Description des Éléments de Base Réutilisables

Projet PoC Sous-Système

Intervention Urgence

Table des matières

1 SOMMAIRE

Ce document représente un document répertoriant tous les éléments de base réutilisables obtenus lors de la conception et de la mise en œuvre.

3 OBJECTIFS

Ce document a pour but de fournir une description complète des éléments de base réutilisable lors de la mise en œuvre et de la conception de la PoC du système d'intervention d'urgence qui permet l'attribution en temps réel de lits d'hôpital en fonction de la pathologie.

4 MODULES DE CONSTRUCTION DE LA PREUVE DE CONCEPT

- **(PoC) Nom du module de construction : Build Automatisé Maven.**
- Fonctionnalité fournie : Permet d'avoir un artefact Maven immédiatement déployable dans un conteneur Spring Boot.

Historique			
Version	**Date**	**Auteur**	**Commentaires**
0.01	19/04/2023	Wiam El yadri	Document sur les éléments de base réutilisables.

Lien vers des exemples d'implémentation ou d'interfaces :

Le repository gitLab du projet MedHead :
https://gitlab.com/medhead/Medheadproject_frontend

Travail supplémentaire pour terminer ce module de construction :

Implémentation sur une plateforme Kubernetes permettant d'automatiser l'orchestration et le déploiement de l'application.

Alignement architectural :

Utilisation du design pattern Facade lors de la conception du PoC afin de ne pas exposer toutes les fonctionnalités des Interfaces JpaRepository et CrudRepository. La conception de ce PoC ne demandait pas d'utiliser toutes les méthodes de ces interfaces.

5 Architecture et communications des microservices

- **Configuration service** : permet de centraliser toutes les configurations nécessaires pour l'ensemble des micros-services
- **Service d'enregistrement** : permet d'enregistrer la localisation de chaque microservice. C'est un annuaire qui publie l'ensemble des informations des micros-services (nom, adresse IP, numéro de port)
- **Service proxy** : permet d'orchestrer l'ensemble des fonctionnalités des micros-services. Chaque microservice avant de démarrer, il envoie une requête http REST vers le *service de Configuration* pour récupérer ces paramètres de configuration et puis il démarre.
  Le microservice va ensuite se connecter au *service d'enregistrement* pour publier son nom, son adresse IP et son numéro de port.

En effet, toutes les requêtes http des applications (web ou mobiles), vient vers le *service proxy* en fournissant le nom du micro-service souhaité. Avec ce dernier le service proxy contact, l'annuaire pour obtenir les informations du microservice (port, adresse IP). Ensuite, le service va envoyer la requête vers le microservice concerné. Le service proxy va ensuite récupérer le résultat pour envoyer au client.

6 Éléments réutilisables

La création de la base de données est un élément réutilisable permettant de faires des opérations CRUD (create, read,update,delete).

Utilisation de GitLab pour l'implémentation de script CI/CD.

Sur le projet nous avions utilisé plusieurs outils et technologies :

intitulé	Technologie/Outils
Base de données	H2/SQL Server
Tests	Junit/Postman
Documentation	Jacoco(Documentation pour les tests)
Backend-Frontend	Maven
	Spring tools
	Bootstrap
	Javascript
	Jpa
	Spring Security
	OpenJDK 17
	Git
	IOS"Système d'exploitation"

6-1 listes des éléments à conserver

- La méthode de calcul de distance, dont deux propositions sont présentées, devront-être éprouvées pour n'en retenir qu'une.
- La Poc pourrait-être la base de la future application dans son intégralité.
- Les tests qui les sollicitent seront en partie à conserver.
- Les requêtes de tries seront intégrables dans la version finale.
- Les tests unitaires le sont tout autant.
- La partie Mockmvc devra être complétée en fonction de l'ajout de BDD.
- Le repository GitLab pourrait servir de base de travail pour l'amendement de la Poc.

7 Description des Microservices

7-1 Microservice gestion des hôpitaux

Ce service permet de rechercher un hôpital en fonction de la spécialisation, la localisation et du nombre de lits disponibles.

