Laboratorium 1: Pierwszy projekt w Angular

https://angular.io/

https://www.jetbrains.com/webstorm/

Ćwiczenie 1. Tworzenie środowiska

Praca z aplikacjami frontendowymi wymaga instalacji:

- Node.js: https://nodejs.org/en/download/
- Angular-cli: https://angular.io/
- GIT – system kontroli wersji : https://git-scm.com/

Poniżej znajduje się opis procesu instalacji dla komputerów z uczelni. Na własnym komputerze trzeba będzie również zainstalować powyższe narzędzia, jednak proces może się różnić w zależności od systemu.

Na systemach Windows zawsze trzeba uaktualnić ścieżkę PATH dla Node.js.

https://angular.io/guide/quickstart#devenv

Pominąć na uczelni ----------

1. Aby rozpocząć pracę z Angularem niezbędna nam będzie biblioteka **node.js**. Jeśli jeszcze nie masz jej zainstalowanej, dokonaj instalacji: https://nodejs.org/en/download/

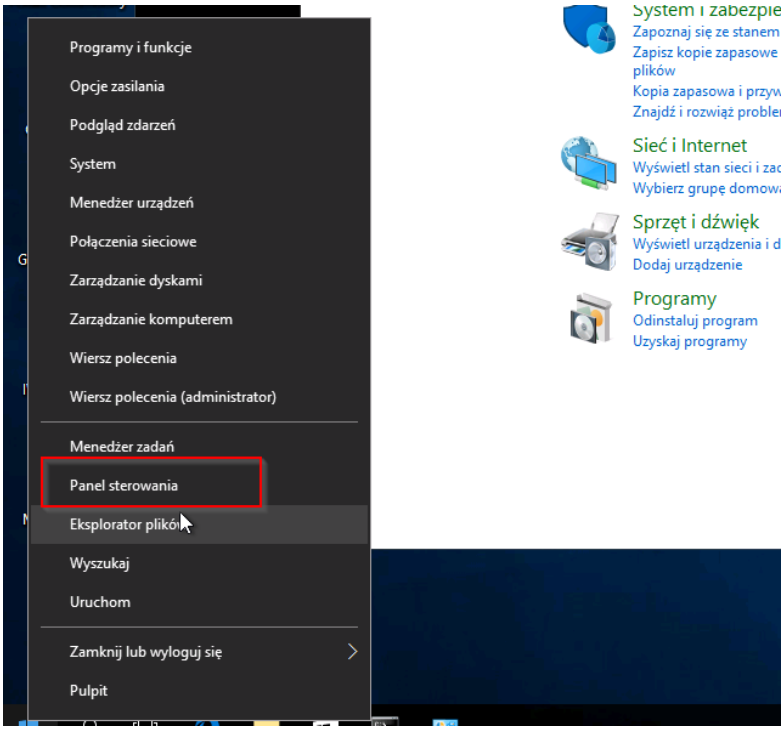
2. Następnie korzystając z **konsoli systemowej cmd** należy sprawdzić wersję node.js:

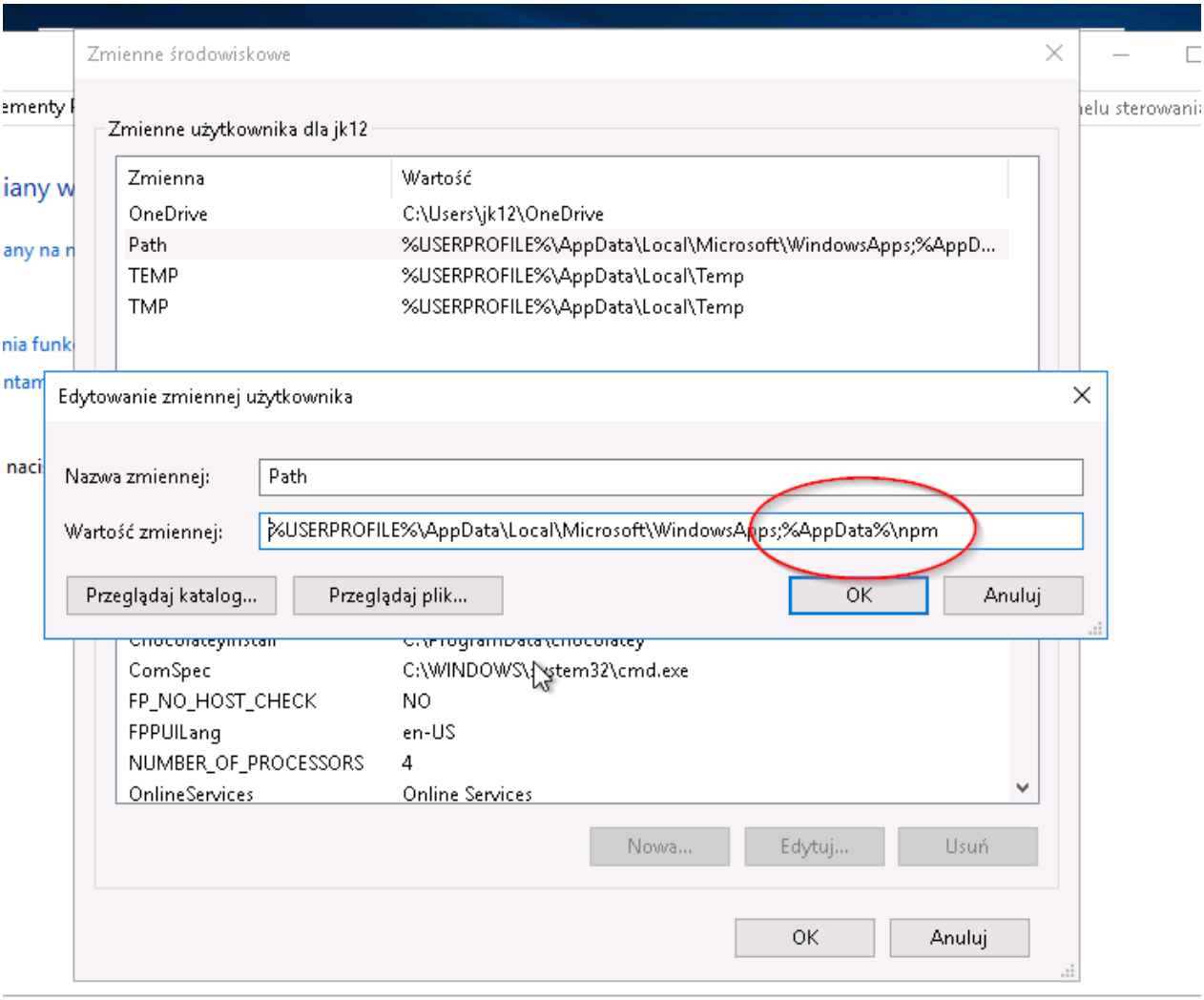
   ```
   npm -v
   ```

---

3. Do zmiennej środowiskowej PATH trzeba dodać ścieżkę do npm
   - Przejdź do: **Panelu sterowania** i wyszukaj **Zmienne środowiskowe**.
   - Wybierz: **Zmienne środowiskowe dla konta**
   - Następnie w PATH dopisz po średniku (lub w nowej linii):

     %AppData%\npm


Programy i funkcje
Opcje zasilania
Podgląd zdarzeń
System
Menedżer urządzeń
Połączenia sieciowe
Zarządzanie dyskami
Zarządzanie komputerem
Wiersz polecenia
Wiersz polecenia (administrator)
Menedżer zadań
Panel sterowania
Eksplorator plików
Wyszukaj
Uruchom
Zamknij lub wyloguj się
Pulpit
System i zabezpie
Zapoznaj się ze stanem
Zapisz kopie zapasowe plików
Kopia zapasowa i przyw
Znajdź i rozwiąż proble
Sieć i Internet
Wyświetl stan sieci i zac
Wybierz grupę domowa
Sprzęt i dźwięk
Wyświetl urządzenia i d
Dodaj urządzenie
Programy
Odinstaluj program
Uzyskaj programy


Zmienne środowiskowe
Zmienne użytkownika dla jk12
Zmienna
Wartość
OneDrive
C:\Users\jk12\OneDrive
Path
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps;%AppD...
TEMP
%USERPROFILE%\AppData\Local\Temp
TMP
%USERPROFILE%\AppData\Local\Temp
Edytowanie zmiennej użytkownika
Nazwa zmiennej:
Path
Wartość zmiennej:
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps;%AppData%\npm
Przeglądaj katalog...
Przeglądaj plik...
OK
Anuluj
ComSpec
C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_CHECK
NO
FPPUILang
en-US
NUMBER_OF_PROCESSORS
4
OnlineServices
Online Services
Nowa...
Edytuj...
Usuń
OK
Anuluj

4. Teraz otwieramy konsolę – wystarczy wyszukać CMD i wybrać konsolę systemową.
5. Wpisujemy :

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

6. I sprawdzamy wersję Angulara:

```
ng -v
```

7. Jeśli Angular nie został rozpoznany musimy wrócić do zmiennej PATH i dopisać:

   %AppData%\npm\node_modules\@angular\cli\bin

8. Zamykamy konsolę i ponownie otwieramy (musimy zrestartować) i sprawdzamy wersję Angulara.

9. Jeśli działa – otwieramy Webstorma (open) na dysku lub katalogu, w którym chcemy mieć projekt aplikacji i w terminalu Webstorma wpisujemy:

```
ng new angular
```

   UWAGA! Polecenie utworzy nam NOWY katalog o nazwie angular!

