**POLITECHNIKA BIAŁOSTOCKA**

**WYDZIAŁ** **Informatyki**

**Katedra Systemów Informacyjnych i Sieci komputerowych**

PRACA DYPLOMOWA INŻYNIERSKA

TEMAT: **Aplikacja webowa wspomagająca działanie klubu sportowego**

WYKONAWCA: Maciej Ziniewicz

*Imię i nazwisko*

PODPIS: .................................

PROMOTOR: dr inż. Jacek Grekow

*Imię i nazwisko*

PODPIS: .................................

**BIAŁYSTOK 2016 ROK**

**Karta dyplomowa**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| POLITECHNIKA BIAŁOSTOCKA | | |  | | | Nr albumu studenta  90563. | |
| Wydział Informatyki | | | Studia stacjonarne  I stopnia inżynierskie | | | Rok akademicki  2015/2016 | |
| Kierunek studiów  Informatyka | |
| Katedra Systemów Informacyjnych i Sieci komputerowych | | |  | | | Specjalność  - | |
| **Maciej Ziniewicz**  **TEMAT PRACY DYPLOMOWEJ:**  **Aplikacja webowa wspomagająca działanie klubu sportowego.**  **Zakres pracy:**   1. Przegląd podobnych rozwiązań 2. Projekt systemu 3. Implementacja modułu administracji, zarządzania zawodami, galerii zdjęć. 4. Implementacja systemu z użyciem technologii JavaScript, Java, Spring Boot, Thymeleaf, Boostrap, MySQL   **Słowa kluczowe (max 5):** serwis internetowy, Java | | | | | | | |
| .........................................................................  *Imię i nazwisko, stopień/ tytuł promotora - podpis* | | | | ......................................................................................  *Imię i nazwisko kierownika katedry - podpis* | | | |
| .....................................................*Data wydania tematu pracy dyplomowej*  *- podpis promotora* | | ...................................................  *Regulaminowy termin złożenia pracy dyplomowej* | | | ........................................................  *Data złożenia pracy dyplomowej*  *- potwierdzenie dziekanatu* | | |
|  | .....................................  *Ocena promotora* | | | ..................................  *Podpis promotora* | | |  |
| ...................................................  *Imię i nazwisko, stopień/ tytuł recenzenta* | | ...........................................  *Ocena recenzenta* | | | | ................................................  *Podpis recenzenta* | |

Thesis topic:

Web application for supporting sport club activity.

SUMMARY

The aim of the project was creating a web application for supporting sport club activity. The motivation behind choosing the thesis was my membership in Parkour Białystok sport club to which the application is dedicated. This project allows to manage various articles, members, users, photo gallery and competitions organized by sport club.

This application was created using modern technologies, these are most important: Spring Framework and based-on child projects like Spring Boot, Thymeleaf, JQuery, Bootstrap.

This thesis consists of six chapters. The first chapter is an introduction. The second presents similar solutions. The next one presents and describes technologies which were used to create application. All diagrams and schemas created during planning process are described in chapter four. Chapter five is most important as it contains the technical and implementation details. These include a description of URL mapping, graphic interface, forms, server implementation, configuration and security aspect. The last chapter is a summary of work.

Załącznik nr 4 do „Zasad postępowania przy przygotowaniu i obronie pracy dyplomowej w PB”

Załącznik nr 4 do „Zasad postępowania przy przygotowaniu i obronie pracy dyplomowej w PB”

Maciej Ziniewicz Białystok, dnia………………………

imię i nazwisko studenta

90563

nr albumu

Informatyka stacjonarne

kierunek i forma studiów

dr inż. Jacek Grekow

promotor pracy dyplomowej

**OŚWIADCZENIE**

Przedkładając w roku akademickim 2015./2016. Promotorowi dr inż. Jackowi Grekowowi pracę dyplomową pt.: Aplikacja webowa wspomagająca działanie klubu sportowego dalej zwaną pracą dyplomową,

**oświadczam, że:**

1. praca dyplomowa stanowi wynik samodzielnej pracy twórczej,
2. wykorzystując w pracy dyplomowej materiały źródłowe, w tym w szczególności: monografie, artykuły naukowe, zestawienia zawierające wyniki badań (opublikowane, jak i nieopublikowane), materiały ze stron internetowych, w przypisach wskazywałem ich autora, tytuł, miejsce i rok publikacji oraz stronę, z której pochodzą powoływane fragmenty, ponadto w pracy dyplomowej zamieściłem bibliografię,
3. praca dyplomowa nie zawiera żadnych danych, informacji i materiałów, których publikacja nie jest prawnie dozwolona,
4. praca dyplomowa dotychczas nie stanowiła podstawy nadania tytułu zawodowego, stopnia naukowego, tytułu naukowego oraz uzyskania innych kwalifikacji,
5. treść pracy dyplomowej przekazanej do dziekanatu Wydziału Informatyki jest jednakowa w wersji drukowanej oraz w formie elektronicznej,
6. jestem świadomy/a, że naruszenie praw autorskich podlega odpowiedzialności na podstawie przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t. j.: Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.), jednocześnie na podstawie przepisów ustawy z dnia 27 lipca 2005 r. Prawo   
   o szkolnictwie wyższym (t. j.: Dz. U. z 2012 r. poz. 572 z późn. zm.) stanowi przesłankę wszczęcia postępowania dyscyplinarnego oraz stwierdzenia nieważności postępowania w sprawie nadania tytułu zawodowego,
7. udzielam Politechnice Białostockiej nieodpłatnie licencji na korzystanie z pracy dyplomowej w celu realizacji przeprowadzenia procedury antyplagiatowej przyjętej w Uczelni oraz na przekazanie pracy do Ogólnopolskiego Repozytorium Prac Dyplomowych, jak również udostępnianie i przechowywanie jej w Bibliotece Politechniki Białostockiej przez okres 50 lat od obrony pracy dyplomowej.

….……………………………………….

czytelny podpis studenta

Na podstawie art. 24 ust. 1 ustawy z dnia 29 sierpnia 1997 roku o ochronie danych osobowych (j.t. z 2014 r., poz. 1182 z późn. zm.) informuję, że administratorem danych jest Politechnika Białostocka, ul. Wiejska 45A, 15-351 Białystok. Dane będą przetwarzane w celach realizacji procedury antyplagiatowej przyjętej w Politechnice Białostockiej i nie będą udostępniane odbiorcom danych w rozumieniu art. 7 pkt 6 ustawy o ochronie danych osobowych. Osobie, której dane dotyczą, przysługuje prawo dostępu do treści swoich danych oraz ich poprawiania. Podanie danych jest obowiązkowe (art. 167b ustawy z dnia 27 lipca 2005 roku Prawo o szkolnictwie wyższym j.t. Dz.U. z 2012 r., poz. 572 z późn. zm.).

Białystok, dnia………………………

imię i nazwisko studenta

................................................

nr albumu

...............................................

kierunek i forma studiów

...............................................

promotor pracy dyplomowej

**OŚWIADCZENIE**

Przedkładając w roku akademickim 2015./2016. Promotorowi ………………………….……………………….. pracę dyplomową pt.: ...............…………………………………………………….……………………………………….. ………………….………………………..………………………………………..……………………………...………….…….…………………………………….........................…………………………………, dalej zwaną pracą dyplomową,

**oświadczam, że:**

1. praca dyplomowa stanowi wynik samodzielnej pracy twórczej,
2. wykorzystując w pracy dyplomowej materiały źródłowe, w tym w szczególności: monografie, artykuły naukowe, zestawienia zawierające wyniki badań (opublikowane, jak i nieopublikowane), materiały ze stron internetowych, w przypisach wskazywałem/am ich autora, tytuł, miejsce i rok publikacji oraz stronę, z której pochodzą powoływane fragmenty, ponadto w pracy dyplomowej zamieściłem/am bibliografię,
3. praca dyplomowa nie zawiera żadnych danych, informacji i materiałów, których publikacja nie jest prawnie dozwolona,
4. praca dyplomowa dotychczas nie stanowiła podstawy nadania tytułu zawodowego, stopnia naukowego, tytułu naukowego oraz uzyskania innych kwalifikacji,
5. treść pracy dyplomowej przekazanej do dziekanatu Wydziału Informatyki jest jednakowa w wersji drukowanej oraz w formie elektronicznej,
6. jestem świadomy/a, że naruszenie praw autorskich podlega odpowiedzialności na podstawie przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t. j.: Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.), jednocześnie na podstawie przepisów ustawy z dnia 27 lipca 2005 r. Prawo   
   o szkolnictwie wyższym (t. j.: Dz. U. z 2012 r. poz. 572 z późn. zm.) stanowi przesłankę wszczęcia postępowania dyscyplinarnego oraz stwierdzenia nieważności postępowania w sprawie nadania tytułu zawodowego,
7. udzielam Politechnice Białostockiej nieodpłatnie licencji na korzystanie z pracy dyplomowej w celu realizacji przeprowadzenia procedury antyplagiatowej przyjętej w Uczelni oraz na przekazanie pracy do Ogólnopolskiego Repozytorium Prac Dyplomowych, jak również udostępnianie i przechowywanie jej w Bibliotece Politechniki Białostockiej przez okres 50 lat od obrony pracy dyplomowej.

….……………………………………….

czytelny podpis studenta

Spis treści

[1. Wstęp 7](#_Toc440926052)

[2. Przegląd podobnych rozwiązań 8](#_Toc440926053)

[2.1 Klub Sportowy Movement 8](#_Toc440926054)

[2.2 Parkour United 10](#_Toc440926055)

[2.3 JOOMLA! 11](#_Toc440926056)

[3. Opis wykorzystanych technologii 13](#_Toc440926057)

[3.1 Platform-as-a-Service (OpenShift) 13](#_Toc440926058)

[3.2 MySQL Database 14](#_Toc440926059)

[3.3 Java Persistence API/Hibernate 15](#_Toc440926060)

[3.4 Spring Framework 16](#_Toc440926061)

[3.4.1 Spring Boot 18](#_Toc440926062)

[3.4.2 Spring Data 19](#_Toc440926063)

[3.4.3 Spring Security 21](#_Toc440926064)

[3.4.4 Spring MVC 24](#_Toc440926065)

[3.5 JavaScript 25](#_Toc440926066)

[3.5.1 JQuery 26](#_Toc440926067)

[3.6 Thymeleaf 26](#_Toc440926068)

[3.7 CSS 28](#_Toc440926069)

[3.8 Bootstrap 29](#_Toc440926070)

[4. Projekt Aplikacji 31](#_Toc440926071)

[4.1 Opis aplikacji 31](#_Toc440926072)

[4.2 Diagram przypadków użycia 32](#_Toc440926073)

[4.3 Schemat bazy danych 35](#_Toc440926074)

[4.4 Diagram pakietów 37](#_Toc440926075)

[5. Implementacja aplikacji 39](#_Toc440926076)

[5.1 Adresacja URL 39](#_Toc440926077)

[5.2 Interfejs Graficzny 41](#_Toc440926078)

[5.2.1 Widok użytkownika 42](#_Toc440926079)

[5.2.2 Widok administracyjny 48](#_Toc440926080)

[5.3 Formularze 49](#_Toc440926081)

[5.3.1 Tworzenie zawodów 50](#_Toc440926082)

[5.3.2 Rejestracja na zawody – walidacja 51](#_Toc440926083)

[5.4 Back-end 52](#_Toc440926084)

[5.5 Mechanizmy bezpieczeństwa 57](#_Toc440926085)

[6. Podsumowanie 59](#_Toc440926086)

[Literatura 60](#_Toc440926087)

# Wstęp

Obecnie świat zdominował Internet oraz technologie mobilne, niemal każdy z nas poszukując informacji zaczyna od przeglądania stron czy portali internetowych, używając do tego komputera lub smartphona. Z tego właśnie powodu każda instytucja, firma czy stowarzyszenie powinno posiadać aplikację internetową w formie witryny internetowej która jest jej wizytówką, pomaga przekazać zainteresowanemu odpowiednie, szukane przez niego informacje. Ponad to taka strona może pomóc w usprawnieniu pracy organizacji, ułatwić przepływ informacji, zdobyć nowych klientów.

**Celem pracy jest** stworzenie aplikacji webowej wspomagającej działanie klubu sportowego o nazwie Parkour Białystok. Cała aplikacja napisana została z pomocą frameworku Spring Boot, który dostarcza bogate API (ang. Application Programming Interface), ułatwiające proces tworzenia nowoczesnych aplikacji internetowych. Tworzona aplikacja powinna być estetyczna, prosta i dobrze prezentująca się, oraz przydatna dla zarządu w prowadzeniu stowarzyszenia. Strona ta ma za zadanie ułatwić chętnym dostęp do informacji dotyczących stowarzyszenia takich jak organizowane zawody, sposób i warunki dołączenia, oraz usprawnić zarządzanie.

Do realizacji aplikacji w ramach pracy inżynierskiej wykorzystano wiele technologii. Część serwerowa została napisana w języku Java z użyciem nowoczesnego frameworka Spring do którego należą Spring Data, Spring MVC, Spring Security, Spring Boot oraz Hibernate to połączenia z relacyjną baządanych MySQL . Część widoku generowana jest przez silnik Thymeleaf na podstawie kodu HTML5 i własnych dodatkowych atrybutów. Do stworzenia estetycznego wyglądu strony został wykorzystany framework Bootstrap dostarczający zdefiniowane kaskadowe arkusze stylów (CSS), natomiast dynamikę stronie nadaje kod JavaScript.

Temat ten zaproponowałem sam, ponieważ jestem członkiem stowarzyszenia Parkour Białystok i od dłuższego czasu nasza organizacja nie posiadała strony internetowej która mogłaby odpowiednio nas prezentować. Przy wyborze tematu kierowałem się również technologiami które chciałbym dodatkowo poznać, poza tokiem nauki akademickiej.

Praca podzielona jest na 6 rozdziałów. Drugi przedstawia przegląd rozwiązań o podobnej tematyce, trzeci opis technologii użytych do zaimplementowania aplikacji, czwarty projekty powstałe podczas pracy z aplikacją, piąty opis zaimplementowanych elementów, ostatni szósty to podsumowanie.

# Przegląd podobnych rozwiązań

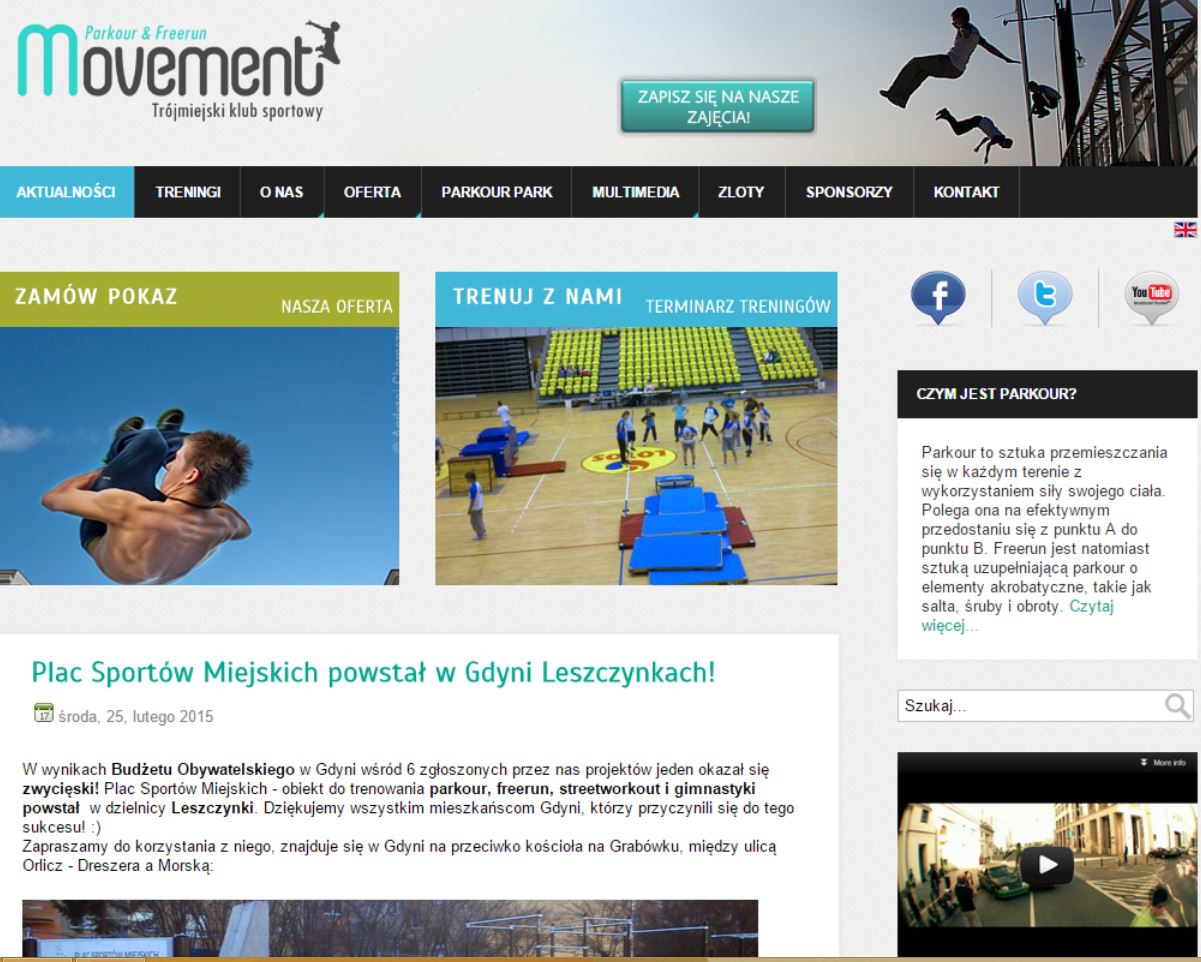
Na polskim rynku funkcjonuje wiele stowarzyszeń sportowych posiadających strony dostarczających podstawowe założenia pracy jednak realizowane jest to w postaci prostej witryny internetowej. Przeglądając istniejące rozwiązania o podobnej tematyce zdecydowałem się wziąć pod uwagę istniejące witryny klubów sportowych o tematyce Parkour, czyli Trójmiejski Klub Sportowy Movement oraz Warszawski Parkour United. Moim zdaniem witryny internetowe tych klubów są warte poświęcenia większej uwagi w pracy inżynierskiej. Co do modułu administracji stroną wzorowałem się na podstawie istniejących systemów CMS.

