

ROUTING I SERWISY W ANGULAR

SPIS TREŚCI

Spis treści	1
Cel zajęć.....	1
Rozpoczęcie	1
Uwaga	2
Inicjalizacja projektu.....	2
Implementacja serwisu przechowującego dane w LocalStorage	6
Implementacja listy danych	8
Dodawanie komponentów i serwisu	Błąd! Nie zdefiniowano zakładki.
Implementacja serwisu RandomService	Błąd! Nie zdefiniowano zakładki.
Implementacja komponentu Random	Błąd! Nie zdefiniowano zakładki.
Implementacja komponentu listComponent	Błąd! Nie zdefiniowano zakładki.
Commit projektu do GIT.....	29
Podsumowanie.....	30

CEL ZAJĘĆ

Celem głównym zajęć jest zdobycie umiejętności:

- przetwarzania danych z zapisem w LocalStorage i z wykorzystaniem serwisów,
- obsługa routingu w Angular.

W praktycznym wymiarze stworzony zostanie prosty projekt pozwalający na dodawanie danych osobowych i zarządzanie nimi w LocalStorage przeglądarki internetowej.

ROZPOCZĘCIE

Rozpoczęcie zajęć. Powtórzenie tworzenia komponentów, serwisów. 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-c
```

Standardowo kreator zapyta o konfigurację routingu (wybrać **Tak**) 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-c
> ng serve --port=00000
```

Tradycyjnie, do pliku **src/styles.css** dodaj znak wodny ze swoim numerem albumu:

```
body {
  background: url("https://placeholder.co/100x100/FFFFFF/EFEFEF/png?text=00000");
}
```

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

Edytuj zawartość pliku **src/app/app.component.html**. Usuń domyślnie wygenerowaną zawartość. Zostaw jedynie znacznik **<router-outlet>**. Utwórz własne, stałe elementy interfejsu aplikacji, np. nagłówek i stopkę. Np.:

```
1  <header>
2    <h1>Local Storage Contact List</h1>
3  </header>
4
5  <div class="router-outlet">
6    <router-outlet></router-outlet>
7  </div>
8
9  <footer>Imie Nazwisko</footer>
10
```

Dodaj style w plikach **src/styles.css** (globalne) i **src/app/app.component.css** wg własnego uznania.

Przykładowy efekt:

Local Storage Contact List

00000 00000 00000 00000 00000 00000 00000 00000 00000

00000 00000 00000 00000 00000 00000 00000 00000 00000

Imię Nazwisko

Następnie dodaj komponenty, pod przyszłe widoki:

- `ListComponent` – wyświetlanie listy osób
- `DetailsComponent` – szczegóły wybranej osoby
- `AddPersonComponent` – dodawanie danych nowej osoby
- `NotFoundComponent` – informacja o błędnej ścieżce routingu.

W tym celu otwórz osobny terminal i wykonaj polecenia:

```
> ng generate component list --skip-tests
> ng generate component details --skip-tests
> ng generate component addPerson --skip-tests
> ng generate component notFound --skip-tests
```

Utworzone zostaną 4 katalogi komponentów.

W automatycznie wygenerowanym module routingu (plik `src/app/app-routing.module.ts`) ustaw ścieżki routingu dodając elementy w tablicy `routes`:

- `list` – powinno kierować do komponentu `ListComponent`
- `details/:id` – powinno kierować do komponentu `DetailsComponent`, ścieżka zawiera parametr
- `add` – powinno kierować do komponentu `AddPersonComponent`

Dodatkowo należy ustawić domyślne przekierowanie ze ścieżki `""` (pusta) na `"/list"` oraz ze ścieżki `""` (dowolna inna) na komponent `NotFoundComponent`.

Ponadto na widoku komponentu `DetailsComponent` należy wyświetlić bieżące ID (patrz. Lab A oraz wykłady).

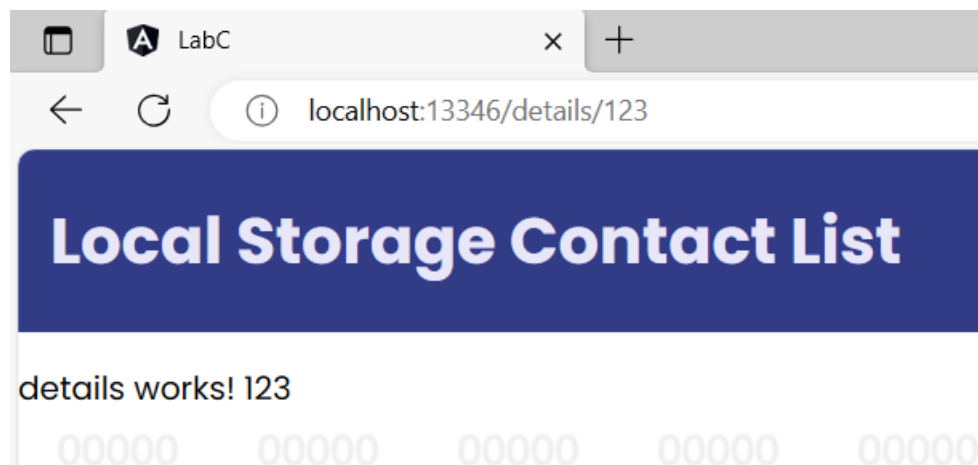
Przykładowa implementacja routingu:

```
8   const routes: Routes = [
9     {path: "list", component: ListComponent},
10    {path: "details/:id", component: DetailsComponent},
11    {path: "add", component: AddPersonComponent},
12    {path: "", redirectTo: "/list", pathMatch: "full"},
13    {path: "**", component: NotFoundComponent},
14  ];
```

Przykładowa subskrypcja parametru ID w `DetailsComponent`:

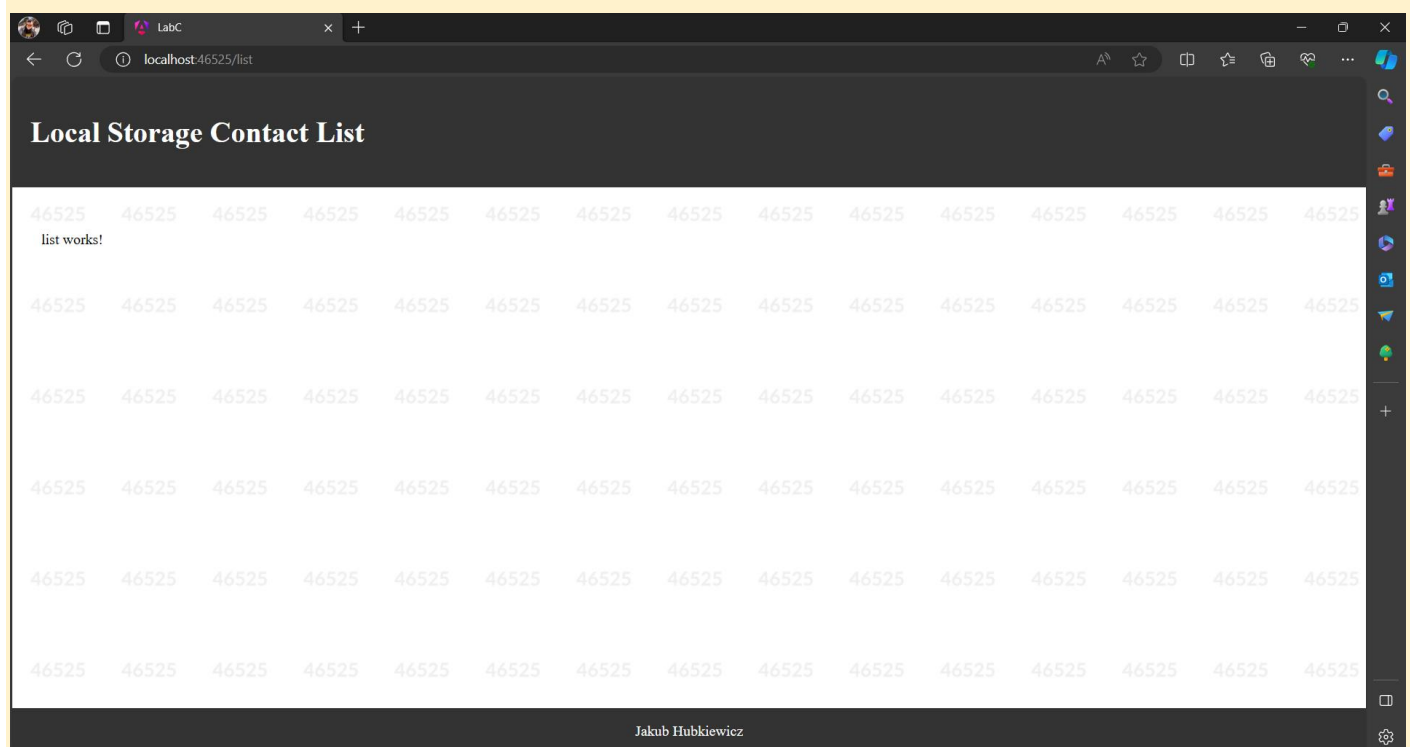
```
ngOnInit(): void {  
  this.route.params.subscribe( observerOrNext: params : Params => {  
    | this.personId = params['id'];  
  });  
}
```

