

**Politechnika Śląska**

**Wydział Automatyki, Elektroniki i Informatyki**

**Kierunek Informatyka**

##### Projekt inżynierski

Inteligentny system oznakowania pomieszczeń

Autor: Kamil Seweryn imię i nazwisko autora (czcionka 14pt, położenie: 19 cm w pionie od górnego marginesu)

Kierujący pracą: dr inż. Michał Maćkowski

Gliwice, styczeń 2016

Spis treści

[1. Wstęp 5](#_Toc471685733)

[1.1 Cel pracy 5](#_Toc471685734)

[1.2 Struktura pracy 6](#_Toc471685735)

[2. Analiza tematu 6](#_Toc471685736)

[2.1 Założenia 6](#_Toc471685738)

[2.2 Dostępne rozwiązania 7](#_Toc471685739)

[3. Wybór narzędzi 8](#_Toc471685740)

[3.1 Środowisko programistyczne 8](#_Toc471685741)

[3.2 Kontrola wersji 9](#_Toc471685742)

[3.3 Baza danych 9](#_Toc471685743)

[3.4 Technologie 10](#_Toc471685744)

[3.4.1 Java 10](#_Toc471685745)

[3.4.2 Spring 10](#_Toc471685746)

[3.4.3 Maven 11](#_Toc471685747)

[3.4.4 HTML 11](#_Toc471685748)

[3.4.5 Java Script 12](#_Toc471685749)

[3.4.6 Bootstrap 12](#_Toc471685750)

[3.4.7 AngularJS 12](#_Toc471685751)

[3.4.8 Apache Tomcat 13](#_Toc471685752)

[4. Interfejs aplikacji 14](#_Toc471685753)

[4.1 Interfejs klienta 14](#_Toc471685754)

[4.1.1 Kontakt użytkownika z interfejsem 15](#_Toc471685755)

[4.1.2 Auto dopasowanie wielkości elementów interfejsu 15](#_Toc471685756)

[4.1.3 Wizualne różnice pomiędzy różnymi interfejsami klienta 15](#_Toc471685757)

[4.1.4 Elementy interfejsu 17](#_Toc471685758)

[4.2 Interfejs edytora 18](#_Toc471685759)

[4.2.1 Logowanie 18](#_Toc471685760)

[4.2.2 Strona główna 19](#_Toc471685761)

[4.2.3 Panel nawigacji 19](#_Toc471685762)

[4.2.4 Podstrona moje wydarzenia 20](#_Toc471685763)

[4.2.5 Podstrona dodaj/edytuj wydarzenie 21](#_Toc471685764)

[4.2.6 Podstrona moje wiadomości 22](#_Toc471685765)

[4.2.7 Podstrona dodaj/edytuj wiadomość 23](#_Toc471685766)

[4.2.8 Podstrona wszystkie pokoje 24](#_Toc471685767)

[4.2.9 Podstrona dodaj/edytuj pokój 25](#_Toc471685768)

[4.2.10 Podstrona dodaj/edytuj nauczyciela 27](#_Toc471685769)

[5. Baza danych 27](#_Toc471685770)

[5.1 Schemat bazy danych 28](#_Toc471685771)

[5.2 Encja user 28](#_Toc471685772)

[5.3 Encja user\_role 29](#_Toc471685773)

[5.4 Encja teacher 29](#_Toc471685774)

[5.5 Encja teacher\_room 29](#_Toc471685775)

[5.6 Encja room 30](#_Toc471685776)

[5.7 Encja message 30](#_Toc471685777)

[5.8 Encja event 31](#_Toc471685778)

[6 Specyfikacja wewnętrzna 31](#_Toc471685779)

[6.1 Architektura 31](#_Toc471685780)

[6.2 Struktura projektu 32](#_Toc471685781)

[6.2.1 Moduł model 33](#_Toc471685782)

[6.2.2 Moduł repository 33](#_Toc471685783)

[6.2.3 Moduł service 34](#_Toc471685784)

[6.2.4 Moduł web 35](#_Toc471685785)

[6.3 Serwis REST 36](#_Toc471685786)

[6.4 Ogólny schemat przepływu danych 36](#_Toc471685787)

[7. Wdrożenie serwera aplikacji na środowisku produkcyjnym 38](#_Toc471685788)

[7.1 Baza danych 39](#_Toc471685789)

[7.2 Konfiguracja projektu 39](#_Toc471685790)

[7.3 Stworzenie pliku war 39](#_Toc471685791)

[7.4 Zainstalowanie pliku war 40](#_Toc471685792)

# 1. Wstęp

Brak wiedzy o planie zajęć jest częstym problemem w życiu każdego studenta. Niewiedza powoduje niepotrzebny stres oraz zamieszanie. Nie ułatwia tego fakt częstych zmian terminów zajęć i spotkań, nieobecności nauczycieli oraz tworzenie nowych wydarzeń odbywających się w jednym tylko terminie. Stojąc przed salą większość osób nie mamy wiedzy o prowadzonych w niej zajęciach. Papierowe plany zajęć nie sprawdzają się w sytuacji częstych zmian i zdarza się że powodują jeszcze większe zamieszanie. Korzystanie z planu zajęć pod adresem plan.polsl.pl również nie jest najwygodniejszym rozwiązaniem. Plan ten nie posiada również możliwości łatwej edycji oraz wyświetlania wiadomości. Celem pracy inżynierskiej stało się więc rozwiązanie tego problemem.

## 1.1 Cel pracy

Celem pracy jest stworzenie systemu internetowego służącego do inteligentnego oznakowania pomieszczeń. Ma przedstawiać informację o danej sali oraz odbywających się w niej zajęciach. System ma będzie miał do dyspozycji tablety usytuowane przy wejściu do każdego pokoju współpracującego z systemem. Na tablecie otwarta zostanie specjalnie w tym celu zaprojektowana strona internetowa w trybie pełnoekranowym. Strona będzie musiała działać bez żadnej pomocy i ingerencji użytkownika z zewnątrz. Korzystać będzie z serwera bazodanowego widoczne w sieci lokalnej. Komunikację z serwerem będą stanowić żądania http. Zostanie stworzony również edytor do zarządzania systemem znajdujący się pod odpowiednim adresem url również w tej samej sieci. Nauczyciele dzięki posiadaniu kont w systemie będą mogli aktualizować własne plany wyświetlane przy salach. Dostępne będzie również wysyłanie wiadomości dla użytkowników planu.

Aplikacja będzie musiała być zabezpieczona przed niepożądanymi działaniami, które mogłyby wpłynąć na błędy systemu. Jednakże musi również bez przeszkód i w szybki sposób dostarczać dane do klientów.

## 1.2 Struktura pracy

asasfasf

# 2. Analiza tematu

W rozdziale przedstawiono główne założenia podczas projektowania i implementacji aplikacji. Opisane zostały również już istniejące rozwiązania, oraz ich wady w odniesieniu do otrzymanych wymagań.



## 2.1 Założenia

Głównym celem przy projektowaniu aplikacji było rozwiązanie problemu braku informacji o zajęciach odbywających się w danej sali laboratoryjnej. Aplikacja musiała stworzyć plan uwzględniając wszystkich nauczycieli mających zajęcia w danej sali. Wygenerować przejrzysty i zrozumiały dla użytkownika interfejs przedstawiający te dane. Musiała również dynamicznie zmieniać wyświetlane informację w zależności od otrzymanych danych. Każdy z nauczycieli musiał również mieć możliwość zmieniania sposobu w jaki dane są wyświetlane przy każdej sali oraz pisania wiadomości wyświetlanych z boku planu zajęć.

Plan zajęć miał zostać wyświetlony na tabletach znajdujących się przy drzwiach do sali. Tablet został umieszczony w szklanej szafce co pozwala tylko na kontakt wzrokowy. Ponadto musiał zostać stworzony edytor w formie strony internetowej służący do edytowania zajęć i pokoi , oraz baza danych przetrzymujące wszystkie informacje.

Drugim z założeń miała być stabilność aplikacji ze względu na brak możliwości kontaktu użytkownika z tabletem wyświetlającym plan zajęć. Wszystkie błędy, chwilowe braki kontaktu z serwerem oraz inne nieprzewidziane sytuacje musiały zostać obsłużone, oraz nie mieć wpływu na wyświetlany plan zajęć. Po rozwiązaniu problemu aplikacja musiała wrócić do normalnego trybu pracy.

Kolejną ważną kwestią było bezpieczeństwo danych. Serwer aplikacji musiał przetrzymywać znaczne ilości informacji, których wprowadzenie zabiera duży okres czasu. Utracenie takich danych musiałoby wiązać się z ich ponownym wprowadzeniem co byłoby dużym problemem dla użytkowników. Ponadto aplikacja musiała również zapewnić bezpieczeństwo podczas wprowadzania danych. Dane mogły pochodzić tylko nauczycieli posiadających konto w systemie. Dane przesyłane przez użytkowników nie posiadających odpowiednich praw miały zostać zignorowane. Otrzymane informacje musiały być również sprawdzane pod kątem ich poprawności, aby wykluczy zaistnienie potencjalnych błędów spójności danych.

Ważnymi kwestiami był również wydajność, skalowalność oraz przenośność.

## 2.2 Dostępne rozwiązania

Politechnika Śląska posiada już system tworzący już plan zajęć. Jest on dostępny pod adresem url: <https://plan.polsl.pl/>. Jednak nie jest on idealnym rozwiązaniem i nie spełnia wszystkich założeń. Głównym problemem jest fakt iż strona wymaga odświeżenia po każdej aktualizacji danych. Co przy braku kontaktu z tabletem sprawia że edycja planu jest bardzo utrudniona. Nie ma również możliwości pisania wiadomości dla użytkowników. Interfejs użytkownika nie był również projektowany z myślą o wyświetlanie go na ekranie tabletu.

