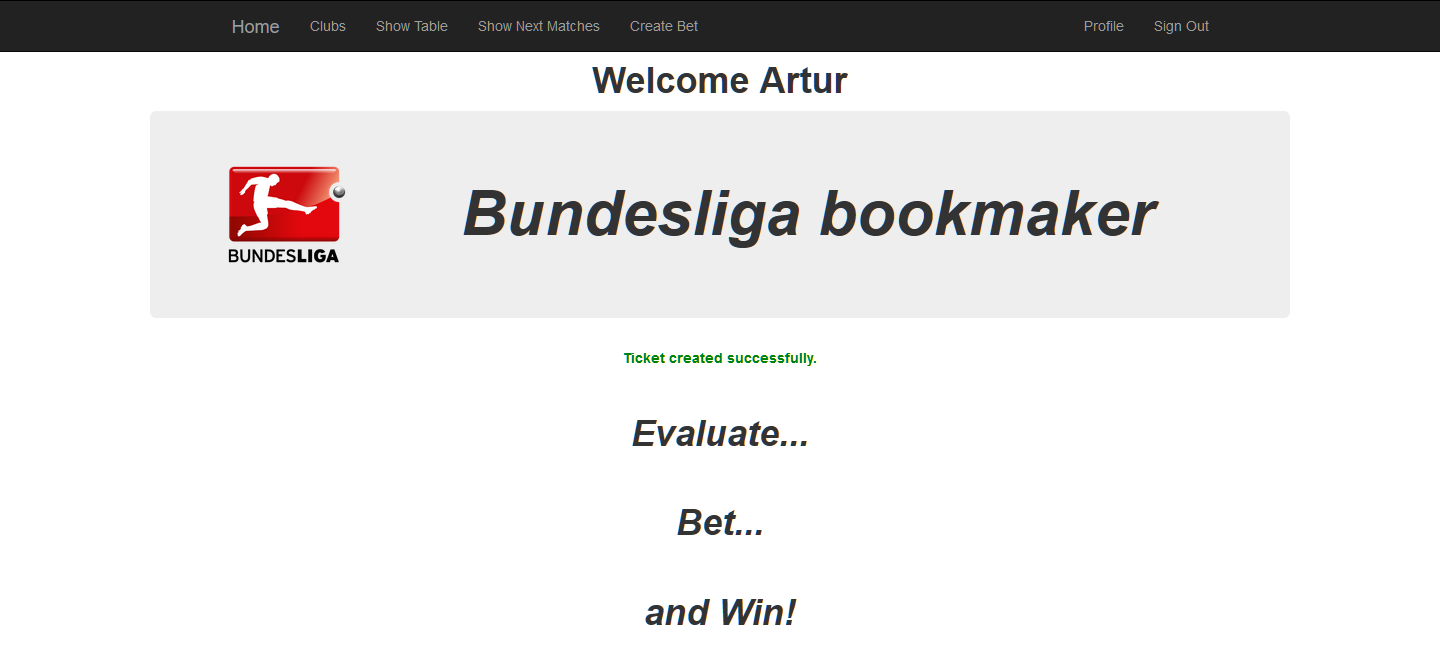
Wówczas aplikacja pokazuje ekran, na którym wypisane są nierozegrane mecze. W swoim kuponie można postawić zakład na spotkania w których zarówno składy nie zostały jeszcze ogłoszone, jak i w tych, w których wyjściowe ustawienie jest już znane. W porównaniu do poprzedniego widoku (Rysunek 12), nie ma tutaj opcji dodania składów drużyn ani rozegrania meczu. Za to ostatnia kolumna z prawej stronie zawiera okienka (tak zwany checkboxy) do zaznaczania wybranych gier. Klient zaznacza te, które chce umieścić w swoim zakładzie, a następnie zatwierdza wybór poprzez naciśnięcie przycisku *„Send types”*. Wtedy aplikacja przekierowywuje użytkownika do następnej strony, w której musi doprecyzować szczegóły swojego kuponu.



Rysunek : Określanie przewidywanych rezultatów

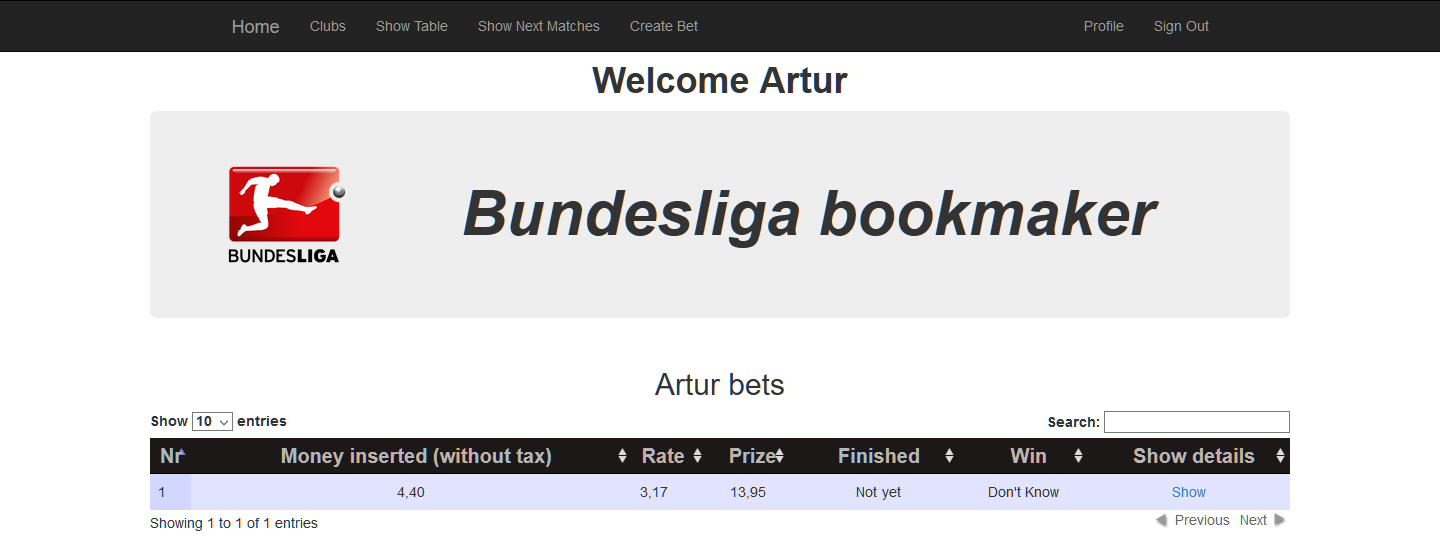
W trakcie przygotowywania tej strony aplikacja sprawdza stan konta gracza, który chce postawić zakład. Jeśli jego stan obecny wynosi zero złotych, to serwis zabrania mu zrealizować kupon (poprzez zablokowanie przycisku zatwierdzającego kupon). W przeciwnym wypadku strona informuje użytkownika o jego możliwościach finansowych i pozwala mu zapisać jego kupon w bazie danych.

Klient widzi wszystkie mecze, na które chciał postawić zakład. Każdy jeden mecz w połączeniu z kursem oraz przewidywanym rezultatem gracza tworzy tak zwany „singiel”, zakład pojedynczy. Użytkownik musi w każdym takim singlu określić jaki rezultat przypuszcza w danym meczu. Aby to zrobić, należy wybrać odpowiednią opcję z rozwijanej listy w kolumnie „Result”. By zapisać wszystkie single w jednym kuponie, wystarczy w polu tekstowym wpisać wartość liczbową mniejszą niż kwota posiadana na koncie i nacisnąć przycisk *„Make”*. Jeśli aplikacja zapisała cały kupon z singlami poprawnie, to przeniesie użytkownika na stronę główną i wyświetli komunikat potwierdzający odpowiednie zapisanie kuponu.



Rysunek : Główna strona z komunikatem

W tym momencie w profilu klienta powinien być widoczny kupon, który przed chwilą został zapisany. Użytkownik znajdzie go w swoim profilu w zakładce *„Show bets”*.



Rysunek : Widok wszystkich kuponów użytkownika

Autor tego projektu starał się go tak zaimplementować, aby jak najbardziej przypominał istniejący serwis bukmacherski. Dlatego tutaj również można się spotkać z podatkami[[9]](#footnote-10). We wszystkich zakładach bukmacherskich w Polsce istnieją dwie niezależne opłaty:

* Podatek od gier w wysokości 12%,
* Podatek od wygranych w wysokości 10%.

Pierwsza z tych opłat pojawia się przy tworzeniu kuponu. Niezależnie od wielkości kwoty przeznaczanej na zakład bukmacherski 12% tej wartości idzie na podatek.

()

Druga z tych opłat nie występuje przy wszystkich zakładach. Tyczy się on tylko tych kuponów, w których możliwa wygrana jest większa od 2280 zł. W przypadku sukcesu takiego kuponu 10% wygranej przeznaczone jest na podatek.