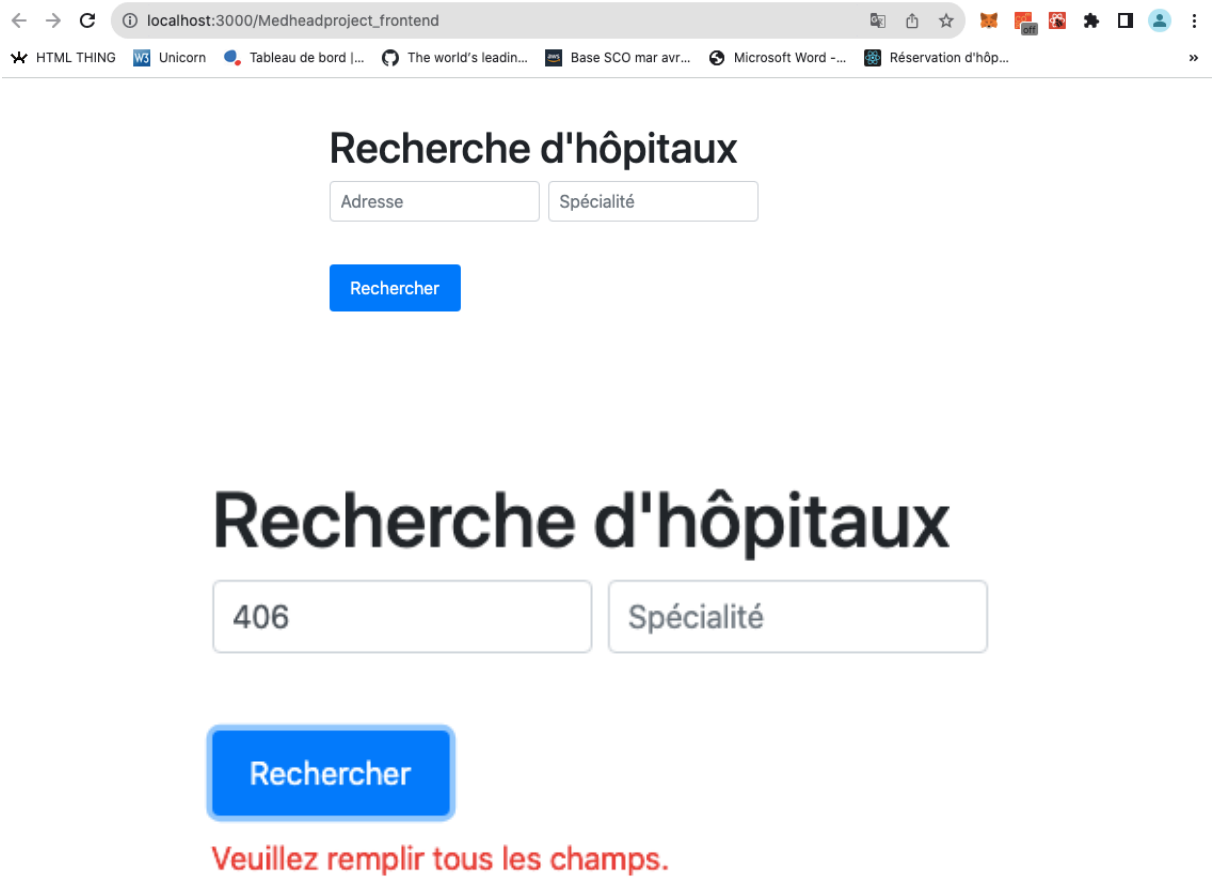
En effet, le personnel médical pourra saisir sur la plateforme sa localisation et doit choisir la spécialité. Et suite aux informations saisies , un hôpital le plus proche de l'adresse saisie doit lui être proposé.

A l’exemple :

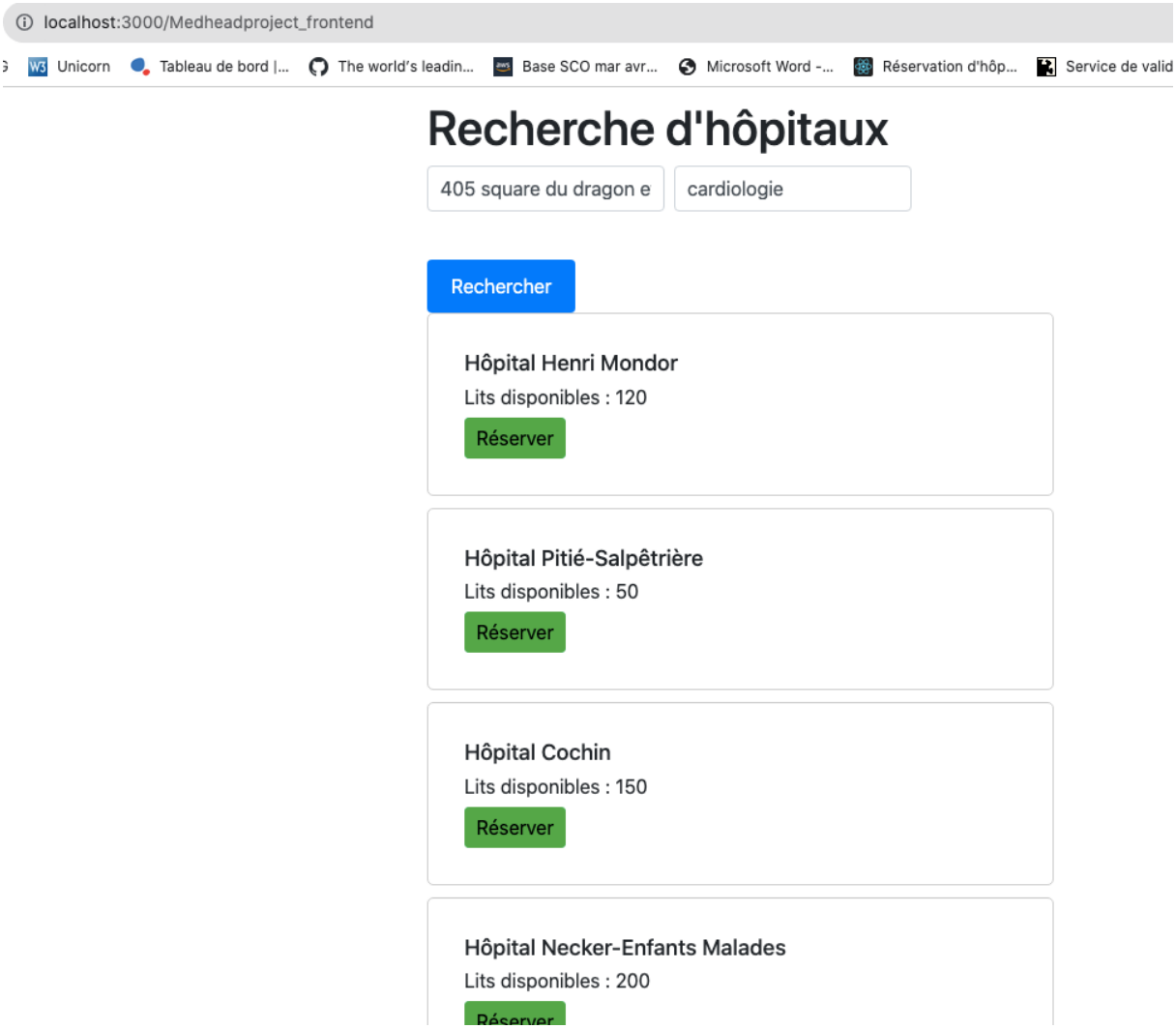
ET un patient nécessitant des soins en cardiologie.
QUAND le personnel sélectionne la spécialisation « cardiologie » ET que l'urgence est localisée A l’ile de France ALORS l'hôpital de Evry va être proposé.

