

DOCUMENT DE REPORTING

Prototype du projet de MedHead



Table des matières

| | |
|--|-----------|
| OBJET DU DOCUMENT..... | 3 |
| CONTEXTE | 3 |
| MISE EN PLACE DE L'ENVIRONNEMENT DE TRAVAIL | 4 |
| Repositories GitHub | 4 |
| Environnement de développement..... | 5 |
| DEVELOPPEMENT DU BACK-END | 6 |
| Base de données..... | 6 |
| API | 9 |
| Création de l'API..... | 9 |
| Structure..... | 9 |
| Configuration..... | 10 |
| Fonctionnalités | 11 |
| Application web..... | 13 |
| Création de l'application web..... | 13 |
| Structure..... | 13 |
| Configuration..... | 14 |
| Fonctionnalités | 14 |
| DEVELOPPEMENT DU FRONT-END..... | 15 |
| Interfaces..... | 15 |
| TESTS..... | 17 |
| Tests unitaires | 17 |
| Tests de montée de charge..... | 17 |
| Tests fonctionnels..... | 17 |
| Données de test..... | 17 |
| LIVRAISON..... | 18 |
| ASPECTS FONCTIONNELS COMPRIS DANS LE POC | 19 |
| Connexion à la plateforme..... | 19 |
| Recherche d'un hôpital..... | 21 |
| Réservation d'un lit | 23 |

OBJET DU DOCUMENT

Ce document de reporting fait lieu de synthèse et de justification des choix techniques et des résultats du développement du PoC du projet de MedHead.

CONTEXTE

MedHead est un regroupement de grandes institutions médicales œuvrant au sein du système de santé britannique et assujetti à la réglementation et aux directives locales (NHS). Les organisations membres du consortium utilisent actuellement une grande variété de technologies et d'appareils. Ils souhaitent une nouvelle plateforme pour unifier leurs pratiques.

L'objectif final du projet est d'améliorer les soins dispensés aux patients grâce à cette nouvelle plateforme. Afin de réduire le risque pour le système d'intervention d'urgence, un prototype a été construit. Ce prototype est donc l'objet du document.

MISE EN PLACE DE L'ENVIRONNEMENT DE TRAVAIL

Avant de pouvoir développer le prototype il a fallu d'abord configurer l'environnement de travail.

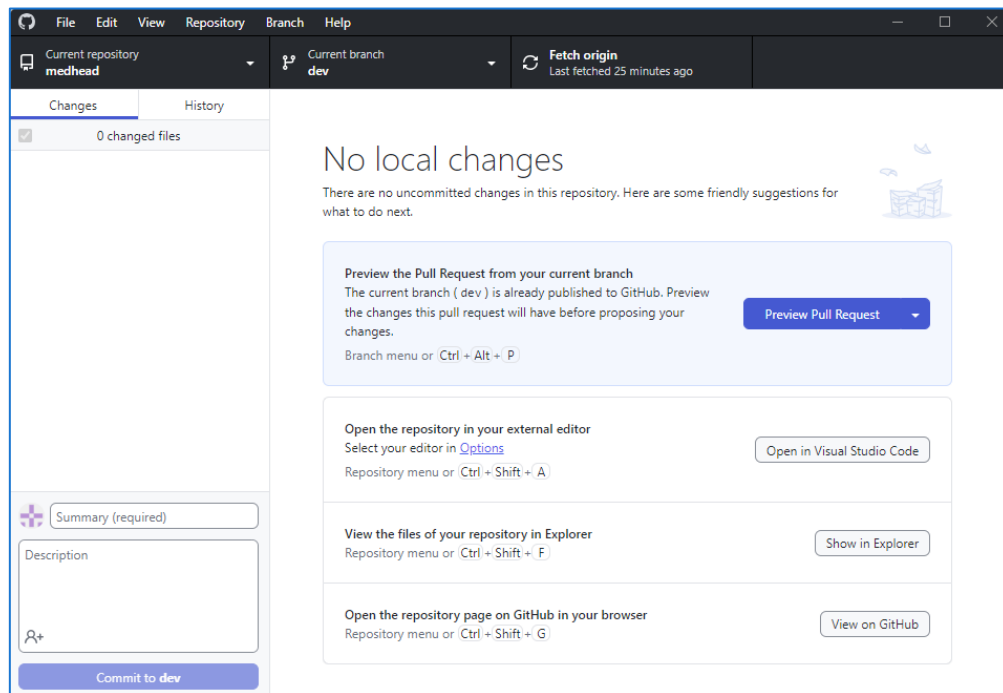
Repositories GitHub

La première étape réalisée pour la configuration de l'environnement a été la création des repositories GitHub du projet :

- Un repository pour le code :
 - Lien GitHub : <https://github.com/ClementineAllard/medhead.git>
 - Ce repository est constitué du code de l'API dans le dossier `./api` et du code de l'application web dans le dossier `./webapp`
 - Pour ce repository, deux branches ont été créées :
 - une branche *main* qui permet de stocker le code stable du projet
 - une branche *dev* qui permet à un développeur de récupérer le code et de le modifier sur son environnement local. Une fois qu'un développeur termine son développement il doit l'archiver sur sa branche puis l'envoyer sur la branche principale.
 - Une branche sera créée pour chaque développeur du projet pour qu'ils puissent collaborer avec l'ensemble de l'équip.
- Un repository pour les documents d'architecture :
 - Lien GitHub : <https://github.com/ClementineAllard/medheaddoc.git>
 - Dans ce repository se trouvent les documents sources du projet et le document de reporting

Ces deux repositories ont été ensuite clonés dans un répertoire local : C:\Users\{user}\Documents\GitHub.