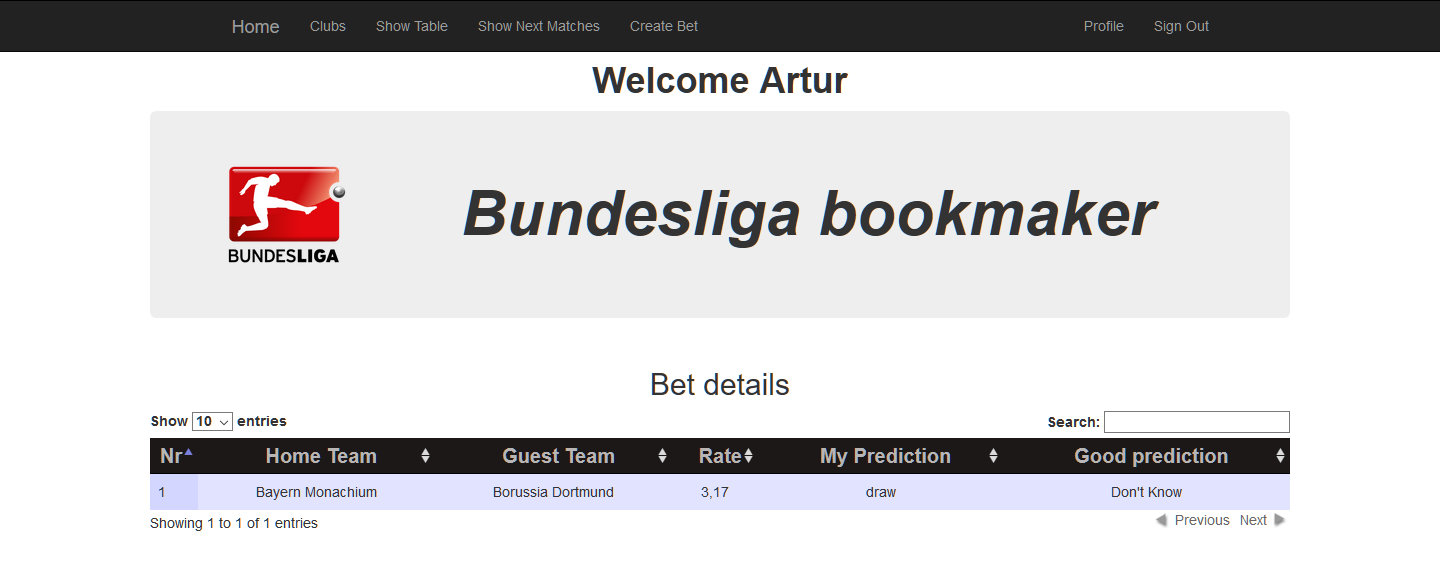
()

Na rysunku nr 19 pokazany przykład. Użytkownik stworzył kupon i poświęcił na niego 5 zł, ale w zapisanych kuponach istnieje kupon z kwotą wpisaną 4.40 zł. Stało się tak dlatego, że 12 procent z kwoty 5 zł jest równe 60 groszy, i taka kwota została pobrana jako podatek.

W tabeli na wspomnianej witrynie można zobaczyć wszystkie zakłady klienta. Są one opisane w następujący sposób:

* Money Inserted (without tax) – kwota przeznaczona na dany kupon,
* Rate – kurs całego kuponu (wylicza się go poprzez pomnożenie kursów wszystkich singli w nim zawartych),
* Prize – możliwa wygrana, jeśli wszystkie single w kuponie zostały dobrze przewidziane,
* Finished – kolumna informująca o aktualnym stanie zakładu:
  + Not yet – nie wszystkie single zostały rozstrzygnięte,
  + Finished – wszystkie mecze z kuponu skończyły się.
* Win – kolumna informująca o końcowym rezultacie zakładu:
  + Don’t Know – kupon nie został jeszcze rozstrzygnięty,
  + Yes / No – użytkownik poprawnie / niepoprawnie przewidział wszystkie single z tego kuponu.

W ostatniej kolumnie znajduje się link, który prowadzi do strony ze wszystkimi szczegółami singli (Rysunek 20).

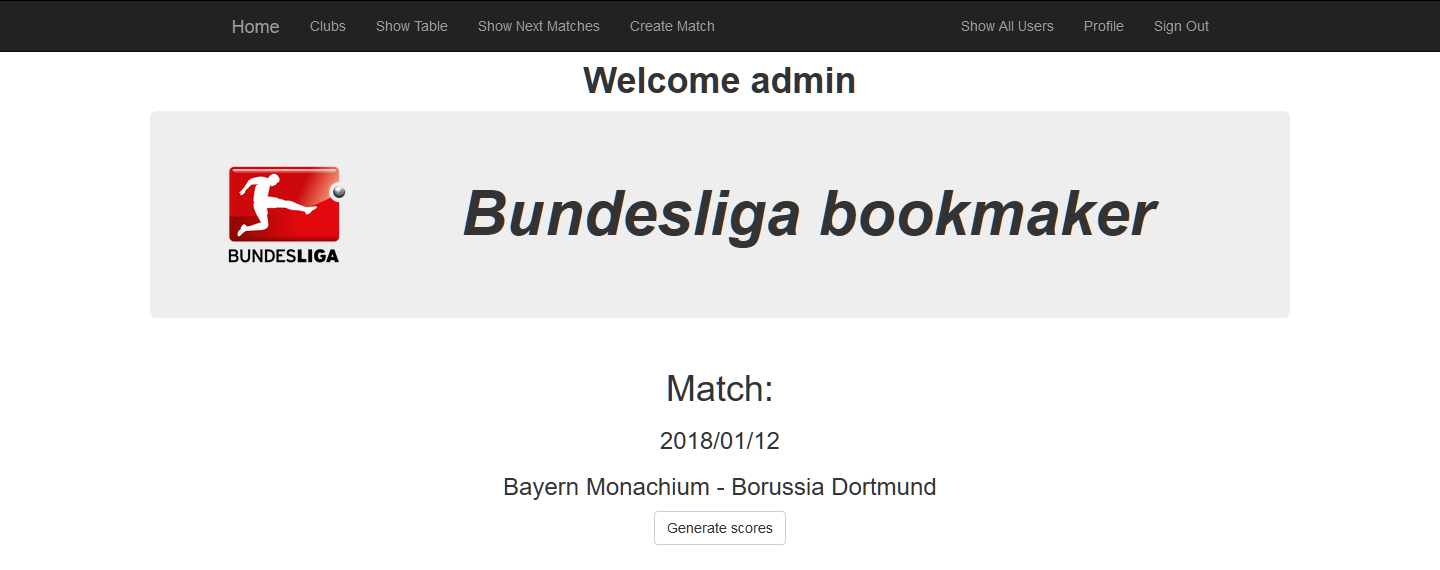


Rysunek : Widok wszystkich singli z kuponu

W kolumnie „Rate” wypisany jest kurs na rezultat zapisany w kolumnie „My Prediction”. W ostatnim pionowym rzędzie znajduje się informacja o trafności singla. „Don’t Know” oznacza, że mecz się jeszcze nie zakończył, a wartość „Yes / No” sygnalizuje poprawność przewidzianego rezultatu.

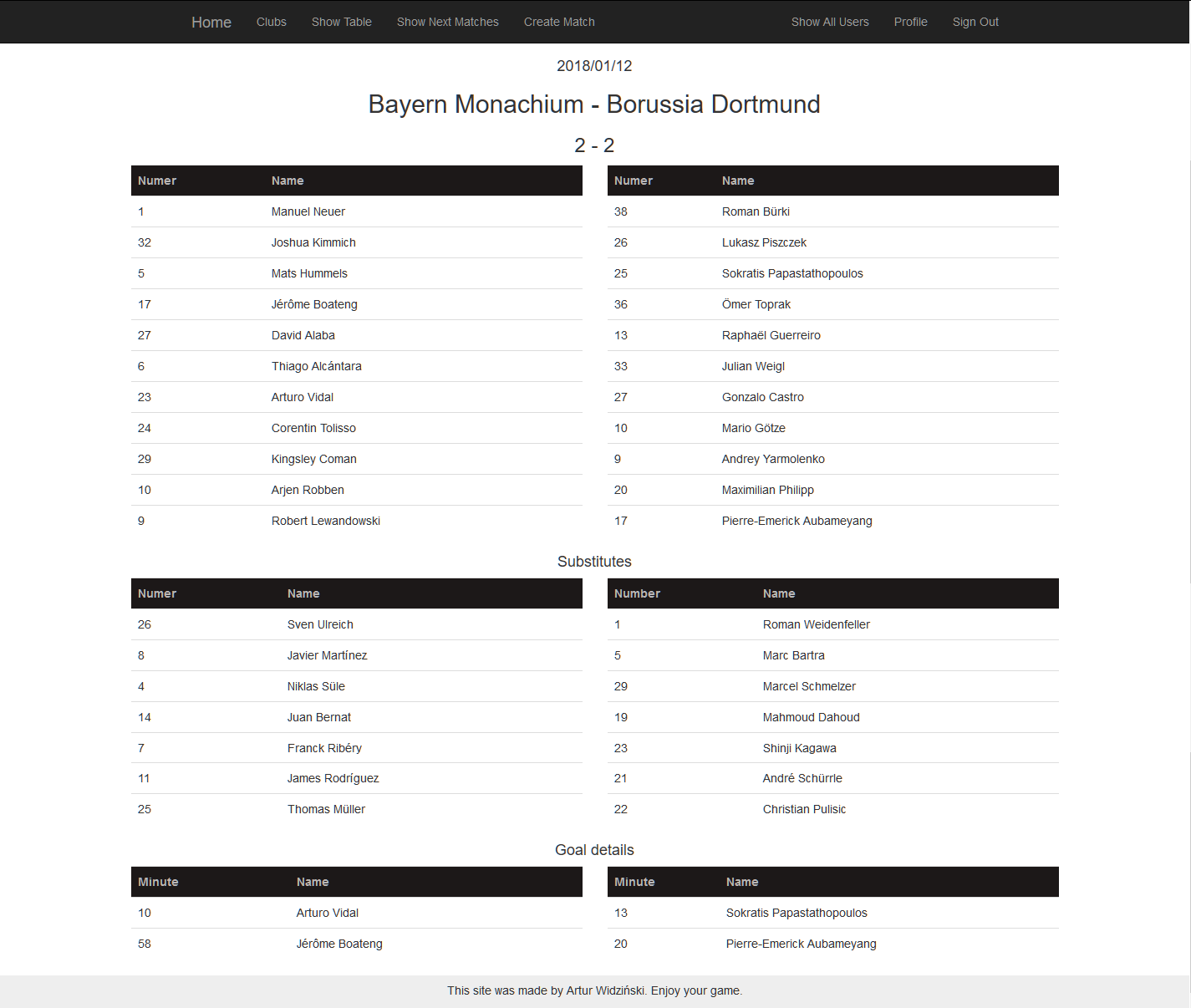
* + 1. Rozgrywanie meczy

Ta operacja może zostać tylko i wyłącznie przez opiekuna strony. Aby zasymulować grę, administrator musi przejść do zakładki „Show next matches”. Z wyświetlonych meczy rozegrane mogą być tylko te, w których są już znane składy wyjściowe i ławki rezerwowe obu drużyn. Tak jak zostało to wcześniej opisane, przycisk *„Play*” rozpocznie proces symulacji spotkania.



