

- OpenProject

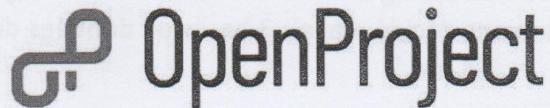


Figure 4 : Logo de OpenProject

OpenProject est une plateforme de gestion de projet open-source que nous utilisons pour organiser et suivre notre projet. Nous pouvons nous organiser grâce au diagramme de Gantt intégré, créer les différentes phases du projet, les fonctionnalités à réaliser ainsi que les tâches des fonctionnalités. Les tâches seront ensuite assignées aux membres, qui pourront suivre leur avancement et prévenir s'il y a des blocages.

### III. Choix techniques

Après plusieurs échanges concernant les langages de programmation, les différents frameworks et les différents outils, nous avons finalement choisi un framework de Java : **Spring Boot**. Spring Boot permet principalement de générer et de gérer des projets orientés web, en full stack et adaptés pour des projets évolutifs qui, grâce à ses dépendances, va nous permettre, entre autres, de définir une sécurité d'authentification convenable, de générer des API rest d'une application et surtout, de gérer et de fluidifier plusieurs bases de données. En effet, nous aurons besoin, au cours de ce projet de différents types de base de données :

- ~~Base de données~~ relationnelles
- ~~Orientées~~ graphes
- ~~Orientées~~ documents
- Ou encore temporelles

Spring Boot possède plusieurs composants et permet le développement rapide et structuré d'un projet, notamment grâce à son architecture *Model View Controller* (MVC - Modèle - Vue - Contrôleur). Ce Framework a été proposé par des membres du groupe qui possédaient déjà une l'expérience, quant aux membres n'ayant jamais eu l'occasion de l'utiliser, ils souhaitaient découvrir de nouveaux outils de travail. La prise en main de Spring Boot étant assez simple, personne ne sera laissé pour compte lors du développement du projet.

Concernant le "Front-end" de l'interface Web nous allons utiliser **Spring Web** qui est le module pour Spring Boot le plus utilisé concernant le développement d'une application web et d'une API. Il sera également couplé avec le module **Thymeleaf** qui est un moteur de

template permettant de traiter de l'HTML standard. De cette manière, les membres du projet pourront coder les interfaces web en HTML.

Concernant le "Back-end" l'API de Spring Boot sera utilisée afin de réaliser les GET/POST sur les bases de données. Nous avons décidé d'avoir 3 bases de données différentes dans le projet :

- MySQL (BDD relationnel) sera utilisé pour le stockage des données structurées ainsi que les fichiers de données (csv,pdf,xml,...) en utilisant des BLOB
- Neo4J (BDD Graphe) sera utilisé pour le stockage des données n'ayant pas de structure propre.
- PostgreSQL + TimescaleDB sera utilisé pour le stockage des données temporel (données collectées au fil du temps), comme les relevés de capteur ou le trafic routier.

La figure 5 doit ...

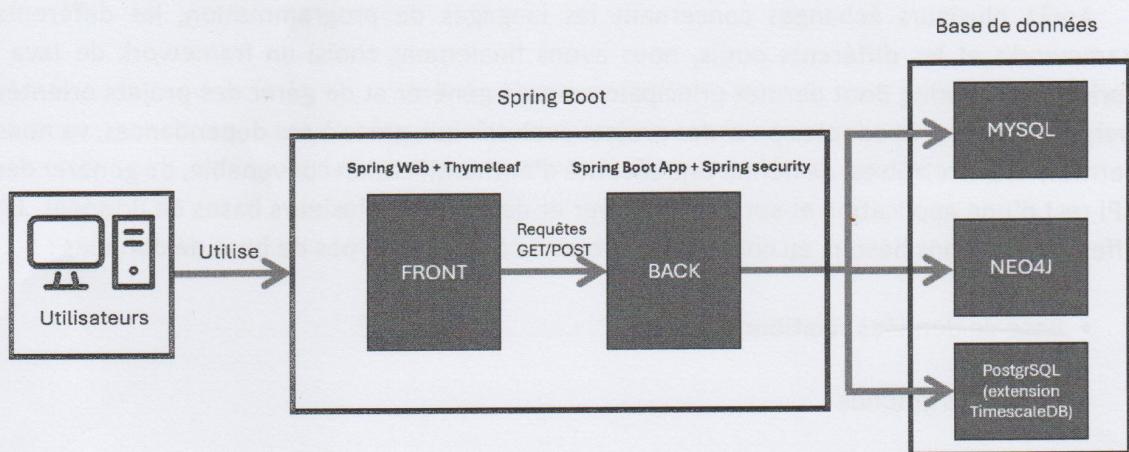


Figure 5 : Schéma des interactions entre les technologies

## IV. Fonctionnalités envisagées

*Le tableau 1 décrit*

Voici les fonctionnalités envisagées pour notre portail web d'open data pour le premier semestre, un utilisateur non-admin représente (citoyens, entreprise, DSI, etc) P. 9

*rappel final*

*offerte par*

Fonctionnalités	Description
Création de compte	L'utilisateur non connecté doit fournir des informations (adresse mail et un mot de passe) pour créer un compte sur le portail.
Connexion	L'utilisateur non connecté ou admin entre son email et son mot de passe pour se connecter.
Modifications des informations du compte (mot de passe, adresse mail)	L'utilisateur connecté ou admin peut modifier ses informations personnelles, comme son mot de passe ou son adresse e-mail depuis les paramètres de son compte.
Importer des données	L'utilisateur admin (la métropole) peut téléverser (uploader) des fichiers de données vers le portail.
Affichage des données <i>et</i>	Une fois les données importées, l'utilisateur connecté, non connecté et admin peut les consulter directement sur le portail.
Télécharger les données <i>dit</i>	L'utilisateur non connecté, connecté et admin peut télécharger les données dans différents formats (CSV, PDF, Excel, etc.). <i>oui</i>
Affichage / Choix politique de licence	L'utilisateur connecté peut sélectionner ou consulter les licences (comme Creative Commons) sous lesquelles les données sont partagées ou accessibles.
Recherche de données <i>dit</i>	L'utilisateur non connecté, connecté et admin peut rechercher des données spécifiques sur le portail en utilisant une barre de recherche en saisissant des mots clés ou en écrivant le nom de la donnée.
Filtrage des données <i>et</i>	L'utilisateur non connecté, connecté et admin peut trier ou filtrer les données en fonction de critères comme le thème, la date, ou le type de données pour faciliter la recherche et l'affichage.