# 3. Wybór narzędzi

Przy wyborze narzędzi użytych w projekcie kierowano się przede wszystkim ich dostępnością i ich powszechnością. W rozdziale zostały one pokrótce opisane.

## 3.1 Środowisko programistyczne

Projekt został stworzony w środowisku InteliJ IDEA. Jest to środowisko firmy JetBrains. Można uzyskać darmową studencką licencję. Środowisko zawiera kompilator kodu Java. Obsługuje projekt zbudowane za pomocą narzędzia maven. Wspiera większość technologii używanych w projekcie. Posiada również rozbudowane funkcje debugowania kodu z możliwość dynamicznej edycji wszystkich składowych projektu.

## 3.2 Kontrola wersji

 Narzędziem do kontroli wersji został GIT. Jest to rozproszony system kontroli wersji co oznacza że każdy użytkownik posiada na swoim komputerze kopię wszystkich poprzednich wersji i może na nich pracować nawet bez połączenia z głównym serwerem. Na głównym serwerze jest natomiast gromadzona aktualna wersja tzw. głowa projektu. Głównym serwerem projektu została platforma GITHUB, ze względu na jej szerokie zastosowanie oraz dobrą renomę.

## 3.3 Baza danych

 W projekcie użyto bazy danych MySql. Jest to relacyjna baza danych. Dane trzymane są w tabelach posiadających wzajemne relacje. Wraz z serwerem Apache oraz PHP tworzy tzw. "LAMP" (Linux - Apache - MySQL - PHP) - obecnie najpopularniejszą architekturę serwerową pod małe i średnie serwisy internetowe. MySQL była tworzona głównie z myślą o wydajności, zgodności z obowiązującymi standardami zeszła na drugi plan.

## 3.4 Technologie

### 3.4.1 Java

 Obiektowy język programowania.

### 3.4.2 Spring

 Biblioteka napisana w języku java. Oparta o wzorzec projektowy wstrzykiwania zależności (ang. Dependency Injection). Zależności między komponentami zostają usunięte. Wszystkie potrzebne zależności zostają dostarczone przez środowisko co ułatwia projektowanie złożonych systemów informatycznych. W bibliotece Spring wszystkie komponenty nazywane są „Bean”, tworzone są i przetrzymywane w kontenerze aplikacji. Po stworzeniu wszystkich kontener sprawdza potrzebne im zależności i dostarcza je. Spring jest kompleksowym narzędziem do budowy aplikacji internetowych. Posiada wiele przydatnych modułów takich jak:

* Spring web – moduł obsługujący zapytania http. Pozwala tworzyć serwer aplikacji internetowych. W zależności od typu oczekiwanych danych zwraca strony HTML, lub dane w postaci JSON. Moduł oparty jest o wzorzec model, widok, kontroler. Model przetrzymuj dane aplikacji. Są to zazwyczaj tabele w bazie danych. Kontroler tworzy widok na podstawie danych z modelu. Widok jest odpowiedzialny za wizualizację danych.
* Spring security – moduł odpowiedzialny za bezpieczeństwo danych. Zabezpiecza on dostęp do zewnętrznych kontrolerów aplikacji które są najbardziej narażone na ataki. Moduł oparty jest o mechanizm sesji. Po zalogowaniu każdy z użytkowników otrzymuje unikalny numer sesji który otwiera mu dostęp do z góry zdefiniowanej ilość praw w aplikacji. Moduł pozwala stworzyć różne rodzaje użytkowników posiadających określone role i prawa w aplikacji.
* Spring data – moduł odpowiedzialny za łatwą obsługę bazy danych. Mapuje on klasy w javie na tabele w bazie danych. Pozwala na transakcje z baza danych bez znajomości języka SQL. Wszystko odbywa się za pomocą repozytoriów służących do obsługi poszczególnych obiektów reprezentowanych jako klasy w języku java. Ponadto bierze na siebie obsługę zależności danych i automatycznie dostarcza je w momencie gdy są potrzebne.
* Spring boot – moduł posiadający swój wewnętrzny serwer aplikacji (tomcat lub jboss). Ponadto ustawia on wszystkie parametry na domyślne wartości, przez co programista oszczędza wiele czasu na konfiguracji środowiska. Może skupić się na biznesowych założeniach projektu.

### 3.4.3 Maven

 Maven to narzędzie służące do budowy projektów oraz zarządzania jego zależnościami jakim mogą być biblioteki, pliki jar oraz kody źródłowe. Służy ono również do generowania z projektu plików jar i war oraz do uruchamiania testów jednostkowych.

### 3.4.4 HTML

**HTML (HyperText Markup Language)** jest językiem używanym do tworzenia stron internetowych. Jest to także język znaczników, czyli oprócz głównego tekstu dokument zawiera w sobie informację opisujące go, jego strukturę, wygląd oraz fukcjonalność.

### 3.4.5 Java Script

Skryptowy język programowania obsługiwany przez przeglądarki internetowe. Używany głownie do budowania dynamicznych stron internetowych. Pozwala edytować kod html strony w czasie jej wyświetlania. Zwykle obsługuję on wydarzenia od użytkownika, takie jak kliknięcia komponentów na stronie, naciśnięcie przycisków i różnego rodzaju inne akcje. Pozwala on również wysyłanie zapytań http pod wskazany adres bez przeładowania strony dzięki technologii AJAX (*ang. Asynchronous JavaScript and XML*).

### 3.4.6 Bootstrap

Jest to darmowa biblioteka już z gotowymi komponentami, których można użyć bezpośrednio na stronie. Mają one zaimplementowany wygląd oraz w zależności o typu, swoją funkcjonalność. Komponenty używają biblioteki jQuery korzystającej z języka java script.

### 3.4.7 AngularJS

Angular JS to framework **JavaScript.** Służy on do prostego budowania tak zwanych „**Single**-**Page Applications**”. Czyli stron które nie przeładowują się podczas korzystania z nich. AngularJS oferuje wiele przydatnych funcji które pomagają w tworzenie stron po stronie klienta jak i po stronie serwera:

* możliwość łatwego dostępu do danych w każdym miejscu strony, ze względu na to że strony korzystaj z wspólnego modelu.
* znacząco wpływ na wydajność strony
* dzięki mechanizmowi wstrzykiwania zależności, w łatwy sposób możemy zaprojektować dość złożone strony internetowe nie martwiąc się szczególnie o architekturę
* biblioteka posiada również mechanizm „bindingów” czyli bezpośredniego połączenia ze sobą pól w formularzach html z danymi zawartymi w zmiennych modelu aplikacji w języku java script
* posiada również walidację powiązanych pól
* udostępnia wiele przydatnych dyrektyw które możemy wpisywać w odpowiednie komponenty html i tym samym wpływać na ich wygląd (ng-style), widoczność(ng-show), ilość pojawień(ng-repeat), powiązanie z modelem(ng-model), wybór odpowiedniego kontrolera (ng-controller) oraz wiele wiele innych

### 3.4.8 Apache Tomcat

Jest to kontener dla servletów. Pozwala na uruchamianie aplikacji internetowych. Cechuje się dużą wydajnością oraz możliwością konfiguracji wielu aspektów jak prawa użytkowników, porty, zabezpieczanie, szyfrowanie itd.

# 4. Interfejs aplikacji

W niniejszym rozdziale zostanie przedstawiony wygląd i budowa interfejsów użytkownika. Aplikacja posiada interfejs klienta, na którym wyświetlany jest plan zajęć, oraz interfejs edytora, służący do odczytu, dodawania i edycji danych przez uprawnionego nauczyciela. Przedstawione również zostaną sposoby obsługi interfejsów w celu uzyskania lub dodania określonych danych w aplikacji.

## 4.1 Interfejs klienta

Jest to strona internetowa napisana przy użyciu języka HTML. Wykorzystywana jest również biblioteka AngularJS, dzięki jej możliwością dynamicznego zmieniania elementów widoku HTML, oraz stylów CSS. Interfejs jest generowany dynamicznie na podstawie danych uzyskanych z głównego serwera aplikacji. Uzyskiwanie tych danych odbywa się poprzez cykliczne wysyłanie zapytań korzystając z funkcji AngularJS. Przy każdorazowej zmianie danych widok również ulega zmianom.

