Table des matières

Cadre du projet

Contexte

Dans le cadre de mes travaux de veille technologique, je me suis fixé un objectif avec mon manager de créer une petite application en utilisant les nouvelles méthodes et technologies de développement. J'ai donc d'aborder l'architecture micro-service, très utilisée dans les entreprises.

Présentation

J'ai développé un annuaire privé permettant après de s'authentifier de :

- Consulter la liste des contacts ainsi que le détail des informations d'un contact choisi
- Si on est Admin :
    - Ajouter un contact
    - Modifier un contact
    - Supprimer un contact

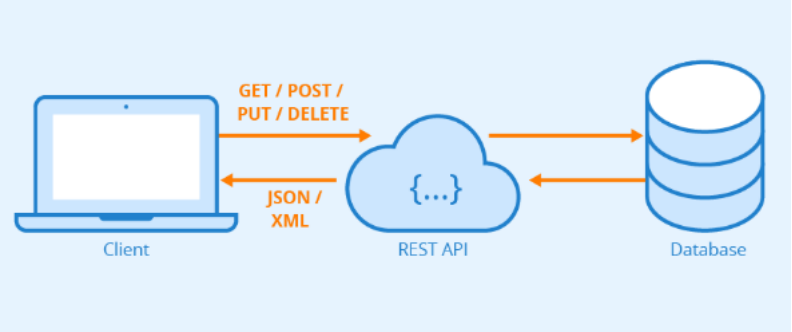
Réalisations

Architecture

L'application est composée de grosses parties :

- Un client ou la partie Front
- Un serveur ou la partie Back

Ces deux parties communiquent via l'architecture REST.



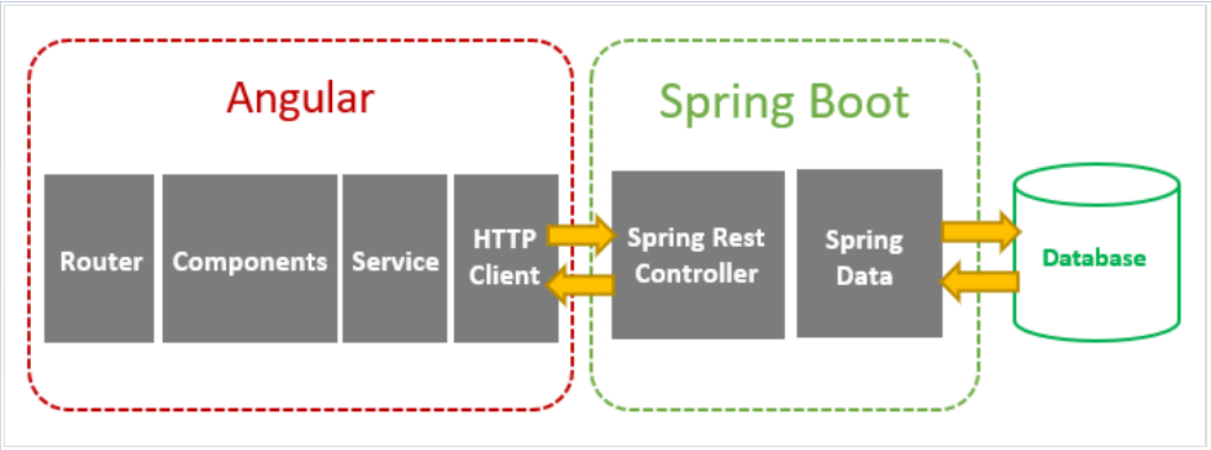
REST (Representational State Transfer) ou RESTful est un style d'architecture permettant de construire des applications (Web, Intranet, Web Service). Il s'agit d'un ensemble de conventions et de bonnes pratiques à respecter et non d'une technologie à part entière. L'architecture REST utilise les spécifications originelles du **protocole HTTP**, plutôt que de réinventer une surcouche (comme le font SOAP ou XML-RPC par exemple).

