Introduction aux APIs

Mattia Bunel

12 février 2025

EHESS

Introduction

Plan

Introduction

Définir le problème

Un tour du web

API Web

Formats de sérialisation

Protocoles

Les différentes architectures

La boîte à outils

Quelques problèmes courants

Et si je veux construire mon API ?

Un peu de pratique

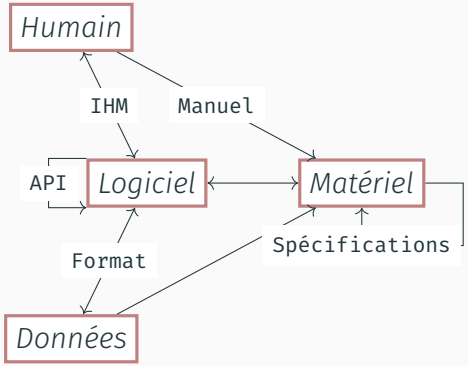
Définir le problème

Qu’est-ce qu’une API ?

API pour *Application Programming Interface*

Interfaces

Une **API** est une interface logiciel — logiciel



Quelques exemples

Exemples d'API :

- Bibliothèque logicielle
- Pilote de matériel
- API web

Quelques exemples

Exemples d'API :

- Bibliothèque logicielle
- Pilote de matériel
- API web

API de bibliothèque logicielle

Mon programme utilise l'API de la bibliothèque `json` pour écrire un fichier `.json`

```python
import json

data = {
    "nom": "John",
    "age": 30,
    "ville": "Paris"
}

# Écriture du dictionnaire dans un fichier JSON
with open("donnees.json", "w") as f:
    json.dump(data, f)
```

Quelques exemples

Exemples d'API :

- Bibliothèque logicielle
- Pilote de matériel
- API web

API d'un matériel

Mon programme utilise une API pour activer un signal sur un port GPIO

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(17, GPIO.OUT)
# On envoie un signal de force maximale sur le pin 17
GPIO.output(17, GPIO.HIGH)
```

Quelques exemples

Exemples d'API :

- Bibliothèque logicielle
- Pilote de matériel
- API web

API WEB

Mon programme utilise l'API de la BAN pour faire du géocodage

