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**

Blablabla….

**Abstract**

Blablabla…

**Spis treści**

1. **Wstęp**

W mojej pracy inżynierskiej zdecydowałem się na zrobienie aplikacji webowej, jaką jest serwis bukmacherski. Ten projekt powstał w mojej głowie ponad pół roku temu. Stwierdziłem, że taka aplikacja może być przydatna dla ludzi, którzy się zastanawiają, czy warto zaczynać swoją przygodę z zakładami bukmacherskimi. Taka rozrywka jest jedną z najpopularniejszych form hazardu w dzisiejszych czasach. Hazard wiąże się z ryzykowaniem własnymi pieniędzmi. Zanim ktoś postawi swój pierwszy zakład w życiu, długo się zastanawia, jakie mecze obstawić, czy zaznaczyć mecze, w których szansa na wygranie jest duża, ale zysk niewielki, czy jednak te bardziej ryzykowne, które oferują większy możliwy profit. Różnorodność opcji, jak zakład może wyglądać jest tak wielka, że początkujący hazardzista będzie miał dużo problemów z tworzeniem opłacalnych kuponów, ponieważ będzie się bał zaryzykować postawienie swoich pieniędzy na zwycięstwo jednej z drużyn w wyrównanym meczu. Moja aplikacja może być dużą pomocą dla takich osób. Użytkownicy nie używają własnych pieniędzy. Ten element sprawia, że ten serwis bukmacherski nie powoduje żadnej presji na klientach.

Celem tego projektu nie jest przekonanie użytkownika, że zakłady bukmacherskie to najmniej ryzykowna inwestycja pieniężna i nie musi się ich bać, ale pokazanie mu, czy jego przewidywanie jest na tyle trafne, że może spokojnie odwiedzać stoiska bukmacherskie, albo wręcz przeciwnie, wyperswadowanie mu tego pomysłu z głowy.

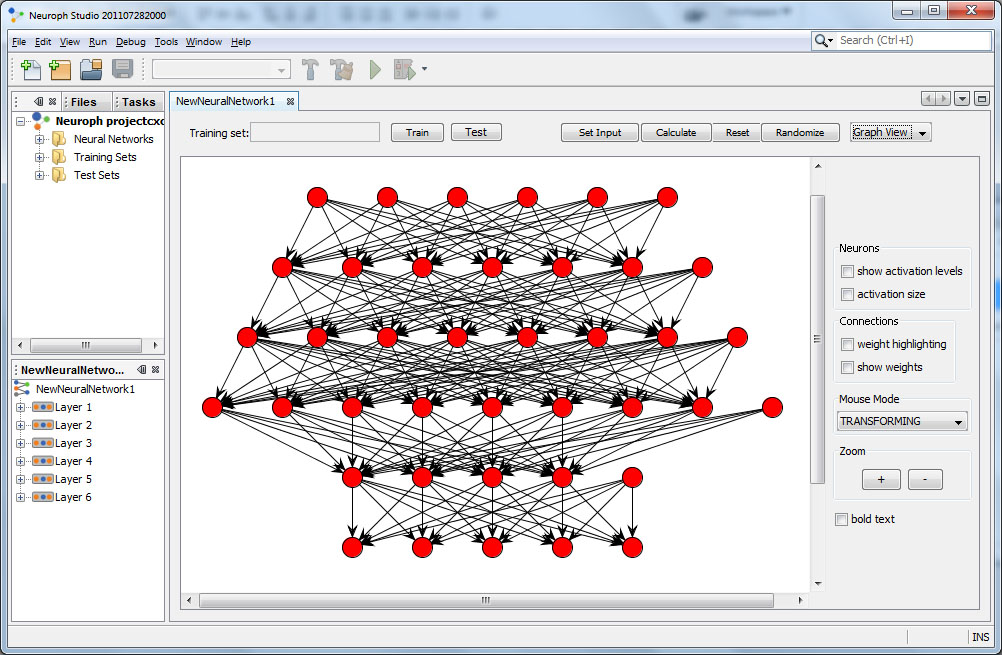
* 1. **Użyte technologie i frameworki**

**Spring Framework** jest szkieletem dla aplikacji napisanych w języku **JAVA EE** (Enterprise Edition). Ten projekt był moją pierwszą stycznością z tym frameworkiem. Dlatego zacząłem szukać różnych poradników, tutoriali w Internecie, jak stworzyć przykładową aplikację napisaną za pomocą wyżej wymienionej technologii. W trakcie tych poszukiwań znalazłem informacje o **Spring Boot**. Jest to rozwiązanie oferowane przez twórców technologii Spring, oparte na tym frameworku, w którym konfiguracja jest znacząco ułatwiona w porównaniu do konfiguracji aplikacji napisanych w „czystym” Springu. Ta zaleta przekłada się na szybkość pisania kodu. Dlatego zdecydowałem się stworzyć ten projekt właśnie w tej technologii.