Tableau 1 : Fonctionnalités du portail

Ces fonctionnalités sont conçues pour atteindre nos objectifs en facilitant l'accès aux données publiques tout en respectant les obligations légales.

*pourquoi faut-il être connecté pour consulter à licence*

*Réponse pour dire que le tableau 1 est en lien avec la fig. 8.*

Il faut rebrousser les m<sup>ême</sup> mots que dans le tableau 1)

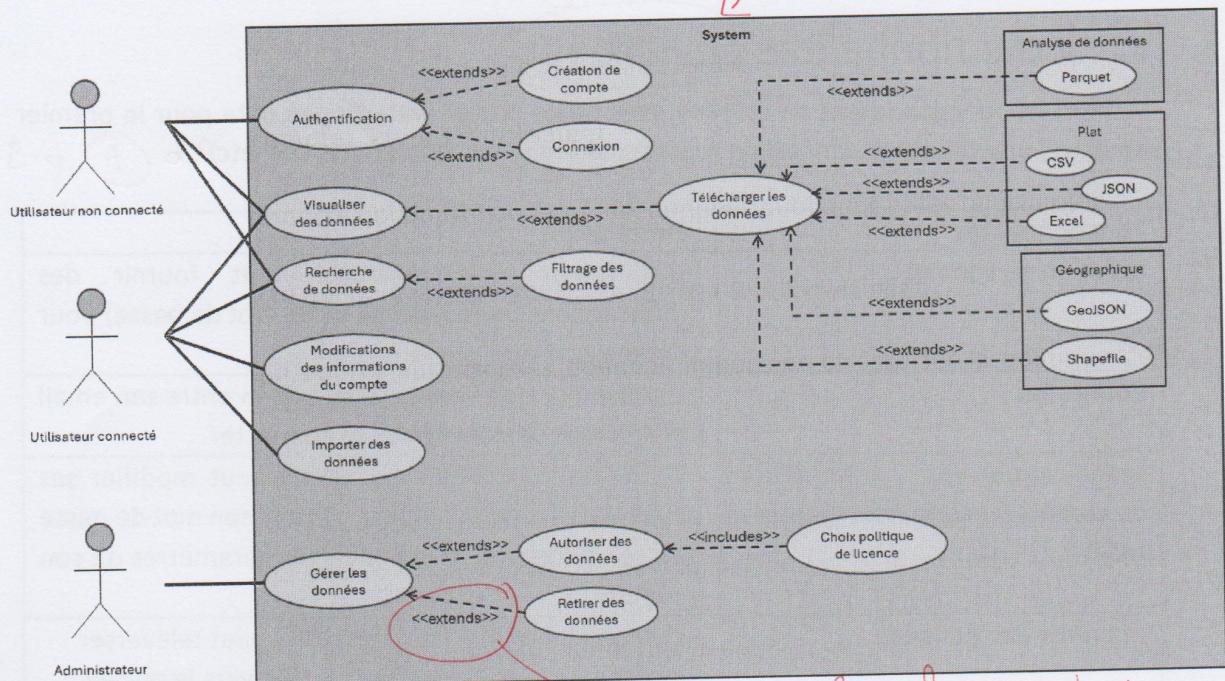


Figure 6 : Diagramme d'utilisation

Revoi la rel. << extends >>  
↑ pt d/extends.

Notre open data va compter 3 types d'utilisateur :

- L'utilisateur non connecté qui pourra accéder aux données publiques et les télécharger.
- L'utilisateur connecté qui pourra importer ses données pour les partager.
- L'administrateur qui pourra autoriser les données importées et associer la politique de licence.

A

## V. Dépendances

Pour que le développement du projet soit efficace, nous avons établi un tableau de dépendance pour l'ensemble de nos tâches. Cela nous permettra d'avancer tous ensemble de manière continue et de s'assurer que notre prévision des tâches est cohérente.

Numéro	Fonctionnalités	Dépendance	Pourquoi ?
A	Création de compte	/	La création de compte ne dépend d'aucune autre fonctionnalité
B	Connexion	A	Pour pouvoir se connecter, il faut que le système de compte soit mis en place et fonctionnel
C	Modifications des informations	A	La modification des informations sur le compte nécessite d'abord la création de compte
D	Importer des données	B	L'import de données ne peut se faire que si l'utilisateur est connecté, et qu'il s'agit d'un administrateur
E	Affichage/ Visualiser des données	/ ?	L'accès aux données ne dépend d'aucune autre fonctionnalité
F	Télécharger les données	E	Le téléchargement des données ne peut se faire que si l'utilisateur y a accès
G	Affichage / Choix politique de licence	D	Le choix de la licence des données est réalisé lors de l'importation des données
H	Filtrage des données	E	Le filtrage des données se fait en fonction des données qui peuvent être affichées à l'utilisateur
I	Recherche de données	H	La recherche des données correspond à une sorte de filtrage plus précis des données

Tableau 2 : Table des fonctionnalités et leurs dépendances

## VI. Fonctionnalités implémentées

La première étape après le démarrage de la phase de développement a été de découvrir Spring Boot ainsi que de rechercher différents styles graphiques d'open Data existant. Après plusieurs comparaisons, nous avons choisi 3 styles graphiques différents :

- Londre

Mette les URL

- Strasbourg

- Paris

Nous avons également choisi 2 thèmes (catégories) existant sur l'open data de Dijon afin de pouvoir travailler et récupérer des données concrètes. Nous nous sommes restreints avec 2 thèmes pour cette première phase du projet afin de nous concentrer sur les bases du projet.