Przykładowy wygląd strony /details/123:

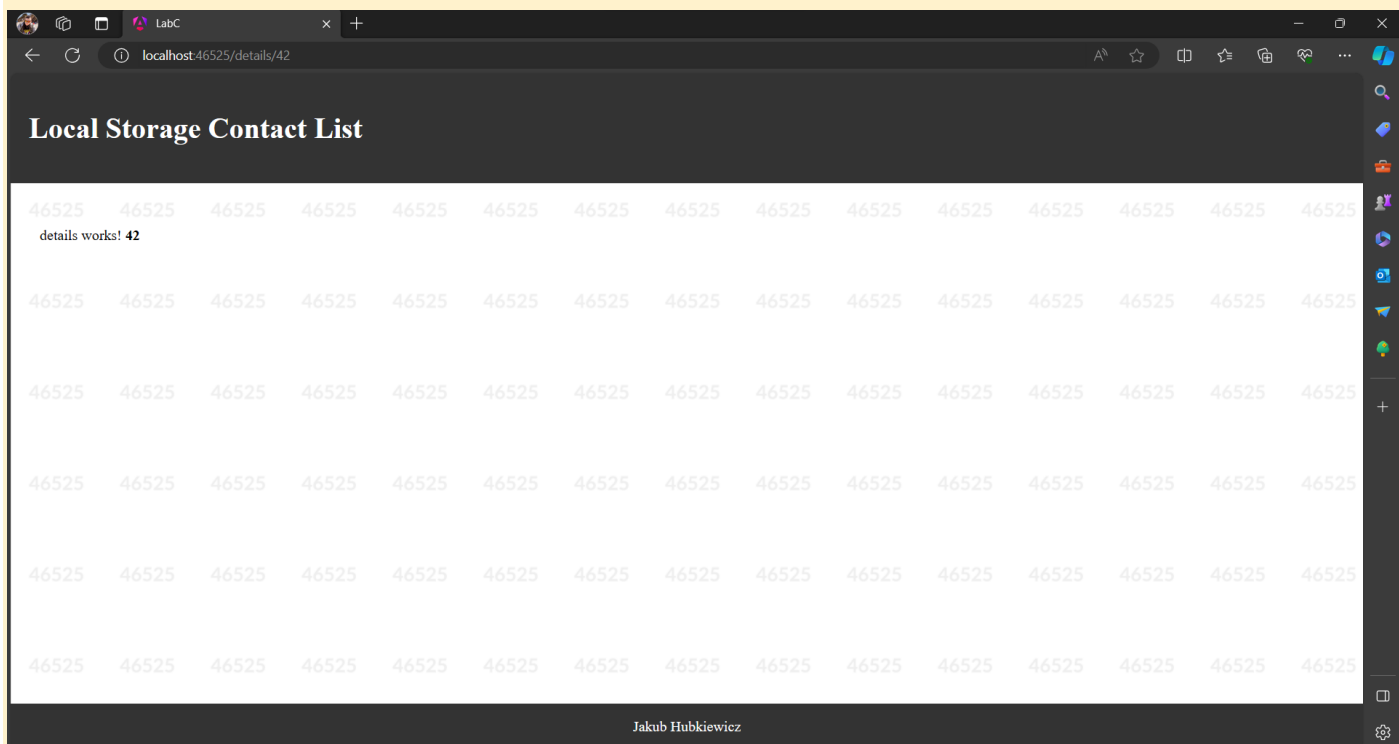


W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/f6902ba9fd002753586a92e61f5575ea43f39aed>

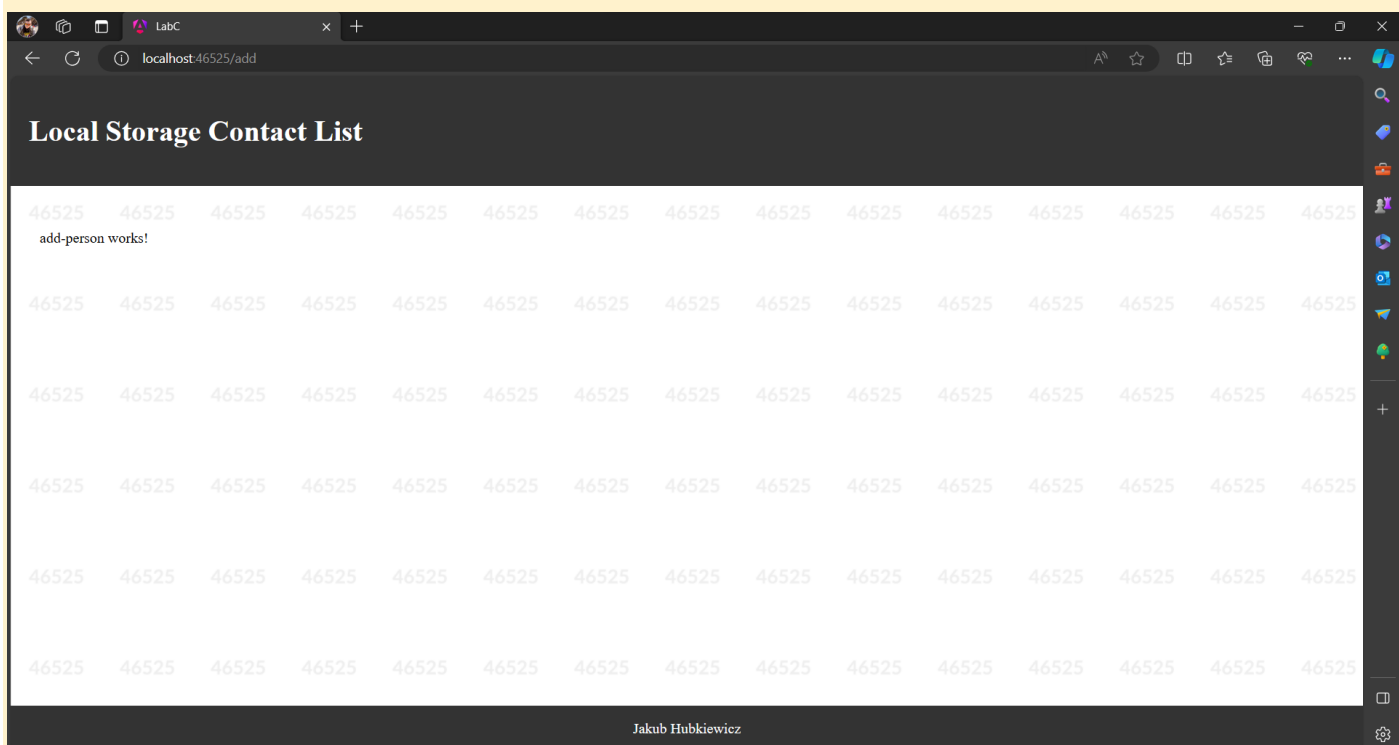
Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę /list. Upewnij się, że na zrzucie ekranu widoczny jest adres URL oraz domyślny szablon „list works!”



