Développement WEB PHP - Séances 3-4

BUT Informatique parcours DACS

Table of contents:

📜 | Qu'est-ce qu'une API

> **INFO**
>
> Cette partie n'est pas à apprendre en détails. Cependant, **la compréhension des différents termes évoqués est nécessaire**.

API

Une **API** (Interface de Programmation d'Application) est un ensemble de règles et de conventions qui permet à différentes applications ou services de communiquer entre eux. Les API sont devenues une composante essentielle du développement logiciel moderne, permettant aux applications d'interagir avec des services web, des bases de données, ou même d'autres applications. Elles standardisent les échanges de données, facilitant ainsi l'intégration et l'extension des fonctionnalités.

> **INFO**
> Une API est côté serveur (back-end).

Qu'est-ce qu'une API ?

Une API définit des **points d'entrée** (endpoints ou URL) que d'autres applications peuvent utiliser pour interagir avec un service donné. Lorsqu'une application envoie une requête à une API, elle reçoit une réponse structurée, généralement au format **JSON** ou **XML**, contenant les données demandées ou un message de statut.

API REST (Representational State Transfer)

REST est un style d'architecture qui repose sur des méthodes HTTP standards telles que `GET`, `POST`, `PUT`, `DELETE`, etc. Les données sont généralement échangées au format **JSON** ou **XML**.

Caractéristiques :

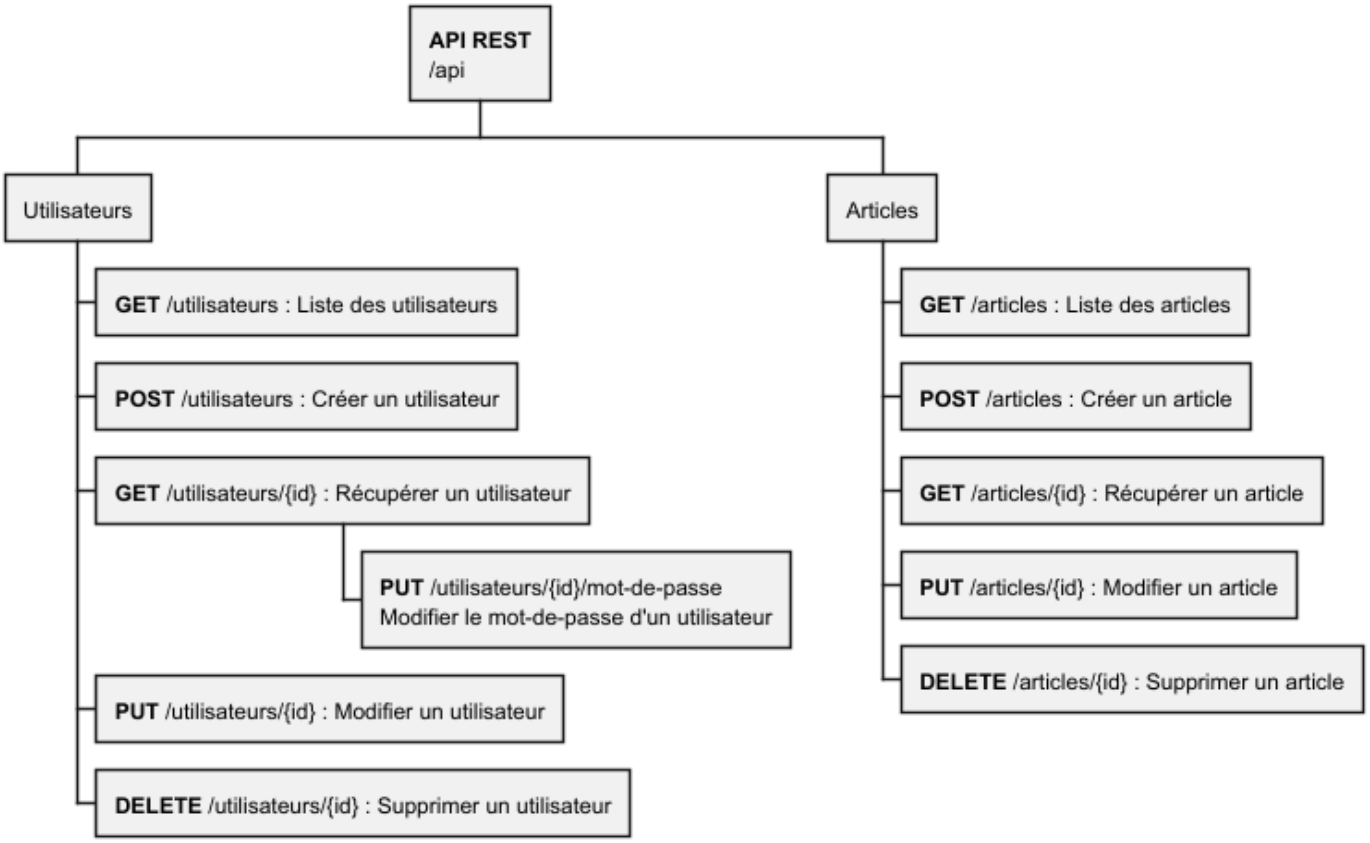
- **Stateless** : Chaque requête contient toutes les informations nécessaires, sans dépendre d'un état stocké côté serveur.
- **Scalabilité** : Très flexible et évolutif, REST est largement utilisé pour les services web modernes.

Exemple :

```
GET /utilisateurs/42
```

Ce type de requête renverrait des informations sur l'utilisateur avec l'ID 42.