   Node.js ściągnie i zainstaluje wszystko to, czego potrzebujesz, żeby zacząć pracę z Angularem.

10. W terminalu Webstorm przejdź do katalogu z projektem i uruchom serwer.

```
cd angular
ng serve --open
```

    Polecenie –open powinno automatycznie otworzyć okno przeglądarki pod adresem: http://localhost:4200/ a Ty powinieneś zobaczyć to:

Welcome to app!!

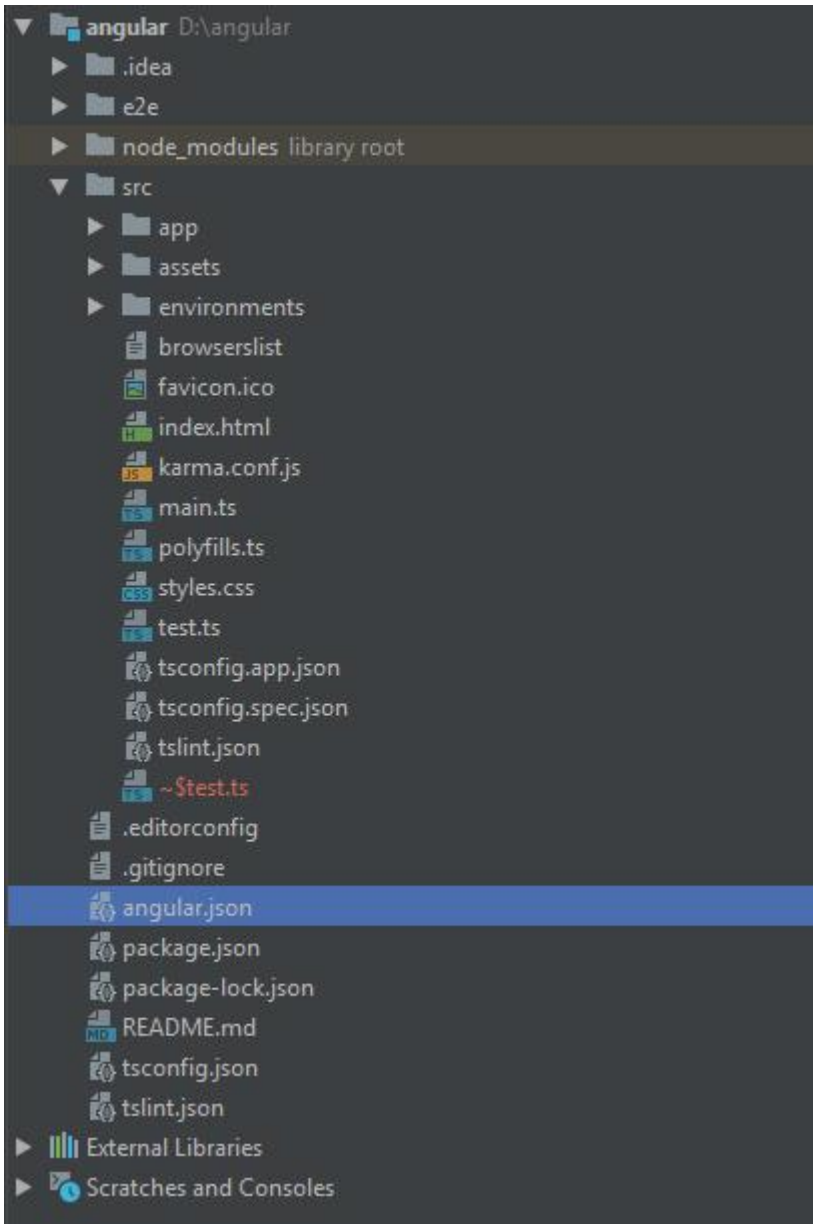
Ćwiczenie 2. Preprocesor SCSS

Ok, Angular zainstalowany, ale zanim przejdziemy do dalszej pracy skonfigurujemy sobie co nieco, żeby ułatwić sobie życie i pracę w przyszłości.

Po pierwsze skonfigurujemy Angulara tak, żebyśmy zamiast plików .css mogli używać preprocesora SASS i plików .scss.

Zaczynamy!

1. Otwórz aplikację Webstorm i wybierz opcję Open i katalog z projektem. Struktura katalogu wygląda tak:



To, co nas interesuje to katalog src – to on tak naprawdę zawiera naszą aplikację. Pozostałe katalogi i pliki to elementy środowiska wspomagające naszą pracę.

W katalogu app znajduje się główny komponent naszej aplikacji. Zanim przyjrzymy mu się bliżej, zwróćmy uwagę na *app.component.css* . To coś, czego nie chcemy – nasza aplikacja ma wykorzystywać SCSS.

