FORMATION DEV WEB WEB MOBILE
CCP 2 - DÉVELOPPER LA PARTIE BACK-END D'UNE APPLICATION WEB

NODE.JS

JOUR 2 : DEVELOPPER UNE API REST AVEC EXPRESS

PRESENTE PAR NATACHA DESSE

AFEC

DEROULE SEMAINE

- Découverte de Node.JS
- **Développer une API REST avec Express**
- Gestion d’erreurs et middleware
- Authentification et JWT

DEVELOPPER UNE API REST AVEC EXPRESS

LES OBJECTIFS

Créer un serveur web avec Node.js

Connaître les différents types d’API

Utiliser Express.js pour créer une API REST

Connaître les composants d’une requête HTTP

NUAGE DE MOTS

Qu’est ce qu’une API ? Que vous évoque-t-elle ?

LES APIS

C’EST QUOI UNE API ?

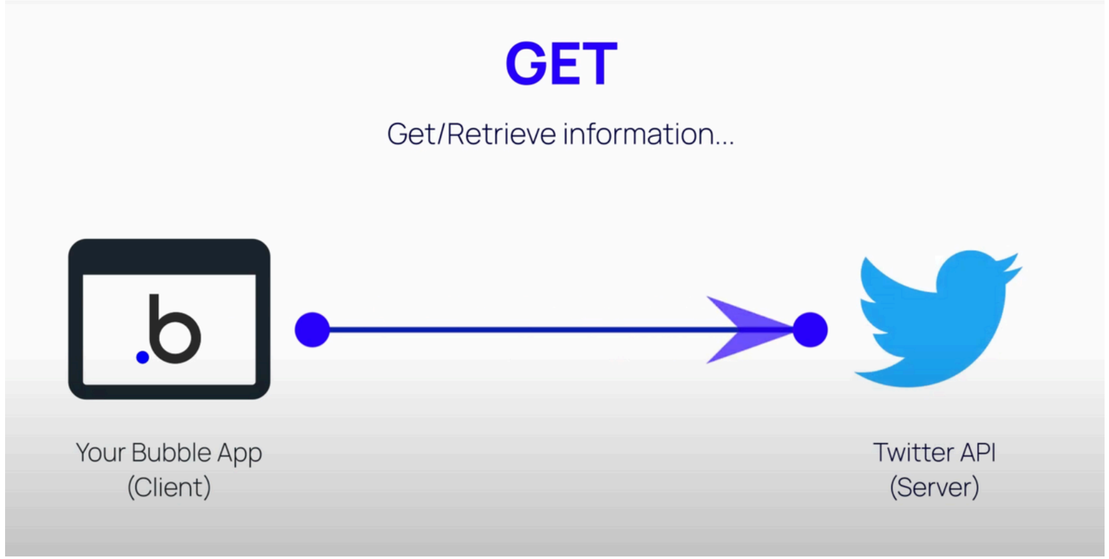
API = **A**pplication **P**rogramming **I**nterface

Une API est une **interface** qui permet à deux programmes (ou services) de communiquer entre eux.

Par exemple :

- Une app météo utilise une API météo pour demander la température.
- L'API renvoie les données : ton app n’a pas besoin de connaître tout le système météo derrière

POURQUOI UTILISER UNE API ?



- Accéder à des données externes (météo, YouTube, Twitter...)
- Séparer les responsabilités entre front-end et back-end
- Permettre l'intégration entre services
- Créer des modules réutilisables

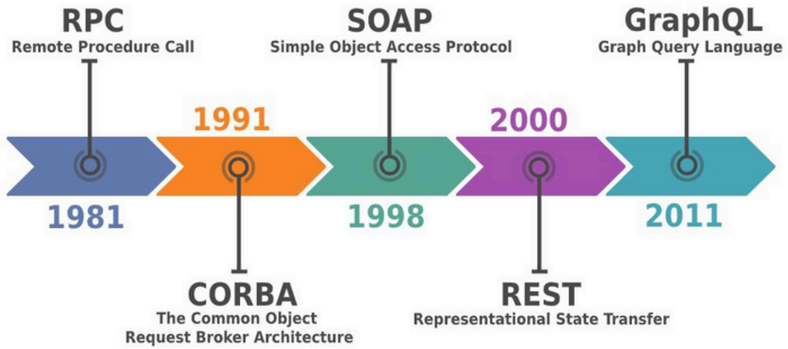
LES GRANDS TYPES D'API

🌐**API Web**
API exposée via **HTTP** / **HTTPS**, utilisée pour les échanges sur le **web**.
Composée des API Rest, API SOAP et GraphQL

🔌**API Système / Locale**
API fournies par un système d'**exploitation** ou un environnement d'exécution pour accéder à des fonctionnalités **internes**.