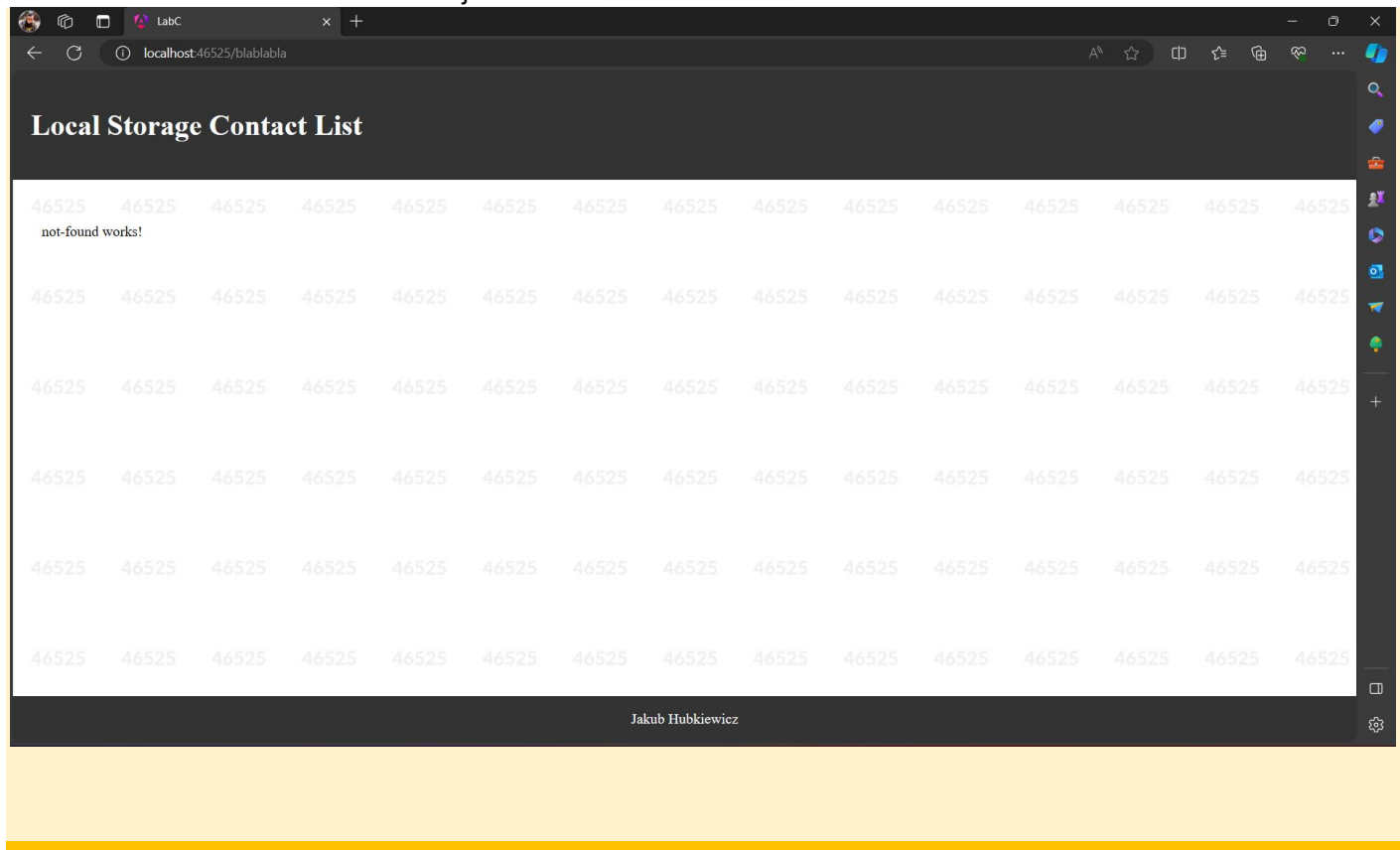
Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę /details/42. Upewnij się, że na zrzucie ekranu widoczny jest adres URL oraz wyświetlona w komponencie DetailsComponent wartość 42 (z URLa).



Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę `/add`. Upewnij się, że na zrzucie ekranu widoczny jest adres URL oraz domyślny szablon „add-person works!”



Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę `/not-found` (lub dowolny inny nieistniejący). Upewnij się, że na zrzucie ekranu widoczny jest adres URL oraz domyślny szablon „not - found works!”



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

IMPLEMENTACJA SERWISU PRZECHOWUJĄCEGO DANE W LOCALSTORAGE

Utwórz interfejs reprezentujący dane osobowe przetwarzane w aplikacji komendą:

```
> ng generate interface person
```

Utworzony zostanie plik `src/app/person.ts`. Uzupełnij jego zawartość, przykładowo:

```
1 export interface Person {
2   firstName?: string;
3   lastName?: string;
4   age?: number;
5   address: {
6     city?: string;
7     street?: string;
8     postCode?: string;
9   }
10 }
```

Następnie użyj terminala, żeby wygenerować serwis `PersonLsService`:

```
> ng generate service personLs --skip-tests
```

Utworzony zostanie plik `src/app/person-ls.service.ts`. Podmień jego zawartość na poniższą:

AI2B LAB C – Hubkiewicz Jakub – Wersja 1

```
import { Injectable } from '@angular/core';
import { Person } from './person';

@Injectable({
  providedIn: 'root'
})
export class PersonLsService {
  readonly KEY = 'stored-people-data';
  constructor() { }

  // public getAll(): Person[] {
  //   // set response people variable to an empty array
  //   // get data from localStorage
  //   // if anything was fetched, parse using JSON.parse() and assign to people
  //   // return all people
  // }
  //
  // public getPerson(index: number): Person {
  //   // get all people and return the one at [index] position
  // }
  //
  // public addPerson(person: Person): void {
  //   // get all people
  //   // push person to the array of people
  //   // update localStorage with the array contents serialized using JSON.stringify()
  // }
  //
  // public deletePerson(index: number): void {
  //   // get all people
  //   // use splice() to remove person at [index] position
  //   // update localStorage with the new values of people array
  // }
}
```

W powyższym kodzie przygotowano sygnatury 4 metod:

- do pobierania listy wszystkich osób z local storage,
- do pobierania osoby pod określonym indeksem w tablicy wszystkich osób
- do dodawania nowej osoby
- do kasowania osoby pod zadany indeksem.

Odkomentuj te funkcje i wypełnij ich definicje. W przypadku problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/6ded9bb153ee349e74f28c595cf2ea3ea6211864#diff-2f7e27c391409db46653869eef8522a755ecad6e46789b4cd8223b9be5551936>

Umieść poniżej zrzut ekranu kodu metody `getAll()`:

```
1+ usages new *
public getAll(): Person[] {
  let people: Person[] = [];
  let data :string | null = localStorage.getItem(this.KEY);
  if (data) {
    people = JSON.parse(data) || [];
  }
  return people;
}
```

Umieść poniżej zrzut ekranu kodu metody `getPerson()`:

```
no usages new *
public getPerson(index: number): Person {
  const people :Person[] = this.getAll();
  return people[index];
}
```

Umieść poniżej zrzut ekranu kodu metody `addPerson()`:

```
no usages new *
public addPerson(person: Person): void {
  let people :Person[] = this.getAll();
  people.push(person);
  localStorage.setItem(this.KEY, JSON.stringify(people));
}
```

Umieść poniżej zrzut ekranu kodu metody `deletePerson()`:

```
no usages new *
public deletePerson(index: number): void {
  let people :Person[] = this.getAll();
  people.splice(index, deleteCount: 1);
  localStorage.setItem(this.KEY, JSON.stringify(people));
}
```

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

IMPLEMENTACJA LISTY DANYCH

Korzystając z narzędzi deweloperskich aplikacji (karta Application -> Local Storage) ustaw przykładowe dane do pobrania przez listę (klucz zgodny z tym określonym w serwisie PersonLsService):

```
[
  {
    "firstName": "Alice",
    "lastName": "Alyska",
    "age": 21,
    "address": {
      "city": "Szczecin",
      "street": "Zolnierska 49",
      "postCode": "71-210"
    }
  },
  {
    "firstName": "Bob",
    "lastName": "Bobsky",
```



```

    "age": 22,
    "address": {
      "city": "Szczecin",
      "street": "Zolnierska 49",
      "postCode": "71-210"
    }
  }
]

```

Następnie zaimplementuj wyświetlanie listy osób. Dla każdej osoby wyświetlaj imię, nazwisko oraz identyfikator na liście. Każda pozycja musi być jednocześnie łączem do ekranu szczegółów danej osoby, przykładowo:

Local Storage Contact List

- [Alice Alyska \(0\)](#).
- [Bob Bobsky \(1\)](#).

Podczas implementacji, w komponencie `ListComponent` należy zainicjalizować tablicę elementów typu `Person[]`, wstrzyknąć serwis i w metodzie `ngOnInit()` wczytać za jego pomocą listę wszystkich osób (metoda `getAll()`).

Na widoku należy użyć dyrektywy `*ngFor`, żeby wyświetlić element listy dla każdej osoby z pozyskanej tablicy osób.