Dans le but de simplifier la gestion de la collaboration, des archivages, des récupérations de codes, etc, l'outil **GitHub Desktop** a été utilisé.



Interface de GitHub Desktop

Cette application bureau permet d'exécuter les commandes git sans utiliser d'invite de commande et offre une interface affichant les documents modifiés et les actions possibles.

Environnement de développement

Ensuite il a fallu configurer l'environnement technique pour écrire le code du PoC.

Pour le code de l'API et de l'application web :

- L'IDE **Visual Studio Code** a été utilisée
- Avec **Postman** pour tester les endpoints de l'API

Et pour la partie base de données :

- Le service **MySQL** a été mis en place
- Accompagné de l'outil **MySQLWorbench**

DEVELOPPEMENT DU BACK-END

Base de données

Après avoir correctement configuré l'environnement, la base de données **MySQL** du PoC a pu être créée. En voici le diagramme :

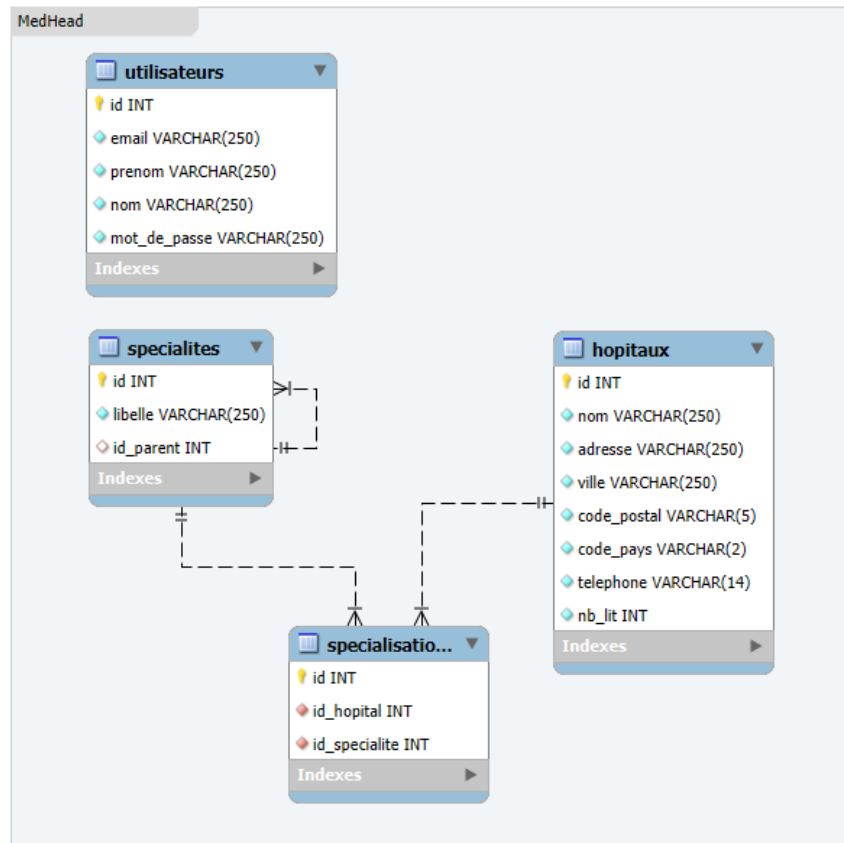
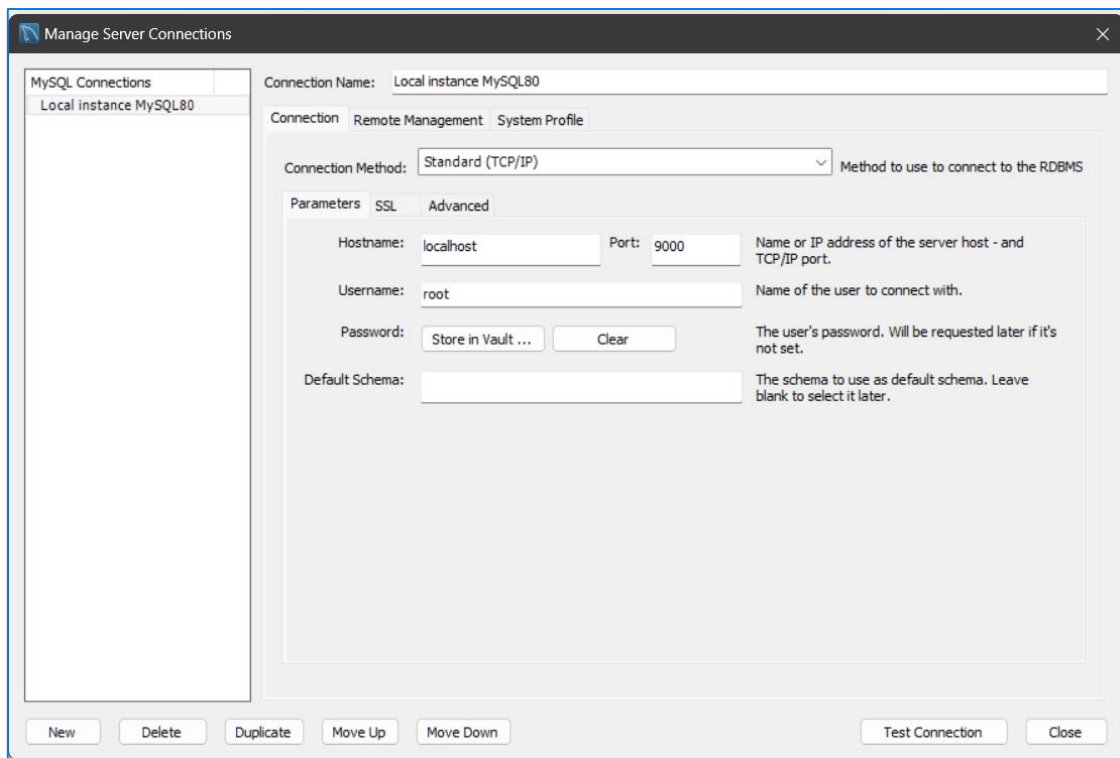