📡 **API temps réel (WebSocket API)**
Permet des échanges **bidirectionnels** entre client et serveur (sans avoir à rafraîchir la page).

PARMI LES API WEB...



API REST
Representational State Transfer

- Utilise les verbes HTTP : GET, POST, PUT, DELETE
- Représente des ressources (ex : /users, /products/123)
- Réponses généralement en JSON

API SOAP
Simple Object Access Protocol

- Plus ancien, basé sur XML
- Très structuré, souvent utilisé dans des environnements métiers (banques, entreprises...)
- Plus verbeux que REST

GraphQL
Graph Query Language

- Alternative à REST
- Le client peut demander exactement les données dont il a besoin
- Une seule URL d'entrée (/graphql) et une requête structurée

LES API SELON LEUR USAGE

API Publique (Open API)

Ouverte à tous
Exemples : Pokémon API, OpenWeatherMap, GitHub API

API Privée

Utilisée uniquement en interne (entre microservices, pour une appli mobile/web...)

API Partenaire

Accessible uniquement à des partenaires avec des clés d'API

API Composite

Regroupe plusieurs appels à différentes API en une seule requête.

AUTHENTIFICATION DES API

API Key

une clé secrète envoyée dans l’URL ou l'en-tête

JWT (JSON Web Token)

token signé, utilisé souvent pour les utilisateurs connectés

OAuth2

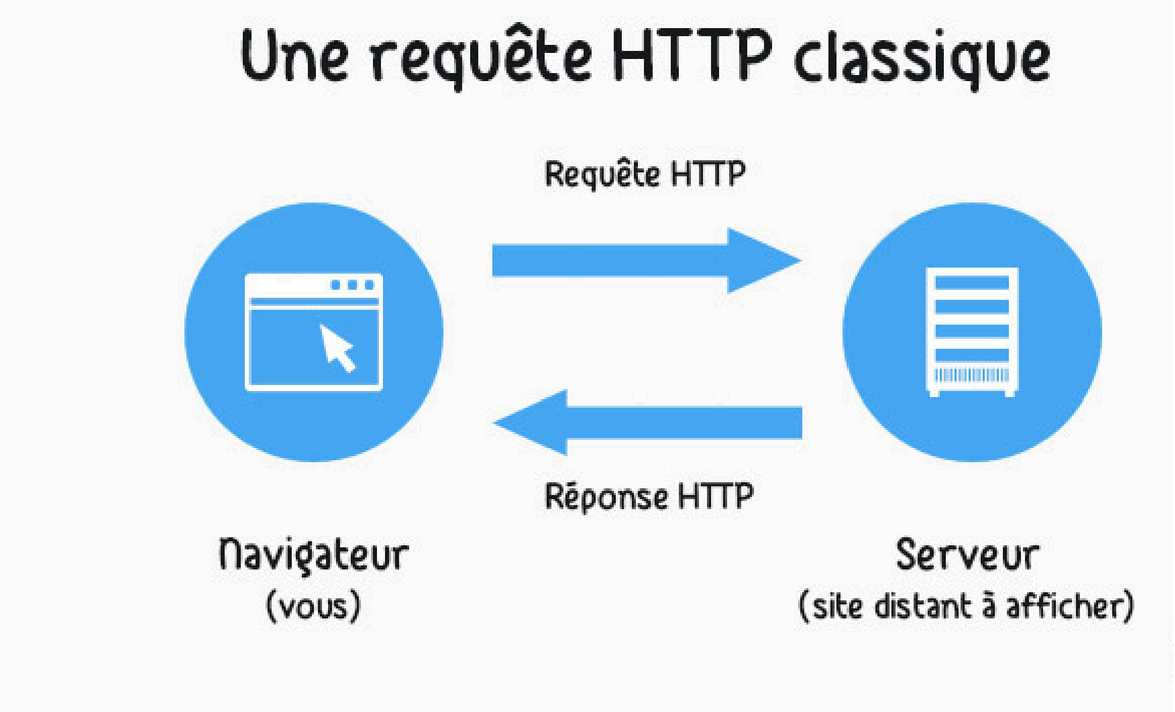
pour des accès délégués à des services (login via Google par exemple)