Przykładowa implementacja:

```

1  import {Component, OnInit} from '@angular/core';
2  import {Person} from "../person";
3  import {PersonLsService} from "../person-ls.service";
4
5  @Component({
6    selector: 'app-list',
7    templateUrl: './list.component.html',
8    styleUrls: ['./list.component.css']
9  })
10 export class ListComponent implements OnInit {
11   people: Person[] = [];
12
13   constructor(
14     private personLsService: PersonLsService,
15   ) {
16   }
17
18   ngOnInit(): void {
19     this.people = this.personLsService.getAll();
20   }
21 }

```

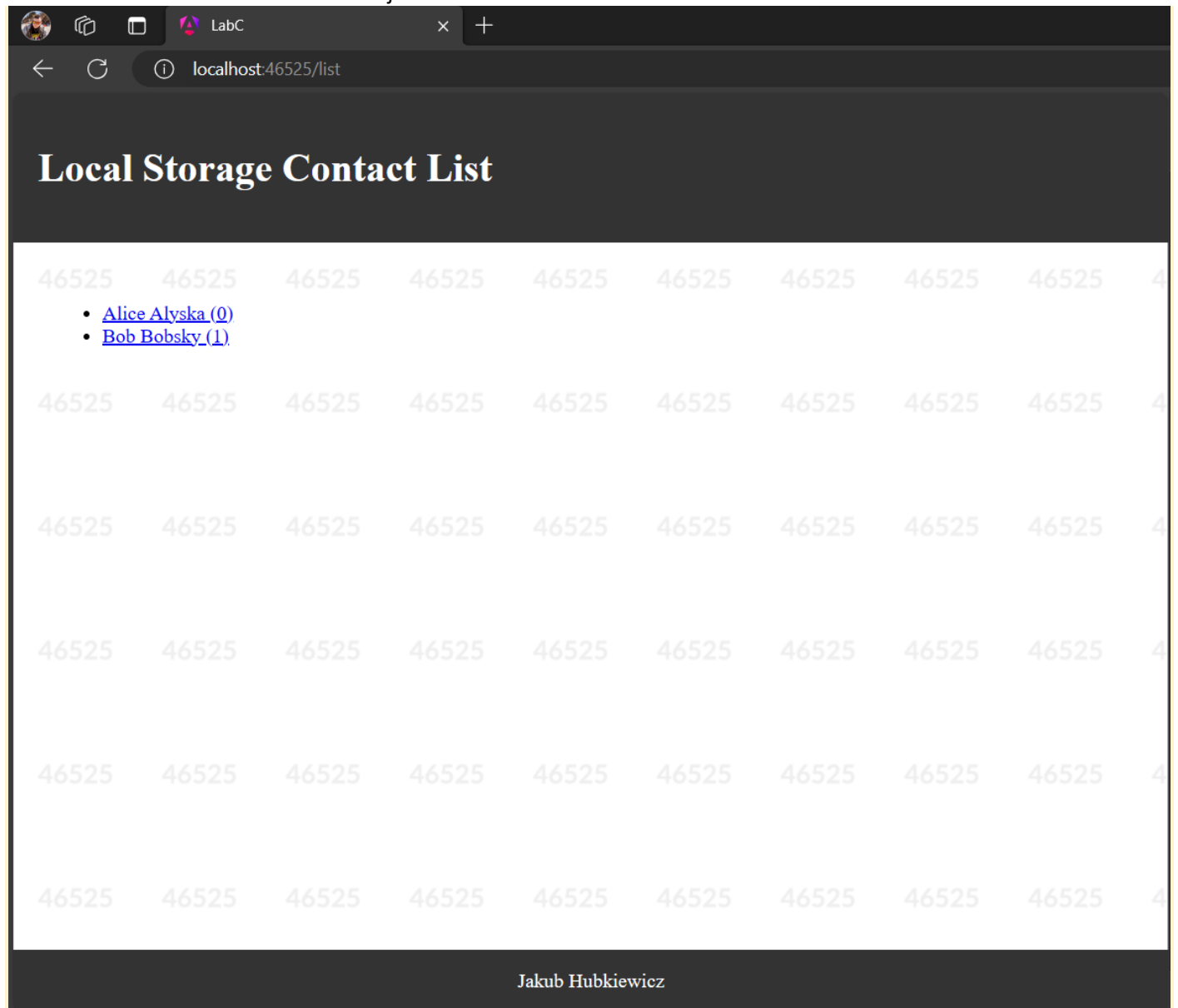
```

1 <ul>
2 <li *ngFor="let person of people; let i = index">
3   <a [routerLink]="['/details', i]">
4     {{ person.firstName }} {{ person.lastName }} ({{ i }})
5   </a>
6 </li>
7 </ul>

```

W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/6ded9bb153ee349e74f28c595cf2ea3ea6211864>

Umieść poniżej zrzut ekranu przeglądarki z wyświetloną listą osób:



Umieść poniżej zrzut ekranu kodu pliku `src/app/list/list.component.ts`:

```

1  import {Component, OnInit} from '@angular/core';
2  import {Person} from "../person";
3  import {PersonLsService} from "../person-ls.service";
4  import {RouterLink} from "@angular/router";
5  import {NgForOf} from "@angular/common";
6
7  1+ usages  new *
8  @Component({
9      selector: 'app-list',
10     templateUrl: './list.component.html',
11     standalone: true,
12     imports: [
13         RouterLink,
14         NgForOf
15     ],
16     styleUrls: ['./list.component.css']
17 })
18 export class ListComponent implements OnInit {
19     people: Person[] = [];
20
21     no usages  new *
22     constructor(
23         private personLsService: PersonLsService,
24     ) {
25     }
26
27
28
29

```

```

24
25     no usages  new *
26     ngOnInit() : void {
27         this.people = this.personLsService.getAll();
28     }
29
30

```

Umieść poniżej zrzut ekranu kodu pliku `src/app/list/list.component.html`:

```

1      <ul>
2          <li *ngFor="let person of people; let i = index">
3              <a [routerLink]="['/details', i]">
4                  {{ person.firstName }} {{ person.lastName }} ({{ i }})
5              </a>
6          </li>
7      </ul>

```

Punkty:

0

1

IMPLEMENTACJA KOMPONENTU SZCZEGÓŁÓW

W tej sekcji zaimplementujemy komponent `DetailsComponent`. Wstrzyknij serwis i wczytaj z local storage pojedynczy element `Person` wskazany wartością przekazaną w ścieżce routingu. W widoku dokonaj interpolacji uzyskanych danych, stylując wedle uznania.

Przykładowa implementacja:

Local Storage Contact List

[Back](#)

Bob Bobsky	
ID	1
Age:	22
City:	Szczecin
Street	Zolnierska 49
Postcode	71-210

Imię Nazwisko

```

import ...

4 usages  Artur Karczmarczyk *
@Component({
  selector: 'app-details',
  templateUrl: './details.component.html',
  styleUrls: ['./details.component.css']
})
export class DetailsComponent implements OnInit {
  personId?: number;
  person?: Person;

  no usages  Artur Karczmarczyk *
  constructor(
    private route: ActivatedRoute,
    private personLsService: PersonLsService,
  ) {
  }

  no usages  Artur Karczmarczyk *
  ngOnInit(): void {
    this.route.params.subscribe( observerOnNext: params : Params => {
      this.personId = params['id'];
      this.person = this.personId
        ? this.personLsService.getPerson(this.personId)
        : undefined;
    });
  }
}

