IUT Nancy-Charlemagne – BUT Informatique S5 – DWM

Développement Web Serveur avancé

TD 2.2 : Authentification et Autorisations dans une architecture microservices

Préliminaires

L'objectif du TD est de mettre en place le processus authn/authz dans le contexte de l'architecture microservices de l'application toubilib en l'appliquant au cas des Rendez-vous. On devra contrôler 3 cas :

- accès au détail d'un RDV, réservé au patient ou au praticien concerné,
- accès à l'agenda d'un praticien, réservé au praticien concerné,
- création d'un RDV, réservé au praticien ou au patient concerné.

Exercice 1: Gateway

Complétez la gateway en y ajoutant les routes pour l'authentification : register, signin, refresh. Les actions correspondantes doivent transférer les requêtes vers l'application toubilib et retourner les réponses. Il faudra notamment gérer les cas d'erreur de façon adéquate en retournant des codes/messages correspondant à chaque cas d'erreur.

Vérifiez que vous êtes bien capables d'obtenir un couple de token JWT grâce à un signin auprès de la gateway.

Exercice 2: microservice d'authentification

L'objectif est d'extraire le service d'authentification de l'application Toubilib pour en faire un microservice indépendant. Procédez comme précédemment :

1. dupliquez l'application toubilib et renommez-la en `app.auth`. Adaptez votre docker compose pour ajouter un nouveau service php. Vérifiez que cette copie de l'application fonctionne correctement.
2. adaptez votre gateway pour transférer les requêtes d'authentification vers ce microservice.
3. nettoyez les répertoires et le code de cette application pour ne conserver que ce qui concerne l'authentification.
4. dans l'application toubilib de départ, il ne doit rester que ce qui concerne la gestion des patients (si cela a été réalisé).

Exercice 3 : validation de tokens JWT

On complète maintenant le microservice d'authentification avec une route permettant de vérifier la validité d'un token JWT. Cette route sera utilisée uniquement par la gateway pour s'assurer de

l'authentification des utilisateurs à la réception de requêtes nécessitant un contrôle d'accès.

Pour cela, créez la route /tokens/validate dans le microservice `app.auth` et l'action correspondante. Cette action extrait le token JWT et vérifie sa validité auprès du provider JWT, et retourne :

- `200` en cas de succès,
- `401` en cas d'échec, avec un message indiquant le problème (token invalide, expiré …)

Exercice 4 : middleware d'authentification dans la gateway

On construit maintenant le middleware d'authentification au sein de la gateway. Ce middleware :

- est appliqué sur toutes les routes qui nécessitent d'être authentifié,
- vérifie uniquement la présence et la validité d'un token JWT dans les requêtes concernées.

Pour cela, ce middleware doit extraire le token JWT de la requête reçue, adresser une requête de validation au microservice d'authentification, puis, en fonction de la réponse, soit lever une exception 401, soit transmettre la requête courante au middleware suivant.

Appliquez ce middleware aux routes de la gateway concernant les RDV : accès au détail d'un RDV, accès à l'agenda d'un praticien, création d'un RDV, et vérifiez que la gateway s'assure de la présence d'un access token valide lorqu'elle reçoit des requêtes correspondant à ces routes.

Exercice 5 : autorisations dans le microservice de gestion des RDV

On remet maintenant en place les contrôles d'autorisation pour la gestion des rendez-vous :

- accès au détail d'un RDV, réservé au patient ou au praticien concerné,
- accès à l'agenda d'un praticien, réserver au praticien concerné,
- création d'un RDV, réservé au praticien ou au patient concerné.

Pour cela, on conserve le service de contrôle d'autorisations associé au service métier de gestion des RDV. Il faut ensuite transformer le ou les middleware de contrôle d'autorisation pour extraire et décoder le token JWT des requêtes. Inutile de valider ces tokens puisque cela a été réalisé par la gateway. Ce middleware utilise le service de contrôle d'autorisations pour vérifier que l'utilisateur authentifié dispose des droits pour réaliser l'action demandée. En cas de défaut d'autorisation, le middleware lève une exception correspondant au cas (401 ou 403) afin de retourner une réponse avec le code adéquat.

Ces cas d'erreurs doivent à leur tour être traités par la gateway qui doit retourner une réponse avec un code adéquat au client.