API (+ Git)

Definition
(10 Slides)

Un peu plus de détail sur Git

- git branch -M main
- git checkout

- git branch -m main <nouveau_nom>

- git remote rename origin <nouveau_nom>

- git remote add origin https://github.com/votreDepot.git

- git push -u origin main
  - main est la branche local
  - origin est la branche distante, càd, votre dans Github

- Si vous avez changer le nom ce sera, par exemple :
  - git push -u origin dev

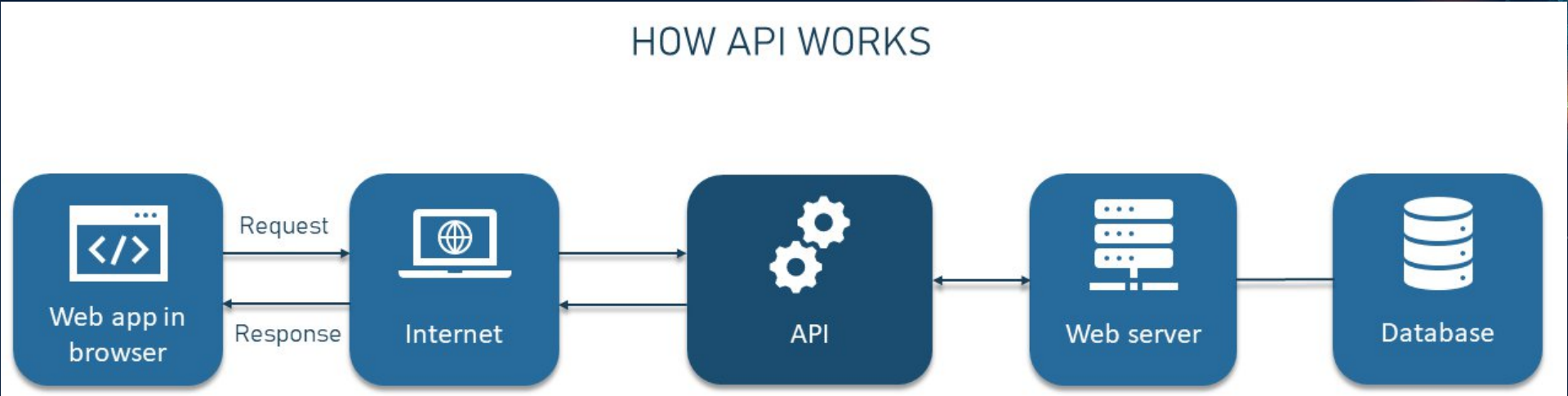
Vous vous souvenez de random.randint, random.choice, ... ?

- random est comme une « petite API locale » intégrée à Python
- On demande un nombre aléatoire avec randint ou choice et Python répond