Rysunek : Widok przed zasymulowaniem spotkania

Przed wygenerowaniem wyników meczu pojawia się okno, w którym mamy podaną datę oraz drużyny rozgrywające dane spotkanie. Ten widok ma na celu upewnienie administratora, czy mecz, który chciał zagrać jest faktycznie wybraną grą. Po naciśnięciu przycisku *„Generate scores”* aplikacja wykonuje operację związaną z sztuczną inteligencją. Kontroler wysyła żądanie wykonania operacji, która przetwarza mecz na wartości liczbowe (szczegółowy opis w dalszej części pracy), a następnie sieć neuronowa za pomocą parametrów wejściowych wykonuje obliczenia. Otrzymane wyniki są przetwarzane na rezultat meczu i są przedstawiane na stronie, do której aplikacja przekierowywuje użytkownika.



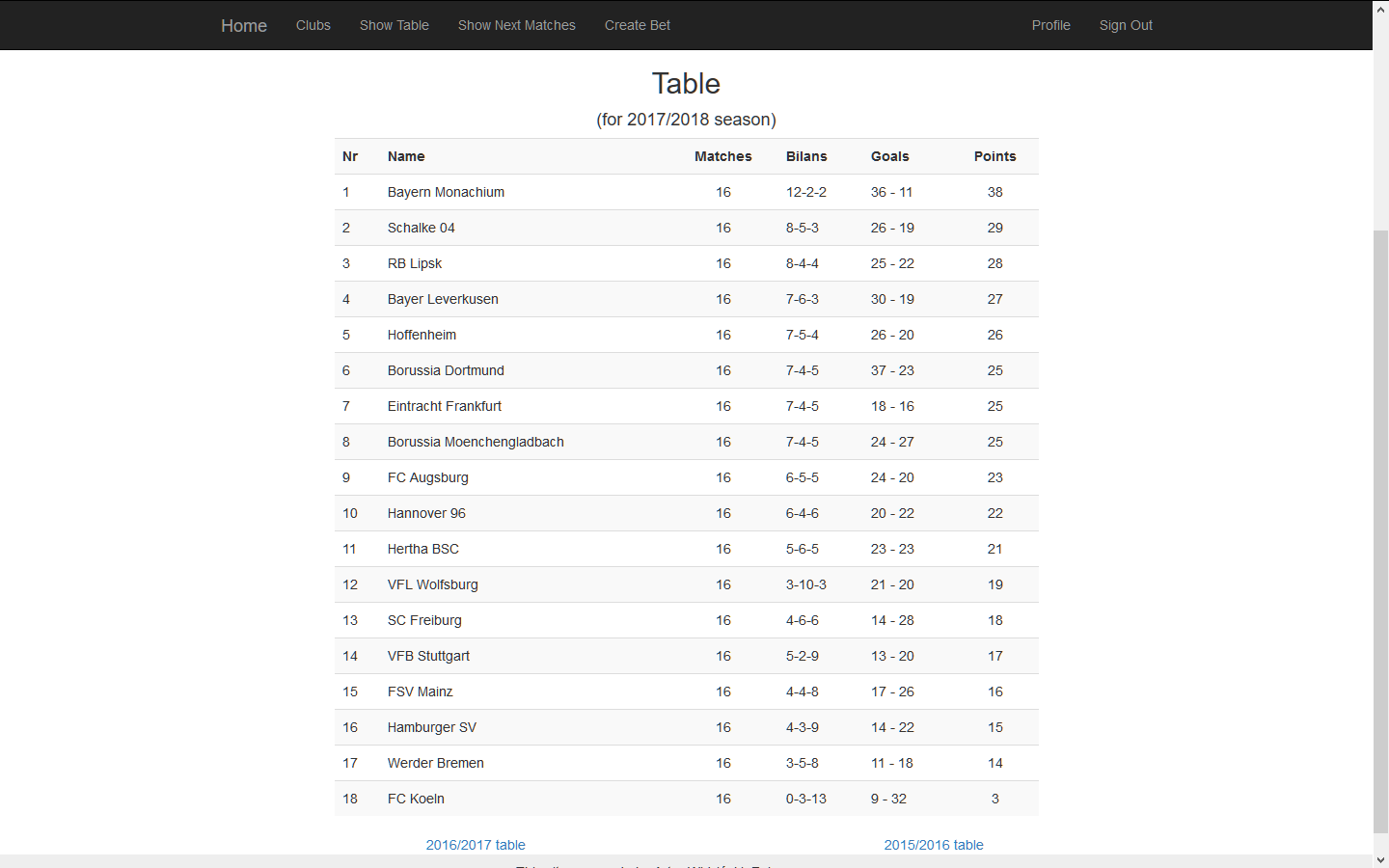
Rysunek : Rezultat meczu wygenerowany przez sieć neuronową

Poza wyznaczeniem, czy mecz zakończył się wygraną gospodarzy, gości lub remisem, aplikacja wylicza ilość oraz strzelców bramek. Wynik tego spotkania zostaje zapisany w bazie aplikacji, a następnie strona aktualizuje dane klubów, które w nim uczestniczyły.

Dodatkowo po rozegraniu spotkania serwis bukmacherski sprawdza wszystkie kupony, w których są single dotyczące tego meczu. Dla każdego singla weryfikowana jest poprawność przewidzianego rezultatu. Jeśli to był ostatni niezweryfikowany zakład pojedynczy w kuponie, aplikacja sprawdza, czy każdy z nich był odgadnięty poprawnie i na tej podstawie wyznacza status całego zakładu. Jeśli jest on wygrany, użytkownik będący właścicielem szczęśliwego losu otrzymuje pieniądze w nim zapisane jako możliwa wygrana. Taką samą kwotę zabiera się z konta administratora. W przeciwnym wypadku gracz nic nie otrzymuje, a opiekun serwisu zgarnia postawioną pulę pieniężną. Po każdym rozegranym meczu wszystkie profile użytkowników są aktualizowane, i każdy z nich może sprawdzić stan swoich kuponów.

* + 1. Statystyki

Poza wspomnianymi już możliwościami serwis bukmacherski udostępnia również strony, na których użytkownik może przeanalizować kluby i wcześniejsze spotkania. Część z tych stron jest dostępna również dla użytkowników niezalogowanych.