Schéma base de données du PoC

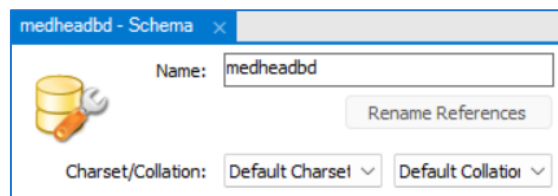
Voici les étapes qui ont été réalisées pour obtenir cette base de données :

- **Connexion MySQL :** établissement d'une connexion depuis le premier onglet de MySQLWorkbench



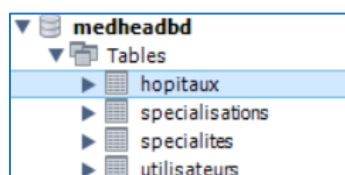
Interface de connexion de MySQL Workbench (Avec Password = root)

- **Création du schéma** : création d'un nouveau schéma nommé *medheadbd* grâce à la connexion réalisée à l'étape précédente



Interface de création de schéma de MySQL Workbench

- **Création des tables de données** : dans le schéma qui vient d'être créé, exécution du script qui contient la structure utilisée par le PoC (celui-ci est stocké sur le repository de code à l'emplacement : [medhead/api/src/main/resources/data.sql](#)). Après cette étape, le schéma doit correspondre à cela :



Liste des tables de la base de données medheadbd

Après ces étapes, la base de données doit également contenir les données de test.

1 • `SELECT * FROM medheadbd.hopitaux;`

| | id | nom | adresse | ville | code_postal | code_pays | telephone | nb_lit |
|---|----|---|------------------------------|-------------|-------------|-----------|----------------|--------|
| ▶ | 1 | Hôpital Saint-Jean | 12 Rue des Lilas | Paris | 75015 | FR | 01 40 23 45 67 | 0 |
| | 2 | Centre Hospitalier Universitaire Dupont | 85 Avenue des Champs Élysées | Lyon | 69006 | FR | 04 78 24 56 78 | 300 |
| | 3 | Clinique de la Paix | 45 Rue de la Paix | Marseille | 13001 | FR | 04 91 53 21 45 | 80 |
| | 4 | Hôpital Général de Bordeaux | 10 Rue Sainte-Catherine | Bordeaux | 33000 | FR | 05 56 90 33 44 | 250 |
| | 5 | Centre Médical de Lille | 22 Boulevard de Strasbourg | Lille | 59000 | FR | 03 20 60 77 88 | 200 |
| | 6 | Hôpital Saint-Vincent | 9 Rue de la République | Toulouse | 31000 | FR | 05 34 45 67 89 | 180 |
| | 7 | Clinique des Cèdres | 18 Allée des Acacias | Nice | 06000 | FR | 04 93 56 78 90 | 150 |
| | 8 | Hôpital de l'Océan | 53 Avenue de la Plage | Biarritz | 64200 | FR | 05 59 22 33 44 | 100 |
| | 9 | Hôpital Universitaire de Strasbourg | 32 Rue de la Liberté | Strasbourg | 67000 | FR | 03 88 45 67 89 | 350 |
| | 10 | Clinique du Soleil | 21 Rue du Soleil Levant | Montpellier | 34000 | FR | 04 67 56 78 90 | 90 |
| | 11 | Centre de Soins d'Orléans | 77 Rue Royale | Orléans | 45000 | FR | 02 38 56 78 90 | 110 |
| | 12 | Hôpital Sainte-Marie | 15 Rue du Château | Rennes | 35000 | FR | 02 99 23 45 67 | 200 |

Exemple du contenu de la table Hôpitaux

Les données de références sur les spécialités NHS, fournies dans les sources du projet, sont présentes dans la table *Spécialités*. Les groupes de spécialités ont un *id_parent* à nul et les sous-spécialités ont l'id du groupe parent renseigné dans leur attribut *id_parent*, c'est ainsi qu'il est possible de différencier les groupes des spécialités.

Voici l'exemple pour le groupe de spécialité *Anesthésie* :

| Groupe de spécialité | Spécialité |
|----------------------|-----------------|
| Anesthésie | Anesthésie |
| Anesthésie | Soins intensifs |

Tableau de références

| | id | libelle | id_parent |
|---|----|-----------------|-----------|
| ▶ | 1 | Anesthésie | NULL |
| | 2 | Anesthésie | 1 |
| | 3 | Soins intensifs | 1 |

Représentation en base

API

Cette partie du projet permet de créer toutes les fonctions nécessaires pour la gestion de la base de données.

