

# Documentation

Projet ANDROID : Gestion Immobilière

Réalisé en Ateliers Professionnels

## Table des matières

Contexte du projet.....	3
Rappel sur les besoins et les fonctionnalités de l'application .....	4
MLD .....	4
Diagramme de cas d'utilisation .....	6
Organisation du code .....	6
Documentation technique.....	8
Les fichiers PHP.....	8
Fichier connexion.php .....	8
Fichier getVisiteurs.php.....	9
Fichier getVisiteurById.php .....	9
Fichier addVisiteur.php.....	10
Fichier modifierVisiteurById.php.....	11
Fonction supprimerVisiteurById.php .....	12
Test des fichiers PHP via FileZilla .....	13
La classe métier : Visiteur .....	14
La classe DAO : VisiteurDAO .....	16
Le fichier HttpPostRequest .....	21
Les contrôleurs .....	22
MainActivity .....	22
PropositionActivity .....	23
AjoutActivity .....	24
ConsultActivity.....	26
DetailsActivity.....	28
ModifierActivity.....	32
Le fichier AndroidManifest .....	34
Mes interfaces .....	35

## Contexte du projet

Le projet **API : Gestion Visiteur** consiste en la conception et la mise en place d'une application mobile de gestion pour gérer des visiteurs. Cette application permettra de faciliter la gestion des visiteurs. De plus, il y aura la possibilité d'ajouter, de modifier ou de supprimer certaines informations à propos du visiteur voulu.

L'application sera conçue pour être facile à utiliser, avec une interface réservée seulement pour les admins pour l'instant. Le développement de l'application sera effectué en plusieurs étapes, en commençant par la conception d'un diagramme de classe sous forme de MLD pour pouvoir l'importer sur phpMyAdmin. Ensuite, des fichiers PHP seront créés pour ainsi pouvoir consulter, modifier, ajouter et supprimer un ou des visiteurs. Ces fichiers seront codés sous forme JSON. Cela permettra de stocker les informations structurées. De plus, cela sera utile pour transmettre les données entre mon application web et mon serveur.

## Rappel sur les besoins et les fonctionnalités de l'application MLD

J'ai d'abord dû réaliser un MLD (Modèle Logique de Données) contenant une seule relation (Visiteur).

J'ai aussi dû y insérer des données pour ainsi me faciliter la tâche plus tard.

```
-- phpMyAdmin SQL Dump
-- version 4.7.6
-- https://www.phpmyadmin.net/
--
-- Hôte : localhost
-- Généré le : mer. 13 juin 2021 à 17:11
-- Version du serveur : 10.0.34-MariaDB
-- Version de PHP : 7.0.29

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

-- Base de données : `infodevslam_ViticulteursAPI`


-----
-- Structure de la table `visiteur`


CREATE TABLE `visiteur` (
    `id` varchar(4) NOT NULL,
    `nom` varchar(30) DEFAULT NULL,
    `prenom` varchar(30) DEFAULT NULL,
    `login` varchar(30) DEFAULT NULL,
    `mdp` varchar(30) DEFAULT NULL,
    `adresse` varchar(30) DEFAULT NULL,
    `cp` varchar(30) DEFAULT NULL,
    `ville` varchar(30) DEFAULT NULL,
    `dateEmbauche` varchar(30) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

## Projet API : « Gestion Visiteur »

```
-- Déchargement des données de la table `visiteur`

-- INSERT INTO `visiteur` (`id`, `nom`, `prenom`, `login`, `mdp`, `adresse`, `cp`, `ville`, `dateEmbauche`) VALUES
('a131', 'Villechalane', 'Louis', 'aribia', 'aaaa', '8 rue des textmes', '46000', 'Cahors', '2005-12-21'),
('a17', 'Andre', 'David', 'dandre', 'oppg5', '1 rue Petit', '46200', 'Lalbenque', '1998-11-23'),
('a55', 'Bedos', 'Christian', 'cbedos', 'gmhxdu', '1 rue Peranud', '46250', 'Montcuq', '1995-01-12'),
('a93', 'Tusseau', 'Louis', 'ltusseau', 'ktp3s', '22 rue des Ternes', '46123', 'Gramat', '2000-05-01'),
('b13', 'Bentot', 'Pascal', 'pbentot', 'doyw1', '11 allée des Cerises', '46512', 'Bessines', '1992-07-09'),
('b16', 'Bioret', 'Luc', 'lbioret', 'hrjfs', '1 Avenue gambetta', '46000', 'Cahors', '1998-05-11'),
('b19', 'Bunisset', 'Francis', 'fbunisset', '4vbnd', '10 rue des Perles', '93100', 'Montreuil', '1987-10-21'),
('b25', 'Bunisset', 'Denise', 'dbunisset', 's1y1r', '23 rue Manin', '75019', 'paris', '2010-12-05');