### 4.1.1 Kontakt użytkownika z interfejsem

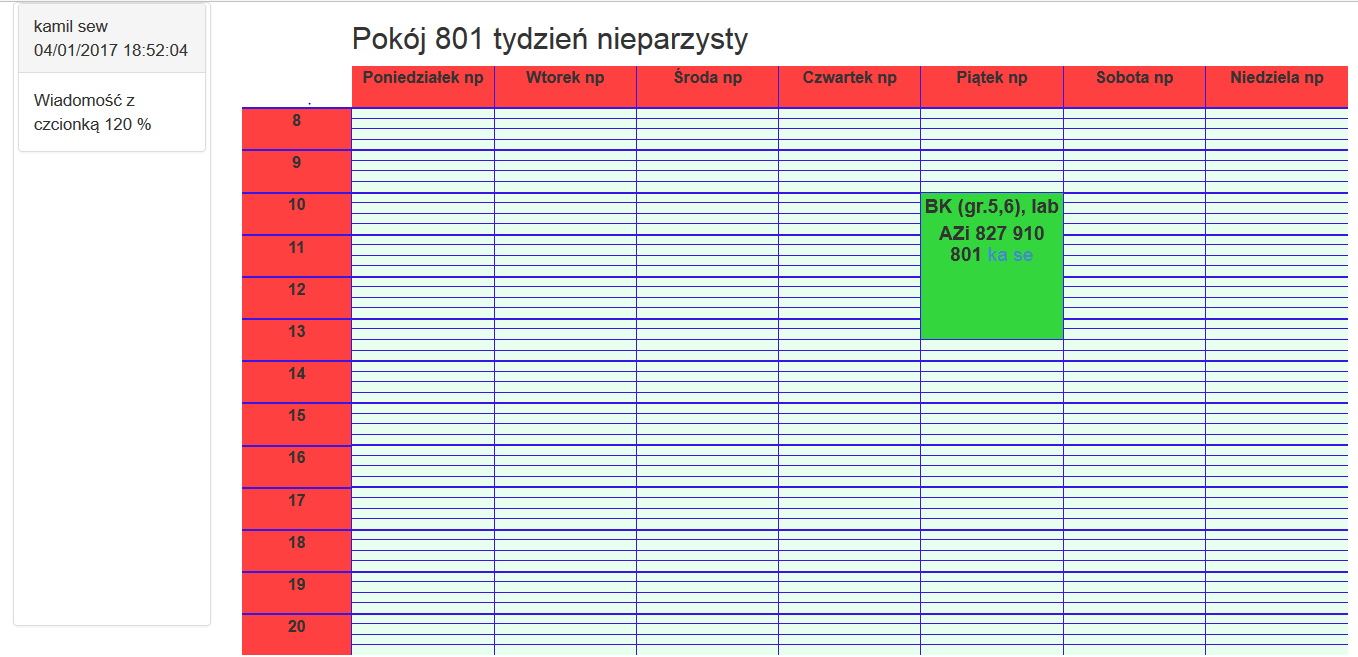
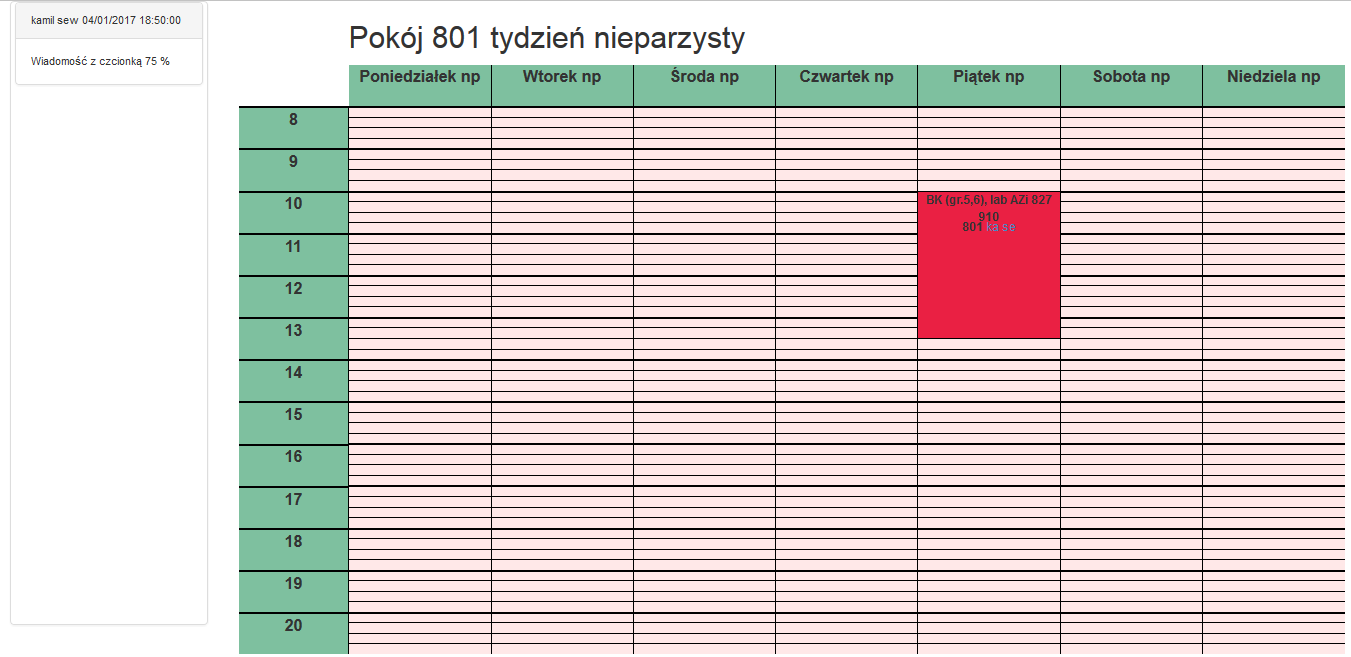
Wymaganiami projektu było aby użytkownik nie miał fizycznego kontaktu z interfejsem klienta, ponieważ tablet na którym wyświetlany jest plan znajduje się w szklanej szafce. Aby sprostać wymaganiom interfejs został zaimplementowany tak, że do poprawnej pracy nie wymaga żadnej ingerencji ze strony użytkownika. Wystarczy otworzyć w przeglądarce odpowiedni adres url i strona zacznie sama obsługiwać wszystkie zdarzenia.

### 4.1.2 Auto dopasowanie wielkości elementów interfejsu

Wszystkie elementy widoku dobierają swoje wymiary na podstawie procentowej wartości wymiarów komponentu rodzica. Nie posiadają sztywno określonych wysokości oraz szerokości w pikselach. Również minimalne i maksymalne wartości nie są określone. Rozwiązanie to na ogół nie jest stosowane w projektowaniu stron internetowych ze względu na różnice w rozmiarach dzisiejszych monitorów i fakt że twórcy zazwyczaj chcą aby układ ich interfejsu był jednakowy na wszystkich rodzajach sprzętu. Ze względu jednak na fakt, że w założeniach interfejs klienta będzie wyświetlany tylko na wąskiej liczbie tabletów zdecydowano się na rozwiązanie oparte na wielkościach procentowych aby lepiej wykorzystać całe dostępne miejsce na ekranie ponieważ interfejs również ma nie posiadać możliwości przewijania widoku

### 4.1.3 Wizualne różnice pomiędzy różnymi interfejsami klienta

Widok ma możliwość kilku wizualnych zmian zadeklarowanych przez użytkownika w celu lepszego przedstawienia danych. Aplikowane są automatycznie po otrzymaniu nowych danych z serwera.



1 – wielkość czcionki wiadomości w celu lepszego rozmieszczenia tekstu.

2 – kolor szczegółów planu zajęć. Możliwa jest zmiana koloru kolumn z nazwami dni oraz wierszy zawierających godziny.

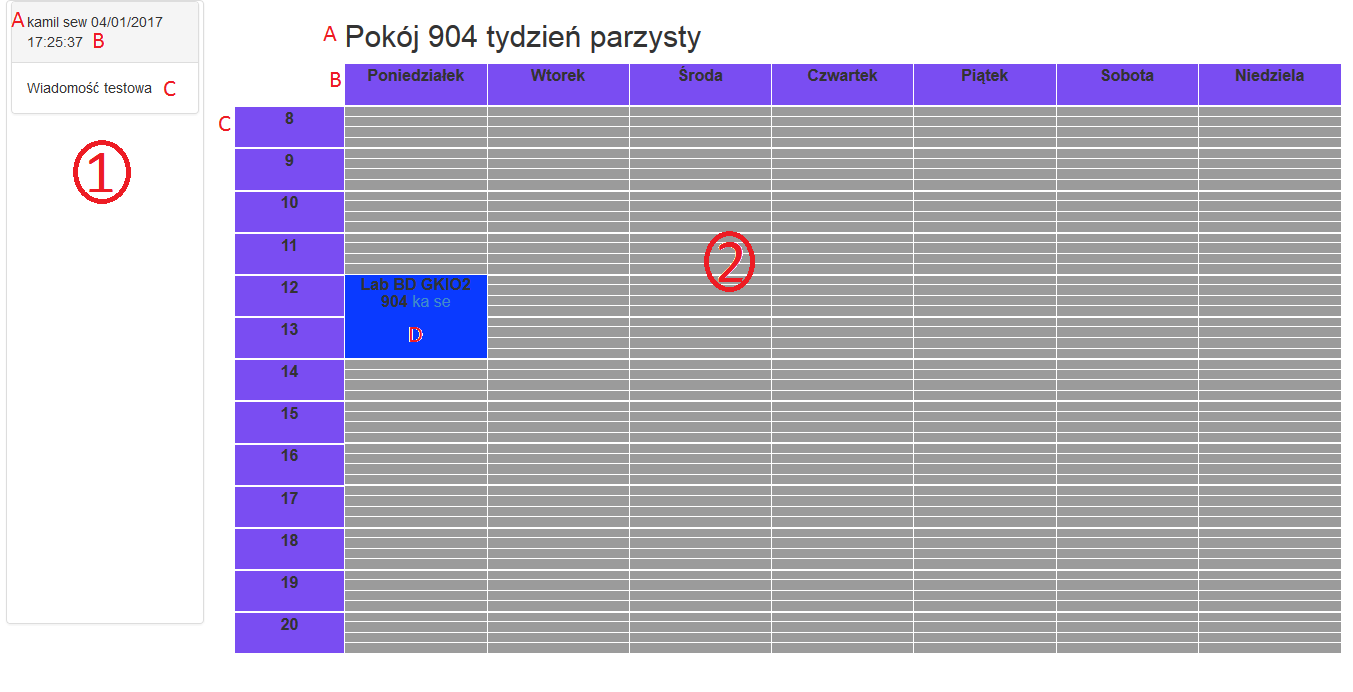
3 – kolor tła planu zajęć.

4 – kolor obramowania elementów planu zajęć

5 – kolor tła wydarzeń

6 – wielkość czcionki wydarzeń

### 4.1.4 Elementy interfejsu

****

1 – panel wiadomości. Są to wiadomości napisane przez nauczycieli. Wyświetlane są w kolejności chronologicznej (najmłodsze u góry panelu).

Wiadomości składają się z:

1. Nazwy adresata
2. Czasu zamieszczenia wiadomości
3. Treści

2 – plan zajęć. Są tu wyświetlane odbywające się zajęcia.

