RAPPORT API REST FULL ET INTERFACE DE CONSOMMATION

REPUBLIQUE DU SENEGAL
UN PEUPLE-UN BUT-UNE FOI
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
DIRECTION GENERALE DE L'ENSEIGNEMENT SUPERIEUR

RAPPORT API REST FULL ET INTERFACE DE CONSOMMATION

MASTER 1 EN RESEAUX ET TELECOMMUNICATION(M1RT)

Présenté par :

M[le]. Dekezandji Therese D'avila

Encadrant :

M. Samuel OUYA
Docteur en Télécoms & Réseaux

RAPPORT API REST FULL ET INTERFACE DE CONSOMMATION

TABLE DES MATIERES

Objectif

- ✓ connaître ce qu'on appelle api
- ✓ connaître quelques cas d'utilisation d'api
- ✓ connaître l'importance des api
- ✓ connaître les environnements et paradigmes de programmation des api
- ✓ connaître les différentes règles à respecter pour avoir une application REST full
- ✓ connaître les différents langages de programmation adaptés aux api REST
- ✓ savoir comment mettre en œuvre des prérequis pour développer des api en php
- ✓ connaître le format de donnée json
- ✓ savoir développer des API REST et leur interface de consommation
- ✓ savoir comment faire la réécriture des url d'apache
- ✓ savoir sécuriser son API

1.1.1

2 Cours 1 : Cas d'utilisation et importance des API

Définition d'API : Application Programming interface, c'est un ensemble de programme et de protocole qui permet à deux ou plusieurs applications de communiquer entre elles

2.1 Nous sommes à l'ère de micro services :

a-il faut une interface de communication entre les applications(API)
b- il faut un format universel d'échange de données
b1- le format xml
b2- le format json

NB : json est plus souple que xml

Quelques cas d'utilisation d'api

cas 1 : opérateurs de télécoms utilisent des api pour permettre a leurs abonnés d'utiliser leur crédit pour acheter des articles.
L' api de l'opérateur doit avoir 4 paramètres
parametre1=le numéro de téléphone du client
parametre2=le prix de l'article
parametre3= le login du boutiquier
parametre4=mot de passe du boutiquier
au retour, il y a plusieurs cas :
Cas 2 d'utilisation : Paiement de facture avec son crédit
cas3 d'utilisation : interfaces de gestion de conteneurs
cas4 : programmation des réseaux(SDN)
exemple : on utilise des switches virtuels et des contrôleurs (opendaylight,ryu)

2.2 Les environnements et paradigmes de programmation des api

Il y'a principalement deux manières de mettre en place des API :
-utilisation du protocole soap
-utilisation du concept REST qui s'appuie sur le protocole HTTP

NB : le paradigme REST s'impose aujourd'hui

2.3 Les règles à respecter pour avoir une applications REST full

il ya 4 réglés de bases à respecter :
Règle n°1 : l'URI comme identifiant des ressources
Règle n°2 : les verbes HTTP comme identifiant des opérations
Règle n°3 : les réponses HTTP comme représentation des ressources
Règle n°4 : un paramètre comme jeton d'authentification

2.4 Les langages de programmation adaptés aux api REST

4.1- python flask
4.2- nodejs qui permet de programmer en javascript coté serveur
4.3- php
4.4- java

2.5 Préparation de l'environnement de développement d'api REST en php

Paquets à installer :
php, php-mysql,php-curl, php-json, libapache2-mod-php
clients HTTP évolués : curl, postman
installation de postman sous linux :
apt-get install snap
snap install postman

2.6 Conclusion :

Avoir des compétences en développement d'API permet aujourd'hui de s'affirmer dans beaucoup domaines :
• Banques
• Services opérateurs télécoms
• intégration de micro services
• cloud computing et virtualisation
• Réseaux gérés par logiciels (SDN)

3 Cours 2 : Mise en œuvre des prérequis pour développer des api en php

Mise en œuvre de la fiche TP1

Dont l'objectif est :

a- Savoir installer, se connecter et faire des requêtes SQL de base
b- savoir utiliser les balises html table
c- Savoir générer des tableaux html à partir d'un code php

d- savoir utiliser les extensions php de manipulations de bases de données
e- Apprendre la bonne pratique en termes de programmation

1- on crée la base de données banque et on y crée la table client et enfin on ajoute le president Macky Sall comme suit :

```
| Database             |
+----------------------+
| information_schema   |
| DAVILA               |
| EC2LT1               |
| callcenter           |
| etudiant             |
| m1                   |
| mysql                |
| nom_de_base          |
| openfire             |
| performance_schema   |
| phpmyadmin           |
| sgbs                 |
| sys                  |
| verrou               |
+----------------------+
14 rows in set (0,00 sec)

mysql> create database banque;
Query OK, 1 row affected (0,36 sec)

mysql> show databases;
+----------------------+
| Database             |
+----------------------+
| information_schema   |
| DAVILA               |
| EC2LT1               |
| banque               |
| callcenter           |
| etudiant             |
| m1                   |
| mysql                |
| nom_de_base          |
| openfire             |
| performance_schema   |
| phpmyadmin           |
| sgbs                 |
| sys                  |
| verrou               |
+----------------------+
15 rows in set (0,00 sec)
```

```
mysql> use banque
Database changed
mysql> create table client (id int not null auto_increment primary key, prenom varchar(10), nom varchar(20), numcompte varchar(20), solde varch
ar(30), code varchar(20));
Query OK, 0 rows affected (0,48 sec)
```

RAPPORT API REST FULL ET INTERFACE DE CONSOMMATION

```
mysql> insert into client(prenom,nom,numcompte,solde,code) values("Macky", "sall", "1001", "100000", "1111");
Query OK, 1 row affected (0,14 sec)

mysql> select *from client;
+----+--------+------+-----------+--------+------+
| id | prenom | nom  | numcompte | solde  | code |
+----+--------+------+-----------+--------+------+
|  1 | Macky  | sall | 1001      | 100000 | 1111 |
+----+--------+------+-----------+--------+------+
1 row in set (0,00 sec)
```

La base étant prête, on commence par créer un dossier dans le document root de notre serveur web pour y développer notre api :

```
root@therese:~# cd /var/www/html/
root@therese:/var/www/html# mkdir tpapi
```

```
root@therese:/var/www/html# cd tpapi/
root@therese:/var/www/html/tpapi#
```

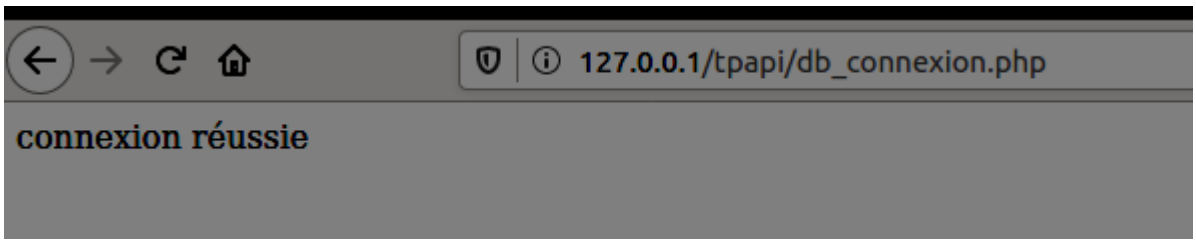
créons le fichier db_connexion.php de connexion a notre
banque avec comme paramètre de connexion :
user=root, mdp=passer, nombase=banque

contenu du fichier db_connexion.php :

```
  GNU nano 2.5.3                                  Fichier : db_connexion.php

<?php
        $servername = '127.0.0.1';
        $user = 'root';
        $pass = 'passer';
        $base = 'banque';

        $conn = mysqli_connect($servername,$user,$pass,$base);

                if($conn){
        echo ("connexion réussie \n");
                        }
                else {
                        echo "connexion impossible";}
?>
```

On teste la connexion a la base de donnée à travers un navigateur comme suit :

RAPPORT API REST FULL ET INTERFACE DE CONSOMMATION



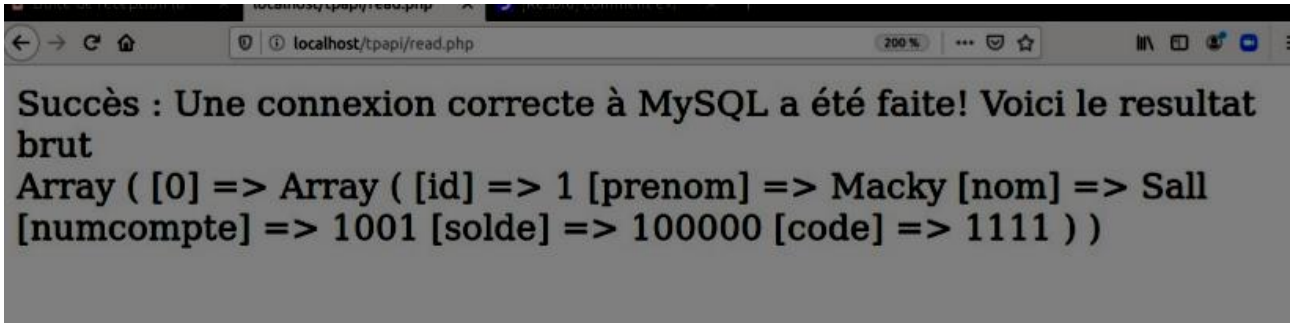
On passe à la création de la fonction read.php qui doit afficher le contenu de la table client sous forme de tableau associatif
Réflexion :
Pour faire une requête en php sur une base de type mysql et récupérer le résultat sous forme de tableau associatif, on procède ainsi :
1- On se connecte à la base de données
2- on prépare la requête dans une variable
3- on exécute la requête
4- on récupère le résultat de la requête dans un tableau
grâce au paramètre MYSQLI_ASSOC de la fonction mysqli_fetch_all(, MYSQLI_ASSOC)

NB : je ferai appel à la print_r() pour afficher les résultats bruts.

Contenu du fichier read.php

```
<?php
require_once("db_connexion.php");
$req="select * from client";
$result=mysqli_query($conn,$req);
$tab=mysqli_fetch_all($result,MYSQLI_ASSOC);
echo "Voici le resultat brut " ;
print_r($tab);
?>
```

On teste et voici le résultat :



Créons la fonction create.php permettant ajouter des informations dans la table client de la base banque :
Algo :

1- on récupère les données à insérer
2- on se connecte à la base
2- on prépare la requête dans une variable
3- on exécute la requête

On teste en envoyant les données avec la méthode get

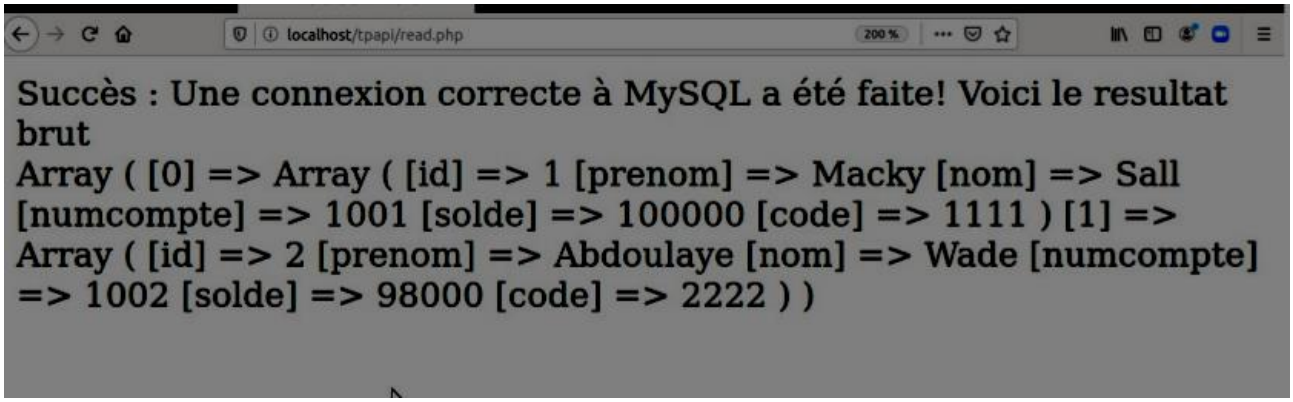
Reflexion :

il y a plusieurs méthodes d'envoi des données à partir d'un client HTTP :
Pour envoyer des données avec la méthode get,
on peut utiliser un navigateur avec une url de la forme :
http://localhost/tpapi/ read,php ?para1=valeur1&para2=valeur2

contenu create .php

```
<?php
require_once("db_connexion.php");
$prenom=$_REQUEST['prenom'];
$nom=$_REQUEST['nom'];
$numcompte=$_REQUEST['numcompte'];$solde=$_REQUEST['solde'];
$code=$_REQUEST['code'];
$req="insert into client(prenom,nom,numcompte,solde,c$
$result=mysqli_query($conn,$req);
?>
```

Voici le résultat de la création du président Wade et on utilise notre programme read.php
pour visualiser le résultat de création du compte Wade



Créons le programme update.php

Algo

1- on récupère id et le nouveau solde du président
2- on se connecte à la base
3- on prépare la requête dans une variable
4- on exécute la requête
code de update.php

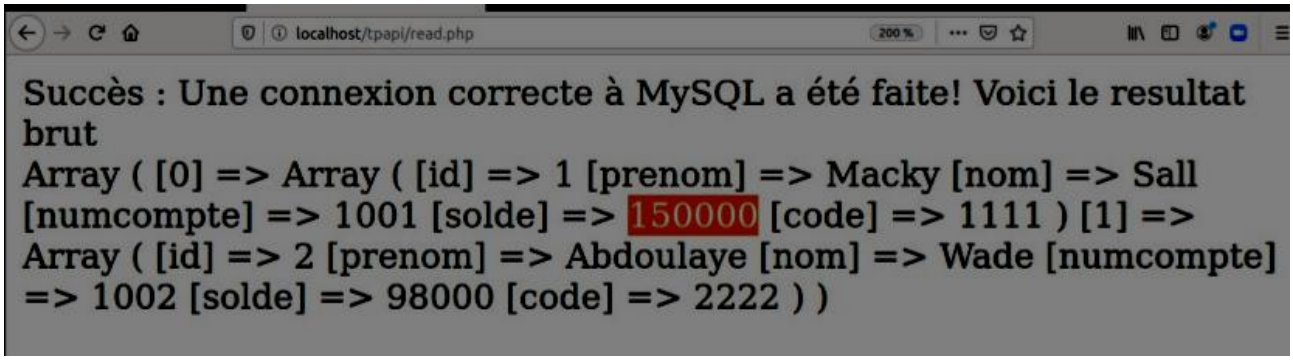
```
<?php
require_once("db_connexion.php");
$id=$_REQUEST['id'];
$solde=$_REQUEST['solde'];
$req="update client set solde='$solde' where id=$id";
$result=mysqli_query($conn,$req);
?>
```

NB : Remarquez que id est un entier c'est pourquoi dans la requête, il n'est pas entouré par griffe

On teste notre programme en faisant passer le solde du président Sall à 150000F ; son id étant 1
On utilise un navigateur en fournissant les données comme suit :

http://localhost/tpapi/update.php?id=1&solde=150000

Utilisant le programme read.php pour voir le résultat

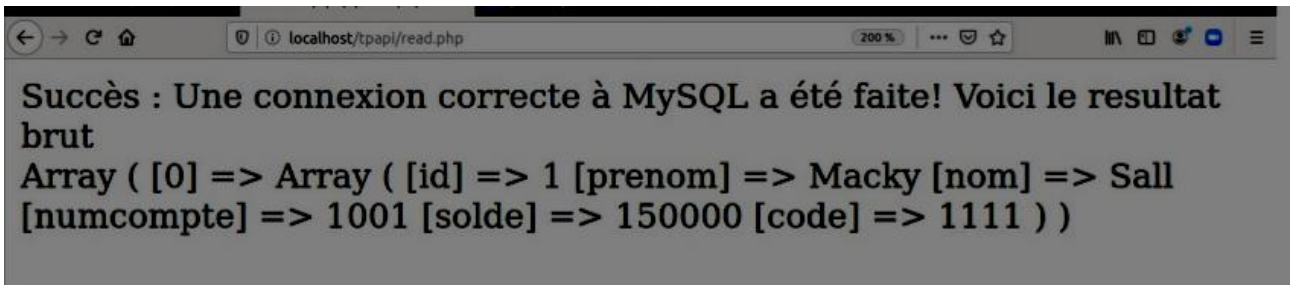


Passons à la dernière fonction delete.php
Algo :
1- on récupère l'id du président à supprimer
2- se connecter à la base
3- préparer la requête dans une variable
4- exécuter la requête