-- Index pour les tables déchargées
--

-- Index pour la table `visiteur`
--
ALTER TABLE `visiteur`
| ADD PRIMARY KEY (`id`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Une fois le modèle logique de données fini, je l'ai importé sur phpMyAdmin.

	<input type="button" value="←"/>	<input type="button" value="→"/>		<input type="button" value="▼"/>	<b>id</b>	<b>nom</b>	<b>prenom</b>	<b>login</b>	<b>mdp</b>	<b>adresse</b>	<b>cp</b>	<b>ville</b>	<b>dateEmbauche</b>
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	a131	Villechalane	Louis	aribia	aaaa	8 rue des textmes	46000	Cahors	2005-12-21	
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	a17	Andre	David	dandre	oppg5	1 rue Petit	46200	Lalbenque	1998-11-23	
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	a55	Bedos	Christian	cbedos	gmhxdu	1 rue Peranud	46250	Montcuq	1995-01-12	
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	a93	Tusseau	Louis	ltusseau	ktp3s	22 rue des Ternes	46123	Gramat	2000-05-01	
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	b13	Bentot	Pascal	pbentot	doyw1	11 allée des Cerises	46512	Bessines	1992-07-09	
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	b16	Bioret	Luc	lbioret	hrjfs	1 Avenue gambetta	46000	Cahors	1998-05-11	
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	b19	Bunisset	Francis	fbunisset	4vbnd	10 rue des Perles	93100	Montreuil	1987-10-21	
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	b25	Bunisset	Denise	dbunisset	s1y1r	23 rue Manin	75019	paris	2010-12-05	

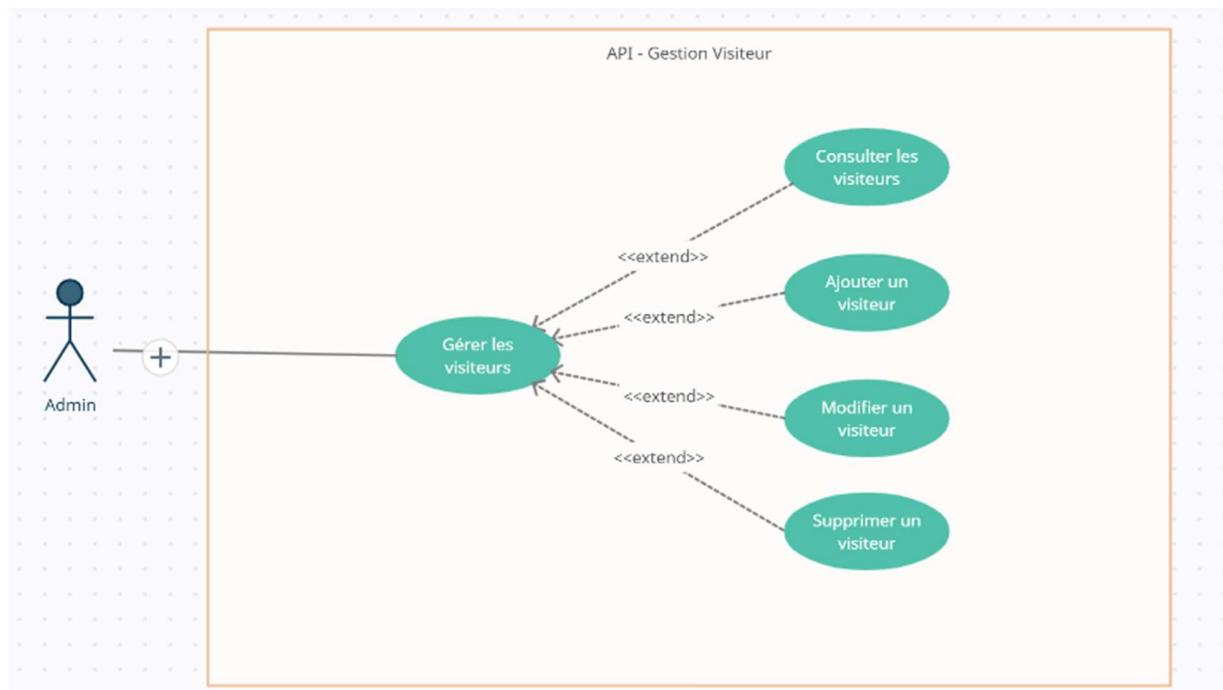
## Projet API : « Gestion Visiteur »