LES API REST

COMPRENDRE LES REQUÊTES HTTP

HTTP (**H**yper**T**ext **T**ransfer **P**rotocol) est le **protocole** qui permet la communication entre un client (généralement un navigateur ou une application) et un serveur web.

Une requête HTTP est un message envoyé par le client pour demander une **ressource** (page HTML, image, API, etc.) au serveur.



STRUCTURE D'UNE REQUÊTE HTTP

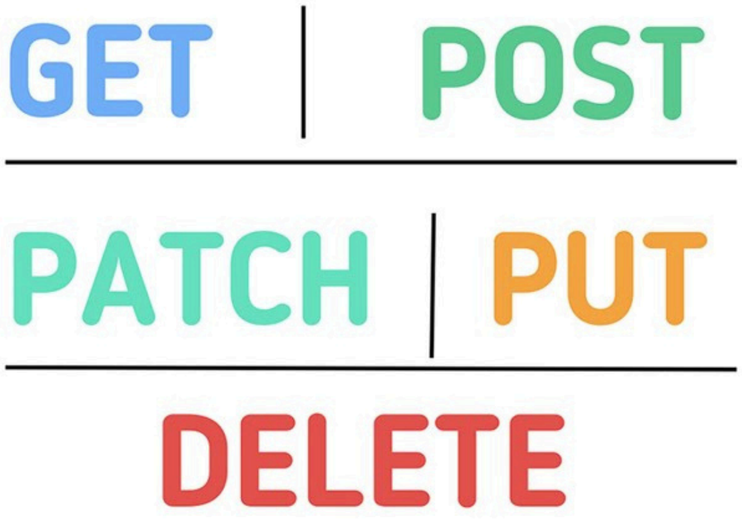
LE VERBE (méthode)
GET , POST, PUT, PATCH, DELETE

l'URL
C'est le chemin vers la ressource demandée.

Un Header (entête)
Les headers transmettent des informations supplémentaires

le body (corps)
Présent pour envoyer de la donnée

LE VERBE



Méthode	Utilisation courante
GET	Lire ou récupérer des données
POST	Envoyer des données (formulaire, création)
PUT	Mettre à jour complètement une ressource
PATCH	Mettre à jour partiellement une ressource
DELETE	Supprimer une ressource
OPTIONS	Obtenir les méthodes disponibles

L’URL

C’est le chemin vers la ressource demandée. On l’appelle aussi le **endpoint** dans le cadre d’une API