```
curl 'https://data.geopf.fr/geocodage/search?q=Refuge%20
↪  du%20pelvoux&limit=1&index=poi&returntruegeometry='
```

APIs publiques & APIs privées

On distingue généralement :

- Les **API publiques**
- Les **API privées**

APIs publiques

Une **API publique** est :

- Destinée à être utilisée par l'extérieur
- Exposée au public

APIs privées

Une **API privée** est :

- Réservée à un usage interne
- Non exposée au public

On ne parlera que d’API Web publiques

Un tour du web

Le Web n’est pas Internet

Figure 1 : Tim Berners-Lee et Vint Cerf lors du 20 ème anniversaire du W3C (2014).

Qu'est-ce qu'Internet ?

Internet est :

- Un **réseau informatique mondial**...
- ... composé d'un ensemble de sous-réseaux (AS) connectés (p.ex Renater, Free)...
- ... connectés à l'aide de **protocoles** communs, la pile **TCP/IP**...
- ... standardisés dans des RFC (*requests for comments*)...
- ... sous l'égide de l'IETF (*Internet Engineering Task Force*)

Qu’est-ce que le Web ?

Le **Web** est :

- **Une des applications** d’internet (autres. ex. Mails [IMAP, SMTP], chat [IRC], synchronisation d’horloges [NTP])...
- ... qui lie un ensemble de ressources ...
- ... à l’aide de **protocoles** (p. ex. HTTP, WebSocket)...
- ... et de **formats** (p.ex HTML, SVG, XML)...
- ... normalisées par le W3C

Le modèle OSI

- Le **modèle OSI** (*Open systems interconnection*) est un ensemble de normes ISO définissant un modèle de communication réseau entre systèmes informatiques.
- Le modèle **OSI** est défini par la norme **ISO 7498**
- Les couches sont définies par leurs fonctions
- Une couche peu contenir différents protocoles

Couche	Nom
7	Application
6	Présentation
5	Session
4	Transport
3	Réseau
2	Liaison
1	Physique

Le modèle TCP-IP

- Pour des raisons historiques, internet est un peu différent
- L'ensemble des protocoles d'internet forme la pile **TCP/IP**
- Il n'y a pas de correspondance exacte entre modèle **OSI** et **TCP/IP**

Couche	Nom
5	Application
4	Transport
3	Réseau
2	Liaison
1	Physique

Les couches de la pile TCP-IP

Couches TCP-IP :

- Application
- Transport
- Réseau
- Liaison
- Physique

Les couches de la pile TCP-IP

Couches TCP-IP :

- Application
- Transport
- Réseau
- Liaison
- Physique

Couche *application*

- Point d'entrée du réseau
- Couche des protocoles applicatifs (**HTTP, FTP, IMAP**)

Analogie :

- Un document formalisé, p.ex un *bulletin de salaire*

Les couches de la pile TCP-IP

Couches TCP-IP :

- Application
- Transport
- Réseau
- Liaison
- Physique

Couche *transport*

- Décrit les processus de fiabilisation et de gestion des erreurs
- Assuré par les protocoles **TCP** (*Transmission Control Protocol*) ou **UDP** (*User Datagram Protocol*)

Analogie :

- Le mode de distribution postale, p. ex. un recommandé (**TCP**), une lettre simple (**UDP**)

Les couches de la pile TCP-IP

Couches TCP-IP :

- Application
- Transport
- Réseau
- Liaison
- Physique

Couche *réseau*

- Décrit le processus de routage
- Transporte les paquets de la machine d'origine à la machine de destination
- Assurée par le protocole **IP** (Internet Protocol)

Analogie :

- Le système d'adressage de la Poste, *i.e.* centres de tri et de distribution

Les couches de la pile TCP-IP

Couches TCP-IP :

- Application
- Transport
- Réseau
- Liaison
- Physique

Couche *liaison*

- Défini la manière dont les données sont transportées sur la couche physique
- Assuré — entre autre — par le protocole **ethernet**

Analogie :

- Le transport entre deux composantes du système d'adressage (par camion, par avion)

Les couches de la pile TCP-IP

Couches TCP-IP :

- Application
- Transport
- Réseau
- Liaison
- Physique

Couche *physique*

- Décrit les caractéristiques du vecteur physique
- Transforme les bits en signaux électriques

Analogie :

- Le fonctionnement du vecteur, p. ex. fonctionnement du camion

Les couches de la pile TCP-IP

Couches TCP-IP :

- Application
- Transport
- Réseau
- Liaison
- Physique

Synthèse

- Une requête **HTTP** est dans un paquet **TCP**...
- ... qui est dans un paquet **IP**...
- ... qui est dans une **trame ethernet**...
- ... qui est transportée par une **fibre optique**

Flux de données

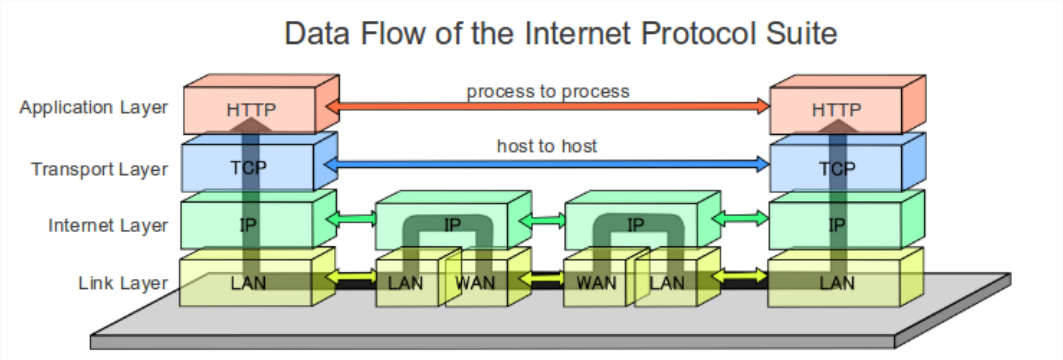


Figure 2 : Illustration d'un flux de données dans la pile TCP/IP (Source : `https://commons.wikimedia.org/wiki/File:Data_Flow_of_the_Internet_Protocol_Suite.PNG`)

Conclusion

- L'intelligence est en périphérie
- Chaque couche peut faire abstraction des couches inférieures
- Les protocoles de la *couche application* sont les seuls qui nous concernent

Pour aller plus loin...

https://www.youtube.com/watch?v=0gE1AtxQ44A

API Web

Exemples d’API Web

Exemples d’API Web :

- API Adresse (BAN)
- `data.gouv.fr`
- HAL
- Zotero
- Free Mobile
- Countries

Exemples d'API Web :

- API Adresse (BAN)
- data.gouv.fr
- HAL
- Zotero
- Free Mobile
- Countries

API de la Base Adresse Nationale

https://adresse.data.gouv.fr/

Fonctions :

- Géocodage (Adresse → Coordonnées)
- Géocodage inverse (Coordonnées → Adresse)

Exemple :