Les technologies qui ont été choisies pour l'API sont les suivantes :

- **Java** : pour développer le système
- **SpringBoot** : pour simplifier la configuration et permettre un développement rapide
- **Maven** : pour la compilation du projet

Création de l'API

L'API a été générée grâce à **Spring Initializr** avec les caractéristiques suivantes :

The screenshot shows the Spring Initializr web interface. On the left, under 'Project', 'Maven' is selected. Under 'Language', 'Java' is selected. Under 'Spring Boot', '3.3.5' is selected. The 'Project Metadata' section contains fields for Group (com.medhead), Artifact (api), Name (api), Description (API with Spring Boot), and Package name (com.medhead.api). The 'Packaging' is set to 'Jar' and the 'Java' version is '17'. On the right, the 'Dependencies' section lists 'Spring Web' (WEB), 'Lombok' (DEVELOPER TOOLS), 'MySQL Driver' (SQL), and 'Spring Data JPA' (SQL). A button 'ADD DEPENDENCIES... CTRL + B' is visible at the top right of the dependencies section.

Interface de Spring Initializr

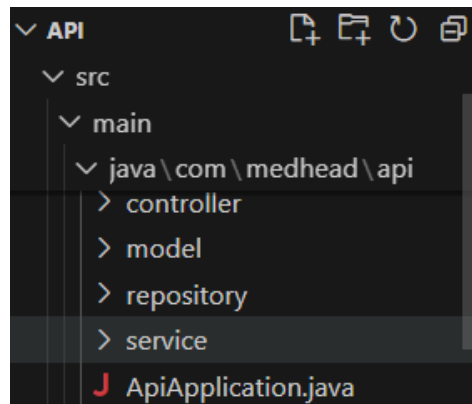
Les dépendances ajoutées permettent :

- **Spring Web** : de faire du RESTful pour exposer des endpoints
- **Lombok** : d'optimiser certaines classes, par exemple pour éviter d'avoir à écrire les getters, setters et constructeurs
- **MySQL Driver** : de connecter l'API à une base MySQL
- **Spring Data JPA** : de gérer la persistance des données

Structure

Les différents packages du projet sont les suivants :

- **Controller** : permet de réceptionner une demande et retourne la réponse
- **Model** : contient les objets métiers
- **Repository** : communique avec la base de données
- **Service** : exécute les traitements métiers



Structure du code de l'API

Configuration

Pour rendre l'API fonctionnelle il a fallu la connecter à la base de données qui a été créée et la rendre accessible depuis un port. Pour cela, le document de propriétés a été modifié :

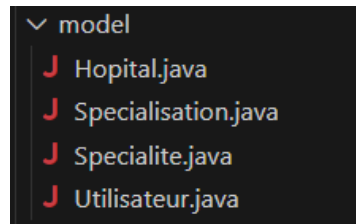
```
src > main > resources > application.properties
1  #Global configuration
2  spring.application.name=api
3
4  #Tomcat configuration
5  server.port=8080
6
7  #Log level configuration
8  logging.level.root=ERROR
9  logging.level.com.medhead=INFO
10 logging.level.org.springframework.boot.web.embedded.tomcat=INFO
11
12 #MySQL Configuration
13 spring.datasource.url=jdbc:mysql://localhost:9000/medheadbd
14 spring.datasource.username=root
15 spring.datasource.password=root
16 spring.jpa.show-sql=true
```

Fichier de propriétés de l'API

Notre API est donc ainsi connectée à la base medheadbd et sera accessible depuis le **port 8080**.

Fonctionnalités

Liste des modèles :



Modèles de l'API

Comme vous pouvez le constater, les modèles correspondent tout simplement aux tables de la base de données.

Chaque modèle a son service, son repository et son controller.

Pour chaque modèle, des fonctions d'insertion, de consultation, de suppression et de modification ont été créés.

Fonctions particulières ajoutées :

- **Utilisateur**
 - Cryptage du mot de passe lors d'une insertion ou modification
 - Cette fonction est faite pour qu'à partir d'une clé, les caractères d'une chaîne soient cryptés un à un :

```
* Cryptage d'une chaîne
* @param texteACrypter - texte devant être crypté
* @param cleCryptage - clé pour crypter le texte
* @return texte crypté avec la clé
*/
public String crypter(String texteACrypter, String cleCryptage){
    // On initialise les variables
    int j=0, i, intTemp, intTemp2, intRes;
    StringBuilder strbTemp = new StringBuilder();
    String texteCrypte = "";
    char[] charACrypter, charCle;

    try {
        // On convertit les chaînes en tableau de caractères
        charACrypter = texteACrypter.toCharArray();
        charCle = cleCryptage.toCharArray();

        // On parcourt la chaîne à crypter pour crypter chaque caractère
        for (i = 0; i < charACrypter.length; i++){
            intTemp = (int) charACrypter[i];
            intTemp2 = (int) charCle[j];

            intRes = intTemp ^ intTemp2; // cryptage du caractère
            strbTemp.append(String.format(format:"%03d", intRes)); // ajout à la chaîne finale

            j++;
            if(j >= charCle.length){ // si on a dépassé la taille de la clé, on recommence le parcours
                j=0;
            }
        }

        texteCrypte = strbTemp.toString(); // on obtient le texte crypté
    }
}
```

Extrait du code de cryptage

- Une fois le mot de passe crypté, il est enregistré en base.
- Comparaison entre le mot de passe saisi et celui qui a été crypté
 - Pour cette fonction il est primordial d'avoir la même clé que celle qui a permis de crypter le mot de passe d'origine
- **Hôpital**
 - Réserve de lit
 - Diminution du nombre de lits disponibles de l'hôpital
- **Spécialité**
 - Consultation de la liste des groupes de spécialités
 - Recherche des sous-spécialités à partir d'un id de groupe
- **Spécialisation**
 - Recherche des hôpitaux ayant des lits disponibles et ayant la spécialité demandée

Toutes ces fonctionnalités peuvent être testées avec Postman grâce au script ajouté au projet à l'emplacement : [medhead/api/src/main/resources/MedHeadCollection.postman_collection.json](#).