Voir annexe 4 : Page du site

## 1. Création de compte

L'objectif principal de cette fonctionnalité est de permettre aux utilisateurs de créer un compte dans l'application. Ce compte sera stocké dans une base de données MySQL. La sécurité des données sera réalisée à l'aide de Spring Security.

### a. Descriptif fonctionnel

Un utilisateur non connecté peut créer un compte en cliquant sur le bouton "Création de compte". Une fois cela fait, il lui suffit de renseigner son nom, prénom, adresse mail et mot de passe et de cliquer sur "Créer". Le système vérifie que l'adresse email n'existe pas déjà dans la base de données, le mot de passe est ensuite encodé avant d'être enregistré pour garantir la confidentialité. Si les données sont correctes, un compte utilisateur est créé et stocké dans MySQL.

### b. Descriptif technique

Technologies utilisées :

Rediger, éviter les listes

- **Spring Boot** : Pour la structure de l'application.
- **Spring Security** : Pour gérer l'authentification et la sécurité des comptes.
- **Thymeleaf** : Pour le rendu des vues côté serveur.
- **SweetAlert2** : Pour afficher des pop-ups interactifs et des messages de confirmation ou d'erreur à l'utilisateur.
- **MySQL** : Base de données pour stocker les informations utilisateur.

Spring Boot est basé sur MCD, donc nous avons réalisé les différentes classes nécessaires :

- figue 7*
- **UserController** : contrôleur principal gérant les requêtes d'inscription, de connexion, et de déconnexion :
    - `@GetMapping("/register")` : affiche la page de création de compte
    - `@PostMapping("/register")` : gérer la création d'un compte utilisateur.
  - **UserServices** : Service entre le contrôleur et le repository, pour appliquer les traitements nécessaires avant d'enregistrer les données :

- La méthode `registerUser(User user)` fait appel au repository pour sauvegarder l'utilisateur.
- **UserRepository** : Interface de type repository qui étend JpaRepository, permettant de communiquer avec la base de données MySQL.
- **User** : Entité représentant un utilisateur dans la base de données MySQL.

#### Gestion des erreurs de création de compte :

En cas d'erreur de saisie (email déjà utilisé, mots de passe non correspondants, ou erreur de connexion à la base de données), une pop-up SweetAlert2 indique à l'utilisateur l'erreur.

Concernant l'encodage du mot de passe Spring Security est configuré pour encoder les mots de passe avec la classe **BCryptPasswordEncoder** et restreindre l'accès aux pages en fonction des droits de l'utilisateur.

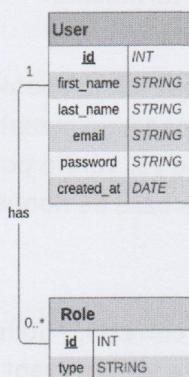


Figure 7 : MCD partie utilisateur

Voir en annexe 5 / Diagrammes des classes et séquences partie utilisateur

#### e. Organisation

La figure 8 présente cette tâche dont ...

OpenProject

#### Tous les ouverts

SUJET	STATUT	DATE DE DÉBUT	DATE DE FIN	DURÉE	ASSIGNÉ À
Création de la vue pour la page de création	Clôturé	14-10-2024	17-10-2024	4 j	Kevin Pradier
Créer la classe UserService	Clôturé	14-10-2024	14-10-2024	1 j	Anthony Michaud
Créer la classe UserController	Clôturé	14-10-2024	14-10-2024	1 j	Kevin Pradier
Créer la classe Model User	Clôturé	14-10-2024	14-10-2024	1 j	Anthony Michaud
Réalisation des tests unitaires	Clôturé	15-10-2024	16-10-2024	2 j	Anthony Michaud

Figure 8 : Tâches création de compte

- **SweetAlert2** : Pour afficher des notifications ou messages d'erreur interactifs si la connexion échoue.

### Structure et composants de la fonctionnalité

#### Contrôleur UserController :

- Le contrôleur UserController expose deux endpoints pour la connexion :
  - **@GetMapping("/login")** : Affiche la page de connexion avec le formulaire.
  - **@PostMapping("/login")** : Gère l'envoi du formulaire et vérifie les informations de connexion.

#### Service UserServices :

- La méthode **getUserByEmail(String email, String password)** vérifie l'existence de l'utilisateur et valide le mot de passe.
- La méthode utilise **PasswordEncoder.matches()** pour comparer le mot de passe saisi par l'utilisateur avec le mot de passe encodé stocké dans la base de données.

#### Gestion de la Session Utilisateur :

- La méthode **connectUser(User user, HttpServletRequest req)** dans UserController initialise une session utilisateur sécurisée.

Elle utilise **UsernamePasswordAuthenticationToken** pour créer un jeton d'authentification avec les informations de l'utilisateur. **AuthenticationManager** utilise et authentifie le jeton et **SecurityContextHolder** sauvegarde l'authentification dans le contexte de sécurité. Une session est ensuite créée pour l'utilisateur avec un délai d'expiration configurable (nous avons choisi 300 secondes d'inactivité).

#### Gestion des erreurs de connexion :

Si les informations sont invalides, le contrôleur renvoie l'utilisateur vers la page de connexion avec un paramètre d'erreur (redirect:/login?error=invalid). SweetAlert2 peut afficher des messages personnalisés pour informer l'utilisateur de l'erreur.