```

```

<div class="details-container">

<div>
  <a [routerLink]="['/'&laquo; Back</a>
</div>

<div *ngIf="person" class="person">
  <h2>{{person.firstName}} {{person.lastName}}</h2>
  <dl class="info">
    <dt>ID</dt>
    <dd>{{person.id}}</dd>

    <dt>Age:</dt>
    <dd>{{person.age}}</dd>

  </dl>

  <dl *ngIf="person.address" class="address">
    <dt>City:</dt>
    <dd>{{person.address.city}}</dd>

    <dt>Street</dt>
    <dd>{{person.address.street}}</dd>

    <dt>Postcode</dt>
    <dd>{{person.address.postCode}}</dd>
  </dl>

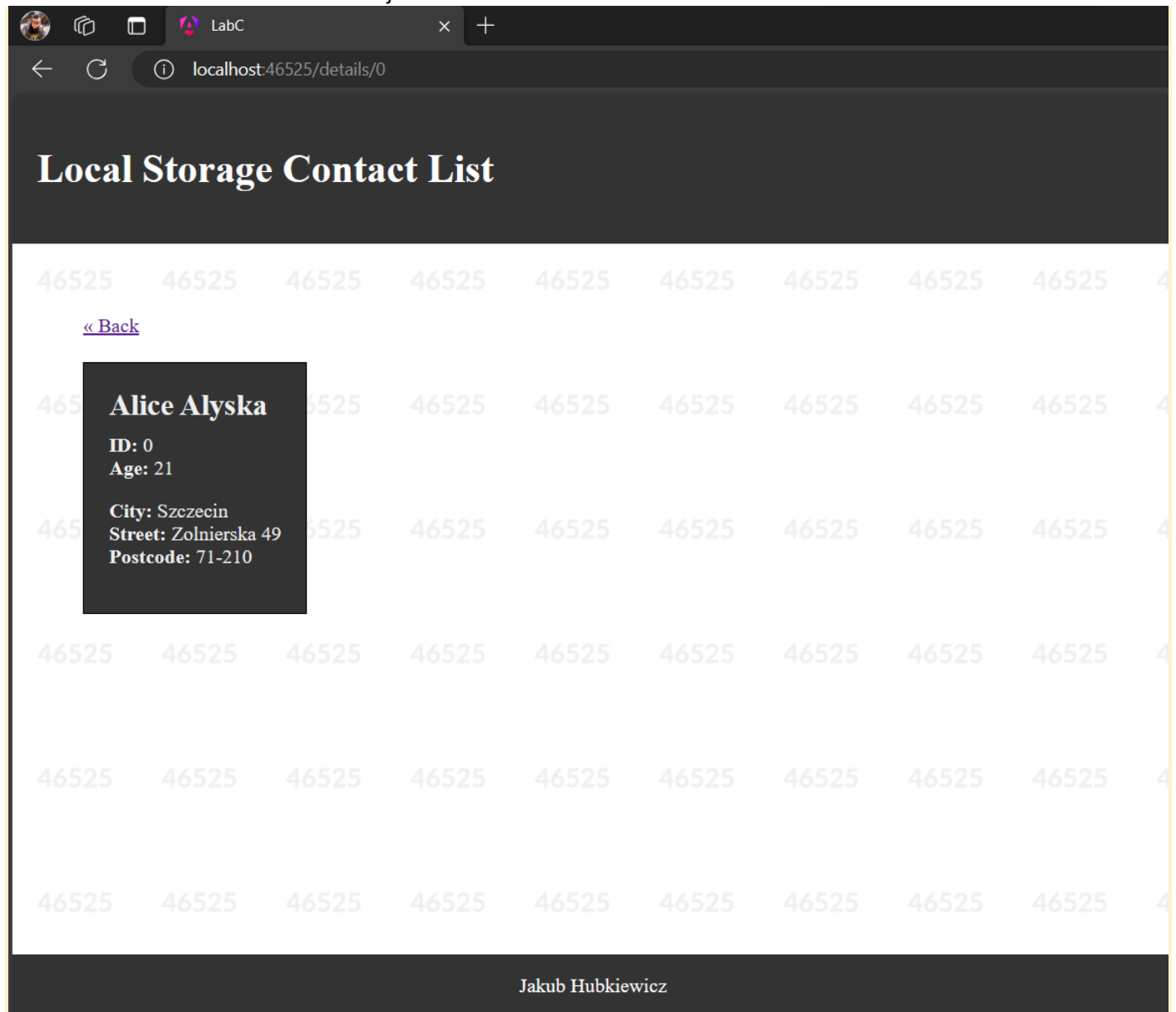
  <div class="alert" *ngIf="!person">
    Person not found!
  </div>

</div>

```

W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/56736f14c7baf280d04bf5f4a72e510a192dada2>

Umieść poniżej zrzut ekranu przeglądarki z wyświetlonymi szczegółami jednej osoby:



Umieść poniżej zrzut ekranu kodu pliku `src/app/details/details.component.ts`:

```

ts details.component.ts × details.component.css <> details.component.html
1 import {Component, OnInit} from '@angular/core';
2 import {ActivatedRoute, RouterLink} from "@angular/router";
3 import {Person} from "../person";
4 import {PersonLsService} from "../person-ls.service";
5 import {NgIf} from "@angular/common";
6
7 1+ usages new *
8 @Component({
9   selector: 'app-details',
10  templateUrl: './details.component.html',
11  standalone: true,
12  imports: [
13    RouterLink,
14    NgIf
15  ],
16  styleUrls: ['./details.component.css']
17 })
18 export class DetailsComponent implements OnInit{
19   personId?: number;
20   person?: Person;
21
22   no usages new *
23   constructor(
24     private route: ActivatedRoute,
25     private personLsService: PersonLsService,
26   ) {
27   }
28
29   no usages new *
30   ngOnInit() : void {
31     this.route.params.subscribe( observerOrNext: params : Params => {
32       this.personId = params['id'];
33       this.person = this.personId
34         ? this.personLsService.getPerson(this.personId)
35         : undefined;
36     });
37   }
38 }

```

Umieść poniżej zrzut ekranu kodu pliku `src/app/details/details.component.html`:

```
TS details.component.ts  details.component.css  <> details.component.html x
1  <div class="details-container">
2    <div>
3      <a [routerLink]="'/'">&laquo; Back</a>
4    </div>
5    <div *ngIf="person" class="person">
6      <h2>{{person.firstName}} {{person.lastName}}</h2>
7      <span class="info">
8        <p><b>ID: </b>{{personId}}</p>
9        <p><b>Age: </b>{{person.age}}</p>
10     </span>
11
12     <p *ngIf="person.address" class="address">
13       <b>City: </b>
14       {{person.address.city}}<br>
15       <b>Street: </b>
16       {{person.address.street}}<br>
17       <b>Postcode: </b>
18       {{person.address.postCode}}
19     </p>
20   </div>
21
22   <div class="alert" *ngIf="!person">
23     Person not found!
24   </div>
25 </div>
26
```

Umieść poniżej zrzut ekranu kodu pliku `src/app/details/details.component.css`:

```

1  .details-container {
2      padding: 25px;
3  }
4
5  .person {
6      display: flex;
7      flex-direction: column;
8      align-items: flex-start;
9      background-color: #333333;
10     color: #efefef;
11     border: 1px solid black;
12     padding: 20px;
13     margin: 20px 0;
14     max-width: 320px;
15     box-sizing: border-box;
16 }
17
18 .person h2 {
19     padding: 0;
20     margin: 0;
21     font-weight: bold;
22 }
23
24 .info > p:nth-child(1){
25     margin-top: 10px;
26 }
27
28 .info > p{
29     margin: 0;
30 }
31

```

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

IMPLEMENTACJA DODAWANIA DANYCH

Gdzieś na stronie umieść link nawigujący do widoku dodania nowego elementu danych `/add`.

W komponencie `AddPersonComponent` zdefiniuj pole typu `Person` reprezentujące nowe dane osobowe i zainicjalizuj je. Przykład inicjalizacji:


```
person: Person = {
  address: {}
};
```

Wstrzyknij opracowany wcześniej serwis oraz obiekt typu Router potrzebny do przełączenia widoku po pomyślnym zapisie danych. Dodaj metodę `save()`, która po wywołaniu zapisze dane osobowe z pola `person` do local storage za pomocą metody `addPerson()` serwisu, a następnie przekieruje użytkownika na listę.

Mając skończoną implementację części komponentowej, w widoku `src/app/add-person/add-person.component.html` zbuduj formularz do uzupełnienia danych osoby i jej adresu. Pola formularza powiąż dwukierunkowo z polami obiektu `person` za pomocą `[(ngModel)]`, np. `[(ngModel)]="person.firstName"`.

Dodaj także przycisk zapisywania, który wykona metodę `save()` komponentu.

Pamiętaj o zaimportowaniu brakujących modułów (w tym `FormsModule`) w `src/app/app.module.ts`.