## Klub Sportowy Movement

Klub Sportowy [Movement](#_Literatura) [16] jest jednym z najlepiej rozwiniętych i prosperujących klubów sportowych Parkour w Polsce. Movement jest Trójmiejskim klubem sportowym, działa na terenie Gdyni, Gdańska, Spopotu oraz okolicznych miejscowości. Klub sportowy w formie stowarzyszenia kultury fizycznej został założony w 2011 roku, jednak sama grupa działa na terenie trójmiasta od 2005 roku. Przez swój czas działania Movement wybudowało dwa parkour parki na terenie trójmiasta oraz od 2008 regularnie organizują letnie zloty miłośników sportu parkour, które przyciągają coraz większe ilości ludzi zarówno z naszego kraju jak i z poza jego granic.

Witryna internetowa Klubu Sportowego [Movement](#_Literatura) (Rys 2.1) składa się z następujących podstron:

* „Aktualności” prezentują aktualne informacje o działaniu klubu, zmianach, osiągnieciach itd.,
* „Treningi” gdzie opisane są organizowane przez nich treningi, oraz opisane miejsca w których ćwiczą,
* Dział „O nas” składający się z dwóch stron jednej opisującej klub sportowy jego cele, oraz drugiej prezentującej członków klubu oraz zarząd,
* „Oferta” dział prezentujący doświadczenie w organizowaniu pokazów, filmów, zlotów oraz można w nim sprawdzić referencje klubu,
* „Parkour park” strona o specjalnych miejscach dedykowanych do ćwiczenia parkour,
* „Multimedia” pełni funkcję galerii jednak, w fakcie są to odnośniki do galerii klubu na stronie z portalu społecznościowego Facebook,
* „Zloty” opisująca aktualnie organizowany zlot oraz umożliwiająca zapisy za pośrednictwem formularzy Google – „Google Forms”,
* „Sponsorzy” i „Kontakt” jak nazwy wskazują kolejno strona prezentująca sponsorów i kolejna z danymi kontaktowymi, współpracująca z pluginem „Google Maps”.

Movement posiadają stronę umożliwiającą zapisy na zajęcia oraz zloty za pomocą formularzy „Google forms” oraz posiadają stronę kontakt zintegrowaną z pluginem „Google Maps” wskazującym lokalizację biura stowarzyszenia na mapie. Również można zobaczyć odnośniki w postaci estetyczny ikon do portalów takich jak Facebook, Twitter czy Youtube. Do tego posiadają wyszukiwarkę gdzie możemy wpisać słowo kluczowe które z ostanie wyszukane na stronach po czym otrzymamy listę odnośników do stron z słowem kluczowym. Witryna jest w dwóch wersjach językowych polskiej i angielskiej. Strona jest oparta o CMS (System zarządzania treścią) [„Joomla!”.](#_Literatura)

Rys. . Strona główna KS Movement.

Źródło: http://ksmovement.pl/PL/

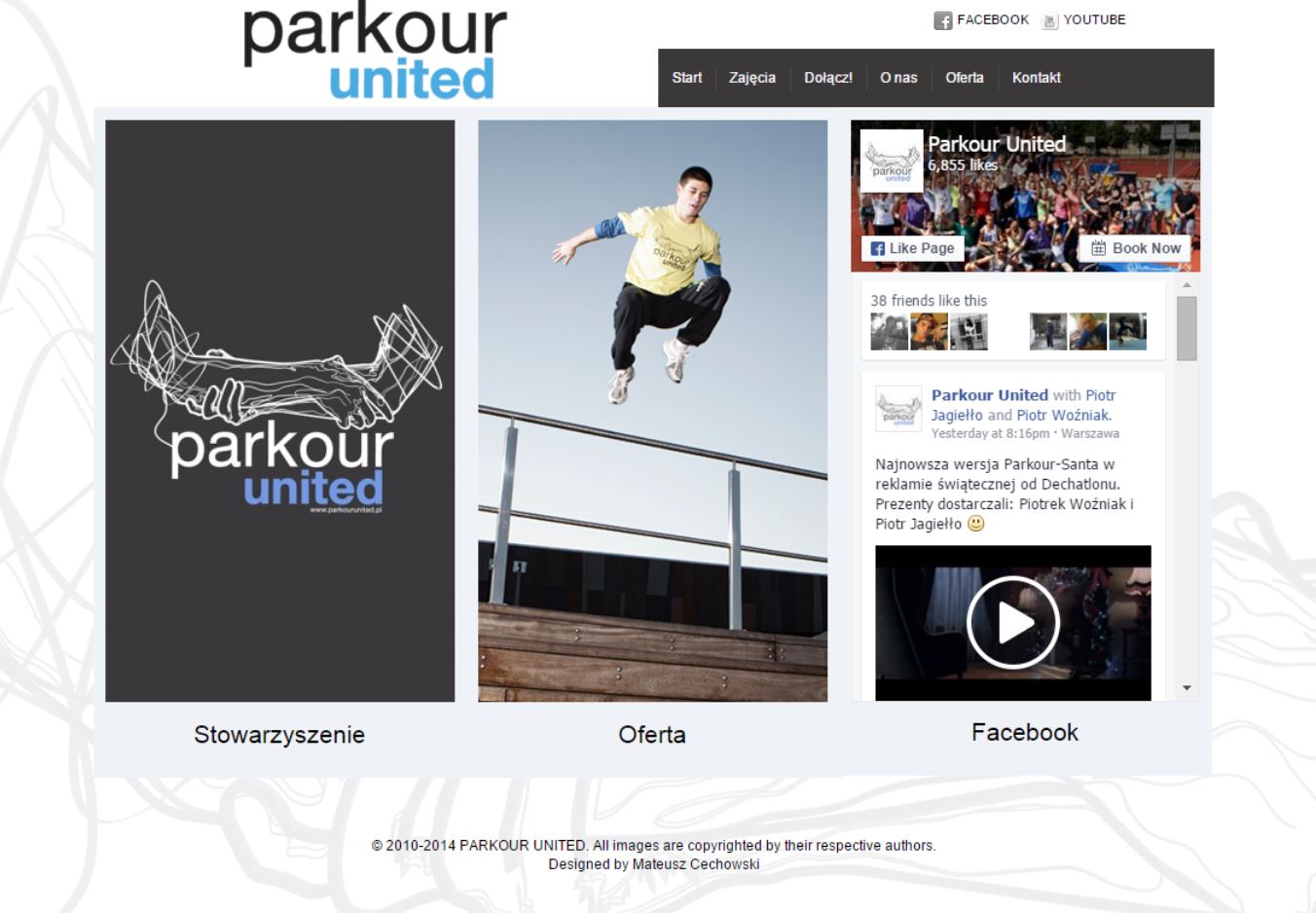
## Parkour United

Warszawski [Parkour United](#_Literatura) [15] jest kolejnym z przodujących klubów sportowych działających na terenie Polski. Grupa założycieli stowarzyszenia działa o rok dłużej od wyżej opisanego KS Movement, czyli od 2004 roku. Parkour United również zajmuje się organizowaniem pokazów, treningów dla dorosłych i dzieci, realizowanie projektów społecznych opartych o sport, kształceniem trenerów, budową miejsc dedykowanych do Parkour oraz organizowaniem zlotów.

[Witryna Parkour United](#_Literatura) (Rys 2.2) składa się następujących stron :

* Strona główna z odnośnikami do oferty i opisu stowarzyszenia oraz z załączonym pluginem prezentującym aktualności na stronie z portalu społecznościowego Facebook,
* „Zajęcia” strona opisująca organizowane zajęcia dla dzieci oraz dorosłych o różnym poziomie zaawansowania,
* „Dołącz” dostarcza informacje w jaki sposób można dołączyć do Parkour United oraz co z tym się wiąże.
* „O nas” prezentująca cele oraz osiągnięcia klubu,
* „Oferta” z informacją o usługach oferujących przez stowarzyszenie oraz z listą dotychczasowych klientów,
* „Kontakt” z danymi kontaktowymi oraz formularzem kontaktowym.

Parkour United posiada niezaprzeczalnie uboższą stronę w porównaniu do wyżej opisanego KS Movement. Analizując tę stronę można zauważyć że brak tu jakiegokolwiek CMS, jest to to czysty kod HTML plus JavaScript. Witryna posiada wtyczkę integrującą ze stroną na Facebook, oraz w stronie kontakt formularz kontatkowy wysyłający maila do zarządu.



Rys. . Strona główna Parkour United.

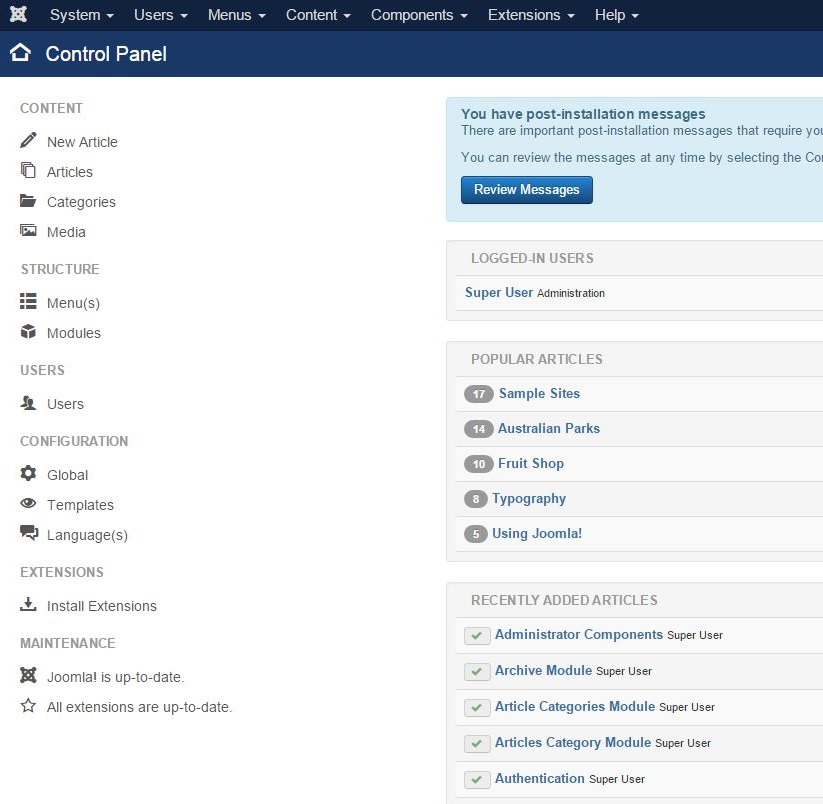
Źródło: http://parkourunited.pl/

## JOOMLA!

[„Joomla!”](#_Literatura) [14] jest to bardzo popularny CMS napisany w języku PHP, wykorzystujący bazę danych MySQL. „Joomla!” jest rozprowadzany na zasadach licencji Open Source, każdy może korzystać z tego CMS bez żadnych opłat.

Jak widać na zdjęciu poniżej (Rys 2.3) projekt Joomla! udostępnia szereg możliwości dla swoich użytkowników, jest to długo już rozbudowywany system zarządzania treścią umożliwiający dla użytkownika o praktycznie zerowej wiedzy na temat tworzenia aplikacji internetowych, zaprojektować swoją własną witrynę. Administrator ma możliwość:

* Dodawania nowych artykułów gdzie formularze obsługują formatowanie tekstu,
* Definiowania kategorii które określają stworzone artykuły,
* Zarządzanie użytkownikami i określanie im poziomów dostępu,
* Dodawanie komponentów takich jak bannery, formularze kontaktowe itd.,
* Dodawanie mediów tj. obrazki, filmy oraz zarządzanie nimi,
* Możliwość tworzenia i instalacji już stworzonych rozszerzeń, dzięki czemu system może być rozwijany również przez sympatyków z odpowiednimi umiejętnościami,



Rys. . Panel administracyjny Joomla!.

Źródło: Uruchomiony serwer Joomla! na własnym komputerze.

Po przeanalizowaniu systemu Joomla! spostrzegłem wiele ciekawych rozwiązań. Najbardziej zainteresowała mnie możliwość dodawania artykułów, mediów i zarządzania użytkownikami z tego względu że w swojej pracy również potrzebuję podobnej funkcjonalności. Część funkcjonalności zaimplementowanej w pracy inżynierskiej bazuje na funkcjonalności Joomla!:

* Dodawanie, edytowanie nowych artykułów,
* Treść tworzonych zawodów bazuje na HTML i użytkownik ma wpływ na to w jaki sposób jest wyświetlana na stronie,
* Zarządzanie użytkownikami,
* Dodawanie mediów tj. obrazków do artykułów i galerii zdjęć.

# Opis wykorzystanych technologii

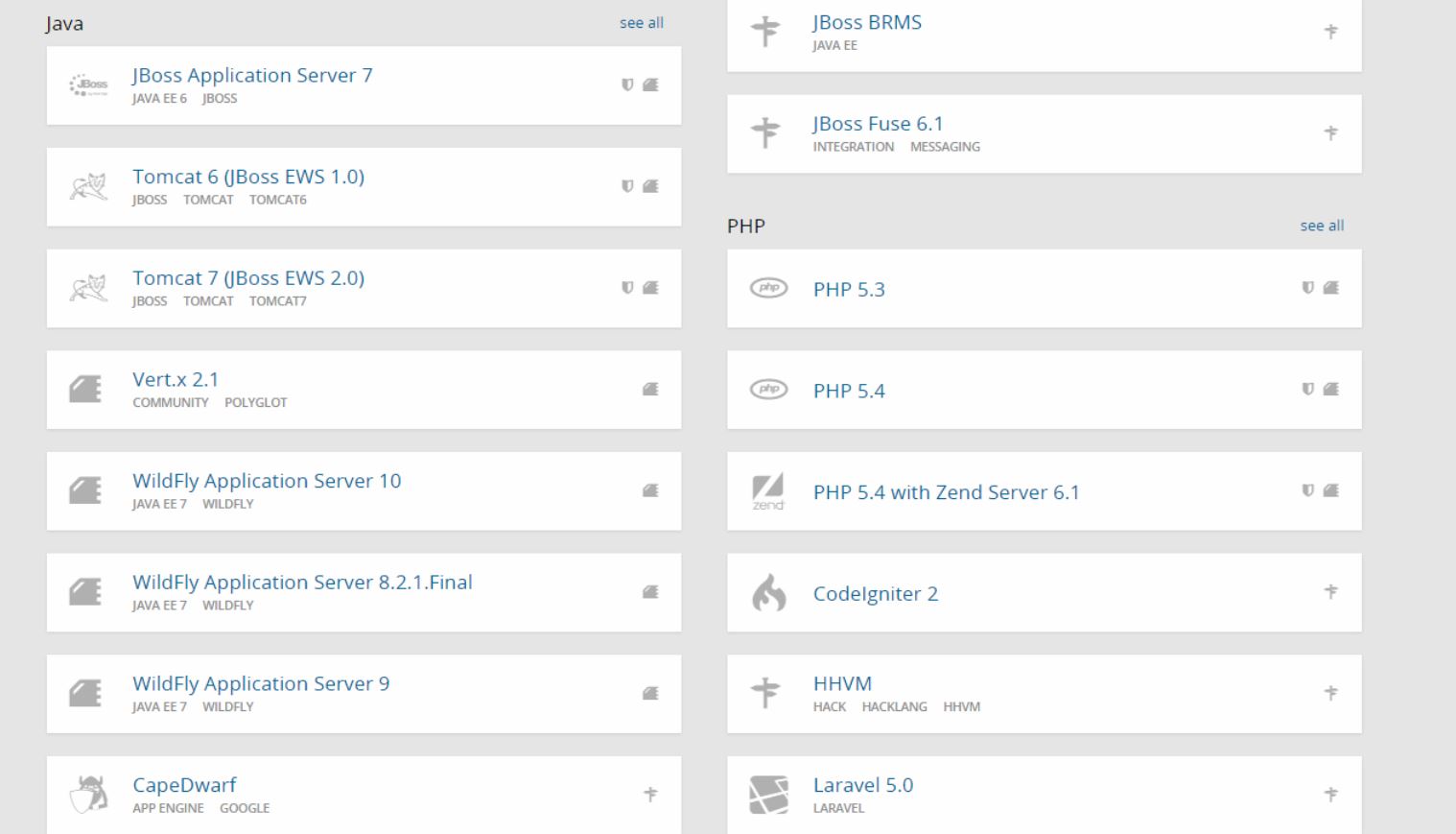
Przy wyborze technologii sugerowałem się własnymi zainteresowaniami, tym co chciałbym jeszcze poznać oraz narzuconymi przez zasady pisania pracy inżynierskiej warunkami. Wybrane technologie musiały być:

* nowoczesne i popularne,
* rozpowszechniane na zasadzie wolnego oprogramowania,
* posiadające dobre wsparcie społeczności użytkowników.

## Platform-as-a-Service (OpenShift)

[Platform-as-a-Service(PaaS)](#_Literatura) [13] jest to rodzaj chmury obliczeniowej, dostarczanej przez platformę, umożliwiającą rozwijanie, uruchamianie i zarządzanie aplikacjami webowymi bez złożoności budowania i utrzymywania infrastruktury która zazwyczaj idzie w parze przy rozwijaniu i uruchamianiu tego typu aplikacji. PaaS może być dostarczane na dwa sposoby pierwszy to jako publiczna chmura od dostawcy, gdzie klient posiada możliwość konfiguracji oprogramowania niezbędnego do uruchomienia aplikacji webowej na chmurze. Dostawca zapewnia również sieć, serwery, pamięć masową na dysku twardym oraz wszelkie niezbędne aspekty do hostwania aplikacji klienta. Drugim sposobem jest dostarczenie PaaS jako oprogramowania instalowanego na prywatnym bądź publicznym serwerze w postaci serwisu zarządzanego przez wewnętrzny zespół IT.

Na potrzeby tej pracy został wykorzystany produkt OpenShift firmy [RedHat](#_Literatura) [15], który jest platformą hostującą w postaci aplikacji. Umożliwia ona uruchomienie aplikacji webowych na serwerze za darmo. Produkt RedHat jest platformą rodzaju PaaS w postaci chmury obliczeniowej. Producent przedstawia OpenShift jako zaprojektowany by dostarczyć jedną rzecz dla developerów: Łatwość użycia bez zmartwień. Misją produktu jest ułatwić pracę developera poprzez zajęcie się całym zamieszaniem związanym z zawiłą konfiguracją usług, by developer mógł skupić się na swojej pracy.