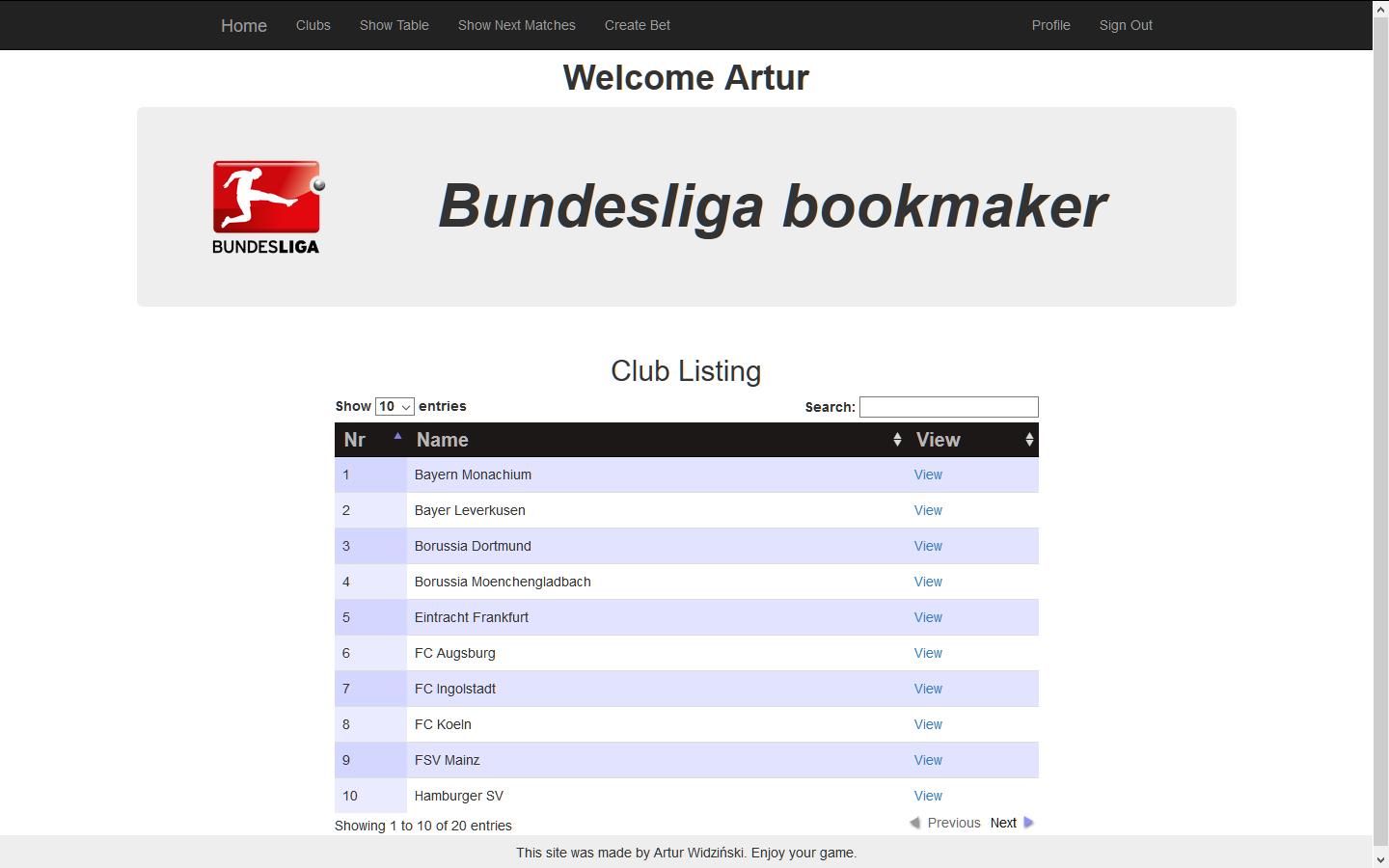


Rysunek : Tabela ligi

Tabela przedstawia zestawienie drużyn, które rywalizują ze sobą w danym sezonie. Tworzona jest ona na podstawie wyników rozegranych meczy. Za każdy wygrany mecz drużyna otrzymuje 3 punkty, za remis 1 punkt, a za porażkę 0. Po zsumowaniu punktów ustawia się je w kolejności malejącej. Pierwszym kryterium sortującym jest ilość zebranych punktów, kolejnym różnica strzelonych bramek od straconych, a następnym ilość zdobytych goli. Jeśli w tym momencie dalej nie jest możliwe ustalenie kolejności drużyn, stosuje się coraz bardziej szczegółowe parametry[[10]](#footnote-11).

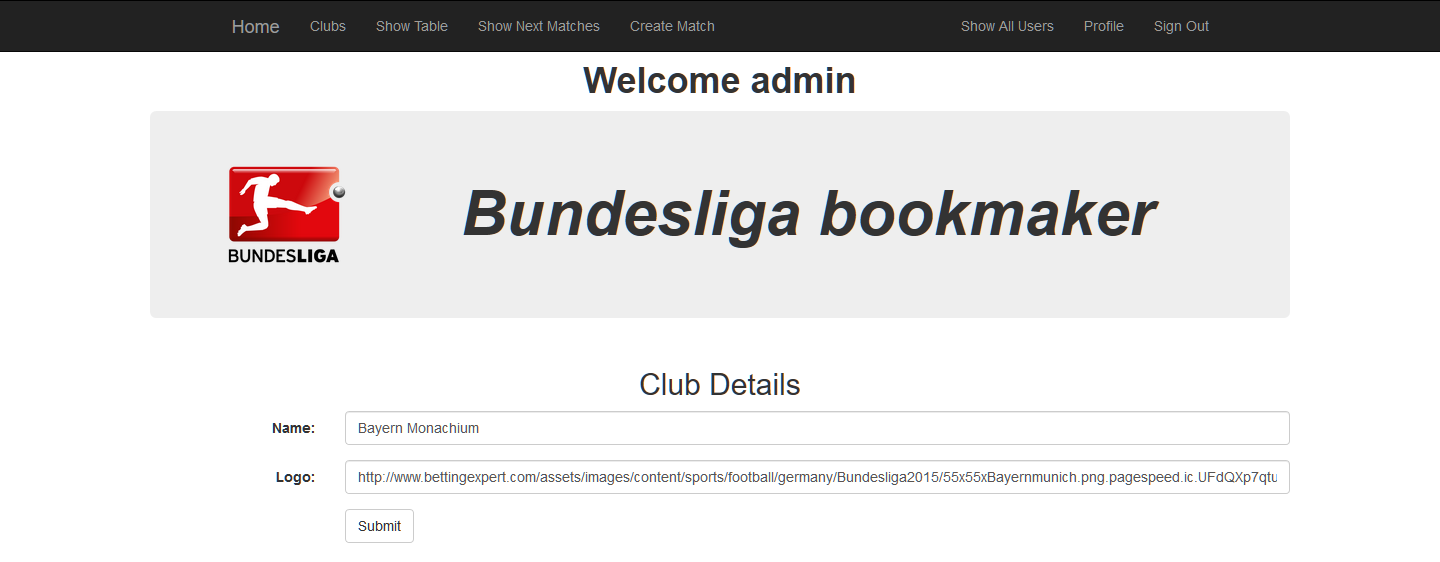
Taka tabela jest dostępna w zakładce „Show table” z górnego menu. Domyślnie pokazywane jest zestawienie z sezonu 2017/2018. Jest ono dostępne dla wszystkich użytkowników, również tych niezalogowanych.

Baza danych tego serwisu opiera się na danych od sierpnia 2015 roku (data rozpoczęcia sezonu 2015/2016). Na dole są dostępne linki, które kierują użytkownika do strony z tabelami ze wszystkich posiadanych w bazie sezonów. Do nich dostęp mają już tylko użytkownicy zalogowani.



Rysunek : Spis wszystkich klubów

Drugim i zarazem ostatnim ekranem, który jest dostępny dla każdego użytkownika jest lista wszystkich klubów, które uczestniczyły w rozgrywkach niemieckiej ligi w piłkę nożną od sezonu 2015/2016. Powyższe okno (Rysunek 24) pokazuje jak tą stronę aplikacja wyświetla dla użytkowników należących do grupy *„users”*. Administrator może w tym miejscu edytować klub lub go usunąć. Dla niezalogowanego użytkownika jedyną dostępną funkcją jest wyświetlenie nazw tych klubów.



Rysunek : Edycja klubu

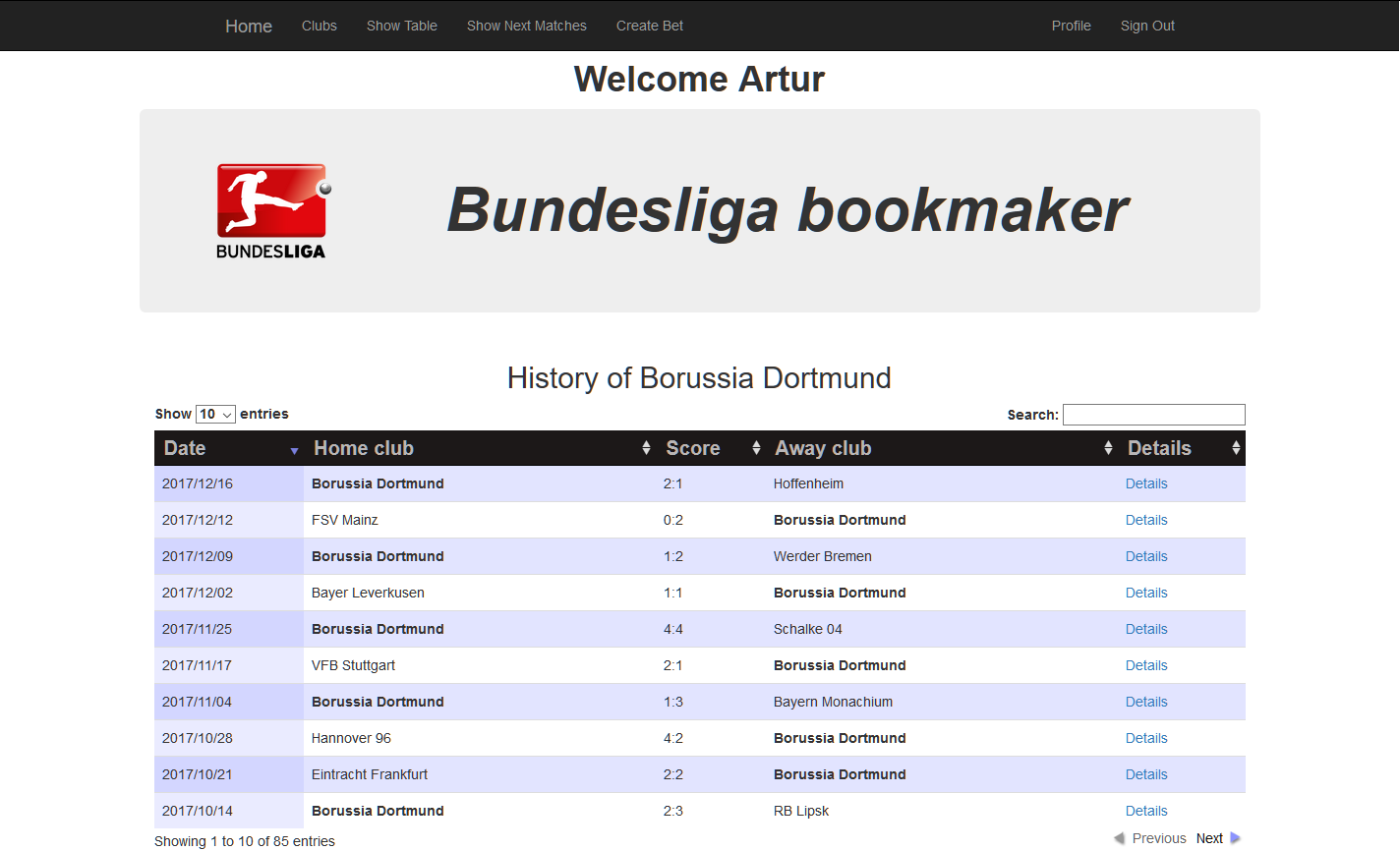
Tak wygląda panel do edycji klubu. W nim administrator może zmienić nazwę klubu, bądź adres http będący linkiem do strony z logiem drużyny. Po zatwierdzeniu zmian aplikacja przekierowywuje opiekuna serwisu do strony z detalami klubu.

