

ORION

Justification des choix techniques **Projet**

MDD

Auteur : BERNADOU Clara
Version 0.0.1

Aperçu / Synthèse	1
Choix techniques	2 - 7

Backend	2 - 3
---------	-------

- Framework
- Base de Données
- JPA
- Sécurité
- Mappage des Objets
- Lombok

Frontend	3 - 4
----------	-------

- Outil de Gestion de Projet
- Librairie de Composants
- Réactivité
- Stylisme

Autres	4 - 5
--------	-------

- Design Patterns

Architecture	5
--------------	---

Tableau de comparaison	6
-------------------------------	----------

Backend	6
---------	---

Frontend	6
----------	---

Aperçu / Synthèse

Le backend sera développé avec **Java** avec **Spring Boot**, qui facilite le développement de microservices et d'applications web. La gestion des données sera confiée à **MySQL**, une base de données relationnelle largement utilisée. **Lombok** simplifiera le code en générant automatiquement les méthodes courantes. **OAuth2** et **Spring Security** seront utilisés pour sécuriser l'application. **ModelMapper** aide à mapper les objets DTO et entités.

Le frontend sera développé en **Angular** avec **Angular CLI** sera utilisé pour gérer le projet Angular, avec **Angular Material** pour des composants UI modernes et réactifs. **Sass** et **Tailwind** offriront des outils puissants pour le stylisme CSS. **RxJS** fournira des outils pour la programmation réactive.

Des design patterns comme le **Service Layer Pattern**, le **Repository Pattern**, et le **Component Pattern** organiseront le code de manière modulaire et maintenable. Les **Reactive Forms** et **Dependency Injection (DI)** amélioreront la gestion des formulaires et l'injection de dépendances respectivement. Le **Observable Pattern** facilitera la gestion des flux de données.

Enfin, l'application suivra une **architecture client-server**, où le client (frontend) envoie des requêtes au serveur (backend), assurant une séparation claire des responsabilités.

Choix techniques

Backend:

Choix de Framework

choix technique	lien vers le site / la documentation / une ressource	but du choix
Spring Boot	https://spring.io/projects/spring-boot	Faciliter le développement d'applications web.

Spring Boot est choisi pour sa facilité d'utilisation, sa capacité à gérer des applications complexes avec une configuration minimale et ses nombreuses fonctionnalités intégrées. Spring Boot était déjà intégré dans le code de Heidi.

Choix de base de données

choix technique	lien vers le site / la documentation / une ressource	but du choix
MySQL	https://dev.mysql.com/downloads/connector/j/	Permet la communication entre l'application et la base de données MySQL.

MySQL est choisi pour sa robustesse, sa performance et sa popularité dans la communauté des développeurs. Il offre une gestion efficace des transactions et une intégration facile avec Spring Boot. MySQL était déjà intégré dans le code de Heidi.

Choix de JPA

choix technique	lien vers le site / la documentation / une ressource	but du choix
Spring Data JPA	https://spring.io/projects/spring-data-jpa	Simplification de l'accès à la base de données en utilisant le modèle de persistance Java (JPA).

Spring Data JPA est choisi pour sa capacité à simplifier l'accès aux bases de données en fournissant une interface de haut niveau pour les opérations de persistance. Il est bien intégré avec Spring Boot, ce qui permet une configuration facile et une gestion fluide des entités JPA.

Choix de sécurité

choix technique	lien vers le site / la documentation / une ressource	but du choix
Spring Security OAuth2 Resource Server	https://docs.spring.io/spring-security/site/docs/current/reference/html5/#oauth2	Sécurisation de l'application via une gestion des accès et des autorisations.

Spring Security OAuth2 est choisi pour sa capacité à gérer de manière sécurisée les autorisations et les authentifications. Il est bien intégré avec Spring Boot et offre une solution complète pour la sécurité de l'application.

Choix mappage des objets

choix technique	lien vers le site / la documentation / une ressource	but du choix
ModelMapper	http://modelmapper.org/	Mapper facilement les objets DTO et entités pour simplifier la gestion des données.

ModelMapper est choisi pour sa simplicité et son efficacité à transformer des objets de manière fluide et automatique. Cela améliore la maintenabilité du code et réduit la complexité des conversions.

Choix Lombok

choix technique	lien vers le site / la documentation / une ressource	but du choix
Lombok	https://projectlombok.org/	Réduire le code boilerplate () en générant automatiquement des getters, setters, et autres méthodes courantes.

Lombok est choisi pour sa capacité à simplifier le code Java en éliminant le besoin d'écrire manuellement des méthodes répétitives. Cela améliore la lisibilité et la maintenabilité du code.

Frontend:

Choix d'outil de gestion de projet

choix technique	lien vers le site / la documentation / une ressource	but du choix
Angular CLI	https://www.npmjs.com/package/@angular/cli	Simplifier la gestion des projets Angular, depuis la création jusqu'à la compilation.

Angular CLI est choisi pour sa capacité à automatiser et simplifier le workflow de développement Angular, permettant de gagner du temps et d'assurer une structure de projet cohérente. Angular CLI était déjà intégré dans le code de Heidi.

Choix librairie de composants

choix technique	lien vers le site / la documentation / une ressource	but du choix
Angular Material	https://material.angular.io/guides	Fournir des composants UI modernes et réactifs conformes aux directives Material Design.

Angular Material est choisi pour ses composants UI prêts à l'emploi, qui permettent de construire rapidement des interfaces utilisateur fonctionnelles. Angular Material était déjà intégré dans le code de Heidi.

Choix de réactivité

choix technique	lien vers le site / la documentation / une ressource	but du choix
rxjs	https://rxjs.dev/	Fournir des outils pour la programmation réactive et la gestion des flux de données asynchrones.

RxJS est choisi pour sa puissance et sa flexibilité dans la manipulation des données asynchrones, essentielle pour la réactivité et l'interactivité de l'application. RxJS était déjà intégré dans le code de Heidi.

Choix style de l'application

choix technique	lien vers le site / la documentation / une ressource	but du choix
Tailwindcss	https://tailwindcss.com/	Proposer une approche utilitaire pour le stylisme CSS, permettant une conception rapide et efficace.
Sass	https://sass-lang.com/	Faciliter le stylisme CSS avec des fonctionnalités avancées comme les variables, les mixins, et les fonctions.

Tailwind CSS est choisi pour sa flexibilité et sa capacité à créer des interfaces utilisateur rapidement sans avoir à écrire du CSS personnalisé.

Sass est choisi pour ses fonctionnalités puissantes qui augmentent la productivité et la maintenabilité du code CSS. Sass était déjà intégré dans le code de Heidi.

Autres:

Choix des Design Patterns

choix technique	lien vers le site / la documentation / une ressource	but du choix
Service Layer Pattern	https://en.wikipedia.org/wiki/Service_layer_pattern	Organiser la logique métier de manière modulaire et maintenable.
Repository Pattern	https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30	Encapsuler la logique d'accès aux données et fournir une API propre pour les opérations CRUD.
Component Pattern	https://en.wikipedia.org/wiki/Composite_pattern	Modulariser le code frontend en composants réutilisables et indépendants.
Reactive Forms Pattern	https://angularindepth.com/posts/1428/angular-forms-reactive-design-patterns	Gérer les formulaires de manière réactive, facilitant la validation et la manipulation des données.
Dependency Injection (DI) Pattern	https://en.wikipedia.org/wiki/Dependency_injection	Faciliter l'injection de dépendances pour améliorer la testabilité et la modularité du code.
Observable Pattern	https://fr.wikipedia.org/wiki/Observateur_(patron_de_conception)	Gérer les flux de données asynchrones, essentiel pour la programmation réactive.

Le **Service Layer Pattern** est choisi pour sa capacité à structurer la logique métier de manière claire et modulaire, facilitant la maintenance et l'évolution de l'application.

Le **Repository Pattern** est choisi pour sa capacité à abstraire la couche d'accès aux données, rendant le code plus propre, testable et maintenable.

Le **Component Pattern** est choisi pour structurer le code Angular de manière modulaire, facilitant la réutilisation et la maintenance des composants UI.

Les **Reactive Forms** sont choisis pour leur capacité à gérer les formulaires complexes et les validations dynamiques de manière réactive, améliorant l'expérience utilisateur.

Le **DI pattern** est choisi pour structurer le code de manière à rendre les dépendances explicites, facilitant ainsi les tests et la maintenance.

L'**Observable Pattern** est choisi pour gérer les événements et les flux de données asynchrones de manière efficace, améliorant la réactivité de l'application.

Choix de l'architecture

choix technique	lien vers le site / la documentation / une ressource	but du choix
Architecture Client-Server	https://en.wikipedia.org/wiki/Client%E2%80%93server_model	Assurer une séparation claire des responsabilités

L'architecture **client-server** permet de séparer les préoccupations de l'interface utilisateur (client) et de la logique métier ainsi que de la gestion des données (serveur). Cela rend le code plus modulaire et plus facile à gérer.

Exemple dans l'appli: le client (**Angular**) est responsable de la présentation et de l'interaction utilisateur, tandis que le serveur (**Spring Boot**) gère les opérations de traitement de données et les règles métier.

Comparaisons

Backend:

Critères	Spring Boot	Microstream	Quarkus	WingsDB	MySQL	PostgreSQL	Microsoft	Oracle	Apache Kudu	Avatica	Debezium	Apache Flink	Phoenix	Spring Security	Apache Thrift	Alibaba	Microsoft	Microsoft
Simple d'utilisation	Très simple	Relativement simple	Très simple	Relativement simple	Très simple	Relativement simple	Relativement simple	Très simple	Modérément simple	Modérément simple	Modérément simple	Modérément simple	Modérément simple	Modérément simple	Modérément simple	Modérément simple	Modérément simple	Modérément simple
Rapide d'exécution	Moyen	Très rapide	Rapide	Rapide pour NoSQL	Rapide	Très rapide	Très rapide	Très rapide	Rapide	Rapide	Rapide	Rapide	Rapide	Rapide	Rapide	Rapide	Rapide	Très rapide
Librairie/Framework connu	Très largement utilisé	De plus en plus connus	Moins connus	Très largement utilisé	Très largement utilisé	Très largement utilisé	De plus en plus utilisé	Très largement utilisé	Moins connus	Moins connus	Très largement utilisé	Moins connus	Moins connus	Très largement utilisé	Relativement connu	Moins connus	Très largement utilisé	De plus en plus utilisé
Débug régulièrement / évolutif	Très actif	Très actif	Actif	Très actif	Très actif	Très actif	Très actif	Très actif	Actif	Actif	Très actif	Actif	Actif	Très actif	Actif	Actif	Actif	Très actif
Beaucoup de questions sur StackOverflow	Très nombreuses	Moins nombreuses	Moins nombreuses	Très nombreuses	Très nombreuses	Très nombreuses	Modérées	Nombreuses	Pas nombreuses	Pas nombreuses	Très nombreuses	Modérées	Très nombreuses	Modérées	Pas nombreuses	Nombreuses	Modérées	Modérées
Modèle de Spring	☒	☐	☐	☐	☐	☐	☐	☐	☐	☒	☐	☐	☐	☒	☐	☐	☐	☐
Techno choisie par Heidi	☒	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

https://docs.google.com/spreadsheets/d/187revjb_WzJLQ_OpZpiWrz1oIMO8vBcMQjIMb-naPBQ/edit?usp=s_haring

Frontend:

Critères	Vite	Angular CLI	Nx (Nécessite Extensions)	PrimeNG	NG-ZORRO	Angular Material	Bootstrap	Sass	Tailwind	ReactJS	NgRx	Redux
Simple d'utilisation	Très simple	Simple	Relativement simple	Relativement simple	Modérément simple	Relativement simple	Très simple	Modérément simple	Relativement simple	Modérément simple	Relativement complexe	Complexe
Rapide d'exécution	Très rapide	Modéré	Modéré	Rapide	Rapide	Modéré	Rapide	Rapide	Très rapide	Rapide	Rapide	Rapide
Librairie/Framework connu	De plus en plus connu	Très largement utilisé	De plus en plus connu	Bien connu	Bien connu	Très connu	Très largement utilisé	Très largement utilisé	De plus en plus connu	Très largement utilisé	De plus en plus utilisé	Très largement utilisé
Débug régulièrement / évolutif	Actif	Très actif	Actif	Actif	Actif	Très actif	Très actif	Très actif	Actif	Très actif	Actif	Très actif
Beaucoup de questions sur StackOverflow	Moins nombreuses	Très nombreuses	Moins nombreuses	Nombreuses	Nombreuses	Très nombreuses	Très nombreuses	Très nombreuses	Modérées	Très nombreuses	Nombreuses	Très nombreuses
Techno choisie par Heidi	☐	☒	☐	☐	☐	☒	☐	☒	☐	☒	☐	☐

https://docs.google.com/spreadsheets/d/10Jxovs9PgnDOHner-XcFlyN9DaDZ09U8LzBnPTz8P-8/edit?usp=s_haring