Rys. . Widok wyboru serwerów usługi OpenShift

Źródło: Kreator aplikacji na www.openshift.com

Na rysunku (Rys. 3.1) przedstawiono część serwerów jakie obsługuje OpenShift, Rysunek (Rys. 3.1) obrazuje wszystkie serwery dotyczące języka Java z tego względu że w nim piszę pracę. Jednak produkt ten obsługuje aplikacje pisane w PHP, Java, Ruby, Ruby on Rails, Python jak również dostarcza CMS Wordperss i wiele innych. Dodatkowo po utworzeniu aplikacji możemy dołączyć do niej bazę danych MongoDB, MySQL lub PostgreSQL. Aby korzystać z usługi i zainstalować aplikację na serwerze należy pobrać narzędzia wiersza poleceń RHC(ang. RHC command tools), one umożliwiają połączenie poprzez klucz SSH z serwerem na którym znajdzie się aplikacja. Poprzez te narzędzia mamy możliwość wykonania wszystkich czynności z interfejsu graficznego, a nawet więcej. Należy pamiętać że łączymy się z serwerem który został uruchomiony na systemie Linux i należy używać składni tego systemu.

## MySQL Database

[MySQL [1]](#_Literatura) to wolnodostępny system zarządzania relacyjnymi bazami danych. Rozwijany początkowo przez firmę MySQL AB, która została kupiona przez Sun Microsystems, a tą z kolei kupił Oracle 27 styczna 2010 roku. MySQL to szybki i solidny, w tej chwili jeden z najpopularniejszych systemów zgodnych ze standardem SQL. Baza danych MySQL została wybrana ponieważ miałem już z nią styczność wiele razy oraz jest dostarczana przez opisany wyżej darmowy hosting OpenShift na którym znajduje się aplikacja. MySQL zawiera wsparcie dla replikacji bazy danych i wielojęzyczności tzn. każda tabela i kolumna może posiadać własne ustawienia kodowania znaków. Baza danych MySQL jest relacyjną bazą danych, w tego typu bazach dane są grupowane w relacje których reprezentacjami są tabele. Każda tabela składa się z wierszy i kolumn, kolumny stanowią atrybuty jakiegoś obiektu i najczęściej mają jeden określony typ, w bazach obsługujących język SQL kolumny posiadają swoje nazwy, natomiast wiersze reprezentują przetrzymywane dane i odpowiadają typem kolumnom.

## Java Persistence API/Hibernate

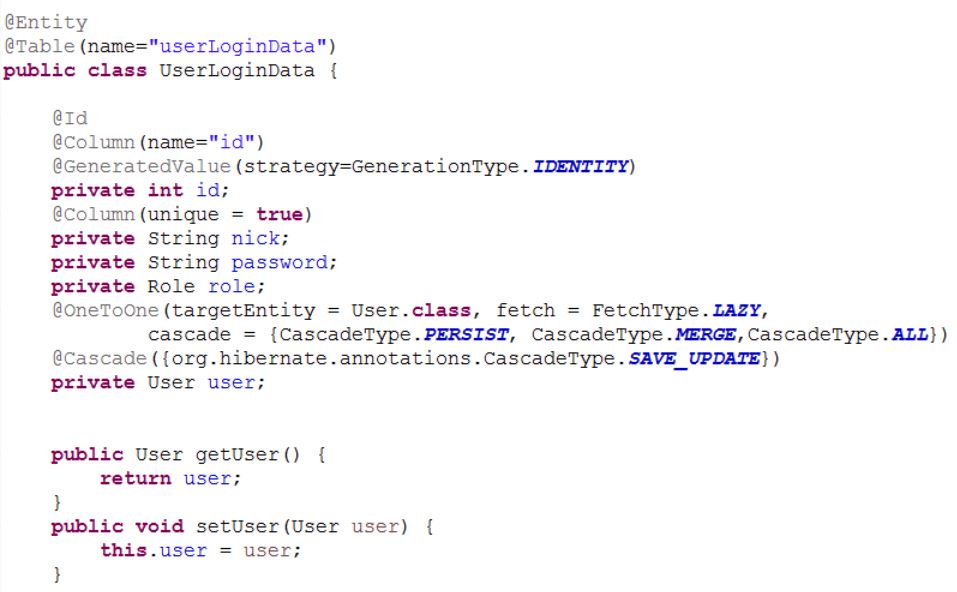
[Java Persistence API](#_Literatura) [3] w skrócie JPA jest to oficjalny standard mapowania obiektowo-relacyjnego(ORM) stworzony przez Sun Microsystems dla języka Java, standard ten definiuje zestaw klas i inferfejsów wykorzystywanych przy komunikacji z bazą danych. JPA daje możliwość programistom mapowania obiektów/relacji w prosty sposób aby zarządzać relacyjnymi danymi. Standard operuje na encjach czyli obiektach w języku Java z odpowiednio opisanymi polami reprezentującymi tabelę, a komunikuje się z bazą danych za pomocą interfejsu EntityManager. Sposób zapisu obiektów do tabel jest opisywany za pomocą adnotacji lub w pliku XML. Dodatkowo JPA dostarcza swój własny język zapytań JPA Query Language. Twórcy ORM mają zadanie dostosować swoje produkty do interfejsów JPA. Dzięki takiemu rozwiązaniu programiści nie muszą poznawać każdego frameworka od początku, oraz zmiana frameworka jest dużo łatwiejsza.

Hibernate jest jedym z wielu frameworków implementujący JPA służącym do komunikacji z bazą danych wykorzystującym ORM, jest on automatycznie dostarczany razem z frameworkiem Spring Boot który opiszę w dalszej części pracy, oraz jednym z najpopularniejszych mechanizmów ORM obecnie. W tej pracy został wykorzystany głownie jako implementacja API JPA.

Na rysunku poniżej (Rys. 3.2) zaprezentowano tabelę userLoginData z bazy danych jako obiekt encji w języku Java. W moim przypadku Hibernate tworzy automatycznie tabele na podstawie encji. Klasę jako encję oznaczamy poprzez adnotację „@Entity”, jeżeli nie określimy nazwy tabeli poprzez adnotację Hibernate nazwie ją nazwą klasy, jednak czasami nazwa klasy może być dłuższa niż dopuszczalna długość nazwy w bazie danych lub chcemy użyć innej nazwy wtedy należy użyć adnotacji „@Table” w której definiujemy ustawienia tabeli. Podobnie jest z polami, jeżeli nie określimy ich nazwy poprzez adnotację „@Column” określającą właściwości kolumny, a w niej pole „name” to framework automatycznie stworzy pola w tabelach o nazwach takich jak w encji. Można również określić strategię generowania id – adnotacja „@GeneratedValue”, oraz definiować połączenia pomiędzy tabelami, istnieją cztery:

* @OneToOne – jeden do jednego,
* @ManyToOne – wiele do jednego,
* @OneToMany – jeden do wielu,
* @ManyToMany – wiele do wielu.

Bardzo ważnym elementem jest jeszcze sposób pobierania wartości tabel połączonych z naszą tabelą, w tym przypadku tabelę user. „FetchType.Lazy” pobiera dane w momencie gdy są naprawdę potrzebne, za to „FetchType.Eager” pobiera natychmiastowo.



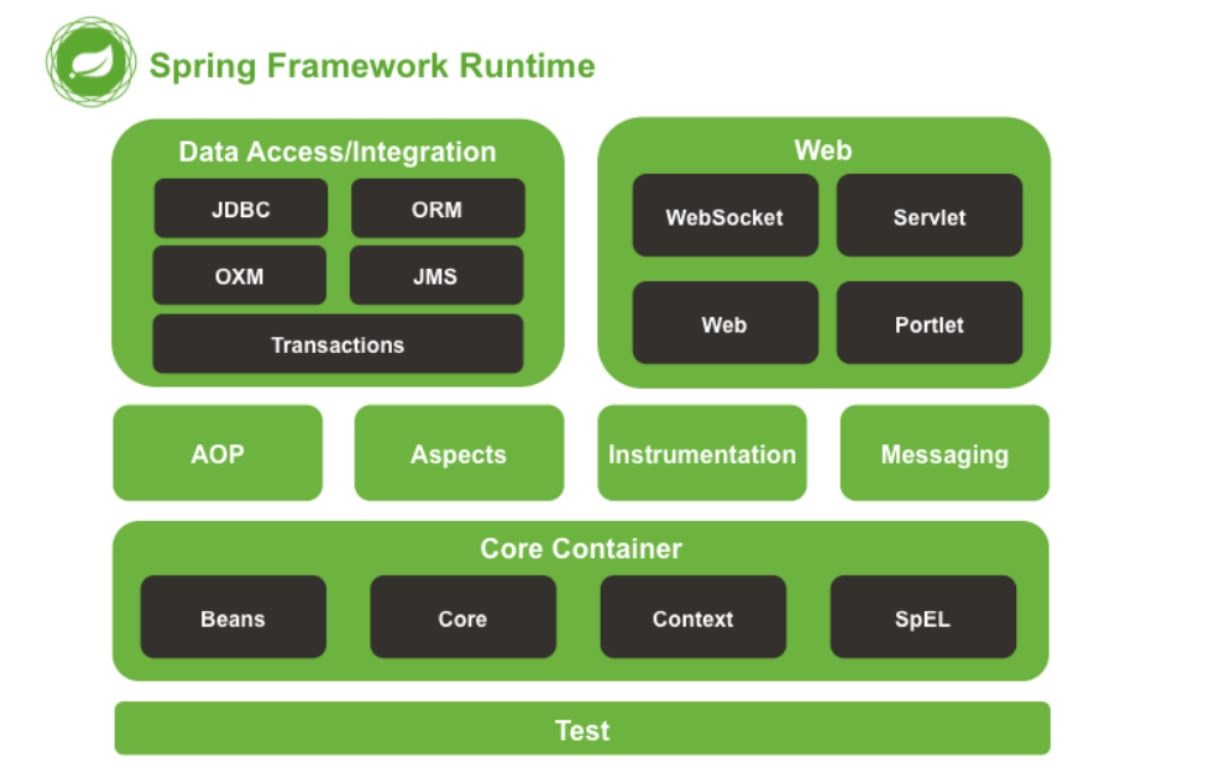
Rys. . Klasa UserLoginData oznaczona adnotacjami JPA i Hibernate

## Spring Framework

[String framewrok](#_Literatura) [5] jest to szkielet tworzenia aplikacji bazujący na platformie Java(istnieje również wersja dla środowiska .NET). Powstał jako alternatywa dla programowania z użyciem Enterpise JavaBeans(EJB), które narzucało wiele ograniczeń związanych z określonym modelem tworzenia oprogramowania, pracochłonnością związaną z tworzeniem aplikacji. Aktualnie tworzenie aplikacji w EJB zostało bardzo poprawione, a duża część z jego wcześniejszych wad została wyeliminowana. Spring posiada odmienna koncepcje tworzenia aplikacji, jego komponenty są rozłączne każdy może być używany oddzielnie, dzięki temu dostarcza lekki szkielet nie wymuszający pracy z określonym modelem programowania, kolejną zaletą Spring jest bardzo dobrze napisana dokumentacja. Framework konfiguruje się poprzez plik xml oraz adnotacje w klasach.

Spring Framewrok składa się z około 20 modułów (Rys. 3.3), pogrupowanych w :

* Core Container,
* Data Access/Integration,
* Web,
* AOP(Aspected oriented programming),
* Instrumentation,
* Messaging,
* Test.



Rys. . Diagram modułów i grup w Spring Framework

Źródło: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/overview.html

Projekt bazujący na Spring Framework składa się z komponentów które reprezentują klasy oznaczone odpowiednimi adnotacjami:

* „@Component” – określa dowolny komponent należący do szkieletu Spring,
* „@Repository” - komponent pasujący do roli repozytorium DAO(Data Access Model),
* „@Service” – komponent typu service w którym znajduje się logika biznesowa,
* „@Controller” – Komponent warstwy prezentacji w nim definiujemy co ma się znaleźć pod żądanym URL w przeglądarce.

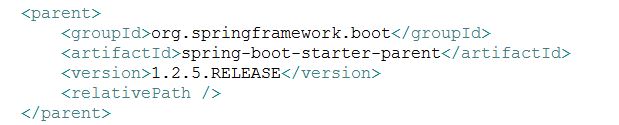
Gdy Spring podczas przeszukiwania paczek w poszukiwaniu komponentów natrafi na klasę oznaczoną którąś z tych adnotacji automatyczne ją zarejestruje i będzie mógł z niej korzystać.

### Spring Boot

[Spring Boot](#_Literatura) [4] jest to projekt bazujący na Spring Framework umożlwiający stworzenie aplikacji w oparciu o Spring za pomocą jednej klasy. Dostarcza on zestaw gotowych konfiguracji Spring dzięki czemu aplikacje tworzone o ten projekt wymagają niewielkich ilości konfiguracji, która w Spring potrafi być kłopotliwa zwłaszcza na początku przygody z tym frameworkiem. Założeniem Spring boot było między innymi:

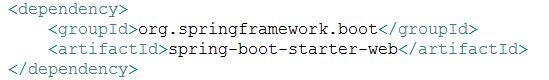
* Dostarczenie znacznie szybszego i szerzej dostępnego startu dla pisania aplikacji w Spring.
* Pozostawić bazową funkcjonalność
* Dostarczyć szeroki dostęp funkcjonalności niezbędnej do tworzenia aplikacji(np. wbudowane serwery, bezpieczeństwo, zewnętrzna konfiguracja)
* Żadnej generacji kodu oraz brak konieczności konfiguracji poprzez XML.

Do użycia Spring Boot potrzebna jest minimum Java 7 oraz Spring Framework w wersji 4.1.5 lub wyższej.Stworzenie aplikacji dzięki Spring Boot jest bardzo proste. Na początek w przypadku budowania projektu za pomocą Mavena należy dodać zależność „spring-boot-staret-web” (Rys. 3.5). Do tego musimy określić projekt z którego będziemy dziedziczyć tak jak na Rys 3.4.



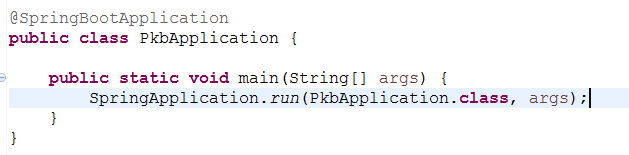
Rys. . Przykad tagów <parent> konfiguracji maven pod Spring Boot.

Kolejnym krokiem jest dodanie powyższej zależności, która powie Springowi że tworzymy aplikację webową. Nie musimy podawać wersji ponieważ automatycznie zostanie pobrana wersja zgodna z rodzicem.



Rys. . Zależność Spring Boot w maven

Następnie możemy zacząć pisać pierwszą klasę która uruchomi naszą aplikację (Rys. 3.6) w Spring Boot



Rys. . Klasa głowna uruchamiająca Spring Framework

Jest to zwykła klasa javy z metodą „public static void main(String[] args)”. Użycie adnotacji „@SpringBootApplication” gwarantuje nam użycie trzech adnotacji które zazwyczaj pojawiają się w główniej klasie każdego projektu czyli:

* „@Configuration” - adnotacja sygnalizująca Springowi że klasa oznaczona nią będzie zawierała konfiguracje projektu,
* „@EnableAutoConfiguration” - anotacja mówi Springowi by zgadnął jak użytkownik chce skonfigurować Springa, bazując na zależnościach jar które zostały dodane,
* „@ComponentScan” – adnotacja potrzebna by Spring wiedział że ma szukać komponentów należących do jego szkieletu, w nawiasach {} możemy zdefiniować które paczki ma przeszukiwać. Jeżeli nie zdefiniujemy ich Spring przeszuka wszystko od katalogu w którym znajduje się oznaczona klasa w dół.

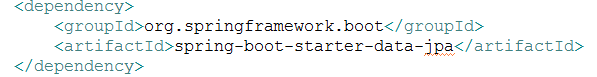
Wymagane jest użycie adnotacji „@EnableAutoConfiguration” by Spring mógł się skonfigurować, bez tego nasza aplikacja nie uruchomi się. Ostatnią częścią jest metoda main, która jest standardową metodą tak jak w konwencji Javy. Nasza metoda main wywołuje klasę SpringApplication poprzez metodę run, musimy podać nazwę klasy jako argument aby Spring wiedział jaki jest jego pierwszy komponent, oraz args by przekazać argumenty z linii komand jeżeli takowe były.

### Spring Data

[Spring Data](#_Literatura) [6] jest to kolejny projekt Spring który rozszerza możliwości Spring. Jego zadaniem jest dostarczenie znajomego i zgodnego modelu programowania bazującego na Spring dostępu do źródeł danych. Jest to dość duży projekt który składa się z wielu modułów, a każdy z nich wspiera inną technologię:

* Spring Data Commons
* Spring Data JPA
* Spring Data MongoDB
* Spring Data Redis
* Spring Data KeyValue
* Spring Data Slor
* Spring Data REST
* Spring Data Neo4j
* Spring Data Cassandra
* Spring Data Couchbase
* Spring Data Elasticsearch

Na potrzebu tego projektu wykorzystany został moduł Spring Data JPA. Aby użyć Spring Data w projekcie należy dodać zależność w pliku konfiguracyjnym Mavena pom.xml (Rys. 3.7).



Rys. . Zależność Spring Data w maven

Podstawowym elementem w pracy z Spring Data JPA są encje czyli reprezentacje tabeli w postaci obiektów w języku Java opisane w „Java Persistence API/Hibernate”.Kolejnym elementem bez którego nie będzie dało się obejść w Spring Data jest praca z repozytoriami, głównym celem Spring Data repository jest znaczne zmniejszenie kodu potrzebnego do zaimplementowania warstwy dostępu do danych. Podstawowym interfejsem jest Repository, który przechwytuje typ klasy którą ma zarządzać oraz jej typ id. W projekcie użyłem rozszerzenia interfejsu Repository – CrudRepository (Rys. 3.8) który jak sama nazwa mówi dostarcza funkcjonalność CRUD(Create,Read,Update,Delete). Użycie interfejsu CrudRepository pozwala zmniejszyć ilość kodu, repozytorium dostarcza takie metody jak:

* count() - zlicz ilość dostępnych encji w bazie,
* delete() - usuwa encje zależnie od podanego argumentu metody, można usunąć podając ID, całą encję lub tablicę encji,
* deleteAll() - usuwa wszystkie encje w bazie danych,
* exitis(ID id) - sprawca czy encja o podanym Id istnieje,
* findAll() - wyszukuje wszystkie encje,
* findOne(ID id)- wyszukuje encje po id,
* save() – zapisuje encje, można zapisać pojedynczą encje jak i całą tablicę encji. Zależnie wiersz istnieje już w bazie czy nie jest wykonywane UPDATE lub CREATE.