Do tej strony można również się dostać w inny sposób. Z okna wyświetlającego listę wszystkich klubów wystarczy wybrać przycisk *„View”*.



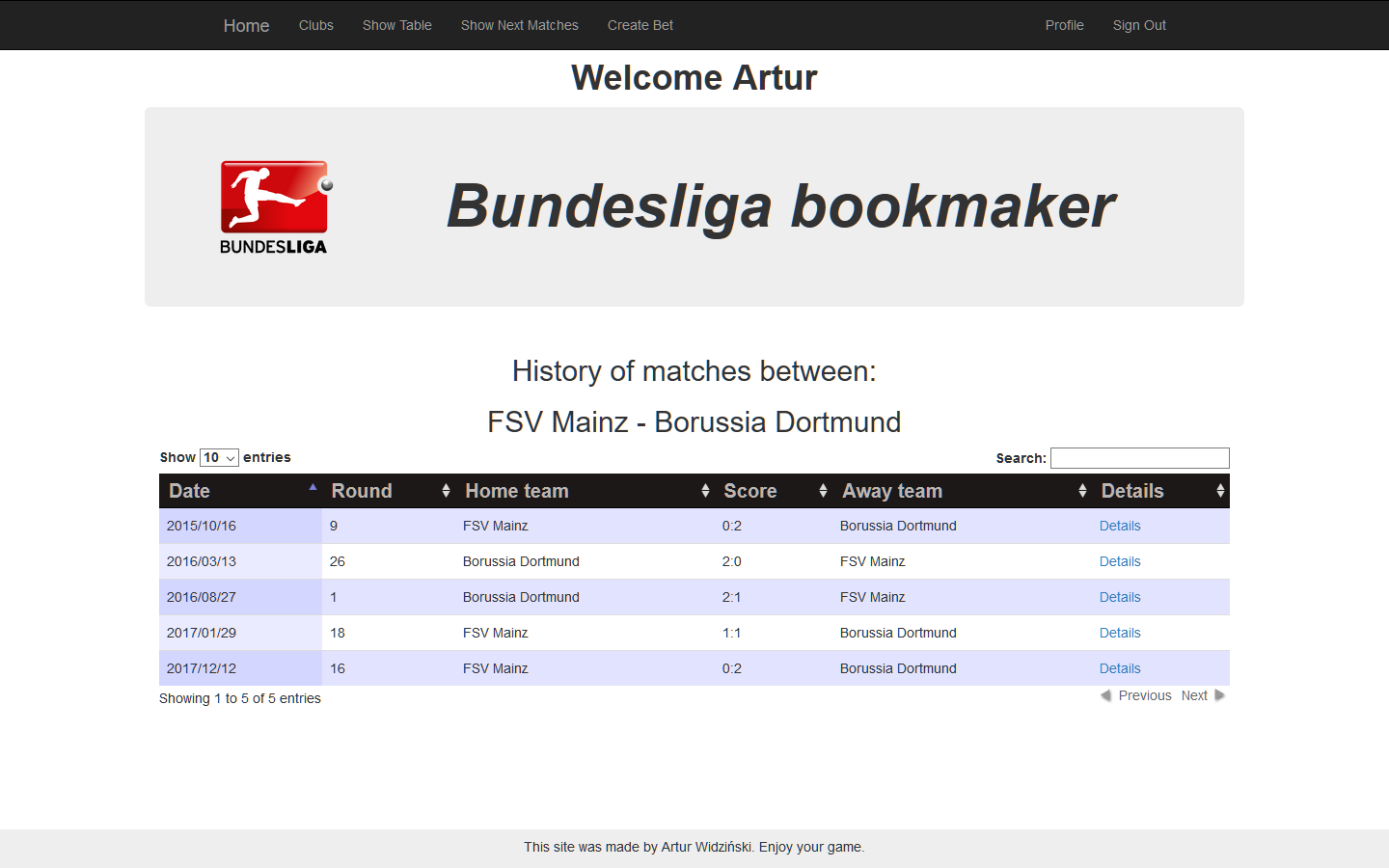
Rysunek : Szczegółowy opis klubu

W tym oknie można zobaczyć jak wygląda logo drużyny, bilans gier (wygrane, remisy i porażki), łączną ilość oraz wartość piłkarzy należących do klubu. Poniżej można zobaczyć listę meczów przyszłych z udziałem tego zespołu, a także wyniki z ostatnich jego spotkań. Pod spodem widnieje lista zawodników, którzy rozgrywają swoje mecze w barwach tej drużyny. Na samym dole strony widnieje link *„Show History”*, który pokazuje użytkownikowi historię wszystkich spotkań klubu, które są zapisane w bazie danych.



Rysunek : Historia klubu

Na poprzedniej stronie (Rysunek 26) można znaleźć jeszcze jedną usługę. W liście ostatnich meczów klubu po prawej stronie znajduje się napis *„Previous matches”*. Jest to link, który prowadzi klienta serwisu do strony, na której może zobaczyć wszystkie starcia pomiędzy wybranymi drużynami.



Rysunek : Historia starć bezpośrednich

* 1. Algorytm obliczania kursów bukmacherskich

W tym podrozdziale zostanie opisany sposób, w jaki opracowano algorytm liczący kursy na mecz bukmacherski.

Pierwszym pytaniem, na które trzeba odpowiedzieć brzmi „Co to jest kurs bukmacherski w znaczeniu matematycznym?”. Otóż jest to odwrotność prawdopodobieństwa wystąpienia danego zdarzenia.

()

Zatem aby obliczyć kurs bukmacherski na dane zdarzenie, wystarczy wyznaczyć prawdopodobieństwo jego wystąpienia. Ale nad tym zadaniem też trzeba się zastanowić, jak to policzyć. Tutaj z pomocą przychodzi rozkład Poissona[[11]](#footnote-12).

Rozkład Poissona jest rozkładem skokowym, który szacuje prawdopodobieństwo wystąpienia następujących po sobie zdarzeń, które mają określoną częstotliwość. Wyraża się on następującym wzorem:

()

gdzie: **–** częstotliwość występowania zdarzenia.

Jak można wykorzystać to pojęcie matematyczne, aby wyliczyć kurs bukmacherski? Po wyznaczeniu średniej arytmetycznej strzelonych bramek na mecz danych drużyn to zagadnienie nie będzie już problemem.

Załóżmy, że drużyna A ma średnią 1,4 strzelonych goli na jeden mecz. Wtedy

()

Jeśli chcemy wyliczyć prawdopodobieństwa strzelenia *k* bramek w meczu przez drużynę A, wystarczy podstawić dane pod wyżej wymieniony wzór.

()

W powyższym równaniu udało się policzyć prawdopodobieństwo zdarzenia, w którym drużyna A nie strzeli ani jednego gola w meczu. Wynosi ono około 24.66%. Poniżej zostało policzonych jeszcze kilka prawdopodobieństw:

()

()

()

()

Wyniki przedstawionego przykładu znajdują się w poniższej tabeli:

Tabela : Prawdopodobieństwa wyliczane rozkładem Poisson

|  |  |
| --- | --- |
| Ilość goli | Prawdopodobieństwo |
| 0 | **24.66 %** |
| 1 | **34.52 %** |
| 2 | **24.17 %** |
| 3 | **11.28 %** |
| 4 | **3.95 %** |

W przedstawiony właśnie sposób można wyznaczyć prawdopodobieństwo strzelenia konkretnej liczby goli. W bardzo prosty sposób można to przełożyć na wyliczenie szansy na konkretny rezultat w meczu.

Załóżmy, że po naszych obliczeniach drużyna A ma szansę 25% na strzelenie 1 bramki w meczu, a drużyna B ma prawdopodobieństwo równe 12.5% na skończenie meczu bez zdobytej bramki. Stąd prawdopodobieństwo na końcowy wynik meczu równy 1-0 meczu pomiędzy drużynami A i B wynosi 3.125%.

()

Jednak takie rozwiązanie nie jest odpowiedzią idealną. Nie można wszystkiego uzależnić od jednego parametru.

