|  |
| --- |
| **Politechnika Wrocławska** |
| **STUDIA PODYPLOMOWE Technologie Internetowe** |
| Projekt i implementacja systemu społecznościowego do gry w szachy w sieci internetowej |
|  |
| Marcin Kużdowicz |
| **[Wybierz datę]** |

|  |
| --- |
| Wrocław 2015 |

Opiekun pracy: Doc. dr inż. Zbigniew Szpunar

Spis treści:

[1. Cel i zakres pracy Error: Reference source not found](#_Toc399016126)

[2. Opis wycinka rzeczywistości Error: Reference source not found](#_Toc399016127)

[2.1 Wstęp Error: Reference source not found](#_Toc399016132)

[2.2 Słownik pojęć biznesowych Error: Reference source not found](#_Toc399016133)

[2.3 Opis wycinka rzeczywistości Error: Reference source not found](#_Toc399016134)

[3. Definicja modelu domenowego oraz definicja reguł biznesowych Error: Reference source not found](#_Toc399016135)

[4. Specyfikacja wymagań funkcjonalnych i niefunkcjonalnych Error: Reference source not found](#_Toc399016136)

[4.1 Wymagania funkcjonalne Error: Reference source not found](#_Toc399016139)

[4.1.1 Ogólne wymagania systemowe Error: Reference source not found](#_Toc399016148)

[4.1.2 Perspektywa Administratora Error: Reference source not found](#_Toc399016149)

[4.1.3 Perspektywa Recepcjonisty Error: Reference source not found](#_Toc399016150)

[4.1.4 Perspektywa Pacjenta Error: Reference source not found](#_Toc399016151)

[4.1.5 Perspektywa Lekarza Error: Reference source not found](#_Toc399016152)

[4.2 Wymagania niefunkcjonalne Error: Reference source not found](#_Toc399016153)

[4.3 Diagram przypadków użycia Error: Reference source not found](#_Toc399016154)

[4.4 Prototyp interfejsu użytkownika Error: Reference source not found](#_Toc399016155)

[5. Wybór technologii, ogólna architektura systemu i opracowanie modelu danych oraz bazy danych Error: Reference source not found](#_Toc399016156)

[5.1 Wybór technologii Error: Reference source not found](#_Toc399016157)

[5.2 Ogólna architektura systemu Error: Reference source not found](#_Toc399016158)

[5.3 Model bazy danych Error: Reference source not found](#_Toc399016159)

[6. Realizacja systemu Error: Reference source not found](#_Toc399016160)

[7. Testy Error: Reference source not found](#_Toc399016161)

[8. Prezentacja użytkowania systemu Error: Reference source not found](#_Toc399016162)

[9. Podsumowanie Error: Reference source not found](#_Toc399016163)

[Literatura Error: Reference source not found](#_Toc399016164)

[Spis rysunków Error: Reference source not found](#_Toc399016165)

[Spis tabel Error: Reference source not found](#_Toc399016166)

[Załączniki Error: Reference source not found](#_Toc399016167)

[Dodatek A Prototyp interfejsu użytkownika Error: Reference source not found](#_Toc399016168)

[Dodatek B Skrypty DDL Error: Reference source not found](#_Toc399016169)

1. **Cel pracy:**

Celem pracy jest zaprojektowanie i implementacja systemu informatycznego, przy użyciu poznanych w toku studiów technologii internetowych, służącej do gry w szachy w sieci internetowej. Użytkownik aplikacji może grać z komputerem lub po utworzeniu konta i zalogowaniu z innym użytkownikiem w zależności od tego czy inni użytkownicy są w tym momencie zalogowani do systemy i obecni w repozytorium gry online. Aplikacja również ma gromadzić i wyświetlać dane statystyczne na temat wyników pojedynków szachowych użytkowników. Ma wyświetlać dla wszystkich widocznych dziesięciu najlepszych graczy, czyli takich którzy wygrali najwięcej pojedynków szachowych, pogrupowanych w według zdobytych punktów w postaci wykresu słupkowego. Program ma również gromadzić dane na temat odbytych gier poszczególnych użytkowników oraz prezentować aktualnie zalogowanemu użytkownikowi w jego panelu gry odbyte przez niego.

