Une api c'est ...
Automate programmable industriel
[...]] est un ensemble d’éléments organisé en fonction d’un but défini par des fonctions et des Constituants.
 - A la fin des années 60, Un fabricant américain de voitures décide de remplacer les systèmes decommande à base de logique cablée (relais électrique) par une logique programmée.
 - Le premier API, modèle 084, a été inventé par Dick Morley en 1969

... au dela de la blague l'API des automates n'est pas très éloigné d'un API informatique.

Automate programmable industriel
https://www.scribd.com/document/446061670/cours-API
https://fr.wikipedia.org/wiki/Automate_programmable_industriel

L'API (pour Application Programming Interface) informatique ou interface de programmation applicative est avant out une interface (au sens objet)

Une API n’est pas :

- Un logiciel : un logiciel n’est pas une API (même s’il peut se présenter sous la forme d’une API pour faciliter l’utilisation de ses fonctionnalités).
- Une interface utilisateur : une interface utilisateur n’est pas une API (mais elle peut s’exécuter sur une interface utilisateur).
- Un serveur : un serveur n’est pas une API (mais il peut héberger une ou plusieurs API qui fournissent les données et fonctions mises à disposition par le serveur)

Encore un peu de vocabulaire lorsqu'on appelle une api on dit qu'elle est consommée, ceci peut aider à se rappeler qu'un api est "stateless", sans état.
Un api serveuse vous amène des frites, mais vous consommer de suit vous ne pouvez pas demander de la mayo avec ça, parce que vousavez déjà consommé les frites, la serveuse ne sait donc pas à quoi vous faites référence

Un facteur important pour une interface est leur stabilité.
Il est important d'indiquer le niveau de stabilité d'une api à une personnes qui l'utilise, les notions de beta, de dépréciation prennent ici toute leur importance (mais nous y reviendrons)

L'autre est la documentation !
Elle complète l'interface et doit suivre ses évolutions.

On documente :

- Les routes / requete
- La réponses

L'histoire des api :
La première mention d'une API en tant que telles est dans un document intitulé "Data structures and techniques for remote computer graphics" (structures de données et techniques pour l'infographie à distance) présenté à une conférence en 1968.

Il présentait une couche d'abstraction pour gérer via une interface des éléments d'affichage, en s'affranchissant de la connaissance du matériel.
En 1974 les API sont introduites en base de données.
Avec l'avénement du WEB et des programmes décentralisés les API sont devenus un élément incontournable de la façon dont une architecture logicielle est construite.
Réduite à ça plus simple expression une API est ce qui permet à des systèmes disparates de communiquer de façon simple et stable.

Les sytèmes d'exploitation sont aussi une sorte d'api (Cocoa pour Apple, Windows API pour ... ben windows), elles permettent d'écrire des programmes en utilisant les composants du système d'exploitation.

On peut distinguer si on veut chercher la petite bête une API d'un ABI (application binary interface) la dernière étant binaire, la première un code.
Mais finalement elle remplissent le même rôle : offrir une interface pour intéragir avec un composant fermé.

En web l'une des API les pls utilisées est le DOM on la manipule avec Javascript
DOM
https://dom.spec.whatwg.org/#what
exemple d'interface pour Document :
https://dom.spec.whatwg.org/#interface-document avec en particulier document.getElementById, qui renvoie un element :
https://dom.spec.whatwg.org/#element
Bien que ce doc soit une norme, c'est une norme "vivante" et chaque navigateur en a une implémentation différente... D'ou les problèmes de comaptibilité.

Différentes API :

- REST : L'api REST (Representational State Transfer), qui pour le web actuelement est souvent en JSON, et donc qualifiée de REST/JSON est peut être la plus récente architecture d'api mais ce n'est pas la seule parmis les autres on a :
- REST/XML : comme la précédente mais avec des réponses XML, voir peut répondre de deux façon
- SOAP (SIMPLE OBJECT ACCESS PROTOCOL) : la aussi on du xml mais normé d'une façon spécifique et propriétaire https://www.w3.org/2003/05/soap-envelope/
- JSON/RPC : RPC = remote procedural call consiste à construire une api pour qu'elle appelle des méthodes distante (ou d'un interface dédiée, win32) on est donc plus dans une logique d'action que d'états
- XML/RPC : comme précédemment... mais en xml
- ET pourquoi pas d'autre ? Rien n'empêche de définir un api… REST/CSV par exemple

Définition API
https://fr.wikipedia.org/wiki/Interface_de_programmation
https://www.ibm.com/docs/en/i/7.3?topic=interfaces-apis-overview
https://developer.ibm.com/articles/introduction-apis-and-messaging/
https://developer.ibm.com/videos/an-introduction-to-rest-apis/

Dans tous les cas une API s'occupe de données pas de look.

Avantages :

- Efficacité : Soulager le front, facilement “scalable”
- Flexible : obeit a une interface, augmenter cette interfcace rend l’api flexible
- Centraliser les données Normalise
- Automatision
- Simplification
- Intégration

Désavantges :

- Pas propriétaire / maître des données / dépendance
- Sécurité

- Coût : développement plus coûteux et en maintenance plus coûteuse cf réponse de [10:10] GUILLIN David desavantages : la mutiplicité des fonctions création des routes

Avantage REST :
- Cacheable

Désanvantage REST
- Un peu plus lent qu'un accès direct puisque communication

Concentrons nous sur le REST : une api rest est

- Simple
- sans état
- Cachable
- *client -serveur*

(et un peu plus en RESTful)
Pour le restful :
https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces#soap-vs-rest
https://restfulapi.net/

- En couche
- Code à la demande

Tests pour API :
https://addons.mozilla.org/fr/firefox/addon/rest-client-apishub/ (mais pas trop mis à jour)
https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop (mais déprécié)
Le mieux à mon sens :
Postman https://www.postman.com/downloads/

Exercice :
Utiliser des api pour avoir la météo à mydigitalschool (1 Rue Pierre de Maupertuis. 35170 Bruz)

Conversion de l'adresse en latitude longitude :
https://www.geoapify.com/

Avec lat et long :
https://www.weatherapi.com/

Pour avoir la météo dans la journée

TDD : test driven developpment

On écrite les test d’abord, puis on implémente le code

Api appleé sur : <url>/address=texte
En GET
Doit rendre la température actuellement à l’adresse donnée sur le format JSON avec
{“temperature”:”XX”} pour XX est un entier
Si address est présent mais vide, renvoyer une erreur

Interface = contrat

API <-> appli mobile
<-> site web
<-> application tierce qui consomme les données

Créer des entrée = Create
Relire ces entrée = read
Update = mettre à jo**U**r
Détruire les données = delete

Dans une API REST on va utiliser des “verbes” / méthodes HTTP :

- Create = post
- Read = get
- Update = post / put / patch
- Delete = delete

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

https://github.com/public-apis/public-apis

https://rapidapi.com/hub

https://octopush.com/api-sms-octopush/

Révision pour Jeff
Choisiur une api publique pour faire des tests (par groupe ?)
Implémenter l’exerccice postman en node

http://localhost:8081/?address=1 Rue Pierre de Maupertuis. 35170 Bruz, france