```
# Renvoie l'addresse correspondant aux coordonées du
↪  bâtiment
curl -L 'https://api-adresse.data.gouv.fr/reverse/?lon=2
↪  .3665&lat=48.9085'
```

C. Connections nécessaire

Exemples d'API Web

Exemples d'API Web :

- API Adresse (BAN)
- `data.gouv.fr`
- HAL
- Zotero
- Free Mobile
- Countries

API de `data.gouv.fr`

`https://guides.data.gouv.fr/guide-data.gouv.fr/api-1`

Fonctions :

- Lister, rechercher les jeux de données
- Créer, modifier ou supprimer un jeu de données (`C.`)
- Mettre à jour son profil utilisateur (`C.`)

Exemple :

```
# Liste TOUS les jeux de données
curl -L 'http://www.data.gouv.fr/api/1/datasets/'
```

`C.` Connections nécessaire

Exemples d'API Web :

- API Adresse (BAN)
- `data.gouv.fr`
- HAL
- Zotero
- Free Mobile
- Countries

API HAL

`https://guides.data.gouv.fr/guide-data.gouv.fr/api-1`

Fonctions :

- Lister, rechercher les jeux de données
- Déposer des documents (`C.`)

Exemple :

```
# Récupérer mes publications en bibtex
# 13740 est mon identifiant auteur
curl -L 'http://api.archives-ouvertes.fr/search/?q=authI⌋
↪  dPerson_i:13740&wt=bibtex'
```

`C.` Connections nécessaire

Exemples d'API Web

Exemples d'API Web :

- API Adresse (BAN)
- `data.gouv.fr`
- HAL
- Zotero
- Free Mobile
- Countries

API Zotero

https://www.zotero.org/support/dev/web_api/v3/start

Fonctions :

- Lister, rechercher les jeux de données

Exemple :

```
# Consulter la bibliographie collaborative du GT Notebook
# 4416056 est l'identifiant du groupe
curl -L
↪  'https://api.zotero.org/groups/4416056/collections'
```

C. Connections nécessaire

Exemples d'API Web :

- API Adresse (BAN)
- `data.gouv.fr`
- HAL
- Zotero
- Free Mobile
- Countries

API Free mobile

Fonctions :

- Permet s'envoyer des sms (C.)

Exemple :

```
# <user> est mon numéro client
# <token> est ma clé d'API
curl -L 'https://smsapi.free-mobile.fr/sendmsg?user=<use
↪  r>r&pass=<token>&msg=coucou'
```

C. Connections nécessaire

Exemples d'API Web :

- API Adresse (BAN)
- `data.gouv.fr`
- HAL
- Zotero
- Free Mobile
- Countries

API GraphQL Countries

`https://studio.apollographql.com/public/countries/variant/current/home`

Fonctions :

- Permet de récupérer des informations diverses sur les pays

Exemple :

```
# On doit utiliser une requête plus compliquée
curl -L --request POST --header 'content-type:
↪  application/json'
↪  'https://countries.trevorblades.com/graphql' --data
↪  '{"query":"query Query {country(code: \"BR\") {name,
↪  native, capital, emoji, currency, languages {code,
↪  name}}}"}'
```

C. Connections nécessaire

La recette d'une API

Pour faire une **API** il faut définir :

- Un moyen de contacter l'API : *le protocole*
- un *format de données* d'échange
- Un moyen de décrire sa requête : *une interface*

Formats de sérialisation

json

- Format de sérialisation
- Inspiré de la notion objet du javascript
- Normalisé par l'IETF

```
{
  "livres" : [
    {
      "id" : "Frege1879",
      "titre": "Idéographie",
      "date": 1879,
      "auteur": {"nom" : "Frege", "prenom": "Gottlob"}
    },
    {
      "id" : "Wittgenstein1921",
      "titre": "Tractatus logico-philosophicus",
      "date": 1921,
      "auteur": {"nom" : "Wittgenstein", "prenom":
      ↪  "Ludwig"}
    }
  ]
}
```

xml

- Format de sérialisation
- Inspiré du SGML et de l'HTML
- Normalisé par le W3C

