Outils et méthodes de développement

Johann Chopin, Université de Lorraine, ISFATES – Octobre 2025

TP noté

Vous allez documenter une API que vous aurez conçue en utilisant la spécification openAPI au sein d'un repository Git public hébergé sur GitHub. Vous visualiserez localement votre documentation générée sous forme de page statique HTML à l'aide de la bibliothèque nodejs *redocly*. Comme vous ne souhaitez pas installer nodejs sur votre ordinateur, vous utiliserez la technologie Docker pour construire une image qui génère la page statique et la rend accessible à partir d'un port spécifié. Dans la phase de développement, vous utiliserez les avantages de Composition pour démarrer rapidement cette image locale et la reconstruire lorsque le fichier API est modifié. Vous devrez vous assurer de la qualité de votre définition d'API au sein d'un CI en utilisant les GitHub Actions et la bibliothèque *redocly*. Lors d'une nouvelle release, votre CD publiera votre documentation sur la GitHub Page de votre repository.

Exigences et barème

1. Gestion de Git (**/0.5**)

- Vous committerez régulièrement votre travail de sorte à judicieusement suivre la convention Conventional Commits. (**0.5**)
- Les releases se feront via des simples tags git.

2. Conception de l'API (**/11**)

- L'API devra être RESTful et documenter les actions suivantes au sein d'un fichier *src/openAPI.yaml*:
  - Récupérer la collection de votre choix (**1**)
  - Ajouter et modifier une ressource à cette liste (**2**)
  - Supprimer une ressource (**0.5**)
  - Récupérer une collection enfant de votre première collection (**0.5**)

- Les actions d'ajout, de suppression et de modification devront être protégées par une authentification utilisant un Bearer token JWT. (**0.5**)
- Chaque appel à une collection devra être paginable et vous définirez un filtre additionnel de votre choix (par exemple *created-by*), (**1.5**)
- Les schémas ne devront pas être dupliqués (**3**)
- Les schémas devront être aussi précis que possible (max, min, …) et des exemples les illustrerons. (**1**)
- L'API ne prendra en charge que le type de média JSON.
- Documentez les différents codes de statut qui peuvent être retournés par vos endpoints. (**1**)

3. Développement avec Docker (**/6**)

- Vous utiliserez un *Dockerfile* créant une image qui génère et hoste une page static HTML sur le port 80. (**3**)
    - La génération se fera avec la librairie *redocly/cli* et devra être la plus efficace possible.
    - Vous utiliserez l'image *nginx:alpine* pour hoster le fichier généré *index.html*.
- Vous ajouterez un fichier *compose.yaml* qui démarrera un service à partir du *Dockerfile* local. Ce service devra se recréer lors d'un changement du fichier de la définition API à l'aide de Compose Watch. (**2**)
- Documentez dans un README les différentes commandes docker utiles au développement. (**1**)

4. Automatisation CI/CD (**/4**)

- Définissez un workflow lint dans un écosystème nodejs installant la librairie *redocly/cli* permettant de vérifier la qualité de votre fichier de *src/openAPI.yaml* à l'aide de la commande *lint*. Ce workflow s'exécutera lors de chaque push. (**1**)
- Définissez un workflow dédié pour le build de la documentation à l'aide toujours de *redocly/cli*. (**1**)
- Ajoutez un workflow de CD (**2**)
    - Celui-ci s'exécutera uniquement lors d'un nouveau tag suivant la convention SemVer
    - Il réutilisera le workflow de build afin de récupérer le fichier généré *index.html* dans un job dédié.
    - Un autre job devra publier ce fichier sur une page GitHub Page (voir la documentation officielle).