- Règle n°1 : l'URI comme identifiant des ressources
- Règle n°2 : les verbes HTTP comme identifiant des opérations
- Règle n°3 : les réponses HTTP comme représentation des ressources
- Règle n°4 : les liens comme relation entre ressources
- Règle n°5 : un paramètre comme jeton d'authentification

Pour plus d'informations sur cette architecture, veuillez visitez le site suivant : https://blog.nicolashachet.com/developpement-php/larchitecture-rest-expliquee-en-5-regles/

Technologies utilisées

Ci-dessous un schéma qui pourra simplifier l'architecture ainsi que les technologies utilisées dans chaque partie du projet.



La partie encadrée en rouge est celle du **front**, elle communique avec un **micro service back end** développé en utilisant le framework Spring ;

Client (front)

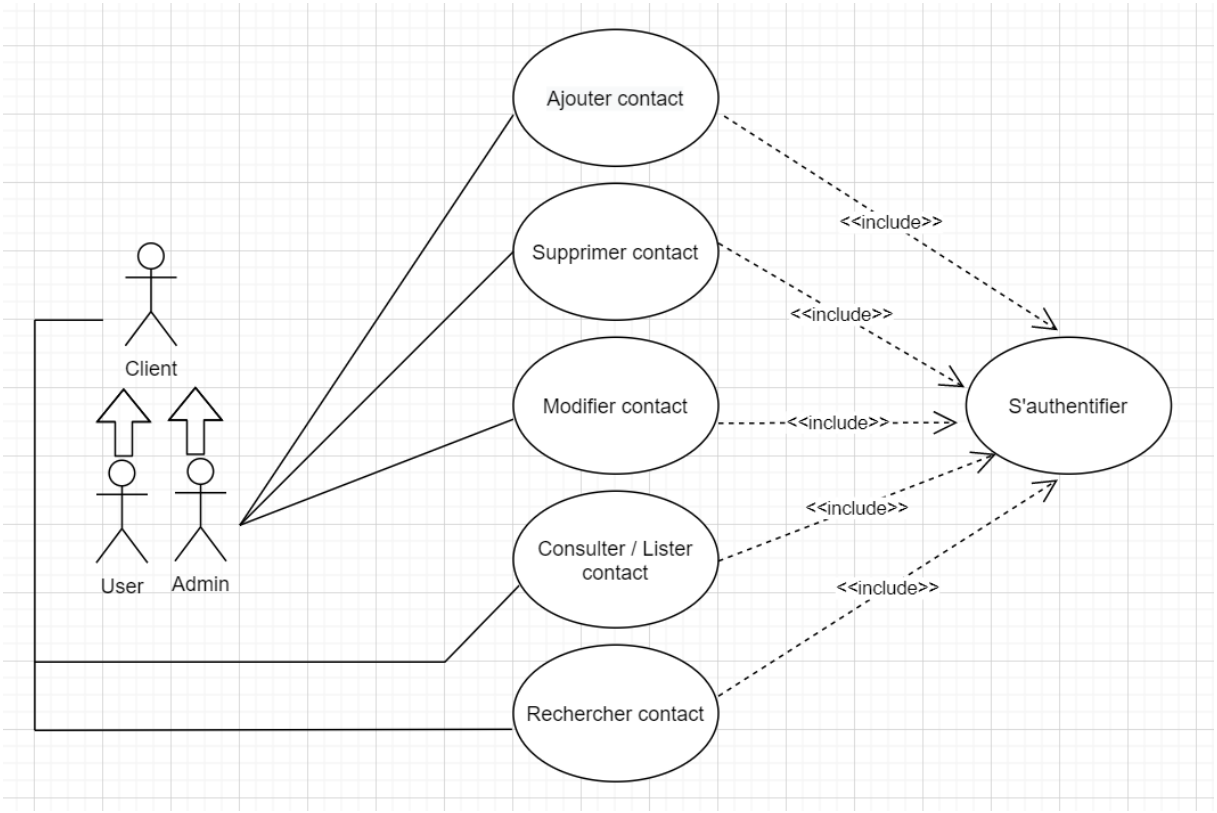
A été développé essentiellement avec Angular est ses librairies. Je montais à chaque fois de version d'Angular, j'ai commencé par Angular 7 et là j'ai mis en place la version 10.