Exemple : /articles, /users/12, /api/data

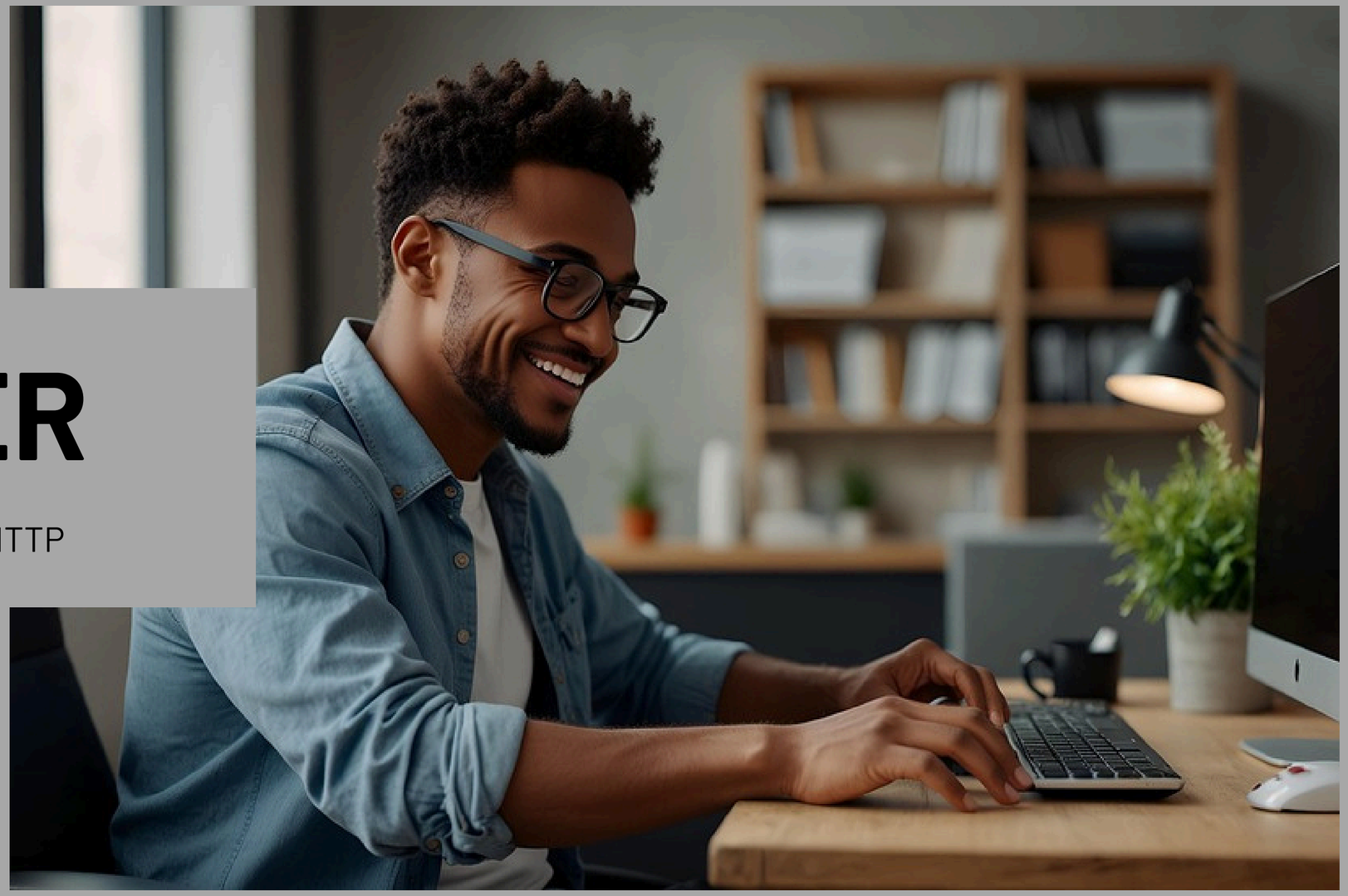
On peut y trouver des paramètres inclus dans l’url précédé d’un point d’interrogation :

Exemple : *https://api.exemple.com/v1/users/42?include=posts*

Méthode	**URL**	**Action**
GET	/users	Récupérer tous les utilisateurs
GET	/users/42	Récupérer l'utilisateur 42
POST	/users	Créer un utilisateur
PUT	/users/42	Modifier l’utilisateur 42
DELETE	/users/42	Supprimer l’utilisateur 42

ATELIER

Retrouver le bon verbe HTTP



MISE EN PRATIQUE

Pour chaque situation décrite, choisis le verbe HTTP le plus approprié :

- GET → Lire / récupérer une information
- POST → Créer une ressource
- PUT → Remplacer complètement une ressource
- PATCH → Modifier partiellement une ressource
- DELETE → Supprimer une ressource