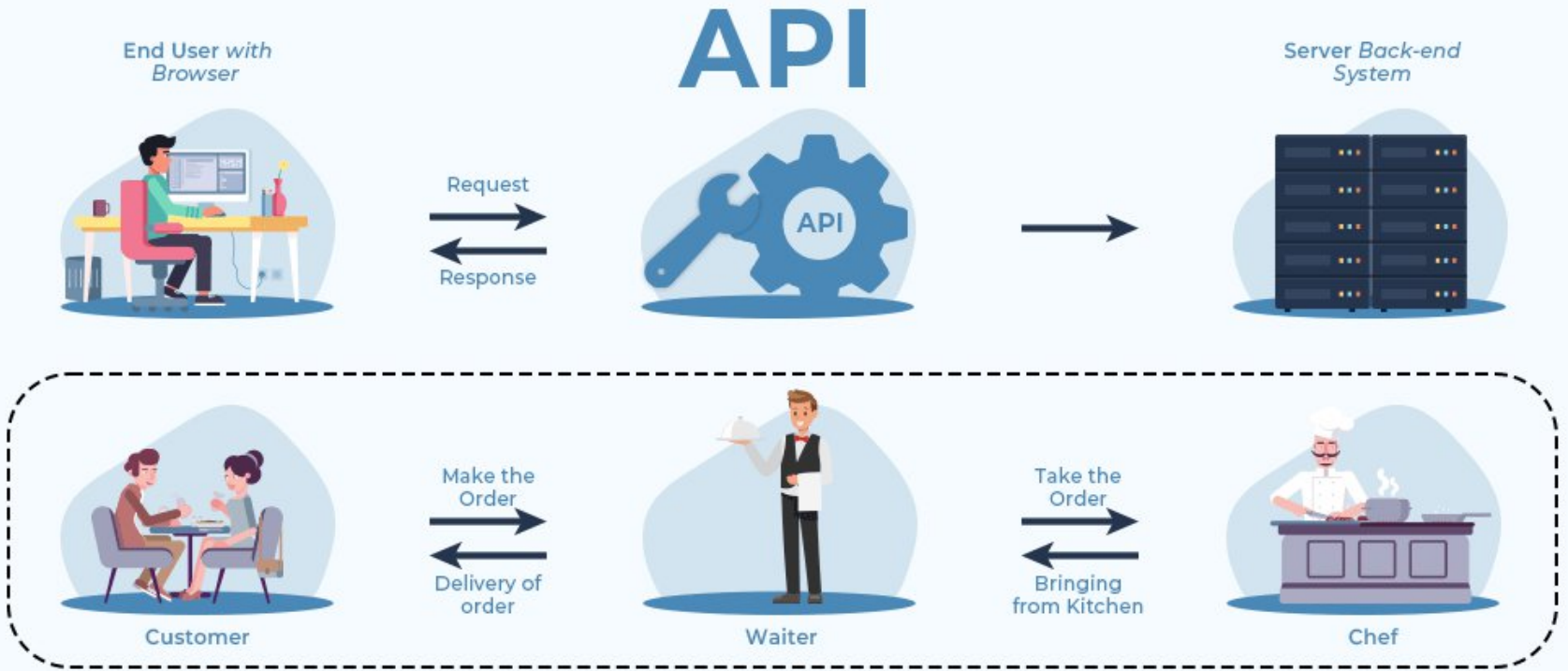
- Une API sur le web est similaire :
  - Le service est hébergé ailleurs
  - Renvoie des données réelles comme la météo, des scores de jeu, une carte Google Maps, etc.

Qu’est ce qu’une API (Application Programming Interface)

- C’est une interface entre 2 applications



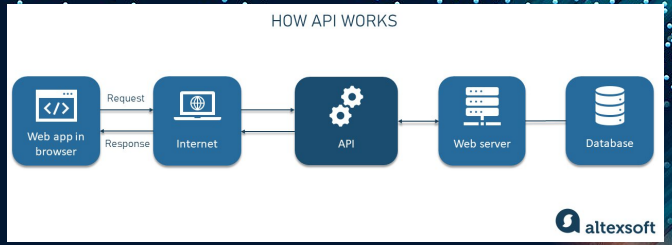
Qu’est ce qu’une API (Application Programming Interface)



Exemple d’API

- Google Maps API : Partout, sur toutes les applis de livraisons
- API Meteo
- Youtube API
- Etc.
- Note : On peut également coder sa propre API !

Exemple concret d’API Météo



- J’ai une appli mobile météo

- Lorsque je la consulte,
  l’appli ne mesure pas la température dehors
- Elle envoie en fait une requête à l’API d’un serveur météo
- Ce serveur météo répond avec le temps qu’il fait à l’API
- L’API répond à notre application
- Notre application nous affiche le temps qu’il fait

Une bonne API est une API bien documentée !