### Diagramme de cas d'utilisation

Après avoir réalisé mon MLD et avoir compris les besoins de l'administrateur lors de sa gestion des visiteurs, je les ai adapté sous forme de diagramme de cas d'utilisation.

L'administrateur après s'être connecté devra pouvoir gérer :

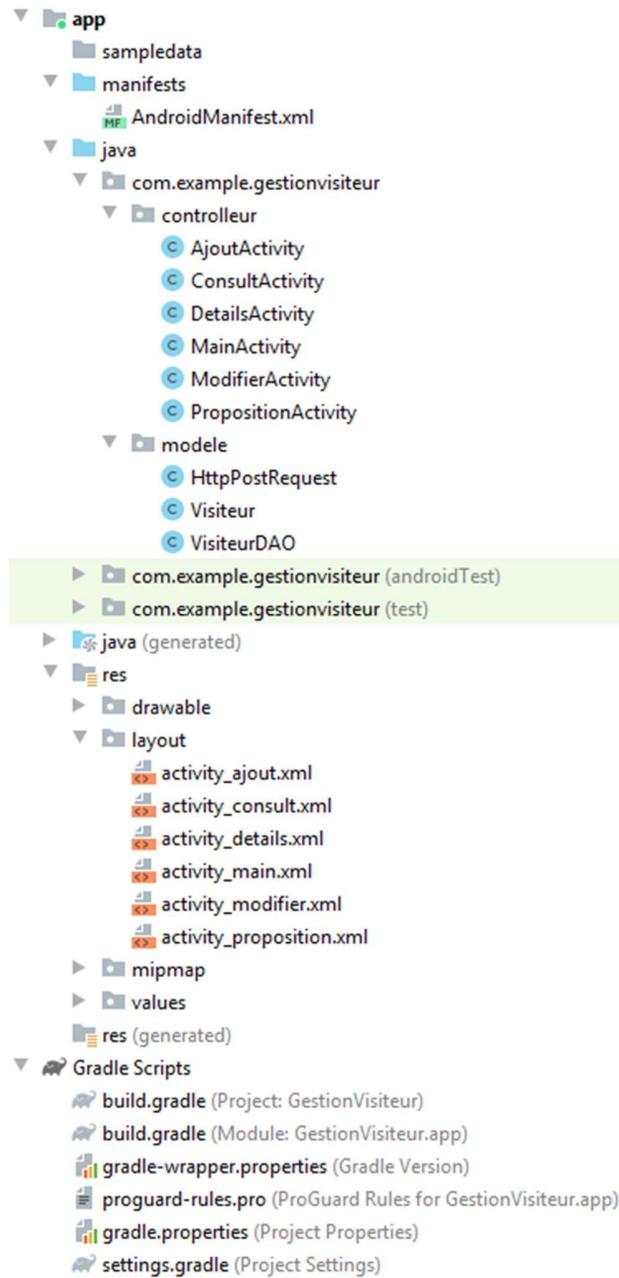
- Les visiteurs (consulter, ajouter, modifier, supprimer).



### Organisation du code

Le projet a été réalisé selon l'architecture MVC (Modèle - Vue - Contrôleur), ce qui permet de séparer et d'organiser efficacement les modèles, les contrôleurs qui les utilisent, ainsi que les vues liées aux contrôleurs.

## Projet API : « Gestion Visiteur »



## Documentation technique

### Les fichiers PHP

Avant de faire mon application mobile, j'ai dû coder des fichiers PHP pour pouvoir faire des tests et vérifier que tout fonctionnait bien. Pour faire cela, j'ai fait des requêtes préparées. Cela permet d'exécuter la même requête plusieurs fois mais protège aussi des injections SQL.