Panel składa się z kilku komponentów:

1. Tytuł – zawarta jest w nim informacja o tym czy wyświetlany plan należy do sali czy nauczyciela oraz o tym czy jest to plan dla tygodnia parzystego czy nieparzystego
2. Kolumny dni tygodnia – każda kolumna zawiera listę zajęć dla podanego dnia. Prefix „np” w nazwie dnia oznacza tydzień nieparzysty.
3. Wiersze godzin – wizualizują początek i koniec zajęć.
4. Wydarzenie - komponent z informacjami o wydarzeniu. Zawiera opis wydarzenia, nauczyciela prowadzącego.

## 4.2 Interfejs edytora

Strona HTML posiadająca komponenty z biblioteki Bootstrap. Są one używane, aby uatrakcyjnić widok oraz dodać pewne złożone mechaniki. Obsługą strony zajmuje się AngularJS. Do każdej z podstron przypisany jest odpowiedni kontroler.

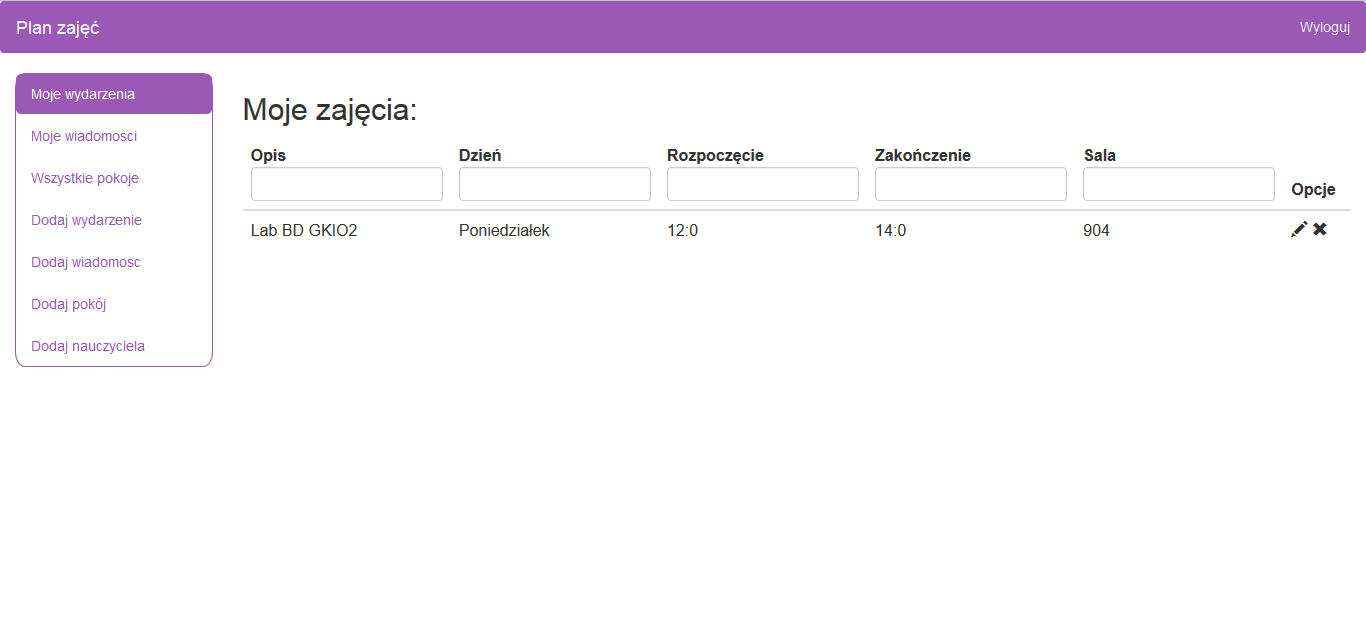
### 4.2.1 Logowanie



Dostęp do strony edytora wymaga podania nazwy użytkownika oraz hasła do konta w systemie.

Strona logowania wygląda następująco.

### 4.2.2 Strona główna

Strona główna składa się zasadniczo z trzech elementów. Z komponentu navbar zawierającego nazwę strony oraz przycisk z możliwością wylogowania z konta. Komponentu nawigacji po podstronach pozwalającego na poruszanie się i korzystanie z funkcjonalności oferowanej przez edytor. Panelu wyświetlającego podstrony w zależności od wybranych opcji w panelu nawigacji. Strona główna w czasie pracy nie zostaje przeładowana, podmieniane zostają tylko podstrony w odpowiednim panelu.

### 4.2.3 Panel nawigacji



Panel nawigacji służy do poruszania się po stronie. Zawiera następujące odnośniki:

1 – moje wydarzenia. Tabela wyświetlające wszystkie wydarzenia przypisane do użytkownika.

2 – moje wiadomości. Tabela zawierająca informację o wszystkich stworzonych przez użytkownika wiadomościach. Wyświetlane są chronologicznie (najmłodsze wydarzenie znajduje się na górze tabeli).

3 – wszystkie pokoje. Tabela wyświetlająca wszystkie dostępne pokoje wraz z ich właścicielami.

4 – dodaj wydarzenie. Formularz służący do dodawania nowych wydarzeń.

5 – dodaj wiadomość. Edytor nowych wiadomości.

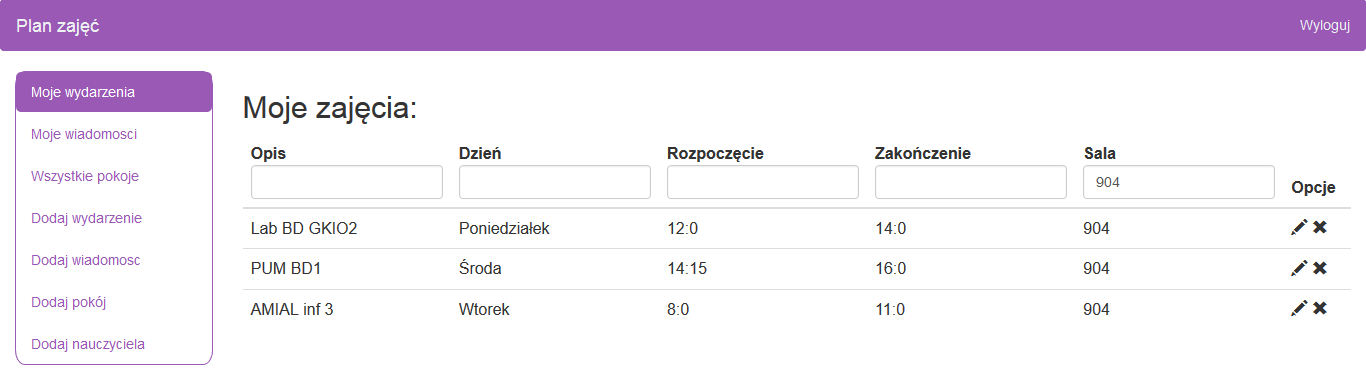
6 – dodaj pokój. Formularz służący do stworzenia nowego pokoju.

7 – dodaj nauczyciela. Podstrona służąca do dodawania nowych nauczycieli.

### 4.2.4 Podstrona moje wydarzenia



Podstrona składa się z jednej dużej tabeli posiadającej sześć kolumn. Pod tytułem każdej kolumny znajduje się pole tekstowe umożliwiające filtrowanie wydarzeń. Dzieje się to bez potrzeby przeładowania strony

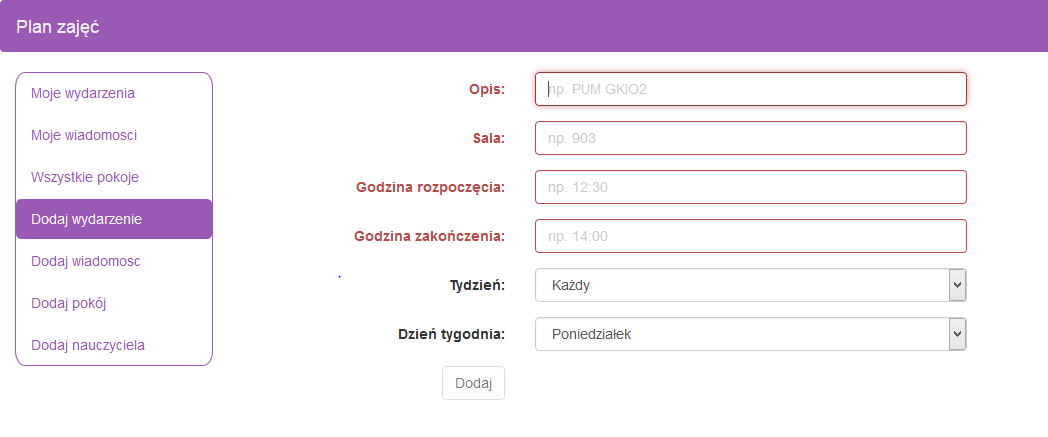


Ostatnia kolumna zawiera dwa przyciski:

- służy do edycji wybranego wydarzenia i przekierowuje na odpowiednią stronę, na której użytkownik może tego dokonać.

- służy do usunięcia wybranego wydarzenia. Po pomyślnym usunięciu dane w tabeli zostaną przeładowane.

### 4.2.5 Podstrona dodaj/edytuj wydarzenie



Składa się z formularza umożliwiającego dodanie lub edycję wydarzenia. Edytowanie wydarzenia jest możliwe tylko poprzez przycisk na stronie ,,Moje wydarzenia” natomiast dodawanie poprzez odnośnik w panelu nawigacji. Po lewej stronie formularza znajdują się opisy pól. Natomiast bezpośrednio po lewej ich stronie pola w które należy wprowadzić dane. Nie poprawny typ lub zakres danych jest sygnalizowany czerwonym podświetleniem tytułu jak i pola z danymi.

Krótki opis pól oraz sposobu ich walidacji:

1 – opis. Pole zawierające pełną nazwę lub skrót odbywających się zajęć. Może również zawierać nazwę grupy studentów. Pole nie może być puste. Może zawierać znaki specjalne. Jego długość jest ograniczona do 50 znaków.

