Laboratorium 11

Routing i usługa HTTP w Angular 7

Nazewnictwo komponentów, modułów, usług

Jeśli korzystamy z Angular CLI to nazwy są automatycznie tworzone z odpowiednią końcówką i słowa zaczynają się z dużych liter, np.:

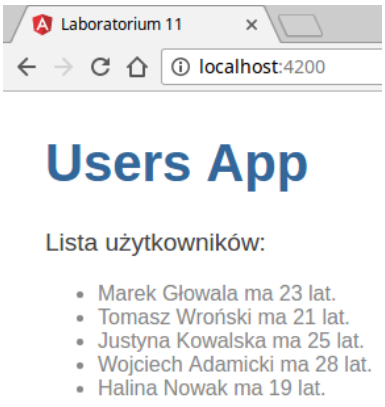
- `UsersComponent` dla polecenia `ng g component users`
- `UserService` dla polecenia `ng g service user`

Zadanie 0 – projekt do pracy

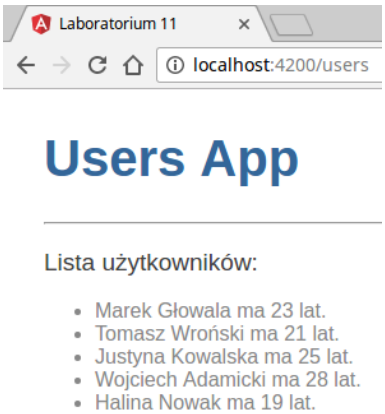
Pod linkiem lab11app.zip dostępny jest niemal pusty projekt wyświetlający tylko tytuł aplikacji. Projekt trzeba pobrać, rozpakować, doinstalować pakiety, skompilować i otworzyć w przeglądarce:

Zadanie 1 – lista użytkowników

Dodanie komponentu `users` (cała nazwa to będzie `UsersComponent`), który będzie wyświetlał listę użytkowników z tablicy z pliku `mock-users.ts`, na razie bez żadnej usługi.



Zadanie 1



Zadanie 2

Zadanie 2 – dodanie routingu

Po kolei:

- Generujemy sobie nowy moduł odpowiedzialny za trasowanie (routing). Nazywamy go jak należy czyli `app-routing` (AppRoutingModule).
- Importujemy w nim wszystkie potrzebne moduły i komponenty: `NgModule`, `RouterModule`, `Routes`, `UsersComponent`. Importowanie `CommonModule` można usunąć – nie potrzebny w module nawigacyjnym. Moduł powinien mieć w ustawieniach tylko dwie sekcje: imports i exsports.
- Dodajemy na razie jedną ścieżkę do routingu:

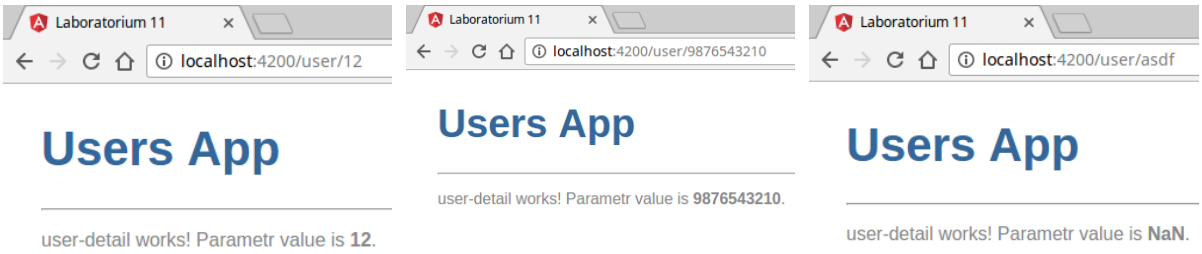
  `{ path: 'users', component: UsersComponent }`
- Dodajemy dyrektywę `<router-outlet></router-outlet>` do `AppComponent`. Można sobie też dodać `<hr>` dla odmiany od zadania 1 :)

Pod adresem `/users` powinniśmy mieć to samo co w zadaniu 1 ale już z obsługą trasowania. Inne adresy wyświetlają na razie tylko tytuł.