J'ai beaucoup utilisé la bibliothèque **material** d'Angular : **Material** est un module d'Angular, c'est un module d'intégration qui permet d'obtenir facilement une interface responsive harmonieuse et unie dans le style 'flat' de Google. Il offre plusieurs composants décorés comme :

- De jolis boutons, des listes de sélection, des icônes…
- Des datepicker
- Des boites de dialogues
- Des tables de données super bien présentées

Serveur (rest api)

Comme indiqué précédemment, il s'agit s'un micro service Spring offrant aux clients la possibilité de gérer des contacts privés d'une manière sécurisée. Il y'a en effet deux types d'utilisateurs, un administrateur ayant tous les droits sur l'application et un utilisateur normal avec des droits restreints. Pour mieux comprendre le fonctionnement de ce petit back-end, je vous mets à la page suivante un diagramme de cas d'utilisation citant ses fonctionnalités.



Implémentation

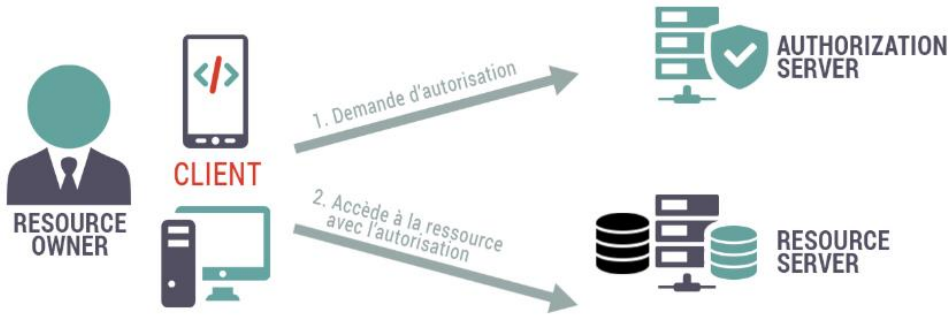
Pour implémenter cette partie, j'ai utilisé le framework Spring Boot, Java 8 et l'environnement de développement éclipse.

Sécurité

Afin de sécuriser mon backend, j'ai mis en place le framework Spring Security en implémentant le protocole OAuth2.

OAuth2 repose sur des échanges entre 4 acteurs. L'utilisateur, ici nommé Resource Owner, est capable d'accorder l'accès à la ressource pour une application nommée Client.

L'Authorization Server occupe le rôle central au sein du protocole, il est chargé d'authentifier le Resource Owner et de délivrer son autorisation sous la forme d'un jeton appelé access token. Le Resource Server quant à lui correspond au serveur où sont stockées les ressources protégées.



Pour plus de détails sur ce sujet, vous pouvez visiter l'URL suivant : https://nexworld.fr/securiser-api-oauth2/

Pour sécuriser les échanges de mot de passe ainsi que leur sauvegarde en BD, j'ai dû les chiffrer en utilisant l'algorithme de hachage BCrypt. Pour tester et avoir des informations sur cet algorithme, merci d'accéder à l'URL suivant : https://md5decrypt.net/Bcrypt/

BD

Pour gérer les échanges avec la base de données, j'ai utilisé Spring Data qui facilite la communication avec plusieurs types de BD (MySql dans ce projet). Pour avoir plus d'informations sur cette couche du projet, vous pouvez visiter cet URI : https://blog.soat.fr/2014/02/spring-data-une-autre-facon-dacceder-aux-donnees/

