Créer un serveur Web API

1. Contexte du projet :

En tant qu'élèves de NSI, vous avez complété une fiche d'informations en début d'années de Première. Ces informations, qui doivent permettre aux enseignants de mieux vous connaître afin d'adapter les enseignements, sont difficilement exploitables sur support papier.
C'est pourquoi ils souhaiteraient pouvoir disposer de ces informations via un serveur Web API.

L'objectif de ce projet est donc de:

- numériser et structurer les informations disponibles,
- concevoir une série de services web offrant des accès ciblés aux données,
- concevoir des clients légers qui utiliseront les services.

2. Web services ou Web API

https://www.wrike.com/fr/blog/api-application-programming-interface-explication-et-donnees-cles/

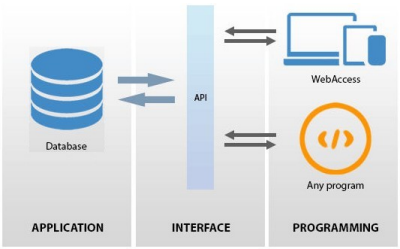
A l'heure actuelle, les API (Application Programming Interface) connectent le monde du numérique d'une façon que nous n'aurions jamais crue possible.

- Au travail, nous sommes capables d'accomplir des projets de façon plus rapide et plus efficace avec des analyses en temps réel et des outils intégrés.
- Dans le domaine de la santé, les API rassemblent des données cliniques collaboratives et des équipes de recherche, accélérant largement le développement de nouveaux traitements et permettant aux fournisseurs d'améliorer la qualité des soins.

Les API sont déjà présentes dans notre quotidien, notamment au travers d'applications smartphone. L'avènement de l'Internet of Things va encore accélèrer ce phénomène.

À quoi servent les API ?

Une API est une interface permettant à deux composants logiciels indépendants d'échanger des informations. Elle agit en tant qu'intermédiaire entre les fonctions logicielles internes et externes, ce qui permet un échange d'informations tellement intégré qu'il passe souvent inaperçu aux yeux de l'utilisateur final.



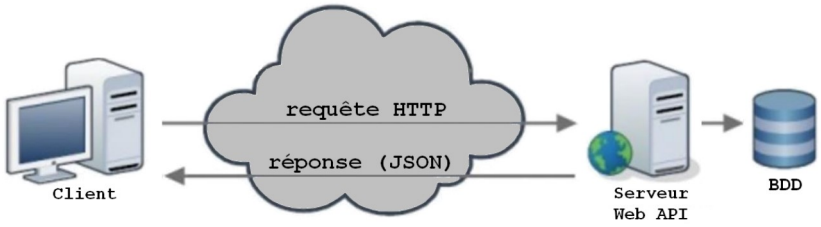
Pourquoi les API sont-elles importantes ?

Les API permettent donc de connecter 2 composants logiciels sans avoir besoin de connaître les détails de leur mise en œuvre. De ce fait, elles simplifient le développement d'applications et facilitent la collaboration entre les équipes informatiques et métier.

3. L'Architecture REST

https://openclassrooms.com/fr/courses/6031886-debutez-avec-les-api-rest

REST (REpresentational State Transfer) est devenu la conception architecturale standard des API Web. La notion de ressources offertes par le serveur est au cœur du concept de services Web REST. Elles sont représentées par des URI.



Les clients envoient des requêtes à ces URI en utilisant les méthodes définies par le protocole HTTP, et éventuellement en conséquence de cela l'état de la ressource affectée change. Les méthodes de requête HTTP sont généralement conçues pour affecter une ressource donnée de manière standard:

Méthode HTTP	Action
GET	Obtenir des informations sur une ressource
POST	Créer une nouvelle ressource
PUT	Mettre à jour une ressource
DELETE	Supprimer une ressource

La conception REST ne nécessite pas de format spécifique pour les données fournies avec les demandes. En général, les données sont fournies dans le corps de la requête en tant qu'objet JSON, ou parfois en tant qu'arguments dans la partie chaîne de requête de l'URL.