Il est aussi possible de représenter une API REST sous forme d'arbre, tel que voici :


API REST
/api
Utilisateurs
GET /utilisateurs : Liste des utilisateurs
POST /utilisateurs : Créer un utilisateur
GET /utilisateurs/{id} : Récupérer un utilisateur
PUT /utilisateurs/{id}/mot-de-passe
Modifier le mot-de-passe d'un utilisateur
PUT /utilisateurs/{id} : Modifier un utilisateur
DELETE /utilisateurs/{id} : Supprimer un utilisateur
Articles
GET /articles : Liste des articles
POST /articles : Créer un article
GET /articles/{id} : Récupérer un article
PUT /articles/{id} : Modifier un article
DELETE /articles/{id} : Supprimer un article

📚 | TP 4 : Routeur

> **INFO**
>
> Différemment des TD, les TP doivent s'effectuer seul. Ainsi, la réponse que vous trouverez à la problématique peut différer d'un étudiant à l'autre.

Dans le cadre du développement d'une API REST, un routeur HTTP est un composant clé. Il permet de diriger les requêtes HTTP vers la bonne partie du code en fonction de l'URL demandée et de la méthode HTTP utilisée (GET, POST, PUT, DELETE, etc.). Ce système de routage est essentiel pour organiser les différentes routes d'une API REST, afin de bien structurer les opérations sur les ressources comme les utilisateurs, les articles, ou les produits.

Par exemple, mon endpoint (URL) `POST /utilisateur` sera relié à ma fonction PHP `createUser($request)`.

Faire le TP

1. Créer la structure du projet

- **Séance 3/TP4/**
  - `autoloader.php` (voir ajouter un autoloader)
  - `routes.php` - Script retournant un simple tableau contenant toutes les routes définies. Par exemple : `return [];`
  - public/
    - `index.php` - Script qui réceptionnera toutes les requêtes HTTP entrantes et effectuera le routage
  - framework/
    - `Method.php` - String - Enumération des différentes méthodes HTTP
      - GET, POST, PUT ou DELETE
    - `ResponseType.php` - String - Enumération des différents types de réponse
      - JSON, XML ou HTML
    - `Response.php` - Classe PHP représentant une réponse HTTP
      - *$type* - ResponseType
      - *$content* - String - contenu (corps) de la réponse
      - *$code* - Int - code de retour
    - `Route.php` - Classe PHP représentant une route
      - *$method* - Method
      - *$endpoint* - String - URL du endpoint, par exemple : /users
      - *$controller* - String - Nom de la classe du contrôleur associé, par exemple : si mon contrôleur s'appelle TestController je pourrais utiliser TestController::class
      - *$function* - String - Nom de la fonction du contrôleur à appeler
  - controllers/
    - `MyTestController.php` - Classe PHP représentant une ressource REST. Cette classe ne contient QUE des fonctions ayant pour paramètre `array $request` et ne retournant que `new Response(...)`
    - Autres contrôleurs...

2. Ajouter un autoloader

Un autoloader est un fichier qui charge automatiquement toutes les classes PHP de votre code, simplifiant ainsi leur import et leur utilisation.

À la racine de votre projet, ajouter ce fichier :

autoloader.php

```php
<?php

/* Code from internet, automatically load classes if they are currently
not loaded */

// https://www.php.net/manual/en/language.oop5.autoload.php

spl_autoload_register(function ($className) {
    $filename = __DIR__ . DIRECTORY_SEPARATOR . str_replace('\\', '/',
$className) . '.php';
    if (file_exists($filename)) require_once($filename);
});
```

Pour utiliser l'autoloader, il suffira d'ajouter cette ligne au tout début de votre `index.php` :

```php
// Import the autoloader, which loads all the namespaces and classes
require(__DIR__ . '/../autoloader.php');
```

3. Lancer le serveur

Une fois à la racine de votre projet `/Séance 3/TP4/`, il faut utiliser la commande suivante :

```
php -S localhost:8080 -t public
```

> **ATTENTION**
>
> À noter que le dossier servi est `public` et non `TP4`. Cela permet d'avoir des accès utilisateurs différents, mon dossier public va être accessible par quasi-tout me monde (774). Là où mon dossier TP4 n'est accessible que par l'utilisateur courant (700).
>
> Cependant, cela n'a pas d'importance dans notre TP

Informations utiles

- Utiliser l'autoloader : https://www.php.net/manual/en/language.namespaces.definition.php
  - N'oubliez pas d'ajouter un namespace au début de vos classes et énumérations
- POO en PHP : https://www.php.net/manual/en/language.oop5.basic.php
- Enumérations typées en PHP : https://www.php.net/manual/en/language.enumerations.backed.php
- Importer une variable d'un autre fichier (`routes.php`) : https://www.php.net/manual/fr/function.require.php

Concernant le fichier `index.php`

Script qui réceptionnera toutes les requêtes HTTP entrantes et effectuera le routage.

Ce script devra ainsi, dans l'ordre :

1. Charger l'autolaoder
2. Importer le tableau des routes (`routes.php`),
3. Récupérer le endpoint et la méthode utilisées par le client
4. Comparer toutes les routes définies à la requête du client (endpoint et méthode identique)
   - Si la route est identique
     a. Instancier le contrôleur (`$controller = new $route->controller;`)
     b. Appeler la fonction du contrôleur (`$response = $controller->{$route->function}($_REQUEST);`)
     c. Récupérer la réponse
   - Si la route n'est pas identique
     a. Créer une réponse 404
5. Ajuster les headers de la réponse (Code, Content-type)
6. Renvoyer la réponse

Vous pouvez demander de l'aide si nécessaire.

📝 | QCM 1 - Bases du PHP

Petit examen concernant les connaissances acquises durant les séances précédentes :

- Propriétés de PHP
- Architecture d'une application WEB