Przykładowa implementacja:

Local Storage Contact List Add Person »

First Name

Last Name

Age

City

Street

Postcode

```
import ...

4 usages  ▸ Artur Karczmarczyk *
@Component({
  selector: 'app-add-person',
  templateUrl: './add-person.component.html',
  styleUrls: ['./add-person.component.css']
})
export class AddPersonComponent implements OnInit {
  person: Person = {
    address: {},
  }

  no usages  new *
  constructor(
    private personLsService: PersonLsService,
    private router: Router,
  ) {
  }

  1 usage  new *
  save(): void {
    this.personLsService.addPerson(this.person);
    this.router.navigate([ '/list' ]);
  }

  no usages  new *
  ngOnInit(): void {
  }
}
```

```
<form>
  <div class="form-row">
    <label for="firstName">First Name</label>
    <input type="text" id="firstName" name="firstName" [(ngModel)]="person.firstName">
  </div>

  <div class="form-row">
    <label for="lastName">Last Name</label>
    <input type="text" id="lastName" name="lastName" [(ngModel)]="person.lastName">
  </div>

  <div class="form-row">
    <label for="age">Age</label>
    <input type="number" id="age" name="age" min="1" max="110" step="1" [(ngModel)]="person.age">
  </div>

  <div class="address">
    <div class="form-row">
      <label for="city">City</label>
      <input type="text" id="city" name="city" [(ngModel)]="person.address.city">
    </div>

    <div class="form-row">
      <label for="street">Street</label>
      <input type="text" id="street" name="street" [(ngModel)]="person.address.street">
    </div>

    <div class="form-row">
      <label for="postCode">Postcode</label>
      <input type="text" id="postCode" name="postCode" [(ngModel)]="person.address.postCode">
    </div>
  </div>

  <button type="button" (click)="save()">Save</button>
</form>
```

W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/f71256ddb3f97dc16d16b2e2402d8aed3e0f01a9>

Umieść poniżej zrzut ekranu przeglądarki z ekranem dodawania osoby:

46525 46525 46525 46525 46525 46525 46525 46525 46525 46525

First Name

Last Name

Age

City

Street

Postcode

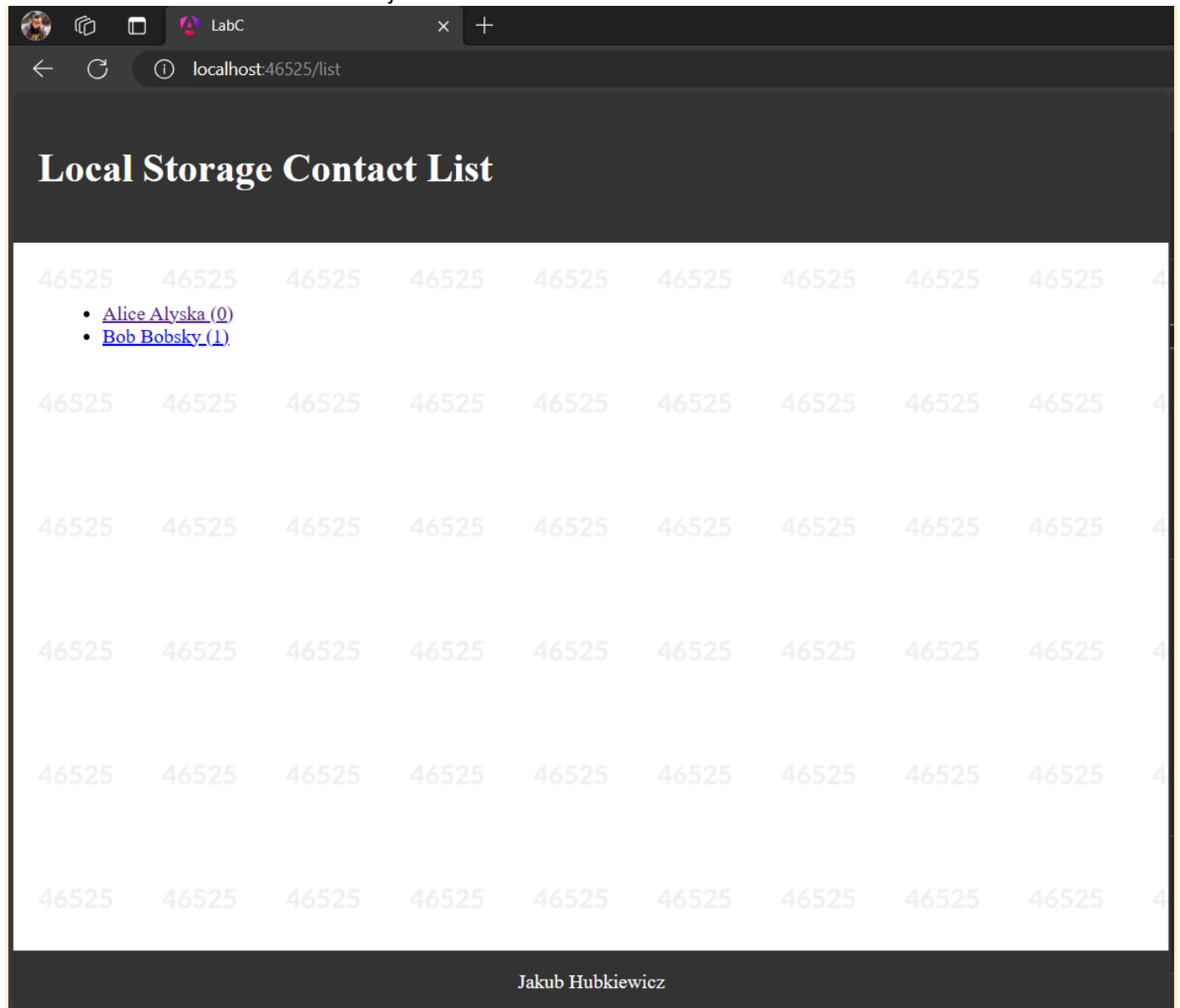
46525 46525 46525 46525 46525 46525 46525 46525 46525 46525

46525 46525 46525 46525 46525 46525 46525 46525 46525 46525

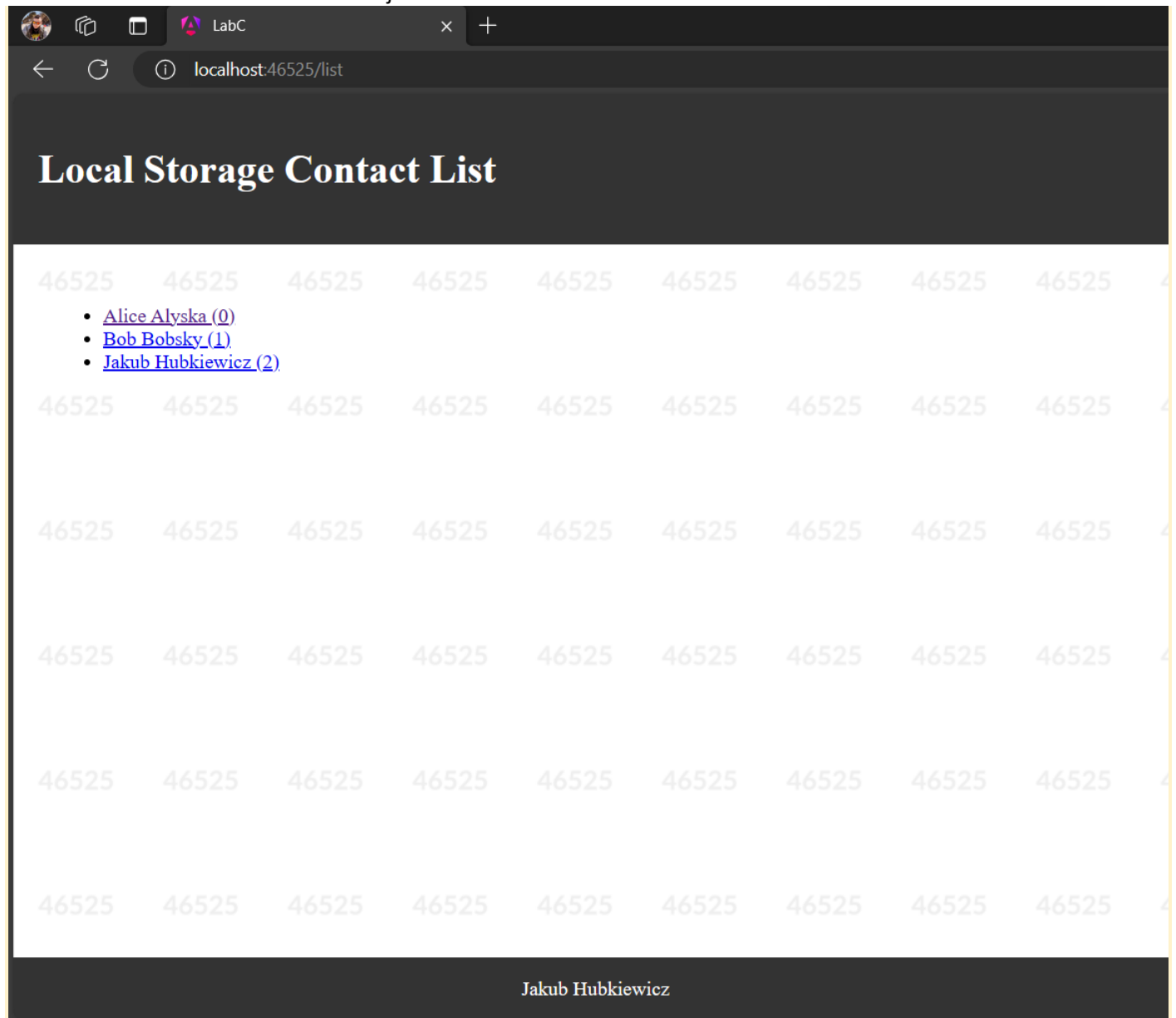
46525 46525 46525 46525 46525 46525 46525 46525 46525 46525