Sprawmy, żeby tak się działo.

2. Na początek powiemy Angularowi, że ma domyślnie korzystać z preprocesora SASS.

   W konsoli wpisz polecenie:

```
ng config schematics.@schematics/angular:component.styleext scss
```

3. Sprawdźmy, czy zadziałało. Otwórzmy plik *angular.json*. Na samym dole powinniśmy zobaczyć:

```
"defaultProject": "angular",
"schematics": {
  "@schematics/angular:component": {
    "styleext": "scss"
  }
}
```

4. Pozostało nam jeszcze zmienić rozszerzenie już istniejącego pliku *app.component.css* na *app.component.scss*

To było proste!

To teraz coś trudniejszego. Używając preprocesora SASS zwyczajowo nie korzysta się z jednego pliku ze stylami css, ale z wielu plików podzielonych według pewnej logiki.

W naszym przypadku Angular sam dołącza pojedyncze pliki scss do komponentów, tworząc style lokalne, obowiązujące tylko w obrębie komponentu. Jednakże korzysta się też ze styli globalnych, gdyż to ułatwia pracę i ujednolica wygląd aplikacji.

Uparliśmy się na ten SASS przede wszystkim dlatego, że umożliwia nam tworzenie w stylach **zmiennych** oraz **mixins** – czegoś w rodzaju funkcji, które możemy dołączać w innych plikach.

(wszystkie paczki poniżej do pobrania na stronie laboratoriów)

1. Na początek do katalogu *assets* rozpakujemy paczkę fonts – to nasza czcionka font-awesome.

2. Następnie w katalogu *assets* **utwórzmy** katalog *styles* i wgramy do niego zawartość paczki:

    a. basic_scss.zip – kilka podstawowych styli, mixins, variables oraz app.scss – plik do ogólnych styli
    b. gridle.zip – bibliotekę do tworzenai siatki grid

3. oraz utwórzmy plik **main.scss** – plik scalający wszystkie pliki scss w jedno i zaimportujmy w nim wgrane pliki.

```
@import "variables";
@import "reset";
@import "mixins";
@import "grid";
@import "buttons";
@import "fonts";
@import "forms";
@import "icons";
@import "text";
@import "app";
```

4. Na koniec zmienimy ścieżkę w *angular.json* w taki sposób, żeby wiedział, z czego ma korzystać przy tworzeniu styli aplikacji.

   Zamiast

   ```
   "styles": [
           "styles.css"
    ],
   ```

   Wpisujemy

   ```
   "styles": [
           "src/assets/styles/main.scss"
   ],
   ```

Ćwiczenie 3. Siatka grid

Na zajęciach z technologii hipertekstowych używaliśmy Bootstrapa, gdyż zapewniał nam wsparcie dla RWD (Responsive Web Design) oraz dostarczał gotowe rozwiązania JavaScript. Teraz jednak do JavaScript mamy Angulara, więc nie ma sensu obciążać się tak wielką biblioteką, skoro wszystko możemy napisać samodzielnie.

Jest jedna rzecz, której nie ma sensu pisać od podstaw: siatka grid. Wykorzystamy sobie do niej niewielką bibliotekę **gridle{.scss}**: http://gridle.org/documentation

Pliki gridle powinniśmy mieć już dołączone, więc je skonfigurujmy.

1. Zajrzyjmy do pliku *grid.scss*

main.scss | grid.scss | _grid-settings.scss

```
@import 'gridle/gridle';
@import 'gridle/settings-mixins';
@import 'gridle/gridle-flex';
@import 'grid-settings';

* {
    box-sizing: border-box;
}

.container {
    @include gridle_container();
    max-width: 1420px;
    margin: 0 auto;
    position: relative;

    padding: 0 10px 20px 10px;

    @include gridle_generate_classes();
}
```

Od góry na początek zaimportowaliśmy gridla i ustawienia reguł, następnie mamy gridle-flex. To oznacza, że będziemy korzystać z możliwości Flexbox: https://www.w3schools.com/css/css3_flexbox.asp Idźmy dalej.

Jak widać mamy tu już ustawione wartości dla głównego kontenera naszej aplikacji. Zmieńmy szerokość na 1640px, ustawmy też padding na: 10px 20px.

2. Otwórzmy plik *grid-settings.scss* i zobaczmy co tam się dzieje. A dzieje się dużo. Na początek mamy rejestrację kilku klas css, które będziemy wykorzystywać później.
   Następnie pojawia się *gridle_setup*.

