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

Angular krok po kroku

# Spis treści

[Spis treści 1](#_Toc150775150)

[Cel zajęć 1](#_Toc150775151)

[Rozpoczęcie 1](#_Toc150775152)

[Jak wypełnić to sprawozdanie? 2](#_Toc150775153)

[Gdzie pracować 2](#_Toc150775154)

[Instalacja Node.js i Angular CLI 2](#_Toc150775155)

[Inicjalizacja projektu 4](#_Toc150775156)

[Struktura plików i katalogów 5](#_Toc150775157)

[Hello World 6](#_Toc150775158)

[Tworzenie komponentów 7](#_Toc150775159)

[Komponent HousingLocation 10](#_Toc150775160)

[Iterowanie po obiektach 13](#_Toc150775161)

[Wyodrębnienie serwisu 15](#_Toc150775162)

[Komponent ze szczegółami budynku. Routingi. 16](#_Toc150775163)

[Wyszukiwarka 22](#_Toc150775164)

[Podłączenie API 24](#_Toc150775165)

[Commit projektu do GIT 26](#_Toc150775166)

[Podsumowanie 26](#_Toc150775167)

# Cel zajęć

Celem głównym zajęć jest zapoznanie się z podstawą tworzenia i rozwijania projektów aplikacji z użyciem frameworku Angular. Niniejsza instrukcja przeprowadzi uczestnika krok po kroku przez proces tworzenia prostej aplikacji Angular wykorzystującej komponenty, routingi, serwisy i zewnętrzne API.

# Rozpoczęcie

Rozpoczęcie zajęć. Przedstawienie prowadzącego. Przedstawienie uczestników. Przedstawienie zasad laboratorium.

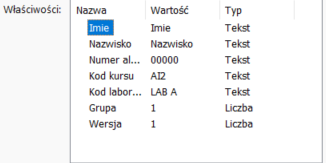
# Jak wypełnić to sprawozdanie?

Zapisz ten plik na dysku twardym jako kopię. Zmień nazwę pliku:

* grN na odpowiedni numer grupy (np. gr3),
* nazwisko-imie na Twoje dane bez polskich znaków.

Otwórz kolejno Plik -> Informacje -> Właściwości -> Właściwości zaawansowane -> Niestandardowe.

Zaktualizuj właściwości:



Czytaj tę instrukcję, wypełniaj polecenia, uzupełniaj zrzuty ekranu zgodnie z poleceniami.

Gotowe sprawozdanie wyślij w nieprzekraczalnym terminie **w postaci pliku PDF**.

# Gdzie pracować

Zaloguj się do systemu Windows / pulpitu zdalnego rdp.wi.zut.edu.pl:

* spoza sieci ZUT potrzebny VPN: <https://uci.zut.edu.pl/uslugi-uci/vpn.html>;
* nazwa użytkownika: WIAD\ab12345
* komputer: rdp.wi.zut.edu.pl

Wykorzystać można PhpStorm (zalecane) lub VS Code z rozszerzeniami Angular Language Service, Angular Snippets.

W trakcie semestru zidentyfikowany został problem ujawniający krytyczne spowolnienie procesów uwzględniających przetwarzanie wielu plików, przy wykorzystaniu dysku I:\, który to problem nie występuje przy wykorzystaniu pulpitu, mimo że oba te foldery znajdują się fizycznie na tej samej macierzy dyskowej. Uwaga: tylko dysk I:\ posiada backupy, dlatego każdorazowo należy upewnić się, że efekty pracy przeniesiono z powrotem na dysk I:\ lub zabezpieczono w inny sposób. Dane na pulpicie mogą zniknąć w dowolnym momencie 😊

# Instalacja Node.js i Angular CLI

Wejdź na stronę <https://nodejs.org/en/download/current>. Pobierz wersję Current -> Windows Binary (.zip) -> 64-bit. Rozpakuj archiwum do C:\Users\ni123456\Desktop\ai2b\node. Dodaj ten folder do zmiennej środowiskowej Path użytkownika:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Obraz zawierający tekst, elektronika, zrzut ekranu, wyświetlacz

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Zamknij wszystkie okna wiersza poleceń, żeby zostały uwzględnione zmiany.

Uruchom nowe okno wiersza poleceń. Wykonaj polecenie npm -v. Powinno zadziałać – udało się zainstalować NODE i NPM lokalnie dla użytkownika.

Uruchom ulubiony terminal. Zainstaluj globalnie Angular CLI w wersji 16. W przypadku problemów (PowerShell lubi robić problemy) – spróbuj na innym terminalu:

> npm install -g @angular/cli@”16”

npm WARN deprecated @npmcli/move-file@2.0.1: This functionality has been moved to @npmcli/fs

added 269 packages in 17s

39 packages are looking for funding

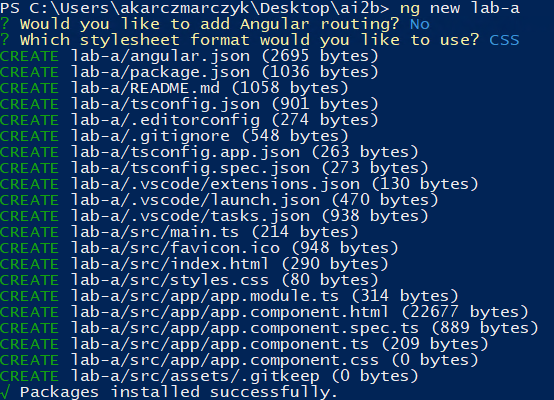
run `npm fund` for details

# Inicjalizacja projektu

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

> ng new lab-a

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-a

> ng serve --port=00000

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

# Struktura plików i katalogów

Otwórz projekt w wybranym przez siebie IDE. Przeanalizuj strukturę plików i katalogów:

* lab-a – katalog projektu
  + src – katalog z kodami aplikacji
    - index.html – główny szablon HTML aplikacji
    - style.css – główny plik ze stylami CSS
    - main.ts – kod odpowiedzialny za uruchomienie aplikacji
    - favicon.ico – ikonka
    - app – pliki głównego komponentu aplikacji
      * app.component.ts – kody źródłowe głównego komponentu aplikacji
      * app.component.css – style komponentu aplikacji
      * wszystkie nowe komponenty będą dodawane do tego katalogu