Application web

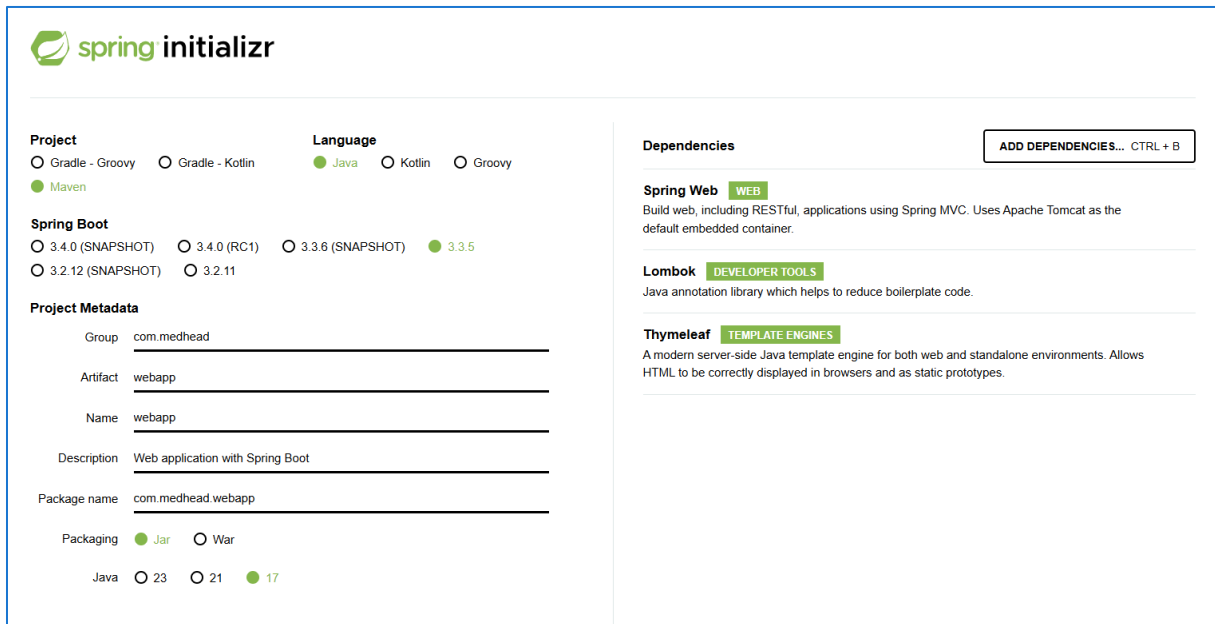
Cette application permet aux utilisateurs d'accéder aux fonctionnalités du système. A la suite d'une demande des utilisateurs, les appels à l'API sont effectués et l'application affiche les résultats.

Les technologies qui ont été choisies pour l'application web sont les suivantes :

- **Java** : pour développer le système
- **SpringBoot** : pour simplifier la configuration et permettre un développement rapide
- **Maven** : pour la compilation du projet
- **HTML+css+javascript** : pour le fonctionnement et l'affichage des interfaces

Création de l'application web

L'application a été générée, comme l'API, grâce à **Spring Initializr** avec les caractéristiques suivantes :



The screenshot shows the Spring Initializr web interface. It is divided into two main sections: Project Metadata and Dependencies.

Project Metadata:

- Project:** Radio buttons for **Gradle - Groovy**, **Gradle - Kotlin**, **Maven** (selected), and **Other**.
- Language:** Radio buttons for **Java** (selected), **Kotlin**, and **Groovy**.
- Spring Boot:** Radio buttons for **3.4.0 (SNAPSHOT)**, **3.4.0 (RC1)**, **3.3.6 (SNAPSHOT)**, **3.3.5** (selected), and **3.2.11**.
- Group:**
- Artifact:**
- Name:**
- Description:**
- Package name:**
- Packaging:** Radio buttons for **Jar** (selected) and **War**.
- Java:** Radio buttons for **23**, **21**, and **17** (selected).

Dependencies:

- Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
- Lombok** (DEVELOPER TOOLS): Java annotation library which helps to reduce boilerplate code.
- Thymeleaf** (TEMPLATE ENGINES): A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

Buttons: **ADD DEPENDENCIES... CTRL + B**

Interface de Spring Initializr

Les dépendances ajoutées permettent :

- **Spring Web** : de fournir des pages HTML à afficher
- **Lombok** : d'optimiser certaines classes, par exemple pour éviter d'avoir à écrire les getters, setters et constructeurs
- **Thymeleaf** : de formater les pages

Structure

Les différents packages du projet sont les suivants, semblables à ceux de l'API :

- **Controller** : permet de réceptionner une demande et retourne la réponse
- **Model** : contient les objets métiers
- **Repository** : communique avec l'API
- **Service** : exécute les traitements métiers

Configuration

Pour rendre l'application fonctionnelle il a fallu la rendre accessible depuis un port. Pour cela, le document de propriétés a été modifié :

```
src > main > resources > application.properties
1  #Global configuration
2  spring.application.name=webapp
3
4  #Tomcat configuration
5  server.port=8081
6
```

Fichier de propriétés de l'application web

Notre application est donc ainsi accessible depuis le **port 8081**.

