

KOMPONENTY

SPIS TREŚCI

Spis treści	1
Cel zajęć.....	1
Rozpoczęcie	1
Uwaga	1
Inicjalizacja projektu.....	1
Dodawanie komponentów i serwisu	4
Implementacja serwisu RandomService	5
Implementacja komponentu Random	7
Implementacja komponentu listComponent	12
Commit projektu do GIT.....	17
Podsumowanie.....	17

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:\Users\akarczmarczyk\Desktop\ai2b\lab-a> ng serve --port=00000
```

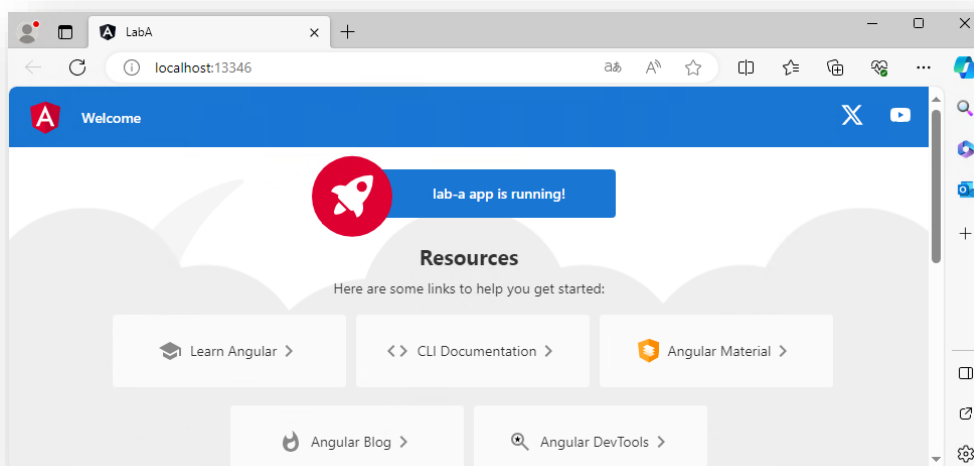
```
PS C:\Users\akarczmarczyk\Desktop\ai2b\lab-a> ng serve --port=13346
✓ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size
vendor.js           | vendor | 2.04 MB
polyfills.js        | polyfills | 333.16 kB
styles.css, styles.js | styles | 230.44 kB
main.js             | main | 46.18 kB
runtime.js          | runtime | 6.51 kB
                    | Initial Total | 2.64 MB

Build at: 2023-11-06T22:50:28.618Z - Hash: 11586f32cb93ac2d - Time: 25432ms
** Angular Live Development Server is listening on localhost:13346, open your browser on http://localhost:13346/ **

✓ Compiled successfully.
```

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



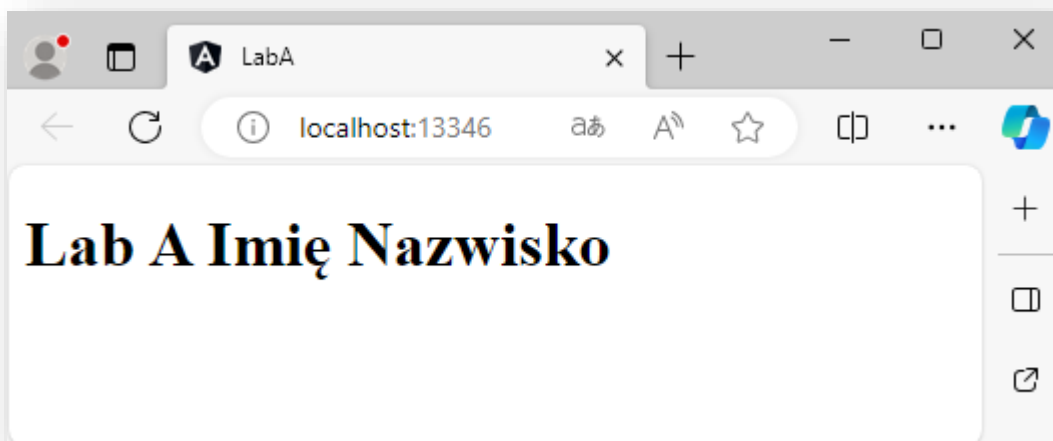
Otwórz projekt w wybranym przez siebie IDE. Edytuj plik `src/app/app.component.html` do postaci:

```
app.component.html x
1 <h1>{{title}}</h1>
2
```

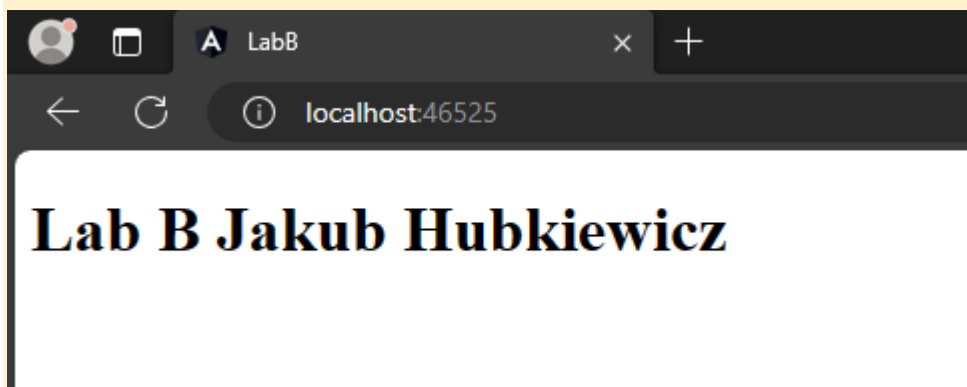
Następnie zmień tytuł w klasie AppComponent w pliku `src/app/app.component.ts`:

```
app.component.ts x
1  import { Component } from '@angular/core';
2
3  5+ usages  Artur Karczmarczyk *
4  @Component({
5    selector: 'app-root',
6    templateUrl: './app.component.html',
7    styleUrls: ['./app.component.css']
8  })
9  export class AppComponent {
10   title : string = 'Lab A Imię Nazwisko';
11 }
```

Strona zostanie automatycznie zaktualizowana:



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



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

DODAWANIE KOMPONENTÓW I SERWISU

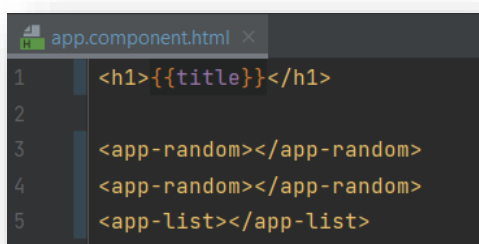
Kolejnym etapem prac jest wygenerowanie (w sposób automatyczny) dwóch komponentów (RandomComponent i ListComponent) oraz serwisu RandomService. W tym celu wykonaj poniższe 3 polecenia:

```
> 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):

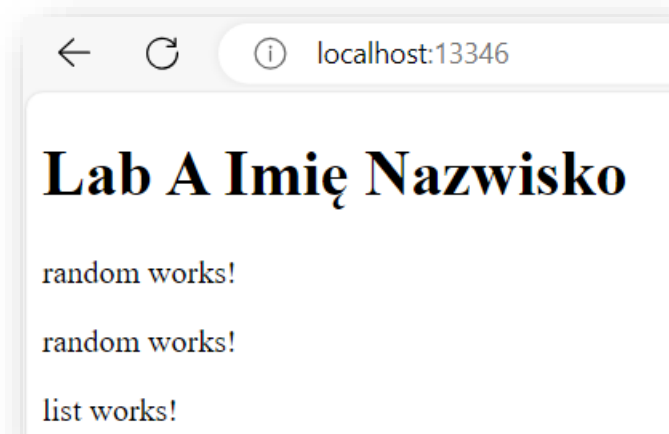
```
PS C:\Users\akarczmarczyk\Desktop\ai2b\lab-a> ng generate component Random
CREATE src/app/random/random.component.html (21 bytes)
CREATE src/app/random/random.component.spec.ts (559 bytes)
CREATE src/app/random/random.component.ts (202 bytes)
CREATE src/app/random/random.component.css (0 bytes)
UPDATE src/app/app.module.ts (396 bytes)
PS C:\Users\akarczmarczyk\Desktop\ai2b\lab-a> ng generate component List
CREATE src/app/list/list.component.html (19 bytes)
CREATE src/app/list/list.component.spec.ts (545 bytes)
CREATE src/app/list/list.component.ts (194 bytes)
CREATE src/app/list/list.component.css (0 bytes)
UPDATE src/app/app.module.ts (470 bytes)
PS C:\Users\akarczmarczyk\Desktop\ai2b\lab-a> ng generate service Random
CREATE src/app/random.service.spec.ts (357 bytes)
CREATE src/app/random.service.ts (135 bytes)
```