Conteneurisation

Docker ; docker file et docker compose

Versionning

Git et Github

Fonctionnement

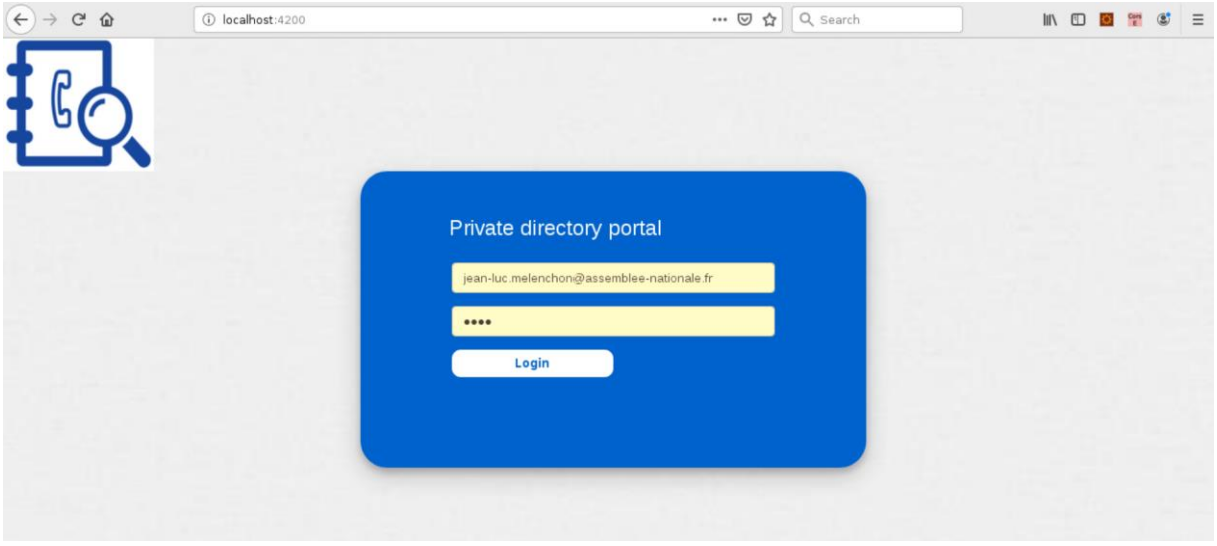
Donnés utilisés

Afin d'alimenter mon annuaire privé, j'ai choisi d'utiliser de le charger avec les données des députés français élus en 2017. Ces informations sont téléchargeables sous plusieurs formats (csv, xml...) sur le site de l'assemblé national suivant : http://data.assemblee-nationale.fr/acteurs/deputes-en-exercice

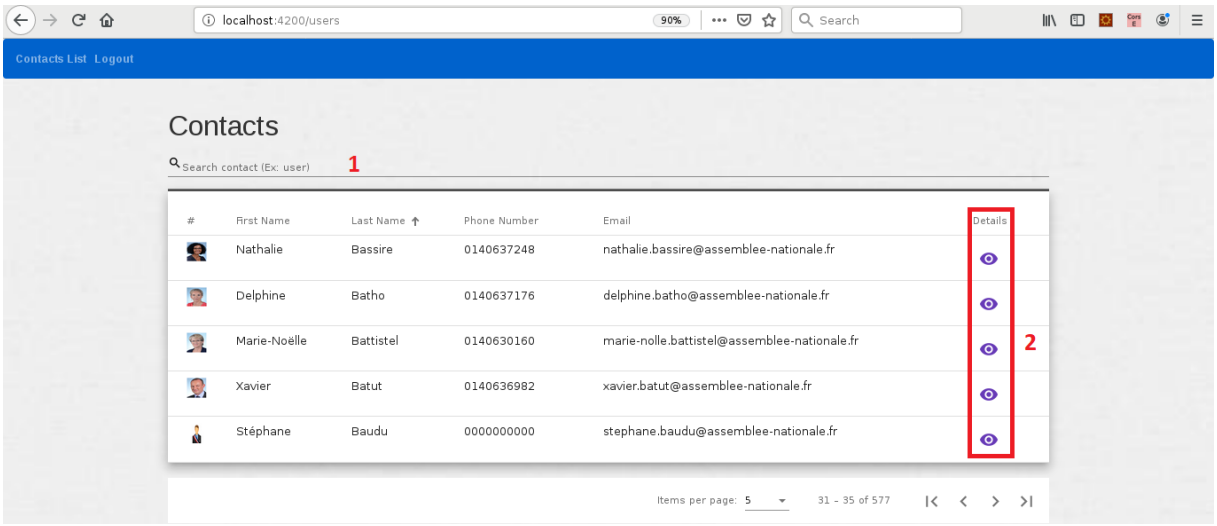
Exemples de cas d'utilisation

Authentification

En allant sur notre application, On se trouve directement sur la page d'authentification suivante :

