

# PIISW, W08, IO, 2018/2019, semestr letni

## Lista zadań nr 5

Maciej Małecki  
maciej.malecki@pwr.edu.pl

27 marca 2019

### Wprowadzenie

1. Zadania z poniższej listy należy zrealizować w ramach prywatnego repozytorium założonego dla listy nr. 1.
2. Projekt z repozytorium należy co najmniej raz zbudować wykorzystując do tego komendę `mvnw install`. Pozwoli to na zainstalowanie niezbędnych narzędzi takich jak `node`, `npm`, `yarn` oraz `Angular CLI` oraz zbudowanie części *backendowej* aplikacji.
3. Do pracy z *frontendem* konieczne jest uruchomienie części *backendowej*:

```
java -jar target/ng-boot-project-seed-0.0.1-SNAPSHOT.jar
```



Jeśli z jakiegoś powodu domyślny port 8080 jest zajęty, można zmienić tę wartość (np. na 8081) modyfikując plik `application.properties`.

4. Interakcja z `Angular CLI` wymaga użycia konsoli/terminala. W ramach terminala należy przejść do katalogu głównego sklonowanego repozytorium oraz uruchomić skrypt `front_cli.bat`. Użytkownicy systemów Linux lub OS X powinni wykonać komendę: `source front_cli`. Wszelkie komendy `Angular CLI` należy wydawać będąc w podkatalogu `src/main/client`.
5. Wszystkie zadania powinny być zrealizowane w ramach prywatnego repozytorium na GitHub. Repozytorium musi być zintegrowane z Travis CI, w momencie oddawania list status CI powinien być **zielony**.
6. Należy dołożyć wszelkich starań, aby tworzona aplikacja była właściwie przetestowana przy pomocy testów jednostkowych (zarówno strona *backend* jak i *frontend*).
7. Przy realizacji zadań pomocne będą materiały z wykładu (`angular`, programowanie reaktywne) oraz przykładowe repozytorium (`angular-wykład`) - warto przeanalizować całą historię zmian.
8. Ilość wszystkich punktów możliwych do uzyskania z tej listy jest większa niż jest to konieczne do uzyskania oceny 5,0. Zadania lub punkty oznaczone symbolem 💀 należy potraktować jako w całości lub w części nieobowiązkowe (odpowiednio). Zadania 1 oraz 2 należy zrealizować co najmniej w takim stopniu, który umożliwi realizację innych zadań (zadania nie są niezależne od siebie).

# Oceny


Punkty:	< 8	8 – 9	10	11 – 12	13	14 – 15	> 15
Ocena:	2,0	3,0	3,5	4,0	4,5	5,0	5,5

## Zadania

- (4 pkt) Zarządzanie zależnościami przy użyciu **yarn**.
  - Przeanalizuj zawartość pliku `src/main/client/package.json`, sekcje: `dependencies` oraz `devDependencies` a także pliku `yarn.lock`. Korzystając ze strony <https://semver.npmjs.com/> sprawdź, jak interpretowany jest numer wersji rozpoczynający się od znaku `~` oraz od `^`, a jak interpretowany jest numer wersji podany „wprost”.
  - Korzystając z odpowiedniej formy komendy **yarn** zmień wszystkie wersje referowane przy pomocy `^` na takie, które są referowane przez `~`.
  - Korzystając z odpowiedniej formy komendy **yarn** zmigruj projekt do Angulara 5.2.0 określając przy okazji, że projekt powinien automatycznie otrzymywać poprawki błędów dla tej biblioteki<sup>1</sup>. Korzystając z komendy **git** przeanalizuj zmiany, jakie pojawiły się w plikach `package.json` oraz `yarn.lock`. Czy potrafisz określić znaczenie pliku `yarn.lock`?
  -  (dodatkowe 2 pkt) W analogiczny sposób zmigruj projekt do Angulara 7.x.x.
  - Po zmianach aplikacja powinna być przetestowana lokalnie oraz na `travis-ci`.
- (5 pkt) Twórca aplikacji *Bookstore* zawartej w repozytorium `oasp4js-ng-boot-project-seed` nie dokończył integracji z backendowym serwisem **BookRest**<sup>2</sup>. Większość integracji z częścią serwerową jest zasymulowana całkowicie po stronie klienta.
  - Aktualna implementacja serwisu frontendowego (`book.service.ts`) korzysta z przestarzałego API klienta `Http`. Użyj nowego API opublikowanego w module `@angular/common/http`. Pamiętaj o refactoringu testów (`*.spec.ts`).
  - Zintegruj istniejący interfejs użytkownika dla komponentów `book-details` oraz `book-over-view` w ten sposób, aby użyte były operacje REST dla wczytania listy książek, wczytania pojedynczej książki, dodania nowej książki oraz aktualizacji istniejącej książki.
  - Dodaj do interfejsu użytkownika możliwość usuwania wybranej książki.
- (5 pkt) Stwórz nową funkcjonalność w aplikacji *Bookstore* który pozwala na zarządzanie użytkownikami. Napisz testy jednostkowe.
  - Funkcjonalność powinna być zrealizowana jako nowy moduł (*ng-module*).
  - Funkcjonalność powinna składać się z dwóch widoków: listy użytkowników oraz edytora szczegółów użytkownika.
  - Funkcjonalność powinna realizować operacje `findAll`, `save` oraz `delete`.
  - Funkcjonalność może być zamockowana całkowicie po stronie frontendu (brak implementacji serwisu REST).
  -  (dodatkowe 2 pkt) : stwórz część backendową dla użytkowników (na poziomie analogicznym do modułu książek) oraz zintegruj ją z frontendem.

<sup>1</sup>Zwróć uwagę, że Angular to tak na prawdę kilka/kilkanaście bibliotek referowanych niezależnie

<sup>2</sup>Zobacz: `src/main/java/com/capgemini/books/rest/BookRest.java`

4. (6 pkt)  Napisz uniwersalny komponent listy elementów.
- (a) Komponent powinien umożliwiać pracę z dowolnymi tablicami elementów (w tym tablicami obiektów).
  - (b) Komponent powinien umożliwiać zdefiniowanie nagłówka listy (nazw każdej z kolumn).
  - (c) Po najechaniu kursorem myszy na dowolny wiersz nie będący nagłówkiem, cały wiersz powinien zostać wyróżniony innym kolorem (podświetlony).
  - (d) Po kliknięciu na dowolnym wierszu nie będącym nagłówkiem komponent powinien generować wstawiać wartość wybranego obiektu do asynchronicznego strumienia danych (*Observable*).
  - (e) Użyj tego komponentu w istniejących widokach listy użytkowników oraz listy książek oraz zintegruj z istniejącą funkcjonalnością. Zaktualizuj odpowiednio testy jednostkowe.