Jakub Hubkiewicz

Umieść poniżej zrzut ekranu listy przed dodaniem osoby:



Umieść poniżej zrzut ekranu listy po dodaniu osoby:



Umieść poniżej zrzut ekranu kodu pliku `src/app/add-person/add-person.component.ts`:

```

ts add-person.component.ts × <> add-person.component.html add-person.component.css
1 import {Component, OnInit} from '@angular/core';
2 import {Person} from "../person";
3 import {PersonLsService} from "../person-ls.service";
4 import {Router} from "@angular/router";
5 import {FormsModule} from "@angular/forms";
6
7 1+ usages new *
8 @Component({
9   selector: 'app-add-person',
10  templateUrl: './add-person.component.html',
11  standalone: true,
12  imports: [
13    FormsModule
14  ],
15  styleUrls: ['./add-person.component.css']
16 })
17 export class AddPersonComponent implements OnInit{
18   person: Person = {
19     address: {},
20   }
21
22   no usages new *
23   constructor(
24     private personLsService: PersonLsService,
25     private router: Router,
26   ) {
27
28   }
29
30   1+ usages new *
31   save() : void {
32     this.personLsService.addPerson(this.person);
33     this.router.navigate(commands: ['/list']);
34   }
35
36   no usages new *
37   ngOnInit() : void {
38
39   }
40 }

```

Umieść poniżej zrzut ekranu kodu pliku `src/app/add-person/add-person.component.html`:

```

1  <form>
2    <div class="form-row">
3      <label for="firstName">First Name</label>
4      <input type="text" id="firstName" name="firstName" [(ngModel)]="person.firstName">
5    </div>
6
7    <div class="form-row">
8      <label for="lastName">Last Name</label>
9      <input type="text" id="lastName" name="lastName" [(ngModel)]="person.lastName">
10   </div>
11
12   <div class="form-row">
13     <label for="age">Age</label>
14     <input type="number" id="age" name="age" min="1" max="110" step="1" [(ngModel)]="person.age">
15   </div>
16
17   <div class="address">
18     <div class="form-row">
19       <label for="city">City</label>
20       <input type="text" id="city" name="city" [(ngModel)]="person.address.city">
21     </div>
22
23     <div class="form-row">
24       <label for="street">Street</label>
25       <input type="text" id="street" name="street" [(ngModel)]="person.address.street">
26     </div>
27
28     <div class="form-row">
29       <label for="postCode">Postcode</label>
30       <input type="text" id="postCode" name="postCode" [(ngModel)]="person.address.postCode">
31     </div>
32   </div>
33
34   <button type="button" (click)="save()">Save</button>
35 </form>

```

Umieść poniżej zrzut ekranu kodu pliku `src/app/add-person/add-person.component.css`:

The screenshot shows a code editor with three tabs: 'add-person.component.ts', 'add-person.component.html', and 'add-person.component.css'. The 'add-person.component.css' tab is active, displaying the following CSS code:

```

1  form {
2      display: flex;
3      flex-direction: column;
4      gap: 10px;
5      max-width: 600px;
6      padding: 10px;
7  }
8
9  .address {
10     display: flex;
11     flex-direction: column;
12     gap: 10px;
13     background-color: #efefef;
14     padding: 10px 0;
15 }
16
17 button {
18     max-width: 200px;
19 }
20
21 .form-row {
22     display: grid;
23     gap: 10px;
24     grid-template-columns: 1fr 3fr;
25 }
26

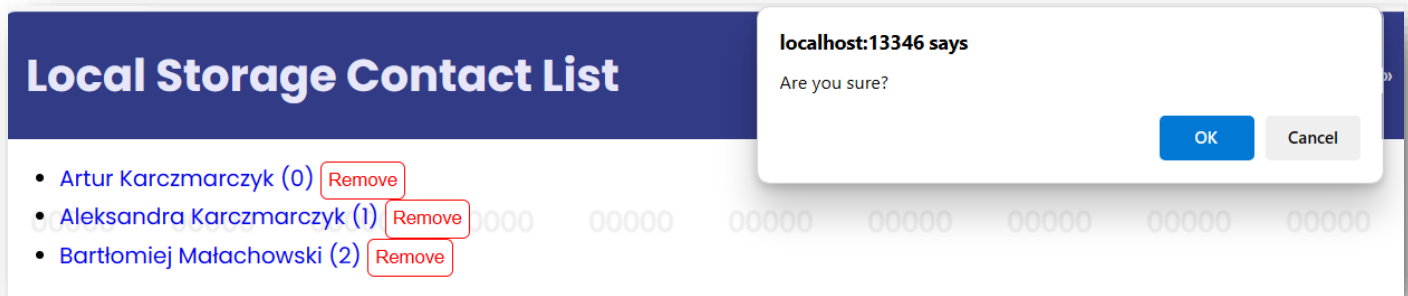
```

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

IMPLEMENTACJA USUWANIA DANYCH

W komponencie `ListComponent` dodaj metodę akcji `delete(index: number)` usuwającą poprzez serwis `local storage` element wskazany wartością parametru `index`. W widoku dodaj dla każdego elementu wyświetlanego na liście link lub przycisk dowiązujący zdarzenie (`click`) do wywołania metody `delete()` z parametrem będącym indeksem tego elementu w tablicy danych.

Przykładowa implementacja:

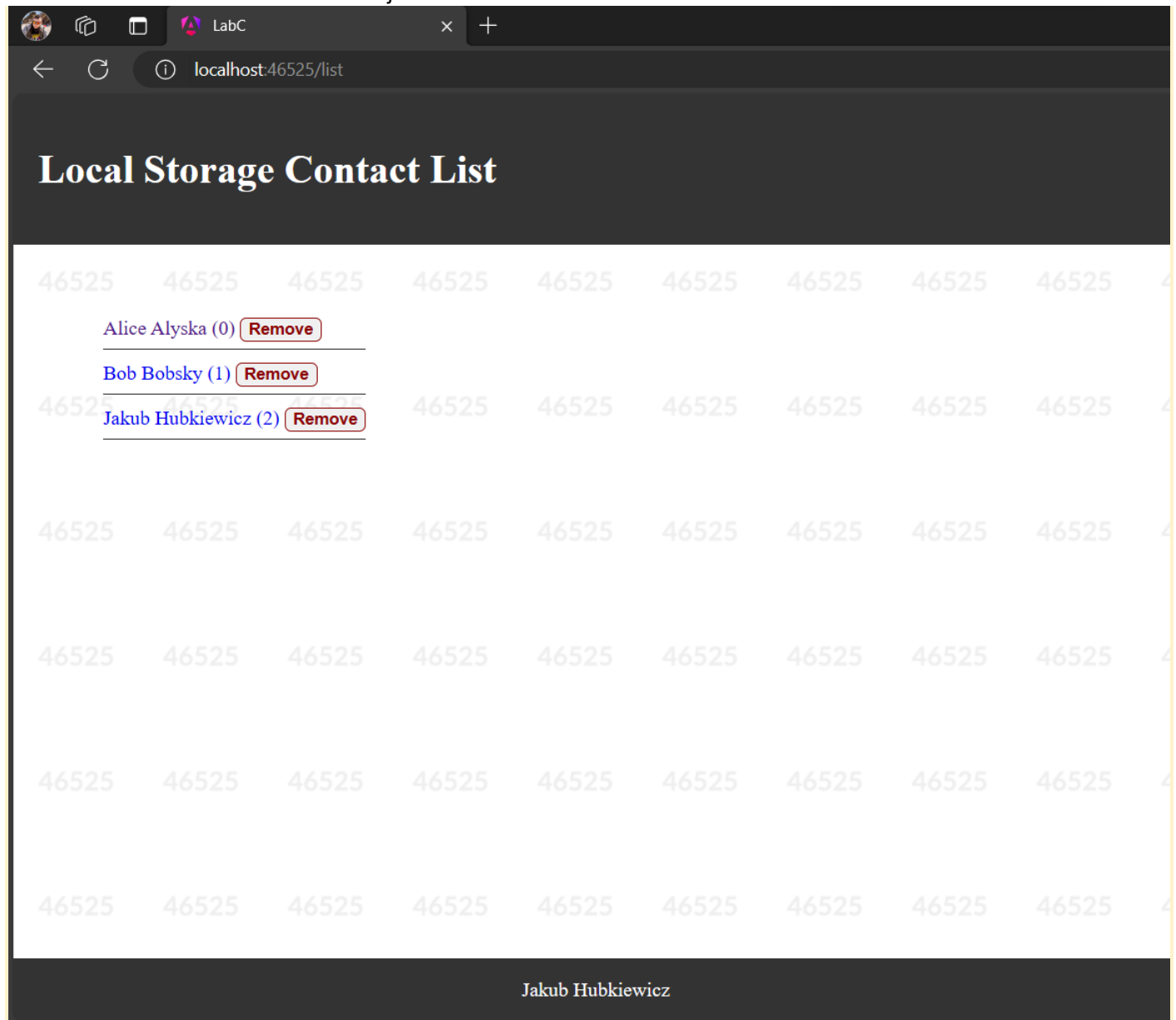


```
<button class="delete" (click)="delete(i)">Remove</button>
```

