AI2B LAB C	Hubkiewicz Jakub Album 46525 Grupa 2	**Wersja 1**

ROUTING I SERWISY W ANGULAR

SPIS TREŚCI

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://placehold.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.:

```
<header>
  <h1>Local Storage Contact List</h1>
</header>

<div class="router-outlet">
  <router-outlet></router-outlet>
</div>

<footer>Imie Nazwisko</footer>
```

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

Przykładowy efekt:

Local Storage Contact List

Imie 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).

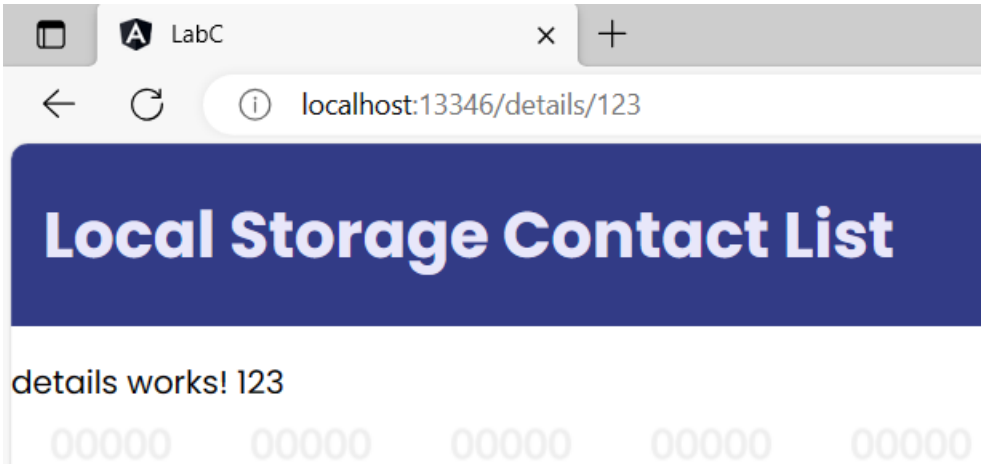
Przykładowa implementacja routingu:

```
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`:

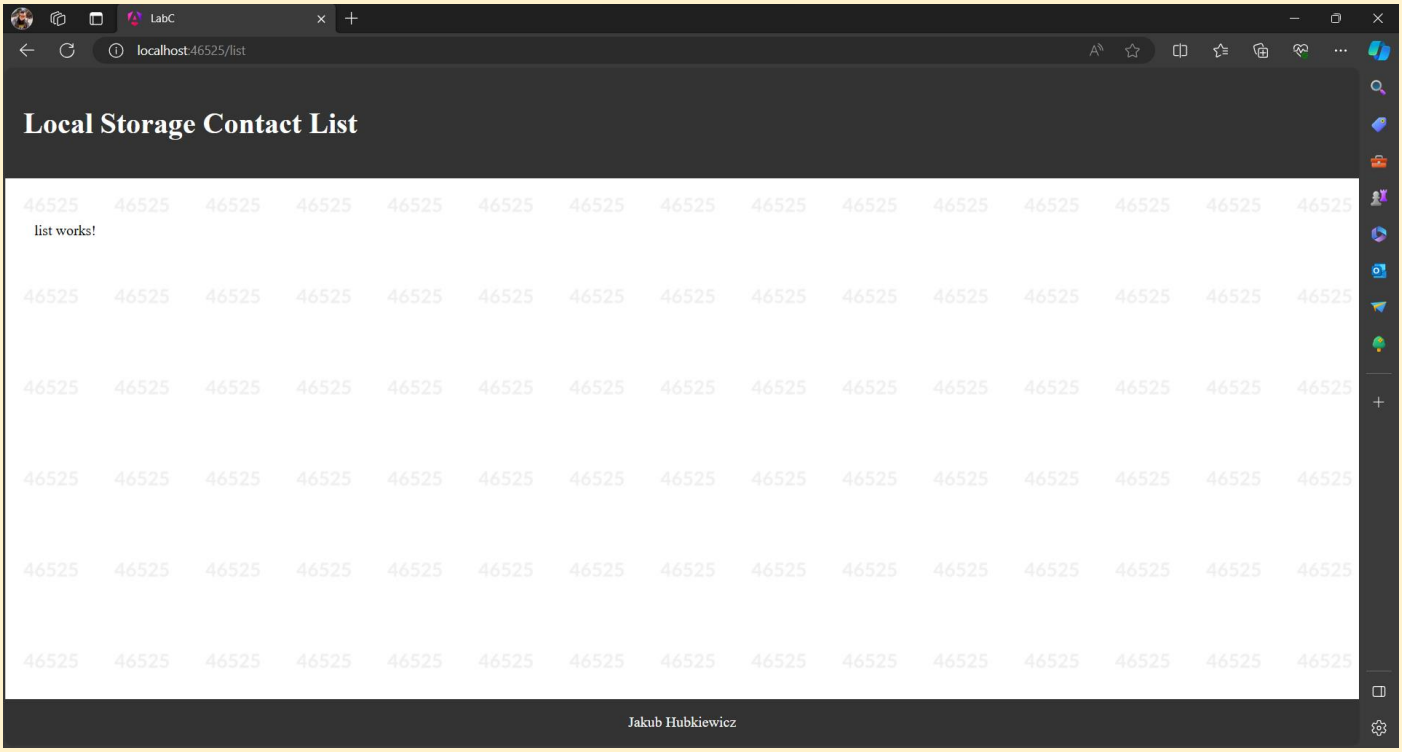
```
ngOnInit() : void  {
