|  |  |  |
| --- | --- | --- |
| **AI2B** | Hubkiewicz Jakub  Album 46525  Grupa 2 | **Wersja 1** |
|  |
| **LAB B** |

Komponenty

# Spis treści

[Spis treści 1](#_Toc150776158)

[Cel zajęć 1](#_Toc150776159)

[Rozpoczęcie 1](#_Toc150776160)

[Uwaga 1](#_Toc150776161)

[Inicjalizacja projektu 1](#_Toc150776162)

[Dodawanie komponentów i serwisu 4](#_Toc150776163)

[Implementacja serwisu RandomService 5](#_Toc150776164)

[Implementacja komponentu Random 7](#_Toc150776165)

[Implementacja komponentu listComponent 12](#_Toc150776166)

[Commit projektu do GIT 17](#_Toc150776167)

[Podsumowanie 17](#_Toc150776168)

# Cel zajęć

Celem głównym zajęć jest zapoznanie się z podstawą tworzenia i rozwijania projektów aplikacji z użyciem frameworku Angular.

# Rozpoczęcie

Rozpoczęcie zajęć. Powtórzenie tworzenia komponentów. Powtórzenie wiązań.

Wejściówka?

# Uwaga

Ten dokument aktywnie wykorzystuje niestandardowe właściwości. Podobnie jak w LAB A wejdź do Plik -> Informacje -> Właściwości -> Właściwości zaawansowane -> Niestandardowe i zaktualizuj pola. Następnie uruchom ten dokument ponownie lub Ctrl+A -> F9.

# Inicjalizacja projektu

Wejdź terminalem do katalogu C:\...\Desktop\ai2b i zainicjalizuj projekt z wykorzystaniem komendy:

> ng new lab-b

Standardowo kreator zapyta o konfigurację routingu (wybrać Nie) oraz preprocesor CSS (zostawić zwykły CSS). Zainicjalizowane zostanie także repozytorium GIT.

Po zakończonej instalacji, uruchom aplikację w trybie deweloperskim z wykorzystaniem komendy:

> cd C:\...\Desktop\ai2b\lab-b

> ng serve --port=

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Uruchom przeglądarkę pod adresem: http://localhost:00000:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Otwórz projekt w wybranym przez siebie IDE. Edytuj plik src/app/app.component.html do postaci:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Następnie zmień tytuł w klasie AppComponent w pliku src/app/app.component.ts:

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

Strona zostanie automatycznie zaktualizowana:

Obraz zawierający tekst, Czcionka, oprogramowanie, numer

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej Twoją aplikację. Upewnij się, że na zrzucie ekranu widoczny jest Twoje nazwisko i numer albumu.

Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Dodawanie komponentów i serwisu

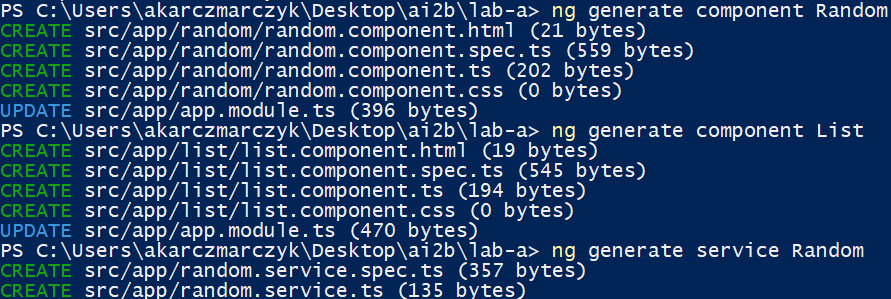
Kolejnym etapem prac jest wygenerowanie (w sposób automatyczny) dwóch komponentów (RandomComponent i ListComponent) oraz serwisu RandomService. W tym celu wykonaj poniższe 3 polecenia:

> ng generate component random

> ng generate component list

> ng generate service random

Utworzone zostaną katalogi dla dwóch komponentów (src/app/random i src/app/list) oraz plik serwisu (src/app/random.service.ts):

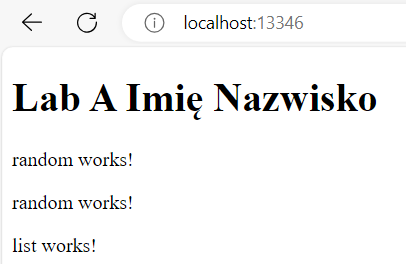


Dodaj do pliku src/app/app.component.html dwa wystąpienia komponentu RandomComponent i jedno wystąpienie komponentu ListComponent:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Oczekiwany efekt:



Umieść poniżej zrzut ekranu przeglądarki wyświetlającej Twoją aplikację z dwoma komponentami RandomComponent i jednym komponentem ListComponent. Upewnij się, że na zrzucie ekranu widoczny jest Twoje nazwisko i numer albumu.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, multimedia

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Implementacja serwisu RandomService

W tej części zaimplementujemy metodę randomNumber(max), która zwraca liczbę całkowitą od 1 (włącznie) do max (włącznie).