```
<livres>
  <livre id="Frege1879">
    <titre>Idéographie</titre>
    <date>1879</date>
    <auteur>
      <nom>Frege</nom>
      <prenom>Gottlob</prenom>
    </auteur>
  </livre>
  <livre id="Wittgenstein1921">
    <titre>Tractatus logico-philosophicus</titre>
    <date>1921</date>
    <auteur>
      <nom>Wittgenstein</nom>
      <prenom>Ludwig</prenom>
    </auteur>
  </livre>
</livres>
```

Protocoles

Qu’est-ce qu’un protocole ?

Les API Web utilisent principalement deux protocoles de la *couche application :*

- le protocole **HTTP**
- le protocole **WebSocket**

Présentation des protocoles

- HTTP
- WebSocket

- HTTP
- WebSocket

Protocole HTTP

Fonctions :

- Protocole historique du web
- Protocole relativement simple
- Actuellement en version 3
- Protocole non *full-duplex*

Exemple :

- L'énorme majorité des API Web

- HTTP
- `WebSocket`

Protocole `WebSocket`

Fonctions :

- Protocole récent (2011)
- Permet une *communication bidirectionelle*
- Construit pour des applications particulières (p. ex. Messagerie instantanée), limitées par le HTTP
- Rarement utilisé pour les API

Exemple d'API WebSocket :

- **RIS Live** : `https://ris-live.ripe.net/`

Le HTTP est un protocole simple

Une requête HTTP est :

- un texte normalisé
- qui commence par un verbe
- suivi d'une adresse
- d'un header
- et d'un corps

```
POST /altimetrie/1.0/calcul/alti/rest/elev
↪  ation.json HTTP/1.1
Host: data.geopf.fr
Content-Type: application/json
Accept: */*
Content-Length: 159

{ "lon": "6.4202", "lat": "44.916367",
↪  "resource": "ign_rge_alti_wld" }
```

Verbes HTTP

- GET
- POST
- PUT
- PATCH
- DELETE
- HEAD
- CONNECT
- TRACE
- OPTIONS

Verbes HTTP

- GET
- POST
- PUT
- PATCH
- DELETE
- HEAD
- CONNECT
- TRACE
- OPTIONS

GET

Fonctions :

- Récupère un ressource

Exemple :

- `GET /datasets/` : Liste tous les jeux de données

Verbes HTTP

- `GET`
- `POST`
- `PUT`
- `PATCH`
- `DELETE`
- `HEAD`
- `CONNECT`
- `TRACE`
- `OPTIONS`

POST

Fonctions :

- Créé une nouvelle ressource
- Les informations sont contenues dans le corps (chiffrables)

Exemple :

- `POST /datasets/` : Crée un nouveau jeu de données

Verbes HTTP

- GET
- POST
- PUT
- PATCH
- DELETE
- HEAD
- CONNECT
- TRACE
- OPTIONS

PUT

Fonctions :

- Met à jour une ressource

Exemple :

- `PUT /datasets/dataset/` : Met à jour un jeu de données

- GET
- POST
- PUT
- PATCH
- DELETE
- HEAD
- CONNECT
- TRACE
- OPTIONS

PATCH

Fonctions :

- Modifie partiellement une ressource

- GET
- POST
- PUT
- PATCH
- DELETE
- HEAD
- CONNECT
- TRACE
- OPTIONS

DELETE

Fonctions :

- Supprime une ressource

Exemple :

- `DELETE /datasets/dataset/` : Supprime un jeu de données

Comment transmettre des informations à une API

Le **verbe HTTP** utilisé permet d'indiquer l'action que l'on souhaite effectuer, mais :

- Il n'indique pas la **ressource** visée
- Il ne précise pas les paramètres
- Il ne donne pas suffisamment de détails sur l'action

On a donc besoin de transmettre plus d'informations à l'API

Paramètres d'url

On dispose de trois moyens pour transmettre des paramètres à une API :

- Le chemin de l'url
- La partie *requête* de l'url
- Le corps de la requête HTTP

Paramètres d’url