2 – sala. Numer sali przy której ma być wyświetlana tworzona wiadomość. Numer musi być z przedziału 0-999. Sala o podanym numerze musi już istnieć w bazie danych.

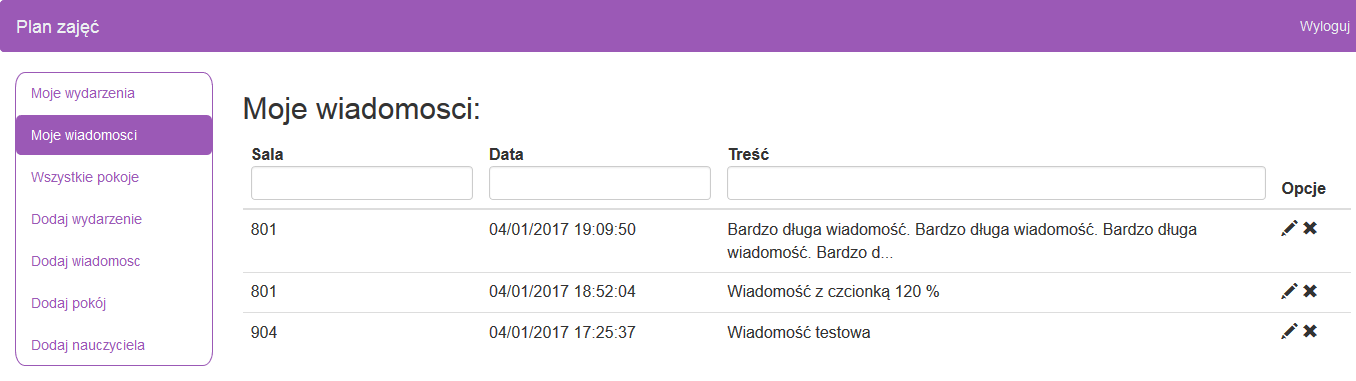
3 – godzina rozpoczęcia. Jest to czas w którym rozpoczynają się zajęcia. Poprawne formaty danych wyglądają następująco g:m gg:m g:mm lub gg:mm gdzie g oznacza cyfrę godzin a m cyfrę minut. Sprawdzany również jest poprawność godziny.

4 – godzina zakończenia. Jest to czas zakończenia wydarzenia. Pole korzysta z walidacji poprzedniego pola. Ponadto jednak sprawdza czy zakończenie nie występuje wcześniej niż rozpoczęcie.

5 – tydzień. Zajęcia mogą odbywać się w tygodnie parzyste lub nieparzysty. Nazwę tygodnia należy wybrać z rozwijane listy.

6 – dzień. Pole z nazwą dnia tygodnia w który odbywają się zajęcia. Nazwę dnia tygodnia należy wybrać z rozwijanej listy.

### 4.2.6 Podstrona moje wiadomości



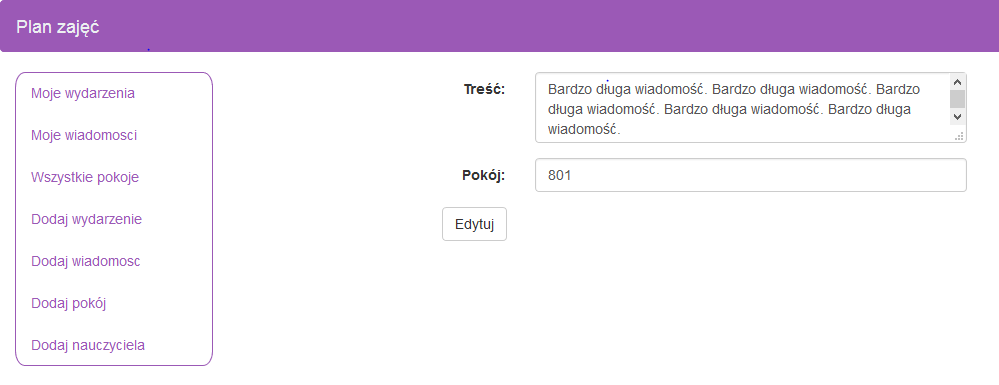
Podstrona wyświetla wszystkie napisane przez użytkownika wiadomości ze wszystkich pokoi. Wyświetla je w formie tabelki. Tak jak w tabeli na stronie moje wydarzenia użytkownik ma możliwość filtrowania wyświetlanych danych. Zawartość kolumny z treścią wiadomości została ze względów stylistycznych ograniczona do 80 znaków (treść wiadomości może zawierać do 128 znaków). Reszta wiadomości zostaje zastąpiona przez „…”.

Ostatnia kolumna zawiera przyciski funkcyjne:

 - służy do edycji wybranej wiadomości i przekierowuje na odpowiednią stronę, na której użytkownik może tego dokonać.

- służy do usunięcia wybranej wiadomości. Po pomyślnym usunięciu dane w tabeli zostaną przeładowane. Użytkownik zostanie poinformowany w razie jakichkolwiek problemów pojawiających się w czasie wykonywania tej czynność.

### 4.2.7 Podstrona dodaj/edytuj wiadomość



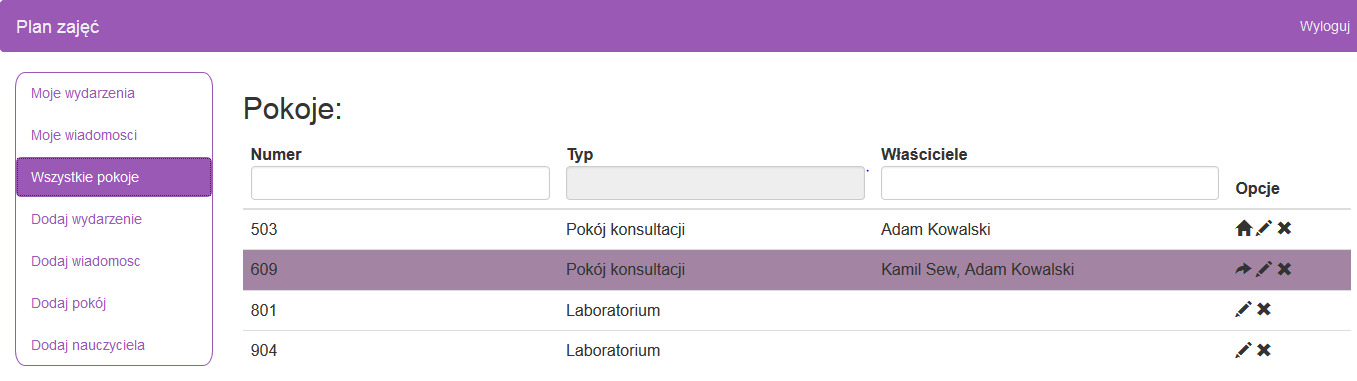
Na podstronie mamy możliwość edycji lub dodania nowej wiadomości. Edytować wiadomość możemy poprzez przycisk funkcyjny na stronie ,,Moje wiadomości” natomiast dodać nową poprzez odnośnik w panelu nawigacji. Walidacja odbywa się na podobnej zasadzie co na innych podstronach aplikacji.

Podstrona zawiera formularz z następującymi polami:

1 – treść. Jest to treść wyświetlanej wiadomości. Pole jest ograniczone do 128 znaków.

2 – sala. Numer sali przy której ma być wyświetlana tworzona wiadomość. Numer musi być z przedziału 0-999. Sala o podanym numerze musi już istnieć w bazie danych.

### 4.2.8 Podstrona wszystkie pokoje



Podstrona posiada tabelę wyświetlającą informację o wszystkich istniejących pokojach w bazie danych. Tak jak inne posiada możliwość filtrowania wyników za pomocą pól znajdujących się pod tytułami poszczególnych kolumn. Kolumna właściciele zawiera imiona i nazwiska osób do których należy sala. Kolorem  zostały zaznaczone pokoje których właścicielem jest użytkownik.

Czwarta kolumna zawiera w sobie przyciski dostępnych opcji:

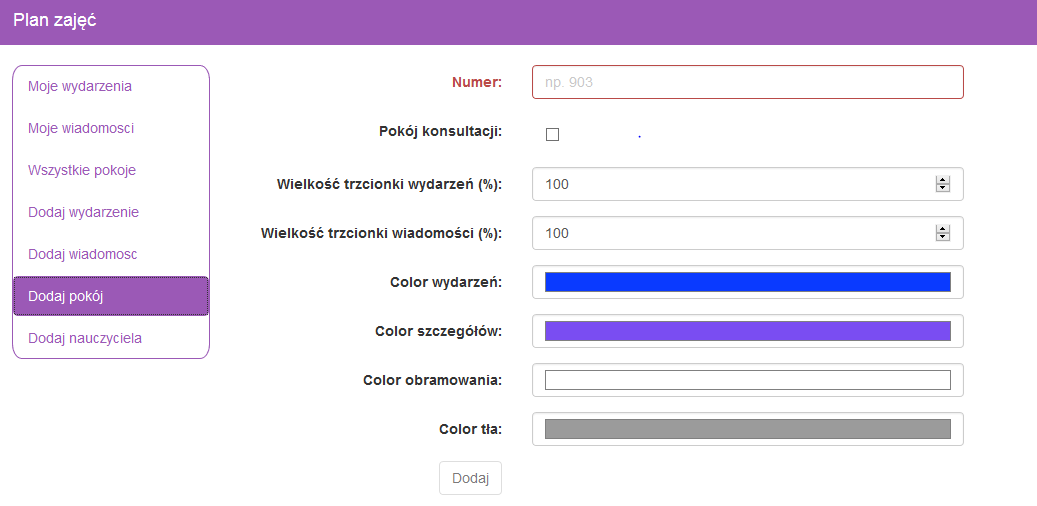
 - przypisuje użytkownika do pokoju. Tylko pokój konsultacji może posiadać właściciela. Przypisanie do laboratorium nie jest dozwolone.