#### Fichier connexion.php

```
<?php
function connexionPDO(){
    $login = "iluobe";
    $mdp = '████████';
    $bd = "iluobe_bddvisiteur";
    $serveur = "mysql-iluobe.alwaysdata.net";

try {

    $connect = new PDO("mysql:host=$serveur; dbname=$bd", $login, $mdp);

    return $connect;

} catch(PDOException $e){
    print "connexion à la base de données impossible";
    die();
}
?>
```

## Projet API : « Gestion Visiteur »

### Fichier getVisiteurs.php

```
<?php

try {
    $login = "iluobe";
    $mdp = "████████";
    $bd = "iluobe_bddvisiteur";
    $serveur = "mysql-iluobe.alwaysdata.net";
    $connect = new PDO("mysql:host=$serveur; dbname=$bd", $login, $mdp);

    // Insertion de la requête préparé dans $req
    $req = $connect->prepare("SELECT * FROM visiteur");

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

    // PDO::FETCH_ASSOC : Retourne une ligne de données de l'ensemble de résultats et la renvoie sous forme de tableau associatif.
    while ($ligne = $req->fetch(PDO::FETCH_ASSOC)){
        $res[] = $ligne;
    }

    // Affiche le résultat au format JSON
    print(json_encode($res));

} catch(PDOException $e){
    print "Erreur !: ". $e->getMessage();
    die();
}
?>
```

### Fichier getVisiteurById.php

```
<?php

$id = $_GET['id'];

try {
    $login = "iluobe";
    $mdp = "████████";
    $bd = "iluobe_bddvisiteur";
    $serveur = "mysql-iluobe.alwaysdata.net";
    $connect = new PDO("mysql:host=$serveur; dbname=$bd", $login, $mdp);

    // Insertion de la requête préparé dans $req
    $req=$connect->prepare("SELECT * FROM visiteur WHERE idV = :idV");
    // bindParam lie les paramètres aux noms de variables utilisé
    $req->bindParam(':idV',$id);

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

    // PDO::FETCH_ASSOC : Retourne une ligne de données de l'ensemble de résultats et la renvoie sous forme de tableau associatif.
    while ($ligne = $req->fetch(PDO::FETCH_ASSOC)){
        $res[] = $ligne;
    }

    // Affiche le résultat au format JSON
    print(json_encode($res));

} catch(PDOException $e){
    | | print "Erreur !: ". $e->getMessage();
    die();
}
?>
```

## Projet API : « Gestion Visiteur »

### Fichier addVisiteur.php

```
<?php

$id = $_GET['id'];
$nom = $_GET['nom'];
$prenom = $_GET['prenom'];
$login = $_GET['login'];
$mdp = $_GET['mdp'];
$adresse = $_GET['adresse'];
$cp = $_GET['cp'];
$ville = $_GET['ville'];
$dateEmbauche = $_GET['dateEmbauche'];

try {

    $loginCompte = "iluobe";
    $mdpCompte = '██████████';
    $bd = "iluobe_bddvisiteur";
    $serveur = "mysql-iluobe.alwaysdata.net";
    $connect = new PDO("mysql:host=$serveur; dbname=$bd", $loginCompte, $mdpCompte);

    // Insertion de la requête préparé dans $req
    $req=$connect->prepare("INSERT INTO `visiteur` (`id`, `nom`, `prenom`, `login`, `mdp`, `adresse`, `cp`, `ville`, `dateEmbauche`)
values (:idV, :nom, :prenom, :login, :mdp, :adresse, :cp, :ville, :dateEmbauche)");

    // bindParam lie les paramètres aux noms de variables utilisé
    $req->bindParam(':idV',$id);
    $req->bindParam(':nom',$nom);
    $req->bindParam(':prenom',$prenom);
    $req->bindParam(':login',$login);
    $req->bindParam(':mdp',$mdp);
    $req->bindParam(':adresse',$adresse);
    $req->bindParam(':cp',$cp);
    $req->bindParam(':ville',$ville);
    $req->bindParam(':dateEmbauche',$dateEmbauche);

    //Exécution de la requête
    $res = $req->execute();

    // Affiche le résultat au format JSON
    print(json_encode($res));

} catch(PDOException $e){
    print "Erreur !: ". $e->getMessage();
    die();
}
?>
```