  this.route.params.subscribe( observerOrNext: params : Params  => {
    this.personId = params['id'];
  });
}
```

Przykładowy wygląd strony `/details/123`:

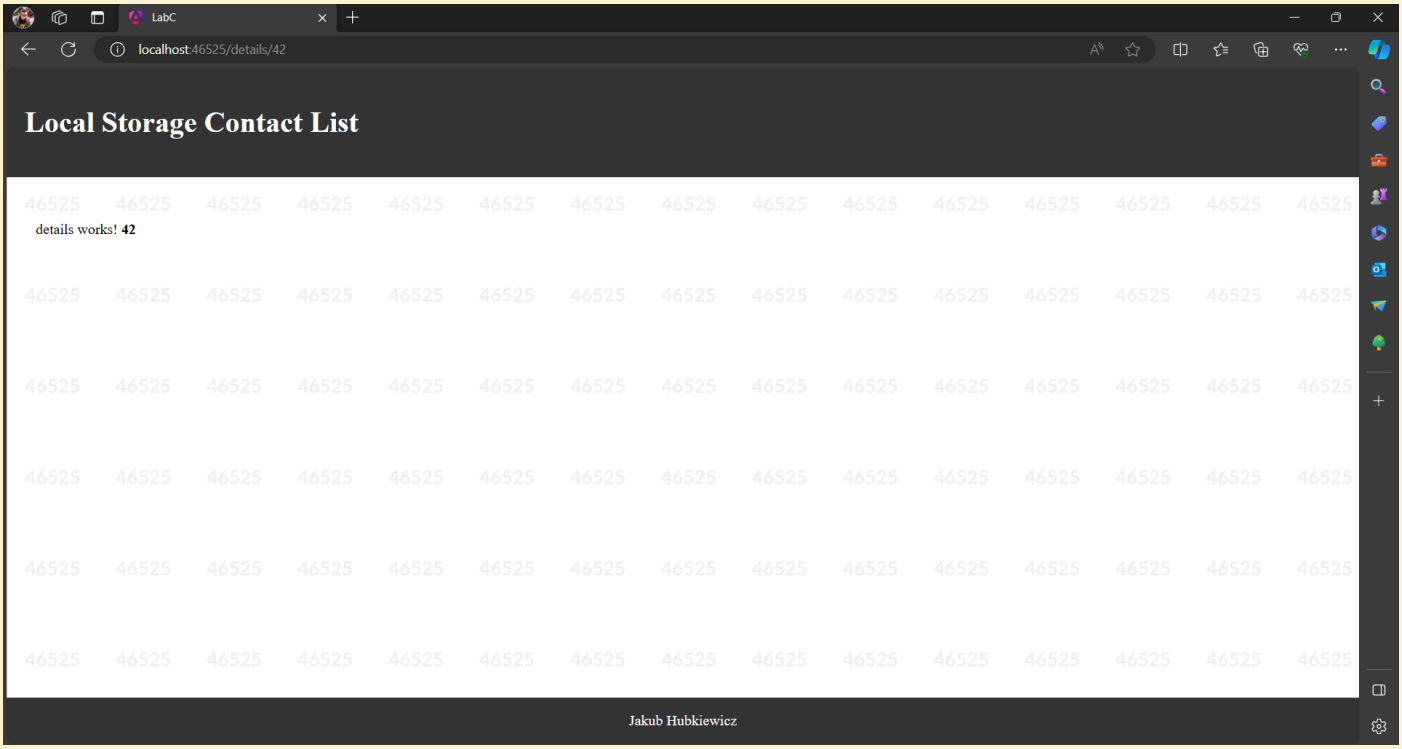


W razie problemów, porównaj kod: https://github.com/ideaspot-pl/ai2b-lab-c-ls/commit/f6902ba9fd002753586a92e61f5575ea43f39aed

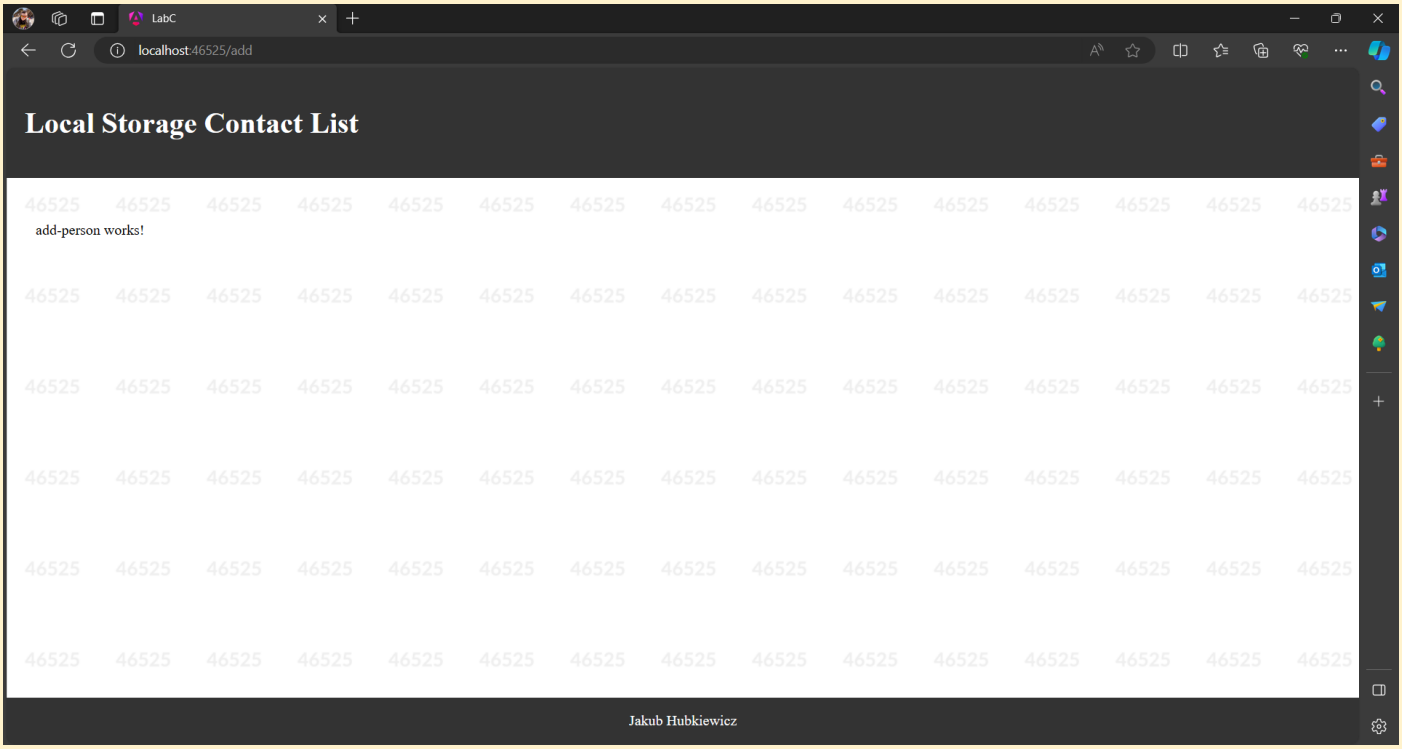
Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę `/list`. Upewnij się, że na zrzucie ekranu widoczny jest adres URL oraz domyślny szablon „`list works!`”



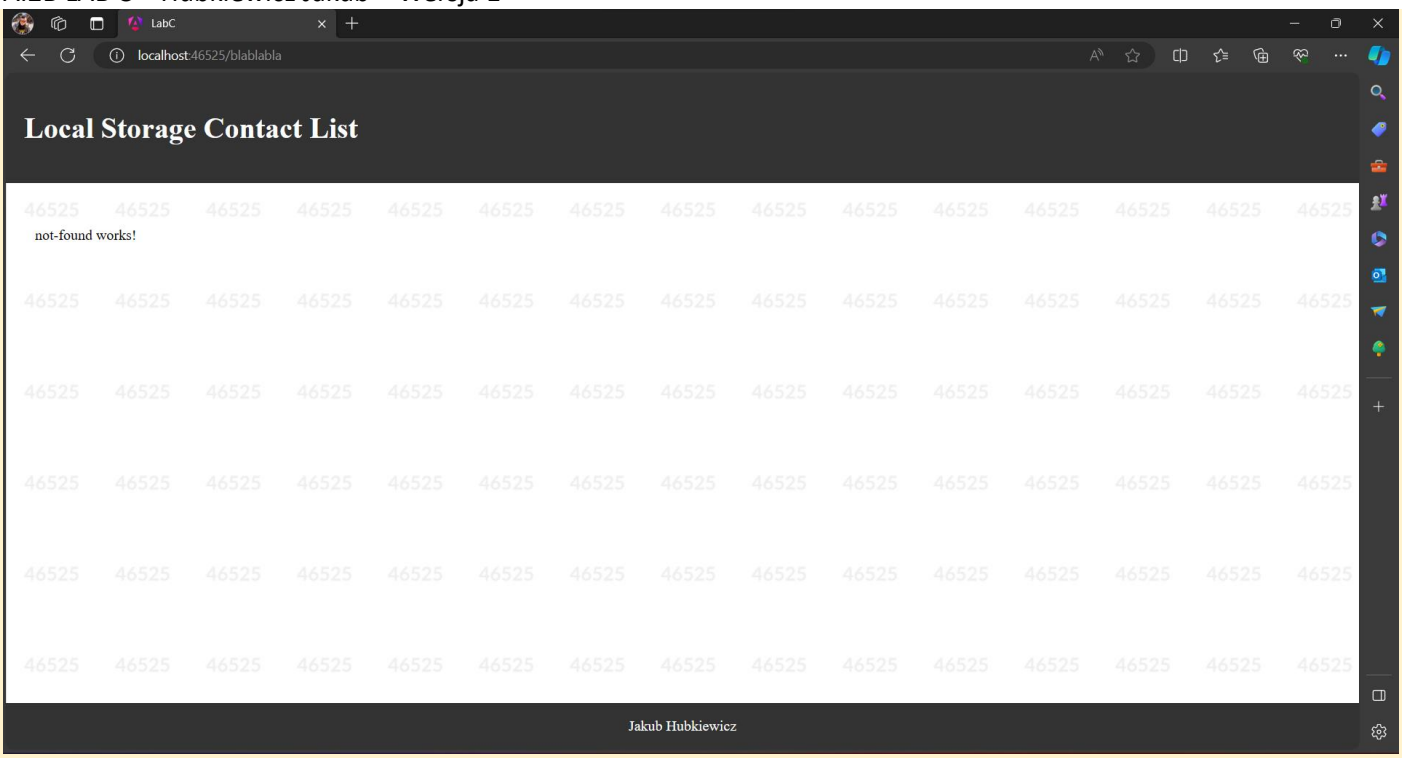
Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę `/details/42`. Upewnij się, że na zrzucie ekranu widoczny jest adres URL oraz wyświetlona w komponencie `DetailsComponent` wartość `42` (z URLa).



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:

```
export interface Person {
  firstName?: string;
  lastName?: string;
  age?: number;
  address: {
    city?: string;
    street?: string;
    postCode?: string;
  }
}
```

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:

```
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 zadanym 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()`:

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

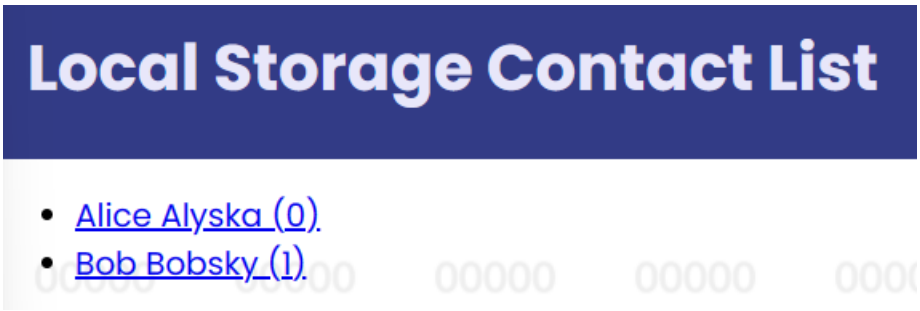
IMPLEMENTACJA LISTY DANYCH

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:



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.

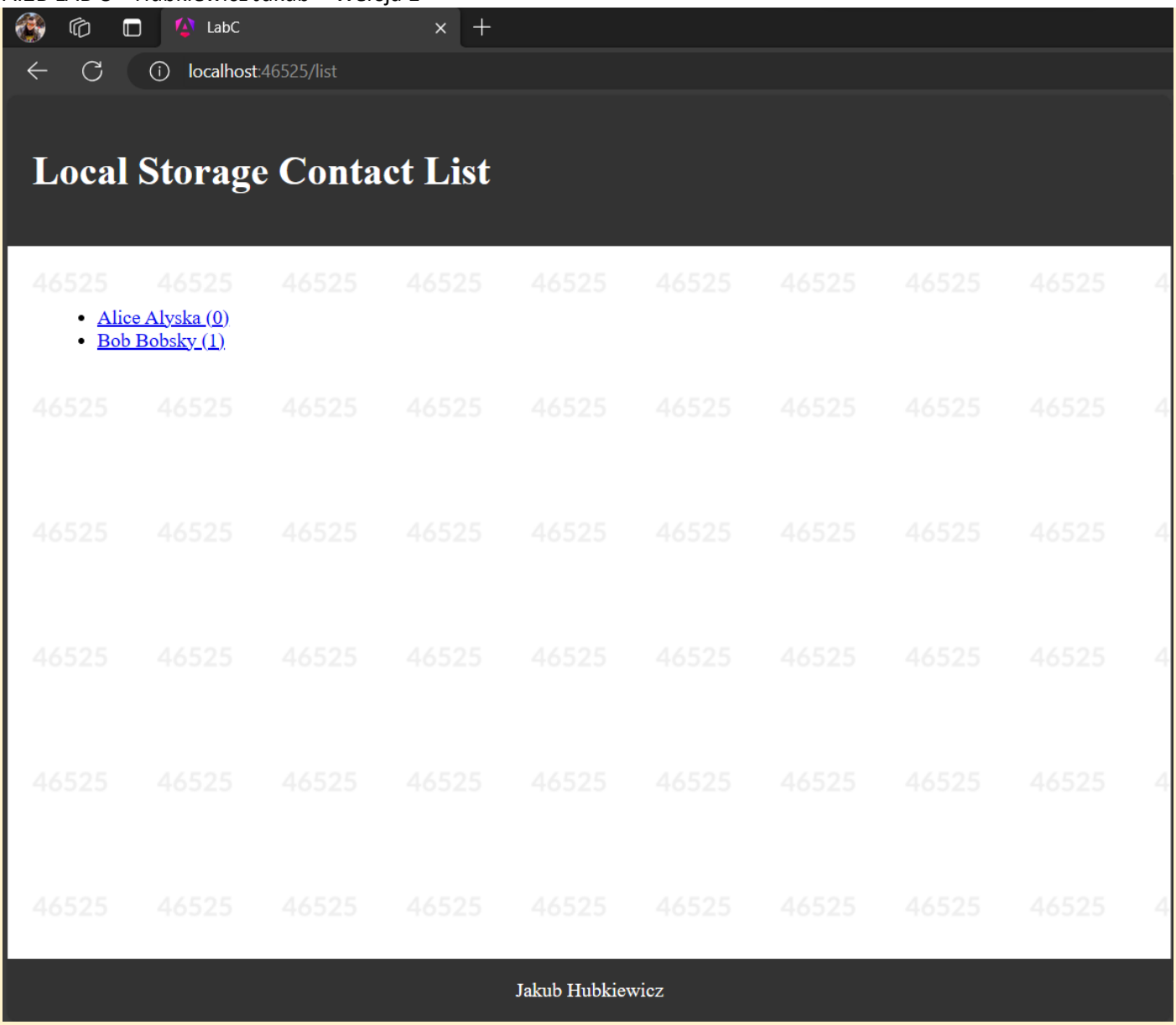
Przykładowa implementacja:

```
import {Component, OnInit} from '@angular/core';
import {Person} from "../person";
import {PersonLsService} from "../person-ls.servi

4 usages  Artur Karczmarczyk *
@Component({
  selector: 'app-list',
  templateUrl: './list.component.html',
  styleUrls: ['./list.component.css']
})
export class ListComponent implements OnInit {
  people: Person[] = [];

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

  no usages  new *
  ngOnInit() : void  {
    this.people = this.personLsService.getAll();
  }
}
```

```
<ul>
  <li *ngFor="let person of people; let i = index">
    <a [routerLink]="['/details', i]">
      {{ person.firstName }} {{ person.lastName }} ({{ i }})
    </a>
  </li>
</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`:

```typescript
import {Component, OnInit} from '@angular/core';
import {Person} from "../person";
import {PersonLsService} from "../person-ls.service";
import {RouterLink} from "@angular/router";
import {NgForOf} from "@angular/common";

1+ usages  new *
@Component({
  selector: 'app-list',
  templateUrl: './list.component.html',
  standalone: true,
  imports: [
    RouterLink,
    NgForOf
  ],
  styleUrls: ['./list.component.css']
})
export class ListComponent implements OnInit {
  people: Person[] = [];

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

  no usages  new *
  ngOnInit() : void  {
    this.people = this.personLsService.getAll();
  }
}
```

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

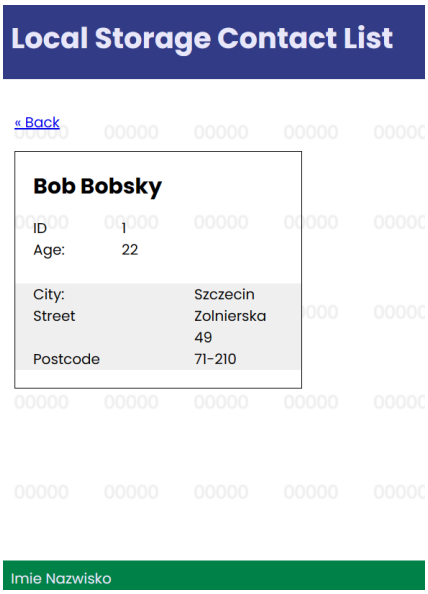
```html
<ul>
  <li *ngFor="let person of people; let i = index">
    <a [routerLink]="['/details', i]">
      {{ person.firstName }} {{ person.lastName }} ({{ i }})
    </a>
  </li>
</ul>
```

Punkty:	0	1

IMPLEMENTACJA KOMPONENTU SZCZEGÓŁÓW

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

Przykładowa implementacja:

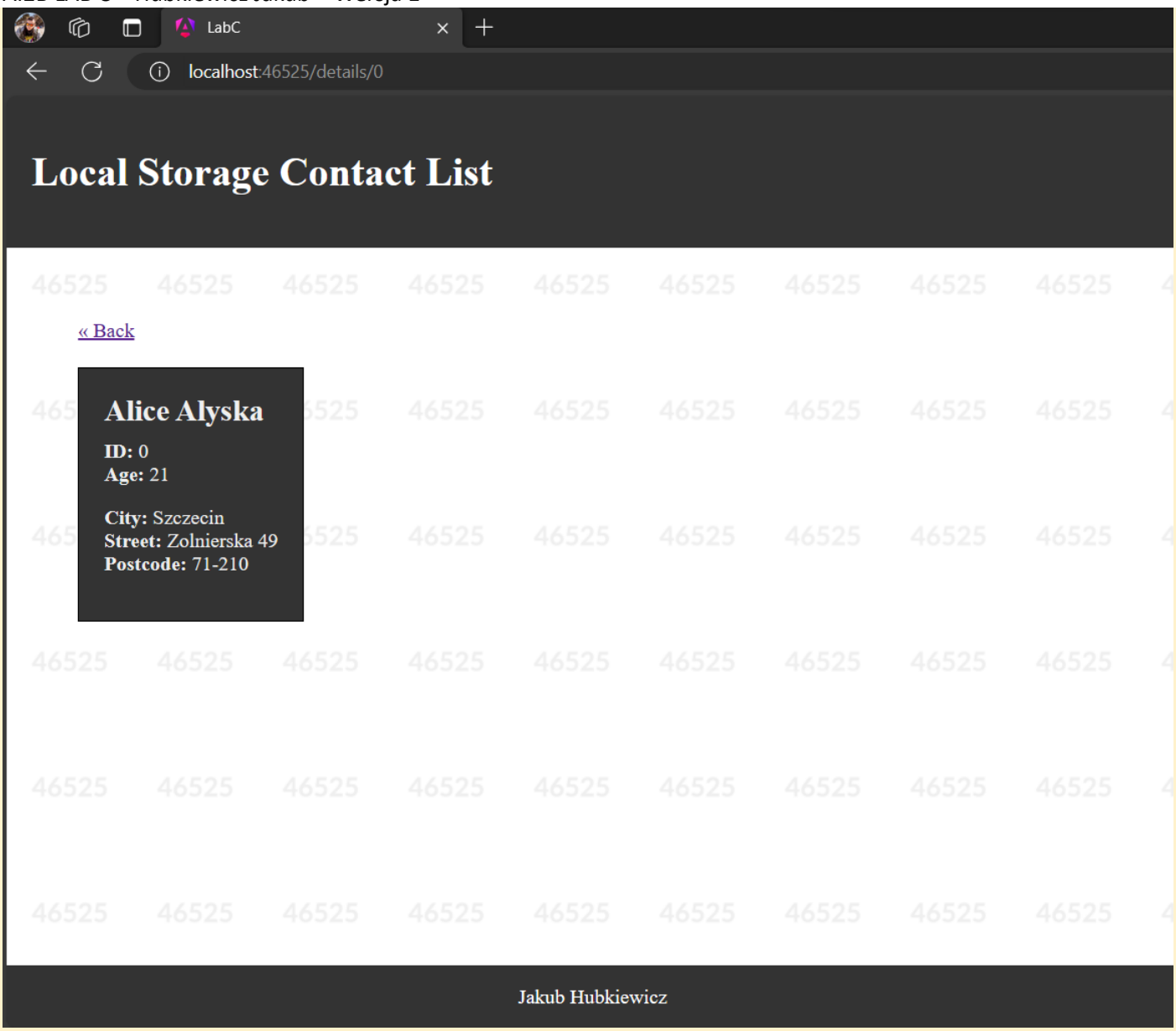



```ts
import ...