1. Tu veux afficher la liste de tous les articles publiés sur un blog

/articles

2. Tu ajoutes un nouvel utilisateur avec son nom et son email

/users

3. Tu modifies uniquement l'adresse email d'un utilisateur existant

/users/42

4.Tu supprimes définitivement un produit de ton catalogue

/products/23

5. Tu remplaces complètement la fiche d'un article avec de nouvelles informations.

/articles/56

6. Tu veux afficher les commentaires associés à une vidéo.

/videos/99/comments

7. Tu ajoutes une nouvelle commande passée par un client

/orders

8. Tu changes uniquement le mot de passe d'un utilisateur

users/12/password

9. Tu veux accéder aux détails d'un produit spécifique.

/ products/999

10. Tu mets à jour le stock d'un produit sans toucher au reste de sa fiche

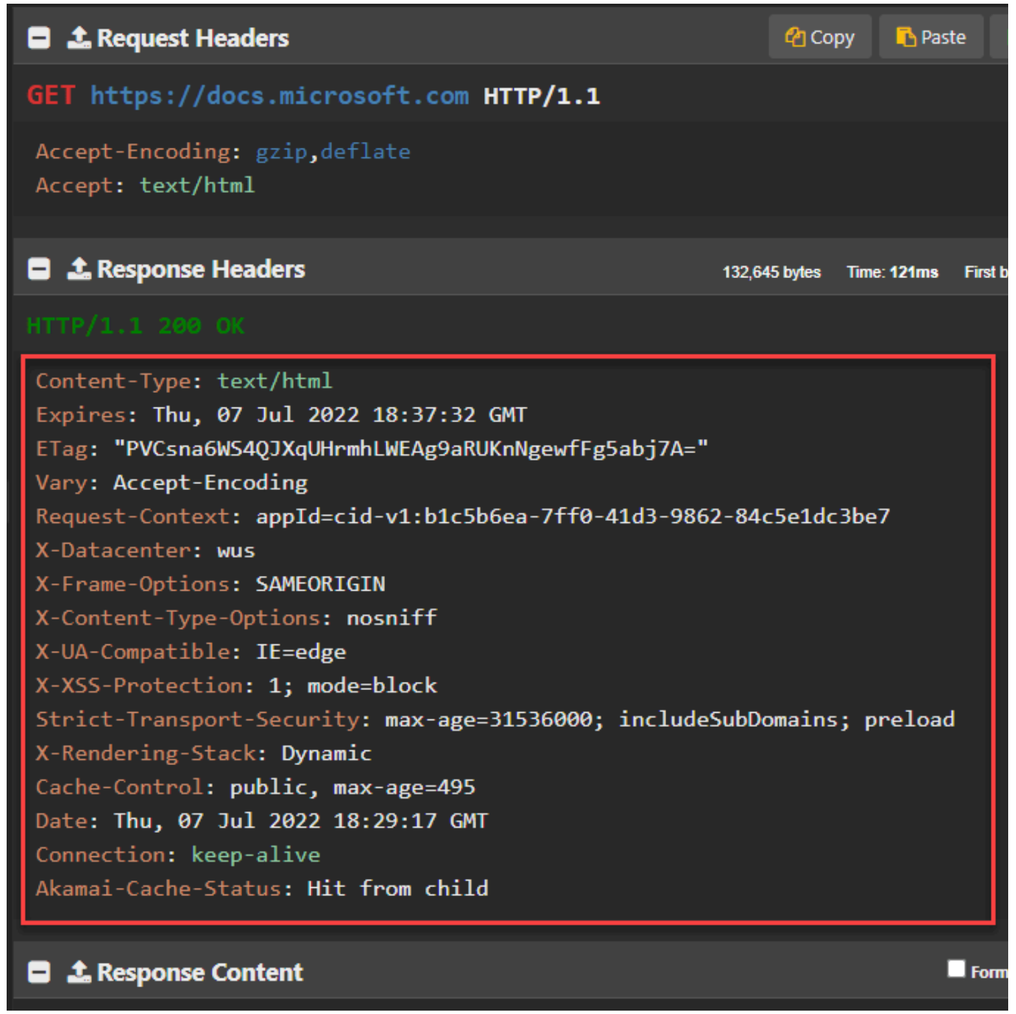
/products/15/stock

LE HEADER

Les headers transmettent des informations supplémentaires :

ex. format des données attendues, identifiants, encodage…

Cela permet d'ajouter des informations au protocole.



LE BODY

Présent uniquement pour certaines méthodes (POST, PUT, etc.).