Fichier modifierVisiteurById.php

```
<?php

try {
    $login = "iluobe";
    $mdp = "████████";
    $bd = "iluobe_bddvisiteur";
    $serveur = "mysql-iluobe.alwaysdata.net";
    $connect = new PDO("mysql:host=$serveur; dbname=$bd", $login, $mdp);

    // Insertion de la requête préparé dans $req
    $req = $connect->prepare("UPDATE visiteur SET nom = :nom WHERE id = :idV");

    // bindParam lie les paramètres aux noms de variables utilisés
    $req->bindParam(':idV', $id);
    $req->bindParam(':nom', $nom);

    $id = isset($_GET["id"]) ? htmlspecialchars($_GET["id"]) : "";
    $nom = isset($_GET["nom"]) ? htmlspecialchars($_GET["nom"]) : "";
    $prenom = isset($_GET["prenom"]) ? htmlspecialchars($_GET["prenom"]) : "";
    $login = isset($_GET["login"]) ? htmlspecialchars($_GET["login"]) : "";
    $mdp = isset($_GET["mdp"]) ? htmlspecialchars($_GET["mdp"]) : "";
    $adresse = isset($_GET["adresse"]) ? htmlspecialchars($_GET["adresse"]) : "";
    $cp = isset($_GET["cp"]) ? htmlspecialchars($_GET["cp"]) : "";
    $ville = isset($_GET["ville"]) ? htmlspecialchars($_GET["ville"]) : "";
    $dateEmbauche = isset($_GET["dateEmbauche"]) ? htmlspecialchars($_GET["dateEmbauche"]) : "";

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

    // Récupère l'ensemble des informations de la requête
    $recup = $req->fetchAll();

    // Affiche le résultat au format JSON
    echo json_encode($recup);

    echo("C'est good !!!");

} catch(PDOException $e){
    print "Erreur !: ". $e->getMessage();
    die();
}
?>
```

Fonction supprimerVisiteurById.php

```
<?php

try{
    $login = "iluobe";
    $mdp = "████████████████";
    $bd = "iluobe_bddvisiteur";
    $serveur = "mysql-iluobe.alwaysdata.net";
    $connect = new PDO("mysql:host=$serveur; dbname=$bd", $login, $mdp);

    $id = isset($_GET["id"]) ? htmlspecialchars($_GET["id"]) : "";

    // Insertion de la requête préparé dans $req
    $sql = ("Delete FROM visiteur where id = '$id'");

    // On prépare la requête
    $step = $connect->prepare($sql);

    // On exécute la requête
    $step->execute();

    // On récupère les résultats
    $recup = $step->fetchAll();

    // On affiche le résultat sous format JSON
    echo json_encode($recup);

    echo("OK");

} catch(PDOException $e){
    print "Erreur !: ". $e->getMessage();
    die();
}
?>
```

## Projet API : « Gestion Visiteur »

### Test des fichiers PHP via FileZilla

Pour pouvoir tester les fichiers PHP, je devais me connecter au serveur de ma base de données et transférer les fichiers un par un. A chaque modification apportée, je devais remettre le nouveau fichier mis à jour. Ensuite, sur un navigateur web, je devais rentrer l'url suivante :

<https://sjpslam.alwaysdata.net/owen/API/nomDuFichierVoulu.php>

Cela a été possible car j'utilisais la méthode GET sur tous mes fichiers PHP.

The screenshot shows the FileZilla interface. At the top, there are fields for 'Hôte' (am.alwaysdata.net), 'Identifiant' (sjpslam\_owen), 'Mot de passe' (redacted), 'Port' (22), and a 'Connexion rapide' button. Below these, a status window displays the following messages:

- Statut : Initialisation de TLS...
- Statut : Vérification du certificat...
- Statut : Connexion TLS établie.
- Statut : Connecté
- Statut : Récupération du contenu du dossier...
- Statut : Contenu du dossier "/" affiché avec succès

The left panel shows the local directory structure under 'Site local': C:\Users\owen.lluobe\, containing folders like ., .., openlluobe, profilwin10, Public, Umap, visualcode, wamp64, Windows, WinPython, iccas, D:\Data, and E:(OWEN). The right panel shows the remote directory structure under 'Site distant': /, containing a single folder named 'API'. Below these panels is a table comparing local and remote files:

Nom de fichier	Taille de fiche...	Type de fiche...	Dernière modif...	Nom de fichier	Taille de fiche...	Type de fiche...	Dernière modif...	Droits d'accès	Propriétaire...
addVisiteur.php	1 291	Ficher PHP	28/03/2023 12:05:30	addVisiteur.php	1 241	Ficher PHP	31/03/2023 11:...	adfrw (6664)	sjpslam_ow...
connecteVisiteur.php	722	Ficher PHP	27/03/2023 10:59:43	connecteVisiteur.php	681	Ficher PHP	07/04/2023 10...	adfrw (6664)	sjpslam_ow...
Connexion.php	381	Ficher PHP	27/03/2023 09:16:51	Connexion.php	341	Ficher PHP	31/03/2023 11...	adfrw (6664)	sjpslam_ow...
getVisiteurByid.php	98	Ficher PHP	27/03/2023 09:22:26	getVisiteurByid.php	734	Ficher PHP	31/03/2023 11:...	adfrw (6664)	sjpslam_ow...
getVisiteurs.php	765	Ficher PHP	27/03/2023 08:36:24	getVisiteurs.php	831	Ficher PHP	07/04/2023 10:...	adfrw (6664)	sjpslam_ow...
modifVisiteurByid.php	1 237	Ficher PHP	27/03/2023 11:02:16	modifVisiteurByid.php	1 262	Ficher PHP	06/04/2023 16...	adfrw (6664)	sjpslam_ow...
supVisiteurByid.php	773	Ficher PHP	27/03/2023 10:28:06	RQgetVisiteur.php	674	Ficher PHP	27/04/2023 13...	adfrw (6664)	sjpslam_ow...
				supVisiteurByid.php	724	Ficher PHP	07/04/2023 10...	adfrw (6664)	sjpslam_ow...

Below the table, a message says 'Sélection de 1 fichier. Taille totale : 703 octets'. The bottom navigation bar includes tabs for 'Serveur / Fichier local', 'Direction', 'Fichier distant', 'Taille', 'Priorité', and 'Statut'. There are also tabs for 'Fichiers en file d'attente', 'Transferts échoués', and 'Transferts réussis (1)'.

## La classe métier : Visiteur

Une fois les fichiers test créés et testés, j'ai pu commencer la création de mon application web. De ce fait, j'ai d'abord dû créer la classe métier de Visiteur.

```
public class Visiteur {  
    private String id;  
    private String nom;  
    private String prenom;  
    private String login;  
    private String mdp;  
    private String adresse;  
    private String cp;  
    private String ville;  
    private String date;  
  
    //Constructeur  
    public Visiteur(String id, String nom, String prenom, String login, String mdp, String adresse,  
                    String cp, String ville, String date) {  
        this.id = id;  
        this.nom = nom;  
        this.prenom = prenom;  
        this.login = login;  
        this.mdp = mdp;  
        this.adresse = adresse;  
        this.cp = cp;  
        this.ville = ville;  
        this.date = date;  
    }  
}
```

Projet API : « Gestion Visiteur »

```
public Visiteur(String nom, String prenom, String login, String mdp, String adresse, String cp,
                String ville, String date) {
    this.nom = nom;
    this.prenom = prenom;
    this.login = login;
    this.mdp = mdp;
    this.adresse = adresse;
    this.cp = cp;
    this.ville = ville;
    this.date = date;
}

//Getter et Setter
public String getId() { return id; }

public void setId(String id) { this.id = id; }

public String getNom() { return nom; }

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

public String getPrenom() { return prenom; }

public void setPrenom(String prenom) { this.prenom = prenom; }

public String getLogin() { return login; }

public void setLogin(String login) { this.login = login; }

public String getMdp() { return mdp; }

public void setMdp(String mdp) { this.mdp = mdp; }

public String getAdresse() { return adresse; }

public void setAdresse(String adresse) { this.adresse = adresse; }

public String getCp() { return cp; }

public void setCp(String cp) { this.cp = cp; }

public String getVille() { return ville; }

public void setVille(String ville) { this.ville = ville; }

public String getDate() { return date; }

public void setDate(String date) { this.date = date; }

@Override
public String toString() { return (nom + " " + prenom + " " + id); }
}
```

