Rapport de veille technologique Backend

Ce rapport présente une veille technologique réalisée dans le cadre du développement d'une API Kanban.

Trois frameworks backend ont été étudiés :

- **NestJS** : NestJS est un framework **Node.js** écrit en **TypeScript**, inspiré d'Angular pour sa structure modulaire et orientée décorateurs. Son principal atout est la **rapidité de prototypage** : la configuration est minimale, la CLI propose des générateurs et l'injection de dépendances est native. NestJS intègre facilement des solutions comme TypeORM, Prisma ou GraphQL, ce qui en fait un choix moderne pour des APIs REST ou microservices. L'écosystème JavaScript/TypeScript le rend accessible à beaucoup de développeurs web.

  Par contre, son adoption reste plus faible que celle de frameworks industriels comme Spring. Certaines bibliothèques peuvent manquer de maturité, et la performance brute est limitée par Node.js en environnement très concurrentiel. NestJS est donc adapté aux start-ups et aux projets nécessitant de l'agilité, mais peut montrer ses limites dans des environnements très exigeants.

- **Symfony** : Symfony est un framework français **PHP** et open source, très **mature et stable**. Il est reconnu pour sa **robustesse**, son **écosystème riche** (Doctrine ORM, Twig, sécurité, API Platform), et une forte communauté francophone. Il s'intègre très bien dans des architectures existantes et bénéficie d'une documentation complète. Sa structure en bundles favorise la réutilisation et la maintenabilité.

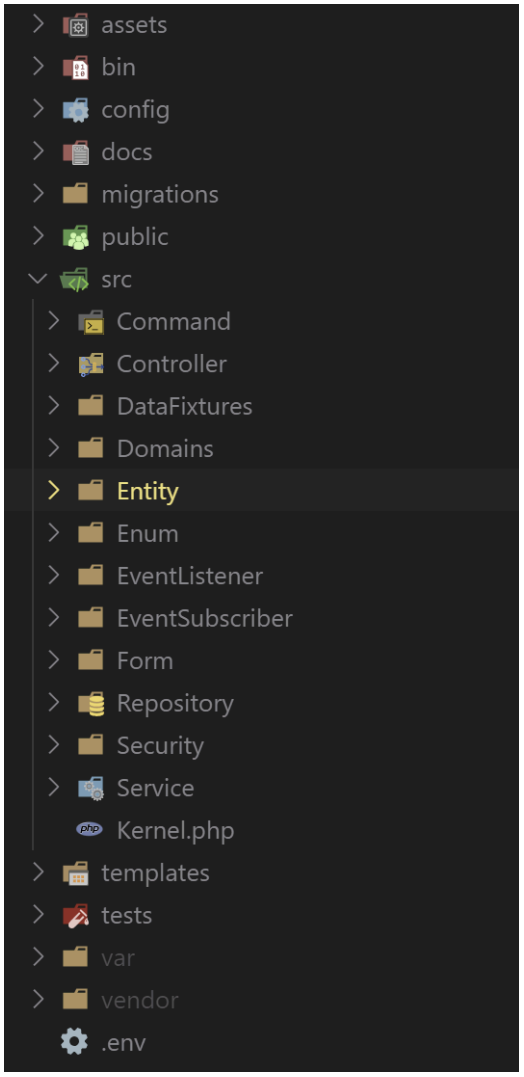
  Ses limites se trouvent dans la **courbe d'apprentissage**, plus raide que celle de *Laravel* par exemple, et des performances moindres face à des frameworks plus modernes. En revanche, sa maturité en fait une valeur sûre dans des contextes d'entreprise, notamment pour des applications métier nécessitant une forte stabilité.

- **Spring Boot** : Spring Boot est une extension du framework **Java Spring** qui supprime la lourdeur de configuration traditionnelle grâce à une approche « convention over configuration ». Il fournit un socle très complet : sécurité (Spring Security), gestion des données (Spring Data JPA), tests unitaires et intégration facile de Swagger. Spring Boot est pensé pour les **projets industriels** : robuste, très performant en production et adapté aux architectures microservices.