#### Sécurité et cryptage des mots de passe :

Spring Security et BCryptPasswordEncoder assurent la sécurité des mots de passe lors de la comparaison dans UserServices. Le mot de passe de l'utilisateur n'est jamais stocké en texte clair mais est toujours encodé, ce qui protège contre les attaques par vol de données en cas de compromission.

Voir en annexe 5 : Diagrammes des classes et séquences partie utilisateur

### c. Organisation

P base

OpenProject

#### Lots de Travaux

SUJET	STATUT	DATE DE DÉBUT	DATE DE FIN	ASSIGNÉ À	DURÉE
Ajout de la méthode dans la classe UserController	Clôturé	15-10-2024	18-10-2024	Rafik Rharmaoui	4 j
Ajout de la méthode dans la classe UserService	Clôturé	15-10-2024	18-10-2024	Ahmed-El-Aziz Kaidi	4 j
Création de la vue pour la page connexion	Clôturé	15-10-2024	18-10-2024	Wassim Ennaji	4 j

Figure 9 : Tâches connexion

## 3. Modification des informations du compte

### a. Descriptif fonctionnel

La fonctionnalité de modification des informations du compte permet aux utilisateurs de mettre à jour leur adresse mail liée au compte ou le mot de passe. Lorsqu'un utilisateur est connecté et clique sur "Mon Profil", son nom, prénom et adresse e-mail sont affichées. Ces données ne peuvent pas être modifiées directement dans les champs affichés. Lors de la modification de l'adresse, celle-ci doit respecter un format valide (présence d'un arobase et d'un nom de domaine), et l'utilisateur doit s'assurer que l'adresse e-mail saisie n'existe pas déjà dans la base de données. Si l'adresse e-mail est invalide ou déjà utilisée, un pop-up d'erreur s'affiche pour avertir l'utilisateur. Pour valider le changement d'adresse e-mail, l'utilisateur doit obligatoirement entrer son mot de passe actuel.

En ce qui concerne la modification du mot de passe, celle-ci se déroule en dessous de la section de modification de l'e-mail. L'utilisateur doit saisir son ancien mot de passe, puis entrer un nouveau mot de passe qu'il devra confirmer. Le système vérifie que l'ancien mot de passe est correct, que le nouveau mot de passe est différent de l'ancien et que le champ de confirmation du nouveau mot de passe correspond. Si l'une de ces conditions n'est pas remplie, des pop-ups d'erreur s'affichent pour signaler le problème. L'utilisateur devra ensuite cliquer sur le bouton "Modifier" pour valider la modification de son mot de passe.

### b. Descriptif technique

Technologies utilisées :

- **Spring Boot** : Pour la structure de l'application.
- **Spring Security** : Pour gérer l'authentification et la sécurité des comptes.
- **Thymeleaf** : Pour le rendu des vues côté serveur.
- **SweetAlert2** : Pour afficher des pop-ups interactifs et des messages de confirmation ou d'erreur à l'utilisateur.
- **MySQL** : Base de données pour stocker les informations utilisateur.

### Structure et composants de la fonctionnalité

#### 1. Contrôleur UserController :

- o Le contrôleur UserController expose deux endpoints pour la modification des informations du compte :

- **@PostMapping("/account/update\_email")** : gère la mise à jour de l'adresse e-mail de l'utilisateur. Le mot de passe actuel est vérifié, le format de l'adresse e-mail doit être correct. Suivant le succès ou l'échec de la modification l'utilisateur est redirigé vers la page de profil avec un message de succès ou d'erreur.
- **@PostMapping("/account/update\_password")** : gère la mise à jour du mot de passe de l'utilisateur en vérifiant que le nouveau mot de passe soit différent de l'ancien et vérifie les informations de connexion avant de procéder à la mise à jour du mot de passe, redirigeant l'utilisateur avec un message de succès ou d'erreur en fonction du résultat.

## 2. Service UserServices :

- La méthode **updateEmail(String currentEmail, String newEmail, String currentPassword)**  
Cette méthode permet de mettre à jour l'adresse e-mail d'un utilisateur.
- La méthode **updatePassword(String email, String oldPassword, String newPassword)**  
: Cette méthode permet de changer le mot de passe d'un utilisateur.

## 3. Gestion des erreurs de connexion :

Si les informations sont invalides, le contrôleur renvoie l'utilisateur vers la page de connexion avec un paramètre d'erreur (`redirect:/login?error=invalid`). SweetAlert2 peut afficher des messages personnalisés pour informer l'utilisateur de l'erreur.

## 4. Sécurité et cryptage des mots de passe :

Spring Security et BCryptPasswordEncoder assurent la sécurité des mots de passe lors de la comparaison dans UserServices. Le mot de passe de l'utilisateur n'est jamais stocké en texte clair mais est toujours encodé, ce qui protège contre les attaques par vol de données en cas de compromission.

Voir en annexe 5 : Diagrammes des classes et séquences partie utilisateur

### c. Organisation

OpenProject

#### Lots de Travaux

SUJET	STATUT	DATE DE DÉBUT	DATE DE FIN	DURÉE	ASSIGNÉ À
Modifications des informations du compte	Clôturé	16-10-2024	23-10-2024	6 j	Aya El-Hassani
Ajout de la méthode dans la classe UserController	Clôturé	16-10-2024	23-10-2024	6 j	Aya El-Hassani
Ajout de la méthode dans la classe UserService	Clôturé	16-10-2024	23-10-2024	6 j	Imane Choukri
Création de la vue de la page	Clôturé	16-10-2024	23-10-2024	6 j	Alan Guilleminot
Ajout d'une notification pour informer l'utilisateur du succès / échec	Clôturé	16-10-2024	23-10-2024	6 j	Imane Choukri
Réalisation des tests unitaires	Clôturé	17-10-2024	23-10-2024	5 j	Alan Guilleminot

Figure 10 : Tâches modification des informations du compte

## 4. Affichage des données

d'autres formats prévus ?