Contenu de delete.php

```
<?php
require_once("db_connexion.php");
$id=$_REQUEST['id'];
$req="delete from client where id=$id";
$result=mysqli_query($conn,$req);
?>
```

Testons le programme en supprimant le compte du président Wade dont l'id est 2 :
Pour cela, on utilise un navigateur avec l'url
http://localhost/tpapi/delete.php?id=2
On utilise le programme read.php pour voir si le compte du président wade a été supprimé



Effectivement, le compte du président Wade a été supprimé

4 Cours 3 Le premier pas vers les API

Il faut commencer à respecter l'utilisation des formats json d'échange de données.
Il faut commencer à préparer les outils permettant de récupérer les verbes(GET, POST,PUT,DELETE) employés par des clients qui vont invoquer ses prochains API

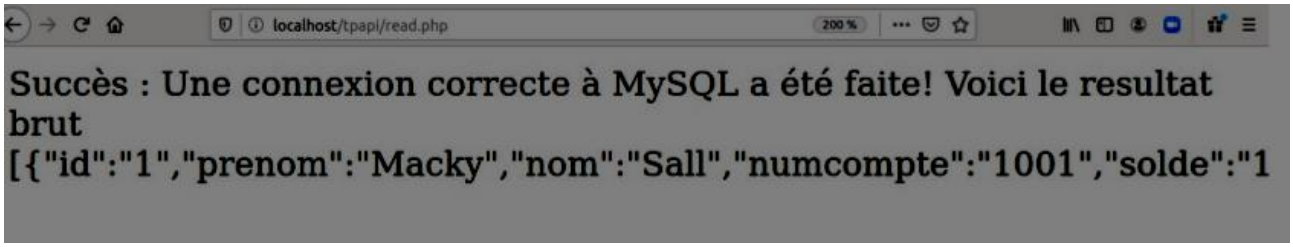
Réflexion

il faut la prise en compte de json par php : paquet php-json.
il faut utiliser la variable superglobale $_SERVER["REQUEST_METHOD"].
Deux fonctions importantes json_encode() et json_decode().
En examinant le cours numéro 1 et la règle 2 de ce cours qui stipule qu'on doit utiliser les verbes HTTP comme opérations, il est important pour moi de savoir récupérer le verbe http qu'un client utilise pour invoquer mon api, c'est fait à travers la fiche 3 qui dit qu'on peut utiliser la variable superglobale $_SERVER["REQUEST_METHOD"], permettant d'obtenir la méthode HTTP utilisée dans une requête.
Pour comprendre la fonction json_encode, nous allons modifier notre fonction read.php pour qu'elle convertisse le tableau des présidents en json

contenu de read.php modifié

```
<?php
require_once("db_connexion.php");
$req="select * from client";
$result=mysqli_query($conn,$req);
$tab=mysqli_fetch_all($result,MYSQLI_ASSOC);
$tabencode=json_encode($tab);
echo "Voici le resultat brut "."<br>" ;
print_r($tabencode);
?>
```

Après test sur un navigateur, on obtient :



NB : json_encode(tableauassociatif) retourne du json
json_decode(chaine,TRUE) retourne tableau associatif lorsque la chaîne est au format json .
Donc on voit l'intérêt de la boucle foreach qu'il faut utiliser dans les api pour parcourir les données après les avoir décodées.
Créons le programme vu.php qui va parcourir le tableau des présidents et les afficher sous forme de tableau comme indiqué sur la fiche 1.
Donc, il savoir générer automatiquement des tableaux html dans du code php.
Nous allons commencer par adopter le paradigme MVC(Modele vue controleur) qui consiste a organiser son application en trois parties :
- la partie modèle s'occupe uniquement de se connecter à la base de donnée pour récupérer les données
-la partie vue reçoit des données du contrôleur et les affiche
- la partie contrôleur est appelée par l'utilisateur de l'application ;le contrôleur fait appel au modèle pour récupérer les données et les transmets à la vue pour présentation.

Mettons en œuvre ce paradigme en php

Voici le contenu du modèle modeleread.php

```
<?php
require_once("db_connexion.php");
$req="select * from client";
$result=mysqli_query($conn,$req);
$tab=mysqli_fetch_all($result,MYSQLI_ASSOC);
?>
```

Contenu de vu.php

```
<?php
$chaine="<table
border='1px'><tr><td>Prenom</td><td>Nom</td><td>Numcompte</td><td>Solde</
td><td>Code</td></tr>";
foreach($tab as $ligne) {
$prenom=$ligne['prenom'];$nom=$ligne['nom'];$numcompte=$ligne['numcompte'];
$solde=$ligne['solde'];$code=$ligne['code'];
$chaine=$chaine."<tr><td>$prenom</td><td>$nom</td><td>$numcompte</td><td>$solde</td><t
d>$code</td>";
}
echo $chaine;
?>
```

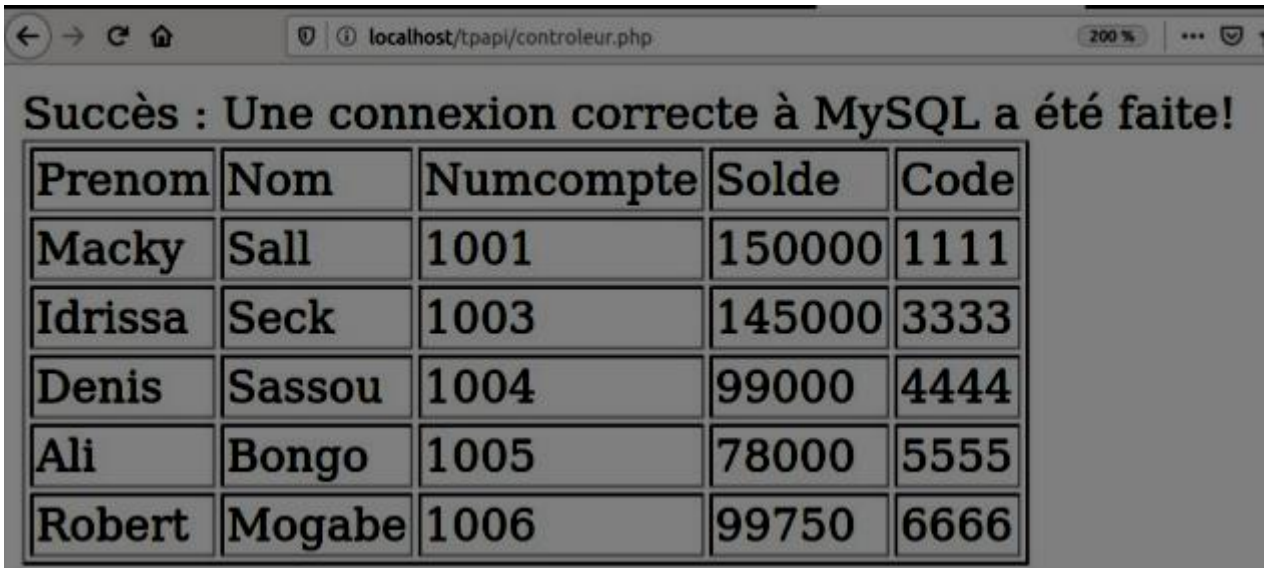
Contenu du contrôleur

```
<?php
require_once("modeleread.php");
require_once("vu.php");
?>
```

NB : on a juste fait appel a la fonction require_once pour inclure le code du modèle et le code de la vue

Présentation des résultats

on utilise un navigateur avec l'url : http://localhost/tpapi/controleur.php



4.1 Conclusion :

Nous avons identifié dans cet exercice les principales fonctions et les variables superglobales à utiliser lors de développement d'API RESTfull pour les règles sur le format des données et les verbes HTTP comme opérations.

La boucle foreach sera utile pour créer des interfaces des interfaces d'exploitation d'API.

Il est important de me familiariser à l'utilisation de :

- la commande curl pour manipuler des API REST
- php-curl pour créer des interface d'utilisation des API

5 Cours 4 : Développement d'Api REST CRUD et interfaces de consommation

5.1 Objectifs :

1- Tranformer nos applications create.php read.php update.php et delete.php en API restfull

1.1 Les données et les résultats manipulées soient reçues ou envoyées au format json

1 .2 En fonction du verbe HTTP employé qu'on puisse faire l'action

2- utilisation de la commande curl pour consommer une API
3- utilisation de postman pour consommer une API
4- création des interfaces de consommation d'api en php

action 1 : Algo de creatAPI.php
1- on récupère les résultats reçus au format json
2- on convertit les résultats en tableau associatif
3- on récupère les champs prenom, nom,
numcompte, solde et code du tableau associatif
4- on prépare la requête d'insertion
5- on exécute la requête
6- si bien exécutée affecte à reponse 'creation reussie'
sinon affecte à reponse « probeme creation »
7- affiche reponse

action 2 ; Utilisation de la commande curl pour créer le compte de emmanuel macron numcompte=1006, solde=1000000 code =6666 :

curl -X POST -H "Content-Type:application/json" -d '{"prenom":"Emmanuel","nom":"Macron","numcompte":"1006","code":"6666","solde":"1000000" }' http://localhost/tpapi/createAPI.php

Conclusion : les clients et serveurs http considèrent par défaut que les données sont au format html ou texte. Donc si ce n'est pas le cas, celui qui envoie les données doit préciser le format pour que l'autre entité puisse mieux exploiter les résultats

6 Cours 5 du mercredi 22 avril 2020

6.1 Règle http :

Les clients et serveurs http considèrent
par défaut que les données sont au format html ou texte.
Donc si ce n'est pas le cas,
Celui qui envoie les données doit préciser
le format pour que l'autre entité puisse mieux exploiter
les résultats

Cas des fichiers :

• pdf , il faut préciser dans l'entête Content-Type : application/pdf
• bureautique(word, excel,powerpoint), il faut préciser dans l'entete Content-Type : application/msword

Éléments langage php :

• la fonction header() permet de spécifier aux client des entêtes
•pour pouvoir utiliser le curseur $conn qui est dans le code db_connexion.php dans des fonctions php, il faut précéder son nom par global.

Rappel de la définition d'un algorithme
un algorithme est :
• la description de comment est ce que les données vont être concrètement représentées
• la description de la méthode pour atteindre les objectifs
voila pourquoi dans un algorithme :
Titre de l'algorithme
données :
résultats :
début
c'est ici qu'on décrit la méthode
fin

6.2 Étapes de conception et développement d'un programme

1- Problème posé :

Écrire une api permettant :
créer,lire,faire de mise à jour et supprimer un compte d'un président(id;prenom;nom;numpcompte;solde;code)

2- Analyse du problème en vue de le découper en sous programmes
Je prévois :
5 fonctions :
a- create() qui permet d'ajouter un compte
b- lectures() qui affiche tous les comptes
c- lecture(id) qui affiche le compte d'un président dont l'id est donné
d- modification() fait une mise a jour d'un compte d'un président
e- delete qui s'occupe de la suppression d'un compte d'un président

3- on donne l'algorithme de chaque sous programme

4- choix du langage de programmation

5- Discuter des structures de données

6- on passe au code de chaque sous programme

7- on passe au code du programme principal qui fait la coordination de tous les sous programmes

Algorithme du programme principal

Données :
Résultats : afficher les résultats de la requête
1-récupère la méthode http employée par le client HTTP
2- Au cas où la méthode est:
a- GET, alors fait appel à lectures() ou lecture() selon qu'un id n'est pas fourni ou fourni
b- POST, alors fait appel à create()
c- PUT alors fait appel à update()
d- DELETE alors fait appel à delete()
e- Dans les autres cas affiche la méthode n'est pas prévue

Reflexion pour passer au code du programme principal

- je vais utiliser la super variable $_SERVER["REQUEST_METHOD"]
pour récupérer la méthode
- au cas ou la méthode sera traduit par : switch($request_method)
-intval() convertir en entier ;
- break; permet de stopper le programme quand on finit de traiter un cas

code du programme principal

```
<?php

$request_method = $_SERVER["REQUEST_METHOD"];
switch($request_method){
    case 'GET':
                    //lecture de comptes des clients de la banque
            if(!empty($_GET["id"]))
            {
             $id=intval($_GET["id"]);
             lecture($id);
             }
            else
            {
                lectures();
            }break;

    case 'POST':  //Ajouter un compte
                    create();
                    break;
    case 'PUT':
                     // Modifier un compte
             $id = intval($_GET["id"]);
            modification($id) ;
            break;

    case 'DELETE':
                    // Supprimer un compte
            $id = intval($_GET["id"]);
            delete($id);break;

    default:
                  //Methode de requete invalide
        header("HTTP/1.0 405 Method Not Allowed"); break;

}
?>
```

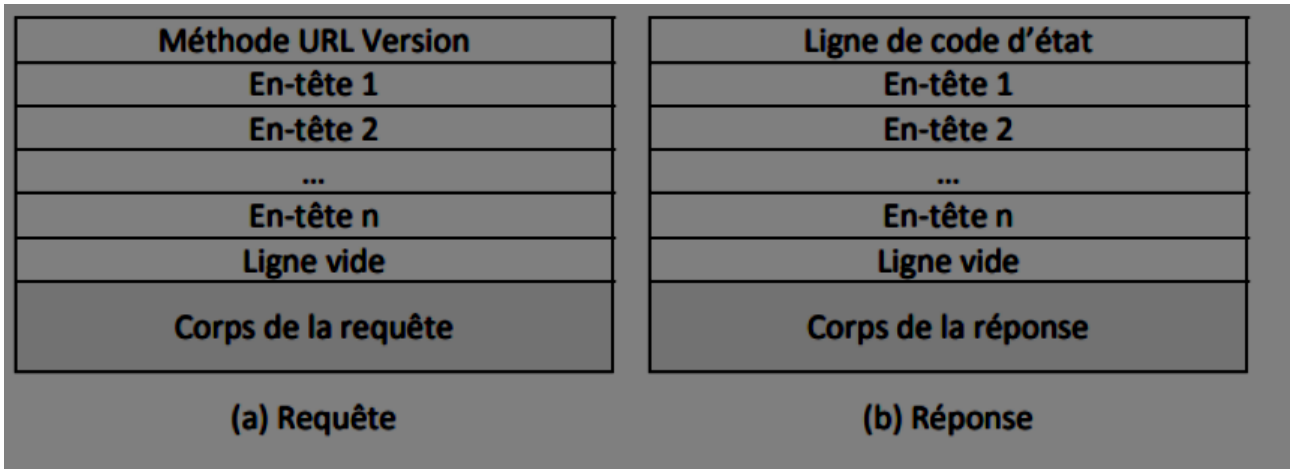
RAPPORT API REST FULL ET INTERFACE DE CONSOMMATION

Une réflexion s'impose pour expliquer la motivation de comment nous avons organisé nos codes et quelles sont les règles du métier sur lesquelles on se base.

comment les utilisateurs vont t-ils utiliser notre application ?