1. **Opis wycinka rzeczywistości.**
   1. **Wstęp**

Jednym z najciekawszych funkcji współczesnego Internetu jest możliwość komunikacji w czasie rzeczywistym miedzy dwoma osobami znajdującymi się fizycznie w innych miejscach. Funkcjonalność ta może służyć do wymiany słów między ludźmi w formie tekstowej, jak również wymianą informacji nie tekstowej, co jest nową sferą komunikacji udostępnionej przez nowe technologie.

Interesującym tematem są tak zwane gry komputerowe online. Sferą zainteresowań tej pracy jest stworzenie takiego właśnie programu, który będzie służył do gry online miedzy użytkownikami Internetu.

Przyczyny podjęcia tematu

Sam temat jest interesujący pod wzglądem programistycznym, gdyż będzie wymagał użycia szerokiej palety technologii, jak i zastosowanie złożonej logiki mającej na celu obsłużyć komunikację w czasie rzeczywistym.

Głównie z tego względu został podjęty.

Aplikacja ma na celu umożliwić zalogowanym użytkownikom prowadzenie pojedynków szachowych między sobą online.

* 1. **Słownik pojęć biznesowych**

|  |  |
| --- | --- |
| Użytkownik | Osoba korzystająca z oprogramowania |
| Administrator | Osoba korzystająca z oprogramowania i mająca uprawnienia do tego aby zarządzać użytkownikami |
| Szachy | Rodzaj gry planszowej będący głównym tematem niniejszego oprogramowania |
| Szachownica | Obiekt stanowiący, w której odbywa się gra |
| Pojedynek szachowy online | Gra w szachy pomiędzy dwoma użytkownikami w czasie rzeczywistym za pomocą oprogramowania |
| Lista najlepszych graczy | Posortowana Lista dziesięciu użytkowników, którzy wygrali największą liczbę pojedynków szachowych |
| Figura szachowa |  |
| Kolor figury szachowej |  |
| Status gry | Aktualna sytuacja jaka występuje w grze: ruch dla figur białych, szach, szach mat. |
| Fen | Notacja Forsytha-Edwardsa. Ciąg znaków reprezentujący, aktualny status gry, położenie wszystkich figur na szachownicy; |
| Ruch szachowy | Ruch szachowy |
| Komunikator do gry online | Część oprogramowania służąca do wysyłania ruchów szachowych między dwoma użytkownikami przez sieć internetową |
| Oczekiwanie na nową grę | Stan w jakim znajduje Się użytkownik, który w danym momencie z nikim nie gra a jest online |
| Zaproszenie do gry | Stan w jakim znajduje Się użytkownik, został zaproszony do gry lub sam zaprosił do gry |
| Stan w trakcie gry | Stan w jakim znajduje Się użytkownik, który aktualnie prowadzi grę z innym użytkownikiem |
| Pojedynek szachowy | Dwóch użytkowników grających przeciwko sobie |
| Lista najlepszych graczy | 10 najlepszych graczy posortowanych według ilości wygranych pojedynków szachowych |
| Wiadomość od gracza do gracza | Wysłany ruch szachowy i aktualny status gry od jednego użytkownika do drugiego po wykonanym ruchu |

* 1. Opis wycinka rzeczywistości

**Profil działalności**

Użytkownik sieci internetowej, zainteresowany grą w szachy, oraz grami online.

**Wizja systemu**

Dostęp do aplikacji ma się odbywać poprzez sieć internetową. Każdy użytkownik Internetu może odwiedzić stronę z aplikacją, ale ma dostęp tylko do jej niektórych zasobów. Do wszystkich zasobów ma dostęp dopiero po założeniu konta i zalogowaniu. Do zasobów administracyjnych ma dostęp wyłącznie użytkownik z rolą administratora.