A l’exemple :

Rechercher un l’hôpital



• Proposition de l’hôpital de la ville avec un ordre croissant de plus proche au plus loin avec l’option réservation de lit :



7-2 Microservice réservation de lit

Ce service permet d'attribuer en temps réel de lits d'hôpital en fonction de la pathologie et de l'hôpital. En cas de proposition d'un hôpital, la réservation du lit va être réalisée de manière automatique.

En effet, quand le personnel aura reçu la proposition d'un hôpital, si un lit est disponible et que le personnel choisi de lui réserver un lit, le micro-service va être en mesure de réserver un lit. Un évènement est déclenché automatiquement pour réserver un lit. Le personnel doit saisir le nom et prénom du patient pour réserver le lit en fonction du patient.

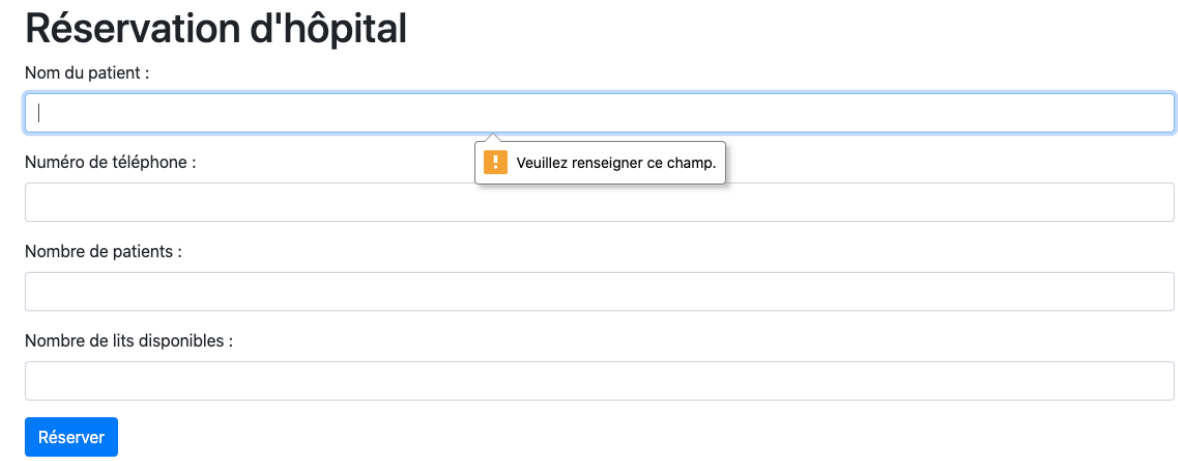
Si l'hôpital est réservé, le micro-service va ensuite réduire le nombre de lits disponibles dans l'hôpital ou la réservation a été effectuée (en fonction de la spécialisation).

Le service réservation de lit appelle et utilise le service gestion des hôpitaux afin de pouvoir réduire le nombre de lits disponibles au moment de la réservation de lit.