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( observerOrNext: params : Params  => {
      this.personId = params['id'];
      this.person = this.personId
        ? this.personLsService.getPerson(this.personId)
        : undefined;
    });
  }
}
```

```html
<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>{{personId}}</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>

  <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`:

details.component.ts × details.component.css details.component.html

```ts
import {Component, OnInit} from '@angular/core';
import {ActivatedRoute, RouterLink} from "@angular/router";
import {Person} from "../person";
import {PersonLsService} from "../person-ls.service";
import {NgIf} from "@angular/common";

1+ usages  new *
@Component({
  selector: 'app-details',
  templateUrl: './details.component.html',
  standalone: true,
  imports: [
    RouterLink,
    NgIf
  ],
  styleUrls: ['./details.component.css']
})
export class DetailsComponent implements OnInit{
  personId?: number;
  person?: Person;

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

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

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

details.component.ts details.component.css details.component.html

```
<div class="details-container">
  <div>
    <a [routerLink]="'/'">&laquo; Back</a>
  </div>
  <div *ngIf="person" class="person">
    <h2>{{person.firstName}} {{person.lastName}}</h2>
    <span class="info">
      <p><b>ID: </b>{{personId}}</p>
      <p><b>Age: </b>{{person.age}}</p>
    </span>

    <p *ngIf="person.address" class="address">
      <b>City: </b>
      {{person.address.city}}<br>
      <b>Street: </b>
      {{person.address.street}}<br>
      <b>Postcode: </b>
      {{person.address.postCode}}
    </p>
  </div>

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

```

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

details.component.ts | details.component.css | details.component.html

```css
.details-container {
  padding: 25px;
}

.person {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  background-color: #333333;
  color: #efefef;
  border: 1px solid black;
  padding: 20px;
  margin: 20px 0;
  max-width: 320px;
  box-sizing: border-box;
}

.person h2 {
  padding: 0;
  margin: 0;
  font-weight: bold;
}

.info > p:nth-child(1){
  margin-top: 10px;
}

.info > p{
  margin: 0;
}

```

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: {}
};
```

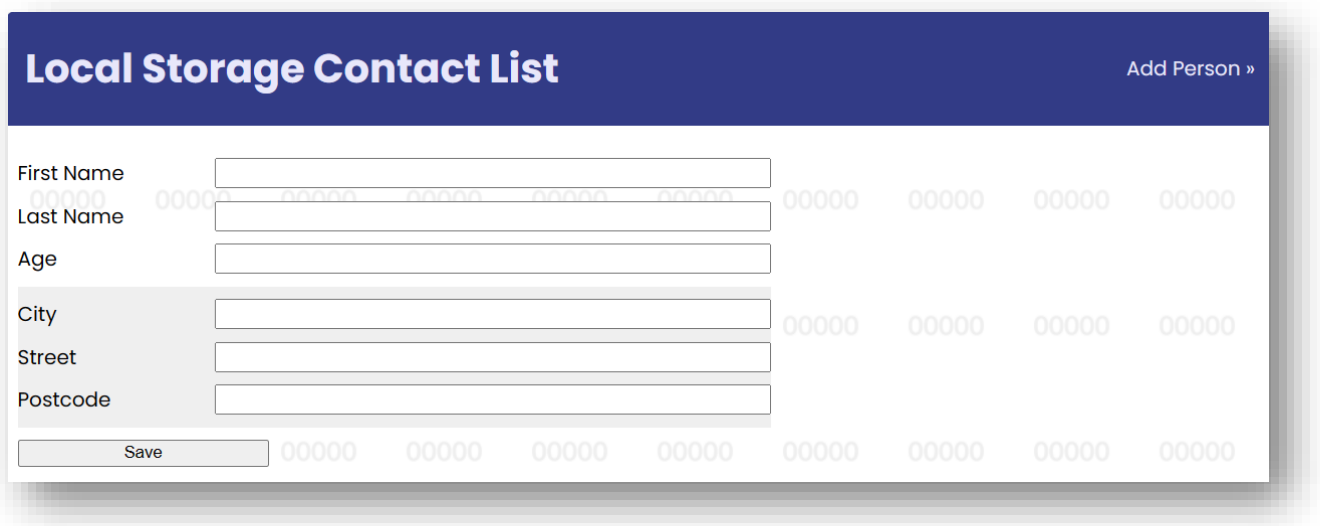
Wstrzyknij opracowany wcześniej serwis oraz obiekt typu Router potrzebny do przełączenia widoku po pomyślnym zapisie danych. Dodaj metodę `save()`, która po wywołaniu zapisze dane osobowe z pola `person` do local storage za pomocą metody `addPerson()` serwisu, a następnie przekieruje użytkownika na listę.

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:

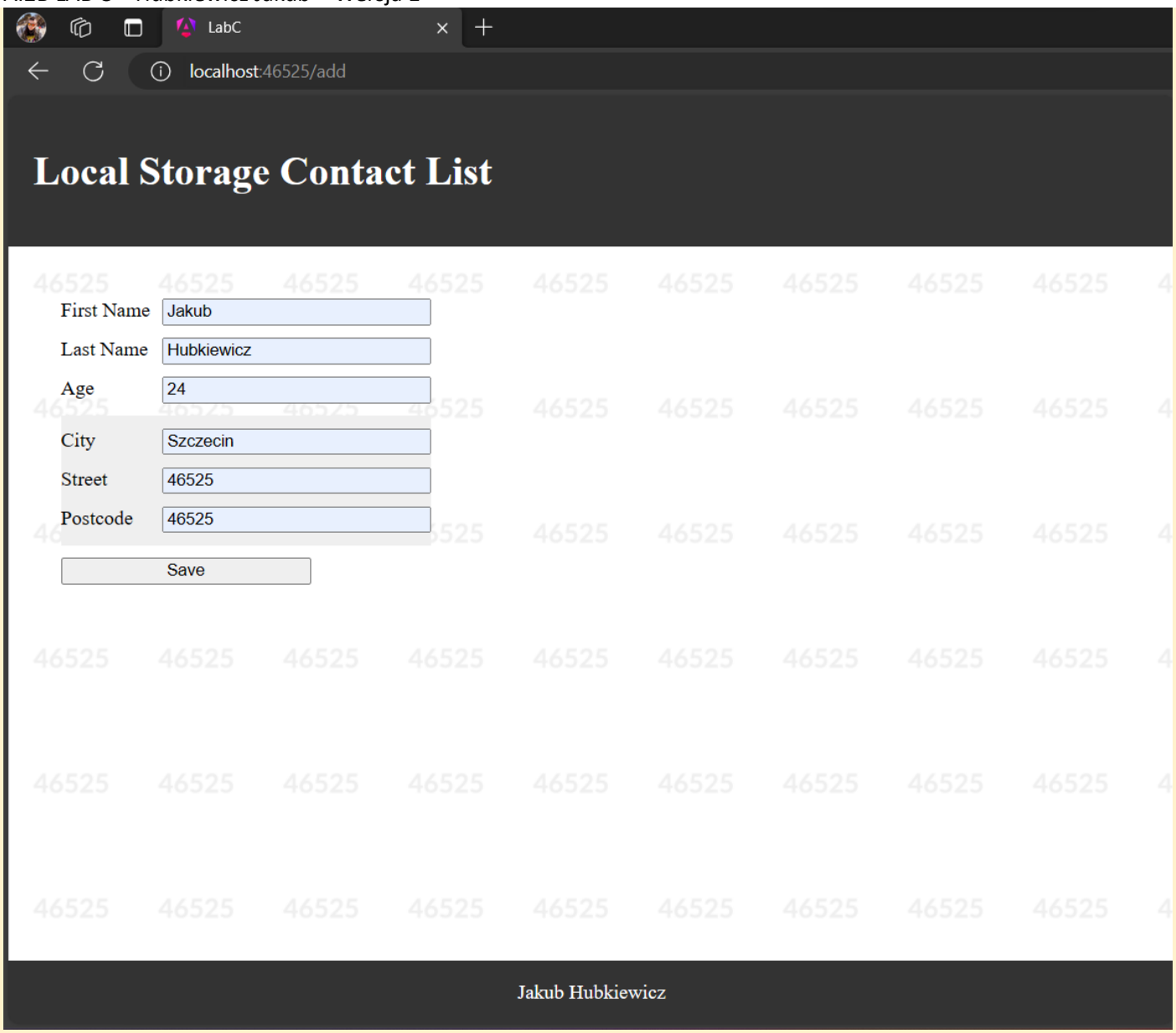


```ts
import ...

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( commands: ['/list']);
  }

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

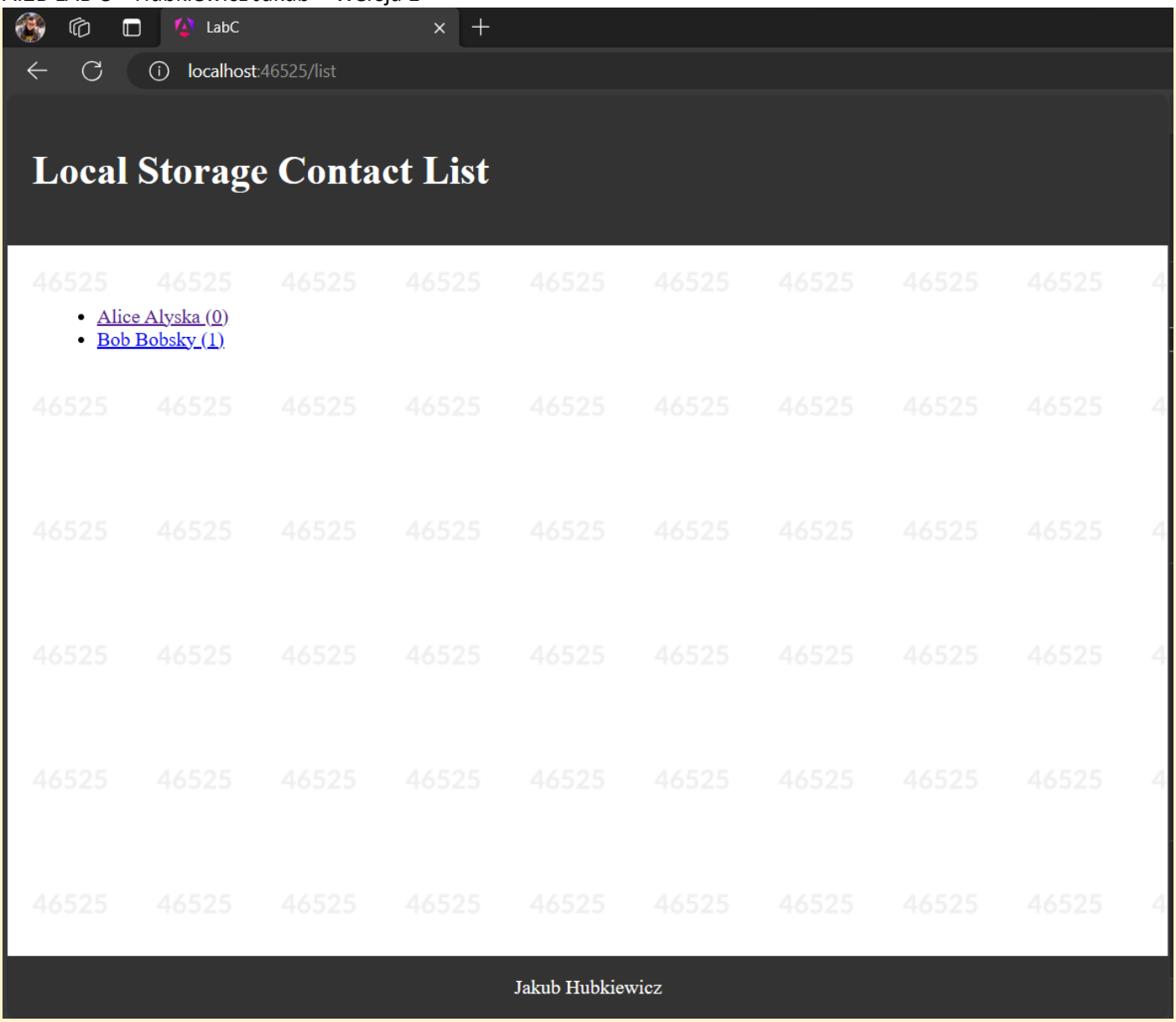
```html
<form>
  <div class="form-row">
    <label for="firstName">First Name</label>
    <input type="text" id="firstName" name="firstName" [(ngModel)]="person.firstName">
  </div>