Podczas nauki tego Framework korzystałem z poradnika znalezionego z Internecie, jak stworzyć podstawową aplikację w tym frameworku[[1]](#footnote-2). Dzięki temu tutorialowi nauczyłem się również korzystać z szablonów **Thymeleaf** oraz frameworka **Spring Security**. Wspomniane szablony generują kod HTML5, za pomocą którego wyświetlam dane zaciągnięte z bazy danych. Jest to możliwe, ponieważ twórcy tej biblioteki zapewnili możliwość integrowania tych szablonów z frameworkiem Spring. Spring Security udostępnia możliwość autoryzacji i autentykacji użytkowników korzystających z tego serwisu. Autoryzacja polega na sprawdzeniu, czy dany użytkownik posiada uprawnienia do wykonania określonych operacji. Celem autentykacji jest sprawdzenie, czy użytkownik, za którego się uważamy rzeczywiście istnieje w systemie (bądź bazie danych). W moim projekcie zapewniłem dostęp do większości stron tylko dla użytkowników, którzy mają określone role (administrator, klient). Każda rola posiada określone uprawnienia, które są weryfikowane przez Framework Spring Security.

Zastanawiając się, którą bazę danych wybrać kierowałem się ilością wpisów w Internecie, które pomagały czy to w rozwijaniu aplikacji, czy to w rozwiązywaniu problemów[[2]](#footnote-3). W taki sposób zdecydowałem się na bazę **MySQL**.

Do zaimplementowania sztucznej inteligencji użyłem biblioteki **Neuroph**. Jest to Framework do tworzenia sieci neuronowych. Został on stworzony 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[[3]](#footnote-4)*

* 1. **Skąd taki pomysł**

Od najmłodszych lat interesowałem się piłką nożną. Kiedy na studiach informatycznych dostaliśmy dowolność wyboru tematu projektu, który mamy stworzyć na dany przedmiot, nie zastanawiałem się długo i zdecydowałem się napisać program związany z piłką nożną. Na początku to były zwykłe programy, w których wpisywało się wynik meczu, a aplikacja pokazywała jak wygląda tabela ligi lub drabinka turniejowa. Wtedy błędnie nazywałem te projekty „symulatorami”. Dopiero na jednych zajęciach na naszej uczelni zrozumiałem znaczenie tego słowa. Doktor prowadzący przedmiot powiedział mi, że taki program piłkarski, który jest symulatorem powinien sam ustalać wynik meczu. Pierwszą moją myślą wtedy było zwykłe losowanie liczby z jakiegoś przedziału, ale szybko odrzuciłem ten pomysł. Wyniki meczu uzyskane w taki sposób nie miałyby za wiele wspólnego z rzeczywistością. Ten sam Doktor poradził mi zapoznanie się z tematem sieci neuronowych. One mogłyby się nauczyć przewidywać wynik meczu, jeśli dostałyby odpowiednie dobrane dane dotyczące poprzednich gier.

Na jednych z przedmiotów mieliśmy stworzyć aplikację internetową (z użyciem technologii Spring) z relacyjną bazą danych. Wiedziałem, że chcę stworzyć aplikację związaną z piłką nożną, jednak nie chciałem po raz kolejny robić tej samej aplikacji, chciałem ją jakoś rozwinąć, dodać element biznesowy. Wtedy przyszedł mi do głowy pomysł serwisu bukmacherskiego.

Wiele osób w moim otoczeniu reagowało negatywnie na mój pomysł, bo obstawianie zakładów bukmacherskich to hazard, a to słowo jest odbierane na ogół negatywnie. Ja jednak nie patrzyłem na ten pomysł w ten sposób.

Podczas pisania tego projektu polubiłem technologię Spring Boot, i szkoda mi było zostawiać ten projekt. Dlatego chciałem go użyć w pracy inżynierskiej. Kiedy się zastanawiałem, jak go rozwinąć, przypomniał mi się temat sieci neuronowych. Sam byłem bardzo ciekawy dwóch wyników tego projektu:

* Czy da się przewidzieć za pomocą sztucznej inteligencji rezultat meczu?
* Jaki dochód można uzyskać prowadząc taki serwis?

Każdy z tych problemów opisałem w dalszej części tej pracy. Wiedziałem, że aby sztuczna inteligencja mogła jak najlepiej przewidzieć wynik meczu, musiałem znacznie rozbudować istniejącą bazę danych. Jak się później okazało, był to jednak duży problem.