Il faut qu'on étudie de près le protocole HTTP :
Notamment :

• Les messages HTTP échangés entre les clients et serveurs HTTP
• Les formats des requêtes HTTP
• Les verbes HTTP
•Les formats d'une ligne de requêtes
•le format des réponses HTTP
•les formats d'une ligne de réponse
•Les entêtes associées aux requêtes
•Les entêtes associées aux réponses
•Les entêtes communes aux requêtes et réponses
•le type MIME permettant de gérer du multimédia
•Le type MIME multipart utilisé pour indiquer que le contenu est composé de plusieurs parties ayant chacune son propre type MIME.
Les requêtes et les réponses HTTP
Il existe deux types de messages dans HTTP : les requêtes et les réponses. Les deux sont de format similaire. Ils sont constitués d'une suite de lignes contenant des caractères en ASCII



Les requêtes HTTP

Une requête HTTP est formée d'une suite de lignes composées comme suit :
une ligne de requêtes contenant la commande de la requête appelée Méthode, l'URL de la ressource demandée et la version de HTTP utilisée;
zéro, une ou plusieurs lignes d'entête;

une ligne vide;le corps de la requête.
Chaque ligne doit se terminer par le caractère « retour de chariot » (CR ou Carriage Return) suivi de « Retour à la ligne » (LF ou line feed).
Les champs dans chaque ligne sont séparés par un ou plusieurs espacements.
Le format d'une ligne de requêtes est le suivant :
<Méthode> <URL> <Version de HTTP>
L'URL désigne la ressource. La version correspond au numéro de version du
protocole HTTP utilisée. Actuellement, la plupart des serveurs et des navigateurs comprennent la version 1.1. On spécifie la version sous la forme HTTP/1.1.
La méthode indique le type d'opération que l'on veut effectuer sur la ressource.
Il existe plusieurs méthodes dont :

GET : Cette méthode permet de lire le document spécifié par l'URL.
Cette lecture peut être conditionnelle si l'on utilise des entêtes qui imposent des conditions de lecture. Lorsqu'on utilise un lien sur une page Web ou que l'on ouvre une page Web avec le navigateur, c'est cette méthode qui est utilisée par défaut.
Voici un exemple de requête de type GET :
GET Programmation/tcpip.html HTTP/1.1
<Ligne vide>
Dans la plupart des cas, le corps de la requête est vide. Par contre, les enêtes peuvent être présents. On décrira ces entêtes plus loin dans ce cours.
-
POST : Cette méthode est utilisée pour soumettre des données à un serveur afin qu'il les traite. Les données soumises sont incluses dans le corps de la requête. Cette méthode est généralement utilisée pour soumettre au serveur des données qu'un usager aura entrées en utilisant un formulaire sur une page Web.
C'est le cas notamment lorsqu'on soumet un fichier à l'aide d'un champ <input> de type file.
Voici un exemple d'utilisation de POST :
POST Programmation/traitement HTTP/1.1
<Ligne vide>
...
... Donnees ...
...

PUT : Cette méthode permet de soumettre un document au serveur.
Si ce document n'existe pas, il sera créé. S'il y est déjà, il sera remplacé.
Ces opérations sont effectuées uniquement si le serveur en donne la permission. La différence entre PUT et POST est que l'URL dans la requête de type POST désigne une ressource qui doit traiter les données soumises alors que dans le cas de PUT elle identifie un document dont le contenu se trouve dans le corps de la requête.

DELETE : Cette méthode demande au serveur de détruire la ressource identifiée par l'URL. Bien entendu, cette méthode n'est exécutée que si le serveur le permet.

Les réponses HTTP
À chaque requête HTTP, le serveur répond par un message. Ces réponses
sont formées d'une suite de lignes composées comme suit :
une ligne de code d'état contenant la version de HTTP utilisée, un nombre qui indique le type de la réponse et un texte descriptif de ce code;
zéro, une ou plusieurs lignes d'entête;
une ligne vide;
le corps de la réponse

Le format d'une ligne de réponse est le suivant :
<Version HTTP> <Code> <Texte descriptif>
Le numéro du code est composé de trois chiffres qui indiquent si la requête a été satisfaite ou non. Le premier chiffre du code indique le type de réponse et les deux autres donnent des informations plus spécifiques.Voici quelques-uns de ces codes et leur signification :
1xx Information : Cette catégorie de codes donne des informations aux clients.
100 Continue : Le serveur accepte de traiter la requête et le client doit continuer.
101 Changement de protocole : Le serveur comprend la requête, mais doit continuer avec une autre version de HTTP.
2xx Succès : Cette catégorie de codes décrit les cas de succès.
200 OK : La requête a bien été exécutée.
201 Créée : La requête a été exécutée et la ressource demandée a été créée.
202 Accepté : La ressource a été acceptée et le traitement demandé est en cours d'exécution.
204 Aucun contenu : La requête a été exécutée, mais aucun contenu ne doit être retourné.
3xx Redirection : Cette catégorie de codes décrit des cas de déplacement de la ressource demandée.
300 Choix multiples : La ressource demandée correspond à plusieurs emplacements. Le client est informé pour qu'il puisse en choisir un en particulier.
301 Ressource déplacée : La ressource demandée a changé d'emplacement, et ce, de façon permanente.
304 Non modifié : La ressource demandée n'a pas été modifiée depuis la dernière requête. Le client peut utiliser la copie de la ressource qui se trouve dans sa mémoire cache.
4xx : Mauvaise requête : Cette catégorie de codes décrit des cas d'erreur.
401 Accès non autorisé : La requête exige que le client soit authentifié.
403 Interdit : Le serveur a compris la requête, mais refuse de l'exécuter.
404 Non trouvé : Le serveur ne trouve pas la ressource demandée.
405 Méthode non autorisée : La méthode demandée dans la requête n'est pas autorisée pour la ressource spécifiée.
408 Timeout dans la requête : Le client n'a pas produit de requête dans un intervalle de temps acceptable par le serveur. Il doit répéter sa requête.
410 Parti : La ressource demandée n'est plus disponible et aucune redirection n'a été spécifiée.
414 URL trop longue : Le serveur refuse de traiter la requête, car l'URL spécifiée dans la requête est trop longue.
5xx Erreurs du serveur : Cette catégorie de codes décrit des situations de problèmes liés au fonctionnement du serveur.
501 Non implanté : Le serveur ne possède pas la fonction requise pour traiter la requête.
503 Service non disponible : Le serveur ne peut pas traiter la requête pour des raisons de surcharge temporaire ou de maintenance. Le client doit réessayer plus
tard.
505 Version de HTTP non offerte : Le serveur n'offre pas ou refuse d'offrir la version de HTTP demandée dans la requête.

Les entêtes HTTP
Une ligne de requête ou de réponse peut être suivie de lignes d'en-tête qui permettent de préciser certains de leurs paramètres. Dans ce qui suit, nous en présentons quelques-uns.
Notons que certains de ces entêtes sont utilisables dans les requêtes ou dans les réponses alors que d'autres peuvent être utilisés dans les deux cas.
Chaque entête est spécifié en donnant son nom suivi des deux-points (:), suivi de la valeur de l'entête.
Par exemple :
GET Programmation/tcpip.html HTTP/1.0
Connections : close
Host : www.rtn.sn

User-agent : Mozilla/5.0
Accept-language : fr
...
Les entêtes associés aux requêtes sont les suivants :
Accept : peut être utilisé pour spécifier les types de médias accepables par le navigateur. Il indique que la requête est limitée à un ensemble restreint de types de médias, par exemple les images JPEG et GIF, que le navigateur peut lui-même interpréter sans recourir à des applications externes.
Accept-Charset : indique l'ensemble des caractères acceptables par le navigateur. Il spécifie également le codage utilisé pour ces caractères, tel que le codage ISO-Latin (nommé ISO-8859-1).
Accept-Encoding : spécifie les types de codages acceptables par le navigateur. Il permet de spécifier, par exemple, que le navigateur accepte des données compressées (par exemple utilisant un codage tel que .zip).
Accept-Language : spécifie les langues (français, anglais, etc.) que le navigateur accepte.
Autorization : permet au client de soumettre des paramètres d'authentification pour une autorisation d'accès.
Host : contient l'adresse du serveur.
Cookie : permet au client de présenter un témoin à un serveur qui le lui a livré auparavant. Les témoins seront expliqués plus loin dans ce cours.
If-Modified-Since : ajoute une condition à la requête. Si le contenu de la ressource n'a pas été modifié depuis la date spécifiée dans cet entête, cette ressource n'est pas retournée par le serveur. Le client peut alors utiliser la copie qui se trouve dans sa mémoire cache.
Referer : indique l'URL de la page à partir de laquelle la référence à la ressource dans la requête a été obtenue.
From : contient l'adresse de messagerie électronique de la personne qui est en charge du navigateur.
User-Agent : donne des informations (nom, version, etc.) sur le logiciel navigateur utilisé.
Voici un exemple de requête utilisant certains de ces entêtes :
GET /document1.html HTTP/1.1
Host : www.rtn.sn
Connection : keep-alive
User-agent : Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_7) ...
Accept : text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8Referer : http://www.rtn,sn/exemple1.html
Accept-encoding : gzip, deflate
Accept-language : fr-FR, fr;q=0.8, en-US;q=0.6, en;q=0.4
Accept-charset : ISO-8859-1, utf-8;q=0.7, *;q=0.3
Cookie : __utma=148357171.1616128815.1327369674.1327613087.1328040331.7;
Les entêtes utilisés dans les réponses sont les suivants :
Content-Encoding : spécifie le type de codage du contenu de la réponse (exemple, : le contenu de la réponse est compressé au format .zip).
Content-Language : décrit la langue (français, anglais, etc.) utilisée dans le document inclus dans la réponse.
Content-Length : indique la longueur en octets des données contenues dans la réponse.
Content-Type : indique le type de contenu de la réponse.
Il s'agit du ype MIME utilisé (voir plus loin pour la description de ces types).
Expires : spécifie la durée de validité d'un document utilisé pour la gestion des mémoires cache.
Location : permet au serveur de demander au client de rediriger sa requête ailleurs.
Server : donne des informations (nom, version, etc.) sur le logiciel serveur (par exemple, Apache,IIS) utilisé.
Set-Cookie : Permet au serveur de fournir un témoin au client.
Les entêtes communs aux requêtes et aux réponses sont les suivants :
Date : date et heure auxquelles la requête ou la réponse a été émise.

Upgrade : permet au client et au serveur de spécifier le protocole de communication qu'ils offrent et qu'ils souhaiteraient utiliser.

Les types MIME
Afin de permettre à des données de divers types d'être échangées entre un client et un serveur HTTP, la communauté Internet a introduit un mécanisme de description des types appelé MIME (Multipurpose Internet Mail Extension).
Les types MIME peuvent être des données en ASCII simple, étendu, ou des types plus complexes tels que des images, des sons, des vidéos, etc.
Ils permettent également des codages spécifiques à des langues ayant des caractères accentués (telles les langues latines) et même des caractères appartenant à des langues orientales (telles que le Chinois et l'Arabe), des images codées à l'aide de différentes techniques de codage (JPEG, GIF, etc.), de la voix, de la vidéo, etc. Ces codes ont été standardisés par la communauté d'Internet sous des nomes tels que UTF8
(pour Universal Transformation Format − 8-bit), UTF16, Unicode, etc.
MIME désigne ces codages par leur type et leur sous-type décrits à l'aide d'une expression sous la forme type/sous type.
Certains types définis dans le standard MIME sont : text, image, audio, video et application.
Pour chacun de ces types, on a plusieurs sous-types qui correspondent au codage utilisé.
Par exemple, text/html désigne un texte HTML et image/gif désigne une image codée en GIF.Pour spécifier le type MIME des données incluses dans un message, HTTP utilise un entête spécial appelé Content-type, comme suit :
Content-type : type/sous-type
Par exemple, pour des données image codées en GIF, on écrira :
Content-type : image/gif
Un nombre relativement grand de types de données MIME ont été définis, dont notamment text/html, text/plain, image/gif, image/jpeg, audio/basic, audio/x-wav, video/mpeg, video/quicktime, application/msword et application/zip.
Pour préciser la technique de codage utilisée pour un type MIME donné, des paramètres peuvent être ajoutés dans l'en-tête Content-type (après un point-virgule), comme dans :
Content-type : text/html; charset=iso-8859-1
Content-Type : text/html; charset=utf-8
La première ligne de l'exemple indique que le contenu est HTML et qu'on y trouve des caractères codés selon le codage ISO-8859-1, qui autorise les caractères latins accentués. La deuxième indique un codage universel.
UTF8. La page peut donc contenir des caractères autres que latins accentués (exemple : des symboles mathématiques).
Un nombre relativement important de types MIME ont été introduits pour désigner des documents produits par des logiciels spécifiques à Microsoft tels que MSWORD. Parmi ces types, mentionnons application/msword, application/pdf et application/zip.
Le type MIME multipart peut être utilisé pour indiquer que le contenu est composé de plusieurs parties ayant chacune son propre type MIME.
Il existe plusieurs types MIME de cette catégorie. On peut indiquer qu'il s'agit de choix de plusieurs contenus, d'une combinaison de plusieurs contenus, etc. Les parties sont séparées par un séparateur contenu dans le mot-clé boundary et débutent par un type MIME simple.
Après ces règles du métier rappelées, il convient de bien noter que :
une requête HTTP utilisant les verbes GET ou DELETE n'a pas de corps
Donc si nous voulons utiliser GET et DELETE avec un identifiant d'une ressource, alors cet

identifiant doit être fourni dans la ligne de requêteFaisons un tableau précisant comment utiliser notre api

Action	**URL**	**Contenu du corps de la requête**
Lister les infos sur tous les présidents	GET http:/localhost/APIPHP/api.php	vide
Lister les infos du président dont id est 2	GET http:/localhost/APIPHP/api.php?id=2	vide
Supprimer le compte du président avec l'id 5	DELETE http:/localhost/APIPHP/api.php?id=5	vide
Modifier le solde du président avec id=2	PUT http:/localhost/APIPHP/api.php?id=2	Nouveau solde du président
Créer un compte à un président avec un solde initial	POST http:/localhost/APIPHP/api.php	Prenom, nom,numcompte,solde,solde

Conclusion : Avant de développer une api il est important de preciser comment les données vont etre envoyées à l'api.
1- Faut il mettre les données a envoyer a l'api uniquement dans le corps de la requete ?
2- Faut il envoyer une partie des données dans la ligne de requete et une autre partie dans le corps ?
3-Faut-il tout envoyer dans le corps du message ?

NB : si les données sont envoyées dans le corps de la requete on peut les recuperer :
1-soit par $_POST[« identifiant »]
2-soit par $_REQUEST[« identifiant »]
3-soit par file_get_contents(« php://input »)

Ainsi je peux écrire les algos des fonctions

Algo ajout
1- on récupère les résultats reçus au format json
2- on convertit les résultats en tableau associatif
3- on récupère les champs prenom, nom, numcompte, solde et code du tableau associatif
4- on prépare la requête d'insertion
5- on exécute la requête
6- si bien exécutée on affecte à reponse un tableau associatif dont status=1 et status_message='creation reussie'
sinon on affecte à reponse un tableau associatif dont status=0 et status_message= 'Erreur de creation'
7- on convertit la reponse en format json et on affiche

Algo lectures
1- On se connecte a la base

2- on prépare la requête
3- on exécute la requête
4- on recupere le resultat de la requete dans un tableau associatif
5- on convertit le tableau en format json et on affiche

Algo lecture(id)
1- On recupere l'identifiant de l'utilisateur
2- on prépare la requête
3- on exécute la requête
4- on recupere le resultat de la requete dans un tableau associatif
5- on convertit le tableau en format json et on affiche

Algo modification
1- On recupere l'identifiant de l'utilisateur par la methode GET
1- on récupère le nouveau solde reçus au format json
2- on convertit le résultat en tableau associatif
3- on récupère le champ solde du tableau associatif
4- on prépare la requête de modification
5- on exécute la requête
6- si bien exécutée on affecte à reponse un tableau associatif dont status=1 et status_message='modification reussie'
sinon on affecte à reponse un tableau associatif dont status=0 et status_message= 'Erreur de modification'
7- on convertit la reponse en format json et on affiche

Algo suppression

1- On recupere l'identifiant de l'utilisateur
2- on prépare la requête
3- on exécute la requête
4- si bien exécutée on affecte à reponse un tableau associatif dont status=1 et status_message='suppresion reussie'
sinon on affecte à reponse un tableau associatif dont status=0 et status_message= 'Erreur de suppressionion'
5- on convertit la reponse en format json et on affiche

Algo transfert

1- On recupere le de l'utilisateur par la methode GET
1- on récupère l'émail du émetteur et l'email du beneficiaire ainsi que le montant à envoyer au format json
2- on convertit le résultat en tableau associatif
3- on récupère le champ email,emailben,montant du tableau associatif
4- on prépare la requête de recherche des utilisateurs dont les emails correspondent aux emails reçu
5- on exécute la requête
6- si bien exécutée alors on vérifie si le code donné est egale au champ code de l'utilisateur dont l'email est égale à l'émail du émetteur' et que son champ solde doit être supérieur au montant donné si oui on effectue le transfert et on affecte a reponse 'Transfert effectue avec succes' sinon on affecte a reponse 'vous n'avez pas asssez de credit'
sinon on affecte à reponse 'Informations erronees'