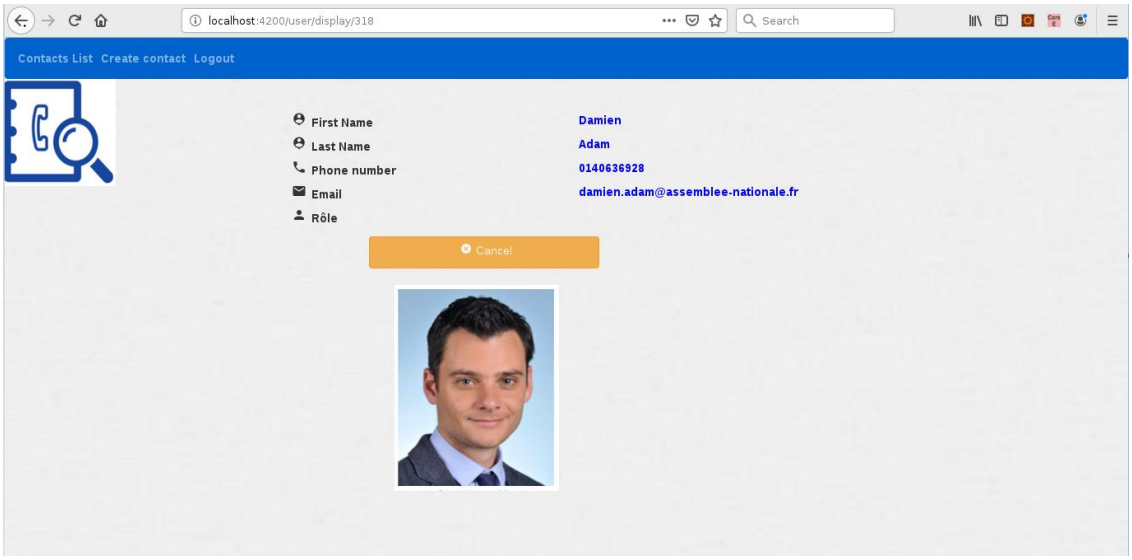


En saisissant un bon User/Password, le système identifiera le rôle que l'utilisateur en question a sur l'application (simple utilisateur ou administrateur). Un simple utilisateur aura l'accès suivant :