- wypisuje użytkownika z pokoju. Akcja jest dozwolona jedynie gdy użytkownik był wcześniej przypisany do tego pokoju.

 - pozwala edytować pokój i przekierowuje na odpowiednią stronę, na której użytkownik może tego dokonać.

- pozwala usunąć zaznaczony pokój. Po pomyślnym usunięciu dane w tabeli zostaną przeładowane. Operacja ta usuwa również wszystkie wiadomości i wydarzenia należące do tego pokoju. Użytkownik zostanie poinformowany w razie jakichkolwiek problemów pojawiających się w czasie wykonywania tej czynność.

### 4.2.9 Podstrona dodaj/edytuj pokój



Podstrona daje nam możliwość edycji lub dodania nowego pokoju do systemu. Walidacja działa podobnie jak w innych formularzach na stronie. Formularz składa się z ośmiu pól. Żadne nie może być puste.

Pola znajdujące się na podstronie to:

1 – numer. Numer nowo tworzonej sali. Numer musi być z przedziału 0-999. Sala o podanym numerze nie może już istnieć w bazie danych. Podczas edycji już istniejącej sali edycja tego pola jest zablokowana.

2 – pokój konsultacji. Pole z wartością prawda/fałsz. Zaznaczenie tego pola powoduje że sala staje się pokojem konsultacji. Nauczyciele mogą się do niej przypisywać. Plan zajęć tej sali pokazuje wszystkie zajęcia przypisanych do niej nauczycieli. Nie pokazuje natomiast zajęć odbywających się w tej sali jeżeli należą one nie do właścicieli sali. Aby podczas edycji zmienić wartość tego pola na prawdziwą, pokój nie może posiadać żadnych wydarzeń. Pokój konsultacji w sekcji wiadomości pokazuje również wszystkie wiadomości jej właścicieli z innych sal.

3 – wielkość czcionki wydarzeń. Procentowa wartość wielkości czcionki wyświetlającej wydarzenia. Wartość musi należeć do przedziału 10 – 500. Jest to komponent 2D w podrozdziale Elementy Interfejsu.

4 – wielkość czcionki wiadomości. Procentowa wartość wielkości czcionki wyświetlającej wiadomości. Wartość musi należeć do przedziału 10 – 500. Są to komponenty 1A, 1B, 1C w podrozdziale Elementy Interfejsu.

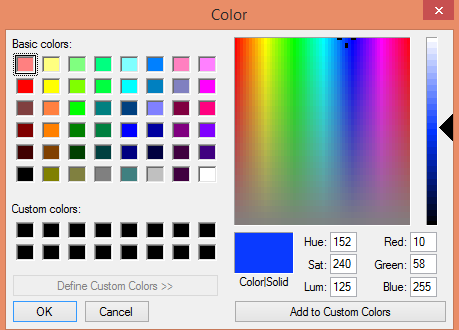
5 – kolor wydarzeń. Komponent do zmiany koloru tła wydarzeń wyświetlanych na planie dla tej sali. Jest to komponent 2D w podrozdziale Elementy Interfejsu.

6 – kolor szczegółów. Komponent do zmiany koloru tła dla pól z nazwami dni kolumn oraz godzin z lewej strony planu, wyświetlanych na planie dla tej sali. Są to komponenty 1B, 1C w podrozdziale Elementy Interfejsu.

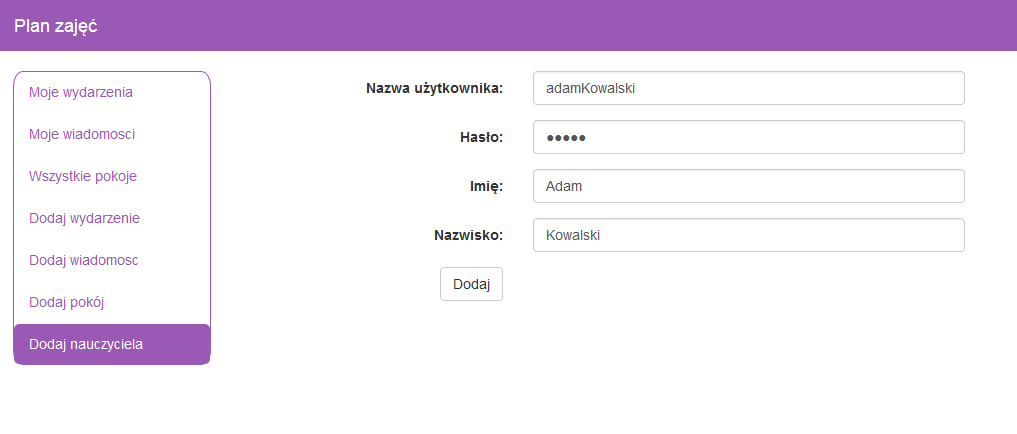
7 – kolor obramowania. Komponent do zmiany koloru obramowania pomiędzy wszystkimi elementami wyświetlanymi na planie dla tej sali.

8 – kolor tła. Komponent do zmiany kolory tła komponentu w którym wyświetlane są wydarzenia dla tej sali.

Interfejs dla zmiany wartości pól 5, 6, 7 i 8 wygląda inaczej ze względu na rodzaj systemu operacyjnego i przeglądarki internetowej. Obrazek poniżej przedstawia interfejs dla systemu Windows 8.1 i przeglądarki Firefox.



### 4.2.10 Podstrona dodaj/edytuj nauczyciela



Podstrona daje nam możliwość edycji lub dodania nowego nauczyciela do systemu. Walidacja działa podobnie jak w innych formularzach na stronie. Formularz składa się z czterech pól. Żadne nie może być puste.

Formularz zawiera następujące pola:

1 – nazwa użytkownika. Nazwa użytkownika w systemie Nauczyciel będzie musiał podać tą nazwę każdorazowo logując się do systemu. Nazwa musi być unikalna, nie może istnieć w systemie dwóch użytkowników z tą samą nazwą. Pole musi zawierać od 5 do 20 znaków.

2 – hasło. Hasło używane podczas logowania do systemu. Pole musi zawierać od 5 do 20 znaków.

3 – imię. Imię nauczyciela. Pole musi zawierać od 3 do 50 znaków.

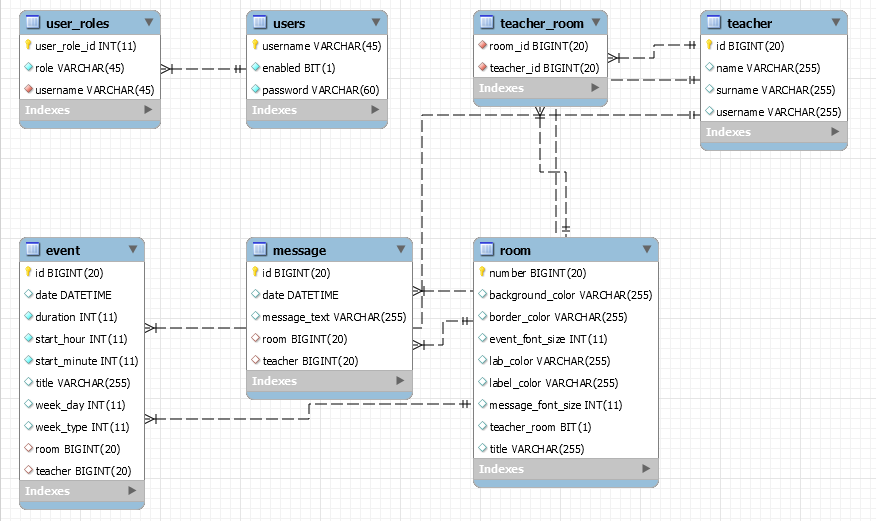
4 – nazwisko. Nazwisko nauczyciela. Pole musi zawierać od 3 do 50 znaków.

# 5. Baza danych

W rozdziale przedstawione zostaną struktura bazy danych, oraz opis każdej encji. W opisie zostaną po krótce przedstawione przeznaczenie, pola oraz relację tabel. W projekcie użyto istniejącej już bazy danych MySql. Do prostszego korzystania z jej funkcjonalności zastosowano narzędzie MySQL Workbench. Posiadało one wszystkie niezbędne funkcje.

## 5.1 Schemat bazy danych

Schemat został wygenerowany za pomocą jednej z funkcji MySQL Workbench.



## 5.2 Encja user

Tabela jest wykorzystywana przez bibliotekę Spring Security i dlatego posiada swoje ograniczenia. Nie może np. posiadać relacji do innych tabel w systemie. Zawiera informację o użytkowniku.

Posiada kolumny:

* Username – nazwa użytkownika. Jest to zarazem klucz główny tabeli i musi być unikatowy.
* Enabled – flaga symbolizująca status użytkownika.
* Password – hasło.

## 5.3 Encja user\_role

Używana jest również przez bibliotekę Spring Security. Określa uprawnienia każdego z użytkowników w systemie. Prawa te definiowane są w konfiguracji podczas dołączania biblioteki do projektu.

Posiada kolumny:

* User\_role\_id – klucz główny tabeli, musi być unikatowy.
* Role – nazwa roli użytkownika w systemie.
* Username – nazwa użytkownika posiadającego daną rolę. Jest to klucz obcy do rekordów tabeli users.

## 5.4 Encja teacher

Zawiera informację o każdym użytkowniku w systemie. Ze względu na to że nie jest używana przez bibliotekę Spring Security nie posiada takich ograniczeń jak tabela users. Są one połączone relacją jeden do jeden. Jednakże jest ona zastosowana jedynie w warstwie biznesowej i nie ma jej w bazie danych.

Posiada kolumny:

* Id – klucz główny tabeli, musi być unikatowy.
* Name – imię nauczyciela.
* Surname – nazwisko nauczyciela.
* Username – nazwa użytkownika. Jest to relacja do rekordów tabeli users.

## 5.5 Encja teacher\_room

Jest to tabela umożliwiająca relację wiele do wielu między tabelami teacher i room.

Posiada kolumny:

* Room\_id – jest to klucz obcy do rekordów tabeli room.
* Teacher\_id – jest to klucz obcy do rekordów tabeli teacher.

## 5.6 Encja room

Zawiera informację o każdym pokoju w systemie. Przetrzymuje dane o tym w jaki sposób wyświetlać plan zajęć dla tego pokoju. Przetrzymuje również wszystkie wydarzenia oraz wiadomości dla pokoju.

Posiada kolumny:

* Numer – klucz główny tabeli, musi być unikatowy.
* Background\_color – kolor tła planu zajęć.
* Border\_color – kolor obramowania elementów na planie zajęć.
* Event\_font\_size – procentowa wielkość czcionki wydarzeń.
* Lab\_color – kolor tła wydarzeń.
* Message\_font\_size - procentowa wielkość czcionki wiadomości
* Teacher\_room – wartość prawda/ fałsz określająca czy dany pokój jest pokojem konsultacji.
* Title – tytuł pokoju.

## 5.7 Encja message

Zawiera informację o wiadomościach wyświetlanych na planach zajęć.

Posiada kolumny:

* Id – klucz główny tabeli, musi być unikatowy.
* Date – data ostatniej modyfikacji wiadomości.
* Message\_text – treść wiadomości.
* Room – numer pokoju do którego przypisana jest wiadomość. Jest to klucz obcy do rekordów tabeli room.
* Teacher – identyfikator nauczyciela, który napisał wiadomość. Jest to klucz obcy do rekordów tabeli teacher.

## 5.8 Encja event

Zawiera rekordy z informacjami o wydarzeniach w bazie danych.

Posiada kolumny:

* Id – klucz główny tabeli, musi być unikatowy.
* Date – data ostatniej modyfikacji wydarzenia.
* Duration – czas trwania wydarzenia.
* Start\_hour – Godzina rozpoczęcia.
* Start\_minute – minuta rozpoczęcia wydarzenia.
* Title – tytuł wydarzenia.
* Week\_day – dzień tygodnia w którym odbywa się wydarzenie.
* Week\_type – rodzaj tygodnia w którym odbywa się wydarzenie.
* Room – numer pokoju do którego przypisana jest wiadomość. Jest to klucz obcy do rekordów tabeli room.
* Teacher – identyfikator nauczyciela, który napisał wiadomość. Jest to klucz obcy do rekordów tabeli teacher.

# 6 Specyfikacja wewnętrzna

Rozdział zawiera informację o budowie wewnętrznej aplikacji. Opis poszczególnych modułów zawiera ich przeznaczenie, cele wewnętrzną budowę oraz strukturę pakietów. Opisane zostały również najważniejsze algorytmy i mechanizmy.

## 6.1 Architektura

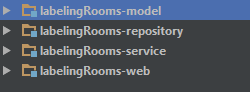
Projekt tworzony jest w architekturze MVC (model-view-controller). Jest to popularny wzorzec projektowy wykorzystywany w większości aplikacji webowych. Opiera się on na separacji elementów interfejsu (widoku) od danych i logiki wewnętrznej aplikacji. Separację uzyskuje poprzez podzielenie aplikacji na warstwy. W niezmodyfikowanej architekturze MVC wyszczególnione są następujące warstwy:

* Model – warstwa danych. Przetrzymuje on wszystkie obiekty służące do gromadzenia danych. Są one obiektami mapującymi tabele w bazie danych wraz z ich relacjami na obiekty klasy java. Do tej warstwy należą również wszystkie obiekty służące do komunikacji aplikacji z bazą danych, a więc obiekty typu DAO (data access object). Warstwa ta powinna również zawierać całą logikę biznesową aplikacji, czyli sposoby i mechanizmy jej działania.
* View – warstwa widoku. Odpowiada ona za wizualizację danych. Są to różnego rodzaju interfejsy użytkownika. Służą one do komunikacji użytkownika z aplikacją i danymi. Odpowiadają również za wyświetlanie komunikatów.
* Controller- warstwa zarządzająca. Posiada obiekty które zarządzają zdarzeniami otrzymywanymi z widoku. Delegują zdarzenia do odpowiednich części modelu. Zbiera otrzymane z modelu dane i na ich podstawie generuje widok dla użytkownika. Odpowiada on również za weryfikację otrzymanych danych, obsługę błędów zaistniałych w aplikacji i zarządzanie przepływem żądań.

Projekt wykorzystuje zmodyfikowaną wersję architektury MVC opartą o bibliotekę Spring. Dzieli ona warstwę danych na klika modułów co znacznie ją odciąża oraz pozwala lepiej sterować przepływem danych.

## 6.2 Struktura projektu

Projekt został podzielony na cztery moduły, z których każdy pełni inną rolę. Kryterium podziału była przynależność obiektów do odpowiednich warstw architektury. Podział został dokonany aby ułatwić zobrazowanie przepływu. Pozwala on łatwiej odnaleźć się w aplikacji. Dzięki podziałowi uzyskujemy również większą separację warstw.



### 6.2.1 Moduł model

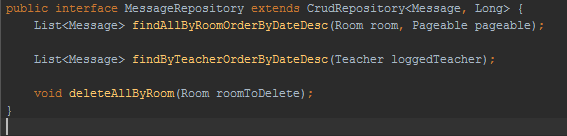
Moduł zawiera tylko modele danych wykorzystywanych w aplikacji. Są to obiekty java mapujące tabelę w bazie danych, obiekty wykorzystywane na wyższych poziomach aplikacji oraz wyjątki rzucane przez inne moduły. Moduł ten nie zawiera żadnej logiki biznesowej, nie posiada klas operujących na danych i obiektach. Jest to pierwsza z modyfikacji wzorca MVC. Moduł został podzielony na trzy pakiety:

* model – zawierający wyjątki, obiekty transportowe dla innych klas oraz obiektu typu enum przetrzymujące w sobie jeden z zadeklarowanych stanów.
* DBO – pakiet składający się z obiektów mapujących tabele z bazy danych (data base object). Każda tabela zawiera taki obiekt. Posiadają one również w sobie wzajemne relację do innych obiektów.
* DTO – pakiet zawierający obiekty transportujące dane w wyższych warstwa aplikacji oraz wysyłanych do widoku. Każdy obiekt DTO oparty jest o obiekt DBO. Celem tworzenia dodatkowy obiektów jest fakt iż można na nich mniejszym kosztem wykonywać złożone operację poprzez logikę biznesową. Separują one relację pomiędzy danymi w pakiecie DBO co pozwala je łatwiej wysyłać protokołem http do interfejsu użytkownika. Sprawiają one również że użytkownik nie ma bezpośredniego wglądu do wewnętrznych relacji między danymi co wpływa na bezpieczeństwo aplikacji.

### 6.2.2 Moduł repository

W module znajdują się obiekty służące do wysyłania zapytań i żądań do bazy danych. Mapują one otrzymane dane z bazy na obiekty w pakiecie DBO modelu. Odpowiadają za tworzenie relacji obiektów. Na tym poziomie została użyta biblioteka Spring Data Jpa która niebywale ułatwia tworzenie repozytoriów do bazy danych. Wszystkie obiekty DAO zostają wygenerowane przez wewnętrzne mechanizmy biblioteki. Programista musi jedynie stworzyć interfejs opisujący funkcję które ma spełniać dany obiekt. Nie wymagana jest nawet implementacja zadeklarowanych metod.

Obrazek poniżej przedstawia przykładowe repozytorium dla obiektów wiadomości:



Oprócz zdeklarowanych metod interfejs dziedziczy również po CrudRepository metody:

* Create - tworzenie
* Read - odczytywanie
* Update - modyfikowanie
* Delete – usuwanie

Żadna z zadeklarowanych oraz odziedziczonych metod nie jest w żadnym miejscu implementowana. Implementację interfejsów dostarcza użyta biblioteka.

### 6.2.3 Moduł service

Moduł zawiera cała szeroko pojętą logikę biznesową aplikacji. Odpowiada za poprawne działanie wszystkich operacji, żądań i zapytań. Generuje on dane na podstawie otrzymanych informacji. Do komunikacji z bazą danych wykorzystuje moduł repository, natomiast to on obrabia dane które są pobierane lub wysyłane do bazy danych i deleguje je wyżej. Sprawdza on również dane otrzymane pod kontem poprawności i spójności za co w wypadku architektury MVC powinno być zastosowane w kontrolerze. Jednakże ze względu na fakt dużego wykorzystania modułu repozytorium w procesie walidacji danych, zdecydowano się przenieść tą funkcjonalność do modułu serwis. Moduł zawiera trzy pakiety z których każdy pełni inna rolę w procesie przypływu danych:

* Service – zawiera klasy posiadające cały logikę biznesową aplikacji. Zarządzają one resztą pakietów w module oraz niższymi warstwami.
* Mapper – zawiera klasy przetwarzające obiekty DBO i tworzące na ich podstawie obiekty DTO. Proces ten może również zostać przeprowadzony w drugą stronę. Obiekty które mają zostać przetworzone muszą zostać dostarczone z zewnątrz.
* Validator – obiekty wykorzystywane do sprawdzania poprawności danych w obiektach DTO. W razie nie spełnienia któreś z reguł generują wyjątek który jest obsługiwany w kontrolerze.

