Département d'Informatique, L3, S6
Cours : Développement Mobile

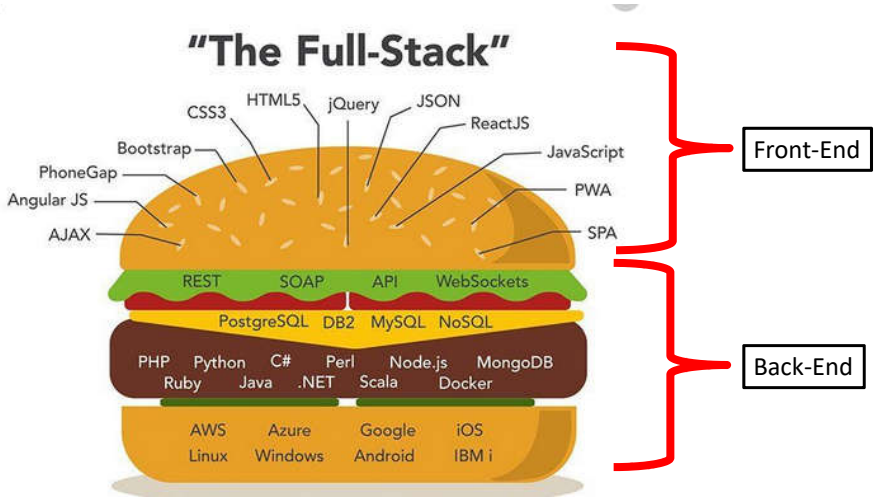
Accès à une base de données distante en utilisant le **REST** (**WEB**)

Université Badji Mokhtar Annaba
Email: sabri.ghazi@univ-annaba.dz

Accès à une base de données distante

- L'architecture d'une solution logicielle est généralement représentée sous forme d'une pile de couches :
  - Couche présentation , appelée aussi **Front-end**
  - **Couche Métier** et Persistance , appelée aussi **Back-end**.
- Il existent des technologies qui peuvent être utilisées dans chaque couche.

Full-Stack ?



Accès à une base de données via une **API REST**

- La plupart des applications mobile et Web utilisent une base de données centralisée dans un serveur.
- Comment accéder à une base de données distante (se trouve sur un serveur dans le réseau).
- L'accès directement à la base de données n'est pas possible, on ne peut pas stocker les mots de passe dans l'application (elle peut être décompilée).
  - **Solution** : Utiliser un **REST API**
  - C'est quoi ? : C'est un program qui permet d'effectuer des traitements et s'exécute dans un serveur et renvoi le résultat.

REST API

- REpresentational State Transfer
- On peut le considérer comme étant une fonction qui s'exécute dans le serveur.
- Invoqué par n'importe quel programme (mobile, desktop ou même un autre web service).
- Ils sont écrits en un langage **back-end** exemple : PHP, Python, Java, GO, Node.js etc

Schéma d'un accès **REST**

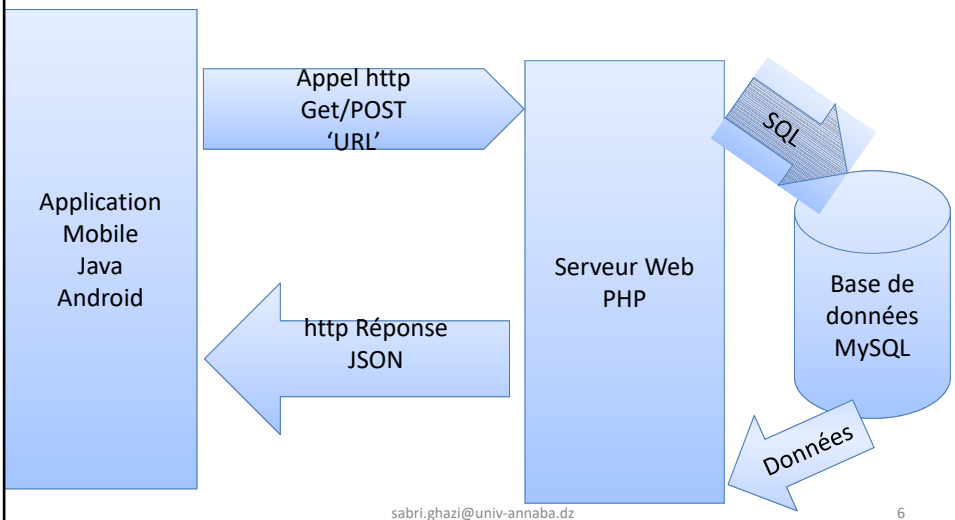
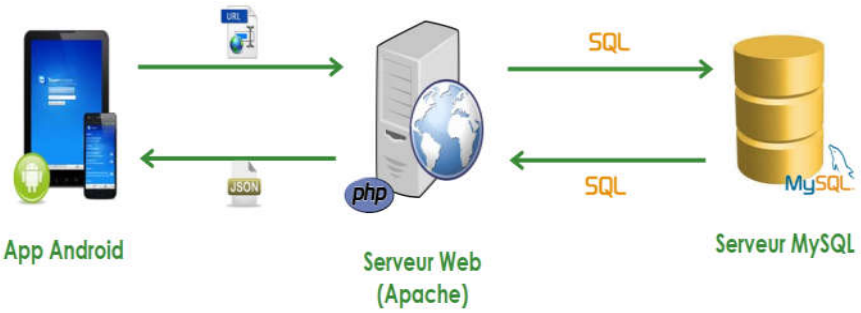