W tym projekcie algorytm wyliczania kursów wygląda następująco:

* W trakcie procesu uruchamiania aplikacji program zlicza ilość bramek zdobytych oraz straconych w meczach przez drużyny będące gospodarzami spotkań, a także ilość wszystkich gier, które znajdują się w bazie danych.
* Kiedy administrator wybierze drużyny, które rozegrają ze sobą mecz, aplikacja sprawdza wyniki osiągane przez te zespoły z ostatnich kilkunastu/kilkudziesięciu spotkań i zapisuję ilość zdobytych i straconych bramek.
* Następnie program sprawdza, czy przeciętna ilość bramek strzelonych drużyny grającej jako gospodarz spotkania jest porównywalna do tej, którą uzyskuje ta sama drużyna grając jako gość meczu.
* Jeśli te dwie wartości są podobne, to jako średnią ataku tej drużyny program uznaje średnią strzelonych goli we wszystkich przeanalizowanych jej grach.
* Jeśli jednak wspomniane wcześniej przeciętne różnią się w znacznym stopniu, to jako średnia ataku tej drużyny wybiera tą średnią, która dotyczy tego spotkania (jeśli gra jako gość, to średnią strzelonych goli na wyjeździe, i odwrotnie).
* W taki sam sposób wyliczane są takie parametry jak:
  + Średnia ataku drużyny przyjezdnej,
  + Średnia obrony drużyny gospodarzy,
  + Średnia obrony drużyny przyjezdnej.
* Dodatkowo aplikacja dla każdego meczu liczy średnią strzelonych oraz straconych bramek drużyn grających na własnym stadionie oraz drużyny gości. Z tej operacji powstaną 4 dodatkowe zmienne:
  + Średnia ataku gospodarzy w lidze,
  + Średnia ataku gości w lidze,
  + Średnia obrony gospodarzy w lidze,
  + Średnia obrony gości w lidze.
* Następnie serwis określa jak współczynniki drużyn, które rozgrywają dany mecz wyglądają w porównaniu współczynników całej ligi. Tworzą się wtedy nowe wartości:

()

()

()

()

* Z wyliczonych właśnie liczb algorytm tworzy dwa końcowe współczynniki:

()

(17)

* Na podstawie tych dwóch wartości liczbowych aplikacja za pomocą rozkładu Poissona (Równanie 4) wylicza prawdopodobieństwo ilości strzelonych bramek przez obie drużyny, i w pokazany sposób wylicza prawdopodobieństwo na wszystkie możliwe rezultaty.
* Aby określić prawdopodobieństwo na wygraną jednej z drużyn, należy zsumować prawdopodobieństwa wszystkich tych zdarzeń, w których ta drużyna strzeliła więcej goli niż przeciwna.

Po wykonaniu wszystkich wymienionych punktów algorytm zwraca 3 liczby, które określają szansę na każdy z rezultatów na podstawie ilości strzelonych i straconych bramek. Jednakże tutaj nie jest koniec algorytmu. Jest jeszcze jeden wskaźnik, który koryguje kursy meczowe.

Istnieją w świecie piłkarskim przypadki, w których drużyna A nigdy nie wygrała z drużyną B. Im dłużej taka seria trwa, tym większą presję wywiera to na zawodników. W algorytmie liczącym kurs na dane spotkanie postanowiono również uwzględnić ten aspekt.

Po wyznaczeniu poprzednich wartości aplikacja wyciąga z bazy wszystkie mecze, w których spotkały się te same drużyny, dla których wyliczane są kursy. Następnie:

* Zliczana jest ilość pobranych spotkań, a także ilość wygranych, przegranych oraz remisów.
* Dla każdego z tych rezultatów wylicza szansę według wzoru:

()

W ten sposób algorytm uzyskał kolejne 3 liczby, określające szanse na dany rezultat meczu.

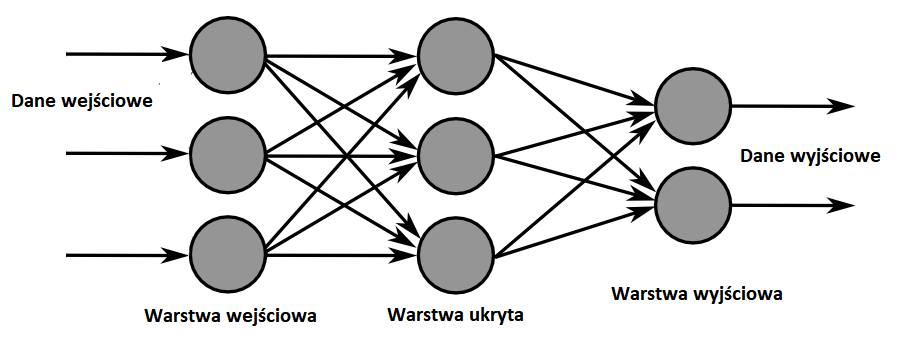
Po tym etapie algorytm łączy odpowiadające sobie prawdopodobieństwa, używając średniej ważonej. Prawdopodobieństwo wyliczone na podstawie strzelonych bramek wydaje się mieć większy wpływ na wynik meczu, dlatego waga tych szans jest zdecydowanie większa.

* 1. Algorytm symulujący rozgrywanie meczy

W tym podrozdziale zostanie opisany algorytm, który odpowiada za symulację rozgrywanej gry. Ten mechanizm jest związany z siecią neuronową.

W niniejszym projekcie postanowiono stworzyć wielowarstwową sieć neuronową z algorytmem wstecznej propagacji (ang. *backpropagation*). W takim modelu na początku trzeba zdefiniować ilość warstw ukrytych, a następnie liczbę neuronów w każdej z warstw (wejściowej, wyjściowej oraz ukrytych). Algorytm ten pozwala na zmodyfikowanie wag w neuronach z poprzednich warstw.

Podczas uruchamiania serwisu aplikacja tworzy z każdego meczu tzw. wektor uczący. Składa się on z wektora wejściowego oraz wyjściowego. Dane z wektora wejściowego są przesyłane do neuronów warstwy wejściowej, a dane z wektora wyjściowego spełniają rolę tzw. nauczyciela. Rola ta jest bardzo istotna w procesie nauczania. Gdy sieć otrzymuje sygnały wejściowe, to na ich podstawie wyznacza sygnały wyjściowe. Wtedy nauczyciel „czuwa” i sprawdza, czy otrzymane dane zgadzają się z tym, co zostało podane przez niego jako dane oczekiwane. Jeśli sieć wyznaczyła inny wynik niż spodziewany, wyliczany jest błąd odpowiedzi, i następnie jest propagowany do poprzednich warstw. W nich neurony modyfikują swoje wagi, do momentu aż błąd w warstwie wyjściowej będzie poniżej ustalonej wartości.



Rysunek : Model sieci wielowarstwowej[[12]](#footnote-13)

W tym projekcie warstwa wejściowa posiada 14 neuronów, wyjściowa 3, a jedyna ukryta 25. Poniżej są zapisane wyjaśnienia dokonanych wyborów.

Warstwa wyjściowa posiada 3 neurony. Każdy z nich przekazuje nam informację o tym jaka jest procentowa szansa na końcowy rezultat meczu (wygrana, remis bądź porażka).

Nie ma dobrej recepty mówiącej o tym, ile neuronów powinno się znajdować w warstwie ukrytej. Teoretycznie im więcej, tym sieć szybciej się uczy, jednakże nie można wybrać ich za dużo, co może skutkować nieustannym uczeniem sieci.

W warstwie wejściowej istnieje 14 neuronów, ponieważ tyle parametrów zostaje wyliczonych z każdego meczu. Aby sieć była w stanie się czegokolwiek nauczyć, musi otrzymać dane znormalizowane, tj. zapisane w przedziale <0; 1> bądź <-1; 1>. Pierwsze osiem neuronów mówią o tym, czy piłkarze wybrani do podstawowych składów są najlepszymi dostępnymi w każdym z klubów. Wyliczane jest to w następujący sposób:

* Z każdego meczu wyciągane są składy wyjściowe wybrane do spotkania, a następnie sumowana jest ich łączna wartość rynkowa, osobno dla każdej z formacji:
  + Bramkarz,
  + Obrońcy,
  + Pomocnicy,
  + Napastnicy.
* Następnie dla każdego z klubów pobierani są piłkarze z bazy danych Wybiera się wtedy tylu najlepszych z nich (czyli zawodników o największej wartości rynkowej) w każdej formacji, ilu zostało wybranych do podstawowego składu. Np. jeśli w meczu drużyna wystawia 4 obrońców, to z bazy jest pobranych 4 najdroższych obrońców w klubie.
* Z każdej formacji powstaje jeden współczynnik. Wylicza się go poprzez podzielenie łącznej wartości piłkarzy z formacji na boisku przez łączną wartość najlepszych piłkarzy w klubie z tej samej formacji.
* W trakcie liczenia wartości piłkarzy sprawdzana jest jego forma. Jeśli zawodnik posiada średnią ilość goli na mecz większą niż bonusowy współczynnik, to jego wartość rynkowa jest zwiększana. Bonusowy współczynnik różni się w zależności od formacji, do której gracz należy (Tabela 3).

Tabela : Bonusowy współczynnik za formę zawodnika

|  |  |
| --- | --- |
| Formacja | Bonusowy współczynnik |
| Bramkarz | **0.5** |
| Obrońca | **0.2** |
| Pomocnik | **0.3** |
| Napastnik | **0.5** |

Dla piłkarzy grających na pozycji bramkarza średnia ilość „czystych kont” na mecz musi być większa niż współczynnik, aby wartość rynkowa wzrosła.

W taki sposób neurony w kolejności o numerach 1-4 wskazują jakość czterech formacji z drużyny grającej jako gospodarz, a z numerami 5-8 z drużyny przyjezdnych.

Kolejne 2 neurony posiadają współczynniki związane ze składami wyjściowymi. Aplikacja oblicza łączną wartość piłkarzy z podstawowego składu obu drużyn (z uwzględnieniem bonusów), a następnie określa współczynniki z wykorzystaniem poniższego wzoru:

()

Na tej samej zasadzie powstają wartości neuronów u numerach 11-12, ale zamiast piłkarzy wyjściowego składu ocenia się zawodników siedzących na ławce rezerwowych.

Ostatnie 2 neurony zawierają informacje o punktach zdobytych w sezonie. Wyliczany jest tam iloraz pomiędzy zdobytą ilością punktów a maksymalnie dostępną.

Jeśli drużyna A po 5 meczach zdobyła 10 punktów, to jej współczynnik jest równy 0.66, ponieważ:

()

W tabeli 4 można zobaczyć jakie parametry zostały przekazane do wektora wyjściowego w zależności od rezultatu rozpatrywanego meczu.