   Domyślnie gridle dostarcza 12 kolumn z odstępem pomiędzy nimi 20px.
   Przestawmy to na 16 kolumn i odstęp 0.

main.scss × grid.scss × _grid-settings.scss ×

```
@include gridle_set_classname_map('grid', ('grid-', '', '%column', '@', '%state'));
@include gridle_set_classname_map('grid-table', ('grid-', '', 'table', '@', '%state'));
@include gridle_set_classname_map('grid-adapt', ('grid-', '', 'adapt', '@', '%state'));
@include gridle_set_classname_map('grid-grow', ('grid-', '', 'grow', '@', '%state'));
@include gridle_set_classname_map('grid-centered', ('grid-', '', 'centered', '@', '%state'));

@include gridle_setup((
        context: 12,
        gutter-width: 20,
        debug: true,
        gridle-vendor-prefix: false,
        debug-show-class-names: true
));

@include gridle_register_state(mobile, (
        max-width : 768px
));

@include gridle_register_state(tablet, (
        min-width: 769px,
        max-width: 1024px
));

@include gridle_register_state(all-mobile, (
        max-width: 1024px
));

@include gridle_register_state(desktop, (
        min-width: 1025px,
        max-width: 1599px
));

@include gridle_register_state(all-desktop, (
        min-width: 1025px
));
```

3. Jeszcze niżej mamy klasy odpowiedzialne za punkty przełamania dla różnych urządzeń. Mamy mobile, tablet, desktop. To odpowiedniki bootstrapowych klas *col-sm, col-md, col-lg*. Podczas pracy nad aplikacją będziemy korzystać z nich w następujący sposób:

```
<div class="grid-4 grid-8@tablet grid-16@mobile"></div>
```

4. Dopiszmy sobie jeszcze jedną regułę dla urządzeń o rozdzielczości większej niż 1600px.

```
@include gridle_register_state(large, (
        min-width: 1600px
));
```

Skończyliśmy! Ale…

Po zakończeniu prac w konsoli wpisujemy:

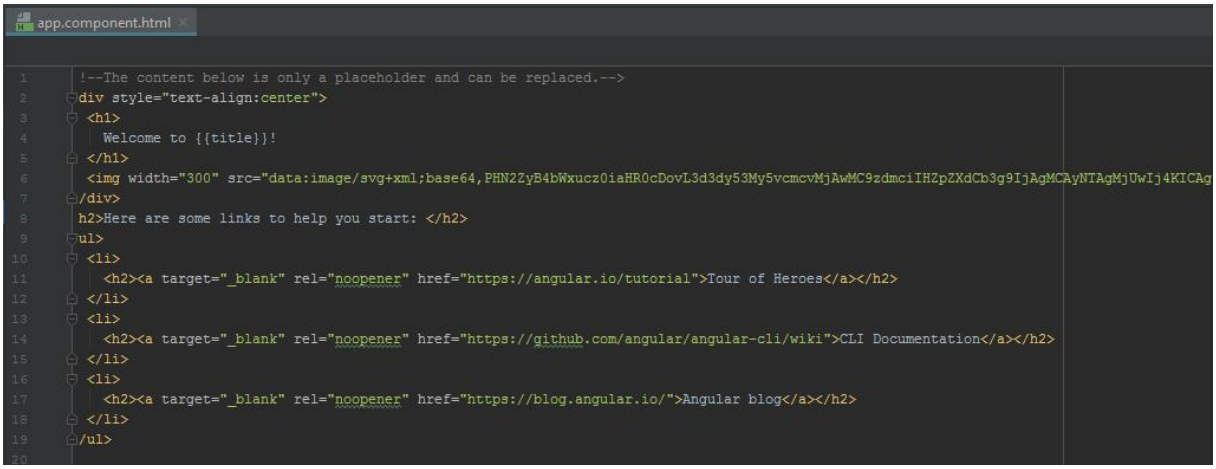
```
ng serve --open
```

Jeśli nie będzie błędów, aplikacja odpali się pod adresem http://localhost:4200/ i będzie można zobaczyć efekty.

Można zmieniać do woli!

Sprawdźmy jeszcze czy to działa.

Przebudujmy trochę plik *app.component.html*



Zamiast `<div style="text-align:center">` wstawimy głowny konterer gridle `<div class=”container”>`

Dodamy `<div class=”row”>` oraz dwie kolumny, do których przeniesiemy nagłówek h1, obrazek oraz listę.

