2025

RAPPORT COMPLET DU PROJET TUDO

APPLICATION FULL-STACK RÉALISÉE AVEC SPRING BOOT, POSTGRESQL ET ANGULAR

ACER

Rediger par : hanane benbel

N : 04

1. Introduction

Le projet « TUDO » est une application web permettant la gestion de tâches quotidiennes. Il a été développé dans le but d'illustrer la conception d'une application **full-stack** reposant sur :

- **Spring Boot** pour le backend (API REST),
- **PostgreSQL** pour la gestion des données,
- **Angular** pour la partie interface utilisateur.

Dans sa version initiale, l'application permettait uniquement :

- d'afficher la liste des TUDO,
- de cocher/décocher une tâche,
- de supprimer un TUDO.

2. Technologies utilisées

2.1 Backend : Spring Boot

Spring Boot a été choisi pour sa simplicité de configuration ainsi que sa capacité à exposer des API REST robustes.
Principales caractéristiques :

- Architecture Model-Service-Controller,
- Utilisation de Spring Data JPA pour l'accès aux données,
- Validation des données via Jakarta Validation,
- Gestion centralisée des erreurs via `RestControllerAdvice`.

2.2 Base de données : PostgreSQL

PostgreSQL a été utilisé comme SGBD relationnel pour stocker les TUDO.
Avantages :

- Fiabilité,
- Support JSON,
- Gestion efficace des transactions,
- Intégration fluide avec Spring Boot via JDBC / JPA.

2.3 Frontend : Angular

Angular a été choisi pour ses capacités de construction d'interfaces dynamiques basées sur des composants.
Fonctionnalités clés utilisées :

- Services HTTP pour consommer l'API Spring Boot,
- Composants dynamiques pour l'affichage des tâches,
- Data binding et événements pour l'interaction utilisateur,
- Angular Material pour les icônes et l'esthétique.

3. Architecture générale du projet

Le projet adopte une architecture **en couches**, favorisant la clarté et la maintenabilité.

3.1 Couches Backend

- **Controller** : expose les endpoints REST.
- **Service** : contient la logique métier et les traitements.
- **Repository** : communique directement avec la base de données.
- **Entity** : représentation objet des données persistées.

3.2 Frontend Angular

- **Composants (Component)** : affichage et interactions.
- **Services** : communication HTTP avec l'API.
- **Template HTML** : définition de l'interface.
- **Styles** : apparence graphique du projet.

3.3 Base de données

- Une table `todos` contenant :
  - id (clé primaire),
  - title (texte),
  - completed (booléen).

4. Fonctionnalités initiales

4.1 Affichage de tous les TUDO

L'API fournit un endpoint `GET /api/todos` pour récupérer toutes les tâches.
Angular les charge au démarrage et les affiche sous forme de liste.

4.2 Suppression d'un TUDO

L'utilisateur peut supprimer une tâche via un appel `DELETE /api/todos/{id}`.

4.3 Cocher ou décocher une tâche

Une case à cocher permet de marquer une tâche comme accomplie ou non. Côté backend, ceci utilise la méthode `PUT`.

5. Nouvelle fonctionnalité intégrée : Barre de recherche

La fonctionnalité de recherche n'était pas présente dans la version d'origine du projet. Elle a été ajoutée afin de permettre à l'utilisateur de filtrer les tâches en fonction d'un mot-clé. Cette amélioration optimise l'expérience utilisateur et facilite la navigation dans la liste des tâches.

Ce que j'ai ajouter :

```
import { Component, signal, effect } from '@angular/core';

import { CommonModule } from '@angular/common';
import { MatIconModule } from '@angular/material/icon';
import { TodoService } from './services/todo.service';
import { MatButtonModule } from '@angular/material/button';
```

```ts
import { Todo } from './models/todo.model';

@Component({
  selector: 'app-root',
  imports: [CommonModule, MatIconModule, MatButtonModule],
  templateUrl: './app.html',
  styleUrl: './app.scss'
})
export class App {
  todos = signal<Todo[]>([]);
  isLoading = signal(true);
  isDark = signal(window.matchMedia('(prefers-color-scheme: dark)').matches);

  …
  searchText = '';

  onSearch(text: string) {
    this.searchText = text;
  }

  get filteredTodos() {
    return this.todos().filter((t: Todo) =>
      t.title.toLowerCase().includes(this.searchText.toLowerCase())
    );
  }
}
```

Dans app.html :

```html
  <section class="search">
    <input type="text"
           placeholder="Rechercher..."
           (input)="onSearch($event.target.value)">
  </section>

  <section class="add">
    <input type="text"
           #todoInput placeholder="Nouvelle tâche..."
           (keyup.enter)="addTodo(todoInput.value); todoInput.value=''">
    <button (click)="addTodo(todoInput.value); todoInput.value=''">+</button>
  </section>

  <ul>
    <li *ngFor="let todo of filteredTodos">
      {{ todo.title }}
    </li>
  </ul>
```

7. Conclusion générale

L’intégration de la barre de recherche constitue une amélioration importante du projet TUDO. Elle renforce l’ergonomie de l’application et offre une fonctionnalité avancée d’exploration des données.

Ce travail a permis de :

- identifier un besoin fonctionnel réel,
- concevoir une solution simple et performante en Angular,
- optimiser l’expérience utilisateur,
- enrichir les fonctionnalités existantes.