Fonctionnalités

Connexion : Tout d'abord en arrivant sur la plateforme, un utilisateur peut renseigner son identifiant ainsi que son mot de passe pour se connecter à la plateforme. Les informations renseignées sont envoyées à l'API pour les vérifier et celle-ci retourne un booléen à vrai si l'utilisateur peut se connecter.

Recherche d'hôpitaux : Si l'identifiant et le mot de passe sont corrects, l'utilisateur a ensuite la possibilité d'effectuer une recherche parmi les hôpitaux enregistrés en base. Cette recherche est faite à partir d'un lieu et d'une spécialité (groupe ou sous-spécialité).

Premièrement, l'API medhead est appelée pour récupérer tous les hôpitaux ayant des lits disponibles et ayant la spécialité demandée

Puis, parmi les hôpitaux trouvés par l'API, on souhaite connaître celui qui est le plus proche de l'adresse saisie. Pour cette recherche nous appelons **l'API Google Maps Distance Matrix**. L'API Google Maps permet de calculer et de comparer les temps de trajet (en voiture) entre le lieu saisi et les adresses des hôpitaux.

Ainsi, grâce à l'API de medhead et l'API Google Maps, l'hôpital correspondant au mieux à la demande de l'utilisateur s'affiche.

Réservation d'un lit : Une fois qu'un hôpital est trouvé par le système, l'utilisateur peut réserver un lit dans cet hôpital. L'hôpital aura donc un lit disponible de moins.

DEVELOPPEMENT DU FRONT-END

Interfaces

Les interfaces intégrées dans l'application web permettent d'afficher aux utilisateurs toutes les fonctionnalités listées dans la partie précédente.

Comme dit plus tôt dans ce document, les interfaces ont été codées en **HTML** et **css**. Des fonctions **Javascript** ont été ajoutées afin de permettre l'interaction des interfaces avec les fonctions de l'application, cela est possible grâce à des appels **AJAX** ou à l'outil **Thymeleaf**.

```
// On recherche ensuite la liste des spécialités pour remplir le champ
$.ajax({
  method: "GET",
  contentType: "application/json",
  url: "/specialites/"+idGroupe,
  success: function (data) {
    // On met un premier champ vide pour qu'il soit possible de ne pas choisir de spécialité
    $('#specialitesListe').append('<li value=0 class="option"></li>');

    // On ajoute toutes les spécialités dans le champ de sélection
    data.result.forEach(function(specialite) {
      $('#specialitesListe').append('<li value='+ specialite.id + ' class="option" >' + specialite.libelle + '</li>');
    });
  },
  error: function () {
    fAlerte(1,'Erreur lors du chargement des spécialités');
  }
});
```

Exemple de requête AJAX pour récupérer les sous-spécialités d'un groupe

```
<form class="form" method="get" th:action="@{/connexion}" th:object="${utilisateur}">
  <div class="form-group">
    <input type="text" name="username" placeholder="Email" required="" th:field="*{email}">
  </div>
  <div class="form-group">
    <input type="password" name="password" placeholder="Mot de passe" required="" th:field="*{mdp}">
  </div>
  <div class="form-group login-btn">
    <button class="btn" type="submit" >Se connecter</button>
  </div>
</form>
```

Exemple d'envoi de données par Thymeleaf via un formulaire (ici le formulaire de connexion)

Afin d'être plus rapide pour créer ces interfaces, un modèle bootstrap a été récupéré et adapté pour ne pas avoir à créer le style des interfaces. ([lien du template](#))

MedHead

Connexion

admin

Se Connecter

© Copyright 2018 | All Rights Reserved by [wpthemesgrid.com](#)

Interface de connexion

MedHead

Recherche

Toulouse

Anesthésie

Rechercher

Hôpital Saint-Vincent

9 Rue de la République, 31000 Toulouse

05 34 45 67 89

Réserver Un Lit

© Copyright 2018 | All Rights Reserved by [wpthemesgrid.com](#)

Interface de recherche et de de réservation

TESTS

Tests unitaires

Les tests unitaires du PoC

Les tests unitaires de l'API permettent actuellement de tester toutes les fonctions présentes dans les controllers : insertion, consultation, modification et suppression ainsi que les fonctions particulières qui ont été ajoutées. Les tests se trouvent dans le dossier [medhead/api/src/test/java/com/medhead/api](#).

Les tests unitaires de l'application web permettent de tester toutes les fonctions présentes dans le controller c'est-à-dire, celles qui sont utilisées par le système. Les tests se trouvent dans le dossier [medhead/webapp/src/test/java/com/medhead/api](#).

Pour les lancer depuis l'IDE il faut exécuter la commande : `./mvnw test` et les résultats des tests s'afficheront.

Tests de montée de charge

Les tests de montée de charge seront assurés par l'outil **JMeter**. Une fois l'image Docker du projet créée, il faudra exécuter les tests avec JMeter et analyser les résultats.

Tests fonctionnels

Pour réaliser les tests fonctionnels du projet, nous pourrions nous servir de l'outil **Selenium**. Cet outil permet d'automatiser les tests utilisateurs.