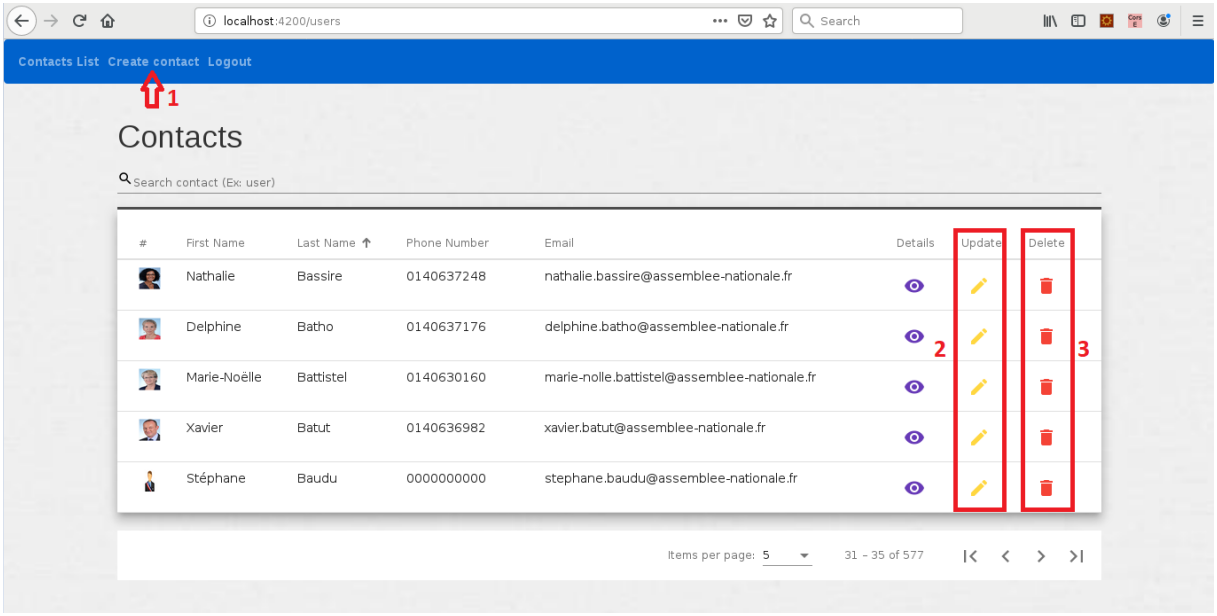


En ayant ce rôle, On peut :

- Rechercher un contact présent dans notre annuaire via la zone de recherche portant le numéro 1 dans la capture ci-dessus.
- Une fois ce contact trouvé, l'utilisateur pourra voir ses informations détaillés en appuyant sur le l'icône de visionnage numéro 2. Voici le résultat :



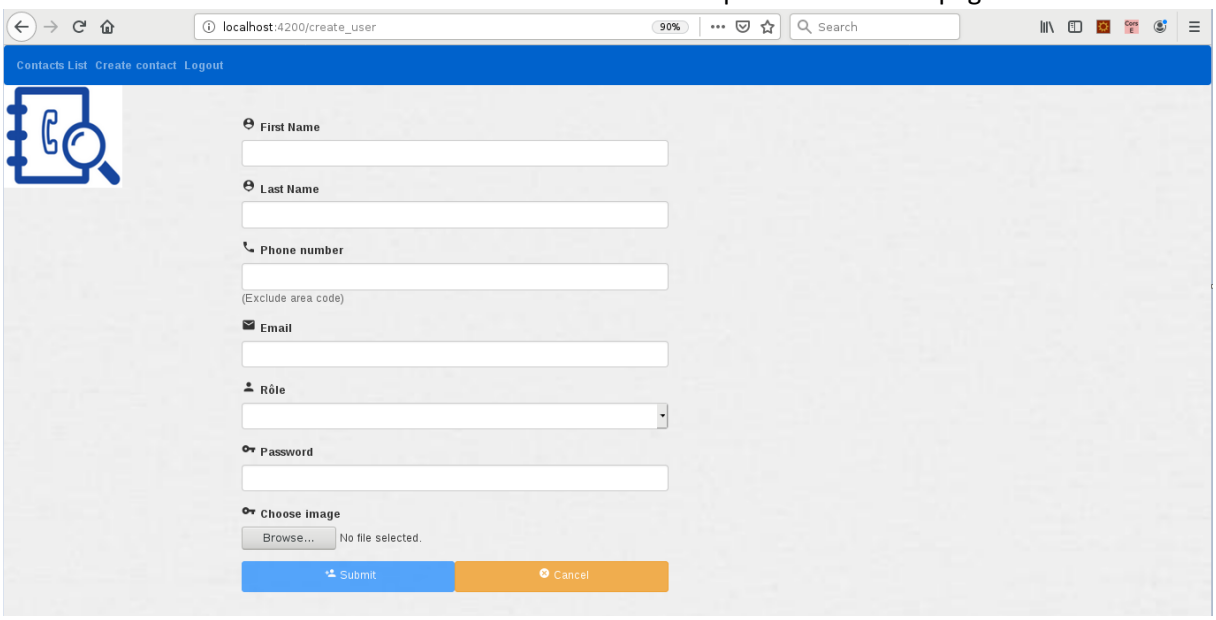
Le rôle administrateur offre logiquement plus de droits sur l'application, On a en effet l'accès suivant avec ce rôle :