Na początku szukałem darmowego dostępnego API, które oferuje wszystkie potrzebne dla mojego projektu dane. Niestety, nie znalazłem żadnej strony, która oferowałaby pożądane informacje bez żadnych opłat. Dlatego postanowiłem zebrać dane w inny sposób. Z użyciem program **WebHarvy** można odczytać dane ze strony (inaczej mówiąc „zescrappować”) i zapisać je w dowolnym formacie (np. XML, JSON, CSV). Do tej operacji skorzystałem ze strony transfermarkt[[4]](#footnote-5). Pomimo tego, że na początku wydawało się to dobrym rozwiązaniem, to równie szybko okazało się, że nie będę w stanie zapisać wszystkich danych przed końcem semestru. Dlatego pozostałe dane pobrałem z płatnego API[[5]](#footnote-6), które zawierało potrzebne informacje.

Kiedy już miałem wszystkie niezbędne pliki zapisane, kolejnym problemem okazał się alfabet. Dane, które pobrałem z API nie posiadały wszystkich znaków specjalnych, przez co piłkarz, który nazywał się *„Jérôme Boateng”* był innym zawodnikiem niż *„Jerome Boateng”*. Poprawienie tych nieścisłości również zajęło trochę czasu.

1. **Aplikacja webowa**

W tym rozdziale opiszę aplikację webową, jej funkcjonalności dla klienta i administratora oraz sposób działania.

* 1. **Opis serwisu**

Strona bukmacherska jest aplikacją skierowaną dla ludzi interesujących się rozgrywkami sportowymi. Wszystkich użytkowników mojej aplikacji można podzielić na kilka ról:

* Administrator
* 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 funkcjonalności jest dostępnych dla wszystkich zalogowanych osób, jednak jako użytkownik niezalogowany mamy również kilka możliwości.

Oto najważniejsze funkcjonalności przeznaczone dla administratora:

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

Serwis bukmacherski jest stroną, która musi posiadać dużą ilość danych na temat rozgrywanych meczów. Jest to ważne z punktu widzenia klienta, który decydując się, czy postawić zakład na daną drużynę, chętnie sprawdzi np. jak klub się spisywał w przeciągu kilku ostatnich gier, jakich piłkarzy posiada czy też na którym miejscu w lidze się znajduje. Dlatego moja aplikacja oferuje dużą ilość różnych możliwości skierowanych głównie dla klienta. Oto niektóre z nich:

* Rejestracja/logowanie
* Sprawdzenie tabeli ligi
* Sprawdzenie danych drużyny

Z materialnego punktu spojrzenia udostępniłem 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
  1. **Schemat bazy danych**

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

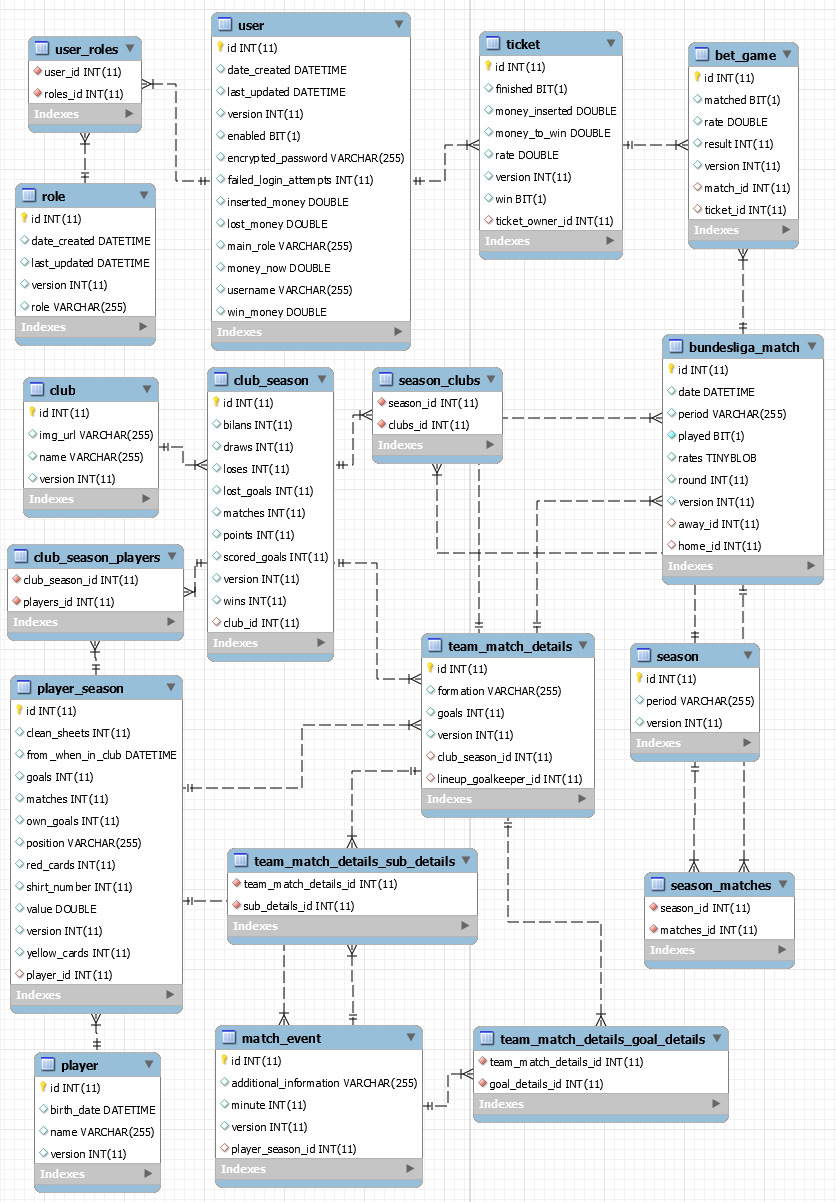
Podstawową tabelą w tej aplikacji jest tabela **USER**. Przechowuje ona dane użytkownika, takie jak nazwa użytkownika, zaszyfrowane hasło, ilość pieniędzy na koncie czy rolę użytkownika. Domyślnie po rejestracji każdy użytkownik posiada rolę *„user”*.

