IUT Nancy-Charlemagne – BUT Informatique S5 – DWM

Développement Web Serveur avancé

TD 2.1 : Microservices et API Gateway

Préliminaires

Faites un fork de votre projet toubilib et installez le nouveau projet sur votre machine. Vérifiez qu'il fonctionne correctement, chargez les données dans les différentes bases SQL. Dans la suite, vous aurez besoin de programmer une api Gateway qui se place devant l'api toubilib. Cette gateway devra rediriger les requêtes vers l'api toubilib. Elle utilisera pour cela la librairie guzzle. Une introduction technique à l'utilisation de guzzle est disponible dans Arche.

Exercice 1: Gateway initiale

Cette première étape est consacrée à la mise en place d'une api gateway qui se place devant l'api toubilib. Dans un premier temps, cette api toubilib ne sera pas touchée, et l'api gateway se contente de rediriger toutes les requêtes qu'elle reçoit vers l'api toubilib. Pour cela, procédez ainsi :

1. Ajoutez un nouveau service docker php dans votre projet : mettez à jour votre docker compose installez les dépendances qui vous seront utiles pour la gateway (par exemple, guzzle).
2. Créez une application Slim correspondant à la gateway : index, configuration (bootstrap, routes), partie application/actions.
3. programmez une route pour accéder à la liste des praticiens. L'action correspondante se contente d'interroger l'api toubilib et de renvoyer la réponse. Vous aurez éventuellement besoin de désactiver les middleware d'authn/authz côté api toubilib.
4. Le client Guzzle doit être injecté dans l'action. Il est instancié dans le container d'injection dépendances de l'application.
5. une fois que la route fonctionne, ajoutez un middleware pour la gestion des headers CORS dans la gateway, vous pouvez ensuite retirer cette gestion de l'API toubilib.

Exercice 2: étendre la gateway

On ajoute maintenant une nouvelle route dans la gateway pour obtenir les informations détaillées d'un praticien.

1. ajouter la route correspondante dans la gateway, qui redirige la requête vers l'api toubilib.
2. vérifier que la route fonctionne correctement, et testez là avec un identifiant de praticien correct, puis un identifiant de praticien inexistant.
3. Faites en sorte que la gateway renvoie une réponse d'erreur 404 si le praticien n'existe pas : pour cela vous devrez gérer les exceptions guzzle dans les actions de la gateway, et déclencher les exceptions Slim adéquates.
4. Transformez vos deux actions spécifiques de la gateway en une action générique qui décompose la requête reçue et le redirige vers l'api toubilib. On fait l'hypothèse que les URI de

l'API exposée par la gateway sont les mêmes que celles de l'API toubilib.

5. Sur le même principe, ajoutez une route pour obtenir la liste des rendez-vous d'un praticien.

Exercice 3 : Décomposer l'application Toubilib, première étape

L'objectif de cette étape est de décomposer l'application toubilib en plusieurs microservices. Pour cela, on va créer un microservice pour les praticiens, un micro-service pour les rendez-vous. Chaque microservice aura sa propre base de données, et exposera une API RESTful. Dans un premier temps, on conserve l'application toubilib complète et on va extraire les micro-services un à un.

1. Dupliquer l'application toubilib, et renommez-la en `app-praticiens`. Adaptez votre docker compose pour ajouter un nouveau service php. Vérifier que cette copie d'application fonctionne correctement.
2. Dans la gateway, faites en sorte que les requêtes concernant le service praticiens soit adressée à cette api. Cela nécessitera d'injecter un nouveau client Guzzle pour ce service, et de modifier ou dupliquer l'action générique pour qu'elle puisse rediriger les requêtes vers le bon service.
3. Nettoyez les répertoires et le code de l'application `app-praticiens` pour ne conserver que ce qui est nécessaire pour le service praticiens. Vous pouvez supprimer les parties inutiles, notamment les parties concernant les rendez-vous.

Exercice 4 : Décomposer l'application Toubilib, deuxième étape

Sur le même principe, on va maintenant extraire le service des rendez-vous de l'application toubilib.

1. Dupliquer l'application toubilib, et renommez-la en `app-rdv`. Adaptez votre docker compose pour ajouter un nouveau service php. Vérifier que cette copie d'application fonctionne correctement.
2. Dans la gateway, faites en sorte que les requêtes concernant le service rendez-vous soit adressée à cette api. Cela nécessitera d'injecter un nouveau client Guzzle pour ce service, et de modifier ou dupliquer l'action générique pour qu'elle puisse rediriger les requêtes vers le bon service.
3. Attention : le service de gestion de RDV a besoin d'informations provenant du service de gestion des praticiens. Cela se traduit par le fait que lors de la création du service métier de gestion des RDV, on injecte une interface permettant de récupérer des informations sur les praticiens. Cette interface sera implémentée non pas par le service de gestion des praticien, mais par un adaptateur implanté dans l'infrastructure qui interrogera le micro-service de gestions des praticiens au travers de son API.
4. Vérifiez que ce service fonctionne correctement et qu'il est accessible depuis la gateway.
5. Nettoyez les répertoires et le code de l'application `app-rdv` pour ne conserver que ce qui est nécessaire pour le service RDV. Vous pouvez supprimer les parties inutiles, notamment les parties concernant les praticiens.