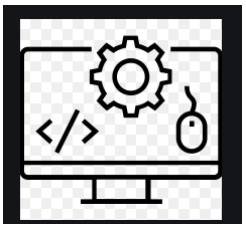
Schéma d'un accès **REST**

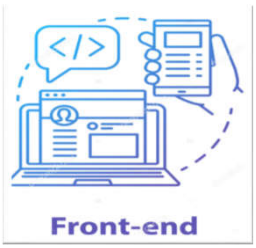


(crédit : **Dr. CHAOUCHE A. C.** , Université de Constantine).

Exemple d'une application Mobile Android qui récupère des données d'une base de données distante

- Etapes **Back-End**:
  - Installation du serveur WAMP.
  - Création de la base de donnée
  - Création du code PHP Back-End.
- Etapes **Front-End**
  - Création de l'application Android
  - Création de l'activité
  - Création de la classe **AsyncTask**
  - Lecture des résultats et affichage,





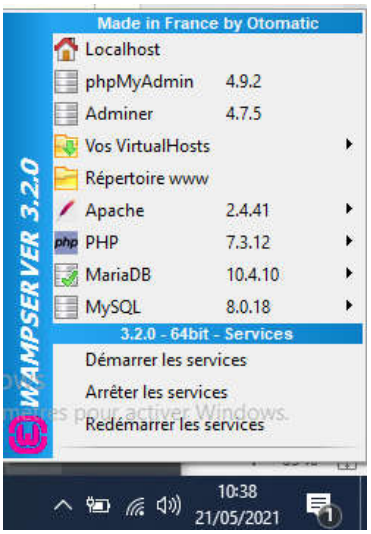
Etapes **Back-end**

Installation du serveur

- **WAMP** est un ensemble d'outils qui permet de transformer votre machine en un serveur , il inclut
  - Un serveur **Apache**
  - Un serveur **Php**
  - Un serveur de base de données **MySQL**
  - Une application pour gérer MySQL nommé **PhpMyAdmin**
- Lien de téléchargement
  - https://www.wampserver.com/