Każdy użytkownik ma przypisaną listę kuponów, które kiedyś stworzył. Trzymane są one w tabeli **TICKET**. Każdy kupon przechowuje informacje między innymi o tym ile użytkownik postawił pieniędzy, ile może wygrać oraz czy dany kupon został już rozstrzygnięty. Dodatkowo do każdego z nich jest przypisanych kolekcja tzw. singli, inaczej mówiąc zakładów pojedynczych, singlowych. Jest to zdarzenie, które dotyczy pojedynczego meczu. W zdarzeniu zapisane są informacje o tym jaki rezultat użytkownik przewidział, i czy się zgadzał z rozegraną już grą. Te dane znajdują się w tabeli **BET\_GAME**.

Najbardziej zewnętrzną tabelą trzymającą informacje o rozgrywkach jest tabela **SEASON**. Każdy rekord przechowywał lata trwania, mecze wtedy rozegrane oraz kluby, które w nim 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** i dodatkowo zawiera zarówno informacje personalne piłkarza (imię i nazwisko, data urodzenia) jak i dane piłkarskie (numer na koszulce, pozycja, wartość rynkowa, ilość meczów, strzelonych bramek oraz meczów bez traconej 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 2: Schemat bazy danych*

* 1. **Jakiś rozdział**

1. **Spis rysunków**

*Rysunek 1: Widok Neuroph Studio* 5

*Rysunek 2: Schemat bazy danych* 12

1. **Spis równań**
2. **Bibliografia**
3. Poradnik jak stworzyć podstawową aplikację za pomocą frameworka Spring Boot <https://springframework.guru/spring-boot-web-application-part-1-spring-initializr/> maj 2017
4. Konfiguracja aplikacji napisanej w Springu pod bazę danych MySQL <https://spring.io/guides/gs/accessing-data-mysql/>, październik 2017
5. Widok IDE Neuroph Studio <http://neuroph.sourceforge.net/images/screenshot-big.jpg> grudzień 2017
6. Strona <https://www.transfermartk.pl/> listopad 2017
7. Strona z API <http://www.xmlsoccer.com/> styczeń 2018

1. Poradnik jak stworzyć podstawową aplikację za pomocą frameworka Spring Boot <https://springframework.guru/spring-boot-web-application-part-1-spring-initializr/> maj 2017 [↑](#footnote-ref-2)
2. Konfiguracja aplikacji napisanej w Springu pod bazę danych MySQL: <https://spring.io/guides/gs/accessing-data-mysql/>, maj 2017 [↑](#footnote-ref-3)
3. Widok IDE Neuroph Studio <http://neuroph.sourceforge.net/images/screenshot-big.jpg> grudzień 2017 [↑](#footnote-ref-4)
4. Strona <https://www.transfermarkt.pl/> listopad 2017 [↑](#footnote-ref-5)
5. Strona z API <http://www.xmlsoccer.com/> styczeń 2018 [↑](#footnote-ref-6)