API Documentation

The API endpoint `/v1/meteofrance` accepts a geographical coordinate, a list of weather variables and responds with a JSON hourly weather forecast for 4 days. Time always starts at 0:0
URL parameters are listed below:

Parameter	Format	Required	Default	Description
latitude, longitude	Floating point	Yes		Geographical WGS84 coordinates of the location. Multiple coordinates can be comma separated. E.g. `&latitude=52.52,48.85` data for multiple locations the JSON output changes to a list of structures. CSV and XLSX formats add a column `location_id`.
elevation	Floating point	No		The elevation used for statistical downscaling. Per default, a 90 meter digital elevation model is used. You can manually set t mountain peaks. If `&elevation=nan` is specified, downscaling will be disabled and the API uses the average grid-cell height. also be comma separated.
hourly	String array	No		A list of weather variables which should be returned. Values can be comma separated, or multiple `&hourly=` parameter in th
daily	String array	No		A list of daily weather variable aggregations which should be returned. Values can be comma separated, or multiple `&daily=` If daily weather variables are specified, parameter `timezone` is required.
current	String array	No		A list of weather variables to get current conditions.
temperature_unit	String	No	`celsius`	If `fahrenheit` is set, all temperature values are converted to Fahrenheit.
wind_speed_unit	String	No	`kmh`	Other wind speed speed units: `ms`, `mph` and `kn`
precipitation_unit	String	No	`mm`	Other precipitation amount units: `inch`
timeformat	String	No	`iso8601`	If format `unixtime` is selected, all time values are returned in UNIX epoch time in seconds. Please note that all timestamp ar unix timestamps, please apply `utc_offset_seconds` again to get the correct date.
timezone	String	No	`GMT`	If `timezone` is set, all timestamps are returned as local-time and data is returned starting at 00:00 local-time. Any time zone is supported. If `auto` is set as a time zone, the coordinates will be automatically resolved to the local time zone. For multiple list of timezones can be specified.
past_days	Integer	No	`0`	If `past_days` is set, past weather data can be returned.

Les differents types d’APIs

Type d’API	Description	Exemple	
API ouvertes / Public	Accessible à tous, partage de données ou services	API météo, Google Maps	
API internes / Private	Réservée à l’usage interne d’une entreprise	API interne d’une app	
API partenaires / Partner	Accessible uniquement à certains partenaires	API PayPal pour e-commerce	
API REST	Basée sur HTTP, ressources, stateless	API GitHub, Twitter	
API bibliothèque / Library API	Interface pour utiliser des fonctions d’une bibliothèque	jQuery, NumPy	
API WebSocket	Communication bidirectionnelle en temps réel	Chat en ligne, jeux	
API GraphQL	Permet de demander exactement les données nécessaires	Shopify GraphQL	
API de streaming	Envoie des données en continu	Twitter Streaming API	
API événementielle	Déclenchée par des événements	Webhooks	
API bas niveau	Accès direct au système ou matériel	API Windows, Linux	

Comment intéragir avec les APIs

- https://pypi.org/
- Librairie requests
- S'installe via l'utilitaire pip (gestionnaire de paquet Python, comme apt, pacman, etc pour Linux)

- Créer un environnement virtuel (venv), pour travailler proprement :
- python3 -m venv nomdevotredossier
- venv\Scripts\activate # Windows
- pip install -r requirements.txt
- requirements.txt :
  - requests, black, Pytest, ruff, etc...

Exercice 1 : Catfacts API

- Reprendre votre Portfolio (Cyberfolio ?)

- Créer un bouton « Cat Facts »

- Ce bouton doit faire une requete à « https://catfact.ninja/fact »
- Le retour de l'API s'affiche sous le bouton !