Zadanie 3 – komponent do edycji danych użytkownika pod adresem /user/:id

a) Routing z parametrem

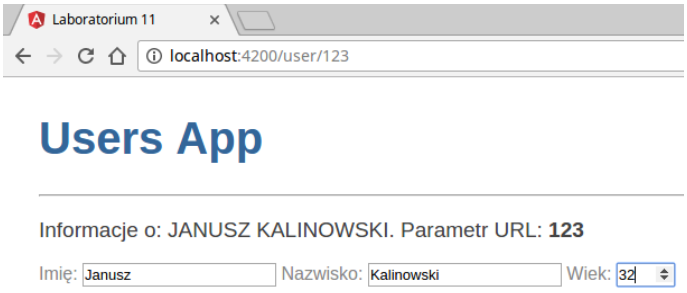
Tworzymy sobie dodatkowy komponent o nazwie user-detail, który na razie wyświetli nam tylko wartość parametru z adresu '/user/:id'. Wynik dla różnych adresów URL:



b) Kontrolki do edycji danych użytkownika

Tworzymy sobie trzy kontrolki, które w przyszłości (zadanie 5) pozwolą nam na edycję danych wybranego użytkownika. Na razie można wyświetlić dane na stałe zapisane w ngOnInit, np.:

```
this.user = { "name": "Janusz", "surname": "Kalinowski", "age": 32 };
```



Zadanie 4 – osobna usługa do zarządzania użytkownikami i odnośniki

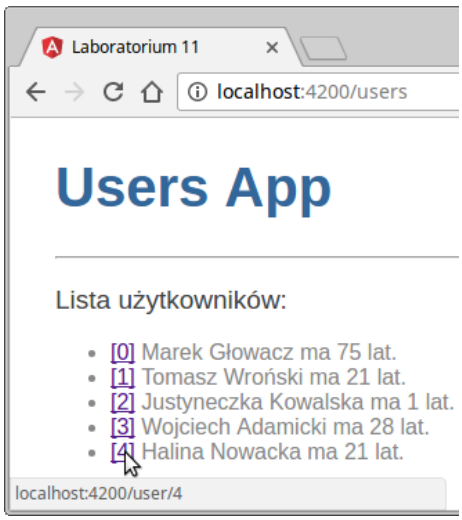
Tworzymy sobie usługę user (cała nazwa to UserService), która już w następnym zadaniu ma nam wczytać dane korzystając z zewnętrznego API. Najlepiej zrobić ją od razu korzystając programowania reaktywnego czyli korzystając z Observable, bo to będzie konieczne do wykorzystania API.

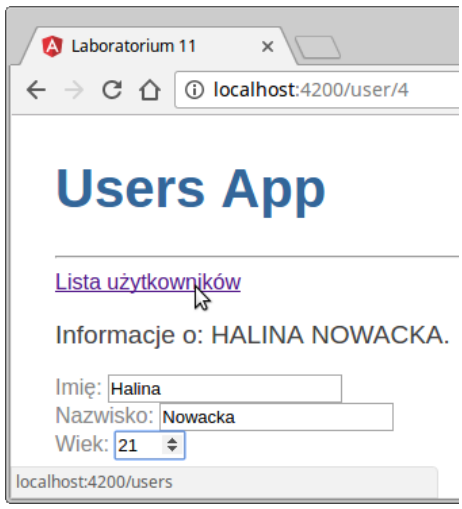
a) Tworzymy usługę, pobieramy w niej tablicę z użytkownikami, zwracamy ją jako Observable<User[]> w metodzie getUsers(). Następnie usługę wstrzykujemy do UsersComponent i tam wykorzystujemy korzystając z metody subscribe(). Wszystko powinno działać tak jak w zadaniu 2.

b) Do usługi dodajemy metodę getUser(), która zwróci obiekt obserwowany typu Observable<User>. Metoda ma zwracać obserwowanie tego użytkownika, którego id podano w parametrze. Następnie usługę wstrzykujemy do UserDetailComponent i wykorzystujemy do wyświetlenia tego użytkownika, którego id podamy w adresie URL.

c) Do komponentu UserDetailComponent dodajemy u góry odnośnik do listy użytkowników, a przy każdym użytkowniku na liście z UsercComponent dodajemy odnośnik do szczegółów danego użytkownika czyli link na odpowiedni adres URL.

Wynik z wszystkich trzech podpunktów:





Zadanie 5 – usługa HTTP

Dodajemy usługę HTTP, która wstrzyknięta do `UserService` pozwoli na pracę ze zdalnym API dostępnym pod adresem http://rperlinski.herokuapp.com/api. Interesuje nas pięć operacji:

- https://rperlinski.herokuapp.com/api/users (metoda GET) - pobranie listy wszystkich użytkowników,
- https://rperlinski.herokuapp.com/api/users/:id (metoda GET) - pobranie danych jednego użytkownika o podanym id,
- https://rperlinski.herokuapp.com/api/users (metoda POST) – utworzenie nowego użytkownika o podanym, dane całego obiektu razem z id zostaną zwrócone,
- https://rperlinski.herokuapp.com/api/users/:id (metoda PUT) - aktualizacja danych jednego użytkownika o podanym id,
- https://rperlinski.herokuapp.com/api/users/:id (metoda DELETE) – usunięcie danych jednego użytkownika o podanym id.

Końcowy wygląd aplikacji pozwalającej na pracę z API: