Dossier d’architecture technique

OCDM Base Stack

Version 0.1

# Historique du document

## Révisions du document

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Version applicative concernée | Description | Auteur |
| 0.1 | 17/02/2023 |  | - Version initiale | TGE |
|  |  |  |  |  |

## Validation du document

|  |  |  |
| --- | --- | --- |
| Validation du commanditaire | <Organisation>  <Nom / Prénom>  <Fonction> | Date et visa : |
| Validation du prestataire | <Organisation>  <Nom / Prénom>  <Fonction> | Date et visa : |
| Approbateurs : | <Nom / Prénom>, <Organisation>  <Nom / Prénom>, <Organisation> |  |

Table des matières

[Historique du document 2](#_Toc129707741)

[Révisions du document 2](#_Toc129707742)

[Validation du document 2](#_Toc129707743)

[1. Contexte 4](#_Toc129707744)

[1.1. Objet du document 4](#_Toc129707745)

[1.2. Enjeux projet 4](#_Toc129707746)

[1.3. Utilisateurs 4](#_Toc129707747)

[2. Fonctionnalités 5](#_Toc129707748)

[1.4. Exigences techniques 5](#_Toc129707749)

[3. Architecture logicielle 6](#_Toc129707750)

[1.5. Architecture générale 6](#_Toc129707751)

[1.6. Backend Spring MVC 7](#_Toc129707752)

[4. Architecture technique et solutions logicielles 8](#_Toc129707753)

[1.7. Infrastructure 8](#_Toc129707754)

[1.7.1. Serveur d’application Java 9](#_Toc129707755)

[1.7.2. SGBD 9](#_Toc129707756)

[1.8. Versions logicielles 9](#_Toc129707757)

[5. Sécurité 11](#_Toc129707758)

# Contexte

## Objet du document

Le présent document est le Dossier d’Architecture Technique du projet Base Stack. Il vise à préciser les spécifications d’architectures fonctionnelle et technique de cette application.

## Enjeux projet

Ce projet a pour objectif de fournir une base de démarrage uniforme et prête à l’emploi pour l’implémentation des projets.

L’objectif du déploiement de cette base commune pour nos futurs projets est multiple

* Gains de temps au démarrage des projets
* Industrialisation des projets facilitée (build, packaging, CI…)
* Amélioration du support de la DT et des autres projets
* Facilité pour intervenir sur différents projets
* Amélioration de la capacité à mettre à jour le socle

Ce socle projet n’implémente pas de logique métier, il propose un socle technique fonctionnel répondant aux problématiques transverses commune des projets

## Utilisateurs

Les utilisateurs de ce logiciel sont les équipes projet qui implémentent cette base pour la décliner comme un projet à destination des utilisateurs finaux.

# Fonctionnalités

Comme évoqué en introduction, ce socle projet n’implémente pas de logique métier.

Il se concentre les problématiques transverses fonctionnelles et techniques suivantes :

* Disposer d’un backend Java permettant d’implémenter les fonctions métiers
* Disposer d’un frontend web permettant aux utilisateurs d’interagir avec l’application via un navigateur
* Stocker les données de l’application dans une base de données dédiée
* Gérer la confidentialité des données et l’accès aux fonctions via une identification utilisateur associée à la notion de rôle
* Proposer une base commune pour l’ensemble des projets éligibles
* Permettre une exécution locale du projet pour les environnements de développement

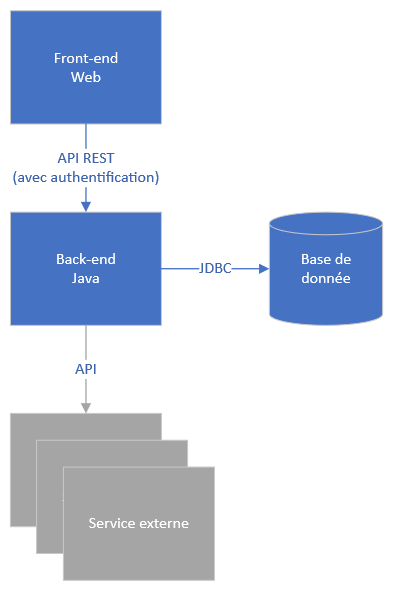
## Exigences techniques

Les fonctionnalités présentées précédemment s’inscrivent dans un cadre d’exigences techniques

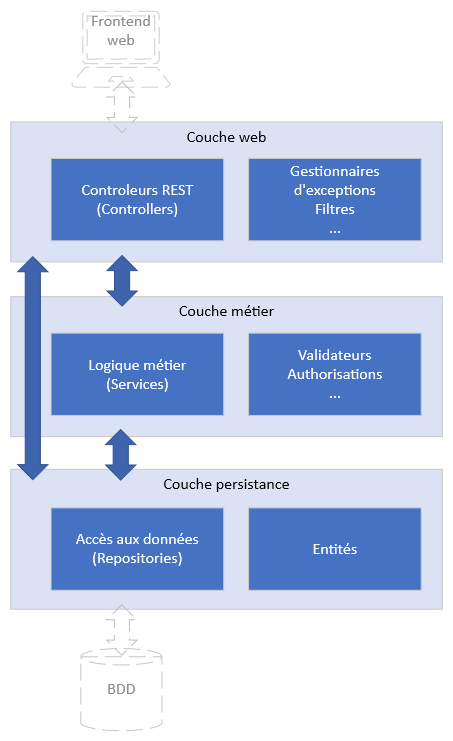
* Rapidité et simplicité de mise en œuvre
* Adaptabilité de la solution
* Respect des normes et de l’état de l’art
* Industrialisation
* Limitation de l’obsolescence au démarrage et pendant le cycle de vie

# Architecture logicielle

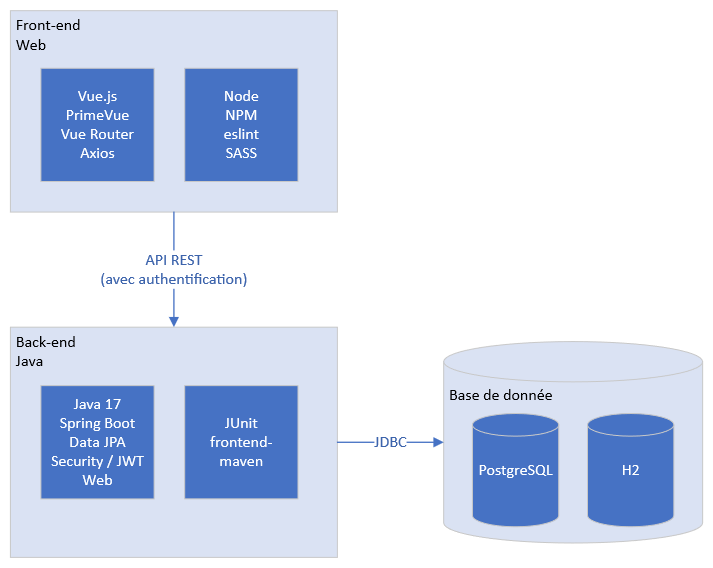
## Architecture générale



## Backend Spring MVC



# Architecture technique et solutions logicielles



## Infrastructure

Le socle est configuré par défaut pour être servi par 2 composants :

* Un serveur d’application Java pour servir le backend et le frontend
* Un SGBD pour la base de données

Les 2 composants peuvent être déployés sur le même serveur ou séparément en fonction des besoins et de la charge attendue sur la plateforme cible.

### Serveur d’application Java

Le build de l’application produit 2 livrables

* Un fichier .war auto exécutable intégrant une version d’apache Tomcat embedded dans lequel l’application est déjà déployée.

C’est le comportement par défaut de Spring Boot, ce mode de fonctionnement réduit les besoins de gestion des serveurs et permet de livrer des mises à jour du serveur d’application en même temps que les mises à jour de l’application elle-même

* Un fichier .war.original qui est un war « classique » pouvant être déployé dans un serveur d’application Java installé sur le serveur

#### Frontend

Le frontend (partie web) de l’application est packagée avec le backend et servie directement par le serveur d’application Java. En cas de besoin, il est envisageable de packager, livrer et servir le frontend séparément, sur un serveur web type Apache httpd ou NGINX, éventuellement hébergé sur une machine différente de celle qui héberge le serveur d’application du backend.