```
https://data.geopf.fr
  /altimetie/1.0/calcul/alti/rest/elevation.json
  ?lon=1.48
    &lat=6.2
    &resource=ign_rge_alti_wl
```

Paramètres d’url

```
https://data.geopf.fr
  /altimetie/1.0/calcul/alti/rest/elevation.json
  ?lon=1.48
    &lat=6.2
    &resource=ign_rge_alti_wl
```

Paramètres d’url

```
https://data.geopf.fr
  /altimetie/1.0/calcul/alti/rest/elevation.json
  ?lon=1.48
    &lat=6.2
    &resource=ign_rge_alti_wl
```

Paramètres d’url

```
https://data.geopf.fr
  /altimetie/1.0/calcul/alti/rest/elevation.json
  ?lon=1.48
    &lat=6.2
    &resource=ign_rge_alti_wl
```

Paramètre de chemin

```
https://rickandmortyapi.com/
  api/episode/3
```

Paramètre de chemin

```
https://rickandmortyapi.com/
  api/episode/3
```

Paramètre de chemin

```
https://rickandmortyapi.com/
  api/character/201
```

Corps de requête

```
POST /altimetrie/1.0/calcul/alti/rest/elevation.json HTTP/1.1
Host: data.geopf.fr
Content-Type: application/json
Accept: */*
Content-Length: 159

{ "lon": "6.4202", "lat": "44.916367", "resource": "ign_rge_alti_wld" }
```

Les différentes architectures

Qu’est-ce qu’une architecture d’API ?

Concrètement qu’est-ce qui change :

- Les verbes utilisés et la sémantique qui leur est attribué
- La manière de requêter la donnée
- Le format de sérialisation utilisé

Quelques architectures

- RPC
- REST
- SOAP
- GraphQL
- Sparql

Quelques architectures

- RPC
- REST
- SOAP
- GraphQL
- Sparql

Quelques architectures

- RPC
- REST
- SOAP
- GraphQL
- Sparql

Quelques architectures

- RPC
- REST
- SOAP
- GraphQL
- Sparql

Quelques architectures

- RPC
- REST
- SOAP
- GraphQL
- Sparql

Quelques architectures

- RPC
- REST
- SOAP
- GraphQL
- Sparql

La boîte à outils

Navigateur Web

Un navigateur Web est un logiciel qui (très grossièrement) :

- Fait des requêtes HTTP
- Affiche du HTML

Navigateur Web

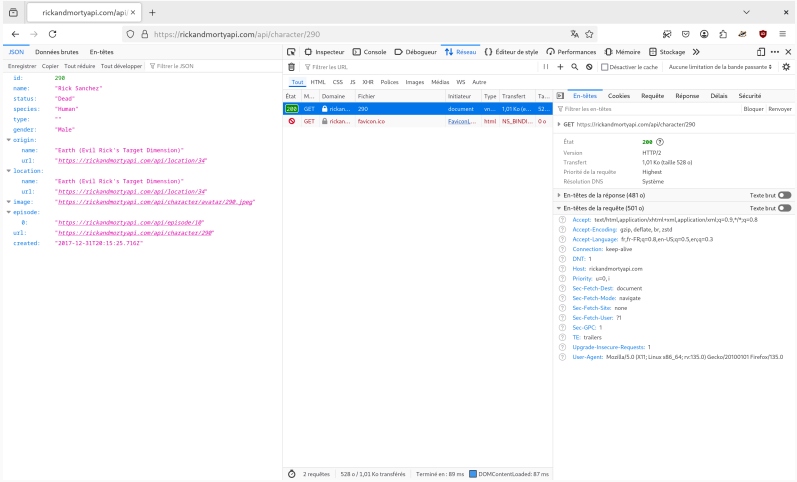


Figure 3 : Requête **GET** avec firefox

Clients graphiques

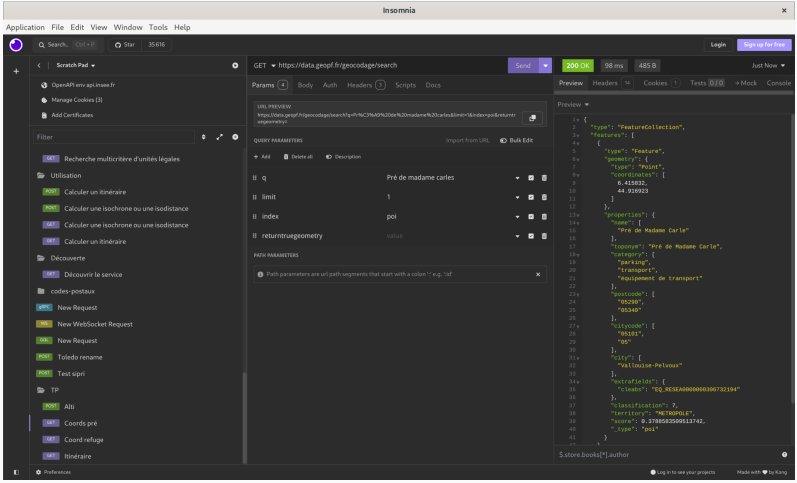


Figure 4 : Interface d'Insomnia

cURL

cURL (client for URL) est un logiciel libre développé depuis 1998. Il permet :

- De faire des requêtes HTTP `GET` et `POST`
- D'utiliser les version 0.9, 1.0, 1.1, 2 et 3 du protocole
- De gérer l'HTTPS
- De spécifier chaque *header*
- De définir le corps de la requête

C'est un bon compromis entre la complexité d'un *langage de programmation* et les limites d'un navigateur

Exemple *curl* (1)

La commande :