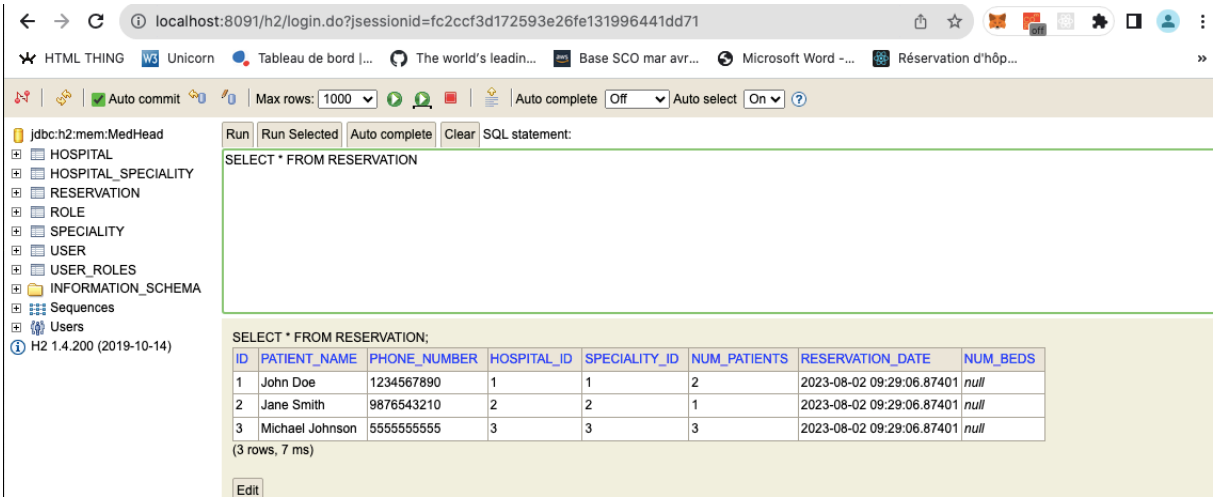
A l'exemple :

Etant donné qu'il y a un hôpital qui est proposé
Quand on saisit le nom et prénom du patient
Et qu'on on clique sur le bouton réserver
Alors le système déclenche un évènement pour réserver un lit en fonction du nom du patient.

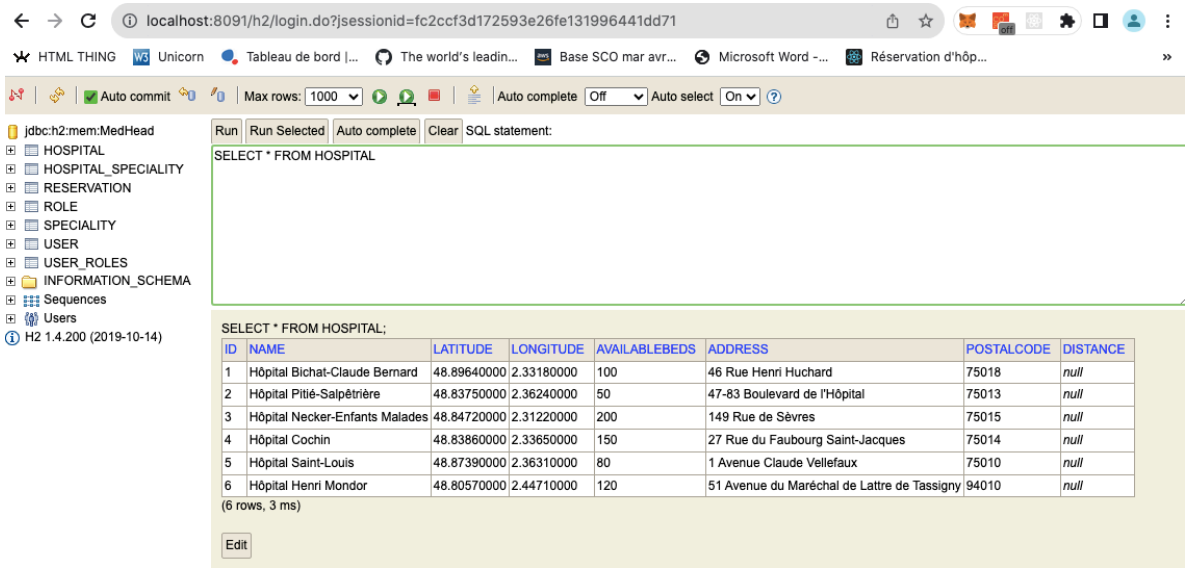
Extrait de l'interface utilisateur réservation de lit : « avec les champs obligatoires à remplir »



La table de réservation sans la nouvelle réservation enregistrée :



La table des hôpitaux avec le nombre de lits :