Contient les données envoyées au serveur (ex: JSON, formulaire...).

exemple de body en JSON

CREATION D'UN SERVEUR NATIF

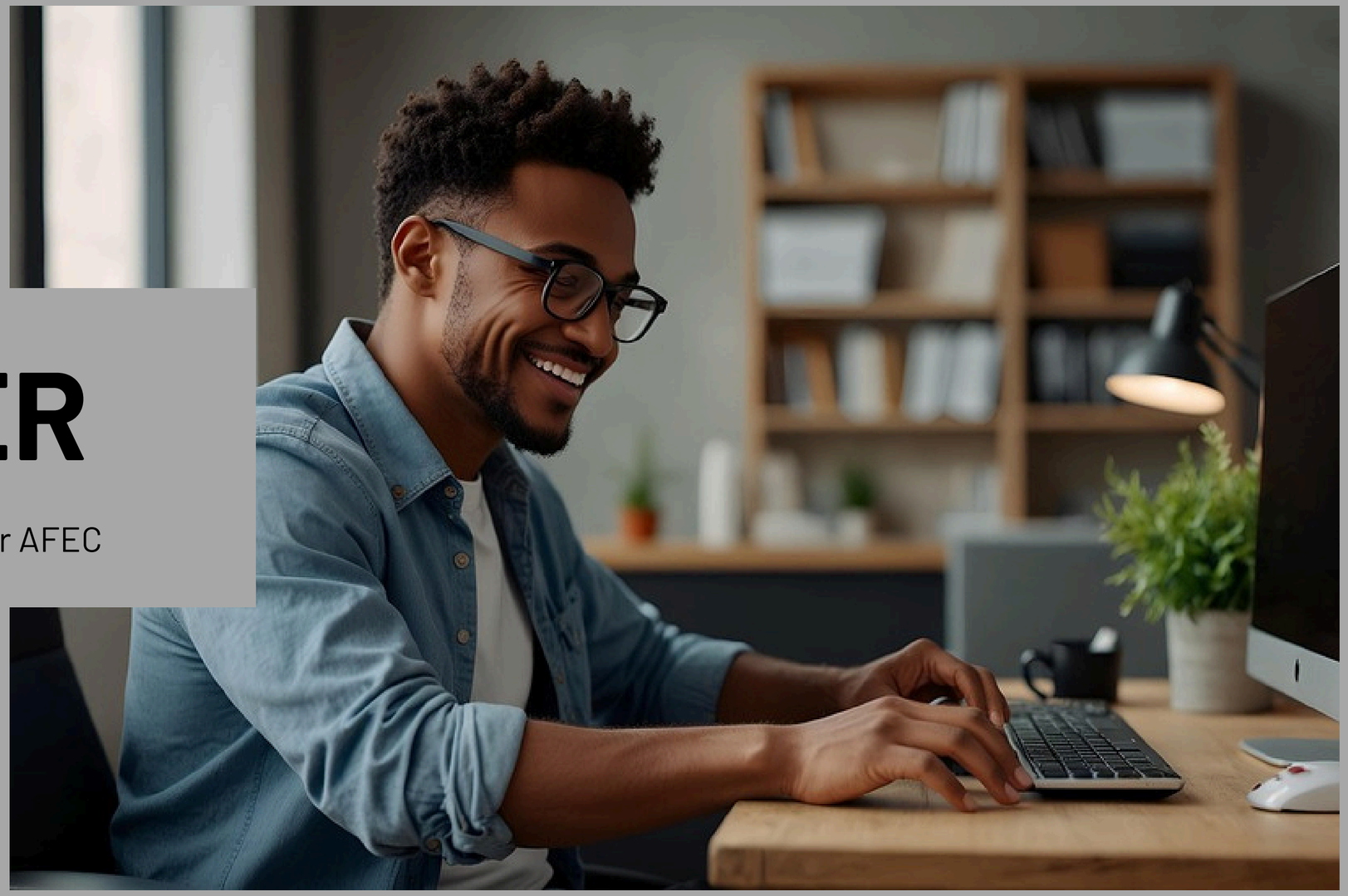
MODULE NATIF HTTP

Nous allons utiliser le module natif http de Node pour créer notre premier serveur

```
1 const http = require('http');
2
3 http.createServer((req, res) => {
4   console.log(req.method); // ex: GET
5   console.log(req.url);    // ex: /hello
6   res.end('Réponse du serveur');
7 }).listen(3000);
```

ATELIER

Lancer son premier serveur AFEC



SERVEUR PAGE AFEC

Exercice

Lancer votre premier serveur qui réalise les actions suivantes :

- Ecris dans la console le nom de la méthode suivi de son url
- Renvoie “Vous vous trouvez sur votre serveur Node” à l’adresse /
- Renvoie “Vous vous trouvez sur la page AFEC à l’adresse */afec*
- Se lance sur le port 3001

💡 **Astuce :**
res.end termine l’execution, il faut l’inclure dans une condition.