```
curl http://google.fr
```

Équivaut à la requête http :

```
GET / HTTP/1.1
Host: google.fr
Accept: */*
```

Soit une requête http :

- de type **GET**
- Suivant la version **1.1** du protocole

Exemple *curl* (2)

La commande suivante :

```
curl --url 'https://data.geopf.fr/altimetrie/1.0/calcul/alti/rest/elevat⌋
↪  ion.json?lon=1.48&lat=6.2&resource=ign_rge_alti_wld'
```

Contacte une API avec une requête `GET` :

```
GET /altimetrie/1.0/calcul/alti/rest/elevation.json?lon=1.48&lat=6.2&res⌋
↪  ource=ign_rge_alti_wld HTTP/1.1
Host: data.geopf.fr
User-Agent: curl/8.9.1
Accept: */*
```

Exemple *curl* (3)

La commande suivante :

```
curl --request POST --url
↪  https://data.geopf.fr/altimetrie/1.0/calcul/alti/rest/elevation.json
↪  --header 'Content-Type: application/json' --data '{ "lon": "6.4202",
↪  "lat": "44.916367", "resource": "ign_rge_alti_wld" }'
```

Crée une requête POST, dont le corps est un `json` :

```
POST /altimetrie/1.0/calcul/alti/rest/elevation.json HTTP/1.1
Host: data.geopf.fr
Content-Type: application/json
User-Agent: curl/8.9.1
Content-Length: 159

{ "lon": "6.4202", "lat": "44.916367", "resource": "ign_rge_alti_wld" }
```

Les API Wrappers

- Certains services web disposent **d'API Wrapper**.
- Un **API Wrapper** est une bibliothèque qui encapsule les appels à une API Web

Exemples :

- `https://github.com/lvaudor/hubeau` (R)
- `https://pypi.org/project/wikipedia/` (Python)

Un exemple d'API wrapper (hubeau)

La fonction `hm_station` exécute le code suivant :

```
hm_station=function(code_station){
  result=jsonlite::read_json(paste0("http://hubeau.eaufrance.fr/api/v1/h⌋
  ↪  ydrometrie/referentiel/stations",
                                    "?code_station=",code_station,
                                    "&size=1",
                                    "&format=json&pretty"))
  result=result$data[[1]]
  return(result)
}
```

Approche *no code*

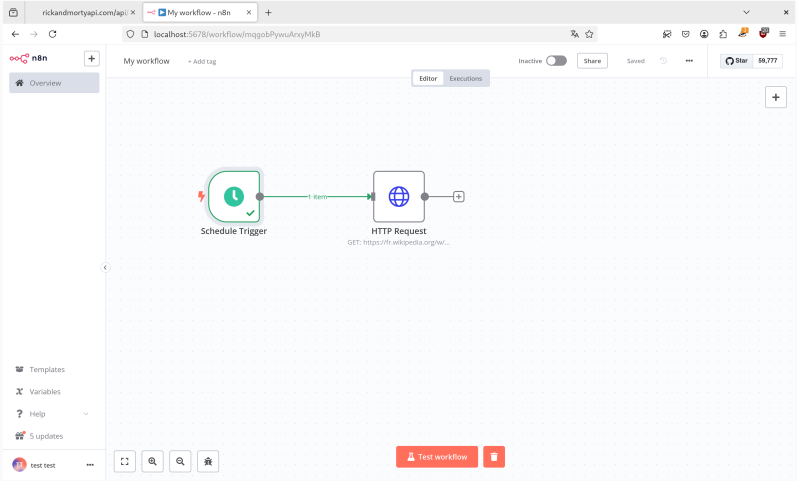


Figure 5 : Pipeline n8n d'appel à l'API de Wikipédia

Où trouver une API ?

Catalogues d'API :

- Le catalogue de `data.gouv.fr` : `https://www.data.gouv.fr/fr/dataservices/` (347 API)
- Méta-catalogue de la Commission européenne : `https://data.jrc.ec.europa.eu/dataset/45ca8d82-ac31-4360-b3a1-ba43b0b07377` (220 entrées)

Quelques problèmes courants

Pagination