  <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

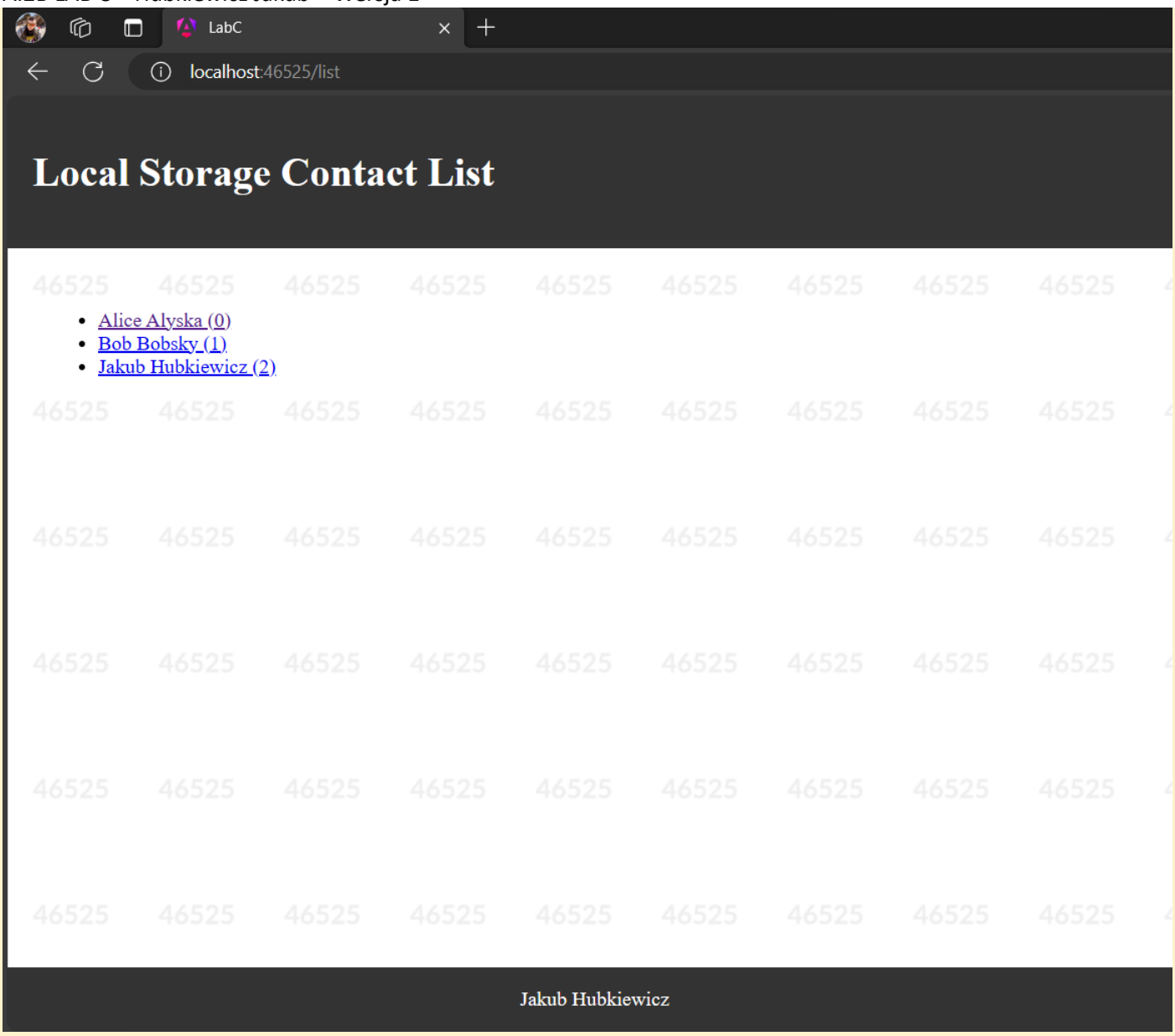
Umieść poniżej zrzut ekranu przeglądarki z ekranem dodawania osoby:



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`:

add-person.component.ts × add-person.component.html add-person.component.css

```ts
import {Component, OnInit} from '@angular/core';
import {Person} from "../person";
import {PersonLsService} from "../person-ls.service";
import {Router} from "@angular/router";
import {FormsModule} from "@angular/forms";

1+ usages  new *
@Component({
  selector: 'app-add-person',
  templateUrl: './add-person.component.html',
  standalone: true,
  imports: [
    FormsModule
  ],
  styleUrls: ['./add-person.component.css']
})
export class AddPersonComponent implements OnInit{
  person: Person = {
    address: {},
  }

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

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

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

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

```html
<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>
```

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

add-person.component.ts | add-person.component.html | add-person.component.css

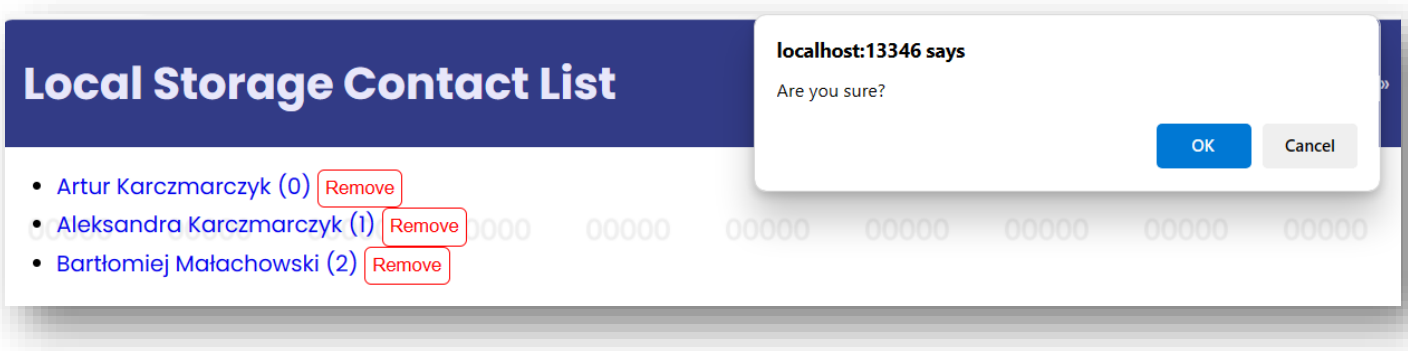
```css
form {
  display: flex;
  flex-direction: column;
  gap: 10px;
  max-width: 600px;
  padding: 10px;
}

.address {
  display: flex;
  flex-direction: column;
  gap: 10px;
  background-color: #efefef;
  padding: 10px 0;
}