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

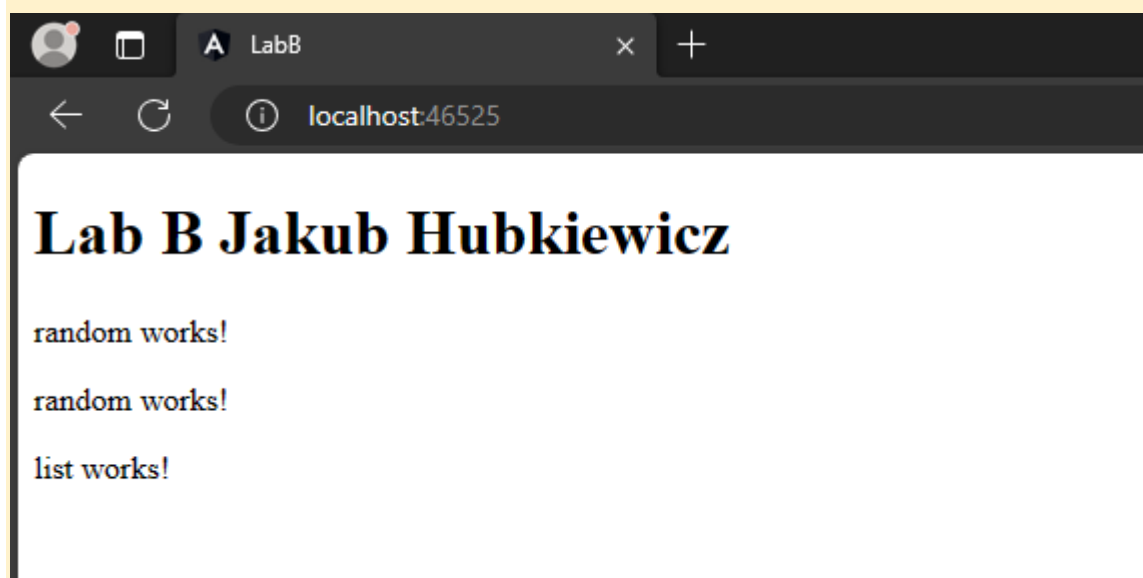


```
1 <h1>{{title}}</h1>
2
3 <app-random></app-random>
4 <app-random></app-random>
5 <app-list></app-list>
```

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.



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:

AI2B LAB B – Hubkiewicz Jakub – Wersja 1

```
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:



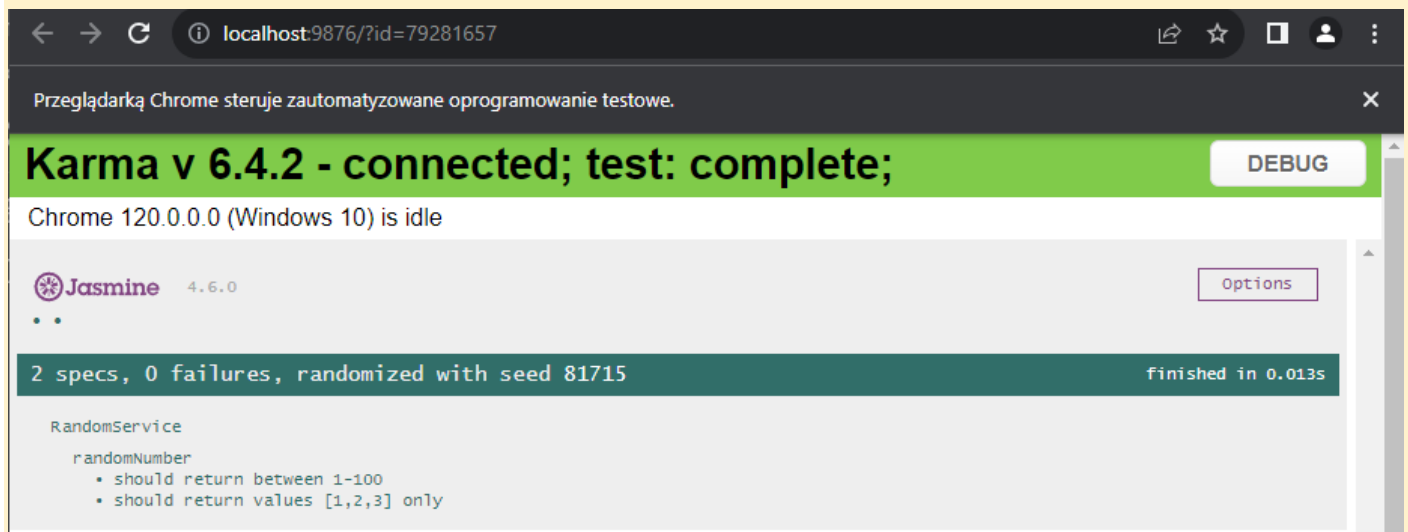
Zmodyfikuj metodę randomNumber, żeby zwracała losową liczbę całkowitą od 1 (włącznie) do max (włącznie). Po każdej edycji serwisu, testy wykonają się ponownie. Ulepszaj swoją metodę aż do momentu, gdy testy przejdą (zielono, 2specs, 0 failures).

Umieść poniżej zrzut ekranu przedstawiający kod metody randomNumber():

```
randomNumber(max: number): number {
  return Math.floor(Math.random() * max + 1); //Losowa liczba od 0 do max (włącznie)
}
```

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

Umieść poniżej zrzut wyniku testu (zakończonego sukcesem):



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

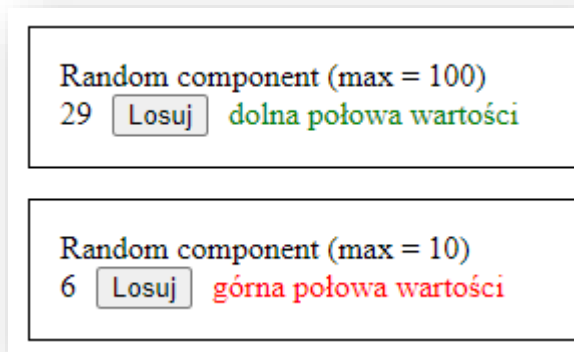
IMPLEMENTACJA KOMPONENTU RANDOM

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 ($< 0.5 * \text{max}$) – wyświetlić zielony napis „dolna połowa wartości”
- większa lub równa od połowy zakresu ($\geq 0.5 * \text{max}$) – wyświetlić czerwony napis „górna połowa wartości”

Przykład:



Przykładowa implementacja komponentu:

```
import {Component, Input, OnInit} from '@angular/core';
import {RandomService} from "../random.service";

@Component({
  selector: 'app-random',
  templateUrl: './random.component.html',
  styleUrls: ['./random.component.css']
})
export class RandomComponent implements OnInit {

  myNumber!: number;
  @Input() max = 10;

  constructor(private randomService: RandomService) { }

  ngOnInit(): void {
    this.myNumber = this.randomService.randomNumber(this.max);
  }

  btnClick() {
    this.myNumber = this.randomService.randomNumber(this.max);
  }

  isSmallerThanHalf(): boolean {
    return this.myNumber <= this.max / 2;
  }
}
```

Przykładowa implementacja widoku:

```
<div id="frame">
  <div>Random component (max = {{max}})</div>
  <span>{{myNumber}}</span>
  <button (click)="btnClick()">Losuj</button>
  <span *ngIf="isSmallerThanHalf()" [class.green]="isSmallerThanHalf()">dolna połowa wartości</span>
  <span *ngIf="!isSmallerThanHalf()" [class.red]="!isSmallerThanHalf()">górną połowa wartości</span>
</div>
```

Przykładowa implementacja stylu:


```
#frame {
  border: 1px solid black;
  padding: 15px;
  margin: 15px;
}

#frame > button {
  margin: 0 10px;
}

.red {
  color: red;
}

.green {
  color: green;
}
```

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:



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

```

random.component.css x
1  #frame{
2    border: 2px dotted black;
3    padding: 10px;
4    margin: 20px;
5  }
6
7  #frame > button{
8    margin: 0 10px;
9    padding: 5px;
10 }
11
12 .red{
13   color: darkred;
14 }
15
16 .green{
17   color: darkgreen;
18 }
19 .none{
20   display: none;
21 }

```

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

```

<> random.component.html x
1  <div id="frame">
2    <div>Random component (max = {{max}})</div>
3    <span>{{myNumber}}</span>
4    <button (click)="btnClick()">Losu, losu</button>
5    <span *ngIf="isSmallerThanHalf()" [class.green]="isSmallerThanHalf()">Dolna połowa wartości</span>
6    <span *ngIf="!isSmallerThanHalf()" [class.red]="!isSmallerThanHalf()">Górna połowa wartości</span>
7  </div>

```

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

```
random.component.ts ×

1  import {Component, Input, OnInit} from '@angular/core';
2  import {RandomService} from "../random.service";
3
4  5+ usages  👤 huuuuubi
5  @Component({
6    selector: 'app-random',
7    templateUrl: './random.component.html',
8    styleUrls: ['./random.component.css']
9  })
10 export class RandomComponent implements OnInit{
11
12   myNumber!: number;
13   @Input() max : number = 10;
14
15   no usages  👤 huuuuubi
16   constructor(private randomService: RandomService) {
17   }
18
19   no usages  👤 huuuuubi
20   ngOnInit(): void {
21     this.myNumber = this.randomService.randomNumber(this.max);
22   }
23
24   1 usage  👤 huuuuubi
25   btnClick() : void {
26     this.myNumber = this.randomService.randomNumber(this.max);
27   }
28
29   4 usages  👤 huuuuubi
30   isSmallerThanHalf():boolean{
31     return this.myNumber <= this.max / 2;
32   }
33 }
```

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

```

<> app.component.html x
1  <h1>{{title}}</h1>
2  <app-random></app-random>
3  <app-random [max]=100></app-random>
4  <app-list></app-list>
5

```

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

IMPLEMENTACJA KOMPONENTU LISTCOMPONENT

Komponent ma umożliwiać podanie w polu formularza dowolnego tekstu. 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:

```

import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-list',
  templateUrl: './list.component.html',
  styleUrls: ['./list.component.css']
})
export class ListComponent implements OnInit {
  elements: string[];
  inputText: string;

  constructor() {
    this.inputText = '';
    this.elements = [];
  }

  ngOnInit(): void {
  }

  inputToArray(): void {
    this.elements.push(this.inputText);
    this.inputText = '';
  }

  remove(index: number): void {
    this.elements.splice(index, deleteCount: 1);
    console.log("remove " + index)
  }
}

```

```

<div id="frame">
  <div>List component</div>
  <div>
    <input type="text" [(ngModel)]="inputText">
    <button (click)="inputToArray()">Dodaj</button>
  </div>
  <ul>
    <li *ngFor="let el of elements; index as i">
      {{el}}
      <a href="#" (click)="remove(i)">usuń</a>
    </li>
  </ul>
</div>

```

Przykładowy wygląd:

List component

- Jeden usuń
- Drugi element usuń
- Kolejny element usuń

oraz widoku

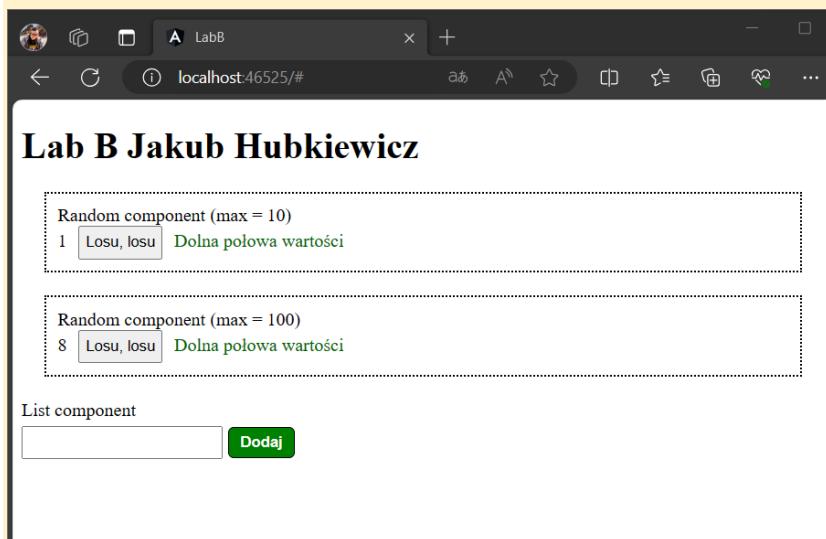
Do app.module.ts będzie trzeba zaimportować:

```

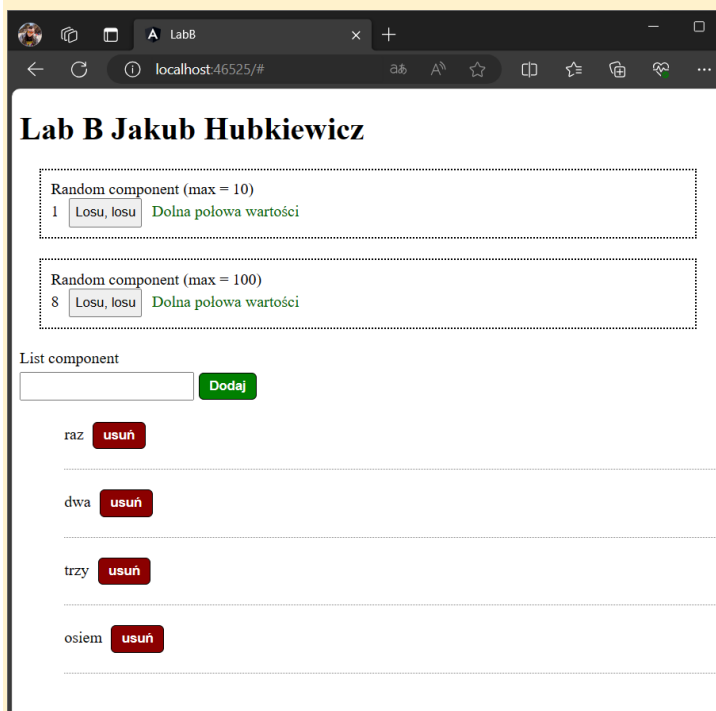
import { FormsModule } from "@angular/forms";
...
imports: [
  BrowserModule, FormsModule
],

```

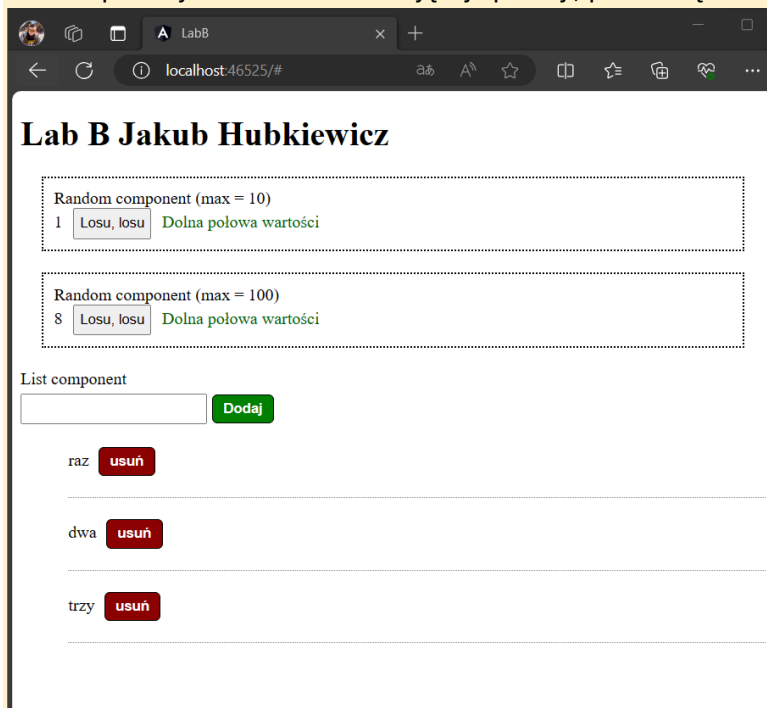
Umieść poniżej zrzut ekranu działającej aplikacji, bez dodanych elementów na liście:



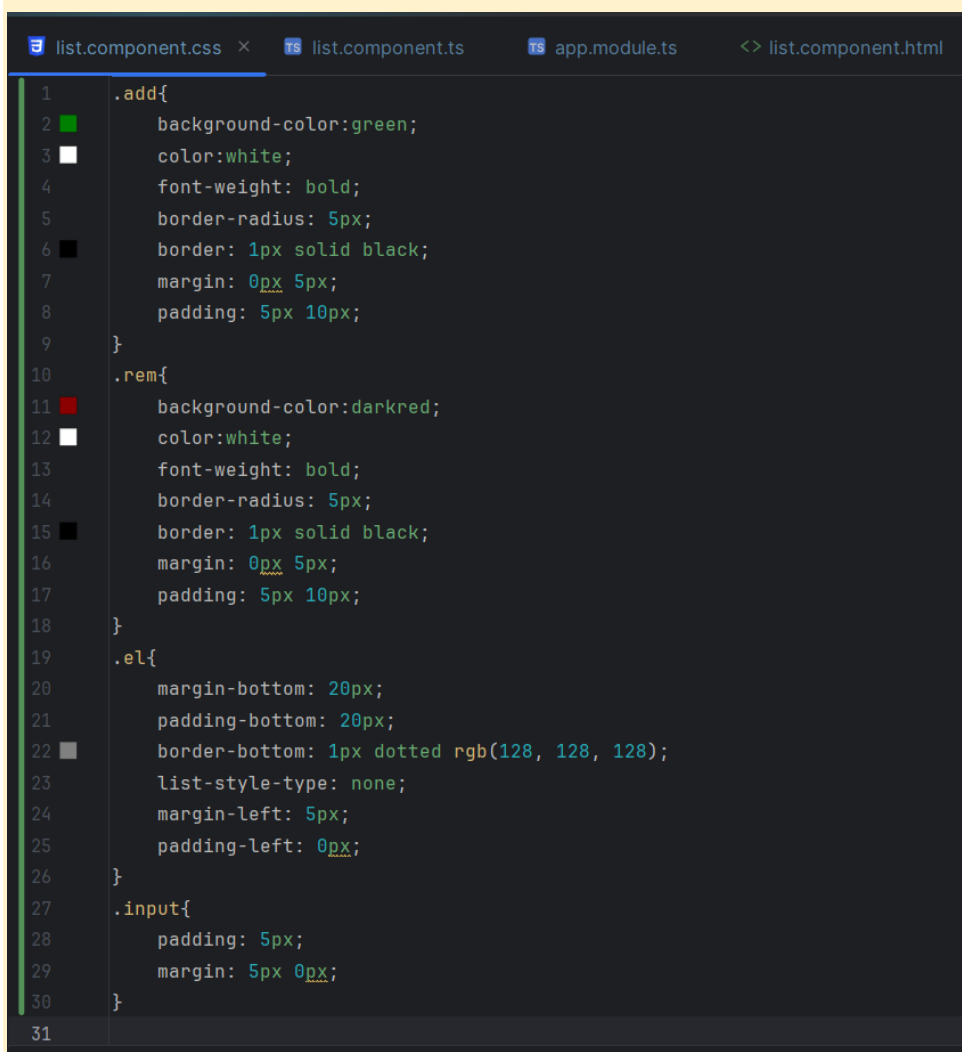
Umieść poniżej zrzut ekranu działającej aplikacji, z dodanymi elementami listy:



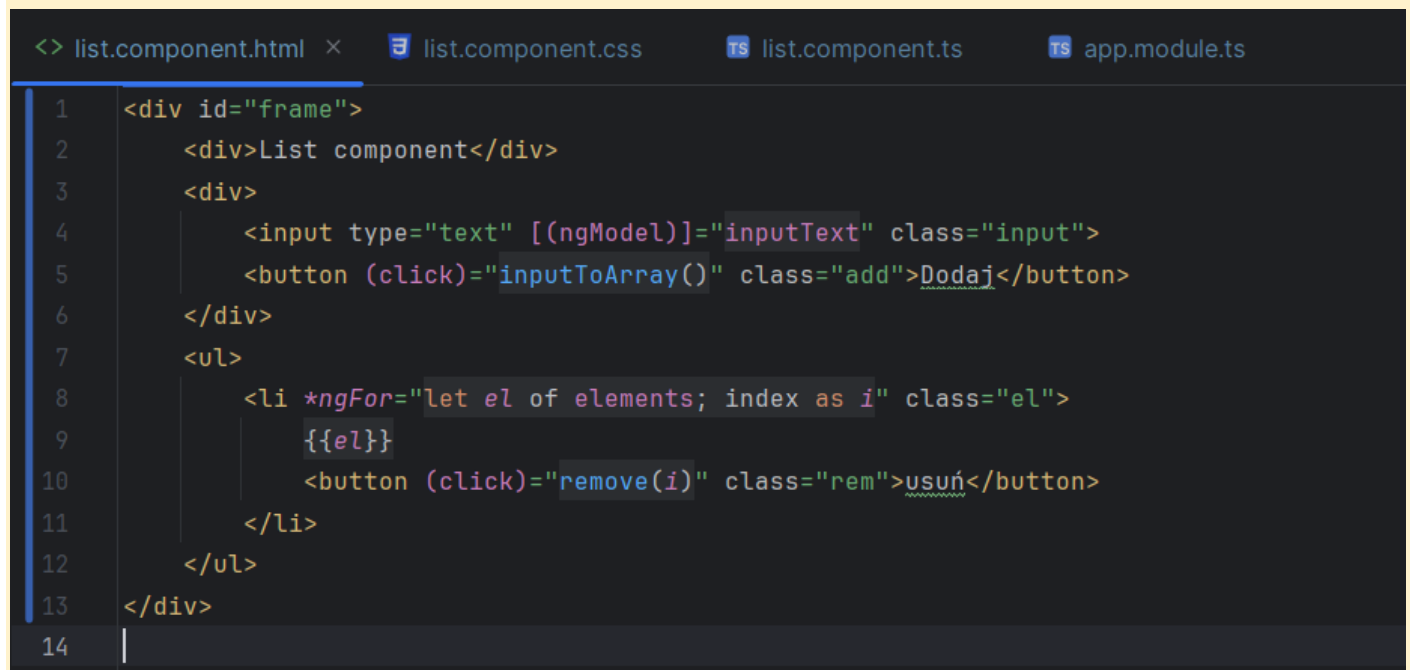
Umieść poniżej zrzut ekranu działającej aplikacji, po usunięciu elementu listy:



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



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



```
1 <div id="frame">
2   <div>List component</div>
3   <div>
4     <input type="text" [(ngModel)]="inputText" class="input">
5     <button (click)="inputToArray()" class="add">Dodaj</button>
6   </div>
7   <ul>
8     <li *ngFor="let el of elements; index as i" class="el">
9       {{el}}
10      <button (click)="remove(i)" class="rem">usuń</button>
11    </li>
12  </ul>
13 </div>
14
```

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

```
1 import {Component, OnInit} from '@angular/core';
2
3 5+ usages huuuubi *
4 @Component({
5   selector: 'app-list',
6   templateUrl: './list.component.html',
7   styleUrls: ['./list.component.css']
8 })
9 export class ListComponent implements OnInit{
10   elements: string[];
11   inputText: string;
12
13   no usages new *
14   constructor() {
15     this.inputText = '';
16     this.elements = [];
17   }
18
19   no usages new *
20   ngOnInit(): void {
21   }
22
23   1 usage new *
24   inputToArray(): void{
25     this.elements.push(this.inputText);
26     this.inputText = '';
27   }
28
29   1 usage new *
30   remove(index: number):void{
31     this.elements.splice(index, deleteCount: 1);
32     console.log("remove " + index);
33   }
34 }
```

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.