FE - Uruchomienie projektu

Odpalenie frontendu

Pobranie InelliJ Ultimate

Bo cudowne i potrzebne. Licencję dostaniecie na studencki mail.

Pobranie Postmana

Super narzędzie do testowania REST API (można łatwo wykonywać metody HTTP). Przyda się zarówno na backendzie, jak i frontendzie

---

Pobranie Node.js

Pobrać z oficjalnej strony i zainstalować. Jest to JS-owe SDK, z którego korzysta też Angular. Głównym narzędziem jest menadżer pakietów `npm` dzięki któremu możemy rozszerzać Node.js oraz dodawać liby do projektu.

Instalacja Angular CLI

Jest to narzędzie terminalowe `ng` pozwalające na sprawne zarządzanie projektem angularowym. Dodajemy je za pomocą `npm` w konsoli:

```
npm install -g @angular/cli
```

Generujemy nowy projekt

Tak dla waszego info, ja przygotuję już projekt.
Jak podczas instalacji zapyta o routing, dać tak. Jak zapyta o style, wybrać CSS.

```
ng new my-app
```

Uruchamiamy projekt

Wchodzimy do głównego folderu projektu i odpalamy:

```
ng serve
```

Kod aplikacji jest udostępniony przez serwerek na `localhost:4200`. Jak tam wejdziecie, to przeglądarka zaciągnie sobie html-e, css-y i js-y i apka zacznie śmigać.

Jak dodacie przełącznik `--open` to przeglądarka sama wam się otworzy na tej aplikacji.

Co ciekawe, serwerek działa w trybie **hot deployment** - to znaczy, że po każdej zmianie w kodzie apka się sama odświeża, jeżeli terminal albo browser dostanie focus.

Import i używanie w InelliJ

Angular tak naprawdę nie wymaga specjalistycznego IDE, jak np. ~~wstrętny~~ cudowny .NET - wystarczy dowolny edytor tekstu i Angular CLI do uruchamiania, generowania komponentów, etc. Przyjemnie kodzi się Angulara w VS Code.

Ja polecam jednak IntelliJ, bo i tak musicie tam odpalić backend, a fajnie mieć wszystko w jednym oknie.

1. Mając zaimportowany backend (albo nie) wejdźcie w File -> Project Structure -> Modules.
2. Kliknąć "+" -> New Module -> Static Web -> Angular CLI -> Next
3. Wybrać nadrzędny folder wygenerowanego już projektu angularowego
4. Finish
5. Nawet jak InelliJ wypluje jakieś błędy, to nie ma problem, ważne że pojawi się moduł z kodem w eksploratorze po lewej i możecie przeglądać kodzik.
6. Zarządzać projektem angularowym, tzn. uruchamiać, instalować liby, generować elementy możecie albo z waszego terminala, basha, muszli mocy, etc. albo okna terminala wbudowanego w IntelliJ (defacto jest to też domyślny systemowy terminal).

Angular - dodawanie elementów

Angular wymusza dosyć sztywną strukturę projektu, także żadnych elementów nie dodajemy z palca. **Wszystko generujemy za pomocą `ng generate` / `ng g`**.

Podstawowe elmenty

- **komponenty** - są to podstawowe "jednostki" aplikacji - trójka .html (view), .ts (controller), .css (style), **generowanie:** `ng g component <nazwa_jak_klasy_java>`, np. `ng g component DoctorList` - powstanie folder o nazwie `doctor-list-component`, a plik .ts będzie posiadał klasę `DoctorList`
- **moduły** - grupują komponenty i inne elementy (klasy, widoki, itp.) w logiczną całość, moduły możemy importować, podobnie jak pakiety w javie, folder modułu składa się z folderów komponentów, innych elementów oraz plik .ts konfigurujący moduł - m. in. widoczność jego komponentów na zewnątrz, **generowanie:** `ng g module <nazwa_jak_klasy_java>`, np. `ng g Doctors` - powstanie folder `doctors` z plikiem modułu doctors.module.ts. Będąc w folderze modułu, możemy generować w nim komponenty i inne elementy - zostaną one wówczas domyślnie dodane do modułu.
- **serwisy** - zwykła klasa na sterydach, domyślnie serwis jest singletonem, któy można wstrzykiwać (**Dependency Injection**) w dowolnym miejscu aplkiacji, serwisy udostępniają jakiś zestaw pomocniczych funkcjonalności (np. do tłumaczenia powiadomień, routing w aplikacji) oraz wykorzystuje się je do opakowywania zapytań HTTP, **generowanie:** `ng g service`

`<nazwa_jak_klasy_java>`, np. `ng g service Patients` stworzy plik serwisu patiens.service.ts z klasą PateintsService

- **klasy** - wiadomka: `ng g class <nazwa_jak_w_java>`
- **interceptory** - nie można ich generować (chyba że wprowadzili nowość w v.9 cli) - są to klasy na sterydach pełniące rolę filtrów - robią coś z zapytaniami HTTP wychdzącymi albo przchdzącymi (np. dodają nagłówek autoryzacji, obsługują błędy HTTP, itp.)
- **dyrektywy** - klasy na sterydach, które dodają własności do dokument HTML - można je potym wykorzystać jako atrybuty HTML, przykład użycia - zmiana koloru po najechaniu myszki na tekst, **generowanie:** `ng g directive <nazwa_dyrektywy_jak_w_java>`
- **pipey** - klasa na sterydach implementująca interfejs `transform()`, w sposób dynmamiczny transformuje prezentację danych w HTML przez użycie operatora `|`, np. wyświetlanie kwoty i waluty w zależności od ustawień lokalnych przeglądarki, **generowanie:** `ng g pipe <nazwa_jak_w_java>`

Pipe - przykład

```
<p>
{{currency}} | localCurrencyPipe
</p>
```

Doczytajcie

Większość jest tu: jest tu: https://angular.io/docs

- Single Page Application (https://en.wikipedia.org/wiki/Single-page_application)
- Angular Routing
- Angular - two way binding (wiązanie HTML view z typescript contollerem) (operatory `{{ }}`, `[]`, `()` w HTML)
- Angular - HTTP Client
- Podstawy typescript'a (JS wam niekoniecznie jest potrzebny) https://www.theserverside.com/tutorial/What-Java-developers-need-to-know-about-TypeScript-syntax
- Angular - Observable, Subscription
- Angular - Depenency Injection, Lifecycle hooks
- Angular Forms

Dodakowe info

- jak coś generujecie, to pojawiają się też pliki .spec.ts - są to pliki testów, które w projekcie studenckim się zlewa albo usuwa

- dodałem już bootstrapa do projektu - nic w html nic nie musicie dodawać, importować, etc - out of the box wpisujecie nazwy klass css w divach i innych elementach, **pamiętajcie tylko**, żeby wszystko w html umieszczać w nadrzędnym divie `<div class="container"></div>` - i dalej już w bootstrapowym gridzie https://getbootstrap.com/docs/4.0/layout/grid/
- do stylizacji korzystajcie ze styli bootstrapowych - to już ma paski, buttony, menu, tabele, itd. - nic nie trzeba samemu stylizować, a jedynie poukładać sensownie https://getbootstrap.com/docs/4.0/components/buttons/
- oprócz bootstrapa zainstalowałem też zbiór ikon wektorowych font awesome, które dodajecie również w atrybucie "class" https://astronautweb.co/snippet/font-awesome/