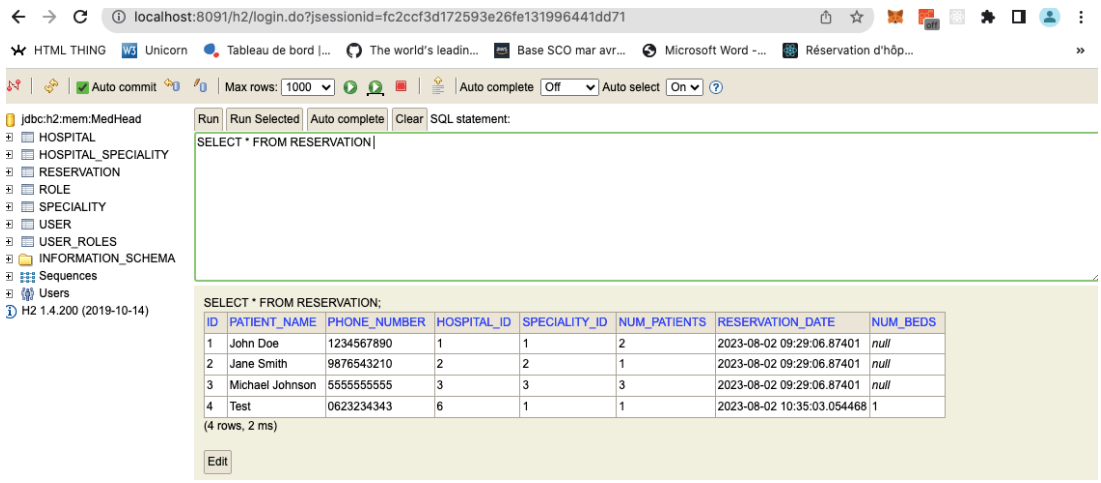


Message sur la console pour la réservation réalisée

```
Hibernate: select speciality0_.id as id1_5_, speciality0_.name as name2_5_ from speciality
speciality0_ where speciality0_.name=?
Patient Name: Test
Phone Number: 0623234343
Num Patients: 1
Num Beds: 1
Hospital ID: 6
Speciality: 1
Hibernate: select hospital0_.id as id1_2_0_, hospital0_.address as address2_2_0_,
hospital0_.availablebeds as availabl3_2_0_, hospital0_.distance as distance4_2_0_,
hospital0_.latitude as latitude5_2_0_, hospital0_.longitude as longitud6_2_0_, hospital0_.name as
name7_2_0_, hospital0_.postalcode as postalco8_2_0_ from hospital hospital0_ where hospital0_.id=?
Hibernate: insert into reservation (id, hospital_id, num_beds, num_patients, patient_name,
phone_number, reservation_date, speciality_id) values (default, ?, ?, ?, ?, ?, ?, ?)
Hibernate: update hospital set address=?, availablebeds=?, distance=?, latitude=?, longitude=?,
name=?, postalcode=? where id=?
```

Message sur la page de réservation indiqué que la réussite de la réservation

La table de réservation après le remplissage de formulaire et la réussite de la réservation
Réservation automatique :
Update la table Réservation avec le nombre choisie de lits.



jdbc:h2:mem:MedHead
HOSPITAL
HOSPITAL_SPECIALITY
RESERVATION
ROLE
SPECIALITY
USER
USER_ROLES
INFORMATION_SCHEMA
Sequences
Users
H2 1.4.200 (2019-10-14)

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM RESERVATION
```

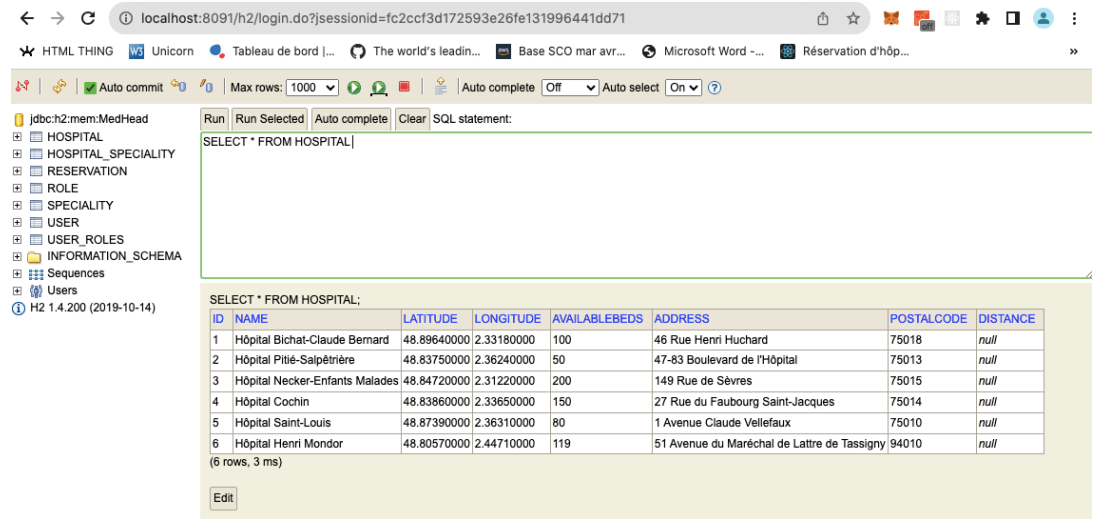
SELECT * FROM RESERVATION;

ID	PATIENT_NAME	PHONE_NUMBER	HOSPITAL_ID	SPECIALITY_ID	NUM_PATIENTS	RESERVATION_DATE	NUM_BEDS
1	John Doe	1234567890	1	1	2	2023-08-02 09:29:06.87401	*null*
2	Jane Smith	9876543210	2	2	1	2023-08-02 09:29:06.87401	*null*
3	Michael Johnson	5555555555	3	3	3	2023-08-02 09:29:06.87401	*null*
4	Test	0623234343	6	1	1	2023-08-02 10:35:03.054468	1

(4 rows, 2 ms)

Edit

la table de l'hôpital réservé on voit le nombre de lit à diminuer de nombre de lit choisie lors de la réservation



jdbc:h2:mem:MedHead
HOSPITAL
HOSPITAL_SPECIALITY
RESERVATION
ROLE
SPECIALITY
USER
USER_ROLES
INFORMATION_SCHEMA
Sequences
Users
H2 1.4.200 (2019-10-14)

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM HOSPITAL
```