✨ **Bonus :**

Ecrivez cette phrase avec de l’ASCI ART

LIMITES DU MODULE HTTP

- Pas de gestion simplifiée des routes ou des paramètres.
- Pas de parsing JSON ou body automatique.
- Utile pour l’apprentissage ou des projets très simples.

C’est pourquoi on utilise souvent un framework comme Express ensuite.

SERVEUR EXPRESS

EXPRESS C’EST QUOI ?

Express.js est un framework web minimaliste pour Node.js.

Il simplifie la création de serveurs web et d’API en ajoutant une couche d’abstraction plus simple et rapide que le module HTTP natif de Node.

Express permet de :

- Gérer facilement les **routes** HTTP
- **Servir** des fichiers **statiques**
- Traiter les **requêtes** et **réponses** plus **rapidement**
- Créer une **API REST** en quelques lignes de code

INSTALLATION DU SERVEUR

Initialisation du serveur

```
// 1. Importer express
const express = require('express');

// 2. Créer une application Express
const app = express();

// 3. Définir une route GET
app.get('/', (req, res) => {
  res.send('Bienvenue sur notre premier serveur Express 🚀');
});

// 4. Lancer le serveur
app.listen(3000, () => {
  console.log('Serveur lancé sur http://localhost:3000');
});
```

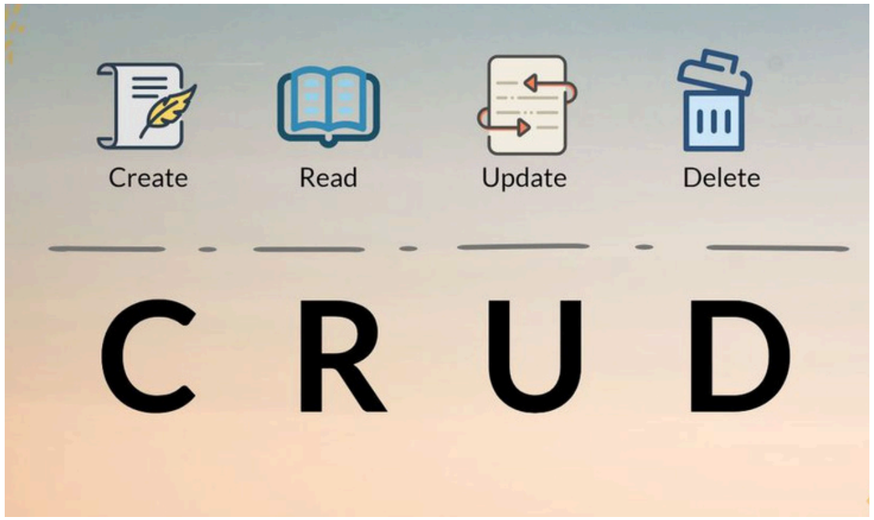
Méthodes **http**

```
app.get('/utilisateur', (req, res) => res.send('GET utilisateur'));
app.post('/utilisateur', (req, res) => res.send('POST utilisateur'));
app.put('/utilisateur', (req, res) => res.send('PUT utilisateur'));
app.delete('/utilisateur', (req, res) => res.send('DELETE utilisateur'));
```

OPERATION CRUD

CRUD est un acronyme qui décrit les quatre opérations de base que l'on peut effectuer sur des données :