### a. Descriptif fonctionnel

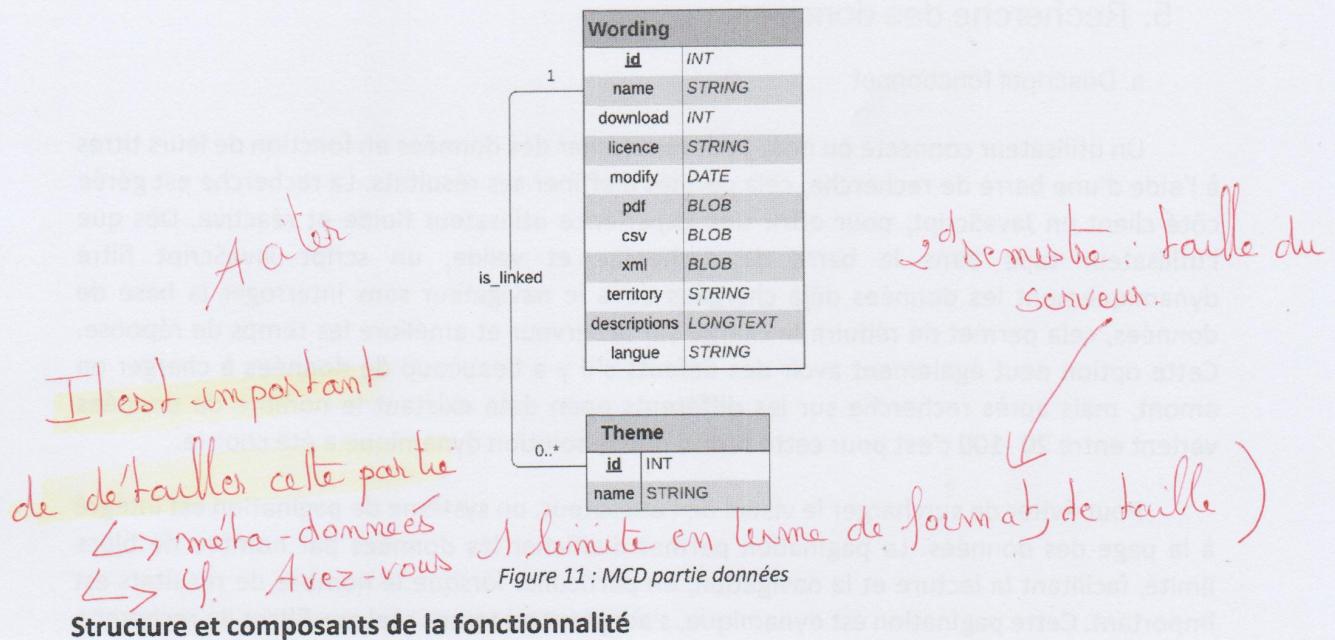
L'affichage des données permet aux utilisateurs de visualiser les données stockées dans la base de données de manière claire et structurée. Lorsqu'un utilisateur (connecté ou non) accède à la page des données, une liste des éléments lui est présentée avec des informations clés telles que le nom, le thème, la description, la licence, la langue et les fichiers (PDF, CSV, XML). Chaque donnée est présentée sous forme d'un élément, un bloc affichant son titre et sa description. Afin de gérer le cas où le nombre de données est important un système de pagination est mis en place. Cela permet d'afficher un nombre limité d'éléments par page, ce qui améliore la lisibilité et la performance. La pagination est dynamique et s'ajuste automatiquement en fonction du nombre d'éléments disponibles et des filtres appliqués par l'utilisateur. Ce système de pagination aide à maintenir une navigation fluide et à éviter une surcharge visuelle. En regardant le détail d'une donnée l'utilisateur peut visualiser en direct les données sous forme d'un tableau et le pdf. (voir annexe p. 35)

Des boutons de téléchargement sont présents pour chaque type de fichier, permettant de récupérer les fichiers dans le format souhaité. En outre, la présentation de chaque fichier peut inclure une petite description ou une icône pour identifier son contenu. L'objectif principal de cette fonctionnalité est d'offrir une interface claire, fonctionnelle et performante, permettant à l'utilisateur de naviguer, trier, filtrer et télécharger les données sans difficulté, tout en maintenant une charge minimale sur le serveur.

### b. Descriptif technique

- Rediger
- **Spring Boot** : Structure de l'application backend.
  - **Thymeleaf** : Utilisé pour le rendu des pages côté serveur, permettant d'afficher dynamiquement les données sur l'interface utilisateur.
  - **JavaScript** : Améliore l'interactivité de l'application en gérant la pagination côté client.
  - **Base de données** : Utilisation de MySQL avec Spring Data JPA pour gérer les données, y compris les relations entre entités et le stockage des fichiers volumineux (PDF, CSV, XML) sous forme de blobs.

Je mettrai l'annexe 6 en valeur (c'est le cœur du projet)



### Structure et composants de la fonctionnalité

#### 1. Contrôleur WordingController :

Le contrôleur WordingController expose deux endpoints pour l'affichage de toutes les données et l'affichage des détails d'une donnée :

`@GetMapping("/search")` : gère l'affichage de toutes les données sous forme de bloc.

`@GetMapping("/wording/{id}")` : gère l'affichage des informations d'un donnée.

#### 2. Service WordingService :

- La fonction `IstWording()` est utilisée afin de récupérer toutes les données disponibles dans la base de données.
- La fonction `getWordingById(int id)` permet de récupérer les informations détaillées d'un jeu de données spécifique en fonction de son identifiant unique (ID).