Tabela : Wartości neuronów w warstwie oczekiwanej w zależności od rezultatu meczu

|  |  |  |  |
| --- | --- | --- | --- |
| Wynik | Neuron 1 | Neuron 2 | Neuron 3 |
| Wygrana drużyny A | 1.0 | 0.0 | 0.0 |
| Remis | 0.0 | 1.0 | 0.0 |
| Wygrana drużyny B | 0.0 | 0.0 | 1.0 |

W opisany powyżej sposób sieć uczy się przewidywać wynik meczu. Kiedy administrator wywołuje symulację meczu (Rysunek 21), aplikacja przekazuje do sieci mecz, który został przetworzony na wektor wejściowy w taki sam sposób, jak podczas procesu uczenia sieci. Następnie sieć przewiduje rezultat meczu korzystając z wzoru, który sama wyznaczyła. Po tym etapie sieć nie ingeruje już w wynik końcowy meczu.

Aplikacja znając już końcowe rozstrzygnięcie meczu określa szanse na wszystkie scenariusze spełniające przewidziany rezultat (za pomocą rozkładu Poissona, jak w poprzednim podpunkcie) i losuje jeden z nich. Następnie każdego strzelca bramki ustala osobno. Najpierw losuje formacje, z której gracz strzelił gola. Szansa na wybór strzelca z linii obrony jest najmniejszy, a z formacji napastników największy. Ostatnim wyborem jest wybór gracza z formacji, który strzelił gola. Tutaj również został on wylosowany. Największą szansę na wylosowanie ma gracz z największą wartością rynkową (z uwzględnieniem bonusu).

1. Doświadczenie

Aby odpowiedzieć na pytania postawione w rozdziale pierwszym, wykonano doświadczenie wykorzystujące powstały produkt. Znaleziono kilka chętnych osób, dla których obstawianie zakładów bukmacherskich nie jest nowym doświadczeniem. Poproszono je o wypełnienie kuponów opartych na kursach wyliczonych przez tą aplikację. Zakres obstawianych meczy sięgał od 12 do 17 kolejki rozgrywek Bundesligi w sezonie 2017/2018.

Po wpisaniu wyznaczonych przez wspomniane osoby (również określani jako „testerzy”) zakładów w aplikacji można było przystąpić do sprawdzania dokładności w przewidywaniu meczy.

Tabela : Kolejka 12 rozgrywek Bundesligi sezonu 2017/2018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mecz | Typ aplikacji | Typ rzeczywisty | Wynik aplikacji | Wynik rzeczywisty |
| Stuttgart – Dortmund | **Wygrana gości** | **Wygrana gospodarzy** | **0-1** | **2-1** |
| Wolfsburg – Freiburg | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **2-1** | **3-1** |
| Mainz –  Koeln | **Remis** | **Wygrana gospodarzy** | **1-1** | **1-0** |
| Leverkusen – Lipsk | **Wygrana gości** | **Remis** | **1-2** | **2-2** |
| Hoffenheim – Frankfurt | **Remis** | **Remis** | **0-0** | **1-1** |
| Bayern – Augsburg | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **3-1** | **3-0** |
| Hertha - Monchengladbach | **Wygrana gospodarzy** | **Wygrana gości** | **2-1** | **2-4** |
| Schalke – Hamburger | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **2-0** | **2-0** |
| Bremen - Hannover | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **1-0** | **4-0** |

Tabela : Kolejka 13 rozgrywek Bundesligi sezonu 2017/2018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mecz | Typ aplikacji | Typ rzeczywisty | Wynik aplikacji | Wynik rzeczywisty |
| Hannover - Stuttgart | **Wygrana gospodarzy** | **Remis** | **1-0** | **1-1** |
| Lipsk –  Bremen | **Remis** | **Wygrana gospodarzy** | **1-1** | **2-0** |
| Freiburg –  Mainz | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **2-1** | **2-1** |
| Frankfurt – Leverkusen | **Remis** | **Wygrana gości** | **2-2** | **0-1** |
| Dortmund – Schalke | **Remis** | **Remis** | **0-0** | **4-4** |
| Augsburg – Wolfsburg | **Wygrana gości** | **Wygrana gospodarzy** | **1-2** | **2-1** |
| Monchengladbach – Bayern | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **3-1** | **2-1** |
| Hamburger – Hoffenheim | **Remis** | **Wygrana gospodarzy** | **1-1** | **3-0** |
| Koeln –  Hertha | **Wygrana gości** | **Wygrana gości** | **2-3** | **0-2** |

Tabela : Kolejka 14 rozgrywek Bundesligi sezonu 2017/2018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mecz | Typ aplikacji | Typ rzeczywisty | Wynik aplikacji | Wynik rzeczywisty |
| Freiburg – Hamburger | **Wygrana gospodarzy** | **Remis** | **1-0** | **0-0** |
| Mainz –  Augsburg | **Wygrana gospodarzy** | **Wygrana gości** | **2-0** | **1-3** |
| Leverkusen – Dortmund | **Remis** | **Remis** | **1-1** | **1-1** |
| Hoffenheim – Lipsk | **Wygrana gości** | **Wygrana gospodarzy** | **0-2** | **4-0** |
| Bremen – Stuttgart | **Remis** | **Wygrana gospodarzy** | **1-1** | **1-0** |
| Bayern – Hannover | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **3-2** | **3-1** |
| Schalke –  Koeln | **Wygrana gości** | **Remis** | **0-1** | **2-2** |
| Hertha - Frankfurt | **Wygrana gospodarzy** | **Wygrana gości** | **2-0** | **1-2** |
| Wolfsburg - Monchengladbach | **Remis** | **Wygrana gospodarzy** | **2-2** | **3-0** |

Tabela : Kolejka 15 rozgrywek Bundesligi sezonu 2017/2018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mecz | Typ aplikacji | Typ rzeczywisty | Wynik aplikacji | Wynik rzeczywisty |
| Stuttgart – Leverkusen | **Wygrana gości** | **Wygrana gości** | **0-1** | **0-2** |
| Lipsk –  Mainz | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **1-0** | **2-2** |
| Hamburger – Wolfsburg | **Remis** | **Remis** | **2-2** | **0-0** |
| Frankfurt – Bayern | **Remis** | **Wygrana gości** | **1-1** | **0-1** |
| Dortmund – Bremen | **Wygrana gospodarzy** | **Wygrana gości** | **2-1** | **1-2** |
| Monchengladbach – Schalke | **Remis** | **Remis** | **3-3** | **1-1** |
| Koeln –  Freiburg | **Wygrana gospodarzy** | **Wygrana gości** | **2-1** | **3-4** |
| Hannover – Hoffenheim | **Wygrana gości** | **Wygrana gospodarzy** | **1-4** | **2-0** |
| Augsburg - Hertha | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **1-0** | **1-1** |

Tabela : Kolejka 16 rozgrywek Bundesligi sezonu 2017/2018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mecz | Typ aplikacji | Typ rzeczywisty | Wynik aplikacji | Wynik rzeczywisty |
| Wolfsburg –  Lipsk | **Wygrana gospodarzy** | **Remis** | **3-2** | **1-1** |
| Mainz –  Dortmund | **Wygrana gospodarzy** | **Wygrana gości** | **2-1** | **0-2** |
| Hamburger – Frankfurt | **Remis** | **Wygrana gości** | **2-2** | **1-2** |
| Freiburg – Monchengladbach | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **1-0** | **1-0** |
| Hoffenheim – Stuttgart | **Wygrana gości** | **Wygrana gospodarzy** | **1-2** | **1-0** |
| Schalke – Augsburg | **Remis** | **Wygrana gospodarzy** | **1-1** | **3-2** |
| Leverkusen – Bremen | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **1-0** | **1-0** |
| Hertha – Hannover | **Wygrana gości** | **Wygrana gospodarzy** | **0-2** | **3-1** |
| Bayern –  Koeln | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **3-0** | **1-0** |

Tabela : Kolejka 17 rozgrywek Bundesligi sezonu 2017/2018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mecz | Typ aplikacji | Typ rzeczywisty | Wynik aplikacji | Wynik rzeczywisty |
| Monchengladbach – Hamburger | **Wygrana gospodarzy** | **Wygrana gospodarzy** | **1-0** | **3-1** |
| Stuttgart –  Bayern | **Wygrana gospodarzy** | **Wygrana gości** | **2-0** | **0-1** |
| Frankfurt – Schalke | **Wygrana gości** | **Remis** | **1-3** | **2-2** |
| Koeln –  Wolfsburg | **Wygrana gości** | **Wygrana gospodarzy** | **0-2** | **1-0** |
| Bremen –  Mainz | **Wygrana gospodarzy** | **Remis** | **2-0** | **2-2** |
| Augsburg – Freiburg | **Remis** | **Remis** | **0-0** | **3-3** |
| Dortmund – Hoffenheim | **Wygrana gości** | **Wygrana gospodarzy** | **0-2** | **2-1** |
| Hannover – Leverkusen | **Remis** | **Remis** | **1-1** | **4-4** |
| Lipsk –  Hertha | **Wygrana gospodarzy** | **Wygrana gości** | **2-1** | **2-3** |

W tabelach 5-10 znajdują się wyniki zasymulowane przez niniejszy serwis bukmacherski. Skuteczność w przewidywaniu rezultatów gier wyniosła około 39% (21 meczów z 54). Efektywność odgadywania dokładnego wyniku bramkowego była na niższym poziomie (4 poprawne spotkania).

Próbując przewidzieć rezultat meczu trzeba wybrać jedną z trzech opcji, co daje nam prawdopodobieństwo prawidłowego odgadnięcia rezultatu 33,(3)%. Ta aplikacja przewiduje wyniki z większą efektywnością. Na tej podstawie można określić skuteczność sieci neuronowej jako dobrą. Można spróbować ją poprawić, np. poprzez zwiększenie ilości danych uczących lub zmianie sposobu liczenia poszczególnych parametrów.