Edytuj uprzednio wygenerowany plik src/app/random.service.ts. Dodaj metodę:

randomNumber(max: number): number {

return 0;

}

Edytuj plik testu src/app/random.service.spec.ts do postaci:

import { TestBed } from '@angular/core/testing';

import { RandomService } from './random.service';

describe('RandomService', () => {

let service: RandomService;

beforeEach(() => {

TestBed.configureTestingModule({});

service = TestBed.inject(RandomService);

});

describe('randomNumber', () => {

it('should return between 1-100', () => {

const max = 100;

for (let i = 0; i < 10; i++) {

const drawn = service.randomNumber(max);

expect(drawn).toBeLessThanOrEqual(max);

expect(drawn).toBeGreaterThanOrEqual(1);

expect(Math.round(drawn)).withContext('Must be integer').toEqual(drawn);

}

});

it('should return values [1,2,3] only', () => {

const max = 3;

const possibleValues = [1,2,3];

for (let i = 0; i < 10; i++) {

const drawn = service.randomNumber(max);

expect(possibleValues).toContain(drawn);

}

});

});

});

Uruchom w terminalu testy serwisu Random:

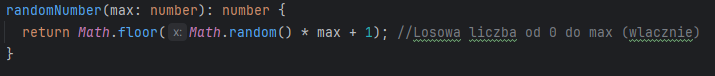
> ng test --include=\*\*/random.service.spec.ts

Otworzy się przeglądarka na stronie wyników testów. Ponieważ randomNumber na razie zawsze zwraca 0, oczywiście testy nie przejdą:



Zmodyfikuj metodę randomNumber, żeby zwracała losową liczbę całkowitą od 1 (włącznie) do max (włącznie). Po każdej edycji serwisu, testy wykonają się ponownie. Ulepszaj swoją metodę aż do momentu, gdy testy przejdą (zielono, 2specs, 0 failures).

Umieść poniżej zrzut ekranu przedstawiający kod metody randumNumber():



\*Losowa liczba od 1 do max (włącznie), nie zmieniłem komentarza

Umieść poniżej zrzut wyniku testu (zakończonego sukcesem):

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Implementacja komponentu Random

Komponent RandomComponent powinien mieć przycisk generujący nową liczbę losową, która jest następnie wyświetlana. Komponent powinien przyjmować parametr wejściowy max (domyślnie 10), określający zakres liczb do losowania (1, 2, …, max). Oznacza to, że każda instancja komponentu może losować liczby z innego zakresu.

Dodatkowo komponent powinien mieć zaimplementowaną logikę określającą, czy aktualnie wylosowana liczba jest:

* mniejsza od połowy zakresu (< 0.5 \* max) – wyświetlić zielony napis „dolna połowa wartości”
* większa lub równa od połowy zakresu (>= 0.5 \* max) – wyświetlić czerwony napis „górna połowa wartości”

Przykład:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Przykładowa implementacja komponentu:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Przykładowa implementacja widoku:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Przykładowa implementacja styli:



Umieść poniżej zrzut ekranu działającej aplikacji, z jednym komponentem o domyślnym max 10, a drugim o zmienionym max 100. Przeprowadź losowanie w komponentach tak, żeby na jednym wyświetlał się zielony napis, a na drugim czerwony:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przedstawiający kod random.component.css:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przedstawiający kod random.component.html:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przedstawiający kod random.component.ts:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przedstawiający kod app.component.html:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Implementacja komponentu listComponent

Komponent ma umożliwiać podanie w polu formularza dowolnego tesktu. Po zatwierdzeniu go przyciskiem powinien on być dodany do listy i wyświetlony. Każdy z elementów na liście powinien mieć obok link „usuń” usuwający go z tej listy.

Przykładowa implementacja komponentu:

Obraz zawierający tekst

Opis wygenerowany automatycznie

oraz widoku

Obraz zawierający tekst

Opis wygenerowany automatycznie

Przykładowy wygląd:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Do app.module.ts będzie trzeba zaimportować:

import { FormsModule } from "@angular/forms";  
...  
imports: [

BrowserModule, FormsModule

],

Umieść poniżej zrzut ekranu działającej aplikacji, bez dodanych elementów na liście:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu działającej aplikacji, z dodanymi elementami listy:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu działającej aplikacji, po usunięciu elementu listy:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przedstawiający kod list.component.css:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przedstawiający kod list.component.html:

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przedstawiający kod list.component.ts:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie Obraz zawierający tekst, zrzut ekranu, Czcionka, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Commit projektu do GIT

Utwórz repozytorium publiczne GitHub na tę część kursu. Wyślij swój projekt do repozytorium (push). Upewnij się, czy wszystko dobrze się wysłało. Jeśli tak, to z poziomu przeglądarki utwórz branch o nazwie lab-b na podstawie bieżącej gałęzi kodu.

W zależności od przyjętej konwencji (jedno repo per laboratorium kontra jedno repo na wszystkie laboratoria), konieczne może być usunięcie katalogu .git i ponowna samodzielna inicjalizacja.

Podaj link do brancha lab-b w swoim repozytorium:

<https://github.com/huuuuubi/AI2B-L-grN2-Hubkiewicz-Jakub/tree/lab-b>

# Podsumowanie

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Nauczyłem się tworzyć losowe wartości i wyświetlać je w zdefiniowanych typach bloków. Tak samo z obsługą list. Super sprawa.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.