Données de test

Les données de test, pour les tables Utilisateurs, Hôpitaux et Spécialisations ont été créées de toutes pièces elles ne sont donc pas réelles.

Pour la table Spécialités, il s'agit des données de références fournies en début de projet celles-ci sont dans le repository de documentation à l'emplacement : [medheaddoc/Données de référence sur les spécialités NHS.pdf](#)

Un compte utilisateur a été créé afin de tester le fonctionnement du PoC :

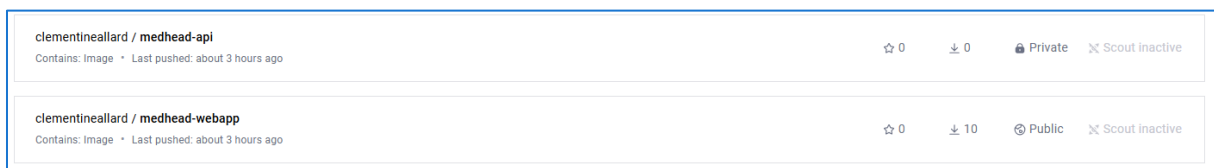
- Email : admin
- Mot de passe : admin

LIVRAISON

Le packaging du projet est exécuté de manière automatique grâce à un workflow **GitHub Actions**, le workflow `CI`. A partir d'un script, à chaque push ou pull sur la branche main **GitHub Actions** :

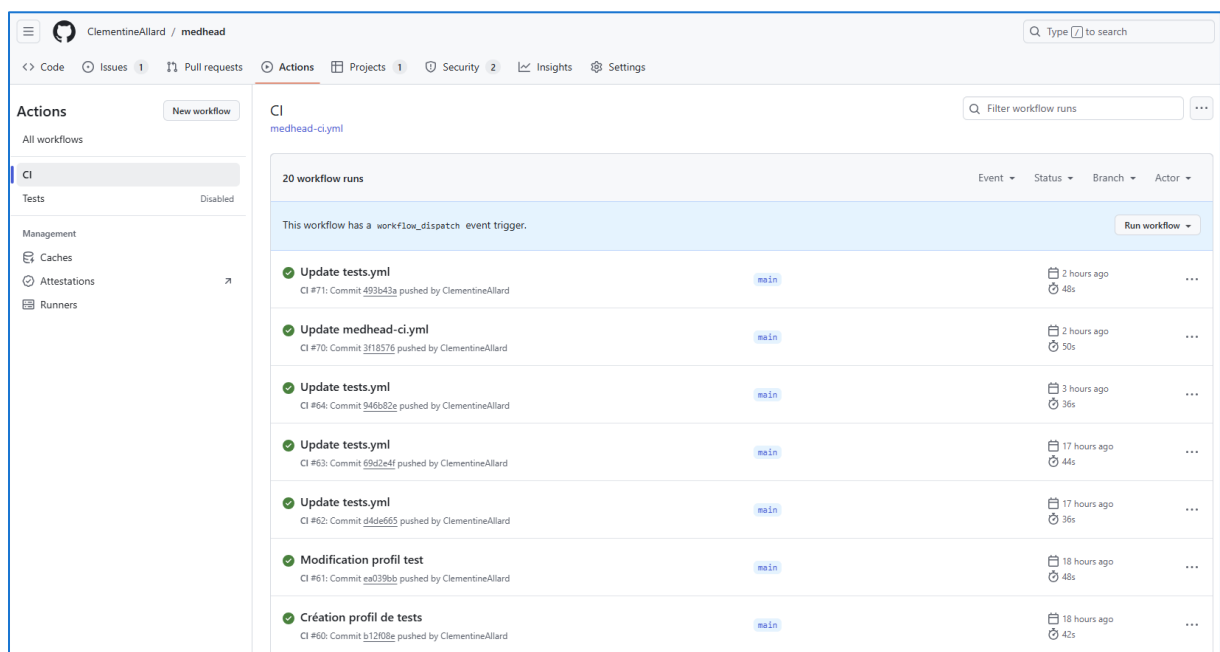
- Lance le packaging de l'api et de l'application web
- Génère les images **Docker**
 - En suivant les instructions décrites dans le **Dockerfile** des deux projets
- Envoie les images à **Docker**

Les images sont donc ensuite présentes sur **Docker**.



Interface des images Docker

Les résultats de l'exécution de workflow se trouvent dans l'onglet **Actions** du projet GitHub.



