Rapport du projet todo_fullstack

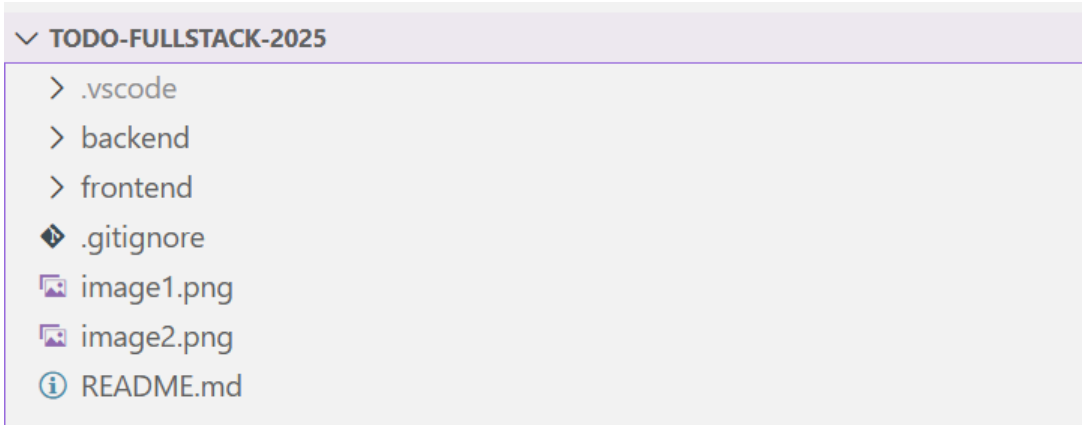
1. Introduction

Le projet todo_fullstack illustre la conception d'une application de gestion de tâches reposant sur une architecture fullstack. Il associe un frontend (interface utilisateur) et un backend (logique serveur et gestion des données), mettant en évidence la complémentarité entre la couche visible par l'utilisateur et la couche technique qui assure la cohérence des informations.

2. Structure du projet

L'organisation du projet est pensée pour séparer clairement les responsabilités entre les deux couches principales :

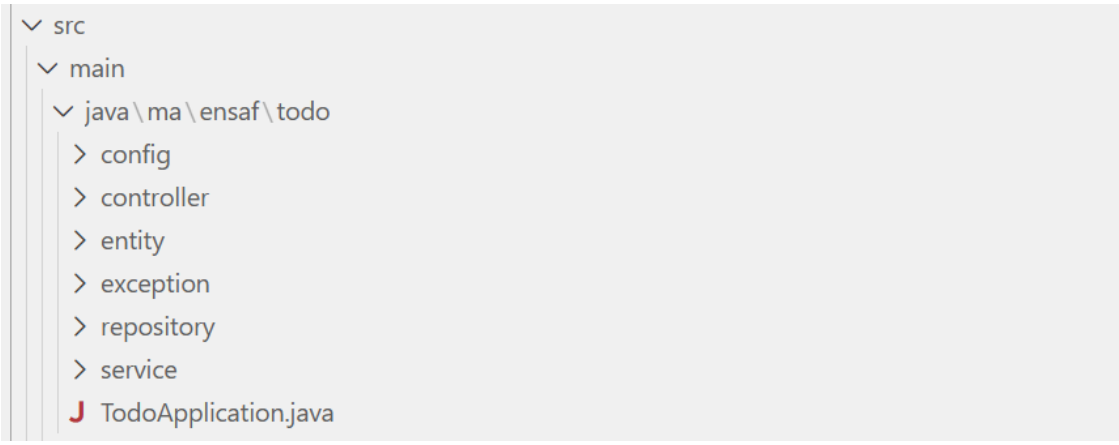
- **frontend/** contient l'application Angular
- **backend/** contient l'application Spring Boot
- **README.md** documentation initiale du projet.
- **image1.png / image2.png** captures d'écran illustrant le rendu visuel.
- **.gitignore** configuration pour ignorer certains fichiers dans Git.