Użytkownik ma również możliwość tworzenia własnych niestandardowych w tym celu należy stworzyć własny interfejs rozszerzający główny Repository lub któryś z jego podinterfejsów. Następnie wystarczy że napiszemy samą definicję metody. Przykładowo jeżeli dostarczona do repozytorium encja ma pole „name” i chcemy stworzyć metodę liczącą ilość encji po polu „name” jedyne co musimy teraz zrobić to napisać deklarację metody np. „Long countByName(String name)” resztę robi za nas Spring Data.



Rys. . Przykład klasy repozytorium

Programista nie musi tworzyć zapytania SQL ponieważ jest ono budowane podczas uruchamiania aplikacji na podstawie podanej klasy domenowej, oraz definicji metody czyli jej nazwy i podanych argumentów. Musimy pamiętać jedynie że nazwa metody musi być stworzona według odpowiedniej konwencji. Jeżeli nie możemy osiągnąć zamierzonego efektu w ten sposób możemy sami zdefiniować zapytanie SQL poprzez adnotacje „@Query”.

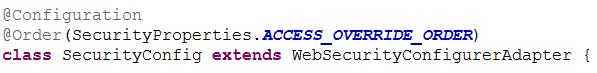
### Spring Security

[Spring Security](#_Literatura) [8] jest kolejnym projektem z rodziny Spring, daje możliwość autentykacji i kontroli dostępu przy czym jest bardzo łatwo go dostosować do własnych potrzeb tak jak każdy z frameworków Spring. Szczególny nacisk przy budowaniu tego narzędzia położono na wspomaganie Spring Framework, choć może być używany niezależnie od niego. Dwoma głównymi polami którymi zajmuje się Spring Security to autentykacja i autoryzacja.

* Autentykacja jest to proces określania/ustalania czy użytkownik jest tym za kogo się podaje,
* Autoryzacja oznacza kontrolę dostępu czyli decydowania o tym czy osoba używająca aplikacji ma dostęp do akcji którą chce wykonać, w momencie gdy dochodzi do potrzeby autoryzacji akcji tożsamość użytkownika została już określona przez proces autentykacji.

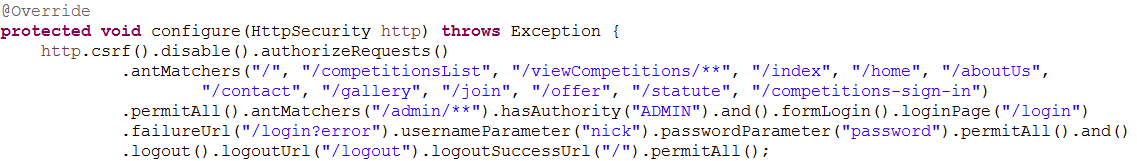
Spring Security wspiera proces autentykacji w wielu technologiach takich jak HTTP czy LDAP, proces autoryzacji opiera się o filtry, zapytanie przechodzi przez szereg filtrów, a każdy z nich decyduje czy odmówić dostępu. Spring Security umożliwia bardzo sprawne i łatwe zarządzanie bezpieczeństwem aplikacji. Programista ma możliwość zdefiniowania reguł dla całych URL, ich części czy wzorców, oraz metod, a nawet usunąć z przesyłanej kolekcji elementy do których użytkownik nie powinien mieć dostępu, i to nadal nie wszystkie możliwości tego frameworka.

Spring Security również możemy konfigurować za pomocą klas (Rys. 3.9), wystarczy że jest oznaczona jako „@Configuration” oraz rozszerza klasę „WebSecurityConfigurerAdapeter”, następnie wystarczy nadpisać wybrane metody. Jednak jest to jeden z wielu sposobów konfiguracji Spring Security za pomocą klas i dotyczy tylko aspektu webowego.



Rys. . Przykład oznaczenia Spring Security

Klasa „WebSecurityConfigurerAdapter” dostarcza nam wiele możliwości konfiguracji, choć dotyczy ona głownie autentykacji użytkowników. Możemy ustalić w jaki sposób ma być weryfikowany użytkownik, oraz ustawić dostęp do określonych URL, lub wzorców URL dla wszystkich bądź dla poszczególnych grup.



Rys. . Przykład definiowania prawd dostępu

Rysunek Rys 3.10 prezentuje sposób w jaki jest definiowany dostęp do strony, działa to na zasadzie wzorca łańcuch zobowiązań(ang. Chain of Responsability) dlatego ważna jest kolejność definiowanych metod. Dla obiektu HttpRequest podajemy kolejno metody dzięki którym definiujemy wybrane przez nas aspekty która mają być chronione:

* „.csrf()” jest to token który wysyła się z każdym wysłaniem forma od użytkownika do serwera, na podstawie tego tokenu można określić czy form jest prawidłowy oraz czy nie został zmodyfikowany,
* „authorizeRequests()” mówi że będziemy chcieli konfigurować autoryzacje próśb o dane wysyłanych na serwer,
* W metodzie „antMatches()” podajemy URL który został zmapowany w kontrolerze bądź wzorzec,
* „permitAll()” daje dostęp wszystkim do wcześniej określonych zasobów,
* „hasAuthority” daje dostęp tylko dla określonej roli (Rys. 3.11).



Rys. . Przykład określenia roli dla mapowania

Powyższy zapis umożliwi dostęp do wszystkich URL rozpoczynających się na „/admin” tylko dla roli Admin.

* „.and()” podobnie jak w zapytaniu SQL dodaje możliwość dodania kolejnych warunków,
* „.formLogin()” tu definiujemy pod jakim URL znajduje się nasz formularz logowania poprzez podanie go w metodzie „.loginPage()”,
* „.falitureLogin” definiuje gdzie zostaniemy przekierowani po nieprawidłowej próbie logowania,
* „.usernameParameret()” i „.passwordParameter()” definiują parametry http które odpowiadją za nazwę użytkownika i hasło,
* „.logout()” z pomoca tej metody definuje sposób wylogowania się, ustawiam w niej URL który wywoła wylogowanie metodą „.logoutURL()”, oraz przekierowanie po udanym wylogowaniu „.logoutSuccessURL()”.

Poza konfiguracją za pomocą rozszerzeni klasy „WebSecurityConfigurer” możliwa jest konfiguracja za pomocą adnotacji. Adnotacje są wykorzystywane do określania ochrony dostępu do metod. Adnotacje te przyjmują jako argumenty wyrażenia napisane w SpEL(Spring Expression Language):

* „@PreAuthorize” – autoryzacja przed wykonaniem metody, wewnątrz wyrażeń SpEL mamy dostęp do podanych argumentów,
* „@PostAuthorize” - autoryzacja po wykonaniu metody, wewnątrz wyrażeń SpEL mamy dostęp do argumentów metody oraz do zwracanej wartości,
* „@PreFilter” – pozwala filtrować kolekcje przesyłane do metody jako argumenty, argument do którego jest przypisywana adnotacja musi należeć do drzewa java.util.Collection.
* „@PostFilter”- pozwala fitrować kolekcje zwracane przez metodę, wartość do któregj jest przypisywana adnotacja musi należeć do drzewa java.util.Collection.

### Spring MVC

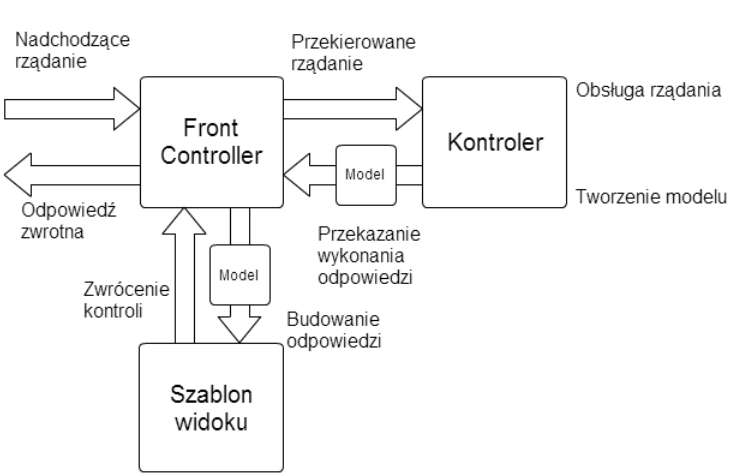
Wzorzec [MVC](#_Literatura) [7] to wzorzec projektowy stosowany w informatyce, zakłada on podzielenie struktury aplikacji na 3 warstwy - model, widok i kontroler od których powstała nazwa wzorca (MVC -Model,View,Controller). Wzorzec ten szczególnie popularny jest przy tworzeniu aplikacji webowych.

* Model – jest to warstwa reprezentująca dane, jako jedyna przechowuje je w sposób trwały. Model ukrywa sposób implementacji dla reszty aplikacji, dostarcza jedynie interfejs którym ona może się posłużyć by odpowiednie dane uzyskać.
* Widok – reprezentuje to co widzi użytkownik. W przypadku aplikacji webowych najczęściej jest to strona HTML, prezentuje ona uzyskane dane dla użytkownika.
* Kontroler – Serce wzorca MVC on odbiera zapytania z widoku decyduje czy użytkownika ma dostęp do akcji, komunikuje widok z modelem, oraz może wprowadzać w nich zmiany, oraz decyduje co zostanie przesłane z modelu do widoku i z widoku do modelu.

Obieg danych w MVC przebiega następująco (Rys. 3.11):

1. Użytkownik wykonuje akcje przez co wysyła rządanie do kontrolera,
2. Kontroler obłsuguje je, komunikując się z modelem i wprowadzając w nim zmiany.
3. Następnie widok generuje na podstawie modelu odpowiednią interfejs użytkownika(np. listę przedmiotów w koszyku), widok pobiera dane z modelu. Model nie wie o widoku.

Spring wspiera wzorzec MVC w swoim frameworku Spring Web MVC. Został zaprojektowany bazując na „DispatcherSerlvet” który wysyła zapytania do odpowiednich kontrolerów które je obsługują.



Rys. . Diagram komunikacji w Spring MVC

Spring MVC dzieli aplikacje na 4 warstwy:

* User Interface Layer – prezentuje ona aplikację dla użytkownika, buduje ona odpowiedź wygenerowaną przez serwer w typ żądany przez użytkownika. Warstwa interfejsu użytkownika została wydzielona ponieważ istnieje wiele różnych technologii widoku, a twórcy Springa chcieli dać możliwość użytkowania tak wielu jak to możliwe,
* Web Layer – ta warstwa ma dwa obowiązki. Pierwszy to pokierować użytkownika przez aplikację webową przykładowi poprzez mapowanie URL do widoków,a drugi to warstwa integracyjna pomiędzy warstwami serwisu i http, która powinna być tak cieńka jak to możliwe. Powinna być warstwą która skonwertuje rządanie http na coś co może być obsłużone przez warstwę serwisu,
* Service Layer - warstwa serwisu która odpowiada za funkcjonalność, przypadki użytkowania systemu przez użytkownika,
* Data Access Layer – warstwa odpowiedzialna za interfejs reprezentujący bazę danych. Ta warstwa posiada wiedzę jak korzystać z obiektów z bazy danych.

## JavaScript

[JavaScript](#_Literatura) [9] jest skryptowym językiem programowania, praktycznie cała interaktywność stron, ich dynamiczność, efekty wizualne, animacje to zasługa tego języka. Wszystkie strony internetowe poza kodem HTML i Kaskadowymi arkuszami stylów składa się z kodu w języku JavaScript, pozwala on pisać zazwyczaj proste choć i zaawansowane programy które są wykonywanie po stronie użytkownika przez jego przeglądarkę, a całość potrzebnych zasobów do obsługi programu jest przenoszona na komputer użytkownika, dzięki czemu możemy często odciążyć serwer z prostych akcji które mogą zostać przeprowadzone po stronie użytkownika, a na serwer wysłać jedynie gotowe dane. JavaScript może być wykorzystywany również po stronie serwera przykładem takiego wykorzystania jest biblioteka NodeJS. JavaScript jest językiem skryptowym oznacza to że jest kompilowany ( kompilacja to proces generowania pliku z kodu programisty na zrozumiały dla komputera) przez interpreter, który jest wbudowany w przeglądarkę. Wadą tego rozwiązania jest to że języki skryptowe muszą być kompilowane przy każdym uruchomieniu, przez to działają wolniej. Kolejną wadą tego języka jest jednowątkowość, może wykonywać tylko jedną czynność w tym samym momencie, i blokuje ona wykonywanie kolejnych dopóki sama się nie zakończy. Mimo wszystko JavaScript zdominował Internet, jest używany na każdej stronie i ciężko jest sobie wyobrazić aplikacje internetowe bez tego języka.

### JQuery

[JQuery](#_Literatura) [9] jest biblioteką napisana w JavaScripcie dostarczającą szereg funkcji których napisanie w JS zajęło by wiele czasu. Mimo że pisanie w JavaScripcie jest łatwiejsze od wielu innych języków to w dalszym ciągu język ten może być trudnym dla nowych użytkowników, dodatkowym problemem jest to że JavaScript jest interpretowany inaczej na różnych przeglądarkach z tego powodu kod napisany w przykładowo Firefoxie może działać a w Chrome już nie, jednak najczęściej działa na tych dwóch a nie na Internet Explorer. Tu jest miejsce dla JQuery ponieważ dostarcza gotowe API, sprawdzone na przeglądarkach ułatwia programistom życie, ponieważ nie muszą pisać własnych funkcji i testować ich godzinami na różnych przeglądarkach tylko wykorzystają gotowe, jednak z doświadczenia wiem że i to nie zawsze działa. Framework ten jest używany w wielu popularnych witrynach, systemach zarządzania treścią takich jak Wordpress.

## Thymeleaf

[Thymeleaf](#_Literatura) [2] jest biblioteką Javy, rozpowszechnianą na darmowej licencji Apache License 2.0, która służy jako silnik szablonów(ang. template engine). Umożliwia procesowanie i generowanie plików HTML/HTML5, XML, CSS, JavaScript oraz tekstowych oraz może pracować zarówno w środowiskach webowych jak i desktopowych. Jest lepiej dostosowany do dostarczania warstwy widoku w środowiskach webowych, jednak ma możliwość procesowania wielu formatów również w środowiskach offline. Dostarcza moduł integracji z Spring MVC dzięki czemu może zostać użyty zamiast technologii JSP, z tego powodu zdecydowałem się na użycie Thymeleaf ponieważ wyżej opisany Spring Boot dostarczył automatycznie skonfigurowany moduł. Głowną funkcją silnika jest procesowanie struktury DOM(Document Object Model) typowej dla html. Wnikając w szczegóły Thymeleaf używa własnej implementacji DOM, do budowania drzewa reprezentującego szablon, na której później operuje przeszukując jej gałęzie i wykonuje odpowiednie procesy które modyfikują strukturę DOM zgodnie z dostarczoną konfiguracją oraz umieszcza dane przesłane do szablonu.

Głownym celem Thymeleaf jest dostarczenie eleganckiego i przejrzystego sposobu na tworzenie szablonów. Jego standardowe dialekty i standardowe dialekty Spring pozwalają na tworzenie dobrych naturalnych szablonów, poprawnie obsługiwanych przez przeglądarki. Każdy użytkownik ma możliwość rozszerzenia możliwości tej biblioteki, o własne pomysły.

W tej pracy Thymeleaf został użyty do tworzenia szablonów dokumentów HTML5. Tworzenie takiego szablonu w oparciu o opisywany silnik jest bardzo jasne i proste, opanowanie składni zajęło mi bardzo mało czasu. Aby stworzyć szablon, wystarczy utworzyć plik o rozszerzeniu html, i zbudować odpowiednią strukturę DOM, składnią identyczną jak ta HTML5 jednak wzbogaconą o nowe możliwości. Poniżej przykładowy kod wyświetlający artykuły na stronie główniej.



Rys. . Przykład kodu w szablonie Thymeleaf

Jak widać na Rys. 3.13 elementy są identyczne jak w HTML5, jednak pojawiają się tam nowe atrybuty o nazwach charakterystycznych dla języka HTML5 jednak z przedrostkiem „th:” określa on atrybuty silnika Thymelaf. Przykładowo jeżeli chcemy się odwołać do obiektu przekazanego do szablonu musimy to zrobić w odpowiedni sposób przez umieszczenie jego nazwy w specjalnym wyrażeniu ${article}", w tym przypadku silnik będzie wiedział że odwołujemy się do obiektu o nazwie article. Za to poprzez wyrażenie @{text} definiuje się odnośniki. „th:each” działa jak zwyczajna pętla for each w języku Java, tak samo jak „th:if” działa jak instrukcja sterująca if. Jeżeli chcemy wyświetlić tekst przekazany do szablony musimy się odwołać do obiektu który go reprezentuje i umieścić w atrybucie „th:text”, a nie jak w HTML wpisać pomiędzy tagami <div></div> ponieważ Thymeleaf zinterpretuje wyrażenie pomiędzy tagami jak zwykły tekst.

## CSS

[CSS](#_Literatura) [12] czyli kaskadowe arkusze stylów jest język którym możemy opisać sposób w jaki ma się prezentować nasza strona. Polega to na tworzeniu reguł które określają wygląd elementów naszej strony, reguły te możemy definiować bezpośrednio dla elementów bądź dla klas które są tym elementom przypisane, elementy oznaczone klasami dziedziczą jej reguły. Można nadpisywać poprzednie reguły, opisując nowe i przypisując je. Reguła w CSS składa się z dwóch elementów:

* Selektorów czyli opisu jakiego elementu ma dotyczyć reguła, pojedyncza reguła może dotyczyć więcej niż jednego elementu a ich nazwy należy oddzielić przecinkami. Istnieje wiele selektorów od prostych takich jak elementy do takich wyszukujących elementy zawarte w innych elementach tzw. dzieci. Selektory nie ograniczają się jedynie do elementów, można wyszukiwać po ich atrybutach np. najpopularniejsze po id czy klasie. Ale również istnieją selektory pseudo elementów jak pierwsza linia akapitu czy pseudo klasy jak pierwsze dziecko, ostanie dziecko, element aktywny schowany itd.
* Deklaracji które wskazują sposób w jaki mają wyświetlać się nasze elementy na stronie. Deklaracja zawsze znajduje się pod selektorem i dotyczy ona elementu wybranego przez selektor. Określić możemy praktycznie wszystko, choć każdy element ma inne atrybuty co jest logiczne. Możemy określić takie proste rzeczy jak czcionka, rozmiar, kolor ale również i takie jak sposób w jaki się ma zachować tekst gdy przekroczy ramy pojemnika do którego należy.

Z kolei każda deklaracja składa również z dwóch części:

* Właściwości wskazujące na parametr wyglądy przykładowo może być to kolor, wysokość czy ustawienie względem strony.
* Wartości które wyrażają konkretnie jaka wartość ma przyjąć dana właściwość.