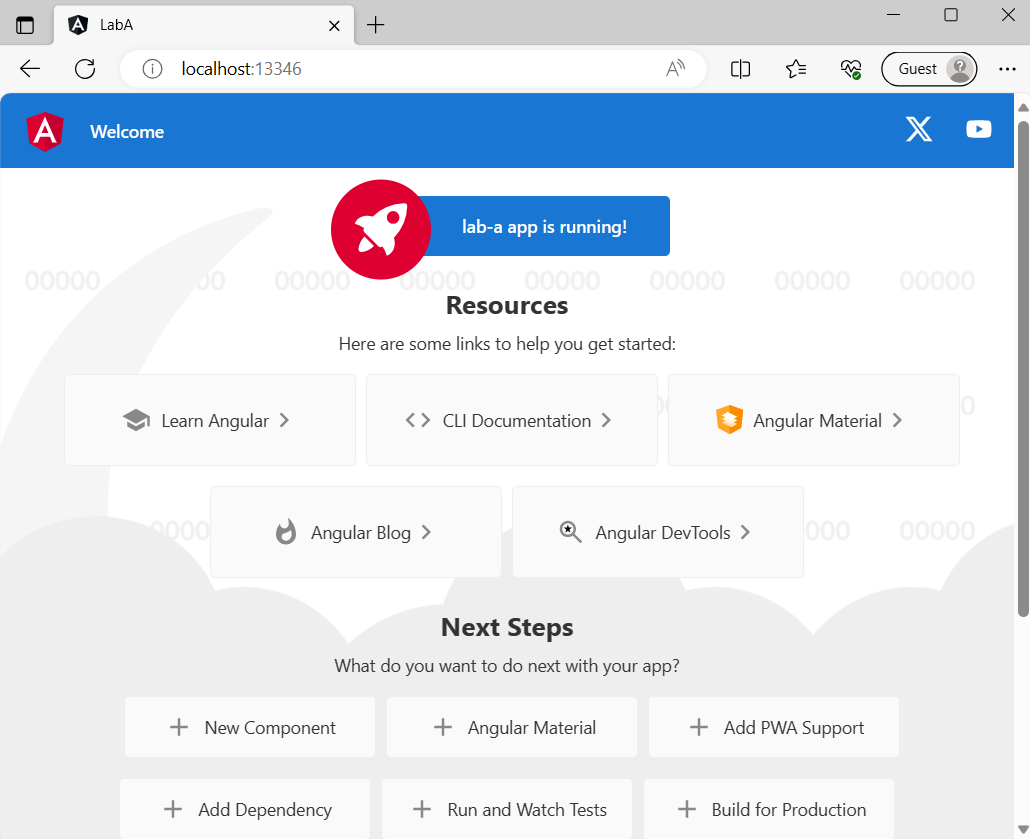
Edytuj plik src/styles.css do postaci:

body {

background: url("https://placehold.co/100x100/FFFFFF/EFEFEF/png?text=00000");

}

W efekcie w tle strony powinien zacząć wyświetlać się Twój numer albumu:



# Hello World

W pliku src/index.html zamień tytuł:

<title>Homes</title>

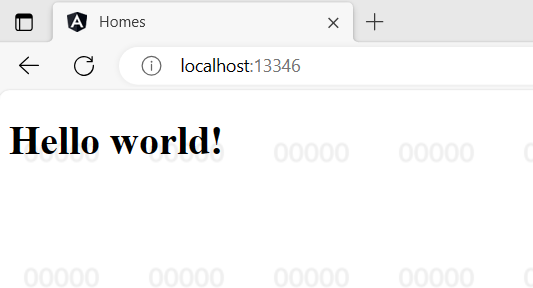
Zawartość pliku src/app/app.component.html zamień na:

<h1>Hello world!</h1>

Zawartość pola title w src/app/app.component.ts zamień na:

title = 'homes';

Oczekiwany efekt:



W przypadku problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-a-intro/commit/449abf5447fbd3db3e154ba7f5c5e29266f30092>

# Tworzenie komponentów

Upewnij się, że w jednym terminal wciąż uruchomione jest polecenie ng serve. Polecenie to powinno być uruchomione przez cały czas trwania laboratorium.

Otwórz drugi terminal i przejdź do katalogu projektu (lab-a). Wykonaj polecenie:

> ng generate component home --standalone --skip-tests

Utworzony zostanie katalog src/app/home.

W pliku src/app/app.module.ts zmodyfikuj dekorator @NgModule do postaci:

@NgModule({

declarations: [

AppComponent,

],

imports: [

BrowserModule,

HomeComponent,

],

providers: [],

bootstrap: [AppComponent]

})

Automatycznie zaimportuj brakujące klasy z wykorzystaniem IDE.

Zamień zawartość src/app/app.component.html na:

<main>

<header class="brand-name">

<img class="brand-logo" src="/assets/house.svg" alt="logo" aria-hidden="true">

<strong>Homes</strong>

</header>

<section class="content">

<app-home></app-home>

</section>

</main>

Wstaw do katalogu src/assets plik house.svg otrzymany wraz z zadaniem.

Zamień zawartość pliku src/app/home/home.component.html do postaci:

<section>

<form>

<input type="text" placeholder="Filter by city">

<button class="primary" type="button">Search</button>

</form>

</section>

Zamień zawartość pliku src/app/home/home.component.css do postaci:

.results {

display: grid;

column-gap: 14px;

row-gap: 14px;

grid-template-columns: repeat(auto-fill, minmax(400px, 400px));

margin-top: 50px;

justify-content: space-around;

}

input[type="text"] {

border: solid 1px var(--primary-color);

padding: 10px;

border-radius: 8px;

margin-right: 4px;

display: inline-block;

width: 30%;

}

button {

padding: 10px;

border: solid 1px var(--primary-color);

background: var(--primary-color);

color: white;

border-radius: 8px;

}

@media (min-width: 500px) and (max-width: 768px) {

.results {

grid-template-columns: repeat(2, 1fr);

}

input[type="text"] {

width: 70%;

}

}

@media (max-width: 499px) {

.results {

grid-template-columns: 1fr;

}

}

Na koniec pliku src/styles.css dodaj:

@import url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,400;0,700;1,400;1,700&display=swap');

body {

font-family: 'Poppins', sans-serif;

}

:root {

--primary-color: #008148;

--secondary-color: #323B86;

--accent-color: #e8e7fa;

--shadow-color: #E8E8E8;

}

button.primary {

padding: 10px;

border: solid 1px var(--primary-color);

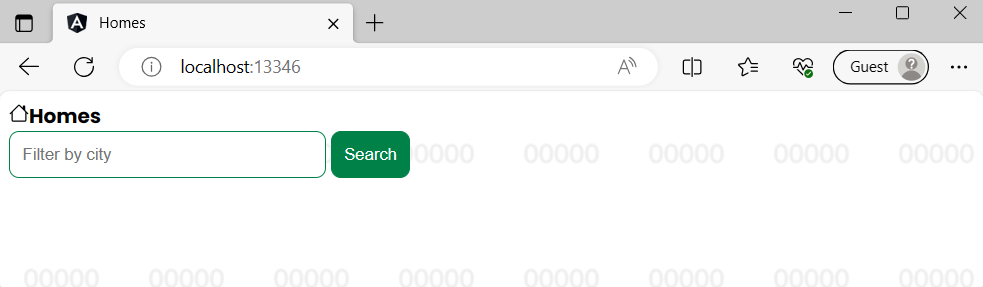
background: var(--primary-color);

color: white;

border-radius: 8px;

}

Oczekiwany efekt:



W przypadku problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-a-intro/commit/1ef8ff2b41a45bb925b3fc334886feaff158c0b0>

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej Twoją aplikację. Upewnij się, że na zrzucie ekranu widoczny jest adres URL z portem odpowiadającym Twojemu numerowi albumu oraz znak wodny z numerem albumu.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

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

# Komponent HousingLocation

W terminalu wykonaj polecenie:

> ng generate component housingLocation --standalone --skip-tests

Utworzony zostanie katalog src/app/housing-location.

Zmodyfikuj importy w dekoratorze @Component pliku src/app/home/home.component.ts do postaci:

imports: [

CommonModule,

HousingLocationComponent,

],

Zastąp zawartość src/app/home/home.component.html:

<section>

<form>

<input type="text" placeholder="Filter by city">

<button class="primary" type="button">Search</button>

</form>

</section>

<section class="results">

<app-housing-location></app-housing-location>

</section>

Ustaw zawartość src/app/housing-location/housing-location.css:

.listing {

background: var(--accent-color);

border-radius: 30px;

padding-bottom: 30px;

}

.listing-heading {

color: var(--primary-color);

padding: 10px 20px 0 20px;

}

.listing-photo {

height: 250px;

width: 100%;

object-fit: cover;

border-radius: 30px 30px 0 0;

}

.listing-location {

padding: 10px 20px 20px 20px;

}

.listing-location::before {

content: url("/assets/geo-alt-fill.svg") / "";

}

section.listing a {

padding-left: 20px;

text-decoration: none;

color: var(--primary-color);

}

section.listing a::after {

content: "\203A";

margin-left: 5px;

}

Wstaw do katalogu src/assets plik geo-alt-fill.svg otrzymany wraz z zadaniem.

Utwórz interfejs reprezentujący pojedynczy dom za pomocą polecenia terminala:

ng generate interface housinglocation

Utworzony zostanie plik src/app/housinglocation.ts. Zastąp jego zawartość poniższym kodem:

export interface HousingLocation {

id: number;

name: string;

city: string;

state: string;

photo: string;

availableUnits: number;

wifi: boolean;

laundry: boolean;

}

Dodamy teraz przykładowe domostwo. Wejdź do pliku src/app/home/home.component.ts i zastąp definicję klasy HomeComponent na następującą:

export class HomeComponent {

readonly baseUrl = 'https://angular.io/assets/images/tutorials/faa';

housingLocation: HousingLocation = {

id: 9999,

name: 'Test Home',

city: 'Test city',

state: 'ST',

photo: `${this.baseUrl}/example-house.jpg`,

availableUnits: 99,

wifi: true,

laundry: false,

};

}

Korzystając z IDE uzupełnij brakujące importy.

Zastąp definicję klasy HousingLocationComponent w pliku src/app/housing-location/housing-location.component.ts na poniższą:

export class HousingLocationComponent {

@Input() housingLocation!: HousingLocation;

}

Korzystając z IDE uzupełnij brakujące importy. Powyższy kod dodaje parametr wejściowy do komponentu HousingLocation o nazwie housingLocation.

Edytuj plik src/app/home/home.component.html do postaci:

<section>

<form>

<input type="text" placeholder="Filter by city">

<button class="primary" type="button">Search</button>

</form>

</section>

<section class="results">

<app-housing-location [housingLocation]="housingLocation"></app-housing-location>

</section>

Do app-housing-location dodaliśmy wiązanie właściwości housingLocation jako parametr housingLocation.

Na koniec dokonamy interpolacji wartości obiektu housingLocation w komponencie HousingLocation. Zastąp zawartość pliku src/app/housing-location/housing-location.component.html:

<section class="listing">

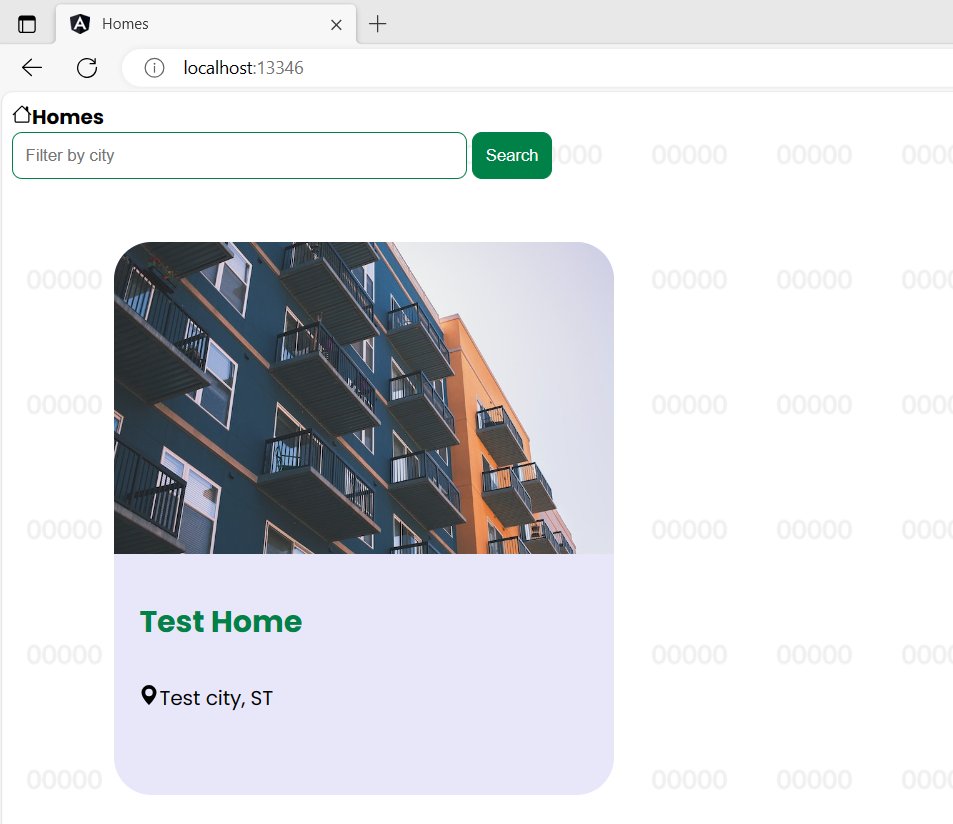
<img class="listing-photo" [src]="housingLocation.photo" alt="Exterior photo of {{housingLocation.name}}">

<h2 class="listing-heading">{{ housingLocation.name }}</h2>

<p class="listing-location">{{ housingLocation.city}}, {{housingLocation.state }}</p>

</section>

Oczekiwany efekt:



W przypadku problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-a-intro/commit/eaf2a8f77833258ea6eb5949050a8b84f5bbfba9>

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej Twoją aplikację. Upewnij się, że widoczny jest Test Home.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, multimedia

Opis wygenerowany automatycznie

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

# Iterowanie po obiektach

W tej sekcji zastąpimy pojedyncze wystąpienie budynku – listą wielu budynków. W tym celu wykorzystamy dyrektywę ngFor.

W pliku src/app/home/home.component.ts zastąp właściwość housingLocation nową właściwością housingLocationList:

housingLocationList: HousingLocation[] = [

{

id: 0,

name: 'Acme Fresh Start Housing',

city: 'Chicago',

state: 'IL',

photo: `${this.baseUrl}/bernard-hermant-CLKGGwIBTaY-unsplash.jpg`,

availableUnits: 4,

wifi: true,

laundry: true

},

{

id: 1,

name: 'A113 Transitional Housing',

city: 'Santa Monica',

state: 'CA',

photo: `${this.baseUrl}/brandon-griggs-wR11KBaB86U-unsplash.jpg`,

availableUnits: 0,

wifi: false,

laundry: true

},

{

id: 2,

name: 'Warm Beds Housing Support',

city: 'Juneau',

state: 'AK',

photo: `${this.baseUrl}/i-do-nothing-but-love-lAyXdl1-Wmc-unsplash.jpg`,

availableUnits: 1,

wifi: false,

laundry: false

},

]

W pliku src/app/home/home.component.html zmień wywołanie komponentu app-housing-location do postaci:

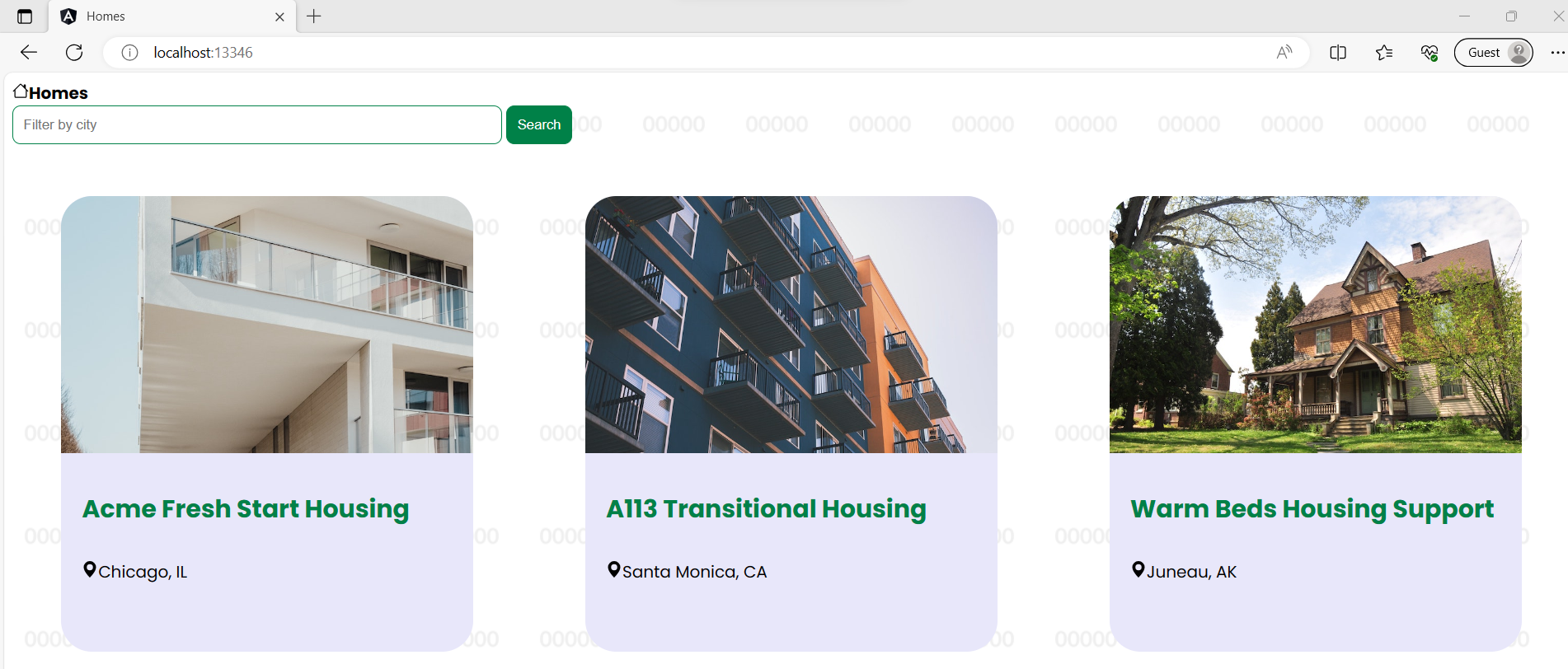
<app-housing-location

\*ngFor="let housingLocation of housingLocationList"

[housingLocation]="housingLocation">

</app-housing-location>

W efekcie szablon będzie iterować po wszystkich 3 testowych lokalizacjach. Oczekiwany efekt:



W przypadku problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-a-intro/commit/07b4642af3892c922aa5af3455d769f4a65c5ebc>

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej Twoją aplikację. Upewnij się, że widoczne są wszystkie 3 budynki.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

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

# Wyodrębnienie serwisu

W Angularze serwisy umożliwiają wyodrębnienie danych i funkcji, które mogą być wykorzystywane w wielu komponentach. Serwisy można następnie wstrzykiwać do komponentów za pomocą mechanizmu Dependency Injection.

Utwórz nowy serwis za pomocą polecenia terminala:

ng generate service housing --skip-tests

Utworzony zostanie plik src/app/housing.service.ts. Przenieś do niego (powyżej konstruktora) właściwości baseUrl i housingLocationList z pliku src/app/home/home.component.ts.

Następnie utwórz (poniżej konstruktora) dwie metody w klasie HousingService w pliku src/app/housing.service.ts:

getAllHousingLocations(): HousingLocation[] {

return this.housingLocationList;

}

getHousingLocationById(id: number): HousingLocation | undefined {

return this.housingLocationList.find(housingLocation => housingLocation.id === id);

}

W razie potrzeby uzupełnij brakujące importy z wykorzystaniem mechanizmów IDE.

W pliku src/app/home/home.component.ts podmień klasę HomeComponent na następującym kodem:

export class HomeComponent {

housingLocationList: HousingLocation[] = [];

housingService: HousingService = inject(HousingService);

constructor() {

this.housingLocationList = this.housingService.getAllHousingLocations();

}

}

W razie potrzeby uzupełnij brakujące importy z wykorzystaniem IDE. Powyższy kod wstrzykuje za pomocą inject() serwis HousingService do komponentu HomeComponent, a następnie w konstruktorze wykorzystuje jego metodę getAllHousingLocations() do pobrania wszystkich lokalizacji do właściwości housingLocationList. W ten sposób wynieśliśmy testowe dane o budynkach z komponentu do serwisu. W przyszłości zastąpimy dane testowe danymi pobieranymi z API.

W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-a-intro/commit/239536291f06d3940f3896eafb747af159c6dc97>

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

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu kodu pliku src/app/housing.service.ts:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

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

# Komponent ze szczegółami budynku. Routingi.

W tej sekcji zaimplementujemy drugą stronę – stronę ze szczegółami budynku.

Utwórz nowy komponent DetailsComponent z wykorzystaniem polecenia terminala:

> ng generate component details --standalone --skip-tests

Utworzony zostanie katalog src/app/details.

Utwórz nowy plik src/app/routes.ts o zawartości:

import {Routes} from "@angular/router";

import {HomeComponent} from "./home/home.component";

import {DetailsComponent} from "./details/details.component";

const routeConfig: Routes = [

{

path: '',

component: HomeComponent,

title: 'Home page',

},

{

path: 'details/:id',

component: DetailsComponent,

title: 'Home details',

}

]

export default routeConfig;

Jest to konfiguracja dwóch obsługiwanych routingów – strony głównej oraz strony szczegółów poszczególnych budynków identyfikowanych ich parametrem :id.

Zmodyfikuj dekorator @NgModule w pliku src/app/app.module.ts do postaci:

@NgModule({

declarations: [

AppComponent,

],

imports: [

BrowserModule,

HomeComponent,

RouterModule,

],

providers: [provideRouter(routeConfig)],

bootstrap: [AppComponent]

})

Wykorzystaj mechanizmy IDE do uzupełnienia brakujących importów.

W pliku src/app/app.component.html zastąp znacznik <app-home> znacznikiem <router-outlet> oraz dodaj link do strony głównej:

<main>

<a [routerLink]="['/']">

<header class="brand-name">

<img class="brand-logo" src="/assets/house.svg" alt="logo" aria-hidden="true">

<strong>Homes</strong>

</header>

</a>

<section class="content">

<router-outlet></router-outlet>

</section>

</main>

Następnie dodamy łącze pomiędzy listą budynków, a stroną szczegółów pojedynczego budynku. W tym celu edytuj plik src/app/housing-location/housing-location.component.html do postaci:

<section class="listing">

<img class="listing-photo" [src]="housingLocation.photo" alt="Exterior photo of {{housingLocation.name}}">

<h2 class="listing-heading">{{ housingLocation.name }}</h2>

<p class="listing-location">{{ housingLocation.city}}, {{housingLocation.state }}</p>

<a [routerLink]="['/details', housingLocation.id]">Learn More</a>

</section>

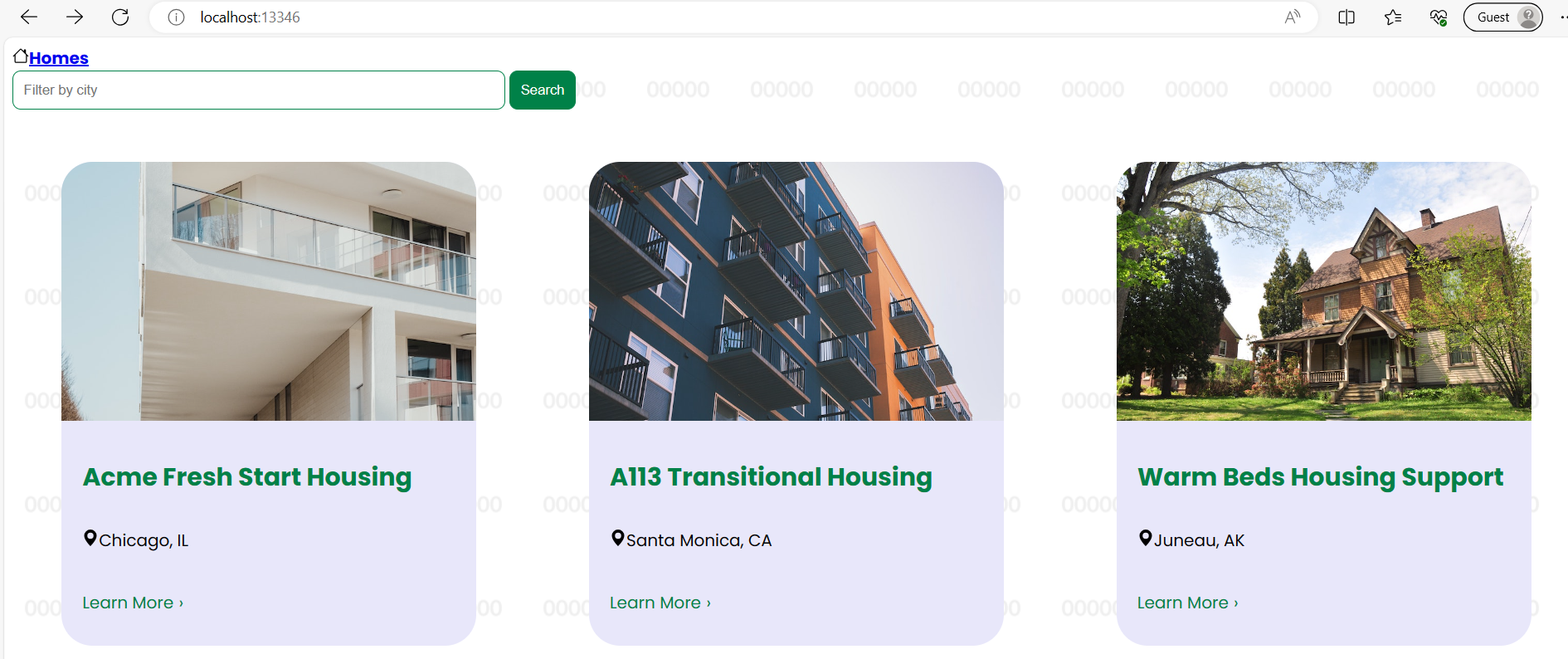
Dyrektywa routerLink będzie oznaczona jako błędna. Wykorzystaj mechanizmy IDE do automatycznego zaimportowania jej do dekoratora @Component w pliku src/app/housing-location/housing-location.component.ts:

import {RouterLink} from "@angular/router";

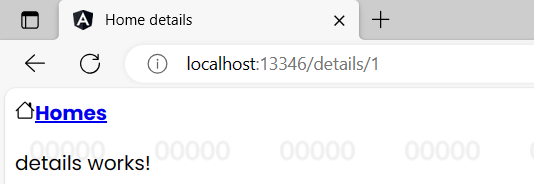
…

imports: [CommonModule, RouterLink],

Na liście powinny pojawić się linki:



Linki powinny umożliwić przejście do strony szczegółów (na razie bez treści):



Zmodyfikuj klasę DetailsComponent w pliku src/app/details/details.component.ts do postaci:

export class DetailsComponent {

route: ActivatedRoute = inject(ActivatedRoute);

housingService = inject(HousingService);

housingLocation: HousingLocation | undefined;

constructor() {

const housingLocationId = Number(this.route.snapshot.params['id']);

this.housingLocation = this.housingService.getHousingLocationById(housingLocationId);

}

}

Wykorzystaj IDE do uzupełnienia brakujących importów.