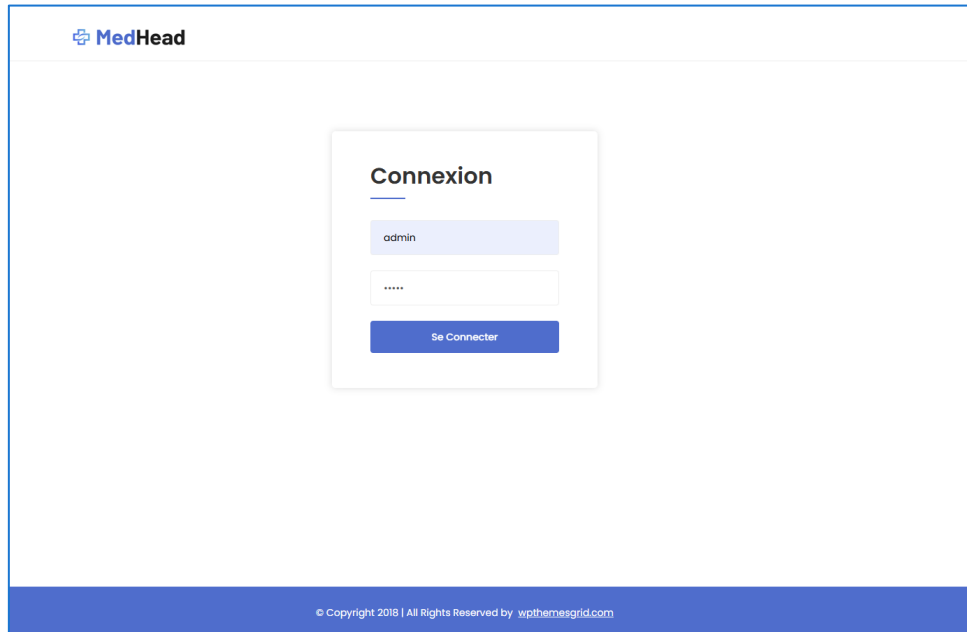
Interface de GitHub Actions

ASPECTS FONCTIONNELS COMPRIS DANS LE POC


Voici un résumé de l'ensemble des fonctionnalités permises par le PoC :

Connexion à la plateforme

- **Prérequis :** aucun
- **Accessible depuis :** l'interface de connexion



- **Données en entrée :**
 - Identifiant : obligatoire
 - Mot de passe : obligatoire
- **Actionnée par :** l'appui sur le bouton *Se Connecter*
- **Résultats possibles :**
 - Accès refusé car mot de passe, identifiant ou clé de cryptage incorrect




Email ou mot de passe incorrect...

Connexion

Se Connecter

© Copyright 2018 | All Rights Reserved by wpthemesgrid.com

- Accès autorisé si mot de passe, identifiant et clé sont corrects



Recherche

Groupe

Spécialité

Rechercher

© Copyright 2018 | All Rights Reserved by wpthemesgrid.com

Recherche d'un hôpital

- **Prérequis** : être connecté en tant qu'utilisateur
- **Accessible depuis** : l'interface de recherche

The screenshot shows the MedHead search interface. At the top left is the MedHead logo. Below it, the title "Recherche" is underlined. The form consists of a text input field labeled "Où ?" for location, two dropdown menus labeled "Groupe" and "Spécialité", and a blue button labeled "Rechercher". The footer contains the copyright notice: "© Copyright 2018 | All Rights Reserved by wpthemesgrid.com".

- **Données en entrée** :
 - Lieu de recherche : obligatoire
 - Groupe de spécialités : obligatoire
 - Sous-spécialités : facultatif
- **Actionnée par** : l'appui sur le bouton *Rechercher*
- **Résultats possibles** :
 - Aucun hôpital ne correspond à la spécialité sélectionnée ou aucun n'a de lits disponibles

This screenshot shows the same MedHead search interface but with specific data entered. The "Où ?" field contains "Toulouse", the "Groupe" dropdown is set to "Psychiatrie", and the "Spécialité" dropdown is set to "Psychiatrie légale". The "Rechercher" button is still present. Below the form, a message states: "Aucun hôpital ne correspond à votre recherche". The footer remains the same: "© Copyright 2018 | All Rights Reserved by wpthemesgrid.com".

- L'hôpital le plus proche ayant la spécialité demandée et des lits encore disponibles est affiché

The screenshot displays the MedHead search interface. At the top, the MedHead logo is visible. Below it, the 'Recherche' (Search) section contains a search bar with 'Toulouse' entered. Below the search bar, there are two dropdown menus: the first is labeled 'Anesthésie' and the second is empty. A blue 'Rechercher' (Search) button is positioned below the dropdowns. The search results section shows 'Hôpital Saint-Vincent' as the top result. Below the hospital name, the address '9 Rue de la République, 31000 Toulouse' and the phone number '05 34 45 67 89' are listed. A blue 'Réserver Un Lit' (Reserve a Bed) button is located at the bottom of the result card. The footer of the page contains the copyright notice: '© Copyright 2018 | All Rights Reserved by wpthemesgrid.com'.

Réservation d'un lit

- **Prérequis** : être connecté et avoir eu un résultat à sa recherche d'hôpital
- **Accessible depuis** : l'interface de recherche avec résultat affiché

The screenshot shows the MedHead search interface. At the top, the MedHead logo is visible. Below it, the 'Recherche' section contains a search bar with 'Toulouse' entered, a dropdown menu with 'Anesthésie' selected, and a 'Rechercher' button. Below the search results, a card for 'Hôpital Saint-Vincent' is displayed, showing its address '9 Rue de la République, 31000 Toulouse' and phone number '05 34 45 67 89'. A 'Réserver Un Lit' button is located at the bottom of the card. The footer of the page contains the copyright notice '© Copyright 2018 | All Rights Reserved by wpthemesgrid.com'.

- **Données en entrée** :
 - Aucune
- **Actionnée par** : l'appui sur le bouton *Réserver Un Lit*
- **Résultat possible** :
 - Lit réservé à l'hôpital demandé, l'hôpital a ainsi un lit de moins disponible

The screenshot shows the MedHead search interface after a successful reservation. A green banner at the top displays the message '✓ Réservation effectuée avec succès : Hôpital Saint-Vincent'. Below the banner, the 'Recherche' section and the 'Hôpital Saint-Vincent' card are still visible. The footer of the page contains the copyright notice '© Copyright 2018 | All Rights Reserved by wpthemesgrid.com'.