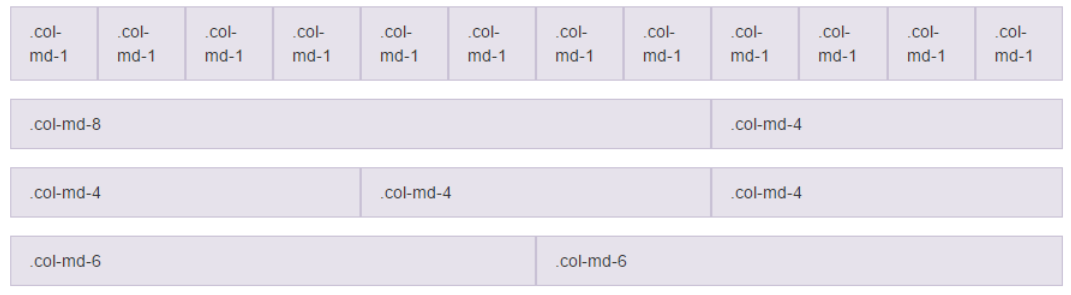
Dzięki CSS jesteśmy w stanie tworzyć bardzo estetyczne strony. Kaskadowe arkusze stylów istnieją już trochę czasu dlatego doczekały się już kolejnych wersji, początkowo była to jedna wielka specyfikacja definiująca różne właściwości, ale z wypuszczeniem CSS3 to się zmieniło i kaskadowe arkusze stylów podzielone zostały na niezależne dokumenty zwane modułami, gdzie każdy moduł odpowiadał za coś innego a w 2012 CSS Working Group miało opublikowanych już 50 modułów. Zachowana jest kompatybilność wsteczna także rozwiązania z wcześniejszych wersji CSS nadal działają i są używane. Obecnie niektóre moduły wchodzą na poziom czwarty.

## Bootstrap

Front-end czyli część aplikacji webowej odpowiedzialnej za interakcje z użytkownikiem, pobieraniu danych, dostarczaniu danych itd. Tej części wytwarzania aplikacji internetowych dotyczą takie technologie jak HTML5, CSS3, JavaScript oraz szereg frameworków, silników szablonów bazujących na tych technologiach. Na pomoc programistom Front-end przychodzi [Bootstrap](#_Literatura) [10,11] czyli framework dostarczający gotowe kaskadowe arkusze stylów (CSS), oraz JavaScripty używające biblioteki JQuery których użycie znacząco przyspiesza budowanie estetycznych interaktywnych resposnywnych stron internetowych. Budowanie tego od podstaw zajmuje masę czasu, a Bootstrap udostępnia nam to za darmo. Jedną z głównych atutów Bootstrap jest system siatki ( Grid system ) używany do tworzenia szablonów stron jako serii wierszy i kolumn które trzymają zawartość. Sposób działania tego systemu opiera się na tym że cała szerokość ekranu jest dzielona na 12 kolumn i zależnie od określonych klas elementy sa umieszczane względem tych kolumn, szerokość kolumn jest zależna od rozdzielczości ekranu dzięki czemu Boostrap jest responsywny oznacza to że elementy dopasowują się dynamicznie do szerokości ekranu, wygląda to następująco:

1. Wiersze muszą być umieszczone w divach o klasach „.container” lub „.container-fluid”
2. Do tworzenia wierszy używamy klasy „.row” a do kolumn „.col-\*-\*” z tym że pierwsza \* oznacza ty kolumny który może być:
   1. xs – Extra small przeznaczony dla małych urządzeń o rozdzielczości mniejszej 786px
   2. sm – Small dla urządzeń o rozdzielczości większej lub równiej 786px
   3. md – Medium dla urządzeń średnich o rozdzielczościach większych lub równych niż 992px
   4. lg – Lagre dla urządzeń dużych o rozdzielczościach większych lub równych 1200 px

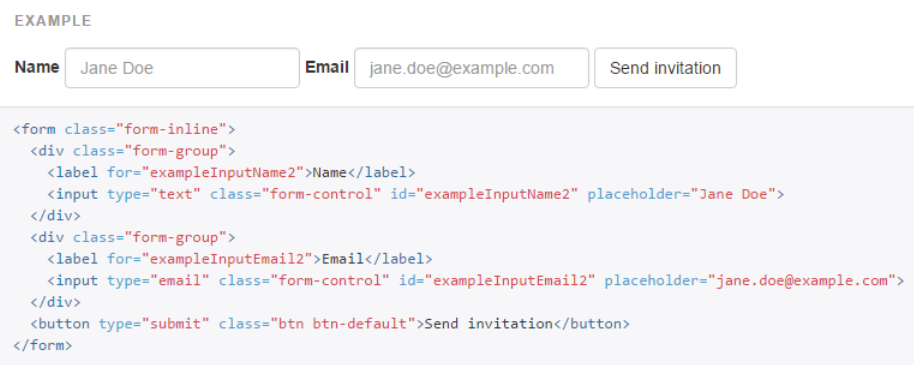
kolejna \* oznacza ilość kolumn jaką ma zajmować oznaczany element, tak jak wcześniej napisałem ilość kolumn wynosi 12, które są ułożone względem jednego wiersza.

1. Kolumny automatycznie tworzą odstępny pomiędzy sobą za pośrednictwem „padding” czyli przesunięcia względem granic wiersza/kolumny.
2. System siatki (Rys. 3.15) ma wpływ na urządzenia, kolejno o większych rozdzielczościach czyli jeżeli użyjemy klasy „.col-md-4” będzie ona miała wpływ również na rozdzielczości przypisane do kolumn typu large jeżeli nie znajdzie się klasa „.col-lg-4” która opisze większą rozdzielczość, nie ma jednak wpływu w dół na urządzenia o niższych rozdzielczościach.

Rys. . Przykład kolumn systemu siatki.

Źródło: http://getbootstrap.com/

Bootstrap dostarcza wiele przydatnych gotowych elementów (Rys. 3.16) takich jak nagłówki, panele, przyciski, tabele, listy, formularze itd. Działają one na tej samej zasadzie co w HTML jednak jest zmieniony ich wygląd na bardzo estetyczny. Użycie jest bardzo proste wystarczy nadać odpowiednią klasę dla elementu żeby wyglądał tak jak Boostrap go dostarczył.



Rys. . Przykład wyglądu elementów i kodu html

Źródło: http://getbootstrap.com/

Framework dostarcza również rozszerzenia m.in. slidery, karuzele, dymki w postaci JavaScriptów, można ich użyć na dwa sposoby poprzez użycie gotowego już kodu HTML, lub za pomocą skryptów JQuery które należy dołączyć do stro

# Projekt Aplikacji

W tym rozdziale zostaną przedstawione wyniki prac projektowych aplikacji.

## Opis aplikacji

Zadaniem tworzonej w ramach pracy inżynierskiej aplikacji wspomaganie działania klubu sportowego. Powinna służyć jako narzędzie dla zarządu klubu sportowego umożliwiające sprawne uzyskiwanie niezbędnych dla niego informacji oraz przekazywanie treści dla osób zewnętrznych. Aplikacja powinna dostarczać administratorom następujące funkcjonalności dostępne po zalogowaniu:

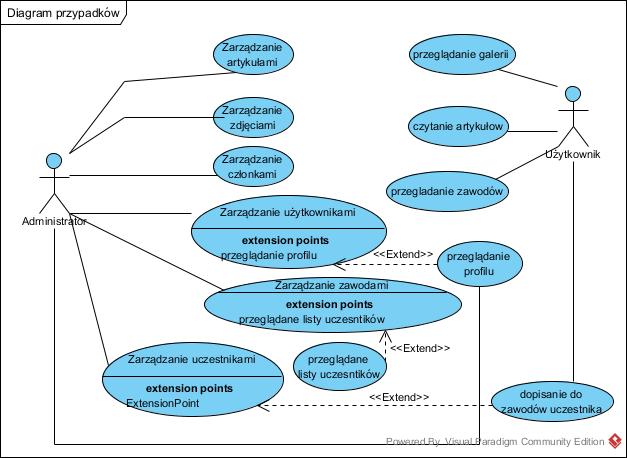
* Zarządzanie użytkownikami – dodawanie nowych, edycja obecnych oraz usuwanie, oraz przeglądanie profili,
* Zarządzanie zdjęciami(galeria zdjęć) – dodawanie, usuwanie zdjęć które będą dostępne do obejrzenia dla zwykłych użytkowników,
* Zarządzanie członkami – dodawanie, edycja, usuwanie członków stowarzyszenia,
* Zarządzanie artykułami – dodawanie, edycja usuwanie artykułów które pojawiać się będą w kolejności od najmłodszego do najstarszego na stronie głównej,
* Zarządzanie zawodami - dodawanie, edycja, usuwanie. Zawody mogą być aktywne czyli użytkownik może się do nich dopisać lub nieaktywne bez możliwości zapisów dla zwykłych użytkowników,
* Zarządzanie uczestnikami zawodów – dodawanie, edycja, usuwanie. Od strony administracji jest możliwość dodanie uczestnika do nieaktywnych zawodów,
* Przeglądanie, edycja własnego profilu.

Każda z tych funkcjonalności dostarcza możliwość przeglądu wszystkich wierszy z bazy danych, wyszukiwanie oraz sortowanie ich po kolumnach, wyświetlanie odpowiednią ilość wierszy.

Użytkownik powinien mieć możliwość dostępu bez potrzeby zalogowania do następujących funkcjonalności:

* Przeglądanie artykułów,
* Przeglądanie zawodów,
* Przeglądanie galerii zdjęć,
* Dopisanie się do zawodów jako uczestnik.

## Diagram przypadków użycia



Rys. . Diagram przypadków użycia

Powyższy diagram (Rys. 4.1) reprezentuje przypadki użycia aplikacji. Wszystkie przypadki użycia do których ma dostęp administrator wymagają wcześniejszego zalogowania się do systemu, jednak zostało to pominięte na diagramie w celu zwiększenia jego czytelności. Poniżej dla przykładu zostaną opisane dwa przypadki użycia z diagramu.

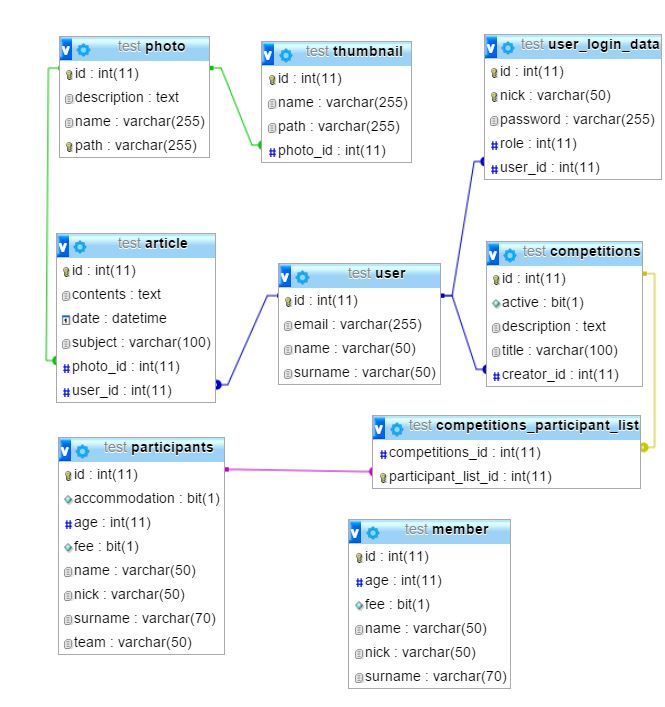
Opis przypadku użycia „Edycja artykułu”

* Uczestniczący:
  + Administrator.
* Podstawowy ciąg zdarzeń:
  + Administrator loguje się do systemu,
  + Administrator wybiera odnośnik „Artykuły” z paska nawigacyjnego po lewej stronie,
  + Administrator wyszukuje w tabeli który artykuł chce edytować,
  + Administrator klika odnośnik „Edycja” z ostatniej kolumny „Akcje” w tabeli,
  + Administrator edytuje nagłówek artykułu, jego treść oraz zdjęcie zależnie od potrzeb,
  + Administrator zatwierdza formularz,
  + System sprawdza czy nowe wprowadzone dane są odpowiednie,
  + System wprowadza nowe dane do artykułu i zapisuje go w bazie danych,
  + System zwraca komunikat o prawidłowej edycji artykułu.
* Alternatywne ciągi zdarzeń:
  + System nie zatwierdza nowych danych, bądź pola są puste,
    - System wyświetla stosowny komunikat z błędem.
  + Administrator rezygnuje z edycji artykułu i klika przycisk anuluj,
    - System nie zapisuje żadnych danych.
* Zależności czasowe:
  + Częstotliwość wykonania : ok 3 razy na tydzień,
  + Średni czas realizacji: ok 5min,
  + Maksymalny czas realizacji: nieokreślony,
  + Przewidywane spiętrzenia: okres letni.
* Wartości uzyskiwane przez aktorów po zakończeniu przypadku użycia:
  + Komunikat potwierdzający prawidłową edycję artykułu,
  + Zmianę treści w wybranym przez aktora, edytowanym artykule,

Opis przypadku użycia: „Dopisanie uczestnika do zawodów”

* Uczestniczący:
  + Użytkownik,
* Podstawowy ciąg zdarzeń:
  + Użytkownik wybiera odnośnik zawody w górnym pasku nawigacyjnym,
  + Użytkownik wybiera zawody z rozwiniętej listy zawodów,
  + Użytkownik wybiera przycisk „zapisz się”,
  + Użytkownik wypełnia formularz zapisowy do zawodów tj. uzupełnia wymagane dane,
  + Użytkownik zatwierdza dane wybierając przycisk „zatwierdź”,
  + System sprawdza czy wprowadzone dane są odpowiednie,
  + System tworzy uczestnika i dodaje go do bazy danych,
  + System przypisuje wcześniej stworzonego uczestnika do zawodów,
  + System zwraca komunikat o udanym zapisie do zawodów.
* Alternatywne ciągi zdarzeń:
  + System nie zatwierdza nowych danych, bądź pola są puste,
    - System wyświetla stosowny komunikat z błędem.
  + Użytkownik rezygnuje z zapisów i klika przycisk anuluj,
    - System nie zapisuje żadnych danych.
* Zależności czasowe:
  + Częstotliwość wykonania : ok 40 razy na tydzień,
  + Średni czas realizacji: ok 3min,
  + Maksymalny czas realizacji: nieokreślony,
  + Przewidywane spiętrzenia: przypadek użycia występuje jedynie w okresie letnim.
* Wartości uzyskiwane przez aktorów po zakończeniu przypadku użycia:
  + Komunikat potwierdzający zapis na zawody, oraz komunikat przypominający o konieczności opłaty uczestnictwa w zawodach,
  + Zapisanie się użytkownika na zawody.

## Schemat bazy danych



Rys. . Schemat relacyjnej bazy danych

**User** – tabela opisująca dane logowania użytkowników.

* Każdy użytkownik rozpoznawany jest po unikalnym numerze ID,
* Każdy użytkownik posiada pola password oraz unikalny nick które są używane do logowania,
* Hasła użytkowników zapisywane są w bazie w postaci skrótu po szyfrowaniu MD5,
* Użytkownicy mają przypisane role,
* Każdy użytkownik ma referencję do wierszu w tabeli „user\_details” z szczegółami konta.

**User\_details** – tabela przechowująca szczegółowe dane każdego użytkownika.

* Użytkownik ma możliwość wprowadzenia danych takich jak imię, nazwisko i email,
* Każdy zestaw szczegółowych danych użytkownika posiada swoje unikalne ID

Competitions – tabela przechowuje informacje o organizowanych zawodach,

* Każde zawody są rozpoznawane pod unikalnym numerze ID,
* Pole „active” określa czy zawody są aktywne czy nie co oznacza czy się już odbyły,
* Zawody można opisać poprzez pola „title” oraz „description”, pierwsze z kolei oznacza nazwę zawodów, drugie ich opis,
* Każda tabela przechowuje ID użytkownika który zawody tworzył.

**Participant** – tabela przechowuje informacje o uczestnikach zawodów.

* Każdy uczestnik jest rozpoznawany po unikalnym numerze ID,
* Podstawowe dane osobowe o uczestniku jakie przechowuje tabela to imie, nazwisko, nick, wiek,
* Poza podstawowymi danymi osobowymi dochodzi zakwaterowanie, opłata oraz drużyna do której należy uczestnik.

**Competitions\_participant\_list** – tabela tworzy połączenie między tabelami „competitions” i „participant”, co w efekcie daje list uczestników przypisanych do każdych zawodów.

* Tabela zawiera unikalne ID zawodów, definiujące do jakich zawodów zapisany jest uczestnik,
* Tabela zawiera unikalne ID uczestnika który zostaje przypisany do zawodów.

**Article** – tabela przechowująca artykuły pisane przez administratorów i wyświetlane na głównej stronie.

* Każdy artykuł rozpoznawany jest po unikalnym numerze ID,
* Każdy artykuł składa się z tematu i opisu, daty utworzenia,
* Każdy artykuł zawiera unikalny numer ID twórcy,
* Każdy artykuł zawiera unikalny numer ID zdjęcia do niego przypisanego.

**Photo** – tabela przechowująca informacje o zdjęciach wysłanych na serwer,

* Każde zdjęcie jest rozpoznawane po swoim unikalnym numerze ID,
* Zdjęcie składa się z opisu i nazwy.
* Zdjęcia nie są fizycznie przechowywane w bazie danych, przechowywane są jedynie ścieżki do tych zdjęć.

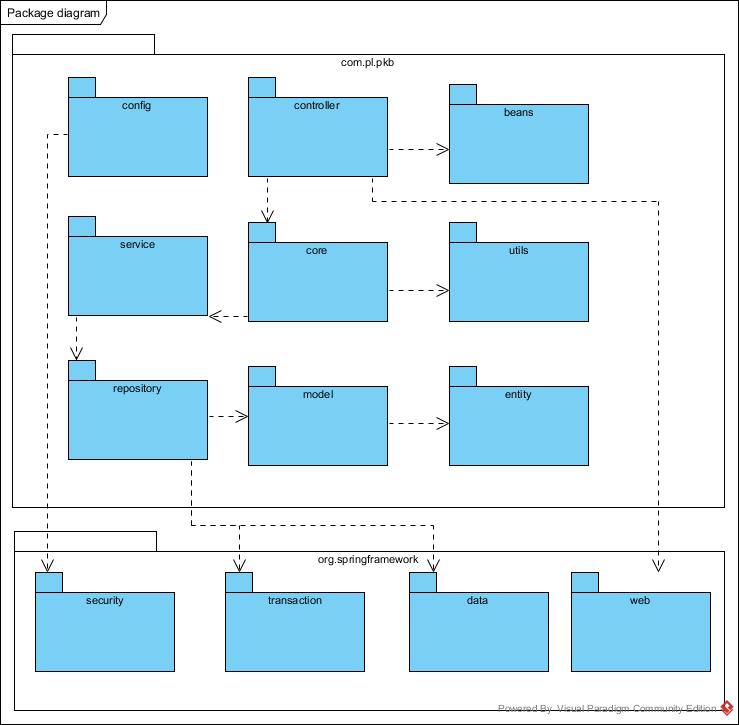
**Thumbnail** – tabela przechowywująca informacje miniaturkach zdjęć.