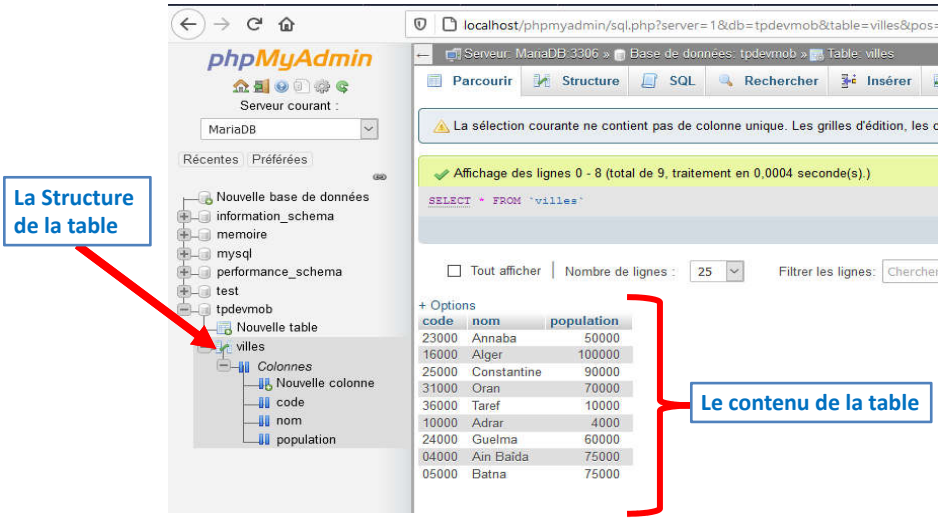
Etapes **Back-end**

Installation du serveur



Etapes Back-end

Création de la base de données:



Etapes Back-end

Le code php en Back-End

- Cette table contient des informations concernant les villes algériennes:
  - Le nom, le code postal et la population.
- Premièrement on doit écrire un script **AfficherVilles.php** qui permet de lister les villes et leurs informations.
- Ce fichier est placer dans le serveur dans un répertoire nommé « **DevMob** »

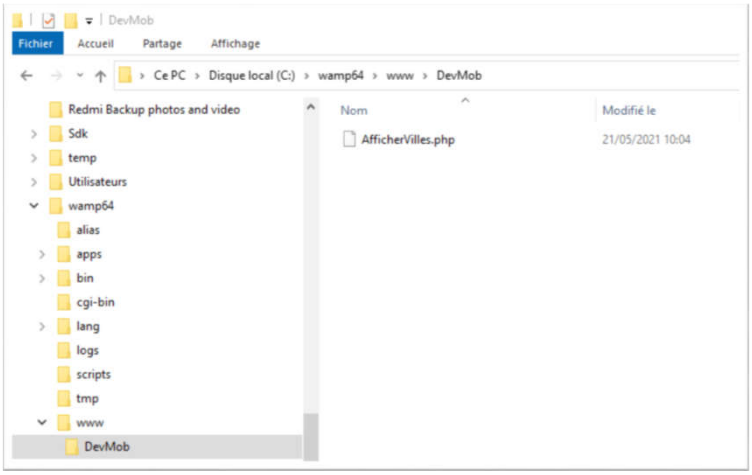
Etapes Back-end

Le programme AfficherVilles.php qui permet de transformer les données

```php
<?php
// Se connecter à la base de données.
$connection=mysqli_connect("localhost","root","","tpdevmob");
// envoyer la requetes SQL.
$response=mysqli_query($connection,"SELECT * FROM villes");

//Lire le résultat
$resultat=array();
while($data=mysqli_fetch_assoc($response))
{
    array_push($resultat,$data);
}
//fermer la connection.
mysqli_close($connection);
header('Content-Type:application/json');
//envoi du résultat en JSON.
echo json_encode($resultat);
http_response_code(200);
?>
```


Etapes Back-end

Configuration

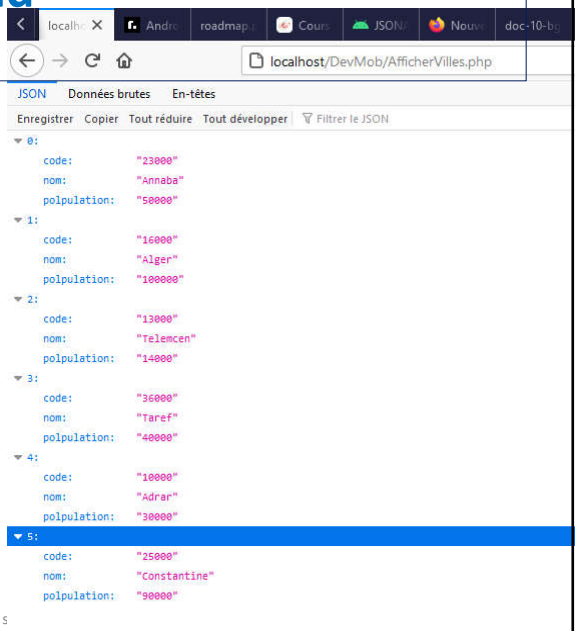


Etapes Back-end
Teste

- Si on lance la requête http dans notre navigateur web
- **http://localhost/DevMob/AfficherVilles.php**
- **On obtient le résultat en format JSON**


```
[
{"code":"23000","nom":"Annaba","population":"50000"},
{"code":"16000","nom":"Alger","population":"100000"},
{"code":"13000","nom":"Telemcen","population":"14000"},
{"code":"36000","nom":"Taref","population":"40000"},
{"code":"10000","nom":"Adrar","population":"30000"},
{"code":"25000","nom":"Constantine","population":"90000"}
]
```