System przewiduje dwie role:

1. Administrator
2. Użytkownik

Rola Użytkownika powstaje poprzez rejestrację w systemie, za pomocą formularza do rejestracji obecnego w aplikacji. Użytkownik musi podać login, który ma być unikalny dla systemu, nie mogą wystąpić takie same nazwy użytkownika. Jeżeli login podany podczas rejestracji już istnieje, zostanie wyświetlony komunikat walidacyjny, aby stworzyć inny login. Oprócz tego użytkownik podaje hasło, potwierdzenie hasła oraz adres email. Następnie konto zostaje utworzone, ale jeszcze należy je aktywować. Na podany adres email zostanie wysłany link aktywacyjny. Jeżeli konto nie zostanie potwierdzone w przeciągu tygodnia to zostanie usunięte.

1. **Specyfikacja wymagań funkcjonalnych i niefunkcjonalnych**

**3.1 Wymagania funkcjonalne**

**3.1.1 Ogólne wymagania systemowe**

Na podstawie przeprowadzonych badań sformułowano następujące wymagania funkcjonalne:

WF/01 dodawanie/usuwanie użytkownika przez administratora

WF/02 tworzenie kont użytkowników przez użytkowników internetu poprzez formularz tworzenia konta użytkownika, weryfikowanych za pomocą linka aktywacyjnego wysyłanego, na wcześniej podany adres email

WF/03 możliwość gry w szachy z komputerem

WF/04 możliwość gry w szachy z innym użytkownikiem w czasie rzeczywistym

WF/05 przeglądanie/danych na temat odbytych wcześniej pojedynków szachowych z innymi użytkownikami

WF/06 przeglądanie aktualnego rankingu najlepszych dziesięciu graczy w postaci wykresu słupkowego

WF/07 pojedynek szachowy wstępuje w postaci użytkownik kontra użytkownik

WF/08 użytkownik może prowadzić tylko jeden pojedynek w czasie rzeczywistym z innym użytkownikiem

WF/09 użytkownik przed rozpoczęciem pojedynku musi zaprosić drugiego użytkownika do gry. Zaproszony użytkownik ma 15 sekund na decyzje, w momencie trwania zaproszenia, dana para użytkowników już jest zablokowana i inni użytkownicy nie mogą do nich wysłać wiadomości.

WF/10 użytkownik po rejestracji nie może zmienić swojego login, gdyż login jest identyfikatorem w grze online dal innych użytkowników

WF/11 użytkownik może edytować po rejestracji swoje dane takie jak: adres email, imię, nazwisko. Może również zmienić hasło

WF/12 administrator może przeglądać listę użytkowników

* + 1. **Perspektywa administratora**
* Administrator ma te same uprawnienia co zalogowany użytkownik
* Administrator ma dostęp do tych samych zasobów co zalogowany użytkownik
* Administrator systemu może dodawać nowych użytkowników
* Usuwać istniejących.
* Tworzyć konta z uprawnieniami administratora
* Nadawać uprawnienia administratora lub je odbierać.
  + 1. **Perspektywa Użytkownika**

Niezalogowany użytkownik ma dostęp do:

* Strony domowej
* możliwości zagrania w szachy z komputerem, aby zapoznać się z systemem. Wyniki pojedynków z komputerem nie są zapisywane do bazy, pełnią rolę warstwy demonstracyjnej.
* wykresu słupkowego z maksymalnie 10 najlepszymi graczami (przy założeniu że w bazie mamy dziesięciu graczy, którzy odbyli zwycięskie pojedynki w szachowe. Jeżeli jest ich mniej, to wyświetlana jest mniejsza liczba użytkowników)
* formularza logowania
* formularza z możliwością utworzenia konta

Zalogowany użytkownik ma dostęp do:

Tych samych zasobów co użytkownik niezalogowany, plus:

* Możliwość grania w szachy z innymi użytkownikami online.
* Danych na temat swojego konta i możliwością ich edycji z wyłączeniem zmiany loginu.
* Danych na temat odbytych pojedynków szachowych
  1. **Wymagania niefunkcjonalne**

