Bonjour

je te prie de trouver ci-après le mini projet à réaliser

**Objectifs**

L'objectif du mini-projet est de développer les apis suivantes:

1. Génération d'utilisateurs (us-01)

method: GET

url: /api/users/generate

content-type: application/json

secured: no

parameters:

- count: number

Cet premier endpoint REST permet de générer un fichier json contenant count éléments en respectant la structure suivante:

[{

  "firstName": "string",

 "lastName": "string",

 "birthDate": "date",

 "city": "ville",

 "country": "code iso2",

 "avatar": "url d'une image",

 "company": "string",

 "jobPosition": "string",

 "mobile": "numéro de téléphone",

 "username": "identifiant de connexion",

 "email": "adresse email",

 "password": "mot de passe alétoire entre 6 et 10 caractères",

 "role": "admin ou user"

}]

Tous les champs de ce fichier JSON doivent être générés de façon à avoir des résultants vraisemblables en utilisant une librairie Java appropriée. Les contenus tels que "example, test, etc" ne sont pas recevables.

Le champs role doit etre généré en choisissant parmi les deux valeurs user ou admin

En mettant l'URL de cette API dans le navigateur (ex: <http://localhost:9090/api/users/generate?count=100> le téléchargement d'un fichier JSON doit être déclenché. Le JSON ne doit pas être affiché sous forme texte dans le navigateur web. (download file text)

1. Upload du fichier utilisateurs et création des utilisateurs en base de données (us-02)

method: POST

url: /api/users/batch

content-type: multipart/form-data

secured: no

parameters:

- file: multipart-file

Ce second endpoint permet d'uploader le fichier JSON généré en #1. Le fichier doit être importé en base de données en vérifiant les doublons sur l'adresse email et le username (uniques). Une réponse JSON doit être retournée pour résumer le nombre d'enregistrements total, importés avec succès et non importés. Avant l'enregistrement en base de données, le mot de passe doit être encodé et non stocké en clair.

1. Connexion utilisateur + génération JWT (us-03)

method: POST

url: /api/auth

content-type: application/json

request-body:

- username: string

- password: string

Ce 3e endpoint permet d'authentifier un utilisateur et de générer un token JWT valide. La réponse est un JSON respectant la structure suivante:

{ "accessToken": "valeur du jeton JWT" }

Le JWT généré doit contenir l'email de l'utilisateur. Pour s'authentifier, il est possible de renseigner dans le champs username soit le username soit l'email contenu dans le fichier JSON importé.

1. Consultation de mon profil (us-04)

method: GET

url: /api/users/me

secured: yes

Ce 4e endpoint permet d'utiliser le jeton JWT pour accéder de façon sécurisée au profil de l'utilisateur associé.

1. Consultation de profil (us-05)

method: GET

url: /api/users/{username}

sécurisée: oui

Lorsque le JWT utilisé correspond à un utilisateur ayant le rôle admin, il est possible de consulter le profil de n'importe quel autre utilisateur. Un utilisateur n'ayant pas le rôle admin ne peut donc pas accéder au profil d'un autre utilisateur

**Contraintes**

Il vous est demandé de :

* Réaliser ce projet en utilisant Gradle ou Maven, Java8 et la dernière version du framework SpringBoot
* La base de données utilisée doit être une H2
* L'application doit démarrer sur le port 9090
* L'application doit démarrer sans aucune configuration manuelle
* Le projet doit exposer un endpoint Swagger en utilisant <https://springdoc.org/>
* Tous les endpoints doivent être testables depuis l'interface Swagger(Postman & curl)
* Des tests unitaires sont fortement souhaités pour les fonctionnalités demandées(junit)

Repository

Controller

Service