Conception & Développement d'Applications et Services Web/Serveur

TD7 : authentification, token JWT

L'objectif du TD est de construire un service d'authentification, qui pourra être utilisé pour authentifier les clients dans les applications front office et pour authentifier les utilisateurs (staff point de vente) des applications de back office.

contexte : On met en place un service d'authentification basé sur l'utilisation de token JWT. Ce service est capable de délivrer des tokens sur la base de credentials qui lui sont transmis (authenticate) et de vérifier la validité d'un token JWT qui lui est soumis (validate).

Ce service est une application indépendante qui expose sa propre API et utilise sa propre base de données. Cette application est la seule à détenir les mots de passe des utilisateurs.

Sur un plan pratique, ce service est implanté dans deux conteneurs docker : un conteneur docker correspondant à l'API, et un conteneur hébergeant la base de données. Il est donc nécessaire de compléter la configuration docker-compose.yml pour ajouter ce service.

1. Mise en place du service, authentification et création d'un token JWT

Mettre en place le service d'authentification dans la composition docker. Le service API doit répondre sur l'url `api.auth.local` et utiliser sa propre base de données. La structure et le contenu de cette base vous est fourni, elle ne contient qu'une seule table `User`.

Programmer la route permettant l'authentification auprès de ce service de d'authentification. On utilise une authentification HTTP Basic. A titre d'exemple, un contrôleur proposant une méthode d'authentification vous est fourni, vous pourrez l'adapter à votre contexte.

Ainsi, la requête d'authentification aura la forme suivante :

```
POST /signin HTT/1.1
Authorization: Basic bWljaGVsOm1pY2hlbA==
```

En cas d'absence du header `Authorization` ou de credentials invalides, la réponse contient un code 401 :

```
HTTP/1.1 401 Unauthorized


Content-Type: application/json

{
    "type": "error",
    "error": 401,
    "message": "no authorization header present"
}
```

En cas de réussite de l'authentification, la réponse contient un access token JWT et un refresh token opaque. Le token JWT contient le profil de l'utilisateur pour lequel l'authentification est accordée :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access-token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJpc3MiOiJodHRwOlw
vXC9hcGkubGJzLmxvY2FsXC9hdXRoIiwiYXVkIjoiaHR0cDpcL1wvYXBpLmxicy5sb2NhbCIsImlhd
```

```
CI6MTUxMzY3MTAwMCwiZXhwIjoxNTEzNjc0NjAwLCJjaWQiOjF9.EtE4iY2XIGf_V0Ai9g62D3XU35
IFgSJr6n8ja9jOLr7JCdqxG-oTosWwruGT028oFm_6pwQzUHwYBCtyZx4AGQ" ,
  "refresh-token": "A675E34FA7B43109FEC43218FEDD56"

}
```

2. contrôle de la validité d'un token JWT

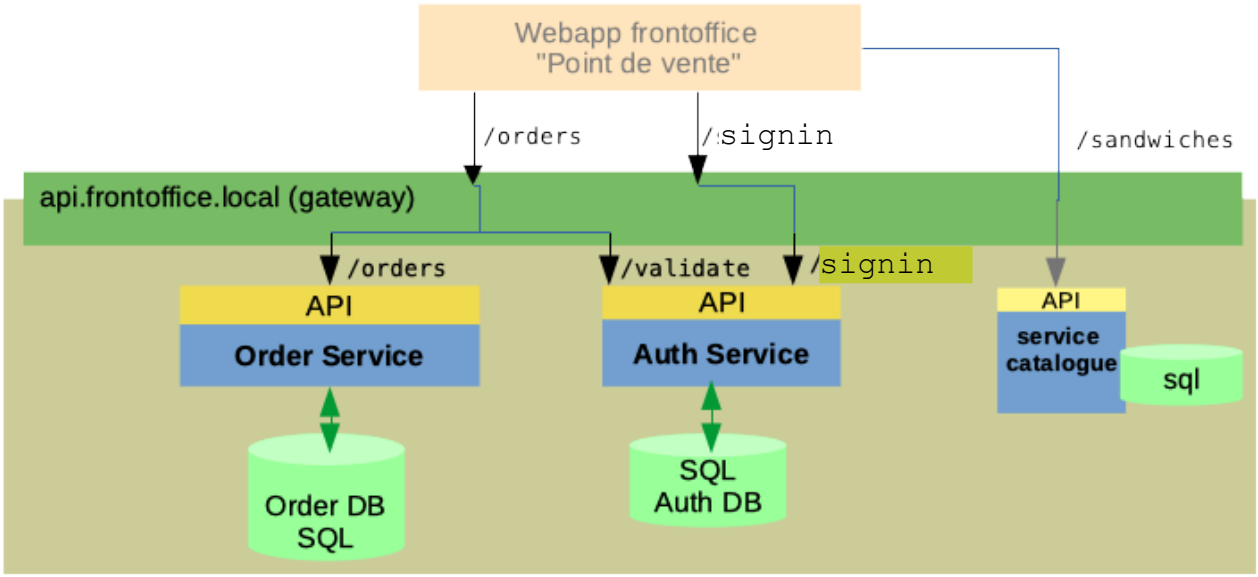
Programmer la route qui permet de vérifier la validité d'un token JWT. Le token est transporté dans le header "Authorization" en mode "Bearer"

```
GET /validate
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJpc3MiOiJodHRwOlw
vXC9hcGkubGJzLmxvY2FsXC9hdXRoIiwiYXVkIjoiaHR0cDpcL1wvYXBpLmxicy5sb2NhbCIsImlhd
CI6MTUxMzY3MTAwMCwiZXhwIjoxNTEzNjc0NjAwLCJjaWQiOjF9.EtE4iY2XIGf_V0Ai9g62D3XU35
IFgSJr6n8ja9jOLr7JCdqxG-oTosWwruGT028oFm_6pwQzUHwYBCtyZx4AGQ
```

Un code 401 est retourné si le header est absent, si le token est invalide ou expiré, accompagné d'un message d'erreur indiquant la raison de la non-validation.
Si le token est valide, un code 200 est retourné, accompagné du profil de l'utilisateur authentifié (email, username, level).

3. contrôle d'accès au front office LBS

Le front office LBS est réalisé par une API passerelle (gateway) dédié qui regroupe et unifie l'accès à l'ensemble des micro-services utilisés par les applications LBS front office, c'est-à-dire destinées aux utilisateurs finaux (clients) de LBS. Cette API expose des routes pour accéder aux ressources du front office (accès au catalogue, création, suivi et paiement de commandes), et assure l'authentification et le contrôle d'accès :



Fonctionnement de l'api front office :

- lors de la réception d'une demande d'authentification (/signin), la requête est acheminée vers le service d'authentification, et la réponse du service d'authentification est retournée au client (token JWT ou message d'erreur),
- lors de la réception d'une requête d'accès à une commande ou une collection de commandes, ou pour la création d'une commande, l'api vérifie la présence d'un accès token et demande sa validation au service d'authentification (validate). Ceci est réalisé par un middleware.
- Si le token est valide la requête est adressée au service de prise de commande et la réponse est retournée au client. Dans le cas contraire un message d'erreur est retourné au client. Lors de l'accès à une commande, le service de prise de commande doit vérifier que l'utilisateur authentifié est bien le propriétaire de la commande.

On implantera uniquement 2 routes :

- la route permettant au client de s'authentifier (signin),
- la route d'accès à la liste des commandes