Program jest aplikacją webową

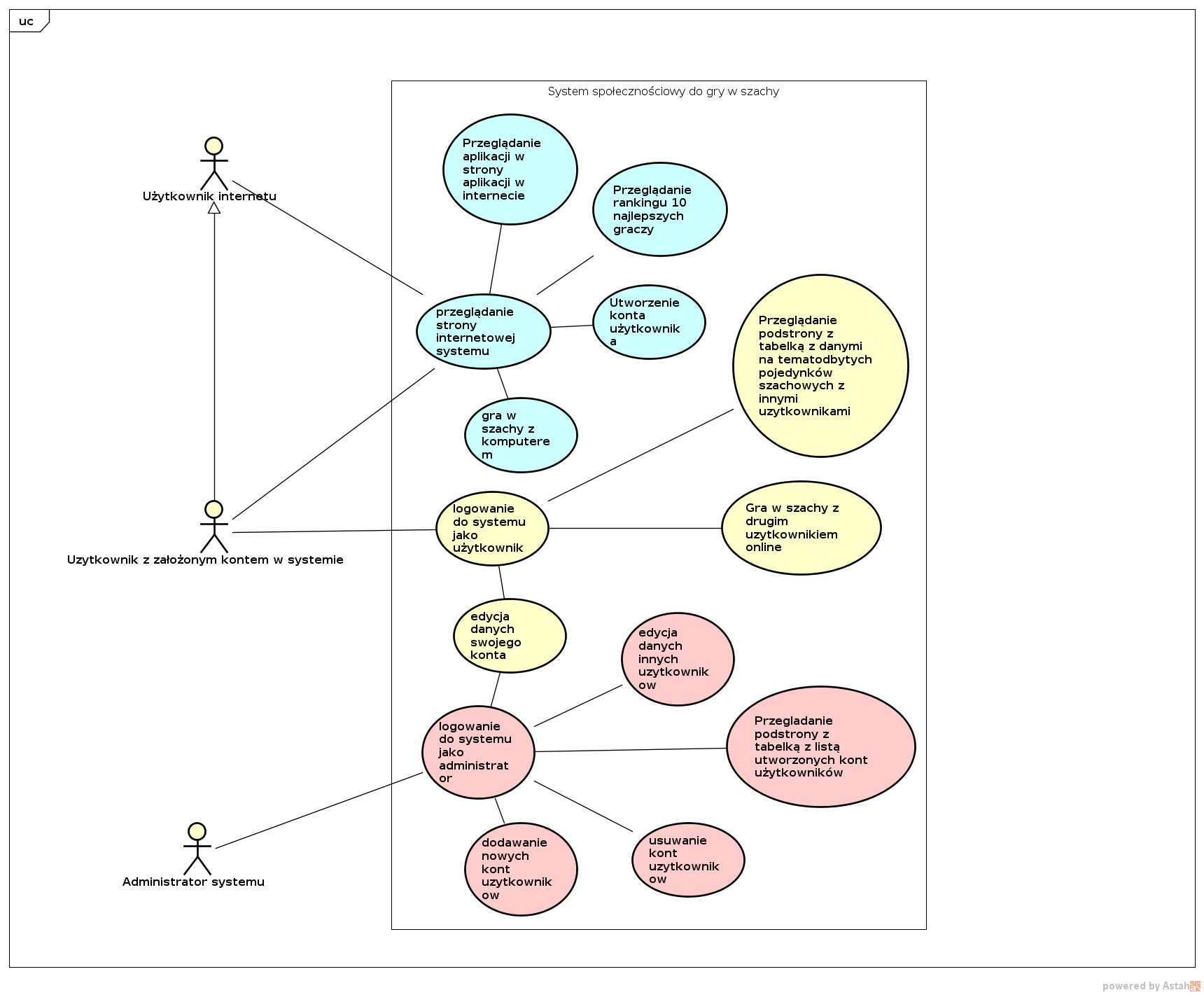
Program ma czytelny interfejs dla użytkownika

Program ma służyć rozrywce

Program propaguję grę w szachy w sieci

Program ma rozwijać umiejętności logicznego myślenia u użytkownika

* 1. **Diagram przypadków użycia**

****

1. **Wybór technologii, ogólna architektura systemu i opracowanie modelu danych oraz bazy danych**

**4.1 Wybór technologii**

* Java – obiektowy język programowania, główna technologia użyta w aplikacji do stworzenia kodu działającego po stronie serwera.