7- on convertit la reponse en format json et on affiche

puis passons à leur code

code de la fonction create() :

```
function create(){
                global $conn;
                $json=json_decode(file_get_contents("php://input"),TRUE);
                $prenom=$json["prenom"];
                $nom=$json["nom"];
                $numcompte=$json["numcompte"];
                $solde=$json["solde"];
                $code=$json["code"];
  if ((!empty($nom)) && (!empty($prenom)) && (!empty($numcompte)) && (!empty($solde)) &&
(!empty($code))){
                $req="INSERT INTO client(prenom,nom,numcompte,solde,code)
VALUES('".$prenom."','".$nom."','".$numcompte."','".$solde."','"$
                if(mysqli_query($conn,$req)){
                    $rep=array(
                            'status'=>1,
                            'status_message'=>'Ajout de votre utilisateur reussie');
                 }
                else{
                     $rep=array(
                             'status'=>0,
                             'status_message'=>'Erreur d ajout ');
                   }
                   header('Content-Type: application/json; charset=UTF-8');
                   echo json_encode($rep);echo "\n";
  }

}
```

Code de la fonction lectures()

```
function lectures(){
                global $conn;
                $sql="select * from client";
                $result=mysqli_query($conn,$sql);
                $tab=mysqli_fetch_all($result,MYSQLI_ASSOC);
                header('Content-Type: application/json');
                echo json_encode($tab, JSON_PRETTY_PRINT);
}
```

Code de la fonction lecture()

```
function lecture($id){
                    global $conn;
                    $sql="select * from client where id=".$id;
                    $result=mysqli_query($conn,$sql);
                    if(mysqli_affected_rows($conn)>0){
                    $row=mysqli_fetch_assoc($result);
                    // $row=mysqli_fetch_all($result,MYSQLI_ASSOC);
                   $rep="le solde de l'utilisateur: ".$row['prenom']." ".$row['nom']."  est :
".$row['solde'];
                    }else $rep="cet utilisateur n'existe pas";
header('Content-Type: application/json');
echo json_encode($rep, JSON_PRETTY_PRINT);echo "\n";

}
```

Code de la fonction modification()

```
function modification($id){
                        global $conn;
                        $data=json_decode(file_get_contents("php://input"),true);
                        $solde=$data['solde'];
                        $req="update client set solde='$solde' where id=$id";
                        $result=mysqli_query($conn,$req);
                        if(mysqli_affected_rows($conn)>0)
                         {
                                $rep=array(
                                        'status'=>1,
                                         'status_message'=>'Modification de votre utilisateur reussie');
                          }
                           else{
                                 $rep=array(
                                         'status'=>0,
                                         'status_message'=>'Erreur de Modification');
                              }
header('Content-Type: application/json');
echo json_encode($rep,  JSON_PRETTY_PRINT);echo "\n";

}
```

Code de la fonction delete()

```
function delete($id){
      global $conn;
      $req="delete from client where id=".$id;
      $result=mysqli_query($conn,$req);
```

```
    if(mysqli_affected_rows($conn)>0)
    {
        $rep=array(
            'status'=>1,
             'status_message'=>'Suppression de votre utilisateur reussie');
    }
    else{
       $rep=array(
            'status'=>0,
            'status_message'=>'Erreur de suppression');
      }
header('Content-Type: application/json');
echo json_encode($rep,  JSON_PRETTY_PRINT);echo "\n";
}
```

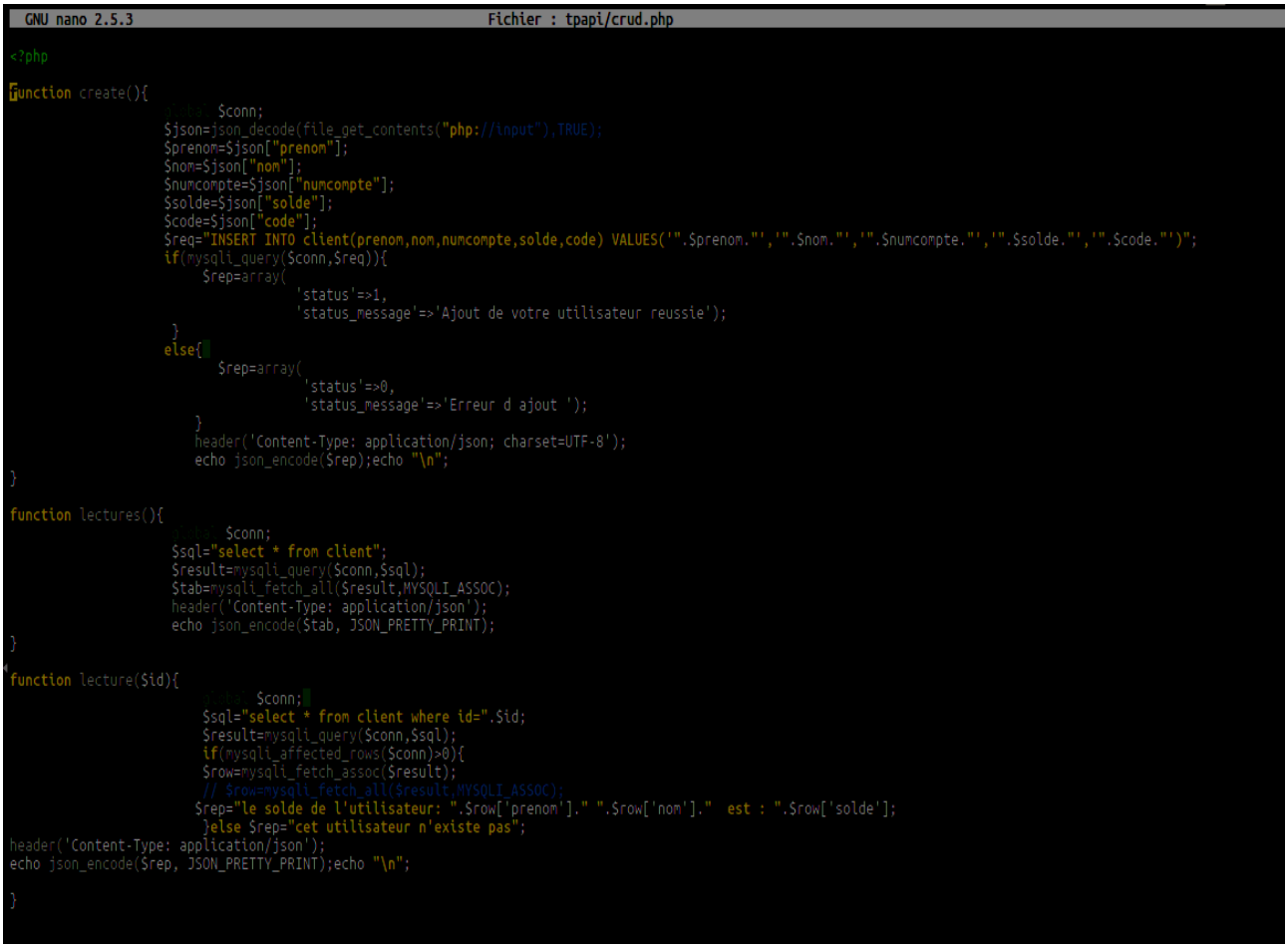
Code de la fonction transfert

```
function transfert($code){
    global $conn;
    $data=json_decode(file_get_contents("php://input"),true);
    $email=$data['email'];$emailben=$data['emailben'];$montant=$data['montant'];
    $sql1="select * from client where email in('$email','$emailben')";

     if($result1=mysqli_query($conn,$sql1)){
        $sql2="select * from client where email='$email'";
        $result2=mysqli_query($conn,$sql2);
        $tab2=mysqli_fetch_assoc($result2);

        if($tab2['code']==$code){
        if($tab2['solde']>$montant){
            $nouveauSoldeDon=$tab2['solde']-$montant;
            $sql3="select * from client where email='$emailben'";
            $result3=mysqli_query($conn,$sql3);
            $tab3=mysqli_fetch_assoc($result3);
            $nouveauSoldeBen=$tab3['solde']+$montant;
            $sqlben="update client set solde=$nouveauSoldeBen where email='$emailben'";
            $sqldon="update client set solde=$nouveauSoldeDon where email='$email'";
            mysqli_query($conn,$sqlben);
            mysqli_query($conn,$sqldon);
            $rep=json_encode("Transfert effectue avec succes !");

             }
        else $rep=json_encode("vous n'avez pas assez de credit .");
         }
        else $rep=json_encode("votre code est incorrect .");

            }
     else $rep=json_encode("Informations erronees");
     echo $rep;
```

RAPPORT API REST FULL ET INTERFACE DE CONSOMMATION

NB : Dans mon cas j'ai regroupée toutes ces fonction dans un seul fichier appelé crud.php que j'ai inclus dans le programme principal appelé api.php

Voici le code de crud.php



RAPPORT API REST FULL ET INTERFACE DE CONSOMMATION

```php
function lecture($id){
                        global $conn;
                       $sql="select * from client where id=".$id;
                       $result=mysqli_query($conn,$sql);
                       if(mysqli_affected_rows($conn)>0){
                       $row=mysqli_fetch_assoc($result);
                       // $row=mysqli_fetch_all($result,MYSQLI_ASSOC);
                      $rep="le solde de l'utilisateur: ".$row['prenom']." ".$row['nom']."  est : ".$row['solde'];
                       }else $rep="cet utilisateur n'existe pas";
header('Content-Type: application/json');
echo json_encode($rep, JSON_PRETTY_PRINT);echo "\n";

}

function delete($id){
        global $conn;
       $req="delete from client where id=".$id;
       $result=mysqli_query($conn,$req);
       if(mysqli_affected_rows($conn)>0)
       {
                $rep=array(
                           'status'=>1,
                           'status_message'=>'Suppression de votre utilisateur reussie');
       }
       else{
            $rep=array(
                      'status'=>0,
                      'status_message'=>'Erreur de suppression');
           }
header('Content-Type: application/json');
echo json_encode($rep,  JSON_PRETTY_PRINT);echo "\n";
}
```

```php
function modification($id){
                                global $conn;
                               $data=json_decode(file_get_contents("php://input"),true);
                               $solde=$data['solde'];
                               $req="update client set solde='$solde' where id=$id";
                               $result=mysqli_query($conn,$req);
                               if(mysqli_affected_rows($conn)>0)
                                {
                                         $rep=array(
                                                    'status'=>1,
                                                    'status_message'=>'Modification de votre utilisateur reussie');
                                  }
                                   else{
                                            $rep=array(
                                                    'status'=>0,
                                                    'status_message'=>'Erreur de Modification');
                                        }
header('Content-Type: application/json');
echo json_encode($rep,  JSON_PRETTY_PRINT);echo "\n";

}

?>
```

```
function transfert($code){
            $conn;
       $data=json_decode(file_get_contents("php://input"),true);
       $email=$data['email'];$emailben=$data['emailben'];$montant=$data['montant'];
       $sql1="select * from client where email in('$email','$emailben')";

        if($result1=mysqli_query($conn,$sql1)){
               $sql2="select * from client where email='$email'";
               $result2=mysqli_query($conn,$sql2);
               $tab2=mysqli_fetch_assoc($result2);

               if($tab2['code']==$code){
               if($tab2['solde']>$montant){
                       $nouveauSoldeDon=$tab2['solde']-$montant;
                       $sql3="select * from client where email='$emailben'";
                       $result3=mysqli_query($conn,$sql3);
                       $tab3=mysqli_fetch_assoc($result3);
                       $nouveauSoldeBen=$tab3['solde']+$montant;
                       $sqlben="update client set solde=$nouveauSoldeBen where email='$emailben'";
                       $sqldon="update client set solde=$nouveauSoldeDon where email='$email'";
                       mysqli_query($conn,$sqlben);
                       mysqli_query($conn,$sqldon);
                       $rep=json_encode("Transfert effectue avec succes !");

                       }
               else $rep=json_encode("vous n'avez pas assez de credit .");
                }
               else $rep=json_encode("votre code est incorrect .");

                       }
        else $rep=json_encode("Informations erronees");
        echo $rep;



}
?>
```

voici le code du progrmme principal api.php

```
<?php

$request_method = $_SERVER["REQUEST_METHOD"];
switch($request_method){
    case 'GET':
                    //lecture de comptes des clients de la banque
            if(!empty($_GET["id"]))
            {
             $id=intval($_GET["id"]);
             lecture($id);
            }elseif(!empty($_GET["code"]))
            {
             $code=$_GET["code"];
             lectur($code);
             }
            else
            {
                 lectures();
            }break;

    case 'POST':  //Ajouter un compte
                    create();
                    break;
    case 'PUT':
```

```
                // Modifier un compte
            if(!empty($_GET["id"]))
            {

            $id = intval($_GET["id"]);
            modification($id);
            }elseif(!empty($_GET["code"]))
            {
            $code=$_GET["code"];
            transfert($code);
            }break;

    case 'DELETE':
                // Supprimer un compte
            $id = intval($_GET["id"]);
            delete($id);break;

    default:
                //Methode de requete invalide
        header("HTTP/1.0 405 Method Not Allowed"); break;

}
?>
```

Testons avec Curl : on se connecte d'abord à la base donné pour lister les présidents déjà enregistrés

```
root@therese:/var/www/html# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.29-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql> use banque
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select *from client;
+----+-----------+-----------+-----------+------------+-------+
| id | prenom    | nom       | numcompte | solde      | code  |
+----+-----------+-----------+-----------+------------+-------+
|  1 | Macky     | sall      | 1001      | 20 000 000 | 1111  |
|  2 | Abdoulaye | wade      | 1002      | 101000     | 2222  |
|  3 | Idrissa   | seck      | 1003      | 145000     | 3333  |
|  4 | Denis     | sassou    | 1004      | 100000000  | 4444  |
|  5 | Ali       | Bongo     | 1005      | 100000     | 5555  |
|  6 | Robert    | mogabe    | 1006      | 99750      | 6666  |
|  7 | babacar   | aba       | 999       | 99900000   | 88888 |
|  8 | boubou    | ababoucar | 12445     | 33344      | 45567 |
|  9 | ba3       | jean      | 2323      | 120000     | 12121 |
| 12 | ange      | patasse   | 77799     | 20 000 000 | 9090  |
+----+-----------+-----------+-----------+------------+-------+
10 rows in set (0,00 sec)

mysql>
```

a- on effectue la requete d'insertion de Emmanuel Macron

```
root@therese:~# curl -X POST -H "Content-Type:application/json" -d '{"prenom":"Emmanuel","nom":"Macron","numcompte":"1000","solde":"20 000","c
de":"1010"}' http://127.0.0.1/tpapi/api.php
connexion réussie
{"status":1,"status_message":"Ajout de votre utilisateur reussie"}
root@therese:~#
```