Etapes Back-end
Configuration

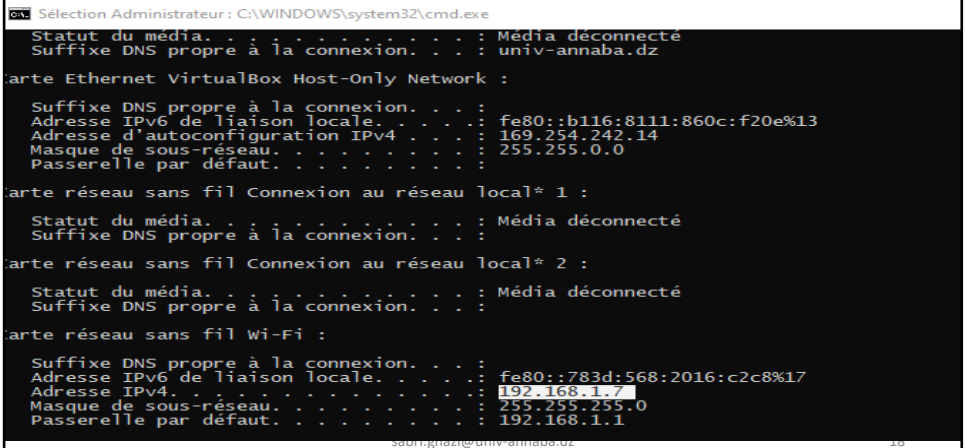


Configuration **Front-end**

- Il faut vérifier que la machine client (le téléphone) arrive à communiquer avec le serveur.
- **L'URL** http://localhost/DevMob/AfficherVilles.php
- Dans l'émulateur elle devient
  - http://10.0.2.2/DevMob/AfficherVilles.php
  - L'émulateur peut communiquer avec la machine de développement en utilisant cette adresse.
- Si vous utiliser un téléphone réel,
  - http://192.168.0.180/DevMob/AfficherVilles.php
  - il faut mettre l'adresse IP du serveur, utilisé **IPConfig** pour savoir l'adresse IP de votre machine.
  - il faut que les deux font partie du même réseau

Configuration **Front-End**

-lancer la ligne de commande et après lancer **IPConfig** pour savoir l'adresse **IP** de votre



Configuration Front-End

Pour tester que tout fonctionne bien,
On peut ouvrir l'URL dans le navigateur du téléphone
Comme illustré dans la figure

Front-End

L'application Android

- On cherche
  - À lancer une requête http, pour lancer le web service.
  - Récupérer le résultat.
  - Affiche ce résultat dans l'interface utilisateur.
- On aura besoin de
  - Donner les permissions a l'application pour se connecter.
  - Une classe Ville.java (contenant les attributs de même nom et type que la table de base de données).
  - écriture du programme qui lance la requête et récupère le résultat.

Front-End

Donner les permissions

- Il faut modifier le fichier Manifest.xml de l'application afin qu'elle puisse se connecter à une machine distante.
- Si non :java.lang.SecurityException: Permission denied (missing INTERNET

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.androiddev.listviewjson"
    android:versionCode="1"
    android:versionName="1.0" >
...
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
...
</manifest>
```


Front-End

Autoriser l'utilisation du HTTP au lieu de HTTPS

- Pour des mesures de sécurité, les version récente d'Android n'autorise pas a une application de faire des communication en Texte clair (non crypté), toute communication doit être en Https.
- Pour simplifier cette démonstration, on va utiliser juste une communication http.
- Cette autorisation se fait au niveau du fichier Manifest.xml comme suit :
- Dans la balise application on utilise l'attribut

```
android:usesCleartextTraffic="true"
```

Front-End Ville.java

```
public class Ville {
    String code;
    String nom;
    int population;


    public Ville(String code, String nom, int population) {
        this.code = code;
        this.nom = nom;
        this.population = population;
    }

