

PLOTS LUMINEUX



# PROCÉDURE API RESTFULL



# Présentation :

Cette architecture est née d'un besoin de simplifier les accès aux données des différents services, tout en assurant une parfaite sécurité.

Il serait dangereux de laisser les utilisateurs Internet accéder directement à la base de données du site :  
risque de piratage  
risque de modifications  
obligation de créer des comptes génériques pour gérer les droits

La solution est un ensemble de codes réagissant aux requêtes définies dans le protocole HTTP.

L'intérêt est de permettre à n'importe quel client, d'accéder aux données que l'on veut rendre accessible. La description de ces données utilise JSON, pour ne plus être dépendant de l'encodage de chaque base de données.

# URI :

Une URI sert de point d'entrée pour un type de ressource. Il est important de formater les URI de manière logique, en utilisant correctement la syntaxe des URL HTTP. On utilisera l'adresse du serveur suivie de la ressource souhaité puis si besoin des paramètres. Exemples :

- Obtenir une liste de livres :

<http://localhost:3000/livres>

- Avec un filtre sur la liste :

<http://localhost:3000/livres/type/roman>

Obtenir un livre à partir de son id :

<http://localhost:3000/livres/87>

Obtenir des commentaires sur ce livre :

<http://localhost:3000/livres/87/comments>

# Méthodes :

Le protocole HTTP propose plusieurs verbes et méthodes, il suffit de les employer correctement :

- Créer : POST
- Obtenir : GET
- Mettre à jour : PUT
- Supprimer : DELETE

# Routes :

Les routes seront les chemins d'accès depuis la requête vers les méthodes d'accès aux données.

Le routeur va se charger de choisir la méthode à exécuter en fonction de la requête reçue.

Une requête est un message qui va du client au serveur. Elle est structurée de la manière suivante :




































Voici un exemple pour un produit. Le type sera la manière de sérialiser les données à l'intérieur de la réponse ou de la requête :

URI	Méthode	Verbe	Type	Description
/produits	getProduit()	GET	JSON	Récupérer toutes les produits.
/produits/{id}	getProduit(\$id)	GET	JSON	Récupérer les données d'un seul produit
/produits	createProduit(\$produit)	POST	JSON	Insérer un nouveau produit dans la base de données.
/produits/{id}	updateProduit(\$produit, \$id)	PUT	JSON	Mettre à jour le produit dans la base de données.
/produits/{id}	deleteProduit(\$id)	DELETE	JSON	Supprimer un produit de la base de données.

# Exemple :

Il faut avant toutes choses créer une table sql que l'on souhaite.

Ici on crée la table joueurs dans la base de données plots lumineux qui contient l'id, l'email, l'identifiant ainsi que le mot de passe d'un joueur.

				id	password	email	identifiant
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	7	\$2y\$10\$oY.3mLDTzMpIQRWpIL7nVOFYtu5K3mlwceUMPx1d3Kb...	test@test	test
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	6	\$2y\$10\$66V7G2XIZ4Sv3tThEf6NJuC3yyTVv395/sL4Tdw3Pm...	bobibo@gmail.com	bobibo11
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	8	\$2y\$10\$IIJ0yZyyvSBtwrwdfBCYh.IJ2tY1UKEbdE3Grv4nocF...	2525@gmail.com	2525
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	9	\$2y\$10\$bVw0Od0hOhAhtfuhKE8wS.LSxKdClz/VTwcjT/2dYca...	dedede@gmail.com	dedede
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	10	\$2y\$10\$2W97jnVDXnoaYJXglQqxC.cd3ltTUSHo8WriEDs7xCu...	killian2003.houard@gmail.com	Kima76
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	11	\$2y\$10\$HzxHzrGMAVGgqstUnKvon.s1oNLX2JMOAajADptKAIS...	marceau.voisin@gmail.com	marceau4444
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	13	\$2y\$10\$WGs0pw6.XYOzuPawKLWoOefxDhpuSIW6er6SBO/e3s...	bobibo111@gmail.com	bobibo111
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	14	\$2y\$10\$nD9qcllJkedAmqTX5rrtk./EYpOLjMQ6tpCY4KUFkb...	bobibo1111@gmail.com	bobibo1111
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	15	\$2y\$10\$4nzp39gekLNAu.ZmClpBXu6ToQT6uH9/ya47b/HieoK...	test2805@gmail.com	test2805
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	16	\$2y\$10\$V41jlF6wfd/DowgECGUQNOTkLihsWW59WtO7o.Ff74T...	test2805@com	jordanL
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	17	\$2y\$10\$J3uB5Zxey9cm5ZdP80rvFOOpsQnuDtX6JSSP9zK0MT...	Azerty*123@gmail.com	Azerty*123