Apres la requête d'insertion on liste les enregistrement

```
mysql> mysql> select *from client;
+----+-----------+-----------+-----------+-------------+-------+
| id | prenom    | nom       | numcompte | solde       | code  |
+----+-----------+-----------+-----------+-------------+-------+
|  1 | Macky     | sall      | 1001      | 20 000 000  | 1111  |
|  2 | Abdoulaye | wade      | 1002      | 101000      | 2222  |
|  3 | Idrissa   | seck      | 1003      | 145000      | 3333  |
|  4 | Denis     | sassou    | 1004      | 100000000   | 4444  |
|  5 | Ali       | Bongo     | 1005      | 100000      | 5555  |
|  6 | Robert    | mogabe    | 1006      | 99750       | 6666  |
|  7 | babacar   | aba       | 999       | 99900000    | 88888 |
|  8 | boubou    | ababoucar | 12445     | 33344       | 45567 |
|  9 | ba3       | jean      | 2323      | 120000      | 12121 |
| 12 | ange      | patasse   | 77799     | 20 000 000  | 9090  |
| 13 | Emmanuel  | Macron    | 1000      | 20 000      | 1010  |
+----+-----------+-----------+-----------+-------------+-------+
11 rows in set (0,00 sec)

mysql>
```

b- on effectue une requete de lectures des comptes

```
root@therese:~# curl -X GET http://127.0.0.1/tpapi/api.php
connexion réussie
[{"id":"1","prenom":"Macky","nom":"sall","numcompte":"1001","solde":"20 000 000","code":"1111"},{"id":"2","prenom":"Abdoulaye","nom":"wade","n
mcompte":"1002","solde":"101000","code":"2222"},{"id":"3","prenom":"Idrissa","nom":"seck","numcompte":"1003","solde":"145000","code":"3333"},{
id":"4","prenom":"Denis","nom":"sassou","numcompte":"1004","solde":"100000000","code":"4444"},{"id":"5","prenom":"Ali","nom":"Bongo","numcompt
":"1005","solde":"100000","code":"5555"},{"id":"6","prenom":"Robert","nom":"mogabe","numcompte":"1006","solde":"99750","code":"6666"},{"id":"7
,"prenom":"babacar","nom":"aba","numcompte":"999","solde":"99900000","code":"88888"},{"id":"8","prenom":"boubou","nom":"ababoucar","numcompte"
"12445","solde":"33344","code":"45567"},{"id":"9","prenom":"ba3","nom":"jean","numcompte":"2323","solde":"120000","code":"12121"},{"id":"12","
renom":"ange","nom":"patasse","numcompte":"77799","solde":"20 000 000","code":"9090"},{"id":"13","prenom":"Emmanuel","nom":"Macron","numcompte
:"1000","solde":"20 000","code":"1010"}]root@therese:~#
```

c- on effectue une requete de lecture d'un compte president avec l'identifiant 1

```
root@therese:~# curl -X GET http://127.0.0.1/tpapi/api.php?id=1
connexion réussie
"le solde de l'utilisateur: Macky sall  est : 20 000 000"
root@therese:~#
```

d- on effectue une requete de modification d'un compte president avec l'identifiant 13

```
root@therese:~# curl -X PUT -H "Content-Type:application/json" -d '{"solde":"100 000"}' http://127.0.0.1/tpapi/api.php?id=13
connexion réussie
{
    "status": 1,
    "status_message": "Modification de votre utilisateur reussie"
}
root@therese:~#
```

d- on effectue une requete de suppression d'un compte president avec l'identifiant 13

```
root@therese:~# curl -X DELETE http://127.0.0.1/tpapi/api.php?id=13
connexion réussie
{
    "status": 1,
    "status_message": "Suppression de votre utilisateur reussie"
}
root@therese:~#
```

7 Cours du 29 avril 2020

On a respecté jusqu'à présent que 2 regles d'api rest sur 4.
il reste à prendre en charge la regle 1 et 4
Concernant la regle 1 :
Règle n°1 : l'URI comme identifiant des ressources :
pour cela on va utiliser la réecriture des url d'apache

Algo :
on va dans le dossier de l'api et on cree le fichier caché .htacces comme suit :
a) contenu .htacces

RewriteEngine on
RewriteRule ^api$ api.php [NC]
RewriteRule ^[^/]+/(\d+)$ api.php?id=$1

b) Creeons un fichier de configuration nommé api.conf dans le dossier */etc/apche2/sites-avalables/api.conf*
Voici le contenu du fichier *etc/apche2/sites-avalables/api.conf*

*<VirtualHost *:80>*

<Directory /var/www/html/tpapi>
AllowOverride All
Require all granted

</Directory>

DocumentRoot /var/www/html/tpapi
ServerName api.ec2lt.sn

</VirtualHost>

c) activer le module rewrite d'apache avec la commande : a2enmod rewrite
d) on redémarre apache2 avec la commande service appache2 restart

é) activé le site virtuel api.conf par : a2ensite api.conf, on recharge apache avec la commande service apache2 reload et on redémarre apache2
f)faire la correspondance entre api.ec2lt.sn et l'adresse Ip de la machine

Testons :
on effectue une requete de lectures des comptes

```
root@therese:/var/www/html/tpapi# curl -X GET http://api.ec2lt.sn/api
connexion réussie
[{"id":"1","prenom":"Macky","nom":"sall","numcompte":"1001","solde":"20 000 000","code":"1111"},{"id":"2","prenom":"Abdoulaye","nom":"wade","nu
mcompte":"1002","solde":"101000","code":"2222"},{"id":"3","prenom":"Idrissa","nom":"seck","numcompte":"1003","solde":"145000","code":"3333"},{"
id":"4","prenom":"Denis","nom":"sassou","numcompte":"1004","solde":"100000000","code":"4444"},{"id":"5","prenom":"Ali","nom":"Bongo","numcompte
":"1005","solde":"100000","code":"5555"},{"id":"6","prenom":"Robert","nom":"mogabe","numcompte":"1006","solde":"99750","code":"6666"},{"id":"7"
,"prenom":"babacar","nom":"aba","numcompte":"999","solde":"99900000","code":"88888"},{"id":"8","prenom":"boubou","nom":"ababoucar","numcompte":
"12445","solde":"33344","code":"45567"},{"id":"9","prenom":"ba3","nom":"jean","numcompte":"2323","solde":"120000","code":"12121"},{"id":"12","p
renom":"ange","nom":"patasse","numcompte":"77799","solde":"20 000 000","code":"9090"}]root@therese:/var/www/html/tpapi#
```

on effectue une requête de lecture d'un compte président avec l'identifiant 1

```
root@therese:/var/www/html/tpapi# curl -X GET http://api.ec2lt.sn/api/1
connexion réussie
"le solde de l'utilisateur: Macky sall  est : 20 000 000"
root@therese:/var/www/html/tpapi#
```

Conclusion : on à tester et tout marche

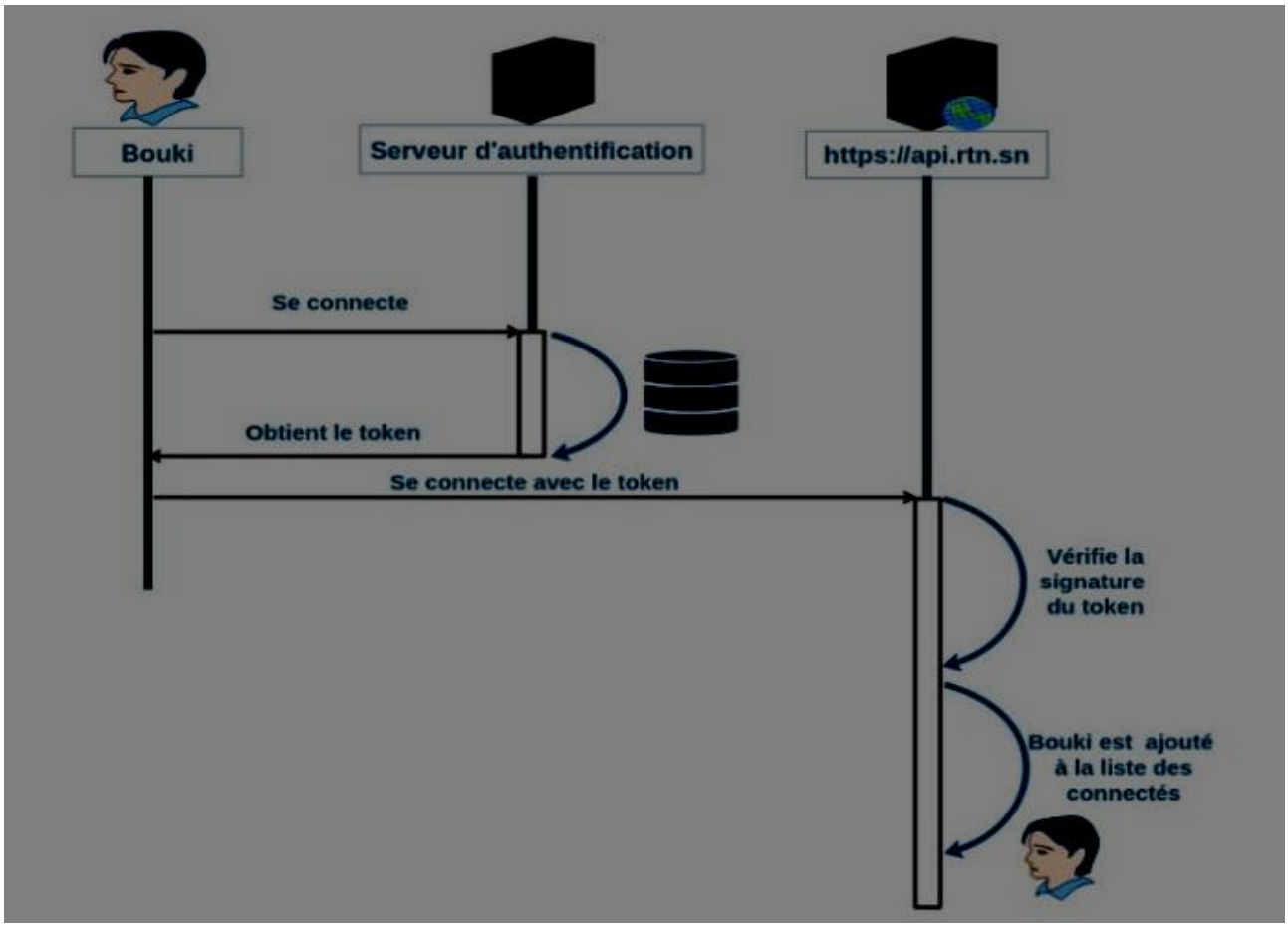
Passons à la dernière règle concernant la sécurité Pour cela on a décidé d'utiliser php-jwt

8 Cours du jeudi 30 Avril 2020 : LA SECURITE

8.1 Concepts de jwt

La mise en place d'une API REST et sa consommation nécessite de bien comprendre le protocole HTTP, notamment les lignes de requêtes HTTP, les lignes de code des réponses HTTP, les entêtes des messages HTTP et la recuperation du corps des messages HTTP mais aussi de comprendre comment utiliser l'extension php-curl pour créer des interfaces de consommation.
La sécurité d'aces à une API doit être aussi une priorité eu égard aux pertes que les opérateurs de télécoms ont subi ces derniers mois suite au piratage des API.
L'un des moyens les plus surs pour sécuriser l'accès à une API est l'utilisation de JWT (Json Web Token) qui est un jeton de sécurité composé de 3 chaînes de caractères séparant par un point.
La première partie, l'en-tête, est un JSON encodé en base64 qui décrit le type de token et l'algorithme utilisé pour la signature.
La seconde partie,le payload, est aussi un JSON encodé en base64 qui contient les

informations à transmettre. Ce JSON peut contenir des clefs prédéfinies (appelées claims) qui permettent de donner des informations supplémentaires sur le token (comme la date d'expiration, la cible, le sujet(pour qui le token a été généré)...)
La dernière partie, la signature, est la partie la plus importante du token JWT car elle permet de s'assurer de l'authenticité du token. Cette signature est générée à partir des 2 premières clefs avec un algorithme particulier (HS256 par exemple avec une clef secrète)
Le principe de fonctionnement est décrit ci-dessous



8.2 L'avantage de JWT

• Il n'est pas nécessaire de faire appel à une autorité extérieure pour obtenir les informations associées au token, un simple base64decode du payload suffit.
• La vérification de l'authenticité d'un token peut se faire de manière autonome, il suffit de connaître la clef secrète (ou d'avoir la clef publique suivant l'algorithme choisit) pour vérifier que la signature est authentique.
On peut inclure de manière facultative l'un ou plusieurs des 7 paramètres suivants, selon la

RFC 7519 :
•"iss" (émetteur) : identifie le service d'authentification qui a émis le jeton ;
•"sub" (sujet) identifie le commettant qui est le sujet du JWT.
Le traitement de ce paramètre est généralement spécifique à
l'application pour qui on a émis le jeton ;
•"aud" (audience) : identifie les destinataires du JWT.
Chaque mandant destiné à traiter le JWT DOIT s'identifier avec une
valeur dans la revendication du public ;
•"exp" (heure d'expiration) identifie l'heure d'expiration le ou
après quoi le JWT NE DOIT PAS être accepté pour traitement ;
•"nbf" (pas avant) identifie le délai avant lequel le JWT NE DOIT
PAS être accepté pour le traitement ;
•"iat" (émise le) identifie l'heure à laquelle le JWT a étépublié. Cette réclamation peut être utilisée pour déterminer l'âge du JWT ;
• "jti" (JWT ID) fournit un identifiant unique pour le JWT. La valeur de l'identifiant DOIT être attribuée de manière à garantir que il y a une probabilité négligeable que la même valeur sera assigné accidentellement à un autre objet de données; si l'application utilise plusieurs émetteurs, les collisions DOIVENT être évitées parmi les valeurs produits par différents émetteurs. La revendication "jti" peut être utilisée pour empêcher le JWT d'être rejoué ;

Rappels et taches préliminaires

Nous avons déjà développé une Api en php permettant aux
utilisateurs (présidents) de consulter leur compte,de transférer de fonds.
Nous décidons de créer des employés de la banque qui peuvent :
• créer un compte
• supprimer un compte
• modifier des infos d'un compte
• consulter des comptes

8.3 Installation de l'outil php-jwt permettant de gérer les token en php

On commence par installer le paquet composer sur notre machne :
apt-get install composer
3.1- On crée les dossiers
/var/www/html/apijwtm1
/var/www/html/apijwtm1/config
3.2- on se place dans le dossier /var/www/html/apijwtm1 pour installer firebase/php-jwt
cd /var/www/html/apijwtm1

```
root@therese:/var/www/html/apijwtm1# composer require firebase/php-jwt
```

répondez par n pour qu'il crée les fichiers nécessaires dans le dossier de votre projet.

4 – Création de code de connexion à notre base banque
4.1 - cd /var/www/html/apijwtm1/configon
édite le fichier db_connexion.php de connexion à la base de donnée banque

voici le code de db_connexion.php

```php
<?php
        $servername = '127.0.0.1';
        $user = 'root';
        $pass = 'passer';
        $base = 'banque';

        $conn = mysqli_connect($servername,$user,$pass,$base);

                if($conn){
        echo ("connexion réussie \n");
                        }
                else {
                        echo "connexion impossible";}
?>
```

NB : En ce moment si vous faite ls dans le dossier de votre projet, votre dossier de projet doit contenir un dossier et 2 fichiers suivants : composer.json composer.lock vendor
Apres vous allez créer vous meme un dossier config dans le dossier de votre projet avec la commande : mkdir config

Création de la table des employés
connecté vous a la base de donnée avec la commande : mysql -uroot -p
Apres vous tapez la commande : use banque
mysql> create table employee(id int primary key auto_increment, prenom varchar(30),nom varchar(30),email varchar(30),password varchar(30));
Query OK, 0 rows affected (0,42 sec)

Création des comptes des employés amy Diouf et therese davila

mysql> insert into employee(prenom,nom,email,mdp)
values("amy","Diouf","amy.douf@rtn.sn","passer1");

insert into employee(prenom,nom,email,mdp)
values("Therese","Davila","therese.davila@rtn.sn","passer2");