- S'aider de la doc de **requests** : https://requests.readthedocs.io/en/latest/

- Push sur votre Github, lorsque vous avez fini une feature

Exercice 2 : PokéAPI

- Dans le meme esprit que catfacts :
  - Créez un bouton sur lequel vous appelez https://pokeapi.co/
  - Choisir votre pokémon
  - Afficher ses statistiques sur votre portfolio
  - Afficher son sprite (sa « photo ») sur votre portfolio, depuis l'API
  - Push sur votre Github, lorsque vous avez fini une feature

Exercice 3 : Amélioration de votre Portfolio / Site Web

- Depuis votre backend Python/Flask :
  - Exposer 2 à 3 routes max. route : chemin / « page » sur votre site
    - 1 route dédiée à l'appel API CatFacts
    - 1 route dédiée à l'appel PokéAPI
  - Bonus : Rendre la page plus agréable à consulter (CSS, images, …)
  - Push sur votre Github, lorsque vous avez fini une feature

Exercice 4 : Documenter votre projet depuis votre README.md (Github)

- Documentation depuis votre README.md

Exercice 5 : API REST plus complexe : PokeTCG API (Cartes Pokémon)

- Manipuler une API plus riche que PokeAPI : pagination, filtres, images HD, types, rarité, etc.
    - Faire un appel GET pour récupérer des cartes
    - Filtrer les cartes par :
        - type (fire, water, psychic, etc.)
        - rareté
        - HP minimum
        - Afficher dans une page dédié votre portfolio (ou site web vide, si trop compliqué) :
            - l'image de la carte
            - nom
            - HP
            - type
            - Ajouter un système de recherche textuel
            - Bonus : pagination de 20 cartes par page

Exercice 6 : API REST avec clef - The Movie DataBase (TMDB)

- Utiliser l'API TMDB (https://developers.themoviedb.org) pour enrichir le portfolio :
  - Recherche de films
  - Affichage des détails (titre, synopsis, date, image)
  - Pagination (10, 20 ou 100 films par page, l'utilisateur peut choisir)
  - Sauvegarde locale des films favoris

- Livrables attendus :
  - routes Flask pour :
    - Retourner résultats paginés (ex : GET /movies/search?q=...&page=...)
    - Afficher détails d'un film (ex : GET /movies/<id>)
    - Ajouter un film aux favoris (stockage local JSON) (ex : POST /favorites)
    - Lister les favoris (ex : GET /favorites)
  - Autres choses à faire :
    - client TMDB dans services/tmdb.py qui lit la clé via os.getenv("TMDB_API_KEY")
    - gestion d'erreurs et retries
    - Tests pytest, formatage, liniting
    - README expliquant obtention clé + config .env + ajout secret GitHub

Exercice 7 : Votre propre API - Banque / Gestionnaire de Comptes

- Créer une API bancaire minimaliste permettant de gérer :
  - des comptes
  - des dépôts
  - des retraits
  - des transferts
  - un historique des opérations
  - un fichier JSON comme base de données
  - une validation basique (interdiction d’être en négatif, etc.)

Suite énoncé slide suivante

Exercice 7 : Votre propre API - Banque / Gestionnaire de Comptes (Suite)

- Endpoints à implémenter
  - Comptes
    - Créer un compte
    - Afficher un compte
    - Supprimer un compte
  - Opérations
    - Déposer de l’argent
    - Retirer de l’argent
    - Transfert entre comptes
- Implementer un historique
- Le solde ne peut pas être négatif
- Les opérations doivent être enregistrées dans un JSON
- Validation des montants (>0)

Exercice 8 : Todolist sécurisé

- Vous êtes développeur dans une petite équipe DevOps d'une startup qui prépare sa première application web (Todolist)
- L'équipe a besoin d'un service d'authentification minimal, capable de :
  - Gérer la création de comptes utilisateurs
  - Permettre la connexion
  - Générer et vérifier un JWT (JSON Web Token) pour les sessions
  - Protéger certaines routes en vérifiant ce token
  - Permettre la gestion (CRUD) du profil utilisateur
  - Faites vos tests, formatage, linting, etc

Exercice 8 : Faites votre propre API

- Exercice libre
- Pensez à une idée d’API qui resoud un probleme, meme fictif
- Coder votre API