Na koniec dodamy do nagłówka h1 ikonke font awesome

`<span class=fa fa-caret-right”></span>`

app.component.html

```
<!--The content below is only a placeholder and can be replaced.-->
<div class="container">
    <div class="row">
        <div class="grid-5 grid-centered">
            <h1><span class="fa fa-caret-right"></span> Welcome to {{title}}!</h1>
            <img width="300" src="
        </div>
        <div class="grid-11 grid-centered">
            <h2>Here are some links to help you start: </h2>
            <ul>
                <li>
                    <h2><a target="_blank" rel="noopener" href="https://angular.io/tutorial">Tour of Heroes</a></h2>
                </li>
                <li>
                    <h2><a target="_blank" rel="noopener" href="https://github.com/angular/angular-cli/wiki">CLI Documentation</a></h2>
                </li>
                <li>
                    <h2><a target="_blank" rel="noopener" href="https://blog.angular.io/">Angular blog</a></h2>
                </li>
            </ul>
        </div>
    </div>
</div>
```

Końcowy wynik nie jest zbyt imponujący, ale widać, że wszystko działa.

Ćwiczenie 4. Repozytorium – tworzenie i zapisywanie zmian

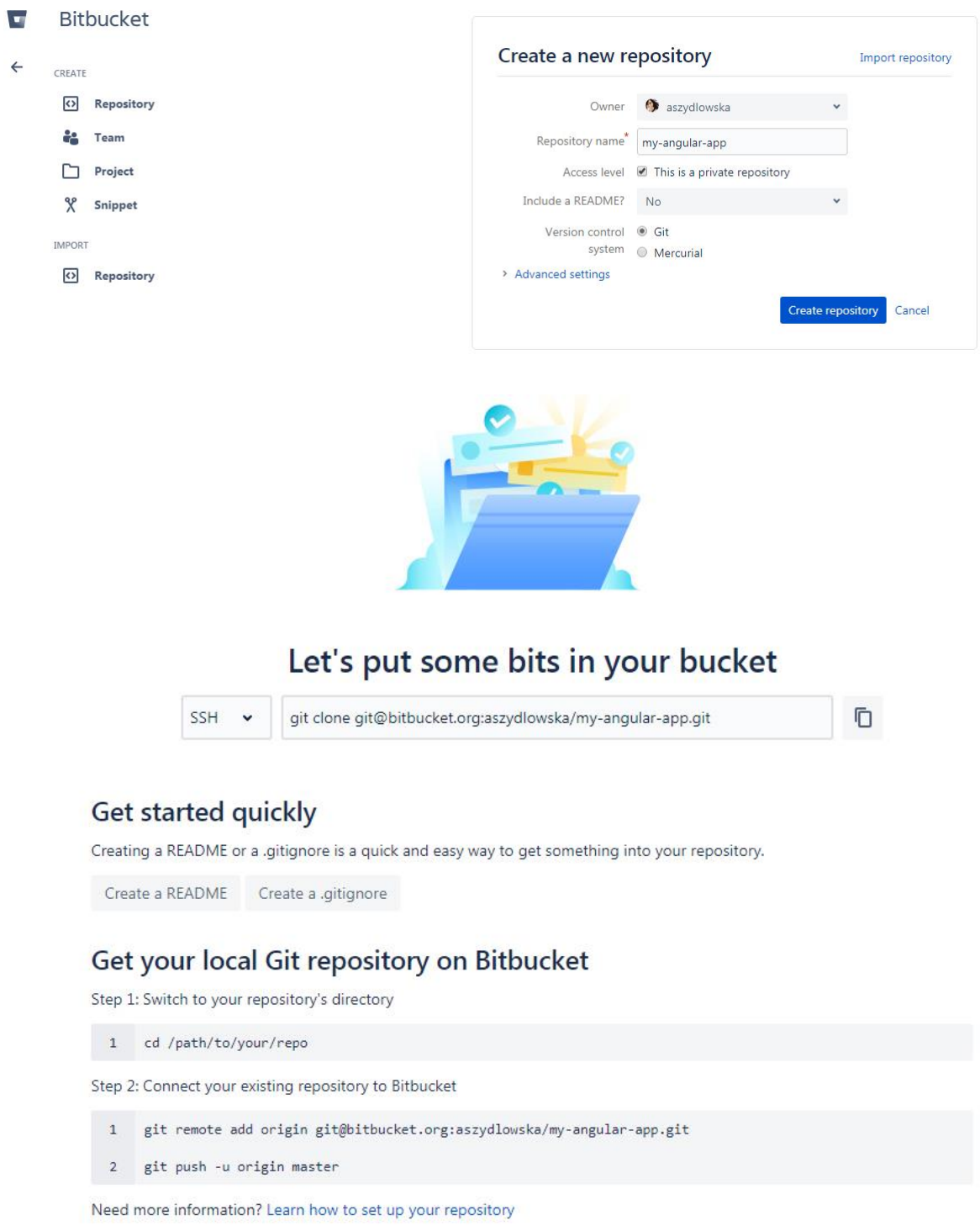
Mamy już aplikację i jeśli pokusimy się o sprawdzenie, okaże się, że razem z bibliotekami jest OGROMNA, a my musimy ją jakoś przenosić między różnymi urządzeniami.