button {
  max-width: 200px;
}

.form-row {
  display: grid;
  gap: 10px;
  grid-template-columns: 1fr 3fr;
}
```

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:

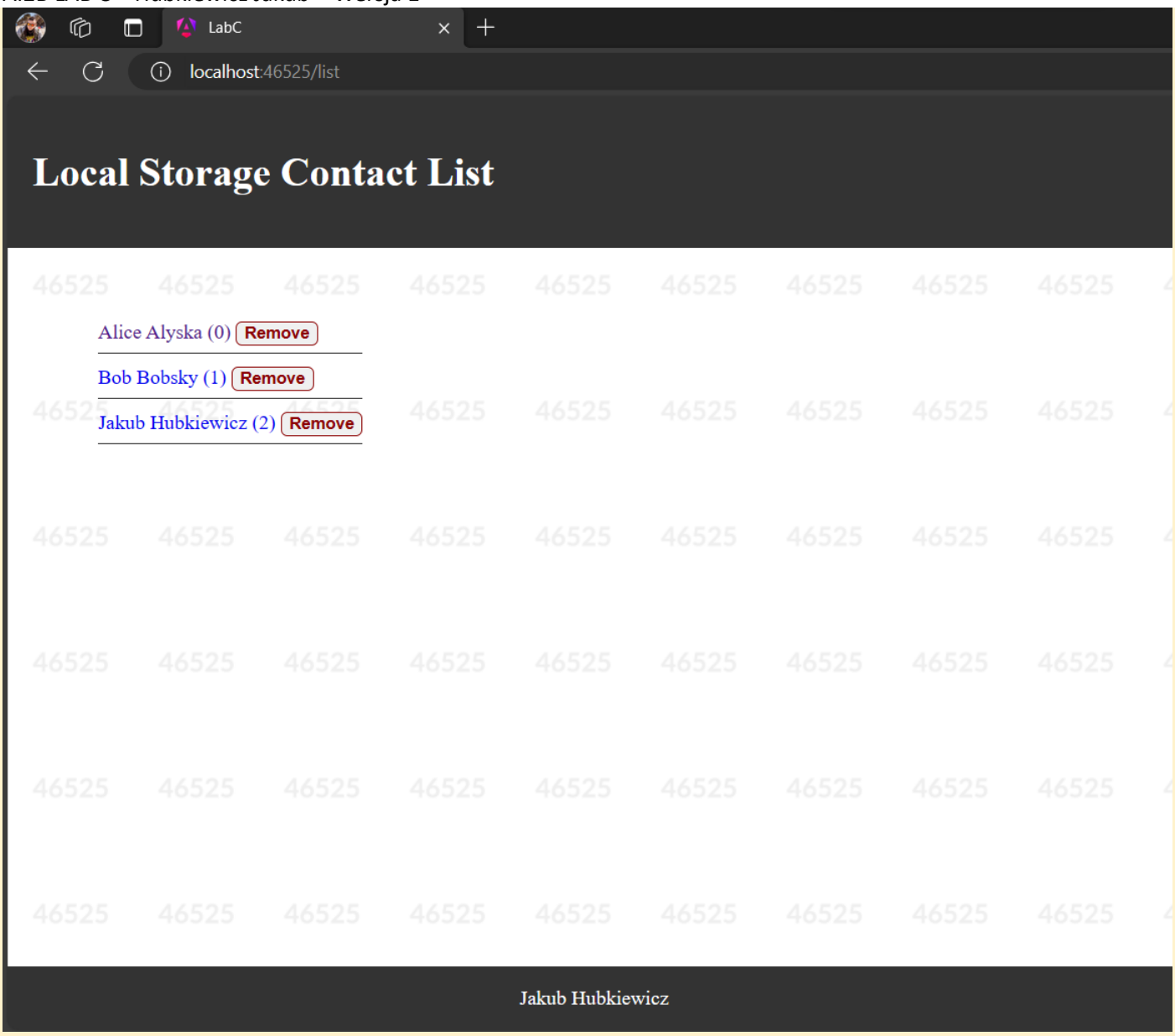



```
<button class="delete" (click)="delete(i)">Remove</button>
```

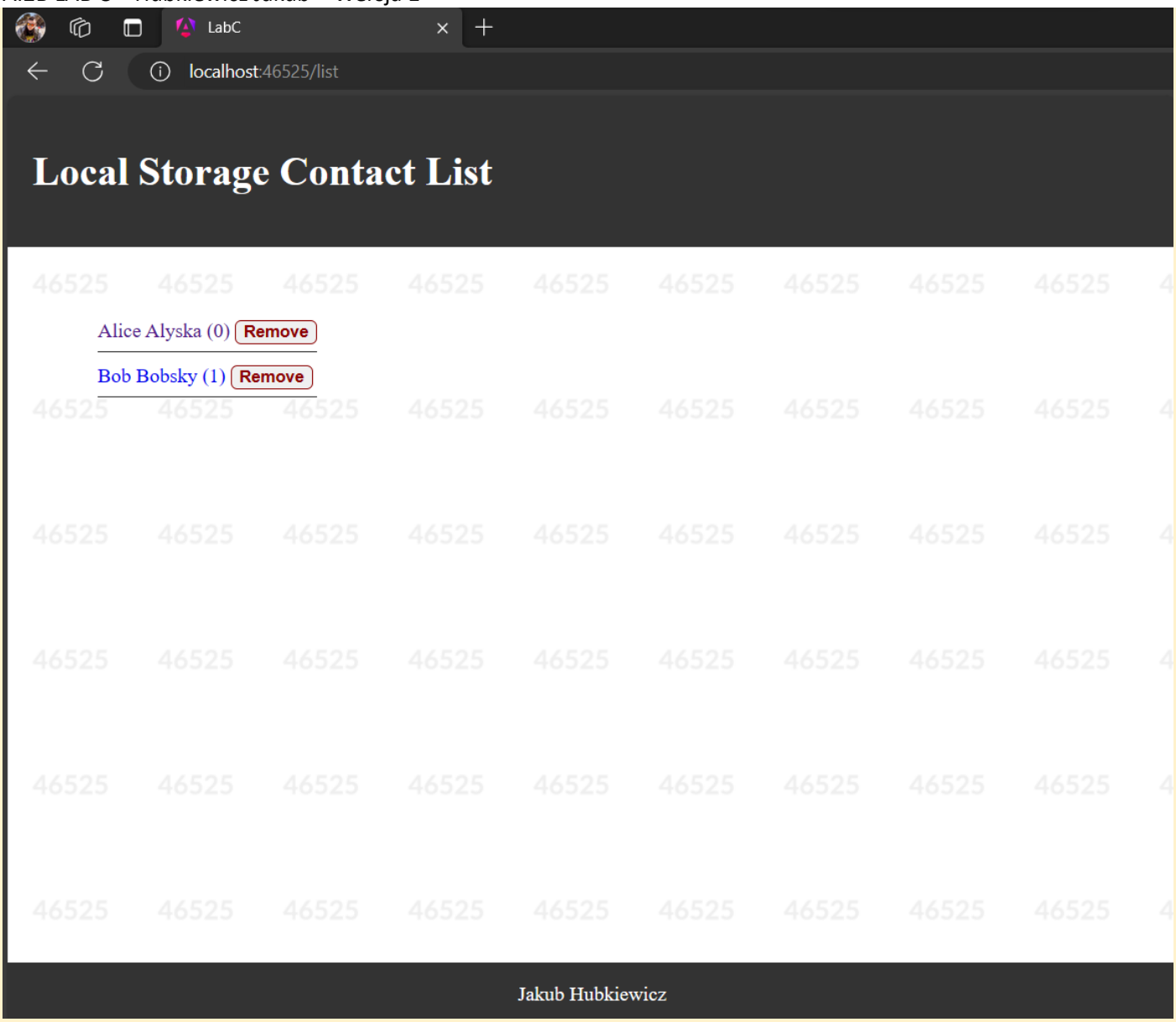
```
delete(index: number): void {
  if (confirm("Are you sure?")) {
    this.personLsService.deletePerson(index);
    this.people = this.personLsService.getAll();
  }
}
```

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`:

```typescript
import {Component, OnInit} from '@angular/core';
import {Person} from "../person";
import {PersonLsService} from "../person-ls.service";
import {RouterLink} from "@angular/router";
import {NgForOf} from "@angular/common";

1+ usages  new *
@Component({
  selector: 'app-list',
  templateUrl: './list.component.html',
  standalone: true,
  imports: [
    RouterLink,
    NgForOf
  ],
  styleUrls: ['./list.component.css']
})
export class ListComponent implements OnInit {
  people: Person[] = [];

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

  no usages  new *
  ngOnInit() : void  {
    this.people = this.personLsService.getAll();
  }

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

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

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

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

```
button.delete{
  border: 1px solid darkred;
  color: darkred;
  font-weight: bold;
  border-radius: 5px;
}

li{
  border-bottom: 1px solid #333;
  padding: 10px 0px 5px 0px;
  list-style-type: none;
}

li>a{
  text-decoration: none;
}
```

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.