Więcej informacji: <https://www.java.com/pl/>

* Spring MVC – Framework języku java wspierający architekturę Model-View-Controller i gotowe komponenty, które mogą być wykorzystane do opracowania aplikacji internetowych.

Więcej informacji: <http://spring.io/>

* Spring security - Framework języku java wspierający zagadnienie bezpieczeństwa takie jak uwierzytelnianie, autoryzację.

Więcej informacji: <http://spring.io/>

* JSP (Java server pages) - to technologia umożliwiająca tworzenie dynamicznych dokumentów WWW z wykorzystaniem języka Java, wplecionego w kod HTML danej strony. Jsp jest formatem pliku w którym można wykorzystać język zarówno htm jak i Java. Skłądnia jsp składa się na elementy: html, elementy skryptowe skryplety (kod w języku Java kontrolujący proces generowania dokumentu) oraz elementy składniowe tzw. Expression Language oraz tagi XML wywołujące określone metody.
* JSTL (Java Standard Tag Library) – kolekcja tagów z przypisanymi funkcjonalnościami. Jstl wspiera typowe zadania takie jak iteracja czy konstrukcje warunkowe, wykonywane na widoku.
* Jetty – serwer www oraz kontener serwletów Javy. Więcej informacji: <http://www.eclipse.org/jetty/>
* Maven - narzędzie do zarządzania projektem Javowym. Projekt mavenowy definiuje się poprzez stworzenie i utrzymywanie pliku pom.xml (POM – ang. Project Object Model). **Pom.xml** jest głównym miejscem pracy z projektem i zawiera wszystkie istotne elementy definiujące projekt, jego strukturę, sposób budowania i przede wszystkim zależności. Zależności to zewnętrzne biblioteki importowane do projektu w postaci plików .jar

Więcej informacji: <http://maven.apache.org/>

* JSON – (Javascript Object Notation) format wymiany danych komputerowych, przypominający swoją strukturą tzw. Hashmapę, organizujący dane w postaci (klucz=wartość). Jest to format transportowy, którym posłużyłem się do wysyłania ruchów szachowych pomiędzy użytkownikami, aktualnego stanu gry, zaproszenia do gry czy zrezygnowania z gry.
* XML (Extensible Markup Language) czyli rozszerzalny język znaczników. W projektach javowoych bardzo często używany do plików konfiguracyjnych, oraz do tagów z przypisanymi funkcjonalnościami za pomocą zaimportowanych przestrzeni nazw.
* Websocket - jest technologią zapewniającą dwukierunkowy kanał komunikacji (full-duplex) za pośrednictwem jednego połączenia TCP. Websocket jest niezależnym protokółem opartym na protokole TCP. W niniejszej aplikacji ta technologia została użyta do stworzenia komunikatora wysyłającego ruchy szachowe, zaproszenia do gry, rezygnacje z gry, koniec gry jaki listę aktualnie połączonych użytkowników. Dzięki gra w szachy zaimplementowana w programie przybrała charakter tzw. *real-time games* (gra czasu rzeczywistego);
* JavaScript - język skryptowy wykonujący się po stronie klienta w przeglądarce, dostarczający programiście interfejsu użytkownika programowania zdarzeniowego. Interfejs do gry w szachy jak i logika gry w szachy jest obsługiwana zdarzeniowo za pomocą javascript drag and drop (m.in. zdarzenia drag start i dragstop).

Więcej informacji: <https://www.javascript.com/>

* Jquery – biblioteka jasvacritpu dostarczająca dodatkowe narzędzia i umożliwiająca zaprogramowanie tych samych funkcjonalności za pomocą mniejszej ilości kodu.