Création du code login.php qui joue le rôle du serveur d'authentification

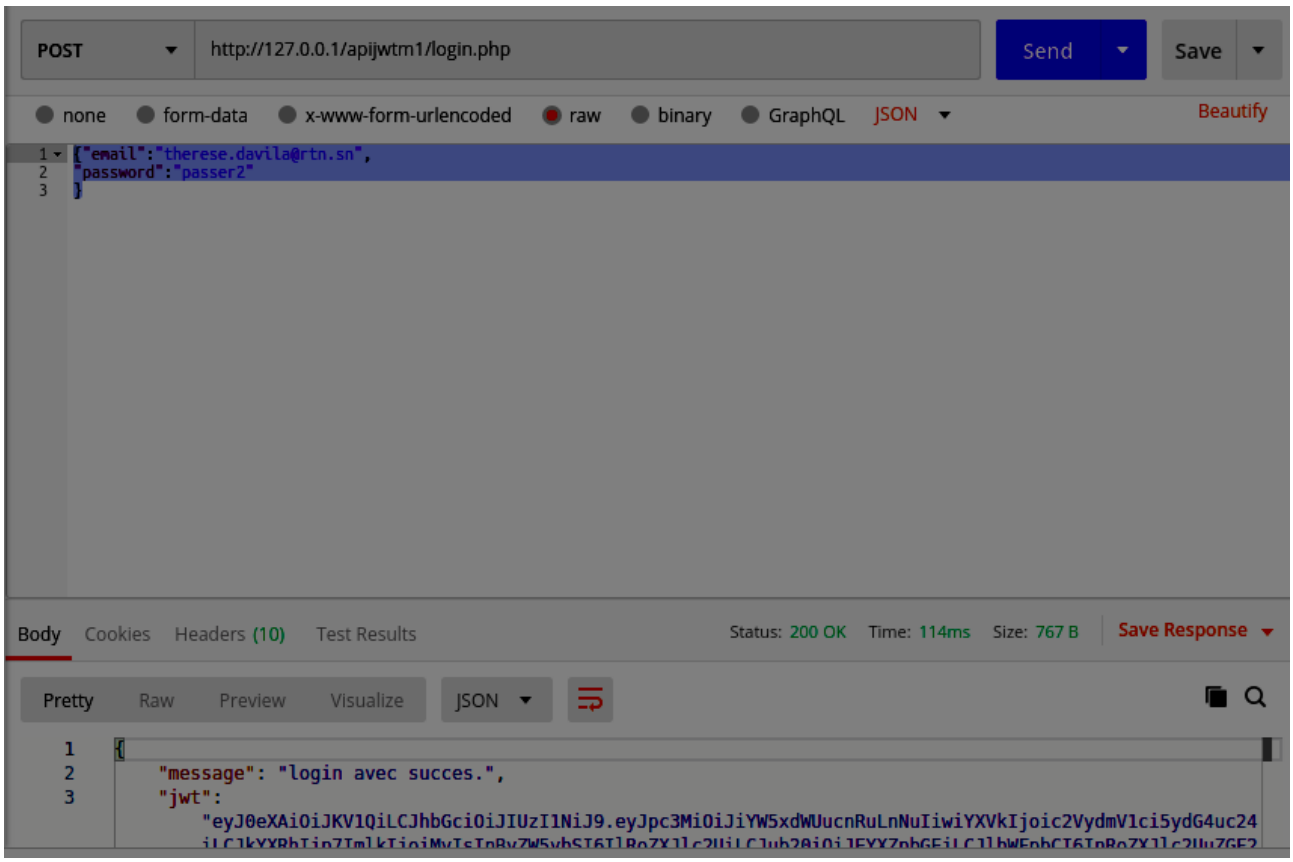
un utilisateur qui veut obtenir un token, s'authentifie avec son email et son mot de passe, apres quoi, on lui génère un token chiffré contenant son id, prénom, nom,email
voici le code login.php

GNU nano 2.5.3 Fichier : login.php

```php
<?php
include_once './config/db_connexion.php';
require "./vendor/autoload.php";
use \Firebase\JWT\JWT;


$secret_key = "passerrtn";
$issuer_claim = "banque.rtn.sn";
$audience_claim = "serveur.rtn.sn";

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

$data= json_decode(file_get_contents("php://input"),true);
$email= $data['email'];
$pass= $data['password'];
//$req= "SELECT id, prenom, nom, password FROM employee WHERE email ='$email' and password='$pass'";

$req="select * from employee where email ='$email' and password='$pass'";
$result=mysqli_query($conn,$req);
if(mysqli_num_rows($result)>0) {
            //$tab=mysqli_fetch_all($result,MYSQLI_ASSOC);
            $tab=mysqli_fetch_assoc($result);
            $id = $tab['id'];
            $prenom = $tab['prenom'];
            $nom = $tab['nom'];
            $password2 = $tab['password'];
            $issuedat_claim = time(); // issued at
            $notbefore_claim = $issuedat_claim + 10; //not before in seconds
            $expire_claim = $issuedat_claim + 600; // expire time in seconds
```

```php
            $token = array(
                            "iss" => $issuer_claim,
                            "aud" => $audience_claim,
                        //    "iat" => $issuedat_claim,
                         //   "nbf" => $notbefore_claim,
                          //  "exp" => $expire_claim,
            "data" => array(
                             "id" => $id,
                             "prenom" => $prenom,
                             "nom" => $nom,
                             "email" => $email
                         ));
        http_response_code(200);

      $jwt = JWT::encode($token, $secret_key);
      echo json_encode(
                        array(
                                "message" => "login avec succes.",
                                "jwt" => $jwt,
                                "email" => $email,
                                "expireAt" => $expire_claim
                            ));
}
else{

          http_response_code(401);
          echo json_encode(array("message" => "Echec de login.", "password" => $pass));
}

?>
```

Testons avec postman



Comme vous l'aurez remarqué, on a fourni dans body, l'email de therese davila et son mot de passe et le serveur d'authentification nous renvoie un token.
Ainsi notre serveur d'authentification est opérationnel

9 Cours du 1/5/2020

Nous rappelons que dans le processus d'authentification avec jwt, il ya 3 entités :
• l'utilisateur de l'api ;
• serveur d'authentification ;
• serveur fournissant l'api a protegé.
Le rôle du serveur d'authentification est d'authentifié l'user et lui génère un token chiffré que l'user va présenter au serveur API
Il appartient maintenant à l'utilisateur de présenter le token au serveur d'api en plus des requetes de L'utilisateur.
Ainsi, on peut décliner l'algorithme de l'api sécurisée comme suit :
1- Obtenir le token chiffré de l'utilisateur
2- essaie de le décoder avec le secret du serveur d'authentification
(le secret du serveur d'authentification est aussi au niveau du serveur api)
si le decodage marche alors debut récupère les données envoyées par l'user ,
exécute l'action demandée et envoie lui les résultats
fin
sinon dis à l'user qu 'il n'est autorisé à accéder à la ressource
Remarque : De quelle manière faut-il que l'api reçoive le token chiffré ?

RAPPORT API REST FULL ET INTERFACE DE CONSOMMATION

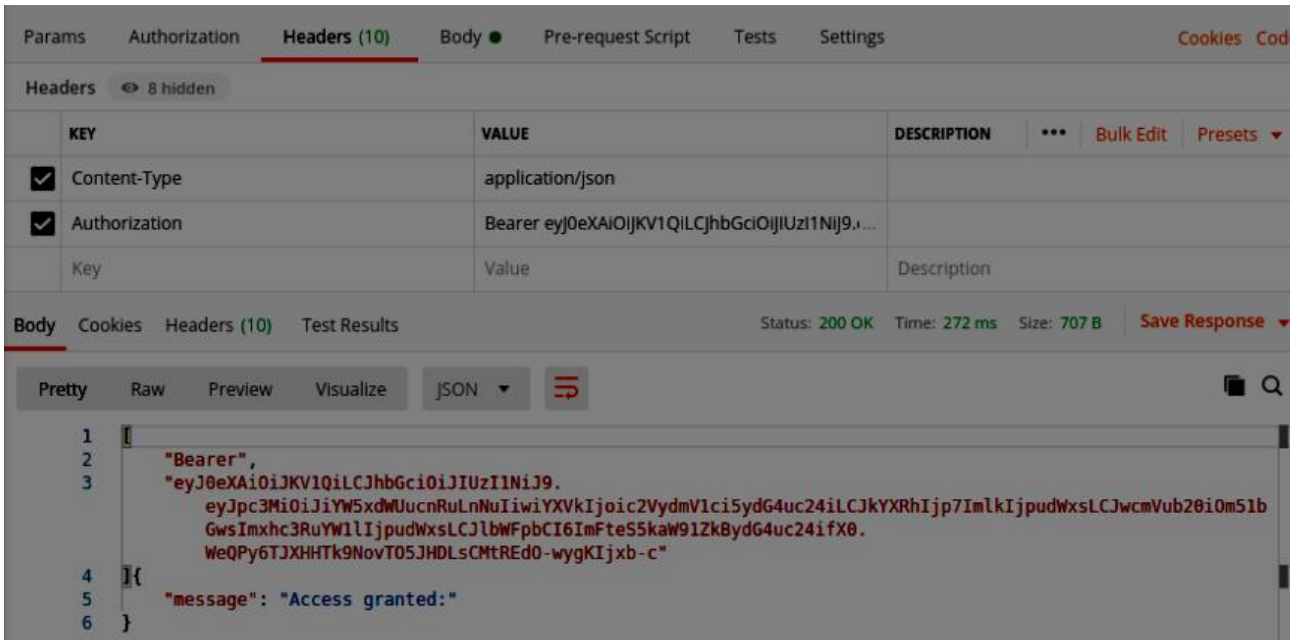
Il y'a plus manières de recevoir des données par HTTP :
1- en mettant les données dans les lignes de requêtes
2- en mettant les données dans l'entête des messages HTTP
3- en mettant les données dans le corps des messages HTTP
Nous avons décidé d'envoyer le token chiffré au serveur api
dans l'entête des requêtes
Donc dans notre api, on utilisera la fonction apache_request_headers()
pour récupérer le token chiffré dans l'entete Authorization
Règle : Quand on veut envoyer un token dans le champs Authorization,
on a le choix entre :
JWT token
ou
Bearer token
Nous choisissons le format : Bearer token
On écrit le code provisoire de notre api qui doit recevoir un token de l'utilisateur et s'il est validé par le serveur api,alors on lui envoie access granted

code api.php

```
<?php

$request_method = $_SERVER["REQUEST_METHOD"];
switch($request_method){
    case 'GET':
                //lecture de comptes des clients de la banque
            if(!empty($_GET["id"]))
            {
             $id=intval($_GET["id"]);
             lecture($id);
            }elseif(!empty($_GET["code"]))
            {
             $code=$_GET["code"];
             lectur($code);
             }
            else
            {
                lectures();
            }break;

    case 'POST':  //Ajouter un compte
                create();
                break;
    case 'PUT':
                 // Modifier un compte
             if(!empty($_GET["id"]))
            {

             $id = intval($_GET["id"]);
             modification($id);
            }elseif(!empty($_GET["code"]))
            {
             $code=$_GET["code"];
             transfert($code);
```

```
            }break;

    case 'DELETE':
                // Supprimer un compte
            $id = intval($_GET["id"]);
            delete($id);break;

    default:
                //Methode de requete invalide
        header("HTTP/1.0 405 Method Not Allowed"); break;

}
?>
```

On utilise postman pour tester notre api provisoire :
Remarquez bien qu'on a ajouté dans l'entête de notre requête les champs content-type et Authorization et on les activer en cochant

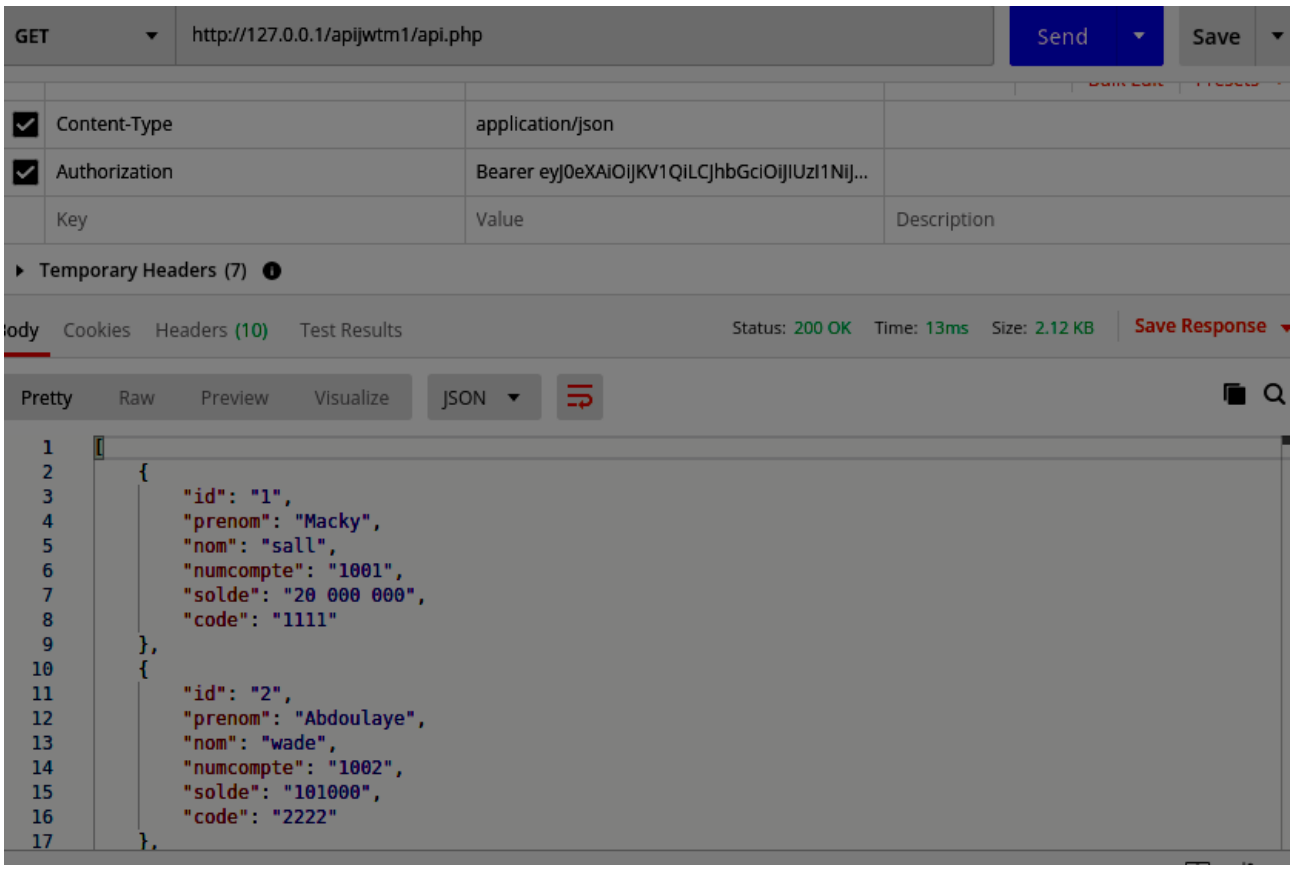


Ainsi notre api sécurisée provisoire marche comme on veut.
L'etape finale reste à ajouter le code réel de notre api qui fait les traitement apiinclud.php. Mais comme dans apiinclud.php, on avait déjà inclus le fichier db_connexion.php de connexion à notre base banque, on va enlevé avant de l'inclure.

Code api.php devient :

```
 GNU nano 2.5.3                                          Fichier : api.php

<?php
include_once './config/db_connexion.php';
require "./vendor/autoload.php";
include("crud.php");
use \Firebase\JWT\JWT;

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST,GET,PUT,DELETE");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Authorization, X-Requested-With");

$secret_key="passerrtn";
$jwt = null;
$authHeader=apache_request_headers();
$tete=$authHeader["Authorization"];
$arr = explode(" ", $tete);
$jwt = $arr[1];
//$jwt = trim($arr[1]);

if($jwt){
   try {

            $decoded = JWT::decode($jwt,$secret_key,array('HS256'));

                     // Acces autorise et on met le code de notre api.
                              include("apiinclud.php");
          /*  echo json_encode(array(
                                       "message" => "Access granted:"
                                    ));*/

   }catch (Exception $e){

                        http_response_code(401);

                             echo json_encode(array(
                                                    "message" => "Access denied."
                                                ));
                       }

}
?>
```

On teste avec postman, en fournissant à l'api que l'utilisateur amy Diouf avait recu du serveur d'authentification pour consulter tous les comptes des presidents.



Ainsi, nous avons finalisé notre api sécurisée.
Il nous reste qu'à utiliser php-curl et html pour créer l'interface de consommation qui va permettre aux utilisateurs de fournir leur email et mot de passe et les requêtes qu'ils veulent faire ;
On fera appel à login.php pour avoir un token chiffré et on fera appel à api.php pour avoir les resultats qu'on arranger et présenter aux utilisateurs.