### c. Organisation

SUJET	STATUT	DATE DE DÉBUT	DATE DE FIN	DURÉE	ASSIGNÉ À
Créer la vue pour l'affichage des données sur le portail	Clôturé	15-10-2024	25-10-2024	9 j	Aichetou N-diaye
Créer la méthode dans la classe service pour récupérer les données	Clôturé	15-10-2024	25-10-2024	9 j	Trinite Mombouli
Créer la méthode dans la classe contrôleur pour afficher les données	Clôturé	15-10-2024	25-10-2024	9 j	Trinite Mombouli
Réaliser les tests unitaires	Clôturé	17-10-2024	25-10-2024	7 j	Trinite Mombouli
Implémenter la pagination pour la visualisation	Clôturé	21-10-2024	25-10-2024	5 j	Aichetou N-diaye

Figure 12 : Tâches affichage des données

## 5. Recherche des données

### a. Descriptif fonctionnel

Un utilisateur connecté ou non, peut rechercher des données en fonction de leurs titres à l'aide d'une barre de recherche, cela permet d'affiner ses résultats. La recherche est gérée côté client en JavaScript, pour offrir une expérience utilisateur fluide et réactive. Dès que l'utilisateur tape dans la barre de recherche et valide, un script JavaScript filtre dynamiquement les données déjà chargées dans le navigateur sans interroger la base de données, cela permet de réduire la charge sur le serveur et améliore les temps de réponse. Cette option peut également avoir des défauts s'il y a beaucoup de données à charger en amont, mais après recherche sur les différents open data existant le nombre de données varient entre 20 -100 c'est pour cette raison que la solution dynamique a été choisie.

Pour éviter de surcharger le visuel de l'utilisateur, un système de pagination est intégré à la page des données. La pagination permet d'afficher les données par nombre de blocs limité, facilitant la lecture et la navigation, en particulier lorsque le nombre de résultats est important. Cette pagination est dynamique, s'adaptant en temps réel aux filtres de recherche appliqués par l'utilisateur.

### b. Descriptif technique

Les technologies utilisées sont

- Spring Boot : Pour la structure de l'application,
- Thymeleaf : Pour le rendu des vues côté serveur et
- Javascript : Gère la logique de recherche dynamique et la pagination côté client pour améliorer l'interactivité et la réactivité de l'application.

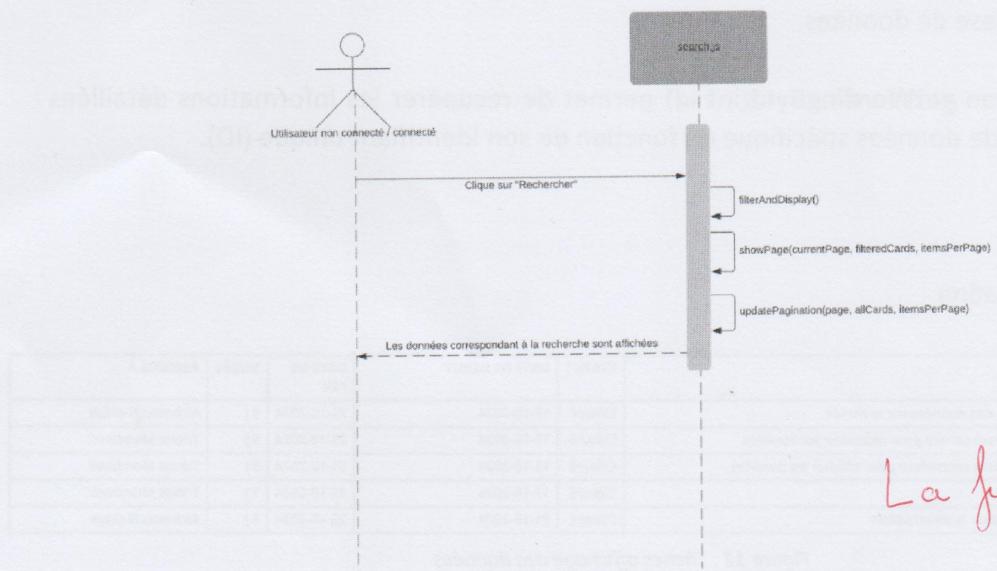


Figure 13 : Diagramme séquence recherche

### c. Organisation

SUJET	STATUT	DATE DE DÉBUT	DATE DE FIN	DURÉE	ASSIGNÉ À
Création fonction js pour gérer l'affichage en fonction de la barre de recherche	Clôturé	04-11-2024	07-11-2024	4 j	Anthony Michaud
Modification de la vue pour l'ajout de la barre de recherche	Clôturé	07-11-2024	08-11-2024	2 j	Anthony Michaud
Vérification et prise en compte des injections SQL	Clôturé	08-11-2024	08-11-2024	1 j	Anthony Michaud

Figure 14 : Tâche recherche des données

## 6. Filtrage des données

### a. Descriptif fonctionnel

Un utilisateur connecté ou non peut filtrer des données en fonction du nombre de téléchargement, ce qui permet de connaître les plus populaires. On peut filtrer en fonction de la licence associée, du thème et du territoire. On peut également filtrer par rapport à la date de modification ce qui permet de récupérer les données les plus récentes. On peut aussi filtrer par langue ou par possibilités d'export (CSV, XML, PDF...). Les filtres sont cumulatifs, c'est-à-dire que les données affichées devront satisfaire tous les filtres renseignés. Le filtrage se fait de manière dynamique, donc les filtres sont appliqués sans recharger la page.

### b. Descriptif technique

Technologies utilisées :

- **Spring Boot** : Pour la structure de l'application.
- **Thymeleaf** : Pour le rendu des vues côté serveur.
- **Javascript** : Gère la logique de recherche dynamique et la pagination côté client pour améliorer l'interactivité et la réactivité de l'application.

