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

Komunikacja z Backend

# Spis treści

[Spis treści 1](#_Toc151306336)

[Cel zajęć 1](#_Toc151306337)

[Rozpoczęcie 1](#_Toc151306338)

[Uwaga 1](#_Toc151306339)

[Omówienie wytycznych aplikacji 2](#_Toc151306340)

[Inicjalizacja projektu 2](#_Toc151306341)

[Serwer Backend API 3](#_Toc151306342)

[Interfejs i serwis 5](#_Toc151306343)

[Komponent Tasks 7](#_Toc151306344)

[Implementacja komponentu ArchiveComponent 14](#_Toc151306345)

[Commit projektu do GIT 17](#_Toc151306346)

[Podsumowanie 17](#_Toc151306347)

# Cel zajęć

Celem głównym zajęć jest zdobycie umiejętności:

* obsługi routingu w Angular,
* komunikacji z wartstwą backend za pomocą serwisów.

W praktycznym wymiarze stworzona zostanie prosta aplikacja „Task manager” do zapisywania zadań do wykonania.

# 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.

# Omówienie wytycznych aplikacji

Widok bieżących zadań:

* Umożliwia dodanie nazwy nowego zadania wraz z opcjonalną datą jego wykonania. Data powinna być ustawiana poprzez standardowy element formularza HTM5 typu type=”date”.
* Wyświetla listę aktualnych zadań.
* Umożliwia zaznaczenie wybranego zadania jako wykonane poprzez użycie checkbox-a.
* Umożliwia przesunięcie wszystkich zadań oznaczonych jako wykonane do archiwum.

Widok zadań zarchiwizowanych:

* Wyświetla listę zadań o statusie archiwalnym
* Umożliwia trwałe kasowanie wybranych zadań poprzez kliknięcie umieszczonego w zadaniu przycisku [usuń].

# Inicjalizacja projektu

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

> ng new lab-d

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-d

> ng serve --port=

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

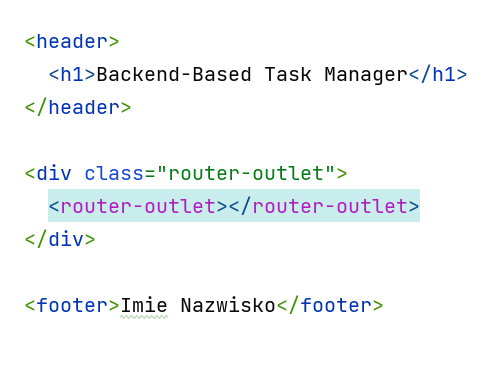
body {

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

}

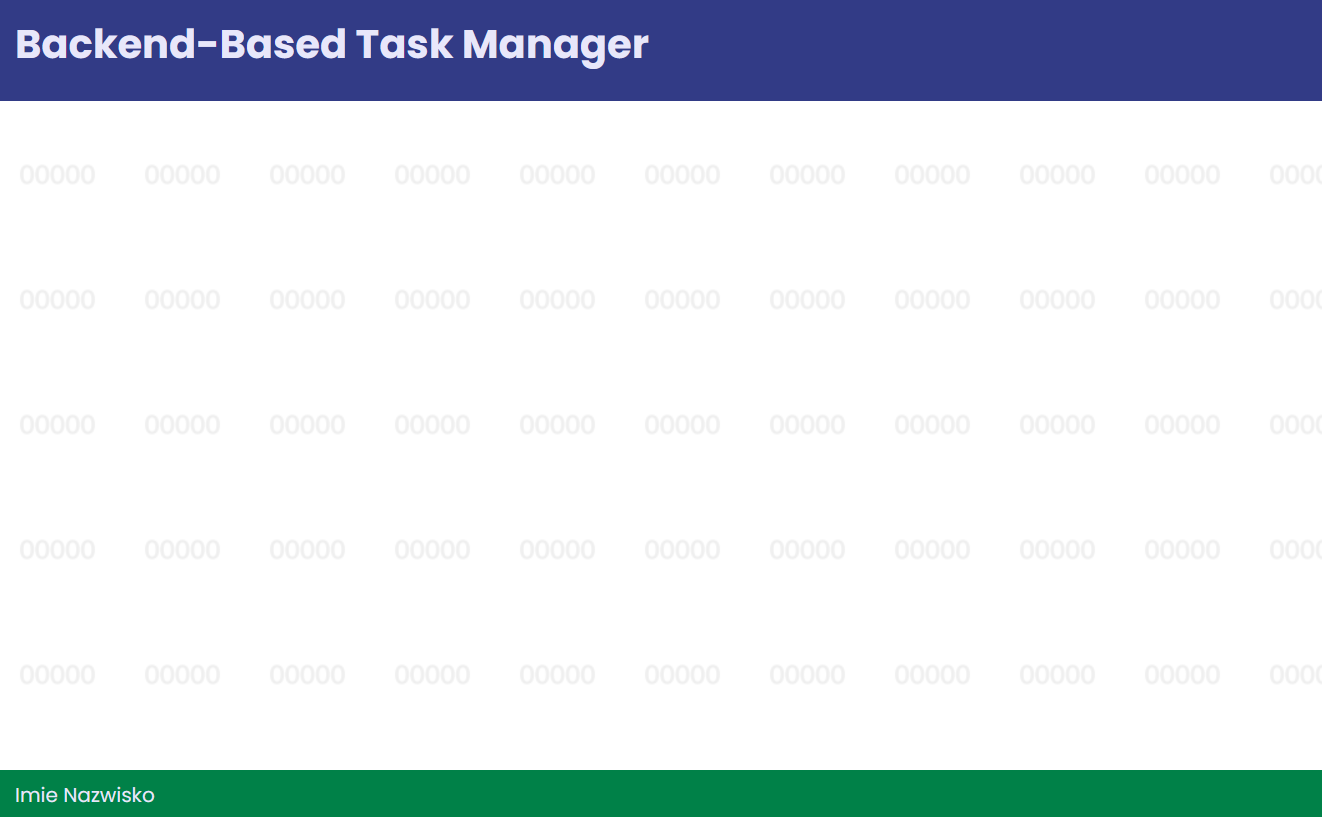
Uruchom przeglądarkę pod adresem: http://localhost:00000.

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.:



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

Przykładowy efekt:



# Serwer Backend API

Do głównego katalogu projektu wgraj otrzymany wraz z zadaniem plik todos.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 todos.json

Po uruchomieniu serwera, można odwiedzić w przeglądarce testowe API:

* http://localhost:00000/todos
* http://localhost:00000/todos/1

Lista dostępnych operacji i przykładowe wywołania znajdują się poniżej (kompatybilne z klientem HTTP PhpStorm/WebStorm):

### GET list of tasks

GET {{baseUrl}}/

Accept: application/json

### GET list of non-comleted tasks

GET {{baseUrl}}/?completed=false

Accept: application/json

### GET list of comleted tasks

GET {{baseUrl}}/?completed=true

Accept: application/json

### GET list of archived tasks

GET {{baseUrl}}/?archived=false

Accept: application/json

### GET list of tasks sorted by ID ASC

GET {{baseUrl}}/?\_sort=id&\_order=asc

Accept: application/json

### GET list of tasks sorted by ID DESC

GET {{baseUrl}}/?\_sort=id&\_order=desc

Accept: application/json

### GET single task

GET {{baseUrl}}/1

Accept: application/json

### POST new task

POST {{baseUrl}}/

Content-Type: application/json

{

"title": "Pierwsze zadanie",

"deadline": "2022-01-28",

"completed": false,

"archived": false

}

### PUT update task

PUT {{baseUrl}}/3

Content-Type: application/json

{

"title": "Pierwsze zadanie",

"deadline": "2022-01-28",

"completed": true,

"archived": false

}

### DELETE task

DELETE {{baseUrl}}/3

Content-Type: application/json

Szczególną uwagę należy zwrócić na filtrowanie po polu completed i archived oraz sortowaniu po dowolnym polu.

Uruchom json-serwer i przetestuj wybrane metody REST serwisu używając wybranego klienta REST (Postman, PhpStorm, Curl).

Przedstaw zrzut ekranu przedstawiającego zapytanie i odpowiedź do endpointa GET /

Obraz zawierający tekst, Czcionka, zrzut ekranu, numer

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Przedstaw zrzut ekranu przedstawiającego zapytanie i odpowiedź do endpointa POST /

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Przedstaw zrzut ekranu przedstawiającego zapytanie i odpowiedź do endpointa PUT / zmieniającego status zadania na skończone:

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Przedstaw zrzut ekranu przedstawiającego zapytanie i odpowiedź do endpointa PUT / zmieniającego status zadania na zarchiwizowane:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Przedstaw zmodyfikowaną zawartość pliku todos.json po wykonaniu powyższych zapytań. Zaznacz wprowadzone zmiany względem oryginału.

Obraz zawierający tekst, zrzut ekranu, Oprogramowanie multimedialne, oprogramowanie

Opis wygenerowany automatycznie

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

# Interfejs i serwis

Utwórz interfejs Task z wykorzystaniem polecenia:

ng generate interface task

Utworzony zostanie plik src/app/task.ts. Zmień jego zawartość na odpowiadającą przetestowanemu powyżej API, przykładowo:

export interface Task {

id?: number;

title?: string;

deadline?: Date;

completed?: boolean;

archived?: boolean;

}

Dzięki zastosowaniu znaku ? umożliwiamy przechowywanie w interfejsie pól niezdefiniowanych, co przyda się przy wykorzystaniu interfejsu nie tylko w opisie elementów listy zadań, ale i przy zapytaniach POST i PUT, gdzie nie wszystkie pola są obecne.

Następnie wygeneruj serwis TasksService z wykorzystaniem polecenia:

ng generate service tasks --skip-tests

Utworzony zostanie plik src/app/tasks.service.ts. Ustaw jego zawartość na:

import { Injectable } from '@angular/core';

import {Observable} from "rxjs";

import {Task} from "./task";

import {HttpClient} from "@angular/common/http";

@Injectable({

providedIn: 'root'

})

export class TasksService {

constructor(

private http: HttpClient,

) { }

public index(archived = false): Observable<Task[]> {

// ...

}

public post(task: Task): Observable<Task> {

// ...

}

public put(task: Task): Observable<Task> {

// ...

}

public delete(task: Task): Observable<any> {

// ...

}

}

Zaimplementuj ciała funkcji index(), post(), put(), delete(). W przypadku index() uwzględnij przekazanie parametru archived do pozyskania listy zadań aktywnych i archiwalnych.

W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2-lab-d-todo/commit/db67085e6949ffe587ee512e7266f27b91c95ba7>

Wstaw zrzut ekranu metody index() serwisu TasksService:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Wstaw zrzut ekranu metody post() serwisu TasksService:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Wstaw zrzut ekranu metody put() serwisu TasksService:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Wstaw zrzut ekranu metody delete() serwisu TasksService:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

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

# Komponent Tasks

Wygeneruj komponent TasksComponent używając polecenia terminala:

> ng generate component tasks --skip-tests

Utworzony zostanie katalog src/app/tasks. W klasie komponentu w pliku src/app/tasks/tasks.component.ts dodaj pola tasks: Task[] and newTask: Task, odpowiednio do przechowywania pobranej z backendu listy zadań oraz do edytowania nowego zadania. Pola te będą powiązane z widokiem.

W konstruktorze wstrzyknij serwis TasksService, a pod konstruktorem zaimplementuj metodę ngOnInit, która podczas inicjalizacji komponentu wypełni go aktualnymi zadaniami (archived === false).

W pliku src/app/app-routing.module.ts skonfiguruj routingi:

const routes: Routes = [

{path: 'tasks', component: TasksComponent},

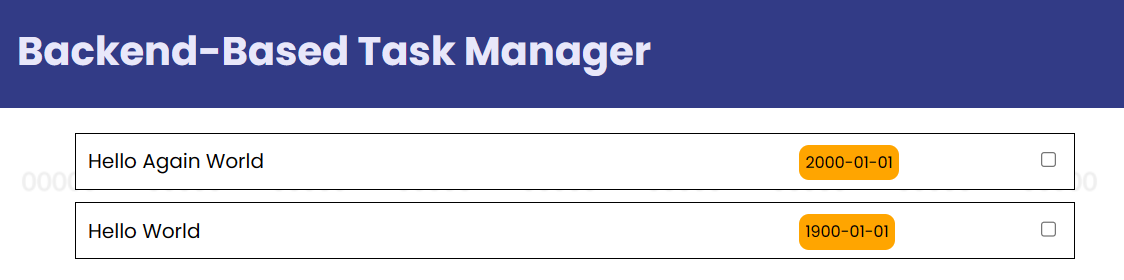
{path: '', redirectTo: '/tasks', pathMatch: 'full'},

];

Zaimplementuj widok listy zadań (na razie bez akcji).

Przykładowa implementacja:



W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2-lab-d-todo/commit/1377e7c8f819acca1f9a6898e98a8d3310bff9bc>

Wstaw zrzut ekranu listy zadań:

![Obraz zawierający biały, design

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

Z niewiadomych przyczyn lista jest pusta cały czas

Wstaw zrzut ekranu kodu src/app/tasks/tasks.component.ts:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Wstaw zrzut ekranu kodu src/app/tasks/tasks.component.html:

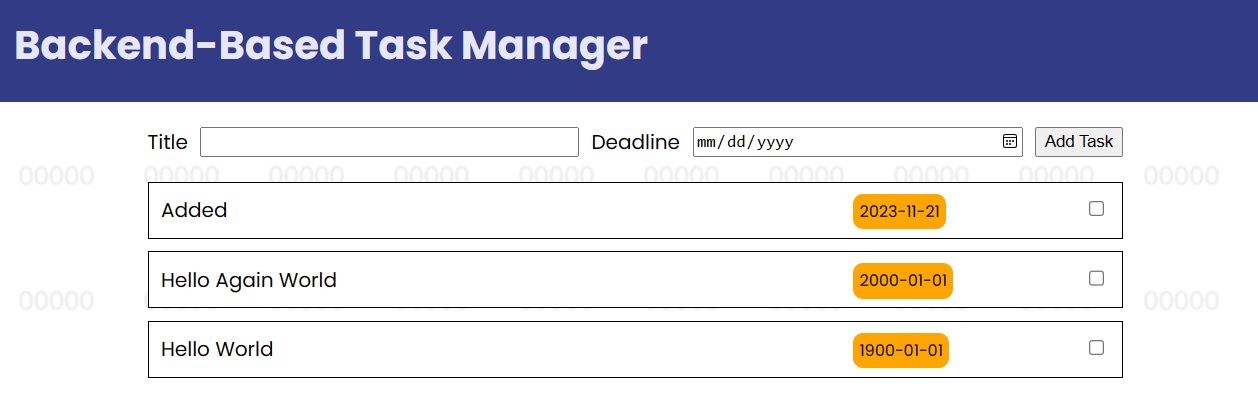
Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

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

Następnie zaimplementuj widok formularza dodawania nowego zadania, powyżej lub poniżej listy zadań. Pola formularza powinny być połączone z pole newTask. Zaimplementuj i wykorzystaj metodę addTask(), która zbierze dane nowego zadania i doda zadanie korzystając z metody post() serwisu.

Przykładowa implementacja:







W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2-lab-d-todo/commit/76ada7324473c9aa60df2e4dc213297f44ad5620>

Wstaw zrzut ekranu listy zadań z wypełnionym formularzem dodawania zadania:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Wstaw zrzut ekranu listy zadań po dodaniu zadania:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

Do bazy dodało

Wstaw zrzut ekranu kodu src/app/tasks/tasks.component.ts:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

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

Opis wygenerowany automatycznie

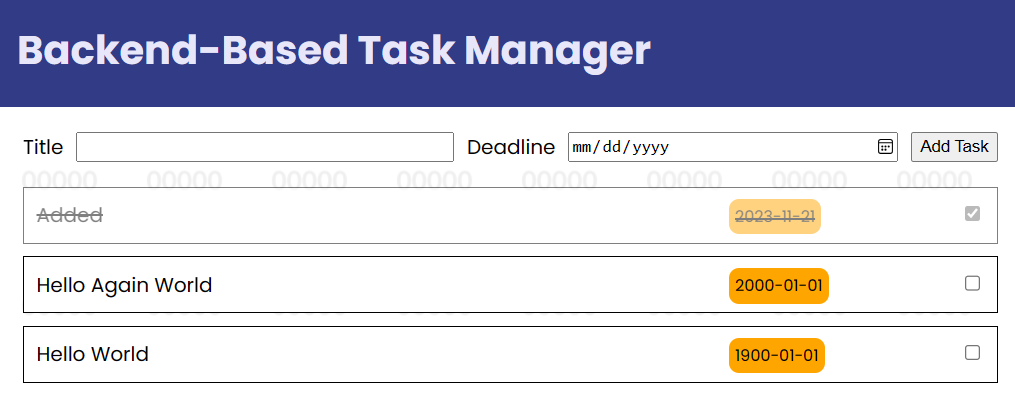
Wstaw zrzut ekranu kodu src/app/tasks/tasks.component.html:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

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

Następnym krokiem jest implementacja oznaczania zadań jako zakończone. Powiąż checkboksy z właściwością completed dla poszczególnych zadań na liście za pomocą [(ngModel)] oraz dowiąż zdarzenie (change) z nową metodą handleChange(task: Task), która wysyła zaktualizowane zadanie do backendu. Zmień styl zakończonych zadań. Przykładowa implementacja:





W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2-lab-d-todo/commit/4d604c8651a31e648841e8f4a651ad8c39b7c3e2>

Wstaw zrzut ekranu żądania PUT w narzędziach deweloperskich po zmianie statusu zadania:

![Obraz zawierający biały, design

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

Wstaw zrzut ekranu odpowiedniego fragmentu kodu src/app/tasks/tasks.component.ts:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Wstaw zrzut ekranu odpowiedniego fragmentu kodu src/app/tasks/tasks.component.html:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

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

Na koniec implementacji komponentu TasksComponent, na widoku dodaj przycisk Archive Completed, powiązany z metodą archiveCompleted, która zarchiwizuje wszystkie zadania oznaczone jako skończone, tj. ustawi ich pole archived na wartość true i wyśle aktualizację do backendu. Z wykorzystaniem forkJoin poczekaj aż wszystkie aktualizacje się zakończą i odśwież komponent ponownie wywołując ngOnInit().

Przykładowa implementacja:



W razie problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2-lab-d-todo/commit/64f7463729d34c7edee0cb49716c5c4eedb1dd39>

Wstaw zrzut ekranu zawartości pliku todos.json przed archiwizacją:

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

Wstaw zrzut ekranu zawartości pliku todos.json po archiwizacji:

Brak zmian po stronie pliku todos.json, ponieważ aplikacja nie wczytuje listy todos’ów w niewiadomych przyczyn (dlatego też ich nie wyświetla)

Wstaw zrzut ekranu zakładki Network przeglądarki, obrazujący wysłanie wielu zapytań PUT po kliknięciu na przycisk archiwizacji zadań:

Nie wykonuje się, bo aplikacja nie zaczytuje zadań.

Wstaw zrzut ekranu odpowiedniego fragmentu kodu src/app/tasks/tasks.component.ts:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Wstaw zrzut ekranu odpowiedniego fragmentu kodu src/app/tasks/tasks.component.html:



Wstaw zrzut ekranu odpowiedniego fragmentu kodu src/app/tasks/tasks.component.css:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

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

# Implementacja komponentu ArchiveComponent

W tej części zaimplementujemy komponent ArchiveComponent odpowiedzialny za wyświetlanie archiwum zadań oraz ich kasowanie.

Wygeneruj komponent ArchiveComponent używając polecenia terminala:

> ng generate component archive --skip-tests

Utworzony zostanie katalog src/app/archive. Wstrzyknij do niego serwis TasksService i zaimplementuj metodę ngOnInit(), tak żeby podczas inicjalizacji komponent wypełnił się wyłącznie zarchiwizowanymi zadaniami. Zaimplementuj metodę delete(task: Task), która permanentnie usunie zadanie, wykorzystując serwis TasksService.

Dodaj do routingu ścieżkę do archiwum:

const routes: Routes = [

{path: 'tasks', component: TasksComponent},

{path: 'archive', component: ArchiveComponent},

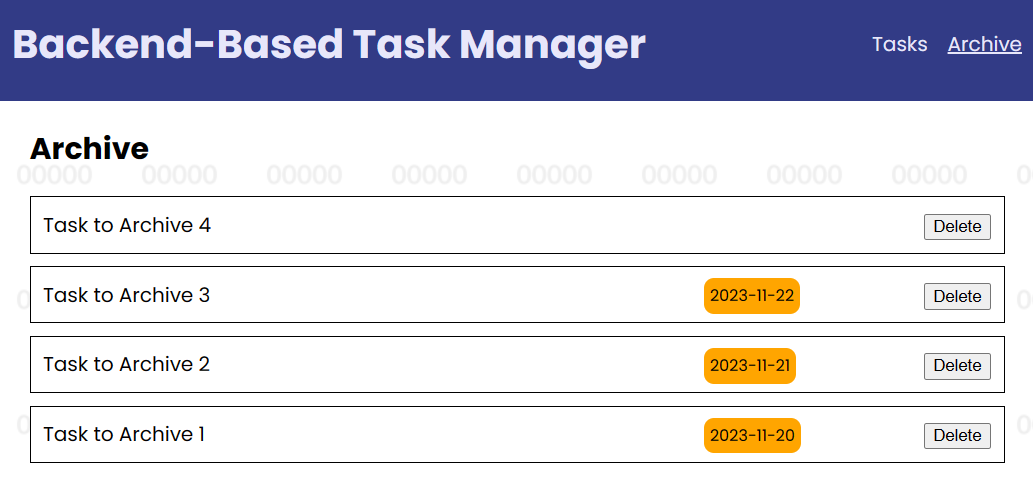
{path: '', redirectTo: '/tasks', pathMatch: 'full'},

];

Dodaj do widoku głównego komponentu AppComponent łącza do obydwu widoków.

Zaimplementuj widok listy archiwum wraz z przyciskami Delete przy każdym zadaniu, odpowiedzialnymi za permanentne kasowanie zadania.

Przykładowa implementacja:





W przypadku problemów, porównaj kod: <https://github.com/ideaspot-pl/ai2-lab-d-todo/commit/0b5722430094c50313b5a87f9d5cdbbb56a451d3>

Wstaw zrzut ekranu strony archiwum wypełnionej zadaniami:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznieDalej to samo, a w konsoli brak błędów

Wstaw zrzut ekranu zawartości pliku todos.json przed kasowaniem archiwalnych zadań:

![Obraz zawierający biały, design

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

Wstaw zrzut ekranu strony archiwum po skasowaniu archiwalnych zadań:

![Obraz zawierający biały, design

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

Wstaw zrzut ekranu zawartości pliku todos.json po skasowaniu archiwalnych zadań:

![Obraz zawierający biały, design

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

Wstaw zrzut ekranu odpowiedniego fragmentu kodu src/app/archive/archive.component.ts:

![Obraz zawierający biały, design

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

Wstaw zrzut ekranu odpowiedniego fragmentu kodu src/app/archive/archive.component.html:

![Obraz zawierający biały, design

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

Wstaw zrzut ekranu odpowiedniego fragmentu kodu src/app/archive/archive.component.css:

![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-d 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-d w swoim repozytorium:

…link…

# Podsumowanie

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

…podsumowanie…

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.