* Każda miniaturka jest identyfikowana po unikalnym numerze ID,
* Miniaturka posiada swoją nazwę,
* Miniaturka nie jest fizycznie przechowywana w bazie danych, przechowywana jest jedynie ścieżka do miniaturki,
* Każda miniaturka posiada unikalne ID zdjęcia do którego jest przypisana.

**Member** – tabela przechowywująca informacje o członkach klubu sportowego.

* Każdy członek jest identyfikowany po unikalnym numerze ID,
* Dane członka to imie, nazwisko, nick, wiek, opłata informująca czy składka została opłacona, data opłaty,

## Diagram pakietów



Rys. . Diagram pakietów

Powyższy diagram pakietów (Rys. 4.3) ilustruje oparcie aplikacji o szkielet Spring Framework.

* config – w tym pakiecie znajdują się klasy konfigurujące aplikację, pod względem zabezpieczeń, oraz mapowania dodatkowych źródeł,
* controller – pakiet zawierający klasy klasy kontrolerów, tutaj mapowane są ścieżki URL do konkretnych akcji w aplikacji. Jest to pakiet w którym znajdujące się klasy pośredniczą w komunikacji pomiędzy widokiem aplikacji a całą jej logiką,
* beans – pakiet zawierający klasy wspomagające przesyłanie informacji z kontrolera do widoku oraz z widoku do kontrolera,
* core – aktualnie pakiet zawiera jedną klasę „PkbWebAppManager” są w niej wstrzyknięte wszystkie serwisy jako obiekty statyczne,
* service – pakiet zawierający serwisy czyli klasy wykonujące konkretne akcje na wybranych encjach w bazie danych.W pakiecie znajdują się interfejsy serwisów jak i ich implementacje w wewnętrznym pakiecie „impl”,
* utils – pakiet z użytecznymi klasami wspomagającymi akcje. Udostępnia metody tworzące miniatury zdjęć i obrabiające zdjęcia odebrane przez kontroler z formy w jakiej należało je przesłać do kontrolera na format jpg,
* repository – pakiet zawierający interfejsy dostarczające metody wykonywujące akcje na bazie danych, interfejsy są dopasowane do każdej encji oraz dziedziczą po „CrudRepository” interfejsie należącym do Spring Framework,
* model – pakiet zawierający modele danych wykorzystywanych w aplikacji takich jak aktualny użytkownik czy role, oraz zawiera pakiet „entity”,
* entity – pakiet zawierający obiekty encji, odpowiadające tabelom z bazy danych. Znajduje się tu cała reprezentacja bazy danych w formie klas java.

# Implementacja aplikacji

W tym rozdziale zostaną opisane elementy implementacji aplikacji.

## Adresacja URL

URL(ang. Uniform Resource Locator) czyli format adresowania zasobów w intrenecie, najczęściej kojarzony jest z adresacją stron www, jednak dotyczy on adresacji wszystkich zasobów dostępnych w intrenecie.

Każda aplikacja internetowa powinna posiadać przemyślane adresy swoich stron internetowych oraz jej zasobów . Nie tylko ze względów estetycznych, ale również może to ułatwić pracę dla programisty, oraz poruszanie się dla końcowego użytkownika.

* Zastosowanie wspólnego prefiksu dla stron oraz zapytań, umożliwia efektywne zarządzanie prawami dostępu użytkowników.
* Użycie słów kluczowych dotyczących zawartości strony może ułatwić użytkownikom odnajdywanie strony oraz może mieć wpływ na jej pozycjonowanie ponieważ wyszukiwarki odczytują słowa kluczowe z adresu. Przykładowo Google rekomenduje używanie łącznika a nie podkreślnika przy adresowaniu stron.

Poniżej zostaną przedstawione mapowania adresów URL zastosowane w budowanej aplikacji internetowej.

**Uwierzytelnienie**

* Adres „/login” dotyczy logowania do panelu administratora,
* Adres „/logout” wylogowuje aktualnie zalogowanego użytkownika.

**Panel administracyjny**

* Wszystkie adresy rozpoczynające się od prefiksu „/admin” należą do panelu administracyjnego.

**Użytkownicy**

* Adresy rozpoczynające się od prefiksu „/admin/users” dotyczą modułu zarządzania kontami użytkowników aplikacji,
* „/admin/users/add” pozwala na dodanie nowego użytkownika dane wysyłane jako parametry typu POST,
* „/admin/users/removeUser/{id}” usuwanie użytkownika o id wskazanym w zmiennej id znajdującej się w URL która jest przesyłane jako parametr typu GET,
* „/admin/profile/” pozawala na przeglądanie profilu aktualnie zalogowanego użytkownika gdy jest wykonywane jako zapytanie typu GET, gdy pod ten adres zostanie wysłane zapytanie typu POST odpowiada ono za edycję profilu aktualnie zalogowanego użytkownika.

**Członkowie klubu**

* Adresy rozpoczynające się od prefiksu „/admin/memeber” dotyczą modułu zarządzania członkami klubu,
* „/admin/memeber/add” pozwala na dodanie nowego członka klubu, dane wysyłane jako parametr typu POST,
* „/admin/memeber/removeMember/{id}” usuwanie członka klubu o wskazanym id które jest przesyłane jakos parametr typu GET,
* „/admin/memeber/editMember/{id}” edycja członka klubu o wskazanym id wysłanym jako parametr typu GET, jest to zapytanie typu REST zwraca ono odpowiedź w postaci obiektu zapisanego w formacie JSON. Nie powoduje tak jak inne zapytania przekierowania, przeładowania strony oraz nie zwraca widoku.

**Uczestnicy zawodów.**

* Adresy rozpoczynające się od prefiksu „/admin/participants” dotyczą modułu zarządzania wszystkimi uczestnikami zawodów,
* „/admin/participants/add” pozwala na dodanie nowego uczestnika do wybranych, dane wysyłane jako parametr typu POST,
* „/participants/{id}/add” pozwala na dodanie uczestnika do zawodów, dane wysyłane są jako parametr typu POST, dostęp do tego URL posiadają wszyscy,
* „/admin/participants/removeParticipant/{id}” usuwanie zawodów o wskazanym id które przesyłane jest jako parametr typu GET,
* „/admin/participants/editParticipant/{id}” edycja zawodów o wskazanym id które jest przesyłane jako parametr typu GET, jest to zapytanie typu REST.

**Zawody**

* Adresy rozpoczynające się od prefiksu „/admin/competitions” dotyczą modułu zarządzania zawodami,
* Moduł zarządzania zawodami składa się podobnie jak pozostale moduły z podstron o adresach „/add”, „/removeCompetitions/{id}”,”/editCompetitions/{id}” które działają podobnie jak te w wyżej opisanych modułach,
* dodatkowo zawiera mapowanie „/competitionList” jest to zapytanie typu REST zwracające listę z przyporządkowanymi tytułami zawodów do odpowiadających im unikalnych numerów id, jest to URL dostępny dla wszystkich,
* „/viewCompetitions/{id}” mapowanie zwracające użytkownikowi widok szczegółowy zawodów, URL dostępny dla wszystkich.

**Artykuły**

* Adresy rozpoczynające się od prefiksu „/admin/articles” dotyczą modułu zarządzania artykułami,
* Podobnie jak poprzednie moduły, ten składa się z podstron „/add”, „/removeArticle/{id}”,”/editArticle/{id}” które działają podobnie jak te w wyżej opisanych modułach.

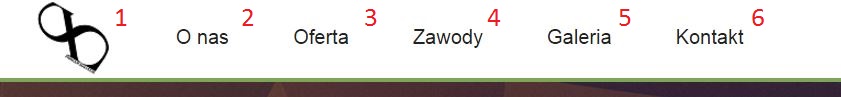
**Zdjęcia**

* Adresy rozpoczynające się od prefiksu „/admin/phoho” dotyczą modułu zarządzania zdjęciami,
* „/admin/phoho/upload” pozwala na dodanie zdjęcia za pomocą formularza typu multipart wysyłanego za pomocą zapytania POST,
* „/admin/phoho/removePhoto/{id}” usuwanie zdjęcia o wskazanym id.

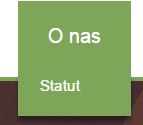
## Interfejs Graficzny

Interfejs graficzny jest niezwykle ważną częścią aplikacji ponieważ odpowiada za przekazanie informacji oraz umożliwia wykonywania akcji użytkownikom. Widok generowany jest przez silnik Thymeleaf który odbierając informacje przekazane z kontrolera, bazując na napisanym kodzie HTML i własnych atrybutach buduje stronę. Estetyczny wygląd elementów został zaimplementowany z pomocą frameowrka Bootstrap który dostarcza gotowe CSS, natomiast dynamikę stronie nadaje bibloteka JQuery napisana w JavaScript.

### Widok użytkownika

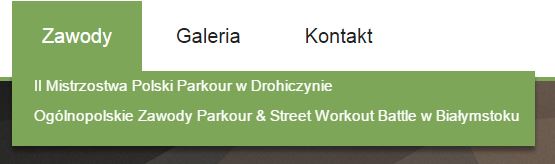
**Pasek nawigacji**

Rys. . Pasek nawigacyjny - widok użytkownika



1. Logo stowarzyszenia Parkour Białystok jednocześnie jest odnośnikiem do strony głownej.
2. „O nas” (Rys. 5.3) odnośnik do strony opisującej stowarzyszenie jednocześnie po najechaniu rozwijane jest dynamiczne menu, zawierające odnośnik do statutu stowarzyszenia.

Rys. . Menu "O nas"

1. „Oferta” (Rys 5.5) po najechaniu na element rozwijane jest dynamiczne menu zawierające odnośniki do strony zawierającej informacje na temat w jaki sposób dołączyć do stowarzyszenia oraz opis pokazów jakie organizuje stowarzyszenie.
2. „Zawody” (Rys 5.4) dynamiczne menu pokazujące się po najechaniu na ten element jest generowane na podstawie ilości organizowanych zawodów przez stowarzyszenie znajdującej się w bazie danych.

Rys. . Menu zawody

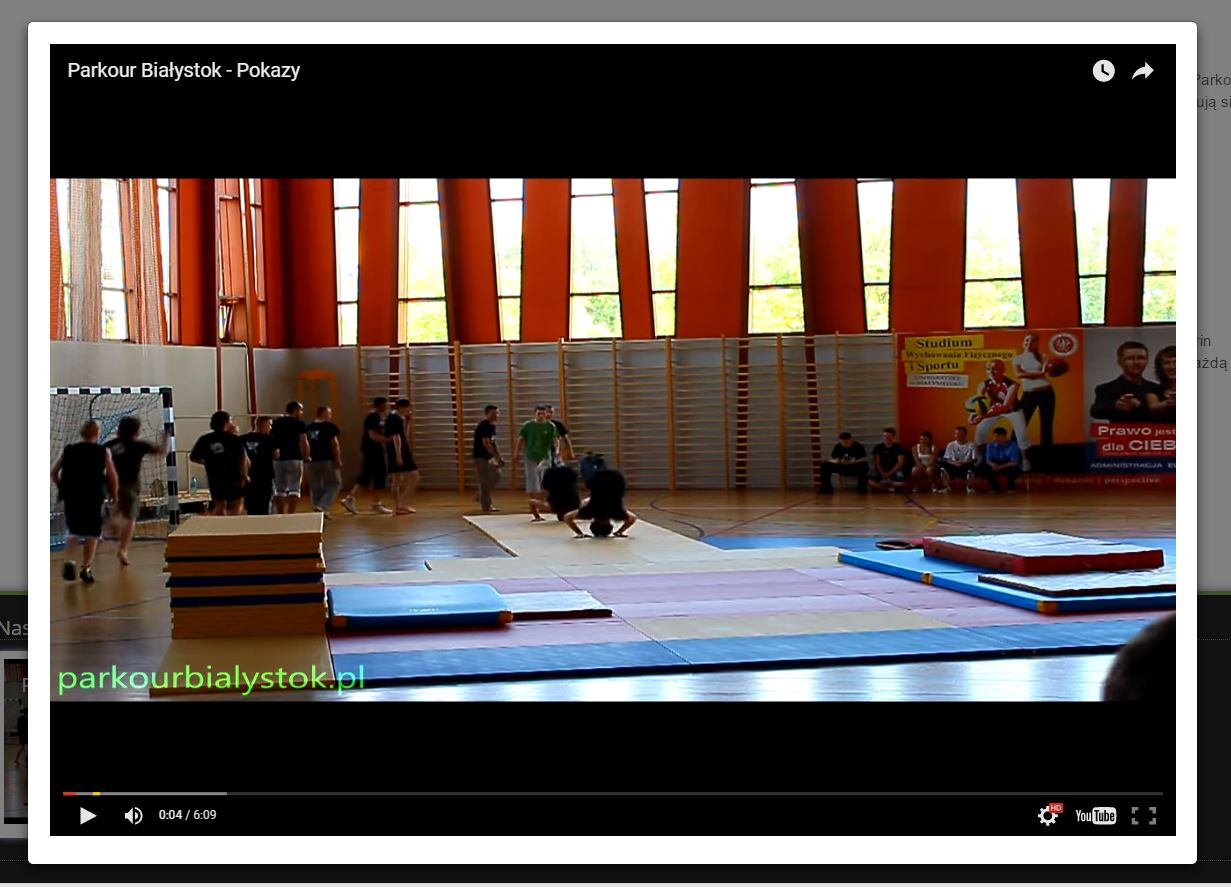
Rys. . Menu "Oferta"

1. Menu składa się jeszcze z elementów „Galera” z odnośnikiem do galerii która będzie opisana w dalszej części pracy oraz „Kontakt” do uzyskana danych kontaktowych.

**Stopka**



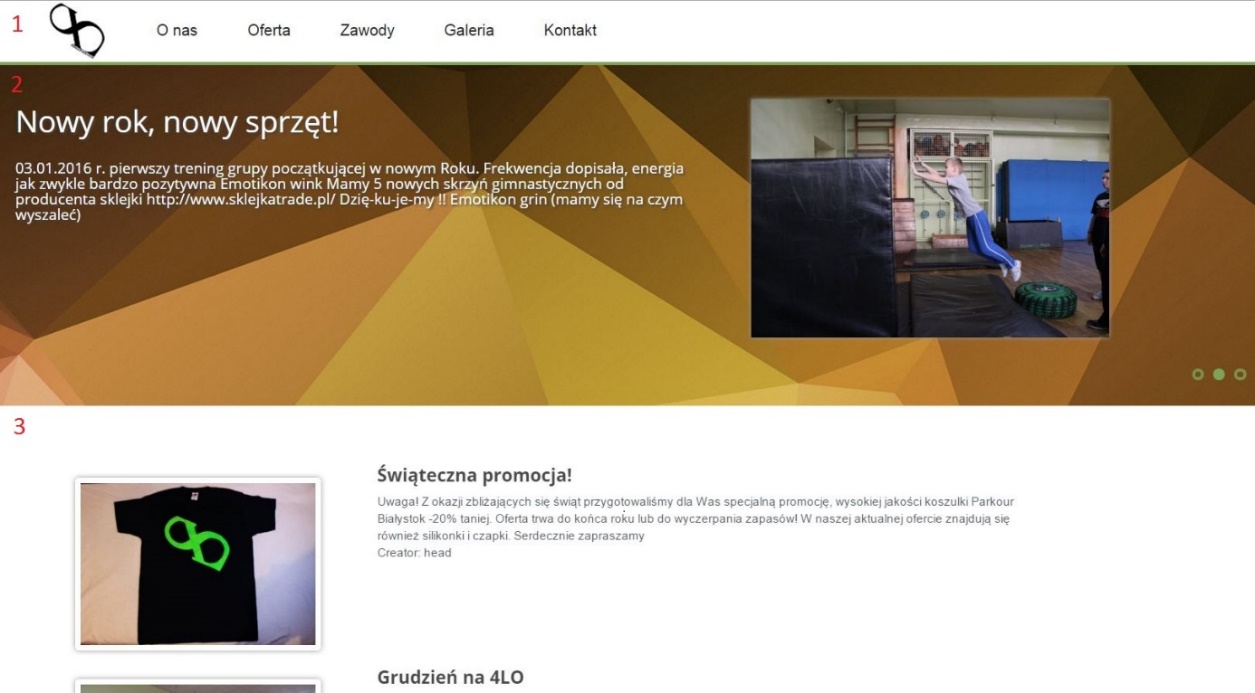
Rys. . Stopka

1. Element wywołujący okno dialogowe (Rys. 5.7) za pomocą którego można obejrzeć film z prowadzonych przez klub sportowy pokazów w serwisie YouTube znajduje się tam również odnośnik do filmu w serwisie YouTube.

Rys. . Okno dialogowe YouTube

1. Nawigacja – zbiór odnośników znajdujących się na stronie, oraz link do panelu administracyjnego.
2. Podstawowe dane kontaktowe stowarzyszenia.
3. Odnośniki do stron na portalach społecznościowych, aktualnie stowarzyszenie posiada jedyne konto na Facebook.

**Index**



Rys. . Strona główna aplikacji

1. Wyżej opisany pasek nawigacyjny.
2. Slider prezentujący trzy ostatnio dodane artykuły. Zawartość prezentowana jest w krótkich odstępach czasowych od najnowszego do najstarszego z 3 prezentowanych artykułów. Po pokazaniu się ostatniego slider powraca do pierwszego. Artykuły nie są wyświetlane w pełni jeżeli tekst jest zbyt długi, kliknięcie na element przeniesie na stronę z pełną zawartością.
3. Pozostałe artykuły wyświetlane w formie listy, artykuł jeżeli jest z byt długi nie jest wyświetlany w całości, kliknięcie na element wywołała przeniesienie na stronę gdzie zostanie wyświetlony w pełnej zawartości.
4. Stopka.