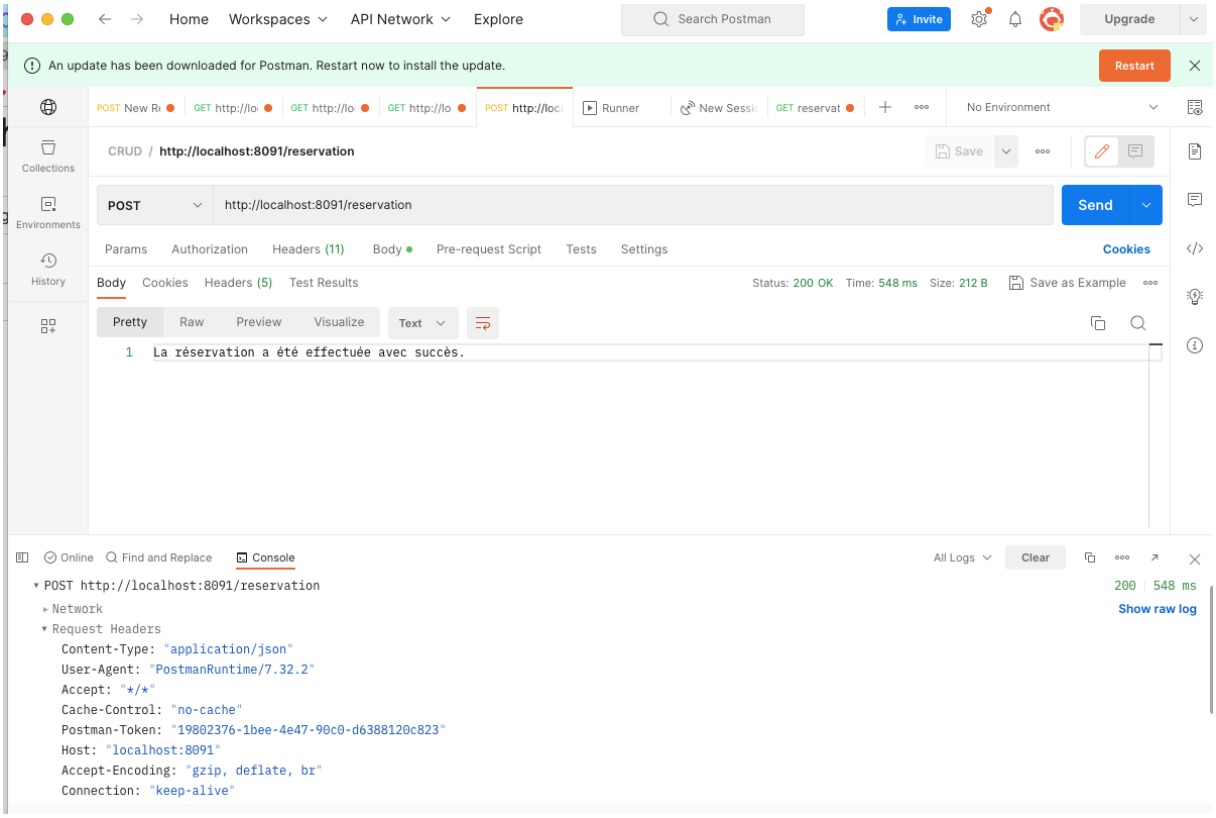
SELECT * FROM HOSPITAL;