```
{
  "batchcomplete": "",
  "continue":
  ↪  {"rccontinue":"20250212112749|1875779555","continue":"-||"},
  "query": {"recentchanges":
    [{"type":"edit","ns":0,"title":"Cool World (song)","pageid":53113427
    ↪  ,"revid":1275330123,"old_revid":1193077340,"rcid":1875779557,"us
    ↪  er":"Burrobert","minor":"","oldlen":4022,"newlen":4110},
    {"type":"categorize","ns":14,"title":"Category:Qajar
    ↪  mosques","pageid":61269983,"revid":1275330120,"old_revid":127533
    ↪  0092,"rcid":1875779558,"user":"Rangasyd","oldlen":0,"newlen":0}]}
}
```

Authentification

- Certaines **API** demandent une authentification
- Il existe plusieurs manières de d'authentifier
- La plus courante est l'utilisation d'une **clé d'API** que l'on doit fournir au serveur

Exemple : `https://api.nasa.gov/planetary/apod?api_key=DEMO_KEY&date=1996-12-03`

Pour résumer

Ça à l'air compliqué, mais :

- On s'en sort 90 % des cas.
- Il suffit de savoir faire une requête HTTP comme il faut…
- … et de traiter le `json` (ou `xml`) en sortie
- *Ou de demander aux ingénieurs*

Et si je veux construire mon API ?

Exemple en python avec FastAPI

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/items/{item_id}")
async def read_item(item_id):
    return {"item_id": item_id}
```

Un peu de pratique

Exemple 1

1. Se connecter au wifi : `ShellyPlugSG3-…`
2. Lancer un navigateur web
3. Ouvrir les outils de développement (`Ctrl + Maj + I`)
4. Aller dans l'onglet « réseau »
5. Accéder à l'url : `http://192.168.33.1/rpc/Switch.GetStatus?id=0`

Exemple 2 : Swagger

1. Accéder à l'API élévation du géoportail : `https://data.geopf.fr/altimetrie/swagger-ui/index.html`
2. Utilisez la route : `/calcul/alti/rest/elevation` pour obtenir l'altitude du point : 6.405647, 44.882622

Exemple 3 : IDL

Toujours sur l'API élévation du géoportail :

1. Télécharger et installer Insomnia : `https://insomnia.rest/download`
2. Récupérer l'adresse du fichier `openapi.json` : `https://data.geopf.fr/altimetrie/swagger-ui/openapi.json`
3. Dans insomnia, aller dans `scratch pad > import`
4. Copier l'adresse du fichier `openapi.json`
5. Essayez de faire la même requête que précédemment avec insomnia

Exemple 4 : Chaînage

- Accédez à l'API de géocodage de la BAN : `https://data.geopf.fr/geocodage/openapi`
  1. Utilisez la route `/search` pour récupérer les coordonnées d'un lieu de votre choix
  2. Renouvelez l'opération avec un second lieu
- Accédez à l'API itinéraire du Géoportail : `https://www.geoportail.gouv.fr/depot/swagger/itineraire.html`
  1. Utilisez la route `/itineraire` pour calculer un itinéraire entre vos deux lieux
- Utilisez le système de *tags d'insomnia* pour chaîner les requêtes

Exemple 5

https://colab.research.google.com/drive/1PIFtZ8yrHDNDJpjOc-1b8NxcFZhUx_IG?usp=sharing

Merci de votre attention