Więcej informacji: <https://jquery.com/>

* Bootstrap 3 – biblioteka css i javascript do tworzenia interfejsów użytkownika stworzona przez programistów Twittera. Główną cechą bootsrapa są dostarczane przez niego gotowe klasy css, gwarantujące dobry wygląd interfejsu i jego responsywność.

Więcej informacji: <https://jquery.com/>

* MongoDB – baza danych typu NoSql.   Charakteryzuje się dużą skalowalnością, wydajnością oraz brakiem ściśle zdefiniowanej struktury obsługiwanych baz danych. Zamiast tego dane składowane są jako dokumenty w stylu JSON, co umożliwia aplikacjom bardziej naturalne ich przetwarzanie, przy zachowaniu możliwości tworzenia hierarchii oraz indeksowania. Do aplikacji typu live-game baza danych tego typu jest dobrym wyborem ze względu na szybkość działania i dynamiczne dostosowywanie się do zapisywanych do niej obiektów. Cechą charakterystyczna jest to że gdy chcemy do tzw. Kolekcji (odpowiednik tabelki w bazach typu sql) dodać dodatkowe pole, to nie musimy najpierw go tworzyć, ono samo się utworzy przy pierwszym zapisie, co jest bardzo efektywne.

Więcej informacji: <https://www.mongodb.com/>

* Html 5 - jest najnowszą wersją standardu opisującego język HTML. Zalety tej technologii zostały użyte np. przy walidacji formularzy po stronie klienta np. za pomocą atrybutu required czy type=”email” oraz w przekazywaniu danych do javascriptu za pomocą data attributes.

Więcej informacji: <http://www.w3.org/TR/html5/>

* CSS 3 – kaskadowe arkusze stylów, język odpowiedzialny za wygląd aplikacji.

Więcej informacji: <http://www.css3.info/>

* DataTables – plugin do jquery umożliwający szybkie oprogramowanie tabelki wraz z paginacją zaimplementowaną po stronie klienta w javascript.

Więcej informacji: <https://www.datatables.net/>

* Canvasjs – biblioteka javascript umożliwiająca wyświetlenie responsywnych histogramów, słupków, wykresów prezentujących dane statystyczne w przyjazny dla użytkownika formacie. W programie użyta do wyświetlenia rankingu 10 najlepszych graczy w postaci wykresu słupkowego.

Więcej informacji: <https://www.datatables.net/>

* Chessboardjs - biblioteka javascript dostarczająca interfejs graficzny do gry w szachy.

Więcej informacji: <http://chessboardjs.com/>

* Chess.js - biblioteka javascript implementująca logikę gry w szachy umożliwiająca integrację z Chessboardjs i obsługę oprogramowanie gry w szachy zdarzeniowo przez programistę.

Więcej informacji: <https://github.com/jhlywa/chess.js/>

* GIT – System kontroli wersji umożliwiający zarządzanie poszczególnymi wersjami aplikacji i przechowywanie wszystkich wersji na lokalnym i zdalnym repozytorium.

Więcej informacji: <https://git-scm.com/>

**4.2 Ogólna Architektura aplikacji.**

Program jest oparty na wzorcu architektonicznym MVC (model view controller) zaimplementowanej przy pomocy komponentów dostarczanych przez framework Spring MVC.



Wszystkie klasy Javowe znajdują się w pakiecie **src/main/java**

Wszystkie requesty HTTP obsługują klasy w pakiecie **com.chessApp.controllers .**

Te klasy pełnią rolę **kontrolerów**. Są opatrzone anotacji @Controller co informuje Springa o tym że ta klasa jest kontrolerem.