ID	NAME	LATITUDE	LONGITUDE	AVAILABLEBEDS	ADDRESS	POSTALCODE	DISTANCE
1	Hôpital Bichat-Claude Bernard	48.89640000	2.33180000	100	46 Rue Henri Huchard	75018	*null*
2	Hôpital Pitié-Salpêtrière	48.83750000	2.36240000	50	47-83 Boulevard de l'Hôpital	75013	*null*
3	Hôpital Necker-Enfants Malades	48.84720000	2.31220000	200	149 Rue de Sèvres	75015	*null*
4	Hôpital Cochin	48.83860000	2.33650000	150	27 Rue du Faubourg Saint-Jacques	75014	*null*
5	Hôpital Saint-Louis	48.87390000	2.36310000	80	1 Avenue Claude Vellefaux	75010	*null*
6	Hôpital Henri Mondor	48.80570000	2.44710000	119	51 Avenue du Maréchal de Lattre de Tassigny	94010	*null*

(6 rows, 3 ms)

Edit

Message de succès de Réservation : testé la requête sur Postman



7-3 Service API Gateway (proxy)

Ce service permet d'orchestrer l'ensemble des fonctionnalités des micros-services. En effet, si l'application envoie des requêtes http, au lieu d'aller directement demander les réponses aux différents micro-services (gestion hôpitaux et réservation de lit), il va d'abord interroger l'API Gateway (proxy) puis ce dernier va interroger par la suite le service d'enregistrement qui va lui fournir les informations de configuration du service souhaité par la demande (exemple le numéro de port).

En effet, c'est le point d'entrée unique pour les API et micro-services. En effet, elle permet d'agréger différents micro-services. L'agrégation des requêtes sera faite directement au travers de l'API Gateway, cela permet de réduire la charge réseau et la multiplication des appels.

Explication :

Le demandeur de service : API Gatewey
L'annuaire de service : le service d'enregistrement

Fournisseur de service : est les microsservices (gestion hôpitaux et réservation de lit et Authentification)

Requete	Endpoint	Détails
Rechercher un Hopital	GET/specialisations	Obtenir une liste des spécialisations
	GET/hopitals/{ codeSpecialisation }/{ localisation}	Obtenir un hôpital en fonction de la spécialisation, la localisation et du nombre de lit disponible
	PUT/nombreLit/{codeHopital}/{codeSpecialisation}	Réduire le nombre de lit de l'hôpital recherché
Réservation d'un lit	PUT/reservationLit/{codeHopital}/{codeSpecialisation}/ {nomPatient}/{prenomPatient}	Réserver un lit d'hôpital

7-4 Service d'enregistrement

Il permet d'enregistrer la localisation de chaque microservice.

C'est un annuaire qui publie l'ensemble des informations des micros-services (nom, adresse IP, numéro de port)