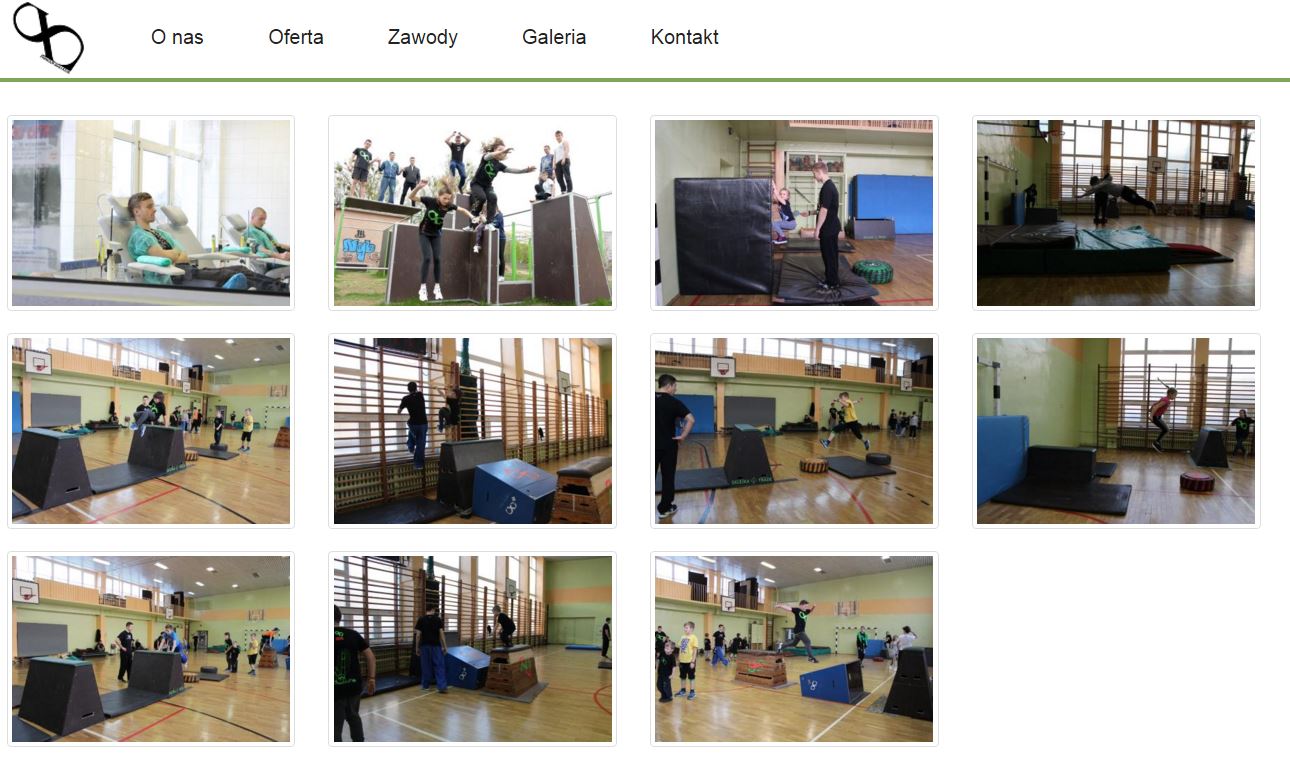
**Zawody**



Rys. . Strona prezentująca zawody

1. Pasek nawigacyjny omówiony wyżej.
2. Tytuł zawodów.
3. Treść opisująca zawody oraz zdjęcie je prezentujące. Treść jest napisania w kodzie html dzięki czemu użytkownik ma wpływ na to w jaki sposób się wyświetla.
4. Przycisk zapisu na zawody umożliwiający przeglądającym zapisanie się jako uczestnik, niewidoczny na rysunku (Rys 5.9).
5. Stopka.

**Galeria**



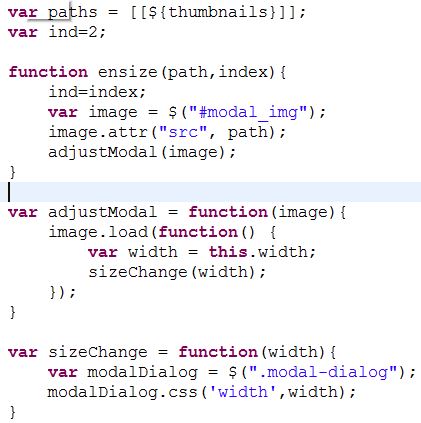
Rys. . Galeria zdjęć

Galeria zdjęć (Rys. 5.10) umożliwia użytkownikowi przeglądanie zdjęć udostępnionych na stronie. Zdjęcia są wyświetlanie jedno po drugim formie tabeli. Kliknięcie w zdjęcie wywołuje okno dialogowe które umożliwia obejrzenie go w pełnej rozdzielczości.



Rys. . Okno dialogowe galerii zdjęć

Powyższe okno dialogowe (Rys 5.11). po lewej i prawej stronie zdjęcia zawiera strzałki służące do poruszania się między zdjęciami. Strzałka w lewo wyświetli poprzednie zdjęcie z tabeli natomiast strzałka w prawo następne. Przykładowo dotrzemy do pierwszego zdjęcia i będziemy przeglądać dalej zostanie wyświetlone ostatnie zdjęcie z tabeli.

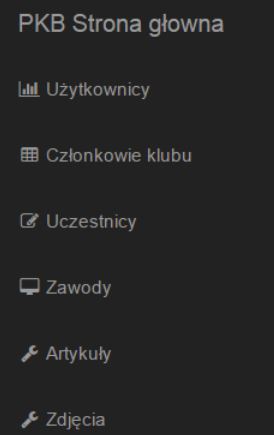
Do realizacji okna dialogowego należało napisać dodatkowe funkcje (Rys 5.12).

* Do zmienne „paths” przypisana została przypisana tablica ze ścieżkami wysłana z kontrolera. Pierwszy element tablicy zawiera ścieżkę do miniaturki natomiast drugi do zdjęcia w pełnym wymiarze.
* Funkcja „ensize” wywołuje dopasowanie rozmiaru okna dialogowego do zdjęcia, oraz podmianę ścieżki.

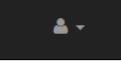
Rys. . Funkcje JavaScript galerii

* Funkcja adjustModal odpowiada za pobranie rozmiaru obrazka i zmianę rozmiaru okna.
* Funkcja „sizeChange” ustawia szerokość okna.
* Niewidoczne na zdjęciu funckje „prev” i „next” wywoływane są w momencie kliknięcia na strzałki w oknie dialogowym, odpowiadają podmianę zdjęć na poprzednie bądź kolejne.

### Widok administracyjny

**Paski nawigacji**

Panel administracyjny posiada dwa paski nawigacji:

1. Boczny - zawierający wszystkie odnośniki do modułów administracyjnych odpowiednich za zarządzanie aplikacją (Rys. 5.14)..
2. Górny – posiadający przekierowanie na stronę główną z lewej strony oraz ikonę profilu z rozwijalnym menu z prawej strony (Rys. 5.13). Menu zawiera odnośnik do profilu aktualnie zalogowanego administratora oraz przycisk wylogowania.

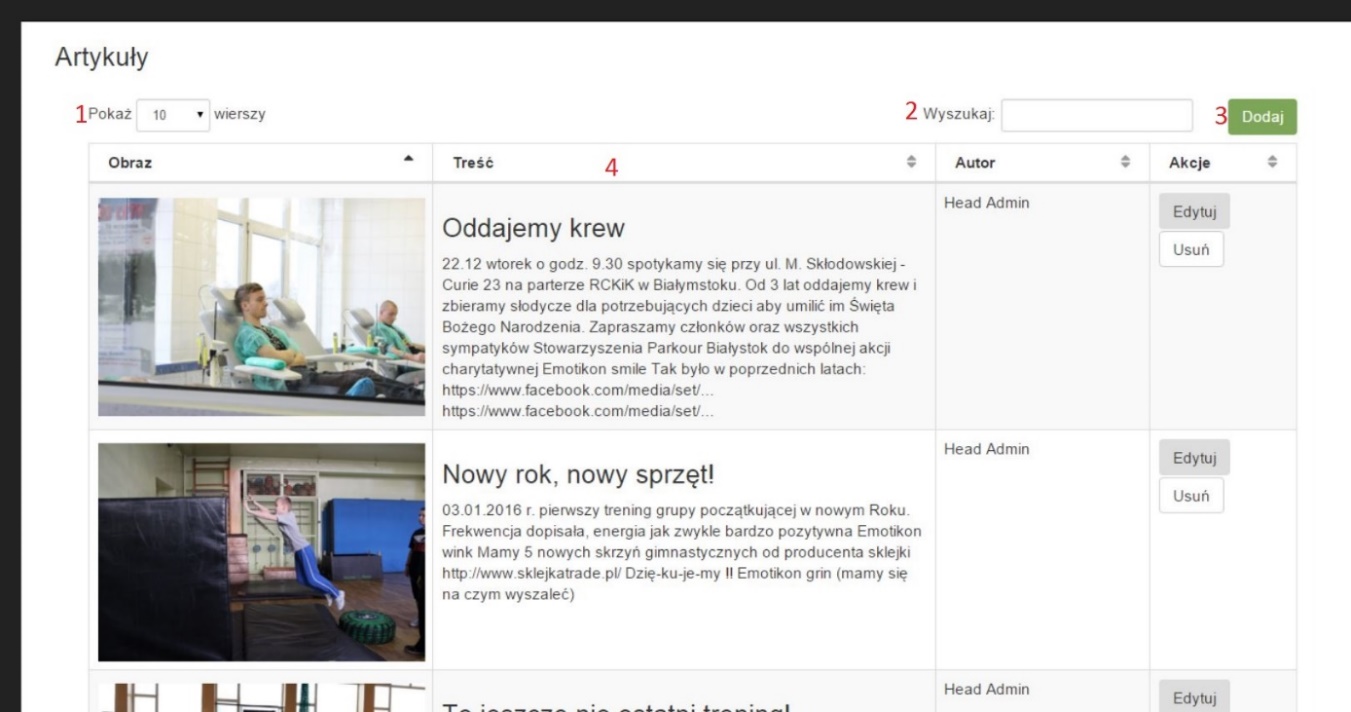
Rys. . kona profilu

Paski nawigacyjne są zawsze na wierzchu co oznacza że poruszają się w momencie przewijania treści przez użytkownika, nie trzeba powracać na górę strony żeby ich użyć.

Rys. . Pasek nawigacyjny

**Tabela danych**

Każdy moduł zarządzania poszczególnym aspektem aplikacji posiada własną tabelę dostosowaną do zawartości. Opisana zostanie jedynie należąca do zarządzania artykułami (Rys 5.13).

****

Rys. .13 Tabela artykułów

1. Administrator ma możliwość określenia ile wierszy ma być pokazanych w tabeli, robi to za pomocą rozwijalnej listy gdzie może wybrać żądaną wartość. Dostępne wartości to 10, 25,50,100.
2. Opcja wyszukiwania - administrator ma możliwość przeszukiwania tabeli. Należy wpisać szukaną frazę w polu, po podaniu drugiej litery aplikacja rozpocznie dynamicznie wyszukiwanie we wszystkich kolumnach tabeli.
3. Przycisk dodaj wywołujący okno dialogowe umożliwiające dodanie nowej zawartości do aplikacji, w tym przypadku dodanie nowego artykułu. Wyświetlane okno dialogowe zawiera formularz do którego należy wprowadzić odpowiednie dane aby dodać nową zawartość do aplikacji.
4. Tabela zawartości - prezentuje dodane już artykuły. W tym przypadku tabela składa się ze zdjęcia artykułu, jego treści czyli tytułu i treści właściwej, autora oraz dostępnych akcji.

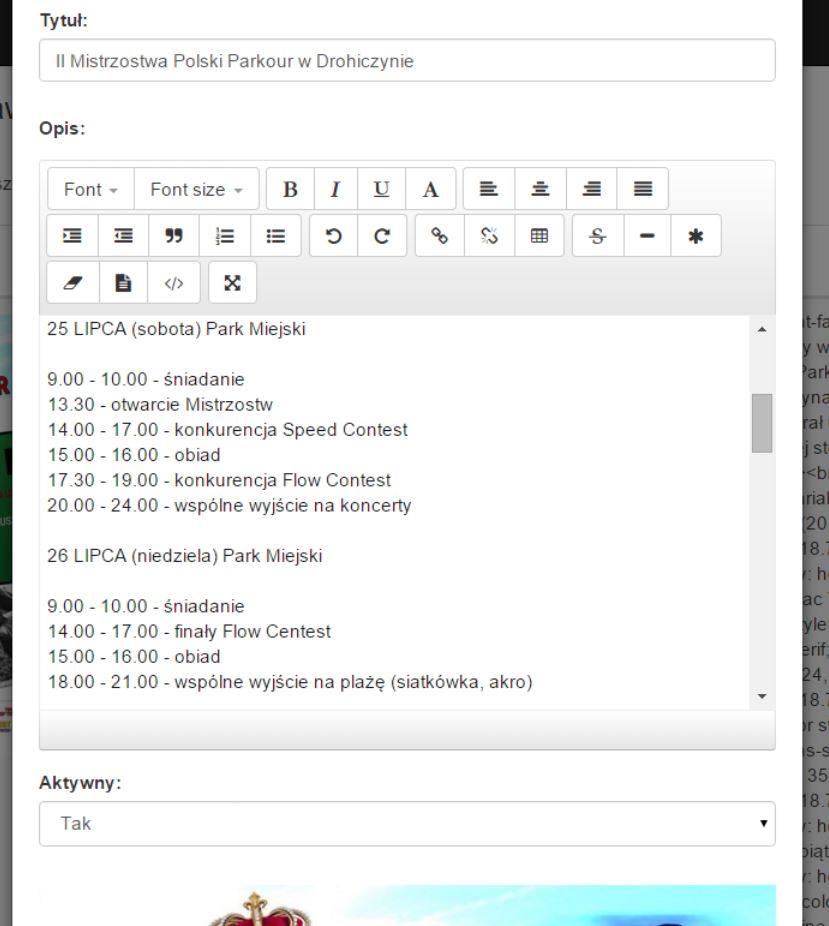
* „Edytuj” – wyświetla okno dialogowe z formularzem edycji artykułu, jest to samo okno z opcji dodaj jednak wypełnione o odpowiednie wartości dzięki którym aplikacja rozpozna że taki artykuł już istnieje i należy go zaktualizować,
* „Usuń” - usuwanie żądanego artykułu.

Tabela w każdym module zarządzania jest dopasowana do pól jakie posiada obiekt encji którymi zarządza dany moduł.

## Formularze

Dodawanie nowych zawartości jak i edycja istniejących już w aplikacji opiera się o formularze, które użytkownik wypełnia i wysyła do kontrolera za pomocą kliknięcia w odpowiedni przycisk. Formularze są wysyłane za pomocą metody POST, więc nie są jawne i nie można zobaczyć ich zawartości przykładowo w wysyłanym URL. Ogólne działanie formularzy jest podobne ponieważ należy wypełnić odpowiednie pola i go zatwierdzić. Każdy formularz ma dopasowane pola do obiektu encji na jakim operuje w aplikacji.

### Tworzenie zawodów



Rys. .14 Formularz edycji istniejących zawodów

Na powyższym rysunku (Rys 5.14) widać formularz edycji istniejących zawodów, jest to jeden z bardziej interesujących formularzy ze względu na sposób dodawania treści.

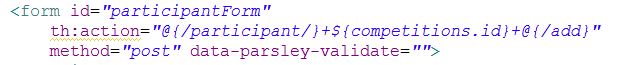
Formularz edycji zawodów składa się z:

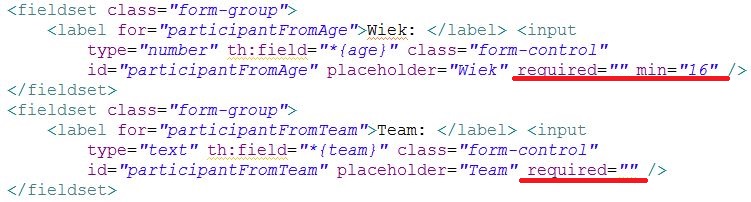
* Pola „Tytuł” typu input do wprowadzania tytułu zawodów, wyświetlanego w pozycji 2 (Rys 5.9),
* Pola „Opis” typu textarea służące do tworzenia opisu zawodów. Pole to narzędzie do generowania kodu html, który generowany jest automatycznie podczas gdy użytkownik tworzy opis zawodów używając funkcji tego narzędzia. Użytkownik ma do dostępu między innymi:
  + zmiana rozmiaru czconki,
  + zmiana czcionki, wytłuszczeni,
  + kursywa,
  + wyrównania do prawej czy lewej,
  + wstawienie tabeli.

Oczywiście nie są to wszystkie funkcje jakie oferuje to pole. Dostarczona jest również możliwość podglądu wygenerowanego kodu HTML. Wygenerowany HTML jest wysyłany i zapisywany w bazie danych, a w momencie odczytu pobierany w całości i dołączany do odpowiedniego tagu <div> w szablonie HTML przeglądania zawodów.

* Pole „Aktywny” informujące czy uczestnicy maja jeszcze możliwość rejestrowania się na zawody. W momencie zamiany na wartości na „Nie” rejestracja na zawody jest zakończona, a dopisać nowych uczestników może tylko administrator,
* Aktualne zdjecie zawodów, widoczne jedynie w części na rys 5.14,
* Przycisk dodawania nowego zdjęcia,
* Przycisk anulujący wprowadzanie zmian do formularzu,
* Przycisk zatwierdzenia formularzu.