**Widok** jest realizowany za pomocą technologii: Jsp (java server pages) , Jtsl (JavaServer Pages Standard Tag Library), HTML 5, CSS 3, JavaScript i Jquery oraz Bootstrap . Pliki widoków znajdują się w głównym folderze: **/src/main/webapp**

w folderze **assets** są pliki css i javascript

w folderze **assets/js** są pliki źródłowe javascript, w folderze **assets/js/lib** są biblioteki javascript

w folderze **assets/css** są pliki źródłowe css, w folderze **assets/css/lib** są biblioteki css

w folderze **WEB-INF/** **views** są pliki .jsp

**Model** jest zmapowany do klas javovwych w pakiecie **com.chessApp.model.**

Aplikacja wykorzystuje bazę danych MonogDB, która jest bazą NoSQL. W programie jest zastosowana koncepcja **ORM** (object relational mapping), za pomocą technologii Spring Data. ORM mapuje obiekty i kolekcje z bazy danych MongoDB do klas Javovych i dzięki zaletmo jakie daje MongoDB, ten mechanizm zachodzi też odwrotnie.

Oprócz tego w folderze **WEB-INF** znajdują się pliki konfiguracyjne .xml

**5.2 Model bazy danych**

Nazwa bazy: chessapp\_db

Konfiguracja połaczenia z bazą danych zawiera się w pliku **mongo-config.xml** pod ścieżką **/src/main/webapp/ WEB-INF**

MongoDb nie jest bazą relacyjną, nie występuje tu koncepcja kluczów obcych, natomiast jest możliwość przechowywania zagnieżdżonych obiektów i list typów prostych i stringów jak również list obiektów. Ta właściwość została wykorzystana w implementacji aplikacji: kolekcja chessGames przechowuje liste obiektów Move, co reprezentuje ruch szachowy.

|  |
| --- |
| users |
|  |

|  |
| --- |
| chessGames |
| List<Move> |

Model zmapowany do obiektów javowych:

|  |
| --- |
| UserAccount |
|  |

|  |
| --- |
| chessGame |
| List<Move> |

|  |
| --- |
| Move |
|  |

Tak więc Można powiedzieć że występuje tutaj relacja Has-a kolekcja chessGames zawiera obiekty Move

**5. Realizacja systemu**

**5.1**. **System do gry w szachy.**

Interfejs do gry w szachy oraz logika silnika do gry jest napisana po stronie klienta w języku javascript. Na ten moment realizują to biblioteki

**chessboard.js** (interfejs)

<http://chessboardjs.com/>

**chess.js** (silnik gry)

<https://github.com/jhlywa/chess.js/blob/master/README.md>

Łatwiej było mi na początek zaprogramować aplikację na czymś wzorcowym, postaram się zaimplementować później swoje wersje.

Te biblioteki też wymagały programowania, pliki zawierające kod prograsmujący te biblioteki pod tą aplikacji znajdują się w katalogu:

**/src/main/webapp/assets/js/**

**5.2. Gra w szachy użytkownik vs użytkownik.**

Możliwość gry online wymagała zaprogramowania komunikatora.

Do tego użyłem technologii Websocket. Klasy implementujące komunikator znajdują się w pakiecie: **com.chessApp.websocket.** Implementacja komunikatora opiera się na obsłudze zdarzeń po stronie serwera w postaci **Server endpoint (oprogramowany w Javie) i** po stronie klienta w postaci Client endpoint (**oprogramowany w JavaScript**), oraz przechowywaniu informacji w sesji połaczenia realizowany przez subprotokół websocket.

Server endpoint jest zaimplementowany w pliku **WebSocketServer.java** w pakiecie **com.chessApp.websocket.**

Client ednpoint w pliku **websocketClientEndpoint.js** obecnym w katalogu pod ścieżką **/src/main/webapp/assets/js/**

Stan gry I ruchy poszczególnych użytkowników są przkazywane między nimi za pomocą notacji

**FEN:** **Notacja Forsytha-Edwardsa.** Polega ona na zapisaniu położenia wszystkich figur I do kogo należy następny ruch w postaci ciągu znaków. Przykład notacji fen:

**„r1bqkbnr/pppp1ppp/2n5/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R”**

Do pracy nad aplikacją używałem systemu do kontroli wersji GIT.