Zmień zawartość pliku src/app/details/details.component.html na poniższy kod:

<article>

<img class="listing-photo" [src]="housingLocation?.photo"

alt="Exterior photo of {{housingLocation?.name}}"/>

<section class="listing-description">

<h2 class="listing-heading">{{housingLocation?.name}}</h2>

<p class="listing-location">{{housingLocation?.city}}, {{housingLocation?.state}}</p>

</section>

<section class="listing-features">

<h2 class="section-heading">About this housing location</h2>

<ul>

<li>Units available: {{housingLocation?.availableUnits}}</li>

<li>Does this location have wifi: {{housingLocation?.wifi}}</li>

<li>Does this location have laundry: {{housingLocation?.laundry}}</li>

</ul>

</section>

</article>

Ustaw zawartość pliku src/app/details/details.component.css na poniższy kod:

.listing-photo {

height: 600px;

width: 50%;

object-fit: cover;

border-radius: 30px;

float: right;

}

.listing-heading {

font-size: 48pt;

font-weight: bold;

margin-bottom: 15px;

}

.listing-location::before {

content: url('/assets/geo-alt-fill.svg') / '';

}

.listing-location {

font-size: 24pt;

margin-bottom: 15px;

}

.listing-features > .section-heading {

color: var(--secondary-color);

font-size: 24pt;

margin-bottom: 15px;

}

.listing-features {

margin-bottom: 20px;

}

.listing-features li {

font-size: 14pt;

}

li {

list-style-type: none;

}

.listing-apply .section-heading {

font-size: 18pt;

margin-bottom: 15px;

}

label, input {

display: block;

}

label {

color: var(--secondary-color);

font-weight: bold;

text-transform: uppercase;

font-size: 12pt;

}

input {

font-size: 16pt;

margin-bottom: 15px;

padding: 10px;

width: 400px;

border-top: none;

border-right: none;

border-left: none;

border-bottom: solid .3px;

}

@media (max-width: 1024px) {

.listing-photo {

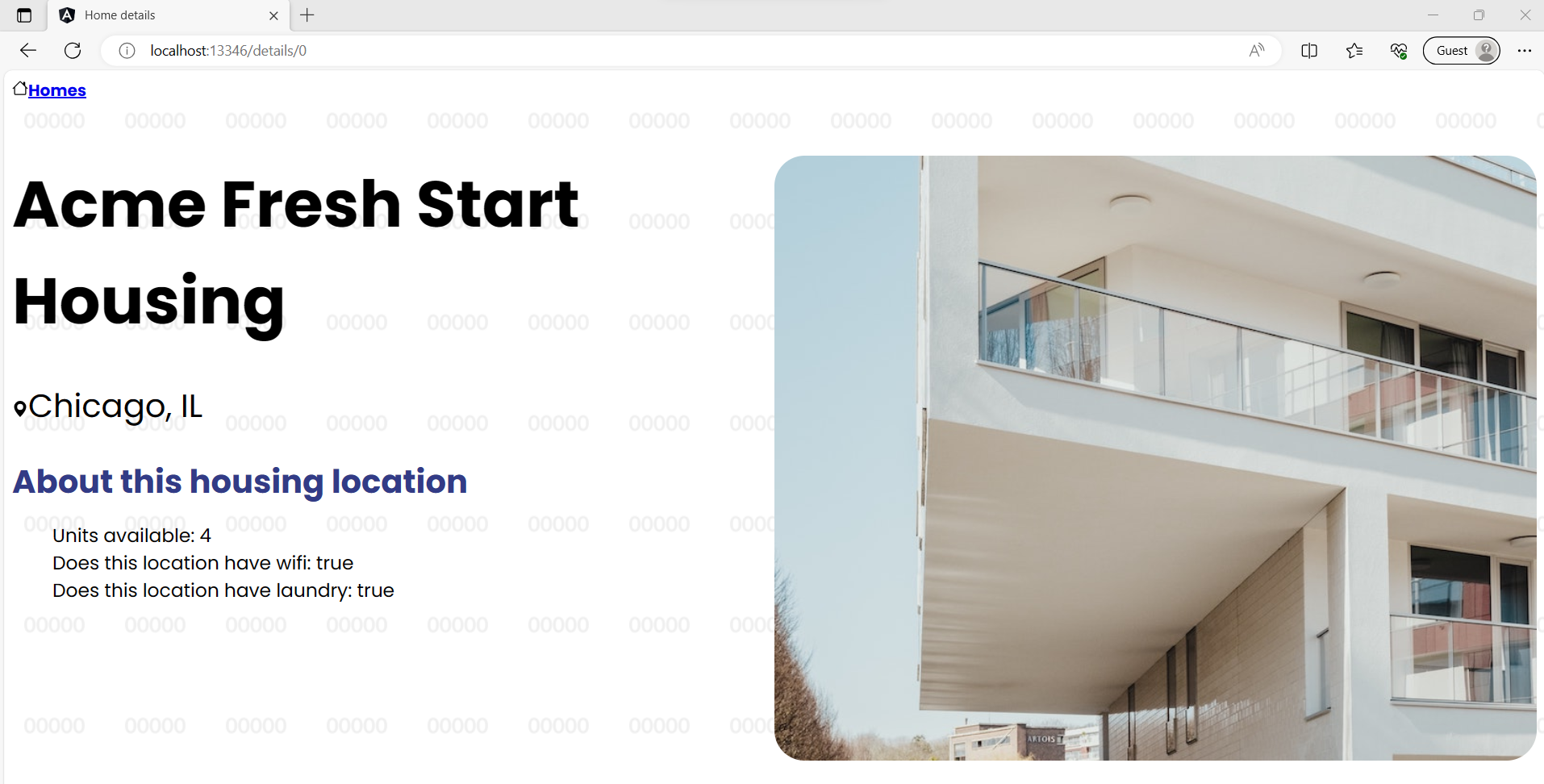
width: 100%;

height: 400px;

}

}

Oczekiwany efekt:



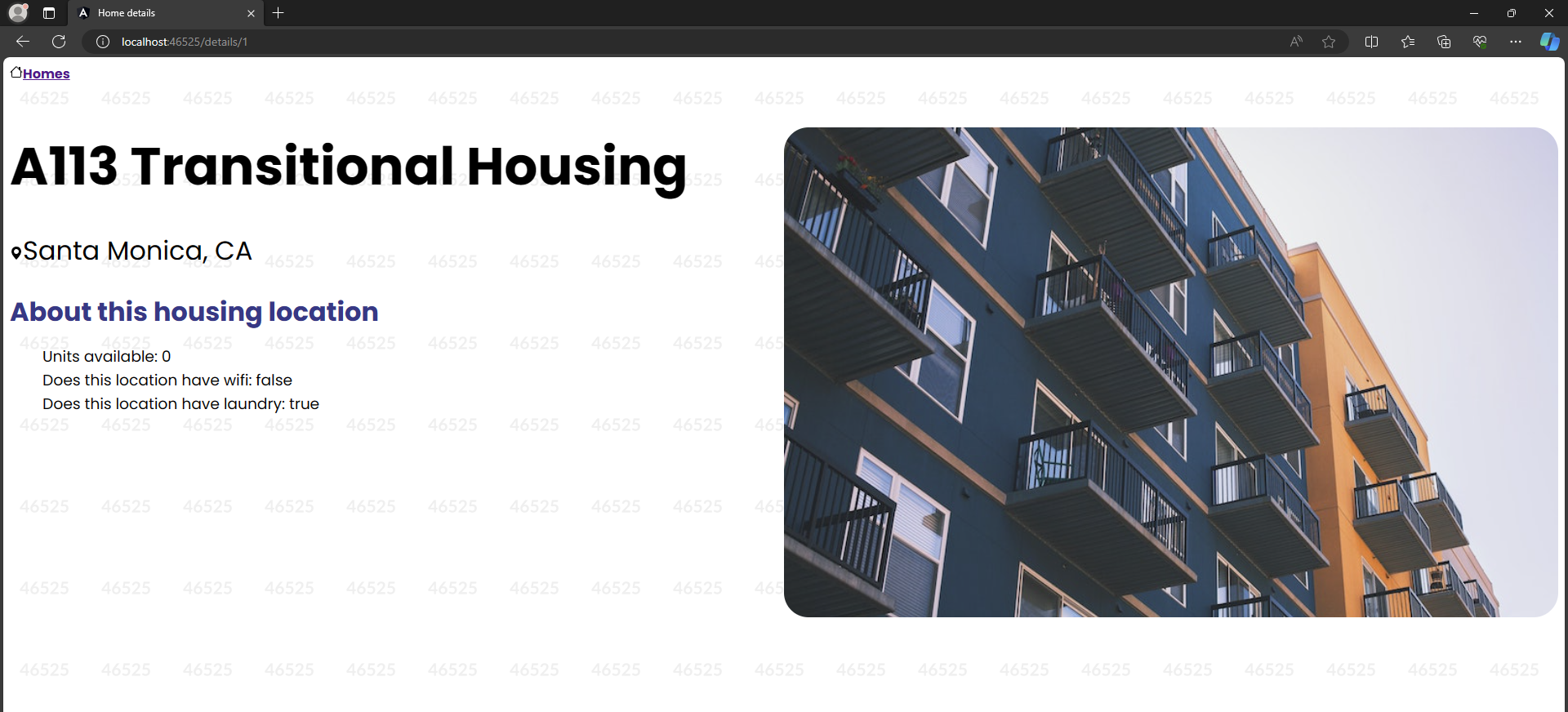
W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-a-intro/commit/6d602daf3001c9054a6fad469b3f2b42aaccb894>

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę szczegółów budynku. Upewnij się, że widoczny jest adres URL, lokalizacja budynku, zdjęcie.

Obraz zawierający tekst, zrzut ekranu, okno

Opis wygenerowany automatycznie

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę szczegółów innego budynku. Upewnij się, że widoczny jest adres URL, lokalizacja budynku, zdjęcie.



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

# Wyszukiwarka

W tej sekcji dodamy możliwość wyszukiwania budynków po lokalizacji.

W tym celu do klasy HomeComponent w pliku src/app/home/home.component.ts dodaj właściwość:

filteredLocationList: HousingLocation[] = [];

Będzie ona przechowywać odfiltrowane wyniki, zgodnie z miejscowością wpisaną w wyszukiwarkę. Następnie w konstruktorze zainicjalizujemy tę właściwość na listę wszystkich dostępnych budynków:

constructor() {

this.housingLocationList = this.housingService.getAllHousingLocations();

this.filteredLocationList = this.housingLocationList;

}

Poniżej tak zmodyfikowanego konstruktora, dodaj metodę odpowiedzialną za filtrowanie wyników:

filterResults(text: string) {

if (!text) {

this.filteredLocationList = this.housingLocationList;

}

this.filteredLocationList = this.housingLocationList.filter(

housingLocation => housingLocation?.city.toLowerCase().includes(text.toLowerCase())

);

}

Działanie tej metody jest następujące – w przypadku braku frazy do wyszukiwania – ustaw listę odfiltrowanych budynków na wszystkie dostępne. W przeciwnym wypadku ustaw listę odfiltrowanych budynków wyłącznie na te, których nazwa miejscowości uwzględnia wpisaną frazę (wszystko porównywane małymi literami).

W pliku src/app/home/home.component.html dodaj do kontrolki input atrybut #filter. Jest to zmienna szablonu:

<input type="text" placeholder="Filter by city" #filter>

Zmodyfikuj kontrolkę button o wywołanie powyższej metody filterResults():

<button class="primary" type="button" (click)="filterResults(filter.value)">Search</button>

Na koniec zmień housingLocationList na filteredLocationList w dyrektywie \*ngFor:

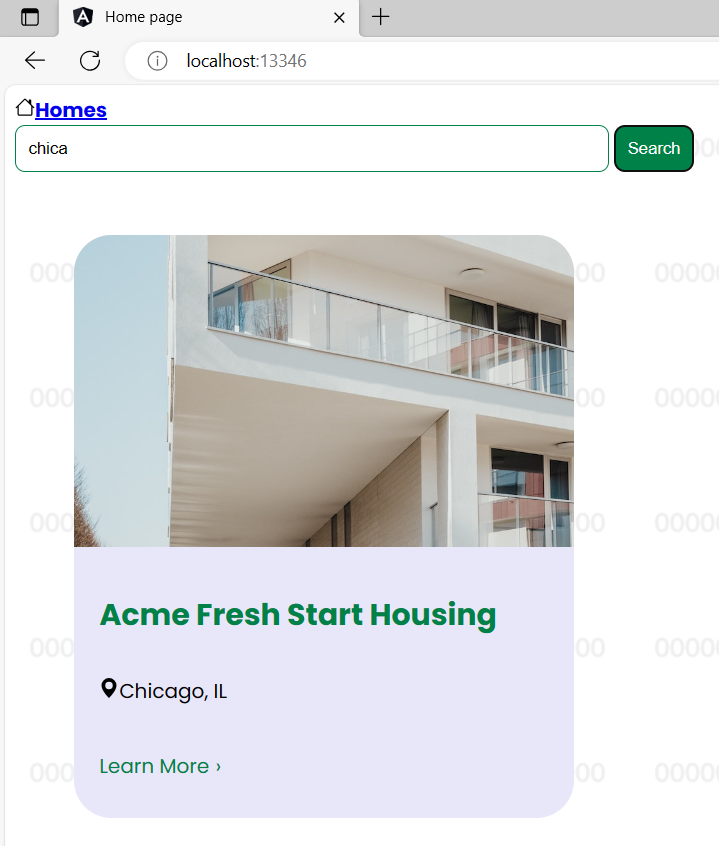
<app-housing-location

\*ngFor="let housingLocation of filteredLocationList"

[housingLocation]="housingLocation">

</app-housing-location>

Filtrowanie powinno działać (wyłącznie po kliknięciu na przycisk Search):



W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-a-intro/commit/511b937c1aaceff05b8664242bdf0ed14cd15901>

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę listy z odfiltrowanym jednym budynkiem.