Dlatego wykorzystamy narzędzie kontroli wersji jakim jest GIT. Git to system kontroli wersji, czyli system trzymający poszczególne wersje kodu i pozwalający do pracy na różnych wersjach tego kodu wielu osobom.

My wykorzystamy go głównie do współdzielenia pracy na różnych urządzeniach, tak, żeby nie trzeba było za każdym razem kopiować całej aplikacji.

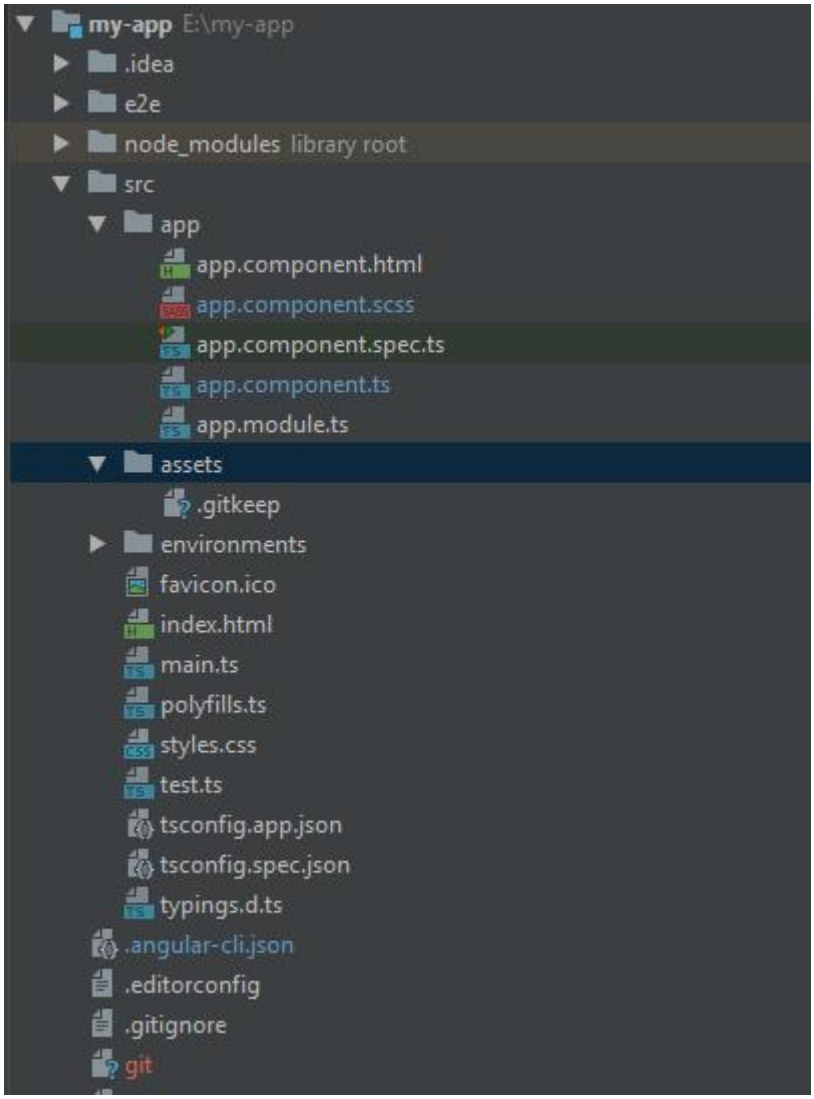
1. Na początek trzeba założyć konto na Bitbucket: https://bitbucket.org

