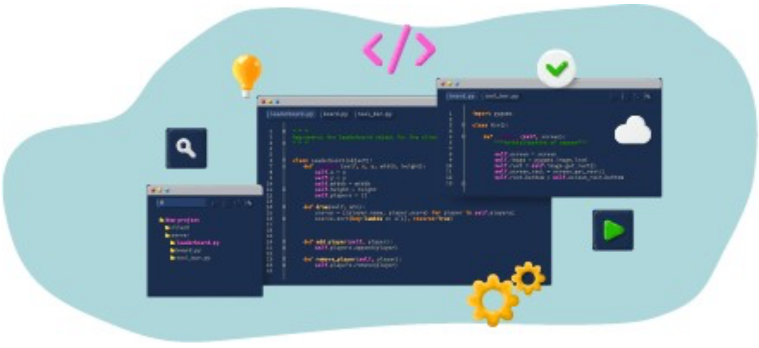
ORION



Justification des choix techniques
Projet MDD

Auteur : Aboubacar DIALLO
Version 0.0.1

Aperçu / Synthèse

Pour répondre aux contraintes techniques du projet, j'ai pris plusieurs décisions concernant les librairies, frameworks, design patterns et outils utilisés.

Côté backend, j'ai choisi Java comme langage principal, car il est reconnu pour sa stabilité et ses performances. J'ai utilisé le framework Spring pour faciliter le développement et intégrer des fonctionnalités telles que la sécurité avec Spring Security et l'utilisation de tokens JWT. Ces choix garantissent une authentification sécurisée et une gestion de l'autorisation efficace.

Pour la base de données, j'ai opté pour MySQL, une solution fiable et bien supportée, qui s'intègre parfaitement avec Spring.

Concernant le frontend, j'ai choisi Angular, un framework JavaScript populaire, pour développer une interface utilisateur réactive et modulaire. J'ai utilisé Angular Router pour gérer la navigation entre les pages de l'application de manière fluide.

Pour l'authentification et l'autorisation, j'ai mis en place des guards, des interceptors et des services dans Angular, permettant ainsi une gestion fine des accès aux fonctionnalités de l'application. J'ai également utilisé des validators pour valider les données saisies par les utilisateurs et maintenir l'intégrité des données.

Pour améliorer les performances de l'application, j'ai utilisé le concept de lazy loading, qui permet de charger les modules de manière asynchrone, réduisant ainsi le temps de chargement initial.

En ce qui concerne le design, j'ai choisi d'utiliser la bibliothèque de styles Tailwind CSS, qui offre une grande flexibilité et facilite le développement d'une interface utilisateur moderne et attrayante.

En résumé, mes choix de technologies et d'outils ont été guidés par les contraintes techniques du projet, en garantissant la sécurité, la performance et une expérience utilisateur agréable.

Choix techniques

BACKEND

choix technique	lien vers le site / la documentation / une ressource	but du choix
Java (Spring boot)	*https://docs.oracle.com/en/java/* *https://spring.io/projects/spring-security*	Spring Boot est un framework Java qui facilite le développement rapide d'applications. Il fournit des fonctionnalités et des conventions par défaut qui réduisent la configuration et permettent de se concentrer sur le développement des fonctionnalités métier plutôt que sur les détails techniques.
Spring Security	*https://spring.io/projects/spring-security*	Garantir la sécurité de l'application en gérant l'authentification, l'autorisation et la protection contre les attaques
JWT (Json Web Token)	*https://jwt.io/*	Assurer l'authentification stateless en utilisant des jetons JSON, ce qui permet d'améliorer la sécurité et de réduire la charge sur le serveur
MySQL	*https://www.mysql.com/fr/*	Choisir une base de données relationnelle stable et bien supportée pour le stockage des données de l'application.
Lombok	*https://projectlombok.org/#*	Simplifier le développement en réduisant le code boilerplate grâce à des annotations, notamment pour la génération automatique des getters, setters et constructeurs

Le choix de Java et Spring Boot permet de bénéficier d'un langage de programmation robuste et populaire, ainsi que d'un framework léger et efficace pour le développement d'applications web. L'intégration de Spring Security offre une solution complète pour la gestion de l'authentification et de l'autorisation, assurant la sécurité des données et des accès. L'utilisation de JWT (JSON Web Tokens) permet une authentification basée sur des jetons sécurisés et évolutifs. L'intégration avec MySQL en tant que base de données relationnelle offre une gestion fiable et performante des données. Enfin, l'utilisation de Lombok facilite le développement en réduisant le code boilerplate et en améliorant la lisibilité et la maintenabilité du code.

FRONTEND

choix technique	lien vers le site / la documentation / une ressource	but du choix
Angular	*https://angular.io/*	Le choix d'Angular permet de développer des applications web riches et interactives. Angular offre une architecture solide basée sur des composants réutilisables, facilite la manipulation du DOM, et fournit des fonctionnalités avancées telles que le routage, la gestion des formulaires, et la communication avec les API.
Tailwind CSS	*https://tailwindcss.com/*	Une bibliothèque de styles pour faciliter le développement d'une interface utilisateur moderne et réactive

Angular a été choisi pour son approche basée sur les composants qui permet de développer des applications modulaires et évolutives. Tailwind CSS a été sélectionné pour son approche utility-first qui offre une grande flexibilité dans la création de styles.

Architecture Frontend

Angular Router	*https://angular.io/guide/router*	Gérer la navigation entre les pages de l'application de manière fluide et maintenir une expérience utilisateur cohérente
Guards (route guards)	*https://angular.io/api/router/CanActivate*	Contrôler l'accès aux routes de l'application en fonction des règles d'autorisation spécifiées.
Interceptors	*https://angular.io/api/common/http/HttpInterceptor*	Intercepter et manipuler les requêtes HTTP pour ajouter des en-têtes, gérer les erreurs ou effectuer des actions supplémentaires
Services	*https://angular.io/guide/architecture-services*	Organiser la logique métier l'application en séparant les responsabilités et en permettant le partage de données entre les composants
Validators	*https://angular.io/api/forms/Validators*	Valider les données saisies par les utilisateurs pour garantir leur intégrité et cohérence.
Git	*https://git-scm.com/*	En choisissant Git dans mon projet, je vise à gérer efficacement les versions de mon code source, faciliter la collaboration avec mon équipe, et suivre les modifications apportées au code. J'ai opté pour Git Flow afin d'adopter une méthodologie de gestion des branches qui me permet d'organiser le développement parallèle des fonctionnalités, correctifs et versions du logiciel.

Les guards et interceptors sont utilisés pour gérer l'authentification, l'autorisation et la manipulation des requêtes HTTP. Les services sont employés pour la gestion des données et la logique métier. Les validators permettent de valider les données des formulaires. Le lazy loading est utilisé pour optimiser les performances en chargement les modules de manière asynchrone au besoin. Ces choix combinés offrent une expérience utilisateur fluide, une maintenance simplifiée et une meilleure performance de l'application.