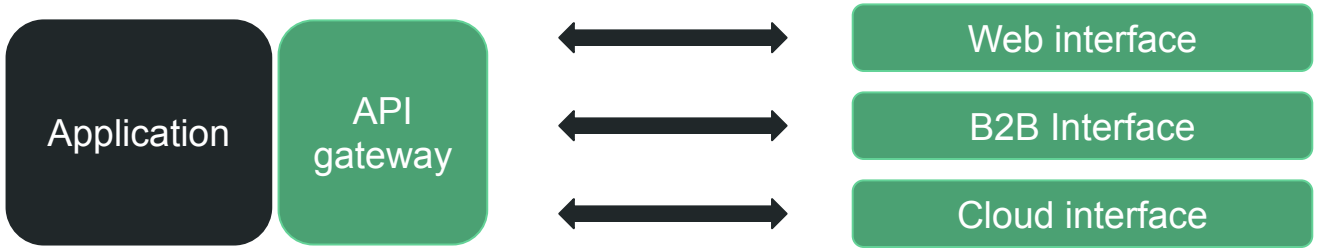
Corriger les vulnérabilités et développer de façon sécurisée

1 - Les passerelles API

Une passerelle API est un point d'entrée unique vers l'API. Le dispatch vers les différentes ressources se fait ensuite au niveau métier.

Cela permet de centraliser la sécurité, le cache, la disponibilité... sur un seul composant.



2 - Utilisation d’outils standardisés

L’utilisation d’outils standardisés permet de bénéficier de l’expérience de logiciels éprouvés et accélère vos développements, moyennant un temps de prise en mains.

- API platform https://api-platform.com/
- Swagger https://swagger.io/tools/
- APIARY https://apiary.io/how-apiary-works

3 - Tester son API

Prévoir une couverture de test la plus exhaustive possible. Le test permet d'avoir confiance en son logiciel.

Les tests unitaires peuvent également avoir un intérêt.

Quelques outils pouvant aider :

- **SOAP UI** : test de services SOAP et REST. Axé sur la sécurité https://www.soapui.org/
- **Postman** : outil d'envoi de requêtes. Bon pour le test https://www.postman.com/postman
- **Runscope** : outil de test d'API https://www.runscope.com/

4 - Documenter son API

La documentation doit être exhaustive et précise. Une bonne documentation permet aux utilisateurs d'utiliser facilement votre API.

Quelques outils pouvant aider :

- **Swagger** pour développement et une doc en ligne https://swagger.io/tools/
- **APIARY** pour développement et doc en ligne https://apiary.io/how-apiary-works
- **ReadTheDocs** pour une doc en ligne https://readthedocs.org/
- **Postman** pour créer et fournir des exemples de requêtes : https://www.postman.com/postman

5 - Contraindre les données

Il faut être précis dans les données que nous recevons et envoyons. Cela permet d'éviter les abus. Un service web ne doit être utilisé **QUE** pour ce à quoi il sert.

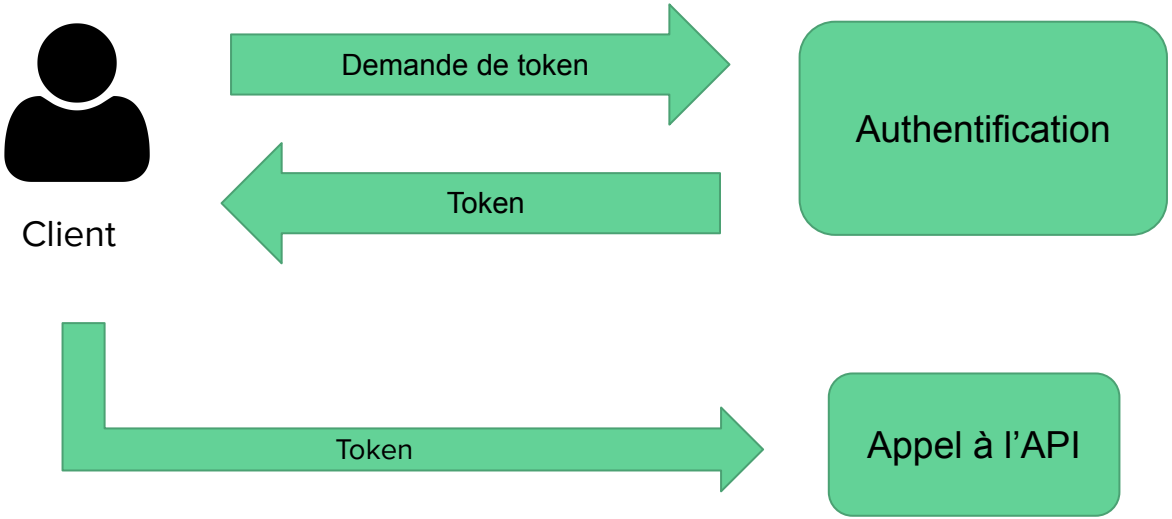
Exemple :

La ressource https://api.monsite.com/users/#ID# ne doit accepter comme paramètre ID que des données correspondant (un int par exemple).

La ressource https://api.monsite.com/users doit renvoyer un array. Elle ne doit pas permettre à l'utilisateur d'accéder à d'autres ressources.

6 - Identification

Une bonne pratique consiste à prévoir un système d'identification et de jeton pour chaque appel à l'API.



Un token doit être individuel et périssable.

Voir système OAuth.

Evaluation - Projet technique - MAJ 6

Webservice REST - Le CRM

Implémenter un système d’identification se rapprochant de Oauth.

Durée de vie d’un token : 10 minutes.