9.1 Développement d'une interface de consommation d'une API sécurisée

9.1.1 Objectif :

Créer une interface de consommation d'une API sécurisé par token
Outils : html, php, php-curl
Concepts : session php, token dans les entêtes HTTP
Procédure :

9.1.2 Coté employé :

Nous allons créer une page d'accueil login.html permettant à un employé de fournir son émail ainsi que son mot de passe pour se connectant à l'interface présentant les différentes actions que notre api fait.
Cette interface que nous appelons essai.php crée deux variables de session $_SESSION['email'] et $_SESSION['pass'] et présente une interface permettant à l'utilisateur d'exécuter l'une des actions suivantes :
• Afficher les comptes de tous les présidents(essaitout1.php) ;
• Afficher le solde d'un président après avoir fourni son id(liste.html) ;
• Modifier le solde d'un président en fournissant son id (modifi.html);

RAPPORT API REST FULL ET INTERFACE DE CONSOMMATION

• Créer un compte à un président en fournissant son prénom, son nom,le solde,le numéro du compte et le code du compte(create.html);
• Supprimer le compte d'un président en fournissant son id(supprime.html).
A- Fonctionnement des 5 programmes intermédiaires ci-dessus
A1 : essaitout1.php invoque le serveur d'authentification login.php en lui fournissant l'émail et le mot de passe gardés en session et après avoir obtenu un token l'utilise pour invoquer notre api api.php ; une fois le tableau des comptes obtenu crée un tableau permettant de présenter le résultat.
A2 : liste.html présente un formulaire permettant à l'utilisateur de fournir l'id du président dont le solde doit être affiché puis fournit cet id à liste.php qui a son tour récupère dans la session l'émail et le mot de passe de l'utilisateur. Ayant ces 3 données, liste.php utilise l 'email et le mot de passe pour invoquer le serveur d'authentification login.php en vue d'obtenir un token qu'il complétera avec l'id pour demander à l'api le solde du président qu'il affichera.
A3 : modif.html présente un formulaire permettant à l'employé de fournir l'id et le nouveau solde du président dont le solde doit être mis à jour et fournit ces 2 éléments à modif.php qui à son tour récupère dans la session l'émail et le mot de passe de l'employé. Ayant ces 3 données, modif.php utilise l'émail et le mot de passe pour obtenir du serveur d'authentification un token qu'il complétera avec l'id et le nouveau solde du président pour invoquer l'api api.php.
A4 : create.html présente un formulaire permettant à l'utilisateur de saisir le prénom, le nom,le solde,le numéro du compte ainsi que le code du président et fournit ces éléments à create.php qui recupere dans les variables session l'email et le mot de passe de l'utilisateur qu'il utilisera pour obtenir du serveur d'authentification login.php un token qu'il completera ave cles informations personnelles pour invoquer l'api api.php.
A5 : supprime.html présente un formulaire permettant à l'employé de fournir l'id du président dont le compte doit être supprimé puis fournit cet id à supprime.php qui a son tour récupère dans la session l'émail et le mot de passe de l'utilisateur. Ayant ces 3 données, supprime.php utilise l 'email et le mot de passe pour invoquer le serveur d'authentification login.php en vue d'obtenir un token qu'il complétera avec l'id pour demander à l'api de supprimer le compte du président en question.Notre interface est composée de trois fichiers :
A6 : Deconnexion.php permet à l'employé de se déconnecter de la page

code login.html

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-compatible" Content="IE-edge">
<meta name="viewport" content="width=device-width">
<title>Page de connexion</title>
<link rel="stylesheet" href="../css/bootstrap.css" />
<link rel="stylesheet" href="../css/mystylecss.css" />
<link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css" />
</head>

<body style="background-color:blue";>

    <div class="container spacer col-md-6 col-xs-12 col-md-offset-3">
    <div class="panel panel-default">
       <div class="panel-heading">Authentification</div>
       <div class="panel-body">
           <form method="post" action="essai.php" >
                <div class="form-group">
                     <label class="control-label">Email:</label>
                     <input type="text" name="email" class="form-control"/>
                </div>
                <div class="form-group">
                     <label class="control-label">Pass:</label>
                     <input type="password" name="password" class="form-control"/>
                </div>
                <div>
                    <button type="submit" name="valider">Connexion</button>   &
                </div>
           </form>
       </div>
    </div>
    </div>
</body>
</html>
```

code essai.php

```
<?php
// On démarre la session AVANT d'écrire du code HTML
session_start();
// On s'amuse à créer quelques variables de session dans $_SESSION
$_SESSION['email'] = $_REQUEST['email'];
$_SESSION['pass'] = $_REQUEST['password'];
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-compatible" Content="IE-edge">
<meta name="viewport" content="width=device-width">
<link rel="stylesheet" href="../css/bootstrap.css" />
<link rel="stylesheet" href="mystylecss.css"/>
<!--<link rel="stylesheet" href="../css/mystylecss.css" />-->
<link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css" />
<title>Site Web Banque africa</title>
</head>
<body style="background-color:blue";>
```

```
<div class="navbar navbar-inverse navbar-fixed-top">
   <ul class="nav navbar-nav">
      <li><a href="essaitout1.php">Listé les présidents</a></li>
      <li><a href="liste.html">Listé un président</a></li>
      <li><a href="create.html">Ajouter un président</a></li>
      <li><a href="modif.html">Modifier le compte d'un président</a></li>
      <li><a href="supprime.html">Supprimer le compte d'un président</a></li>
            <li><a href="deconnexion.php">Déconnecter</a></li>
   </ul>
</div>

<br /><br /><br />

<h2 style='font-seize:20px';>
<center>Salut <?php echo $_SESSION['email']; ?> !<br />
Bienvenu(e) sur la page d'accueil de la banque africa.</center>
</h2>

</body>
</html>
```

code essaitout1.php

```
<?php
  session_start(); // On démarre la session AVANT toute chose
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Consommation API</title>
</head>
<body>
<?php
      $email = $_SESSION['email'];
  $password = $_SESSION['pass'];
$data = array(
            'email' => $email,
            'password' => $password);

$data1=json_encode($data);
$headers1 = array();
$headers1[]='Content-Type:application/json';
$url="http://127.0.0.1/apijwtm1/login.php";

 $ch=curl_init($url);
 curl_setopt($ch,CURLOPT_HTTPPOST,true);
 curl_setopt($ch,CURLOPT_POSTFIELDS,$data1);
 curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
 curl_setopt($ch,CURLOPT_HTTPHEADER,$headers1);
 curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'POST');
```

```
$reponse_json=curl_exec($ch);
curl_close($ch);
$response=json_decode($reponse_json,true);

//print_r($reponse);
//echo $response['message'];

if ($response['message']=="login avec succes.")
{
 $jwt=$response['jwt'];
        $url="http://127.0.0.1/apijwtm1/api.php";
        $ch=curl_init($url);
        //curl_setopt($ch,CURLOPT_URL,'http://127.0.0.1/apijwtm1/api.php');
        curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
        curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'GET');
        $headers=array();
        $headers[]="Content-Type:application/json";
        $headers[]="Authorization:Bearer ".$jwt;
        //$headers["Content-Type"]="application/json";
        //$headers["Authorization"]="Bearer ".$jwt;

        curl_setopt($ch,CURLOPT_HTTPHEADER,$headers);
        $response1_json=curl_exec($ch);
        curl_close($ch);
        $response1=json_decode($response1_json,true);

//         print_r($response1);
       echo "Voici le tableau des comptes des presidents";
$chaine="<table
border='1px'><tr><td>Prenom</td><td>Nom</td><td>Solde</td><td>Code</td></tr>";
       foreach ($response1 as $line){

$prenom=$line['prenom'];$nom=$line['nom'];$solde=$line['solde'];$code=$line['code'];

$chaine=$chaine."<tr><td>$prenom</td><td>$nom</td><td>$solde</td><td>$code</td></tr>";
                        }
       $chaine=$chaine."</table>";
       echo $chaine;
       $email=$_SESSION['email'];$pass=$_SESSION['pass'];
       echo "<a href='essai.php?email=$email&password=$pass'>Page accueil </a>";
}
else{ echo "Echec authentification";echo "<a href='index.php'>Retour</a>";}
?>
</body>
</html>
```

code de liste.html

```
<html>
```

```
<head>
<link href="style.css" rel="stylesheet"/>
</head>
<body style="background-color:blue";><p> FORMULAIRE DE CONSULTATION D'UN
COMMPTE PRESI PAR ID </p>
<form action="liste.php" method="post">
ID: <input type='text' name='id'><br/><br/>
<input type='submit' value='Valider'>
</form>
</body>
</html>
```

code de liste.php

```
<?php
   session_start(); // On démarre la session AVANT toute chose
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Consommation API</title>
</head>
<body>
<?php
        $email = $_SESSION['email'];
  $password = $_SESSION['pass'];
  $id = $_POST['id'];
$data = array(
                'email' => $email,
                'password' => $password);

$data1=json_encode($data);
$headers1 = array();
$headers1[]='Content-Type:application/json';
$url="http://127.0.0.1/apijwtm1/login.php";

 $ch=curl_init($url);
 curl_setopt($ch,CURLOPT_HTTPPOST,true);
 curl_setopt($ch,CURLOPT_POSTFIELDS,$data1);
 curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
 curl_setopt($ch,CURLOPT_HTTPHEADER,$headers1);
 curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'POST');
$reponse_json=curl_exec($ch);
curl_close($ch);
$response=json_decode($reponse_json,true);

//print_r($reponse);
//echo $response['message'];

if ($response['message']=="login avec succes.")
```

```
{

            // deuxieme partie
        $jwt=$response['jwt'];
        $url="http://127.0.0.1/apijwtm1/api.php?id=$id";
        $ch=curl_init($url);
        //curl_setopt($ch,CURLOPT_URL,'http://127.0.0.1/apijwtm1/api.php');
        curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
        curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'GET');
        $headers=array();
        $headers[]="Content-Type:application/json";
          $headers[]="Authorization:Bearer ".$jwt;
        //$headers["Content-Type"]="application/json";
        //$headers["Authorization"]="Bearer ".$jwt;

        curl_setopt($ch,CURLOPT_HTTPHEADER,$headers);
        $response1_json=curl_exec($ch);
        curl_close($ch);
        echo $response1_json;$email=$_SESSION['email'];$pass=$_SESSION['pass'];
        echo "<a href='essai.php?email=$email&password=$pass'>Page accueil </a>";
       }
else{ echo "Echec authentification"; echo "<a href='index.php'>Retour</a>";}
?>
</body>
</html>
```

Code de modif.html

```
<html>
<head>
<link href="style.css" rel="stylesheet"/>
</head>
<body><p> FORMULAIRE MODIFICATION PRESI </p>
<form action="modif.php" method="post">
ID: <input type='text' name='id'><br/><br/>
NouveauSolde: <input type='text' name='solde'><br/><br/>
<input type='submit' value='Valider'>
</form>
</body>
</html>
```

Code de modif.php

```
<?php
  session_start(); // On démarre la session AVANT toute chose
?>
<!DOCTYPE html>
<html>
<head>
```

```
<meta charset="utf-8" />
<title>Consommation API</title>
</head>
<body>
<?php
      $email = $_SESSION['email'];
  $password = $_SESSION['pass'];
  $data = array(
            'email' => $email,
            'password' => $password);

$data1=json_encode($data);
$headers1 = array();
$headers1[]='Content-Type:application/json';
$url="http://127.0.0.1/apijwtm1/login.php";

 $ch=curl_init($url);
 curl_setopt($ch,CURLOPT_HTTPPOST,true);
 curl_setopt($ch,CURLOPT_POSTFIELDS,$data1);
 curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
 curl_setopt($ch,CURLOPT_HTTPHEADER,$headers1);
 curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'POST');
$reponse_json=curl_exec($ch);
curl_close($ch);
$response=json_decode($reponse_json,true);


if ($response['message']=="login avec succes.")
{

          // deuxieme partie
        $jwt=$response['jwt'];
         $id = $_POST['id'];
         $solde = $_POST['solde'];
        $data = array(
            'solde' => $solde);
        $data1=json_encode($data);
        $url="http://127.0.0.1/apijwtm1/api.php?id=$id";
        $ch=curl_init($url);
        curl_setopt($ch,CURLOPT_HTTPPOST,true);
        curl_setopt($ch,CURLOPT_POSTFIELDS,$data1);
        curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
        curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'PUT');
        $headers=array();
        $headers[]="Content-Type:application/json";
        $headers[]="Authorization:Bearer ".$jwt;
        curl_setopt($ch,CURLOPT_HTTPHEADER,$headers);
        $response1_json=curl_exec($ch);
        curl_close($ch);
        echo $response1_json;$email=$_SESSION['email'];$pass=$_SESSION['pass'];
        echo "<a href='essai.php?email=$email&password=$pass'>Page accueil </a>";
      }
```

```
else{ echo "Echec authentification"; echo "<a href='index.php'>Retour</a>";}
?>
</body>
</html>
```

Code de supprime.html

```
<html>
<head>
<link href="style.css" rel="stylesheet"/>
</head>
<body><p> FORMULAIRE SUPPRESSION COMPTE PRESI PAR ID </p>
<form action="supprime.php" method="post">
ID: <input type='text' name='id'><br/><br/>
<input type='submit' value='Valider'>
</form>
</body>
</html>
```

Code de supprime.php

```
<?php
   session_start(); // On démarre la session AVANT toute chose
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Consommation API</title>
</head>
<body>
<?php
        $email = $_SESSION['email'];
  $password = $_SESSION['pass'];
  $id = $_POST['id'];
$data = array(
                'email' => $email,
                'password' => $password);

$data1=json_encode($data);
$headers1 = array();
$headers1[]='Content-Type:application/json';
$url="http://127.0.0.1/apijwtm1/login.php";

 $ch=curl_init($url);
 curl_setopt($ch,CURLOPT_HTTPPOST,true);
 curl_setopt($ch,CURLOPT_POSTFIELDS,$data1);
 curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
 curl_setopt($ch,CURLOPT_HTTPHEADER,$headers1);
```

```
 curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'POST');
$reponse_json=curl_exec($ch);
curl_close($ch);
$response=json_decode($reponse_json,true);

//print_r($reponse);
//echo $response['message'];

if ($response['message']=="login avec succes.")
{

            // deuxieme partie
          $jwt=$response['jwt'];
          $url="http://127.0.0.1/apijwtm1/api.php?id=$id";
          $ch=curl_init($url);
          curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
          curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'DELETE');
          $headers=array();
          $headers[]="Content-Type:application/json";
          $headers[]="Authorization:Bearer ".$jwt;
           curl_setopt($ch,CURLOPT_HTTPHEADER,$headers);
          $response1_json=curl_exec($ch);
          curl_close($ch);
          echo $response1_json;$email=$_SESSION['email'];$pass=$_SESSION['pass'];
          echo "<a href='essai.php?email=$email&password=$pass'>Page accueil </a>";
        }
else{ echo "Echec authentification"; echo "<a href='index.php'>Retour</a>";}
?>
</body>
</html>
```

Code de cree.html

```
<html>
<head>
<link href="style.css" rel="stylesheet"/>
</head>
<body><p> FORMULAIRE CREATION PRESI </p>
<form action="cree.php" method="post">
Prenom: <input type='text' name='prenom'><br/><br/>
Nom: <input type='text' name='nom'><br/><br/>
Solde: <input type='text' name='solde'><br/><br/>
NumCompte: <input type='text' name='numcompte'><br/><br/>
Code: <input type='text' name='code'><br/><br/>
<input type='submit' value='Valider'>
</form>
</body>
</html>
```

code de create.php

```
<?php
  session_start(); // On démarre la session AVANT toute chose
```

```
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Consommation API</title>
</head>
<body>
<?php
      $email = $_SESSION['email'];
  $password = $_SESSION['pass'];
  $data = array(
              'email' => $email,
              'password' => $password);

$data1=json_encode($data);
$headers1 = array();
$headers1[]='Content-Type:application/json';
$url="http://127.0.0.1/apijwtm1/login.php";