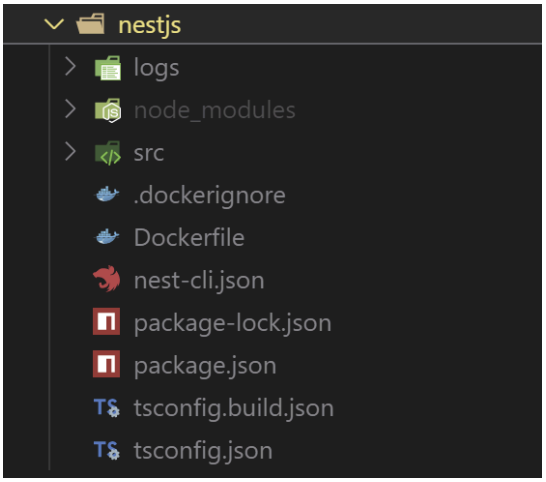
  Ses points faibles : une **complexité plus élevée**, une consommation mémoire plus importante et une courbe d'apprentissage **raide** pour un débutant en Java.

Voici à quoi ressemblent les 3 différentes structure sur un projet :

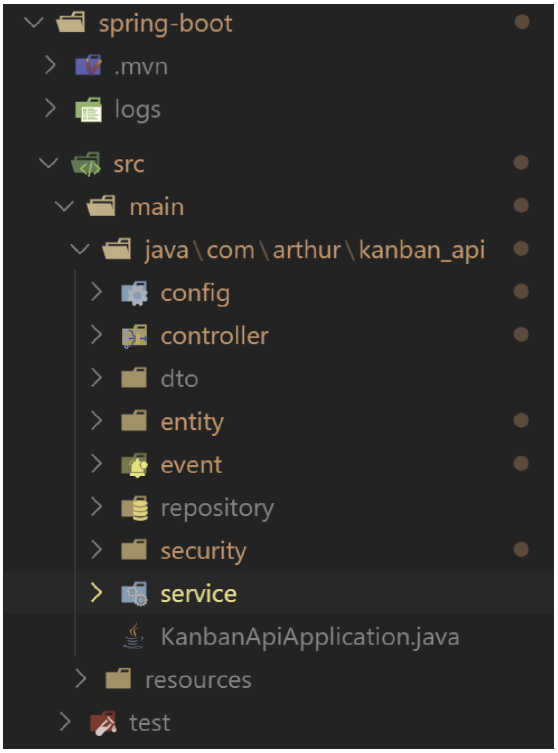
Symfony :



NestJS :



Spring Boot :



Justification du choix

Pour ce projet, j'ai préféré choisir **Spring Boot**. En tant que technologie robuste et largement utilisée dans l'industrie, cela offrait une courbe d'apprentissage pertinente et professionnalisante.

Ayant déjà travaillé avec des frameworks PHP (comme Symfony et Laravel) et Node.js (comme NextJS ou Hono), il me paraissait aussi intéressant de toucher à **Java**, car cela s'écarte nettement de ce que je connaissais déjà et élargit mes compétences.

De plus, Spring Boot fournit un socle complet (sécurité, JPA, tests, documentation Swagger) qui permet de développer une API Kanban claire et extensible.

Un autre facteur : le marché local montre une forte demande pour les développeurs Java. À Marseille, il existe de nombreuses offres Java / Spring –, ainsi que dans la région Aix-Marseille. En Suisse romande, on compte aussi plusieurs centaines d'emplois ouverts pour des profils Java.

Mon choix s'aligne avec l'objectif pédagogique d'approfondir les bases du backend tout en travaillant avec un framework moderne et reconnu, tout en visant des technos à forte employabilité.

Conclusion

Spring Boot apparaît comme la solution la plus adaptée pour un projet pédagogique visant la robustesse, la sécurité et la maintenabilité.
Ce choix permet d'allier apprentissage et bonnes pratiques industrielles.