Po rozegraniu wszystkich meczy została również sprawdzona skuteczność testujących osób. Każdy z nich zrobił po jednym kuponie na każdą kolejkę. Ich zadaniem było przewidzenie rezultatu, który wyznaczy ten serwis bukmacherski. Na 30 stworzonych zakładów tylko 5 okazało się zgodnie przepowiedzianych z tą aplikacją. Może to oznaczać, iż albo algorytm liczący kursy na dane spotkanie zmylił testerów, albo testerzy nie mieli tym razem szczęścia.

1. Podsumowanie i wnioski

Celem tej pracy inżynierskiej było zaimplementowanie takiego serwisu bukmacherskiego, która za pomocą sztucznej inteligencji możliwie jak najdokładniej przewidywała wynik danego spotkania piłkarskiego. Ta aplikacja posiada zaimplementowaną sieć neuronową, która jest w stanie przewidzieć wynik części meczów. Nie da się mieć stuprocentowej pewności co do zwycięstwa jednego z zespołów, ponieważ mecze piłki nożnej są nieprzewidywalne, nie ma jednego wzoru na wygrywanie.

Bez żadnych pomocnych urządzeń, danych, człowiek ma 33 % szansy na prawidłowe przepowiedzenie końcowego rezultatu. Jednak bazując na podstawowych danych, które zostały odpowiednio przekazane do sieci, udało się uzyskać większą skuteczność. Dzięki temu można postawić tezę, iż w przypadku dostępu do większej puli danych statystycznych na temat rozgrywek istnieje możliwość zwiększenia efektywności tego programu.

Serwis bukmacherski został przetestowany przez grupę ochotników. Ich skuteczność zmusza do zastanowienia się gdzie leży tego przyczyna: czy osoby testujące nie umiały poprawnie przeanalizować wszystkich danych dotyczących drużyn rozgrywających przyszłe spotkanie, czy wyliczony kurs nie był wyliczony dokładnie.

W kolejnych etapach rozwoju tej aplikacji autor skoncentrowałby się głównie na dopracowaniu działań w omawianych algorytmach (rozdział 2.5 i 2.6). Za pomocą bardziej złożonych wzorów matematycznych można spróbować wyznaczyć nowe, lepsze wartości, które następnie można przekazać neuronom w warstwie wejściowej. Dodatkowym zagadnieniem, które zostanie dopracowane jest korzystanie z jednej bazy, która udostępnia więcej parametrów na temat meczy, piłkarzy czy klubów.

1. Spis rysunków

Rysunek 1: Widok Neuroph Studio 7

Rysunek 2: Schemat wzorca projektowego MVC 11

Rysunek 3: Schemat bazy danych 16

Rysunek 4: Strona główna 17

Rysunek 5: Ekran rejestracji 18

Rysunek 6: Ekran logowania 18

Rysunek 7: Ekran główny zalogowanego użytkownika 19

Rysunek 8: Ekran główny administratora 19

Rysunek 9: Profil użytkownika 21

Rysunek 10: Widok wszystkich użytkowników 22

Rysunek 11: Tworzenie meczu 23

Rysunek 12: Następne mecze 24

Rysunek 13: Wybieranie graczy składu wyjściowego 25

Rysunek 14: Wybieranie graczy do ławki rezerwowych 27

Rysunek 15: Składy meczowe 28

Rysunek 16: Wybieranie meczy do kuponu 29

Rysunek 17: Określanie przewidywanych rezultatów 30

Rysunek 18: Główna strona z komunikatem 31

Rysunek 19: Widok wszystkich kuponów użytkownika 31

Rysunek 20: Widok wszystkich singli z kuponu 33

Rysunek 21: Widok przed zasymulowaniem spotkania 34

Rysunek 22: Rezultat meczu wygenerowany przez sieć neuronową 35

Rysunek 23: Tabela ligi 37

Rysunek 24: Spis wszystkich klubów 38

Rysunek 25: Edycja klubu 39

Rysunek 26: Szczegółowy opis klubu 40

Rysunek 27: Historia klubu 41

Rysunek 28: Historia starć bezpośrednich 42

Rysunek 29: Model sieci wielowarstwowej 48

1. Spis tabel

Tabela 1: Rozwinięcie skrótów pozycji na boisku 26

Tabela 2: Prawdopodobieństwa wyliczane rozkładem Poisson 44

Tabela 3: Bonusowy współczynnik za formę zawodnika 50

Tabela 4: Wartości neuronów w warstwie oczekiwanej w zależności od rezultatu meczu 51

Tabela 5: Kolejka 12 rozgrywek Bundesligi sezonu 2017/2018 52

Tabela 6: Kolejka 13 rozgrywek Bundesligi sezonu 2017/2018 53

Tabela 7: Kolejka 14 rozgrywek Bundesligi sezonu 2017/2018 53

Tabela 8: Kolejka 15 rozgrywek Bundesligi sezonu 2017/2018 54

Tabela 9: Kolejka 16 rozgrywek Bundesligi sezonu 2017/2018 54

Tabela 10: Kolejka 17 rozgrywek Bundesligi sezonu 2017/2018 55

1. Spis równań

*Równanie () 32*

*Równanie () 32*

*Równanie () 43*

*Równanie () 43*

*Równanie (*5*) 43*

*Równanie () 44*

*Równanie () 44*

*Równanie () 44*

*Równanie () 44*

*Równanie () 44*

*Równanie () 45*

*Równanie () 46*

*Równanie () 46*

*Równanie () 46*

*Równanie () 46*

*Równanie () 46*

*Równanie (*17*) 46*

*Równanie () 47*

*Równanie () 50*

*Równanie () 51*

1. Bibliografia
2. Opis Session Fixation <http://wortal.php.pl/Wortal/Artykuly/Bezpieczenstwo/Sesja-uzytkownika-w-PHP-zagrozenia-i-ochrona/Session-Fixation> styczeń 2018
3. Opis Clickjacking <https://niebezpiecznik.pl/post/clickjacking-i-framebusting-czyli-ochrona-przed-clickjackingiem/> styczeń 2018
4. Opis CSRF <https://pl.wikipedia.org/wiki/Cross-site_request_forgery> styczeń 2018
5. Widok IDE Neuroph Studio <http://neuroph.sourceforge.net/images/screenshot-big.jpg> grudzień 2017
6. Strona producenta <https://www.webharvy.com/> listopad 2017
7. Strona piłkarska <https://www.transfermarkt.pl/> listopad 2017
8. Strona udostępniająca API <http://www.xmlsoccer.com/> styczeń 2018
9. Schemat wzorca projektowego <http://blog.karolak.it/wp-content/uploads/2010/11/cakephp_mvc.jpg> styczeń 2018
10. Podatki w polskich zakładach bukmacherskich <https://bukmacherzy24.net/wynosi-podatek-zakladow-bukmacherskich-polsce/> styczeń 2018
11. Kolejność drużyn w tabeli <https://en.wikipedia.org/wiki/Bundesliga#Competition_format> styczeń 2018
12. Użycie rozkładu Poissona w przewidywaniu wyniku meczu [https://www.pinnacle.com/pl/betting-articles/Soccer/how-to-calculate-poisson-distribution/MD62MLXUMKMXZ6A8 listopad 2018](https://www.pinnacle.com/pl/betting-articles/Soccer/how-to-calculate-poisson-distribution/MD62MLXUMKMXZ6A8%20listopad%202018)
13. Model sieci wielowarstwowej <http://web.utk.edu/~wfeng1/spark/_images/fnn.png> styczeń 2018

1. Opis Session Fixation <http://wortal.php.pl/Wortal/Artykuly/Bezpieczenstwo/Sesja-uzytkownika-w-PHP-zagrozenia-i-ochrona/Session-Fixation> styczeń 2018 [↑](#footnote-ref-2)
2. Opis clickjacking <https://niebezpiecznik.pl/post/clickjacking-i-framebusting-czyli-ochrona-przed-clickjackingiem/> styczeń 2018 [↑](#footnote-ref-3)
3. Opis CSRF <https://pl.wikipedia.org/wiki/Cross-site_request_forgery> styczeń 2018 [↑](#footnote-ref-4)
4. Widok IDE Neuroph Studio <http://neuroph.sourceforge.net/images/screenshot-big.jpg> grudzień 2017 [↑](#footnote-ref-5)
5. Strona producenta <https://www.webharvy.com/> listopad 2017 [↑](#footnote-ref-6)
6. Strona piłkarska <https://www.transfermarkt.pl/> listopad 2017 [↑](#footnote-ref-7)
7. Strona udostępniająca API <http://www.xmlsoccer.com/> styczeń 2018 [↑](#footnote-ref-8)
8. Schemat wzorca projektowego MVC <http://blog.karolak.it/wp-content/uploads/2010/11/cakephp_mvc.jpg> styczeń 2017 [↑](#footnote-ref-9)
9. Podatki w polskich zakładach bukmacherskich <https://bukmacherzy24.net/wynosi-podatek-zakladow-bukmacherskich-polsce/> styczeń 2018 [↑](#footnote-ref-10)
10. Kolejność drużyn w tabeli <https://en.wikipedia.org/wiki/Bundesliga#Competition_format> styczeń 2018 [↑](#footnote-ref-11)
11. Użycie rozkładu Poissona w przewidywaniu wyniku meczu <https://www.pinnacle.com/pl/betting-articles/Soccer/how-to-calculate-poisson-distribution/MD62MLXUMKMXZ6A8> listopad 2017 [↑](#footnote-ref-12)
12. Model sieci wielowarstwowej <http://web.utk.edu/~wfeng1/spark/_images/fnn.png> styczeń 2018 [↑](#footnote-ref-13)