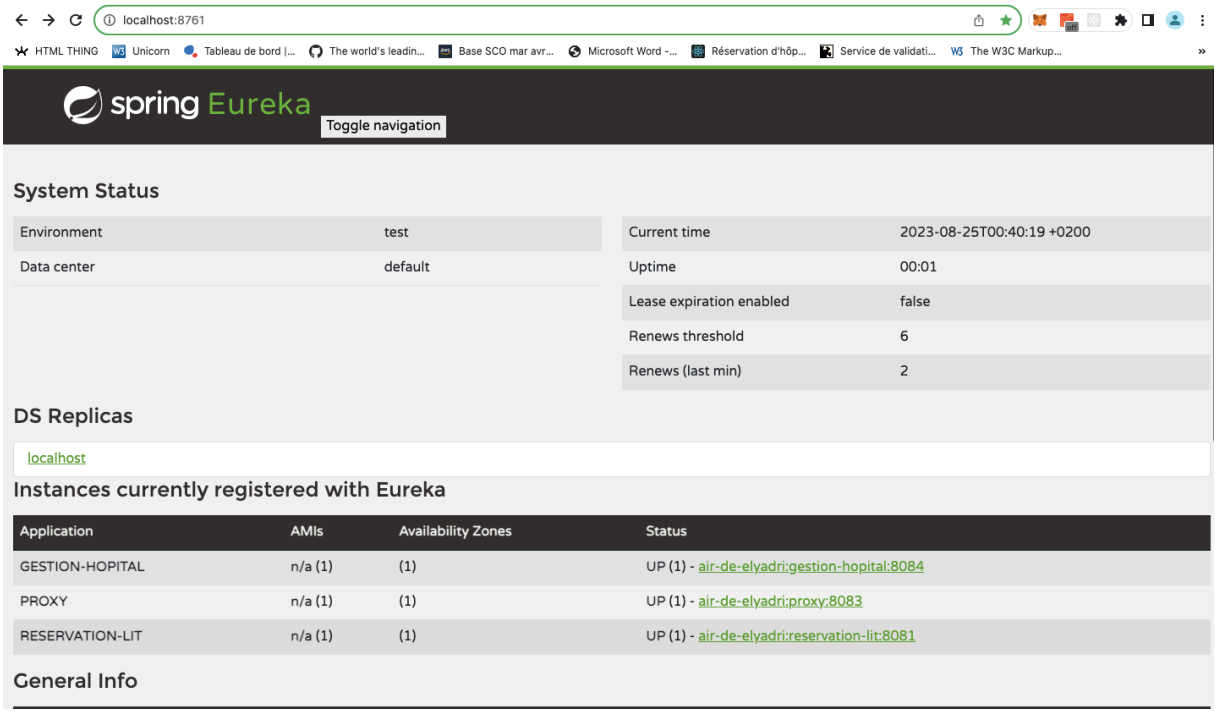
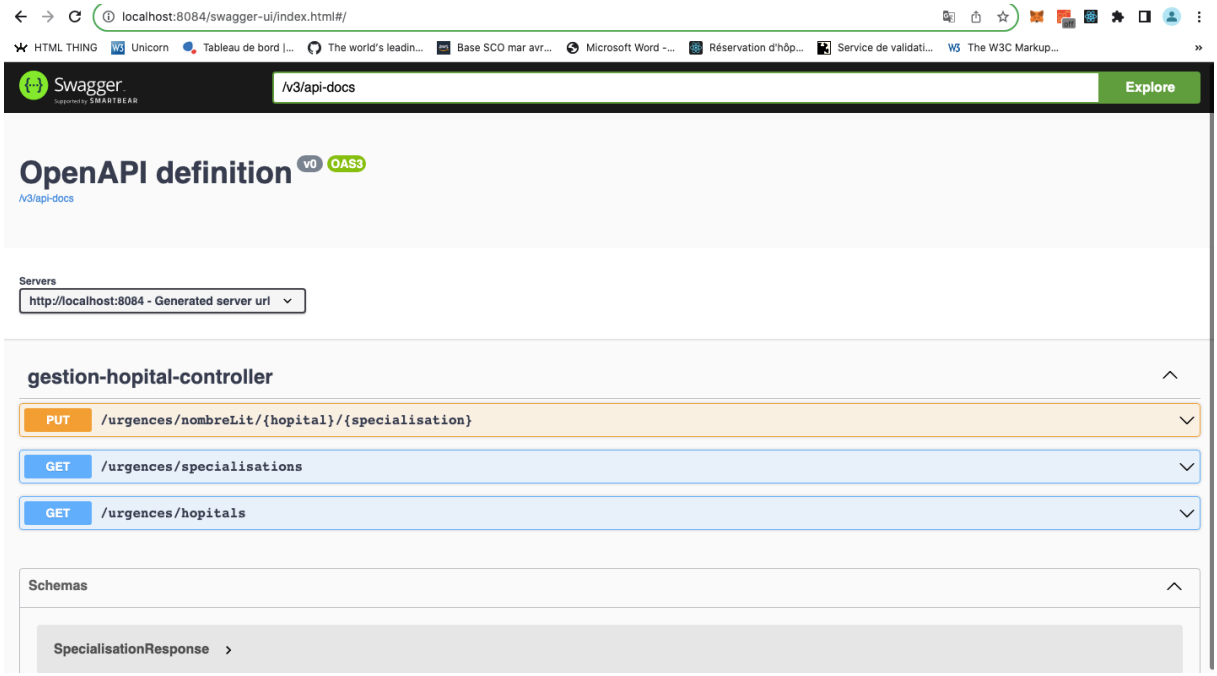


Figure N°1 représentant l'interface Eureka

Utilisation de la documentation Swagger :



les avantages clés de l'utilisation de Swagger pour documenter vos API :

Documentation Interactive : Swagger génère une documentation interactive pour votre API, qui est accessible via une interface utilisateur conviviale. Cette documentation inclut des informations sur les endpoints, les paramètres, les types de données, les réponses attendues, les codes d'état, etc.

Facilité de Conception : En utilisant Swagger, vous pouvez concevoir vos endpoints d'API en utilisant des annotations Java ou des annotations spécifiques à d'autres langages. Cela facilite la spécification des paramètres d'entrée, des réponses attendues et des autres informations pertinentes.

Génération de Clients : Swagger permet de générer automatiquement des clients pour différents langages de programmation à partir de la documentation de l'API. Cela facilite l'intégration des clients de l'API dans différentes applications.

Validation et Tests : pour tester notre API directement à partir de l'interface utilisateur Swagger UI. Cela nous permet de vérifier rapidement si vos endpoints fonctionnent correctement.

Collaboration : Swagger fournit une spécification basée sur le format OpenAPI, qui peut être partagée avec d'autres membres de l'équipe, des partenaires ou des développeurs tiers. Cela permet une meilleure collaboration autour de l'API.

Meilleure Communication : En fournissant une documentation claire et interactive, Swagger améliore la communication entre les développeurs backend et frontend, ainsi qu'avec les clients ou les utilisateurs de l'API.

8. AXES D'AMELIORATIONS

Mise en place d'une API externe « L'API Distance Matrix », ou utilisé une api externe fournie par le gouvernement.

- Elle proposera l'hôpital le plus proche de la localisation : il va pouvoir comparer les distances des hôpitaux qui sont autour de la localisation et proposé l'hôpital le plus proche
- Elle fournira la distance et la durée des déplacements en temps réel

L'api proposée pour ce projet : https://adresse.data.gouv.fr/api-doc/adresse