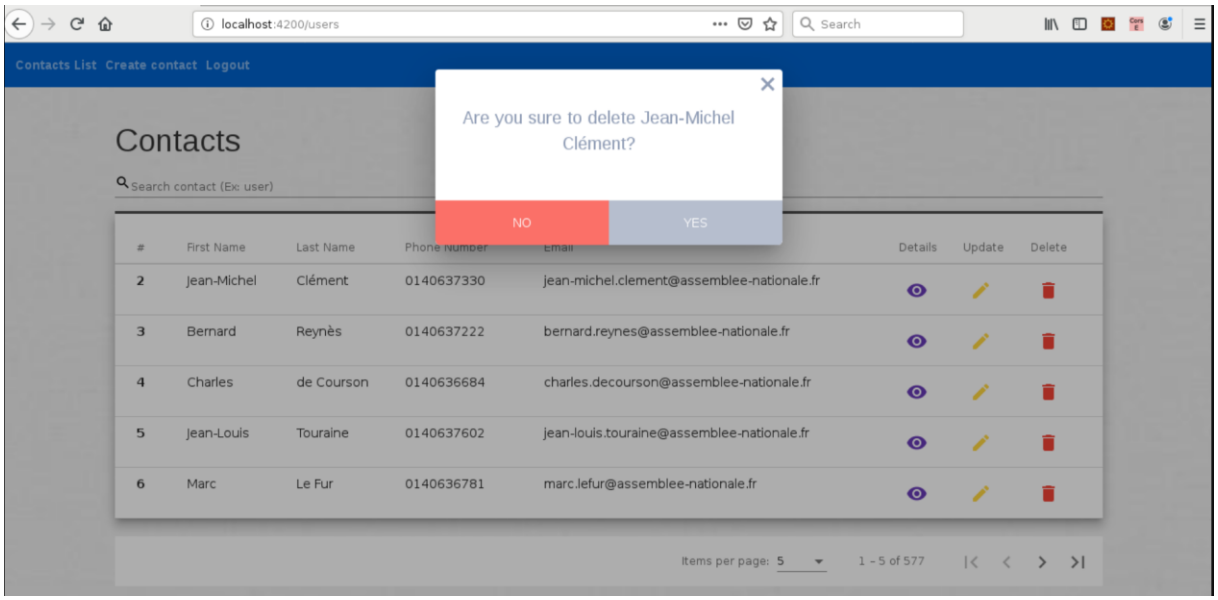
Ajout / Création / Suppression d'un contact

Comme la montre la capturé dans la partie suivante, en étant Administrateur On peut :

- Créer de nouveaux contacts en utilisant le menu 1 de l'écran qui nous mène à la page suivante :



- Mettre à jour les information d'un contact donné via le crayon 2 de l'écran
- Supprimer un contact via l'icône 'poubelle' portant le numéro 3. Le système nous demande ainsi de confirmer la suppression avant de l'effectuer (voir image ci-dessous)



Perspectives

Comme perspectives à ce travail, On peut bien imaginer les améliorations/propositions suivantes :

- Ajouter d'autres données personnelles d'un contact
- Changer le type de données alimentant l'application, une entreprise par exemple pourra bien l'utiliser comme annuaire interne regroupant ses employés
- Interactions entre les différentes personnes de l'annuaire (s'échanger des messages, des mails…).

Annexe

Dans cette partie, vous trouverez les mots clés de tous les technologies/langages utilisés :

1. Back-end
    a. Java 8
    b. Spring Boot
    c. Spring Core
    d. Spring Security
    e. Maven
    f. Oauth/jwt
    g. REST
    h. Mysql
    i. Docker
    j. Swagger
    k. Eclipse
    l. Git

2. Front-end
    a. Angular 7 et 10
    b. Bibliothèque material (Angular)
    c. BootStrap 4.4.1
    d. CSS
    e. Git

Conclusion

Ce travail m'a été très utile il m'a permis en effet de:

- Mettre à jour mes connaissances sur les technologies Spring et Angular en utilisant des versions récentes
- Mettre en pratique mes connaissances sur la conteneurisation en manipulant le logiciel Docker sur mon back-end
- Découvrir et travailler avec de nouvelles technologies très sollicitées sur le marché de travail