2. Następnie zakładamy nowe repozytorium:



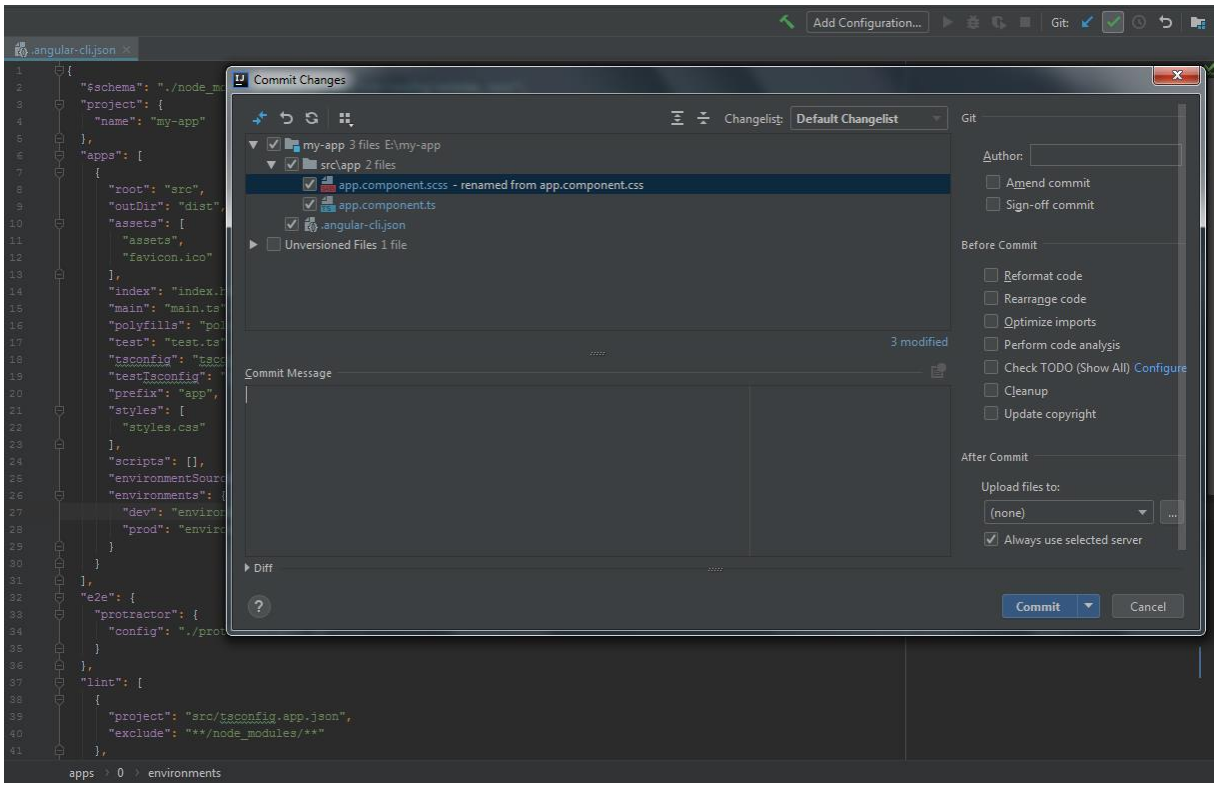
3. W terminalu Webstorma wpisujemy polecenia z kroku drugiego.
4. W menu bocznym na Bitbucket klikamy Commits i powinniśmy tam zobaczyć commit Angular CLI – jesteśmy podłączeni do własnego repo z Angularem. Od tej pory zmiany nam nie straszne.

5. Jeśli teraz popatrzymy w Webstorma to zobaczymy, że nasze pliki mają różne kolory:



- Pliki białe - pliki, których w ogóle nie zmienialiśmy.
- Pliki niebieskie – pliki które zmienialiśmy i nie wysłaliśmy do repozytorium po zmianie.
- Pliki czerwone – nie należą do repo, a więc ich zmiany nie będą śledzone.
- Pliki zielone – zupełnie nowe pliki dodane do aplikacji i GIT, ale jeszcze nie wysłane do repozytorium.

6. Wyślijmy więc zmienione przez nas pliki do repozytorium. W prawym górnym rogu Webstorma znajduje się zielona ikonka Check. Po jej kliknieciu otworzy się okno Commita:



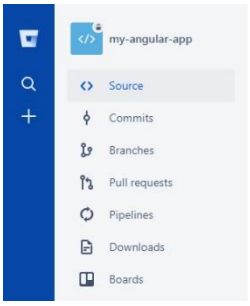
Wpisujemy comment message, np. Basic app settings i rozwijamy dropdown „Commit” – wybieramy Commit and Push.

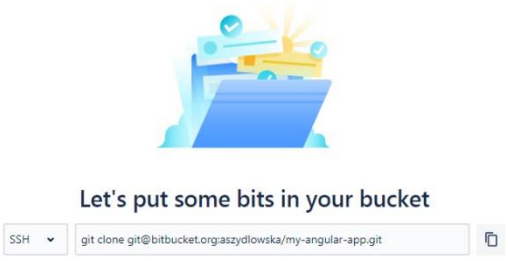
Commit oznacza, że zmiany zostaną zapisane lokalnie na maszynie, na której właśnie pracujemy. Dopiero Push wyśle je na serwer bitbucketa.

Ćwiczenie 5. Repozytorium – odtworzenie i pobranie zmian

Odtworzenie istniejącego repozytorium

Na nowym komputerze wystarczy teraz, że w wybranej lokalizacji w konsoli Webstorma skopiujemy polecenie, które znajdziemy na stronie Source Bitbucketa:





Przejdziemy do katalogu z aplikacją: `cd nazwa_katalogu`

i wpiszemy: `npm install`

Pobieranie zmian z repozytorium

Aby pobrać zmiany z repozytorium wystarczy skorzystać z możliwości Webstorma – z poniższego okna wybieramy Pull i akceptujemy wybór brancha w kolejnym oknie.