![Obraz zawierający biały, design

Opis wygenerowany automatycznie](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAFoAAABSCAIAAABbmVQ4AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAAEnQAABJ0Ad5mH3gAAACgSURBVHhe7dAxAQAwEAOh+rcVY13/PIAE3jh0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdBzbB7Ne5ESwH6TiAAAAAElFTkSuQmCC)

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę listy z odfiltrowanym innym budynkiem.

![Obraz zawierający biały, design

Opis wygenerowany automatycznie](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAFoAAABSCAIAAABbmVQ4AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAAEnQAABJ0Ad5mH3gAAACgSURBVHhe7dAxAQAwEAOh+rcVY13/PIAE3jh0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdBzbB7Ne5ESwH6TiAAAAAElFTkSuQmCC)

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

# Podłączenie API

Do głównego katalogu projektu wgraj otrzymany wraz z zadaniem plik db.json. W terminalu wykonaj polecenie, które zainstaluje json-server:

> npm install -g json-server

Uruchom serwer poleceniem (zastąp 00000 przez swój numer albumu, obniżony o 1000):

> json-server --watch --port=00000 db.json

Po uruchomieniu serwera, można odwiedzić w przeglądarce testowe API:

* http://localhost:00000/locations
* http://localhost:00000/locations/1

Zmodyfikuj plik src/app/housing.service.ts. Usuń właściwości baseUrl i housingLocationList. Zamiast nich utwórz właściwość url wskazującą na adres uruchomionego json-servera, np.:

url = 'http://localhost:12345/locations';

Zmodyfikuj metody getAllHousingLocations i getHousingLocationById:

async getAllHousingLocations(): Promise<HousingLocation[]> {

const data = await fetch(this.url);

return await data.json() ?? [];

}

async getHousingLocationById(id: number): Promise<HousingLocation | undefined> {

const data = await fetch(`${this.url}/${id}`);

return await data.json() ?? {};

}

Następnie trzeba zmodyfikować kod HomeComponent w pliku src/app/home/home.component.ts, żeby wykorzystywał zmodyfikowane asynchroniczne metody serwisu:

constructor() {

this.housingService.getAllHousingLocations().then((housingLocationList: HousingLocation[]) => {

this.housingLocationList = housingLocationList;

this.filteredLocationList = housingLocationList;

});

}

I podobnie z DetailsComponent w pliku src/app/details/details.component.ts:

constructor() {

const housingLocationId = Number(this.route.snapshot.params['id']);

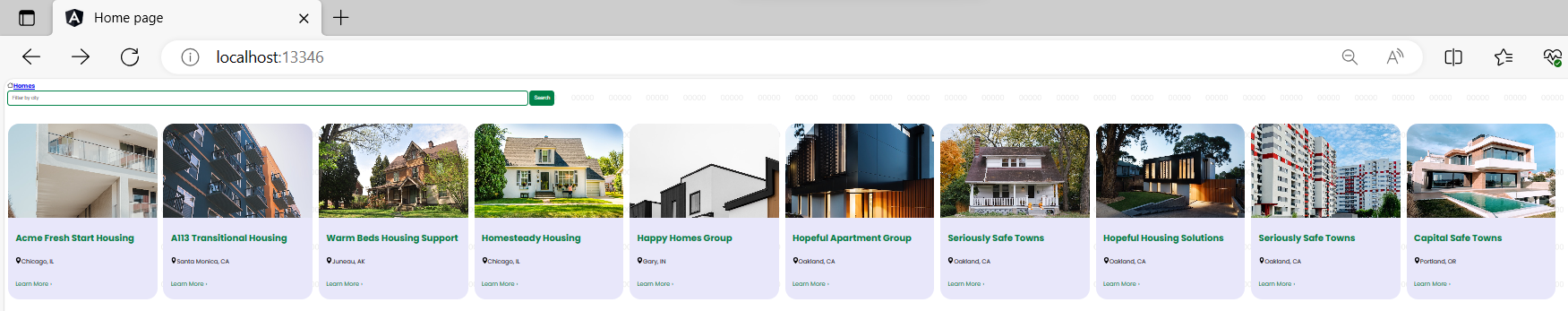
this.housingService.getHousingLocationById(housingLocationId).then(housingLocation => {

this.housingLocation = housingLocation;

});

}

W efekcie aplikacja powinna działać z wykorzystaniem zewnętrznego testowego API:



W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2b-lab-a-intro/commit/e8b4d089a0e653fcdacb142f5d0d91d212c02d05>

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę listy budynków bez filtrowania.

![Obraz zawierający biały, design

Opis wygenerowany automatycznie](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAFoAAABSCAIAAABbmVQ4AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAAEnQAABJ0Ad5mH3gAAACgSURBVHhe7dAxAQAwEAOh+rcVY13/PIAE3jh0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdBzbB7Ne5ESwH6TiAAAAAElFTkSuQmCC)

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę listy budynków z zastosowanym filtrem po miejscowości.

![Obraz zawierający biały, design

Opis wygenerowany automatycznie](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAFoAAABSCAIAAABbmVQ4AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAAEnQAABJ0Ad5mH3gAAACgSURBVHhe7dAxAQAwEAOh+rcVY13/PIAE3jh0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdBzbB7Ne5ESwH6TiAAAAAElFTkSuQmCC)

Umieść poniżej zrzut ekranu przeglądarki wyświetlającej stronę szczegółów pojedynczego budynku.

![Obraz zawierający biały, design

Opis wygenerowany automatycznie](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAFoAAABSCAIAAABbmVQ4AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAAEnQAABJ0Ad5mH3gAAACgSURBVHhe7dAxAQAwEAOh+rcVY13/PIAE3jh0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdISO0BE6QkfoCB2hI3SEjtAROkJH6AgdoSN0hI7QETpCR+gIHaEjdBzbB7Ne5ESwH6TiAAAAAElFTkSuQmCC)

|  |  |  |
| --- | --- | --- |
| 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-a 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-a w swoim repozytorium:

…link…

# Podsumowanie

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

…podsumowanie…

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.