### 6.2.4 Moduł web

Moduł zawiera wszystkie komponenty sieciowe aplikacji. Służy jego głównym zadaniem jest generowanie widoku. W tym przypadku jest to strona HTML. Wizualizuje on dane oraz zapewnia możliwość interakcji z użytkownikiem. Odbiera on wszystkie żądania od użytkownika i deleguje je do niższych warstw aplikacji. Obsługuje wszystkie błędy aplikacji z niższych warstw i generuje na nich podstawie wiadomości wysyłane do interfejsu. Jest to najbardziej złożony z modułów aplikacji ze względu na użycie w nim wielu technologii oraz synchronizację i kontakt pomiędzy nimi. Posiada on również konfigurację wszystkich bibliotek, wszystkie skrypt, inne zasoby aplikacji jak strony HTML. Ze względu na jog złożoność został podzielony na cztery pakietów:

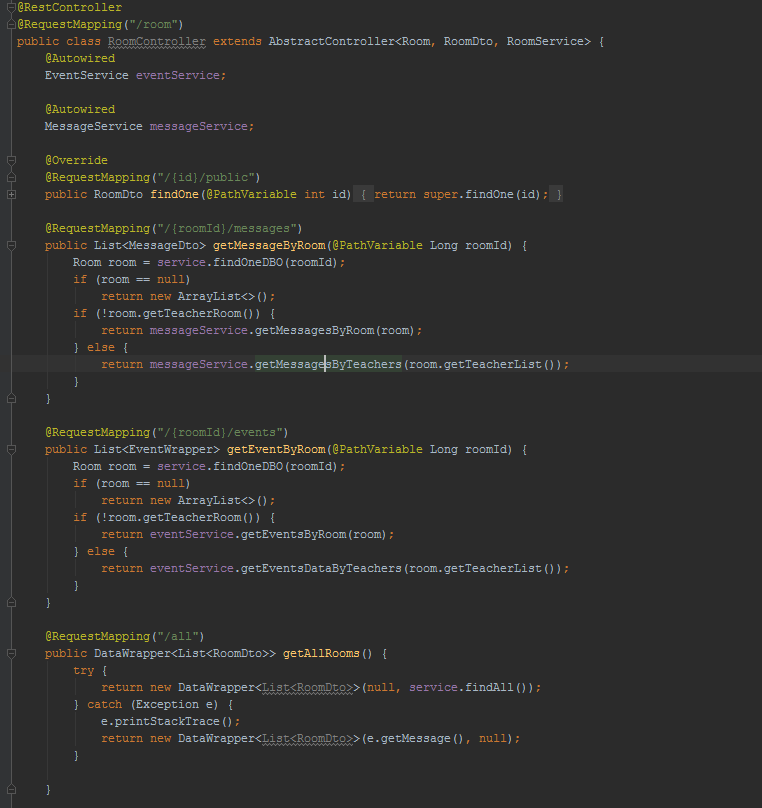
* Config – zawiera konfigurację wszystkich bibliotek w aplikacji. Określa zasady bezpieczeństwa, prawa użytkowników. Zawiera informację o bazie danych i sposobie komunikacji z nią.
* Js – przetrzymuje wszystkie plik javascript. Jest miejscem przechowywania wszystkich bibliotek odpowiadających za wygląd i obsługę interfejsu takich jak Bootstrap, AngularJS. Posiada kontrolery stron html oraz skrypty odpowiedzialne za wysyłanie żądań do serwera aplikacji. Struktura tego pakietu odpowiada prawą użytkowników na różnym poziomach bezpieczeństwa aplikacji.
* Templates – pakiet widoków. Zawarte w nim są wszystkie strony HTML. Nie zawiera żadnej logiki aplikacji służy jedynie do gromadzenia zasobów wykorzystywanych przez inne pakiety modułu.
* Controller – kontrolery aplikacji. Otrzymują żądania http i decydują o reakcji aplikacji. Jest to najbardziej zewnętrzna warstwa serwera. W odpowiedzi na odpowiednie żądania zwraca widok wraz z wszystkimi danymi i zasobami. Obsługuje również wszystkie żądania REST. Każdy z obiektów DBO posiada własny kontroler, który jest odpowiedzialny za obsługę żądań dotyczących tego obiektów.

## 6.3 Serwis REST

Aplikacja w dużej mierze jest serwerem i klientem REST. Serwer napisany jest w języku java, obsługiwany jest prze wszystkie cztery moduły, odpowiada on na zapytania klienta. Klientem natomiast jest strona internetowa, która poprze bibliotekę AngularJS generuje zapytania wysyłane do serwera. Wszystkie zapytania są asynchroniczne co oznacza że nie są zależne od siebie i nie powodują przeładowania strony. REST to akronim od representational state transfer. Zorientowany jest on na zasoby. Każdy z zasobów dostępny jest poprzez unikalny adres URL jak np. /room/{roomId}/messages . Pod adresy możemy wysyłać jeden z siedmiu rodzajów zapytań protokoły http jak GET, POST itp. Zapytania mogą zawierać dane w formacie JSON, XML lub mieć po prostu formę binarną. W zależności od adresu, użytego rodzaju zapytania oraz wysłanych danych serwis REST dokonuję wybranej operacji na konkretnych danych i zwraca wynik tych operacji.

## 6.4 Ogólny schemat przepływu danych

Aplikacja napisana została z wykorzystaniem biblioteki Spring w oparciu o zmodyfikowaną wersję wzorca MVC. Najbardziej zewnętrzną warstwą aplikacji są kontrolery przyjmując żądania poprzez protokół HTTP. Dzięki kontrolerom aplikacja umożliwia modyfikację oraz otrzymywanie danych. Każdy zasób dostępny jest po unikalnym adresem URL. Kontroler posiada metodę obsługującą każde żądanie. Przykładowa implementacja kontrolera:



Na podstawie implementacji od razu możemy stwierdzić pod jakim adresem dostępna jest każda metoda, na jaki rodzaj zapytania reaguje oraz jakie dane przyjmuje i zwraca. Kontroler wywołuje odpowiednie funkcjonalności dostarczonych mu serwisów, musi jednak obsługiwać pojawiające się w nich błędy.

Żądanie gdy już trafi do odpowiedniego serwisu zostaje w nim w całości obsłużone. W pierwszej kolejności zostaje sprawdzona poprawność danych. Sprawdzana jest długość poszczególnych pól, ich format, typ oraz relacje z już istniejącymi danymi. Jakakolwiek niezgodność z regułami sprawia że obsługa żądania jest przerywana i generowany jest wyjątek. Wyjątek ten obsługiwany jest w kontrolerze i na jego podstawie tworzona jest odpowiednia informacja dla użytkownika opisująca zaistniały problem.

Gdy walidacja danych się powiedzie serwis przystępuje do wykonywania odpowiedniej logiki biznesowej na dostarczonych danych. Są to operacje zapisu, odczytu, modyfikacji, usunięcia, relacji itp. Podczas tego etapu serwis wielokrotnie wysyła zapytania do bazy danych poprzez warstwę repozytoriów.

Wszystkie operację na danych odbywają w większości na obiektach typu DTO. Obiekty te mają spłaszczoną strukturę i nie posiadają bezpośrednich relacji do innych obiektów co chroni przed niepożądanymi zmianami modelu oraz błędami powiązania elementów. Dane dostarczone poprzez kontroler już są obiektami DTO natomiast wszystkie inne otrzymane poprzez repozytoria są obiektami DBO.

Do dwustronnej konwersji obiektu DTO i DBO wykorzystywana jest warstwa maperów. Maper tworzy żądany obiekt na podstawie informacji uzyskanych z dostarczonego obiektu innego typu. Ważne jest aby obiekt nie zawierał żadnych błędu ponieważ dane na tym etapie nie są weryfikowane. Jakiekolwiek błędy na tym etapie są bardzo niebezpieczne i nie należy do nich dopuścić.

Po pomyślnym wykonaniu wszystkich operacji serwis generuje wynik operacji. Operacje zapisu edycji i usunięcia zwracają dane opisujące rezultat działań. Dane te zawierają status oraz w razie wystąpienia opis błędów. Zadaniem kontrolera jest te dane zebrać, opakować odpowiednim obiektem oraz zwrócić jako wynik żądania.

# 7. Wdrożenie serwera aplikacji na środowisku produkcyjnym

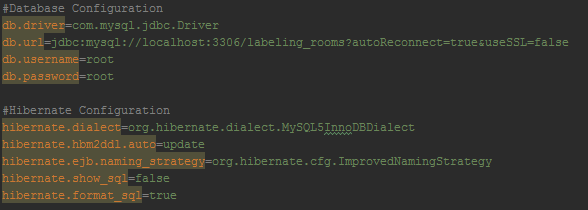
Rozdział będzie zawierał sposób konfiguracji środowiska na którym zostanie osadzony serwer aplikacji. Opisane zostaną poszczególne kroki niezbędne do prawidłowego działania aplikacji.

## 7.1 Baza danych

W środowisku należy zainstalować bazę danych MySql. Baza musi być cały czas dostępna w trakcie działania aplikacji. Należy stworzyć użytkownika z prawami administratora. Wymagane jest również stworzenia pustego schematu bazy danych.

## 7.2 Konfiguracja projektu

W module web aplikacji w głównym folderze znajduje się plik o nazwie application.properties. Jego wnętrze przedstawione jest poniżej:



Wymagana jest konfiguracja następujących elementów:

* Db.url – jest to url pod którym widoczna jest baz danych.
* Db.username – nazwa użytkownika w bazie danych posiadającego uprawnienia administratora.
* Db.password – hasło użytkownika w bazie danych posiadającego uprawnienia administratora.

Zmiana innych parametrów jest opcjonalna i nie zalecana.

## 7.3 Stworzenie pliku war

Plik ten można wygenerować poprzez narzędzie Maven poleceniem „mvn clean install”. Można również użyć opcji „export project” w używanym IDE. Należy również zmienić nazwę pliku na „ROOT”.

## 7.4 Zainstalowanie pliku war

Plik war należy zainstalować na dostarczonym wraz z projektem serwerem tomcat. Aby tego dokonać plik war musi znaleźć się w folderze „webapps”. Następnie należy uruchomić skrypt „catalina.bat” w katalogu „bin” z parametrem „run”. Po zakończonej operacji aplikacją będzie widoczna w przeglądarce po adresem http:/localhost:8080. Pierwszego logowania należy dokonać z danymi nazwa : admin, hasło: admin1.