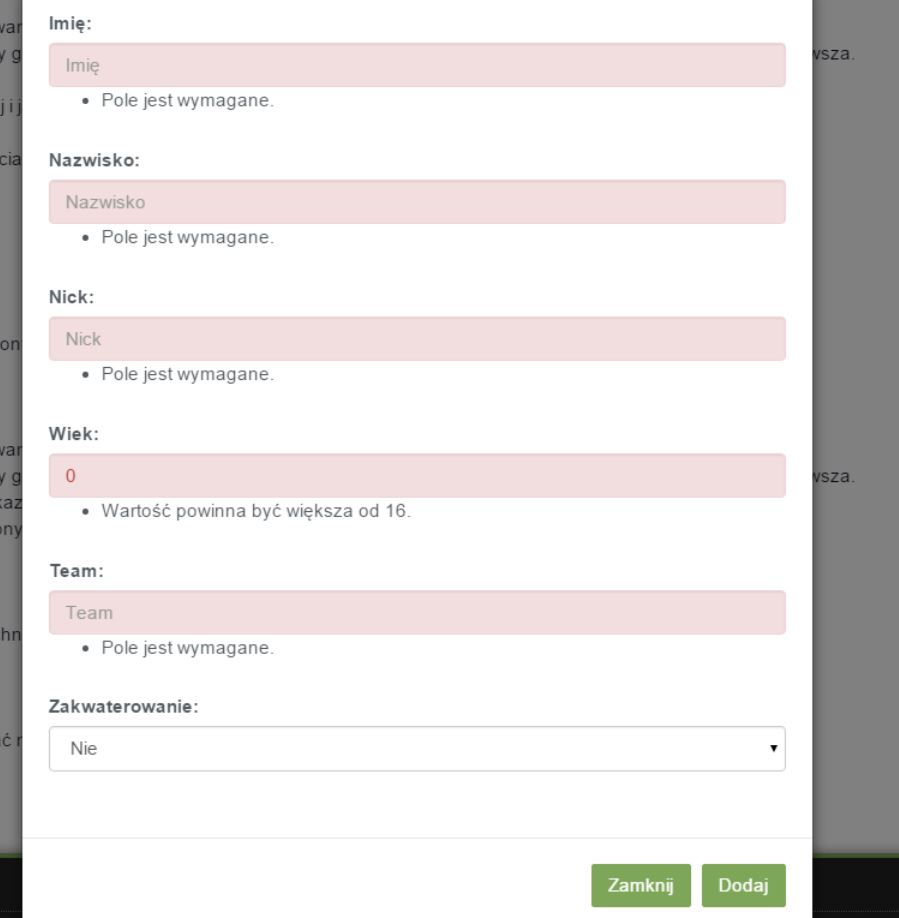
### Rejestracja na zawody – walidacja

****



Rysunek .15 Dodatkowe atrybuty HTML służace do walidacji.

Każdy formularz przesyłany do serwera należy wcześniej sprawdzić czy dane zostały wpisane poprawnie. Weryfikację danych uzyskano za pomocą biblioteki JavaScript „ParsleyJS”. Aby użyć wtyczki wystarczy oznaczyć formularz i pola odpowiednimi atrybutami (Rys. 5.15) oraz wykonać funkcję inicjalizującą bibliotekę w JavaScript „$("#participantForm").parsley();” .



Rys. .16 Formularz rejestracji na zawody - przykład walidacji danych.

Powyższy rysunek przedstawia (Rys 5.16) pola z nieprawidłowymi wartościami są podświetlane na czerwono, a pod każdym pojawia się informacja o błędzie. Jeżeli zostaną wykryte błędy wysyłanie formularza jest zatrzymywane.

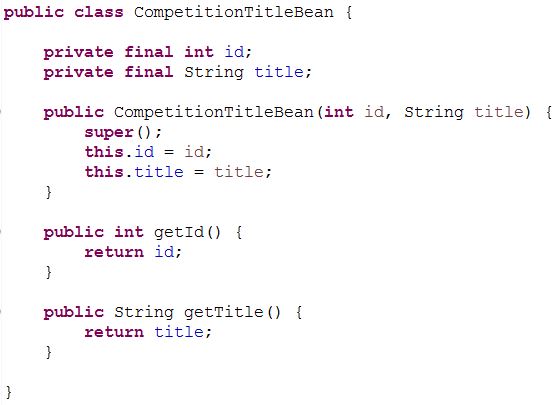
## Back-end

Podrozdział opisujący implementację części serwerowej. Opisane zostały tu zaimplementowane kontrolery, klasy konfiguracyjne, oraz manager aplikacji.

**Kontrolery & beany**

Do przesyłania informacji między widokiem a serwerem używane są kontrolery które z kolei wykorzystują beany.

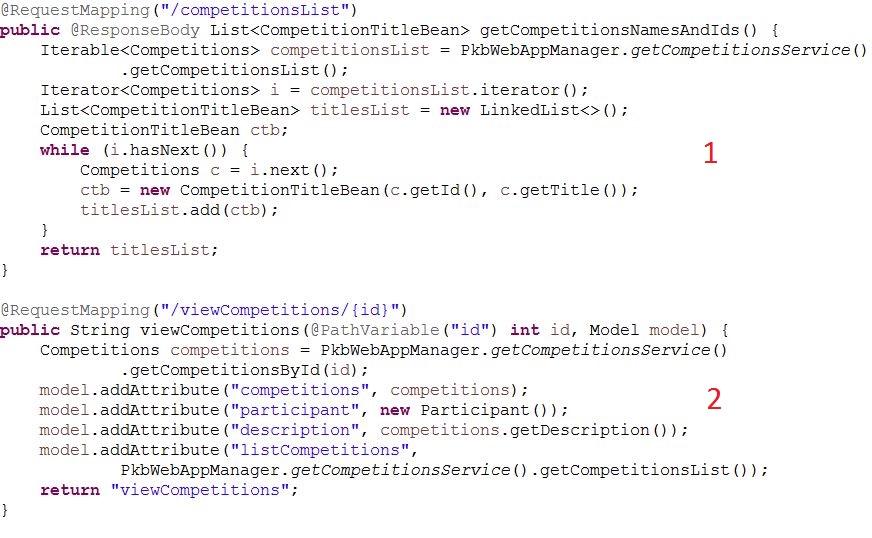
**Bean** jest to obiekt reprezentujący pewien model danych który używany przez kontroler mówi mu w jaki sposób ma mapować przychodzące lub wychodzące obiekty.



Rys. .17 Przykładowy bean z aplikacji.

Patrząc na powyższy rysunek (Rys. 5.17) można odnieść wrażenie że bean jest bardzo podobny do entity, I jest to prawda z jedną różnicą enitity służy do mapowania tabeli bazy danych, jednak nie wyklucza to wykorzystania go jako beana. Każdy bean składa się z konstruktora którym można zainicjalizować obiekt oraz metod gettorów i setterów do pobierania i ustawiania wartości obiektu.

**Kontroler** to element odpowiadający za mapowanie adresów URL do konkretnych funkcji w aplikacji. To on łączy widok aplikacji z całą funkcjonalnością aplikacji. Każdy da klasa kontrolera jest oznaczona adnotacją „@Controller” a każda funkcja jeżeli ma być mapowana pod adresem URL również musi zostać oznaczona poprzez adnotację „@RequestMapping” która określa nie tylko mapowany adres URL, ale również metodę przesyłania danych i wiele innych. Ten sam URL może wykonywać jedną akcję przy metodzie GET i inną przy metodzie POST. Metody kontrolerów zwracają cały widok z dodanymi parametrami lub przekierowanie na inny URL, jednak powoduje to przeładowanie strony. Również mogą być zwracane odpowiedzi typu REST czyli obiekty w typu JSON(ang JavaScript Object Notation), jest to reprezentacja tekstowa obiektu która może zostać odebrana przez skrypt JavaScript, wymiana informacji tego typu nie przeładowuje strony dzięki czemu aplikacja znacznie zyskuje na dynamiczności. Zasada działania jest bardzo prosta, za pomocą JavaScript wysyłane jest zapytanie AJAX pod konkretny adres url z odpowiednimi parametrami, a następnie odbierana jest odpowiedź tego adresu pod postacią obiektu JSON.



Rys. .18 Przykładowe metody kontrolera Competitions

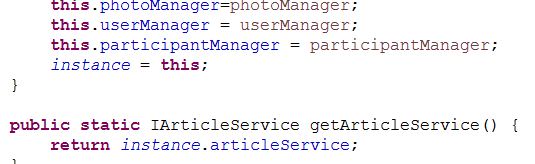
1. Jest to metoda typu REST, charakterystyczne jest dla nich oznaczenie adnotacją „@ResponseBody, wysłane zapytanie pod adres URL którym oznaczona jest metoda wywoła pobranie listy organizowanych zawodów, a następnie zostanie ona przeiterowana w celu pobrania tytułu i id każdych zawodów i stworzenia listy obiektów typu „CompetitionTitleBean” opisanych wyżej. Taka lista jest zwracana przez kontroler i konwertowana do formatu JSON.
2. Jest to zwykły kontroler który pobiera id z mapowanego URL, na jego podstawie wyszukuje organizowane zawody o pobranym id następnie dodaje caly obiekt zawodów jako paramter przesyłanego modelu, oraz dodatkowo jako oddzielny parametr dodaje opis zawodów. Zwracany obiekt typu String jest nazwą pliku z szablonem html odpowiadającym żądanemu widokowi, do generowanego szablonu są dodane załączone atrybuty do których odwoływać się będzie silnik Thymeleaf.

**Manager**



Rys. .19 Klasa PkbWebAppManager

Podczas tworzenia aplikacji w wielu miejscach wstrzykiwane były obiekty serwisów, co przy większym rozbudowaniu kodu zaczęło wyglądać bardzo chaotycznie i utrudniało pracę. Rozwiązaniem tego problemu stała się klasa PkbWebAppManager (Rys. 5.19), zawiera ona prywatne finalne pola dotyczące każdego serwisu i managera w aplikacji, oraz jedno pole prywatne statyczne „instance” z własną referencją. Podczas inicjalizacji obiektu wstrzykiwane i inicjalizowane są wszystkie wymagane pola przez konstruktor, a na końcu do pola „instance” przypisywana jest referencja do tego obiektu.



Rys. .20 Przykład metody get z klasy PkbWebAppManager

Rys 5.20 przedstawia sposób pobierania pól z obiektów za pomocą metody get, która zwraca nam żądany obiekt pobierając go wczesnej z obiektu „instance”. Takie rozwiązanie gwarantuje że obiekt „instance” będzie identyczny dla każdego obiektu klasy PkbWebAppManager, co za tym idzie każdy z serwisów również będzie identyczny.

**Konfiguracja**

Tworzona aplikacja posiada dwie klasy konfiguracyjne:

WebConfigure (Rys 5.21) rozszerzająca klasę WebMvcConfigurerAdapter. Ta klasa konfiguracyjna odpowiada za dodanie nowych folderów z zasobami w których przetrzymywane będą zdjęcia, porzed dodaniem źródeł sprawdza czy foldery te istnieją jeżeli nie tworzy je.

Rys. .1 Klasa konfiguracyjna "WebConfigure"

Kolejną funkcją tej klasy jest zmiana bazowych zasad serializacji obiektów leniwie inicjalizowanych, czyli takich które pobierane są z bazy jedynie gdy są używane, a nie w momencie pobrania innych obiektów w których się zawierają. Przy serializacji do JSON inicjalizacja obiektów leniwych jest wymuszana dzięki czemu mogą zostać serializowane.

Kolejna klasa „SecurityConfig” rozszerzająca klasę WebSecurityConfigurerAdapeter odpowiada za prawa dostępu określonych ról do poszczególnych mapowań, oraz zadeklarowanie interfejsu i kodowania hasła na bazie którego przebiega proces logowania.

Rys. .22 Klasa konfiguracyjna "SecurityConfig"

## Mechanizmy bezpieczeństwa

Mechanizm określania praw dostępu do adresów URL został już opisany w rozdziale „Opis wykorzystanych technologii” podrozdziale „Spring Security” dlatego nie będzie ponownie opisywany. W tym rozdziale opisany zostanie proces autoryzacji użytkownika czyli logowania oraz związane z nim elementy aplikacji.

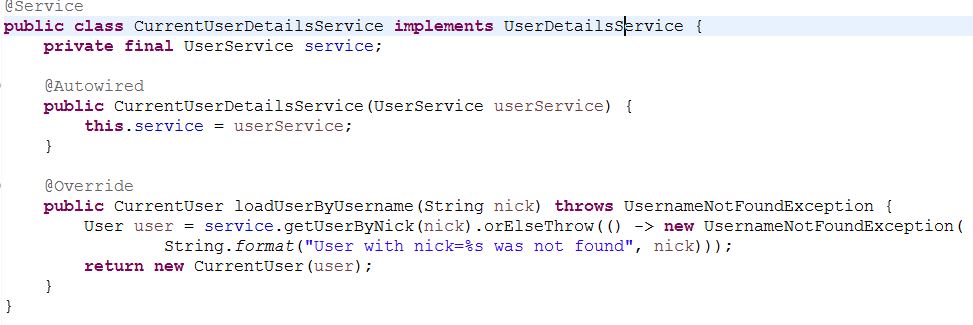
Aby dostać się do panelu administracyjnego aplikacja musi dokonać autoryzacji użytkownika, aby nie dopuścić sytuacji gdy osoby niepożądane posiadają dostęp do funkcji administracyjnych aplikacji. Zaimplementowanie logowania wymagało stworzenia dodatkowych klas rozszerzających interfejsy należące do Spring Security.



Rys. .23 Klasa CurrentUser reprezentująca zalogowanego użytkownika.

Klasa „CurrentUser” (Rys. 5.23) reprezentuje aktualnie zalogowanego użytkownika, rozszerza ona klasę User należącą do Spring Security. Ciało klasy stanowi prywatne pole typu User należące do aplikacji, metody get zwracające dane o zalogowanym użytkowniku, oraz konstruktor który wymaga podania obiektu typu User należącego do aplikacji, pobiera z niego nick, hasło oraz rolę i wywołuje konstruktor nadrzędny oraz przypisuje przekazany parametr do wewnętrznego prywatnego pola user. Obiekt klasy CurrentUser można ustawić tylko raz przy inicjalizacji obiektu za pomocą konstruktora, po tym można jedynie pobierać dane z klasy.

Kolejnym elementem jest klasa „CurrentUserDetailsService” (Rys 5.24) implementująca interfejs „UserDetailsService” należący do Spring Framework. Implementuje ona interfejs o którym wspomniano przy opisie klasy konfiguracyjnej „SecurityConfig” (rys 5.22). Klasa posiada prywatne finalne pole „service” do którego jest wstrzykiwany obiekt przez konstruktor. W klasie tej nadpisana została metoda „loadUserByUsername”, w taki sposób że po podanym w parametrze nicku czyli nazwie użytkownika(ang. user name), sprawdzane jest czy taki użytkownik istnieje a następnie zwraca zalogowanego użytkownika, hasła spring porównuje automatycznie rozkodowując te zakodowane bazie danych. Natomiast jeżeli nie zostanie odnaleziony taki użytkownik zostaje wyrzucany błąd „UsernameNotFoundException” informujący o braku takiego użytkownika w bazie danych.



Rys. .24 Klasa CurrentUserDetailsService klasa definiująca logowanie.

# Podsumowanie

Efektem wielomiesięcznej pracy jest gotowa do użycia aplikacja webowa wspomagająca działanie klubu sportowego spełniająca wszystkie postawione założenia zdefiniowane przez zakres pracy. Stworzona aplikacja będzie służyła zarządowi stowarzyszenia Parkour Białystok do przekazywania informacji, podstawowego zarządzania stowarzyszeniem, oraz organizacji i prowadzenia zapisów na zawody, powinno to ułatwić pracę nad rozwojem stowarzyszenia oraz oszczędzić czas i energię zarządu.

Budowa aplikacji złożyła się na następujące po sobie etapy. Pierwszym był przegląd podobnych rozwiązań, aby poznać już działające systemy. Kolejnym projekt aplikacji, to tutaj powstały przedstawione diagramy i schematy. Ostatnim etapem była implementacja, którą rozpocząłem od części serwerowej, a następnie dopisany został widok.

Praca nad aplikacją przyniosła wiele nowych doświadczeń z racji tego że Spring Framework był dla mnie nowością. Przed rozpoczęciem prac poświeciłem dużo czasu na naukę o tym frameworku jednak, to praktyka pozwoliła mi zrozumieć zasadę jego działania. Podczas pracy zdałem sobie sprawę jak bardzo ważny jest etap projektowania aplikacji, etap przemyślenia przebiegu pracy i sposobu jej działania. Odpowiednie przygotowania mogą oszczędzić wiele czasu i kłopotów, które wynikają w czasie implementacji. Niewątpliwie wiedza którą posiadam dzięki napisaniu tej aplikacji przyda się w przyszłości przy projektach których będę się podejmował. Im bardziej się zagłębiałem w niektóre pojęcia i technologie tym więcej nowych możliwości się pojawiało, przez co jestem wiem że jedynie poznałem jedynie mniejszą część możliwości które oferują użyte przeze mnie technologie.

Aplikacja została napisana w taki sposób ale próby jej dalszego rozwoju nie wymagały edycji obecnego kodu. Projekt może zostać rozwinięty o dodatkowe role, którym można rozdzielić istniejące funkcjonalności co wprowadziłoby podział obowiązków.

Aplikację można znaleźć pod adresem: [bialystok-parkour.rhcloud.com](http://bialystok-parkour.rhcloud.com/)

Hasło do konta administratora : admin/admin

# Literatura

1. Oracle. Witryna internetowa http://www.oracle.com/us/products/mysql/overview/index.html stan na 3.12.2015.
2. THYMELEAF Team. Witryna internetowa <http://www.thymeleaf.org/documentation.html> stan na 5.12.2015.
3. Rychlicki-Kicior K., *Java EE 6. Programowanie aplikacji WWW,* Wydawnictwo Helion, Gliwice 2010.
4. Pivotal software. Witryna internetowa <http://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/> stan na 29.11.2015.
5. Pivotal software. Witryna internetowa <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html> stan na 29.11.2015.
6. Pivotal software. Witryna internetowa <http://docs.spring.io/spring-data/jpa/docs/current/reference/html/> stan na 29.11.2015.
7. Deinum M., Serneels K., Yates C., Ladd S., Vanfleteren C., *Pro Spring MVC with Web Flow*,Wydawnictwo Apress, 2012
8. Scarioni C., *Pro Spring Security*,Wydawnictwo Apress, 2013.
9. Sawyer McFarland D., *Java Script I Jquery Nieoficjalny Podręcznik*,Wydawnictwo Helion, Gliwice 2012.
10. Fazle Rahman S., *Technologia na start Bootstrap tworzenie interfejsów stron www*, Wydawnictwo Helion, Gliwice 2015.
11. Twitter, Witryna internetowa <http://getbootstrap.com/> stan na 05.12.2015.
12. Duckett J., *HTML i CSS Zaprojektuj i zbuduj stronę www*,Wydawnictwo Helion, Gliwice 2014.
13. Red Hat. Witryna internetowa <https://www.openshift.com> stan na 10.12.2015.
14. Polskie Centrum Joomla!. Witryna internetowa <http://www.joomla.pl/> stan na 20.11.2015
15. Parkour United. Witryna internetowa <http://parkourunited.pl/> stan na 19.11.2015
16. Stowarzyszenie Klub Sportowy Movement. Witryna internetowa <http://ksmovement.pl/PL/> stan na 19.11.2015

Załącznik