# Exemple :

```
<?php

class DataJoueurs {

    private $bdd=NULL;

    public function __construct(){

    }

    private function connexionBDD(){

        try{
            $this->bdd = new PDO('mysql:host=localhost;dbname=plotslumineux', 'adminplots', 'Azerty*123');
            return true;
        }catch (Exception $e){
            return false;
        }
    }

    public function getJoueur($id) {
        $reponse = array('statut' => 1, 'message' => 'ok');

        // Connexion à la base de données
        if (!$this->connexionBDD()) {
            $reponse['statut'] = 0;
            $reponse['message'] = 'connexion bdd impossible';
            return $reponse;
        }

        try {
            // Préparation de la requête
            $sql = 'SELECT * FROM joueurs WHERE id = :id';
            $s = $this->bdd->prepare($sql);
            $s->bindParam(':id', $id, PDO::PARAM_INT);

            // Exécution de la requête
            $s->execute();

            // Vérification de l'exécution de la requête
            if ($s->rowCount() == 0) {
                $reponse['statut'] = 0;
                $reponse['message'] = 'Aucun joueur trouvé pour cet ID';
                return $reponse;
            }

            // Extraction des données
            $donnees = $s->fetch(PDO::FETCH_ASSOC);

            // Attribution des données à la réponse
            $reponse['joueur'] = $donnees;
        } catch (PDOException $e) {
            $reponse['statut'] = 0;
            $reponse['message'] = 'Erreur PDO : ' . $e->getMessage();
        }
    }
}
```

Création d'une classe dataJoueurs qui va contenir les méthodes pour accéder à la table SQL créée précédemment

# Exemple :

routeurJoueur.php

```
1  k?php
2  // inclure les fonctions d'accès à la base de données
3  require("dataJoueurs.php");
4
5  // Récupérer le verbe (méthode http) de la requête
6  $request_method = $_SERVER["REQUEST_METHOD"];
7
8  $dataJoueurs = new DataJoueurs();
9
10 // test de la méthode de la requête http
11 switch ($request_method) {
12     case 'GET': // Si c'est une méthode GET
13         // Récupérer les données du joueur
14         if (!empty($_GET["id"])) // Si la propriété id existe
15         {
16             $id = intval($_GET["id"]);
17             $reponse = $dataJoueurs->getJoueur($id); // Appel de la méthode getJoueur($id)
18
19             // Vérification si le joueur est trouvé
20             if ($reponse['statut'] == 1) {
21                 // création de la réponse en JSON
22                 header('Content-Type: application/json');
23                 echo json_encode($reponse, JSON_PRETTY_PRINT);
24             } else {
25                 // Si le joueur n'est pas trouvé, renvoyer un message d'erreur
26                 header("HTTP/1.0 404 Not Found");
27                 echo json_encode($reponse, JSON_PRETTY_PRINT);
28             }
29         } else {
30             // Si aucun identifiant de joueur n'est fourni, renvoyer un message d'erreur
31             header("HTTP/1.0 400 Bad Request");
32             echo json_encode(array("message" => "ERREUR"), JSON_PRETTY_PRINT);
33         }
34         break;
35
36     default:
37         // Requête invalide
38         header("HTTP/1.0 405 Method Not Allowed");
39         break;
40 }
41 ?>
42
```

## Exemple :

On doit créer un fichier routeurJoueur.php.

En début de fichier, notre programme inclura le fichier dataJoueurs.php pour avoir accès aux méthodes d'accès aux données.

Le fichier routeurJoueurs.php va analyser le contenu de la requête et faire appel à la méthode correcte. Dans un premier temps le programme va récupérer la méthode de la requête http puis avec un switch case choisir la méthode à appeler. Dans le cas d'une méthode GET, alors il faut tester si un id a été ajouté à l'URI pour savoir si le client veut la liste ou juste un élément.

Pour accéder aux routes écrites précédemment, il faudrait écrire l'URI suivant dans votre navigateur : `http://localhost/<notreProjet>/`



# Exemple :

```
void DataJoueurs::getJoueur(const QString &id)
{
    connect(manager, &QNetworkAccessManager::finished,
             this, &DataJoueurs::lireDataGetJoueur);
    QNetworkRequest request(QUrl("http://192.168.190.4:38080/API/routeurJoueurs.php?id=" + id));
    manager->get(request); // récupérer le joueur spécifié par identifiant
}

void DataJoueurs::lireDataGetJoueur(QNetworkReply *reply)
{
    QString dataJson(reply->readAll());
    qDebug() << "Data Json Joueur : " << dataJson;
}
```

Puis on utilisera de cette manière la méthode  
getJoueur( ) dans QT