4. Cahier des charges

4.1. Serveur Web API :

Ressources :
Le serveur Web API sera développé grâce au module Flask de Python. Les données que vous intègrerez à vos requêtes seront des structures JSON. Voici les ressources à votre disposition :

- _Questionnaire Première NSI.docx, fiche de renseignement type.
- fiche pratique JSON, pour appréhender le format JSON.
- structure_base_simplifiee.json, la structure JSON pour une fiche élève.
- elevesNSI.json, la structure contenant la liste des élèves.
- fiche pratique Flask, pour débuter avec Flask.
- TP2-serveurAPI_eleve.py, le fichier serveur avec 1 premier service fonctionnel.

Contraintes de conception :
Le travail préalable de conception consiste en un exercice d'identification des ressources qui seront exposées et de la manière dont elles seront affectées par les différentes méthodes de demande.

1. La première chose à faire est de décider quelle est l'URL racine pour accéder à ces services. Nous choisirons : http://[hostname]/api/
2. L'étape suivante consiste à définir les ressources qui seront exposées par ces services.
   Les enseignants souhaitent donc pouvoir disposer de 2 ressources distinctes :
   - d'une structure JSON décrivant un modèle vierge d'une fiche élève,
   - et d'une structure JSON contenant la liste complètes des élèves.

3. Nos ressource seront être offertes en utilisant les méthodes HTTP comme suit:

Méthode HTTP	URI	Action
GET	http://[nom d'hôte]/api/eleves	Récupérer la liste des fiches des élèves en JSON
GET	http://[nom d'hôte]/api/modele	Récupérer une fiche modèle vierge en JSON
POST	http://[nom d'hôte]/api/add	Ajouter un élève via un fichier JSON
GET	http://[nom d'hôte]/api/eleve/NOM	Récupérer la fiche d'un élève en JSON
GET	http://[nom d'hôte]/api/getRS	Récupérer la liste des réseaux sociaux
GET	http://[nom d'hôte]/api/poursuiteNSI	Récupérer la liste des réseaux sociaux
DELETE	http://[nom d'hôte]/api/delete/NOM	Supprimer la fiche d'un élève

Mise en oeuvre:

Pour bien débuter, vous pourrez effectuer les premières tâches dans l'ordre suivant:

1. Compléter le fichier elevesNSI.json avec 2 ou 3 exemples.
2. Exécuter le script TP2-serveurAPI_eleve.py dans un projet serveur Flask.
3. Vérifier le fonctionnement du premier service mis en oeuvre via un navigateur.
4. Comprendre la mise en oeuvre du premier service et compléter les 2 autres services.
5. Tester ces 2 nouveaux services en utilisant https://reqbin.com/ (nécessaire pour le POST).
6. A vous de créer tous les autres services décris ci-dessus.

4.2. Clients légers :

Un client léger est un script python (ou autre) en mode console ou avec une IHM qui consommera certains services au travers de son exécution.

Réaliser les clients légers d'exploitations suivants :

1. **client_ajout.py** : client console qui ajoutera un élève à partir de la récupération du modèle vierge.
2. **sondageReseauxSociaux.py** : client affichant un histogramme dynamique des réseaux sociaux utilisés.

4.3. Pour aller plus loin:

Un bon nombre d'évolutions pourront être ajoutées à notre projet :

- Améliorer le traitement des informations reçues (expressions régulières, ...)
- Ajouter d'autres services comme par exemple un service permettant à un élève de modifier sa fiche,
- Limiter l'accès à certains services par un mécanisme d'identification (token),
- Stocker les informations dans une base de données et non sur un fichier JSON,
- Héberger votre serveur en ligne (cf. https://www.pythonanywhere.com/,
- Créer d'autres clients légers d'exploitation,
- etc.