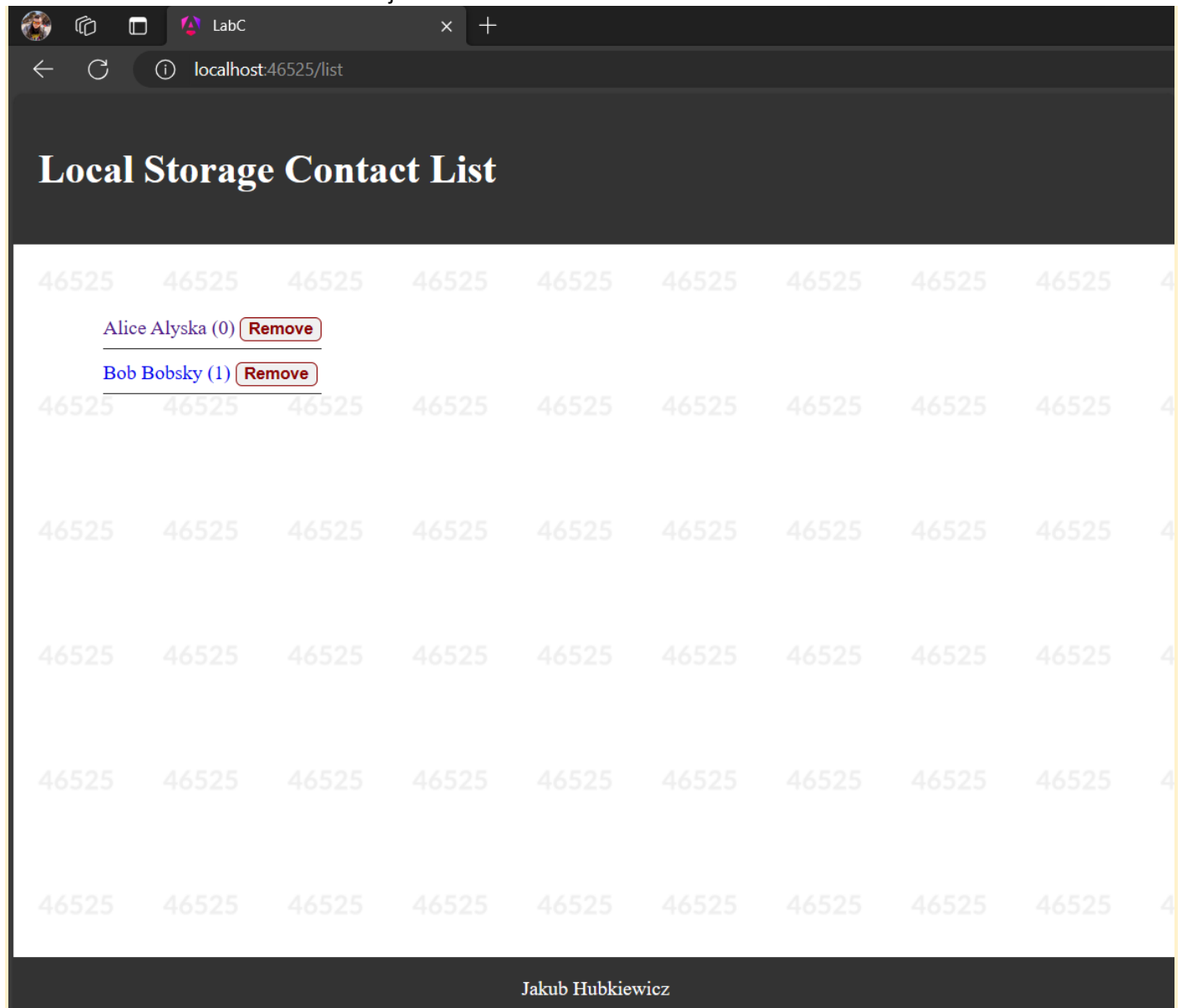
```
delete(index: number): void {  
  if (confirm("Are you sure?")) {  
    this.personLsService.deletePerson(index);  
    this.people = this.personLsService.getAll();  
  }  
}
```

W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/8af49aff01884efafd9dd86d6135362edb31a0a2>

Umieść poniżej zrzut ekranu przeglądarki z ekranem listy osób przed usunięciem osoby:



Umieść poniżej zrzut ekranu listy po usunięciu osoby:



Umieść poniżej zrzut ekranu kodu pliku `src/app/list/list.component.ts`:

```

1  import {Component, OnInit} from '@angular/core';
2  import {Person} from "../person";
3  import {PersonLsService} from "../person-ls.service";
4  import {RouterLink} from "@angular/router";
5  import {NgForOf} from "@angular/common";
6
7  1+ usages  new *
8  @Component({
9      selector: 'app-list',
10     templateUrl: './list.component.html',
11     standalone: true,
12     imports: [
13         RouterLink,
14         NgForOf
15     ],
16     styleUrls: ['./list.component.css']
17 })
18 export class ListComponent implements OnInit {
19     people: Person[] = [];
20
21     no usages  new *
22     constructor(
23         private personLsService: PersonLsService,
24     ) {
25
26     no usages  new *
27     ngOnInit(): void {
28         this.people = this.personLsService.getAll();
29     }
30 }

```

```

28
29  1+ usages  new *
30  delete(index: number): void {
31      if (confirm("Are you sure?")) {
32          this.personLsService.deletePerson(index);
33          this.people = this.personLsService.getAll();
34      }
35  }
36

```

Umieść poniżej zrzut ekranu kodu pliku `src/app/list/list.component.html`:

```

1  <ul>
2    <li *ngFor="let person of people; let i = index">
3      <a [routerLink]="['/details', i]">
4        {{ person.firstName }} {{ person.lastName }} ({{ i }})
5      </a>
6      <button type="submit" class="delete" (click)="delete(i)">Remove</button>
7    </li>
8  </ul>

```

Umieść poniżej zrzut ekranu kodu pliku `src/app/list/list.component.css`:

```

1  button.delete{
2    border: 1px solid darkred;
3    color: darkred;
4    font-weight: bold;
5    border-radius: 5px;
6  }
7
8  li{
9    border-bottom: 1px solid #333;
10   padding: 10px 0px 5px 0px;
11   list-style-type: none;
12 }
13
14 li>a{
15   text-decoration: none;
16 }
17

```

W międzyczasie były zmiany

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-c` 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-c` w swoim repozytorium:

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

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Nie ukrywam – to było ciężkie zadanie. Już od początku miałem problemy z wyświetlaniem czegokolwiek, a potem kod nie działał i się nie pokrywał z tym z repozytorium. Szczerze muszę przyznać, że musiałem się mocno posiłkować załączonymi pomocami naukowymi.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.