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public int getPopulation() {
        return population;
    }

    public void setPopulation(int population) {
        this.population = population;
    }
}
```


Front-End

Dans notre application Android

- On doit développer la classe qui permet de récupérer les résultats d'un web service.
- Cette classe doit hériter la classe **AsyncTask.**
- L'invocation et la récupération des résultats d'un web service se fait en arrière plan, c'est à dire sans bloquer le Thread principal (Celui de l'interface utilisateur.

Front-End

La classe AsyncTask

- Elle permet de lancer des traitements asynchrone , c'est-à-dire dans un Thread autre que le Thread principal.
  - Pour ne pas bloquer l'interface utilisateur.
- Elle possède la méthode **doInBackground** :
  - fait le traitement en arrière plan.
- Est une méthode **onPostExecute** :
  - qui s'exécute après la fin du traitement. Elle est utilisé par récupérer le résultat.


```
public class ListerVilleNom extends AsyncTask<String, Integer, String> {
    ProgressDialog dialog;
    public static DisplayVille context;

    protected void onPreExecute() {
        dialog = ProgressDialog.show(context, "Please wait", "Loading Data from server");
        dialog.setCancelable(true);
    }

    @Override
    protected String doInBackground(String... params) {
        HttpURLConnection urlConnection = null;
        BufferedReader reader = null;
        StringBuffer strb = new StringBuffer();
        try {
            URL url = new URL("http://192.168.1.7/DevMob/AfficherVilles.php");
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setRequestMethod("GET");
            urlConnection.connect();
            InputStream inputStream = urlConnection.getInputStream();
            if (inputStream != null) {
                reader = new BufferedReader(new InputStreamReader(inputStream));
                String line;
                while ((line = reader.readLine()) != null) {
                    strb.append(line + "\n");
                }
            }
        } catch (Exception e) {
            Log.e("Error", e.getMessage());
        } finally {
            if (urlConnection != null)
                urlConnection.disconnect();
            if (reader != null) {
                try {
                    reader.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return strb.toString();
    }
```

Afficher un **ProgressDialog**

Adresse IP **Du serveur**

Ouverture de la connexion **L'URL**

Récupere la réponse **du serveur**

Fermer la **connexion**

La réponse **Du serveur**

Front-End

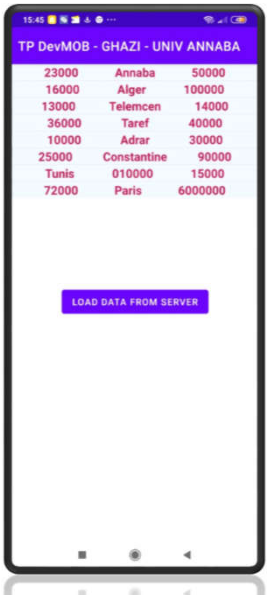
From JSON to java Object

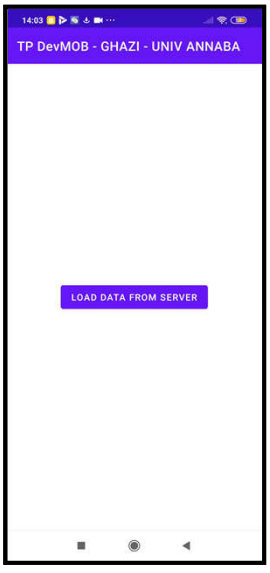
```
public  void  DisplayResult(String contents){

    JSONArray jsonarray = null;
    ArrayList<Ville> ret= new ArrayList<Ville>();
    try {
        jsonarray = new JSONArray(contents);
        for (int i = 0; i < jsonarray.length(); i++) {
            JSONObject jsonobject = jsonarray.getJSONObject(i);
            String code = jsonobject.getString("code");
            String nom = jsonobject.getString("nom");
            int population=jsonobject.getInt("population");
            ret.add(new Ville(code,nom,population));
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
    ListView lv=(ListView)findViewById(R.id.listView);
    lv.setAdapter(new VilleAdapter(this,ret));

}
```

Lire le contenu JSON
Et Transformer le en une collection d'objet de type Ville

Afficher la collection d'objet dans un compostant ListView

Démonstration





D'autres Technologies

- **Retrofit** : https://square.github.io/retrofit/
  - Permet de transformer une REST en une API très simple à utiliser.
- **ion** : https://github.com/koush/ion
  - Permet de charger les URL en asynchrone.