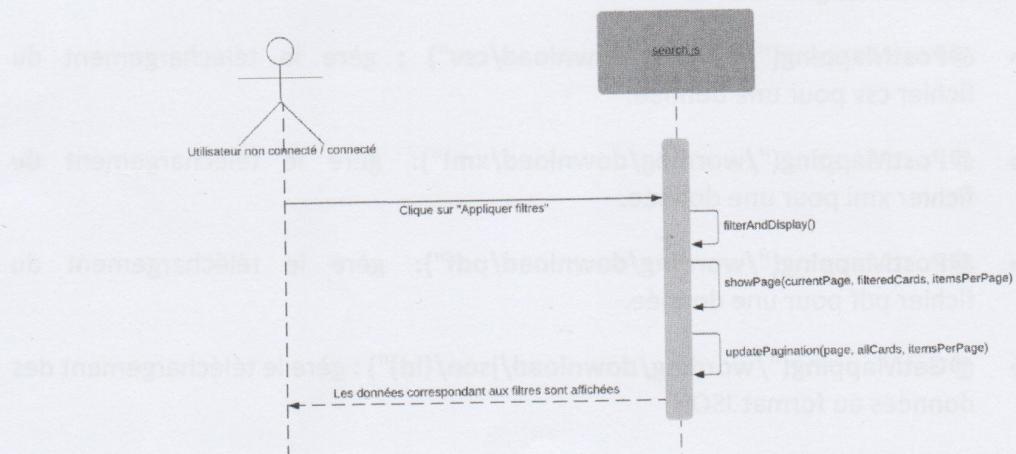


Figure 15 : Diagramme séquence filtre

### Service ThemeService et WordingService :

- La fonction **add(Theme theme)** permet d'ajouter dans la bdd un nouveau thème
- La fonction **add(Wording wording)** permet d'ajouter dans la bdd une nouvelle donnée

#### c. Organisation

SUJET	STATUT	DATE DE DÉBUT	DATE DE FIN	DURÉE	ASSIGNÉ À
Création de la classe DataController	Clôturé	21-11-2024	22-11-2024	2 j	Trinite Mombouli
Créer la classe DataService	Clôturé	25-11-2024	25-11-2024	1 j	Aichetou N-diaye
Création de la vue pour permettre à l'administrateur de téléverser les données	Clôturé	26-11-2024	27-11-2024	2 j	Trinite Mombouli

Figure 18 : Tâches importation des données

### 9. Choix de la licence

- a. Descriptif fonctionnel
- b. Descriptif technique
- c. Organisation

## VII. Normes et tests unitaires

Au début du projet, nous nous sommes mis d'accord sur les normes de codage à respecter. Étant donné que nous développons en Java, nous avons décidé de suivre les conventions standard de codage Java.

- Noms méthodes, fonctions, attributs en anglais avec la première lettre du premier mot en minuscule ensuite en majuscule.
- Noms des classes en anglais et première lettre en majuscule
- Commentaires de commit, documentation et commentaire de code en français

Pour les tests unitaires, un test est réalisé pour chaque méthode et fonction cela s'applique aux couches controller, service, et model. En parallèle, des tests d'intégration sont également réalisés. Les référents qualité s'assurent que la couverture des tests unitaires est complète. Nous avons également mis en place un runner GitLab afin d'automatiser la validation des tests unitaires à chaque nouveau commit sur la branche principale, cela permet d'assurer une qualité constante du code et de détecter rapidement les régressions. Ce pipeline CI/CD (Intégration Continue et Déploiement Continu) est configuré pour exécuter automatiquement les tests unitaires.

Voici les tests unitaires exécuté ainsi que le niveau de couverture du code

La figure 19 est 1 capture d'écran présentant

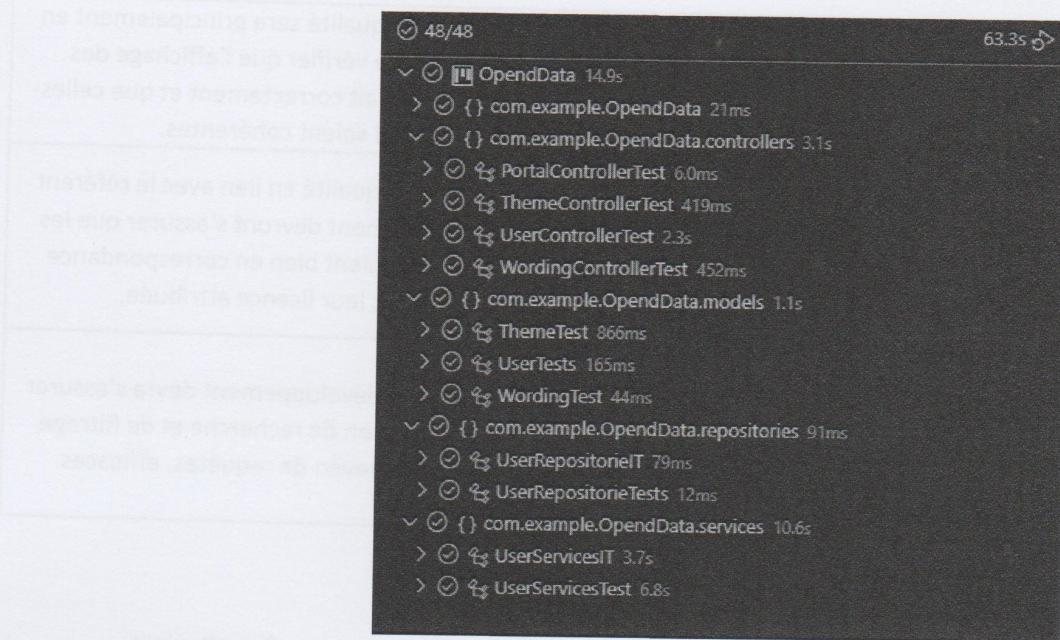


Figure 19 : Tests unitaires validé

Pour la validation de chaque fonctionnalité, plusieurs étapes sont suivies :

1. Les référents développement vérifient que les normes de codage sont respectées.
2. Les référents qualité s'assurent que tous les tests unitaires passent et que la couverture est totale.
3. Pour terminer, le chef de projet valide la fonctionnalité puis intègre la branche de la fonctionnalité dans la branche principale (main).

