REST

Bonnes pratiques

Bonnes pratiques

- Utiliser JSON comme format pour l'envoi et la réception de données.
  - Dans le passé, accepter et répondre à une requête d'API étaient principalement faits en XML ou en HTML. Aujourd'hui, le JSON (JavaScript Object Notation) est en grande partie devenu de facto le format d'envoi et de réception des données d'API.
  - Avec XML, c'est souvent compliqué de décoder et encoder de la donnée, ce qui a amené à ce que l'XML ne soit plus vraiment supporté par les frameworks.
  - JavaScript possède une méthode intégrée pour convertir la donnée JSON à travers l'API fetch, car JSON a principalement été créé pour ça.
  - Pour s'assurer que le client interprète les données JSON correctement, vous devez définir le type de Content-Type dans l'en-tête de réponse comme application/json quand vous faites la requête.

Bonnes pratiques

- Utiliser les noms plutôt que les verbes pour les Endpoints.
  - Quand vous concevez une API REST, vous ne devriez pas utiliser des verbes dans le chemin des endpoints. Les endpoints doivent utiliser des noms, signifiant ce que chacun fait.
  - Les méthodes HTTP telles que GET, POST, PUT et DELETE sont déjà en forme de verbes pour effectuer des opérations CRUD basiques (Create, Read, Update, Delete).
  - Vous devez laisser les verbes HTTP gérer ce que les endpoints font. GET récupère de la donnée, POST crée de la donnée, PUT la met à jour, et DELETE la supprime.

Bonnes pratiques

- Nommez les Collections avec des noms au pluriel
  - Imaginez les données de votre API comme étant une collection de différentes ressources pour vos consommateurs.
  - Si vous avez un endpoint comme https://mysite.com/produit/1, il pourrait être acceptable de supprimer un post avec une requête DELETE ou mettre à jour avec une requête PUT ou PATCH, mais ça ne dit pas à l'utilisateur qu'il peut y avoir d'autres produits dans la collection. C'est pourquoi votre collection devrait utiliser des noms aux pluriels.

Bonnes pratiques

- Utilisez les codes d'états dans la gestion des erreurs
  - Vous devez toujours utiliser des codes d'états HTTP en réponse aux requêtes faites à vos API.
  - Ça aide les utilisateurs de vos API à savoir ce qui se passe, que la requête est réussie, échouée, ou autre chose.

Bonnes pratiques

- Utilisez l'imbrication sur les endpoints pour montrer les relations
  - Parfois, différents endpoints peuvent être interreliés, vous devez donc les imbriquer pour les rendre plus faciles à comprendre.
  - Exemple : différents produits peuvent appartenir à différentes catégories, donc un endpoint tel que https://monsite.com/produits/categorie est un bon cas d'imbrication.
  - Dans le même esprit, les produits ont leurs propres catégories, donc pour récupérer les catégories d'un produit, un endpoint comme https://monsite.com/produits/produitid/categories ferait sens.
  - Vous devez éviter d'imbriquer sur plus de 3 niveaux car cela peut rendre l'API moins élégante et plus difficile à lire.

Bonnes pratiques

- Définissez le versionnage
- Les API RESTs doivent avoir plusieurs versions, comme ça, vous ne forcez pas les clients (utilisateur) à migrer vers une nouvelle version. Cela peut même casser l'application si vous ne faites pas attention.
- Un des systèmes de versionnage le plus commun dans le développement web est le versionnage sémantique.
  - Exemple de versionnage sémantique : 1.0.0, 2.1.2 et 3.3.4.
  - Le 1er nombre représente la version majeure, le 2ème nombre la version mineure, et le 3ème la version du patch.

https://monsite.com/v1.0.0/ pour la version 1

Bonnes pratiques

- Fournir une documentation d'API correct
- Quand vous faites une API REST, vous devez aider les clients (consommateurs) à apprendre et trouver comment l'utiliser correctement. Le meilleur moyen est de fournir une documentation pour l'API.
- Cette documentation doit contenir :
  - Les endpoints pertinents de l'API.
  - Des exemples de requêtes de ces endpoints.
  - L'implémentation dans plusieurs langages de programmation.
  - Les messages répertoriés pour différentes erreurs et leurs codes d'états.
- Un des outils les plus communs que vous pouvez utiliser pour la documentation d'API est Swagger.