La configuration CORS (autorisation pour le frontend d’appeler les API du backend sur une autre url que celle du frontend) est déjà réalisée pour les besoins de l’exécution via NPM avec hot reload pour le développement.

### SGBD

Le SGBD proposé par défaut dans le socle est PostgreSQL

Aucun développement spécifique à ce SGBD n’est porté par le socle de base, il pourra donc être remplacé au besoin (MySQL, MariaDB…).

/!\ L’environnement Docker devra également être mis à jour.

## Versions logicielles

|  |  |  |
| --- | --- | --- |
| **Solution** | **Version** | **Description** |
| Java | 17 | Langage de programmation et environnement d’exécution de l’application |
| Spring | 5.3.23 | Framework d’application Java |
| Spring Boot | 2.7.4 | Extension de Spring pour la création et la configuration de projets Spring |
| Apache Tomcat | (embedded) 9.0.65 | Serveur d’application Java, ici en version embedded via Spring boot |
| Spring Security | 5.7.3 | Module de sécurité (authentification et autorisation) de Spring |
| Spring Web MVC | 5.3.23 | Module Spring permettant de faire des application web avec Spring via le pattern MVC, fourni notamment la couche REST |
| Spring Data JPA | 2.7.3 | Module de persistance, permet notamment de faire le mapping objet relationnel et la couche d’accès aux données |
| Hibernate | 5.6.11 | Framework de persistance et d’ORM utilisé par JPA |
| Jjwt | 0.9.1 | Librairie de manipulation de JWT pour la JVM |
|  |  |  |
| PostgreSQL | (docker) 15.1 | SGBD relationnel et objet |
| H2 | 2.1.214 | SGBD relationnel pouvant être embarqué, utilisé ici avec des données en mémoire pour faciliter le démarrage d’un environnement de dev |
| HikariCP | 4.0.3 | Pool de connexion JDBC |
|  |  |  |
| Node | 18.14.0 | Environnement d’exécution JavaScript |
| NPM | 9.3.1 | Gestionnaire de paquet JavaScript de Node |
| Vue.js | 3.2.13 | Framework JavaScript |
| vue-router | 4.0.3 | Module de routage d’url pour les applications SPA |
| vuex | 4.0.0 | Module de gestion d’état des données |
| primevue | 3.13.1 | Librairie de composant IHM pour Vue.js |
| sass | 1.32.7 | Langage de script préprocesseur pour CSS, permet de définir ses feuilles de style dans un langage offrant plus de fonctionnalité que CSS |
| eslint | 7.32.0 | Outil d’analyse de la qualité et du style du code statique |

# Sécurité

La sécurité de base de l’application est gérée par un mécanisme de comptes utilisateurs associés à des rôles.

Chaque fonction métier, chaque opération d’API et chaque composant IHM est en mesure de vérifier si l’utilisateur courant est identifié et s’il dispose du rôle requis pour la fonction désirée.

|  |
| --- |
| /!\ L’IHM web étant composée de scripts JS statiques interprétés par le navigateur de l’utilisateur, il n’est pas possible de la sécuriser de manière fiable.  Par exemple, n’importe quel utilisateur est théoriquement en mesure d’afficher des boutons ou écrans auxquels il ne devrait pas avoir accès dans l’IHM en modifiant le code JS dans la console de développement du navigateur.  **Un contrôle des droits doit TOUJOURS être réalisé côté serveur.** |

Ces mesures de sécurités sont implémentées via des jetons JWT à l’aide de Spring Security.

En cas de besoin, des mécanismes complémentaires pourront être ajoutés

* Authentification par annuaire
* Authentification SSO
* Sessions HTTP au lieu de JWT
* Firewall / Antivirus
* Gateway proxy

**Il est recommandé d’exposer le frontend et les API via un reverse proxy ajoutant une couche TLS (url HTTPS) pour que les données ne circulent pas en clair dans des paquets HTTP**