Backend (Spring Boot)

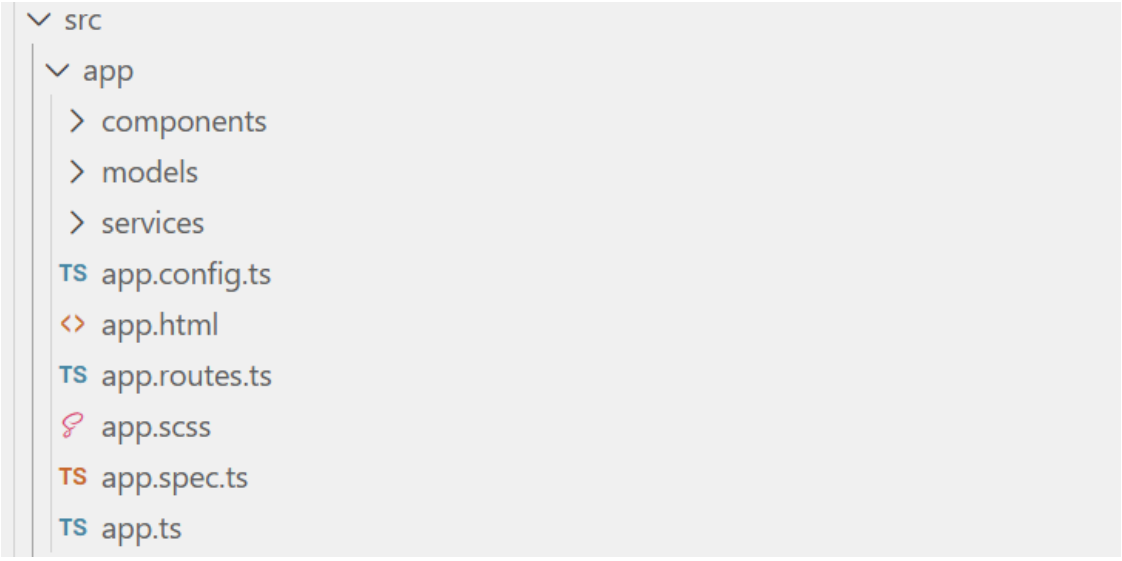
- **entity/** définit les entités (tables de la base de données).
- **repository/** gère l'accès aux données via JPA.

- **service/** contient la logique métier.
- **controller/** expose les endpoints REST (GET, POST, PUT, DELETE).
- **exception/** gère les erreurs structurées de l'application.



Frontend (Angular)

- **components/** regroupe les blocs fonctionnels de l'interface
- **models/** définit la structure des données
- **services/** contient la logique métier côté client et assure la communication avec le backend



3. Frontend : Interface Utilisateur (Angular)

Le frontend correspond à la partie de l'application accessible directement à l'utilisateur.

Technologies mobilisées : HTML pour la structure, CSS pour la mise en forme, et TypeScript pour la logique et l'interactivité.

Fonctions principales :

- Afficher la liste des tâches existantes.
- Permettre l'ajout, la modification et la suppression de tâches.
- Communiquer avec le backend afin de récupérer ou mettre à jour les données.

4. Backend : Logique Serveur (Spring Boot)

Le backend constitue la partie invisible de l'application, exécutée côté serveur.

Fonctions principales :

- Assurer la gestion de la base de données (opérations CRUD : Create, Read, Update, Delete).
- Fournir une API REST permettant au frontend d'interagir avec le serveur.
- Garantir la sécurité, la validation des données et la gestion des sessions.

5. Interaction Frontend - Backend

La communication entre les deux couches s'effectue via des API REST. Le frontend envoie des requêtes HTTP (GET, POST, PUT, DELETE), et le backend traite ces requêtes, interagit avec la base de données, puis renvoie une réponse adaptée.

Par exemple, lorsqu'un utilisateur ajoute une tâche, le frontend transmet une requête POST au backend, qui enregistre la tâche et renvoie une confirmation.

6. Conclusion

Le projet illustre de manière claire l'intégration harmonieuse entre un backend développé avec Spring Boot et un frontend conçu avec Angular. Le backend assure la gestion des données ainsi que l'exposition d'une API REST, tandis que le frontend propose une interface utilisateur moderne, dynamique et réactive. La communication entre les deux couches, réalisée via le

protocole HTTP/JSON, garantit la cohérence et la fiabilité des informations échangées. Par ailleurs, l'utilisation de modèles, de services et de tests unitaires confère à l'application une structure robuste, maintenable et évolutive, offrant ainsi une expérience complète et représentative du développement fullstack.