 $ch=curl_init($url);
 curl_setopt($ch,CURLOPT_HTTPPOST,true);
 curl_setopt($ch,CURLOPT_POSTFIELDS,$data1);
 curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
 curl_setopt($ch,CURLOPT_HTTPHEADER,$headers1);
 curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'POST');
$reponse_json=curl_exec($ch);
curl_close($ch);
$response=json_decode($reponse_json,true);


if ($response['message']=="login avec succes.")
{

            // deuxieme partie
        $jwt=$response['jwt'];
        $nom = $_POST['nom'];
        $prenom = $_POST['prenom'];
        $solde = $_POST['solde'];
        $numcompte = $_POST['numcompte'];
        $code = $_POST['code'];
        $data = array(
            'prenom' => $prenom,
            'nom' => $nom,
            'solde' => $solde,
            'numcompte' => $numcompte,
             'code' => $code);
        $data1=json_encode($data);
        $url="http://127.0.0.1/apijwtm1/api.php";
        $ch=curl_init($url);
        curl_setopt($ch,CURLOPT_HTTPPOST,true);
        curl_setopt($ch,CURLOPT_POSTFIELDS,$data1);
```

```
        curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
        curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'POST');
        $headers=array();
        $headers[]="Content-Type:application/json";
        $headers[]="Authorization:Bearer ".$jwt;
        curl_setopt($ch,CURLOPT_HTTPHEADER,$headers);
        $response1_json=curl_exec($ch);
        curl_close($ch);
        echo $response1_json;$email=$_SESSION['email'];$pass=$_SESSION['pass'];
        echo "<a href='essai.php?email=$email&password=$pass'>Page accueil </a>";
      }
else{ echo "Echec authentification"; echo "<a href='index.php'>Retour</a>";}
?>
</body>
</html>
```

9.1.3 Coté utilisateurs ;

Nous allons créer une page d'accueil login1.html permettant à un utilisateur de fournir son émail ainsi que son mot de passe pour se connectant à l'interface présentant les différentes actions que notre api fait.
Cette interface que nous appelons essai1.php crée deux variables de session $_SESSION['email'] et $_SESSION['pass'] et présente une interface permettant à l'utilisateur d'exécuter l'une des actions suivantes :
• Consulter son compte(liste1.html) ;
• Transferé de l'argent sur un compte(transfert.html) ;

A- Fonctionnement des 2 programmes intermédiaires ci-dessus
A1 : liste1.html présente un formulaire permettant à l'utilisateur de fournir son code secret puis fournit cet code à liste1.php qui a son tour récupère dans la session l'émail
et le mot de passe de l'utilisateur. Ayant ces 3 données, liste.php utilise l 'email et le mot de passe pour invoquer le serveur d'authentification login1.php en vue d'obtenir un token qu'il complétera avec le code pour demander à l'api le solde du président qu'il affichera.
A2 : Transfert.html présente un formulaire permettant à l'utilisateur de fournir son code,l'email de l'utilisateur a la quel il veut transféré de l'argent et le montant à envoyer et fournit ces 3 éléments à transfert.php qui à son tour récupère dans la session l'émail et le mot de passe de l'utilisateur. Ayant ces 5 données, transfert.php utilise l'émail et le mot de passe pour obtenir du serveur d'authentification un token qu'il complétera avec le code, l'email du bébéficiaire, et le montant à envoyer pour invoquer l'api api.php.
A3 : Deconnexion.php permet à l'utilisateur de se déconnecter de la page

code de liste1.html

```
<html>
<head>
<link href="style.css" rel="stylesheet"/>
</head>
<body style="background-color:blue";><p> FORMULAIRE DE CONSULTATION D'UN
COMMPTE PRESI PAR CODE SECRET </p>
<form action="liste1.php" method="post">
```

```
Mettez votre code: <input type='text' name='code'><br/><br/>
<input type='submit' value='Valider'>
</form>
</body>
</html>
```

code de liste1.php

```
<?php
  session_start(); // On démarre la session AVANT toute chose
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Consommation API</title>
</head>
<body style="background-color:blue";>
<?php
      $email = $_SESSION['email'];
 $password = $_SESSION['pass'];
 $code = $_POST['code'];
$data = array(
            'email' => $email,
            'password' => $password);

$data1=json_encode($data);
$headers1 = array();
$headers1[]='Content-Type:application/json';
$url="http://127.0.0.1/apijwtm1/login1.php";

 $ch=curl_init($url);
 curl_setopt($ch,CURLOPT_HTTPPOST,true);
 curl_setopt($ch,CURLOPT_POSTFIELDS,$data1);
 curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
 curl_setopt($ch,CURLOPT_HTTPHEADER,$headers1);
 curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'POST');
$reponse_json=curl_exec($ch);
curl_close($ch);
$response=json_decode($reponse_json,true);
//print_r($response);

if ($response['message']=="login avec succes.")
{

        // deuxieme partie
    $jwt=$response['jwt'];
    $url="http://127.0.0.1/apijwtm1/api.php?code=$code";
    $ch=curl_init($url);
    //curl_setopt($ch,CURLOPT_URL,'http://127.0.0.1/apijwtm1/api.php');
    curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
    curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'GET');
```

```
        $headers=array();
        $headers[]="Content-Type:application/json";
        $headers[]="Authorization:Bearer ".$jwt;
        curl_setopt($ch,CURLOPT_HTTPHEADER,$headers);
        $response1_json=curl_exec($ch);
        curl_close($ch);
        echo $response1_json;$email=$_SESSION['email'];$pass=$_SESSION['pass'];
        echo "<a href='essai1.php?email=$email&password=$pass' style='background-
color:white';>Page accueil </a>";
      }
else{ echo "Echec authentification"; echo "<a href='interfacepresi.php' style='background-
color:white';>Retour</a>";}
?>
</body>
</html>
```

code de transfert.html

```
<html>
<head>
<link href="style.css" rel="stylesheet"/>
</head>
<body style="background-color:blue";><p> FORMULAIRE DE CONSULTATION D'UN
COMMPTE PRESI PAR CODE SECRET </p>
<form action="transfert.php" method="post">
Mettez votre code: <input type='text' name='code'><br/><br/>
Mettez l'email du beneficiaire: <input type='text' name='emailben'><br/><br/>
Mettez le montant à envoyer: <input type='text' name='montant'><br/><br/>
<input type='submit' value='Valider'>
</form>
</body>
</html>
```

code de transfert.php

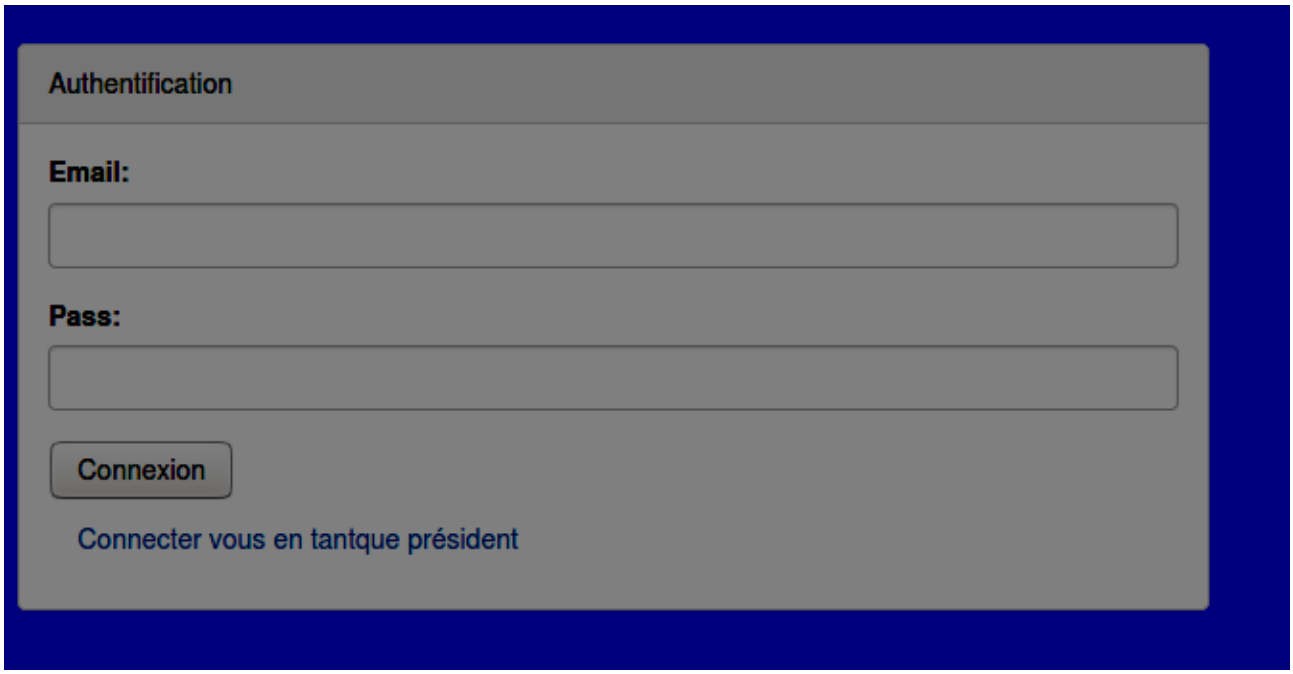
```
<?php
  session_start(); // On démarre la session AVANT toute chose
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Consommation API</title>
</head>
<body style="background-color:blue";>
<?php
   $email = $_SESSION['email'];
 $password = $_SESSION['pass'];
$data = array(
            'email' => $email,
            'password' => $password);
```

```
$data1=json_encode($data);
$headers1 = array();
$headers1[]='Content-Type:application/json';
$url="http://127.0.0.1/apijwtm1/login1.php";

 $ch=curl_init($url);
 curl_setopt($ch,CURLOPT_HTTPPOST,true);
 curl_setopt($ch,CURLOPT_POSTFIELDS,$data1);
 curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
 curl_setopt($ch,CURLOPT_HTTPHEADER,$headers1);
 curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'POST');
$reponse_json=curl_exec($ch);
curl_close($ch);
$response=json_decode($reponse_json,true);
//print_r($response);

if ($response['message']=="login avec succes.")
{

          $jwt=$response['jwt'];
          $email = $_SESSION['email'];
          $code = $_POST['code'];
          $emailben = $_POST['emailben'];
          $montant = $_POST['montant'];
         $data = array(
              'email' => $email,
              'emailben' => $emailben,
              'montant' => $montant);
         $data1=json_encode($data);
         $url="http://127.0.0.1/apijwtm1/api.php?code=$code";
         $ch=curl_init($url);
         curl_setopt($ch,CURLOPT_HTTPPOST,true);
         curl_setopt($ch,CURLOPT_POSTFIELDS,$data1);
         curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
         curl_setopt($ch,CURLOPT_CUSTOMREQUEST,'PUT');
         $headers=array();
         $headers[]="Content-Type:application/json";
         $headers[]="Authorization:Bearer ".$jwt;
         curl_setopt($ch,CURLOPT_HTTPHEADER,$headers);
         $response1_json=curl_exec($ch);
         curl_close($ch);
         echo $response1_json;$email=$_SESSION['email'];
              $pass=$_SESSION['pass'];
         echo "<a href='essai1.php?email=$email&password=$pass' style='background-
color:white';>Page accueil </a>";
       }
else{ echo "Echec authentification"; echo "<a href='interfacepresi.php' style='background-
color:white';>Retour</a>";}
?>
</body>
</html>
```

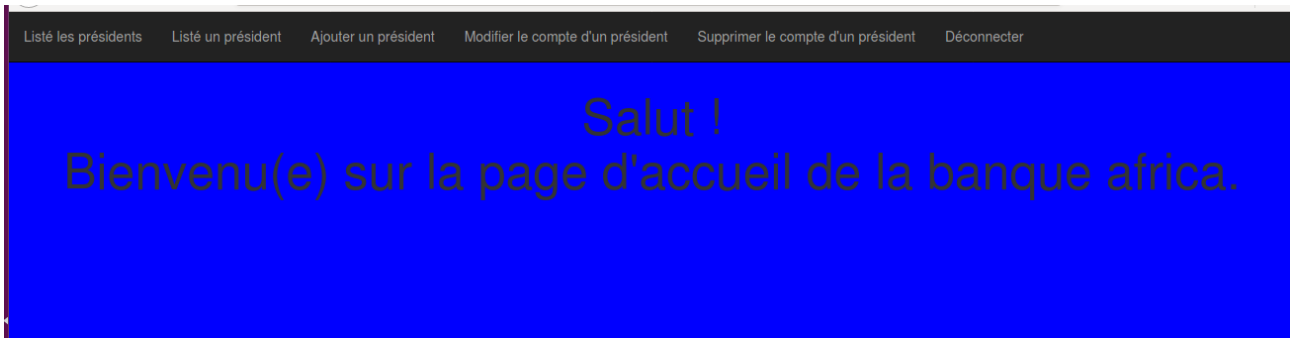
Interface de consommation de API

1- Coté employée

Interface de connexion d'un utilisateur : si c'est un employé de l'entreprise il peut fournir son email et mot de passe pour se connecter ; si c'est un simple user il faut qu'il clique sur le lien :
« connecter vous en tant que president »

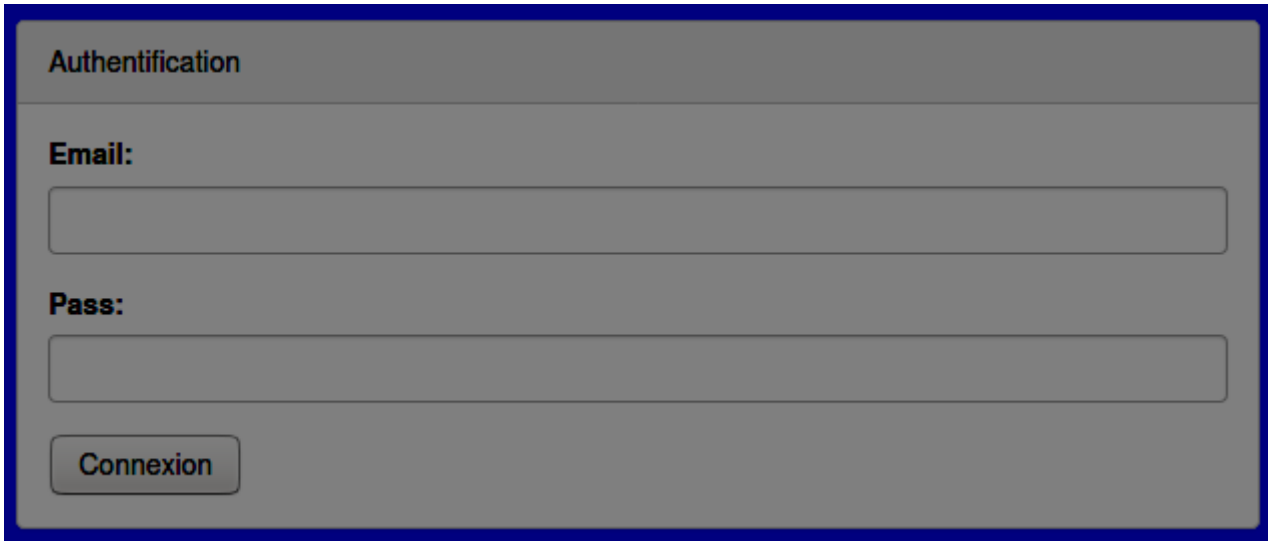


Si l'email et mot de passe correcte on affiche à l'employé la page suivantes à travers ou il peut effectuer des actions



2- Coté utilisateur

Interface de connexion d'un utilisateur



Si l'email et mot de passe correcte on affiche à l'utilisateur la page suivantes à travers ou il peut effectuer des actions

10 Conclusion générale:

Le métier de développeur d'interface d'API, tel que nous avons présenté, nécessite d'avoir des Connaissances en HTML, PHP.
Une bonne connaissance du protocole HTTP s'avère importante pour utiliser en bon escient les lignes de requêtes, les entêtes et les corps des requêtes pour transmettre des informations nécessaires.
L'extension curl de php php-curl fournit un client HTTP très riche permettant d'utiliser toutes les méthodes HTTP (GET,POST,PUT,DELETE, etc.) pour exploiter des ressources fournies par un serveur HTTP.
L'extension jwt de php php-jwt fournit un outil efficace pour gérer et exploiter les jetons de sécurité.
Aussi nous avons expérimenté la pertinence des variables super globales, notamment les sessions, dans le transport des données de page à page.