Voir annexe 7 : Classe de tests unitaires *1 phase*

## VIII. Organisation du groupe

*Le tableau 3 présente résume ...*

Fonctionnalités	Rôle principal	Rôles secondaires	Explications du rôle principal
Création de compte	Référent développement	Référent qualité	Toutes les parties liées à la création, modification de compte et connexion seront gérées par le référent développement, qui assistera les développeurs pour un avancement conjoint.
Connexion	Référent développement	Référent données	
Modifications des informations	Référent développement	Référent données / Référent qualité	
Importer des données	Référent données	Référent qualité	Le référent des données, responsable de la cohérence, s'assurera qu'il n'y a pas d'incohérence avec les bases de données.
Télécharger les données	Référent données	Référent qualité	

Voir les données	Référent qualité	Référent données	Le référent qualité sera principalement en charge de vérifier que l'affichage des données se fait correctement et que celles-ci soient cohérentes.
Affichage / Choix politique de licence	Référent qualité	Référent développement	Le référent qualité en lien avec le référent développement devront s'assurer que les données soient bien en correspondance avec leur licence attribuée.
Recherche de données	Référent développement	Référent Données / Réf qualité	Le référent développement devra s'assurer que le moyen de recherche et de filtrage soit, au moyen de requêtes, efficaces.
Filtrage des données	Référent développement	Référent données	

Tableau 3 : Table des fonctionnalités et leurs rôles

Voici un aperçu des tâches de fonctionnalités que nous avons ajoutées sur OpenProject. Nous avons réparti les tâches de manière logique entre les alternants et les non-alternants, tout en estimant le temps nécessaire pour chacune, afin de respecter les délais du projet. Certaines fonctionnalités possèdent beaucoup de membres car ce sont des alternants. Les alternants possèdent moins de temps de travail sur le projet donc nous les avons regroupés pour qu'il soit plusieurs et avons également ajouté un membre en formation initiale pour les aider. Chaque membre ne travaillera pas sur toutes les fonctionnalités, mais en cas de problème des membres ayant fini leurs tâches ou n'en possédant plus pourront aller aider les autres.

*Le tableau xx présente*

Vous trouverez ci-dessous la répartition des fonctionnalités par membre.

Fonctionnalités	Membres assignés
Création de compte	Anthony MICHAUD, Kévin PRADIER
Connexion	Rafik RHARMAOUI, Ahmed KAIDI, Wassim ENNAJI
Modifications des informations	Aya ELHASSANI, Imane CHOUKRI, Alan GUILLELINOT
Importer des données	Aichetou N'DIAYA, Trinité MOMBOLU
Affichage des données	Aya ELHASSANI, Imane CHOUKRI, Alan GUILLELINOT, Aichetou N'DIAYA
Télécharger les données	Trinité MOMBOLU
Affichage / Choix politique de licence	Rafik RHARMAOUI, Ahmed KAIDI, Wassim ENNAJI
Filtrage des données	Kévin PRADIER
Recherche de données	Anthony MICHAUD

*Tableau xx*

Figure 20 : Tableau de répartition des fonctionnalités par membre

✓ ✓ ✓ Voir annexe 1, 2 et 3 : Gantt + Détails des tâches + Organisation des réunions

Chaque fonctionnalité a été réalisée en suivant les mêmes étapes. D'abord la réalisation d'une maquette fil de fer que ce soit pour l'ajout ou la modification d'éléments sur l'interface. Une fois la maquette validée, le développement de la fonctionnalité est réalisé en suivant les normes de codage définies en début de projet. L'implémentation respecte l'architecture du projet notamment le modèle MVC (Modèle-Vue-Contrôleur). Après le développement, des tests unitaires sont réalisés. Il n'y a pas de différence entre le planning initial et final, nous avons réussi à réaliser toutes les tâches. Grâce à une bonne organisation nous n'avons pas pris de retard sur notre travail.

## IX. Conclusion

Ce projet nous a permis de découvrir et de travailler dans toutes les étapes du développement d'une application web, de la conception à l'implémentation. Nous avons amélioré nos compétences techniques, notamment en Java, HTML, CSS, et JavaScript, tout en explorant des outils et des frameworks comme Spring Boot et Thymeleaf. Ce projet nous a également permis d'améliorer nos méthodes de travail en équipe. Nous avons appris à mieux planifier nos tâches, à respecter les délais, et à collaborer de manière plus efficace en tirant parti des retours et des validations de nos référents et du chef de projet.

La prise en main de Spring Boot qui était un framework nouveau pour la majorité de l'équipe a été un défi nécessitant une autoformation et une période d'adaptation. Travailler à dix personnes sur un projet a également complexifié la coordination et ralenti certaines décisions, cependant cela a renforcé nos compétences collaboratives.

Lors de la deuxième partie du projet au semestre 2, nous nous concentrerons sur la partie statistique du projet.

Avez-vous aîné toutes vos annexes ?

## X. Liens

- Lien du rapport sur google drive :

<https://docs.google.com/document/d/1brRBeXUFYW26YZNqscM9c04Kw5B8FnxG/edit?usp=sharing&ouid=114233599138467094838&rtpof=true&sd=true>

- Lien du wiki GitLab compte rendu des réunions :

<https://gitlabvigan.iem/groupe6/opendatametro/wikis/Comptes-Rendus-de-R%C3%A9union>