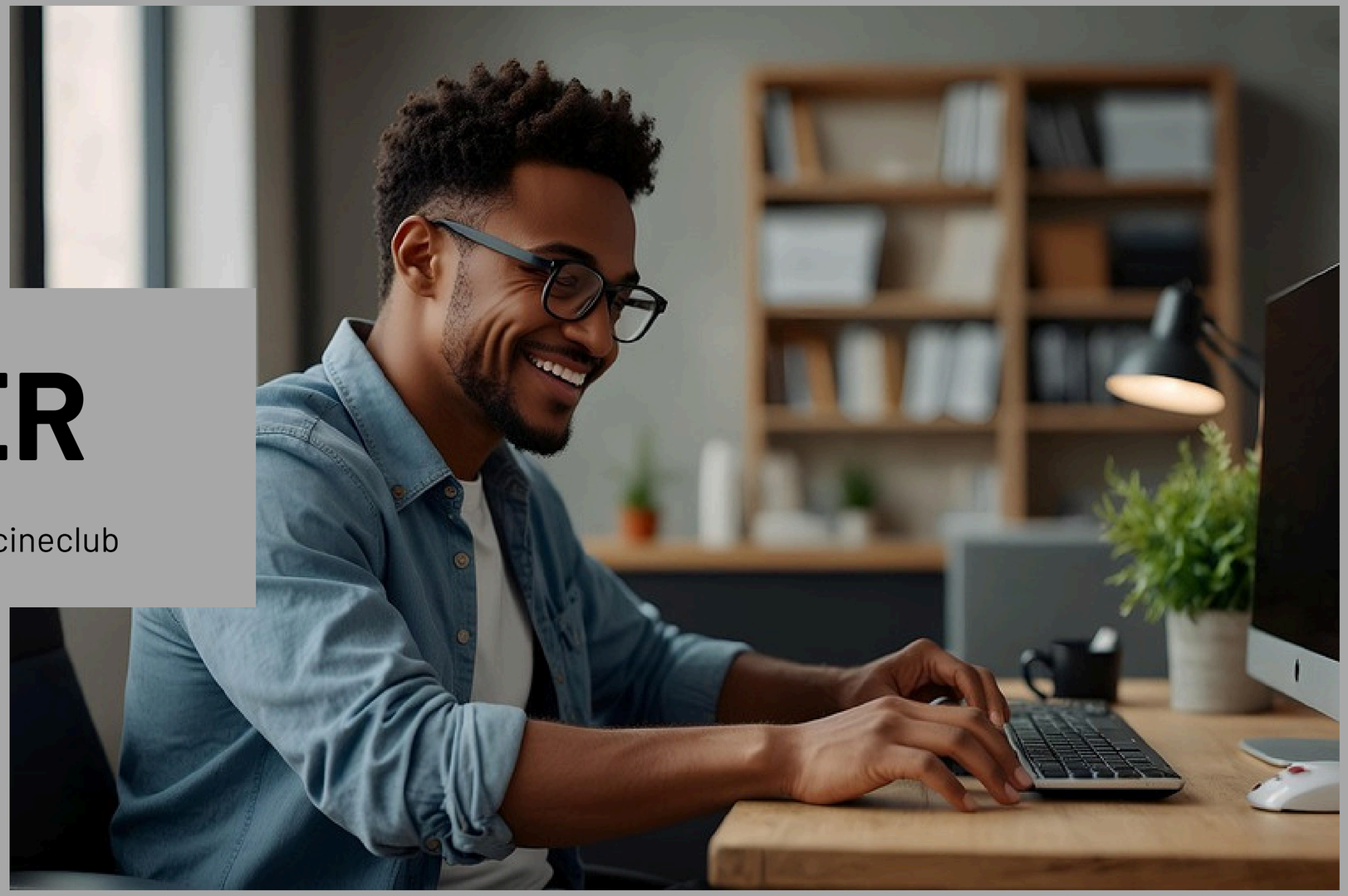
Lettre	Action	Signification en français
C	**Create**	Créer une donnée
R	**Read**	Lire ou récupérer une donnée
U	**Update**	Mettre à jour une donnée
D	**Delete**	Supprimer une donnée



Créer un article → POST /articles
Lire tous les articles → GET /articles
Lire un seul article → GET /articles/:id
Mettre à jour un article → PUT /articles/:id
Supprimer un article → DELETE /articles/:id

ATELIER

Créer une API REST pour le cineclub



REALISER UN CRUD AVEC EXPRESS

https://github.com/dessna12/02-Server-Express

🎯 Enoncé

On va réaliser notre première API REST pour le cineclub.

Suivez les instructions du README

QUIZZ