## La classe DAO : VisiteurDAO

Ensute, j'ai créé « VisiteurDAO » qui me permet d'accéder aux données des visiteurs se trouvant sur ma base de données. De plus, c'est dans ce fichier aussi que je fais appel aux fonctions PHP crée au préalable.

```
public class VisiteurDAO {

    public VisiteurDAO(AjoutActivity ajoutActivity) {
    }

    public VisiteurDAO(ConsultActivity consultActivity){
    }

    public VisiteurDAO(DetailsActivity detailsActivity){
    }

    public VisiteurDAO(ModifierActivity modifierActivity){
    }

    // Fonction qui permet d'ajouter un visiteur
    public String addVisiteur(Visiteur unVisiteur) {
        String result = "";

        //adresse de l'URL de l'API à interroger et fichier php permettant d'ajouter un visiteur
        String myUrl="https://sjpslam.alwaysdata.net/owen/API/addVisiteur.php";

        //informations à transmettre pour effectuer l'ajout
        String params =
            "id=" +unVisiteur.getId() + "\n"
            + "&nom=" +unVisiteur.getNom() + "\n"
            + "&prenom=" +unVisiteur.getPrenom() + "\n"
            + "&login=" +unVisiteur.getLogin() + "\n"
            + "&mdp=" +unVisiteur.getMdp() + "\n"
            + "&adresse=" +unVisiteur.getAdresse() + "\n"
            + "&cp=" +unVisiteur.getCp() + "\n"
            + "&ville=" +unVisiteur.getVille() + "\n"
```

Projet API : « Gestion Visiteur »

```
+ "&dateEmbauche="+unVisiteur.getDate();
Log.d("requete",params);

HttpPostRequest postRequest = new HttpPostRequest();
try{
    result = postRequest.execute(new String []{myUrl, params}).get();
}
catch (InterruptedException e) {
    e.printStackTrace();
} catch (ExecutionException e) {
    e.printStackTrace();
}
Log.d("resultat",result);
return result;
}

// Fonction qui récupère une collection de visiteur
public ArrayList<Visiteur> getVisiteurs() throws JSONException{

String result = "";

//adresse de l'URL de l'API à interroger permettant d'afficher le tous les visiteurs
String myUrl="https://sjpslam.alwaysdata.net/owen/API/getVisiteurs.php";
String params = "";
HttpPostRequest postRequest = new HttpPostRequest();
try{
    result = postRequest.execute(new String []{myUrl, params}).get();
}
catch (InterruptedException e) {
    e.printStackTrace();
} catch (ExecutionException e) {
```

Projet API : « Gestion Visiteur »

```
e.printStackTrace();
}

ArrayList<Visiteur> visiteurs = new ArrayList<~>();
JSONArray array = new JSONArray(result);
for (int i=0; i<array.length();i++){
    String id=array.getJSONObject(i).getString("id");
    String nom=array.getJSONObject(i).getString("nom");
    String prenom=array.getJSONObject(i).getString("prenom");
    String login=array.getJSONObject(i).getString("login");
    String mdp=array.getJSONObject(i).getString("mdp");
    String adresse=array.getJSONObject(i).getString("adresse");
    String cp=array.getJSONObject(i).getString("cp");
    String ville=array.getJSONObject(i).getString("ville");
    String date=array.getJSONObject(i).getString("dateEmbauche");
    visiteurs.add(new Visiteur(id,nom,prenom,login,mdp,adresse,cp,ville,date));
}
return visiteurs;
}

// Fonction qui permet de modifier un visiteur en fonction de l'id
public String modifierVisiteurById(Visiteur unVisiteur, String idV) {
    String result = "";
    /*
        Adresse de l'URL de l'API à interroger et fichier php permettant de modifier un visiteur
        en fonction de l'id
    */
    String myUrl="https://sjpslam.alwaysdata.net/owen/API/modifVisiteurById.php";

    // Informations à transmettre pour effectuer la modification du visiteur
    String params =
        "id="+ idV
```

Projet API : « Gestion Visiteur »

```
+ "&nom=" + unVisiteur.getNom() +
"&prenom=" + unVisiteur.getPrenom() +
"&login=" + unVisiteur.getLogin() +
"&mdp=" + unVisiteur.getMdp() +
"&adresse=" + unVisiteur.getAdresse() +
"&cp=" + unVisiteur.getCp() +
"&ville=" + unVisiteur.getVille() +
"&dateEmbauche=" + unVisiteur.getDate() ;

Log.d("requete", params);

HttpPostRequest postRequest = new HttpPostRequest();
try{
    result = postRequest.execute(new String []{myUrl, params}).get();

}
catch (InterruptedException e) {
    e.printStackTrace();
} catch (ExecutionException e) {
    e.printStackTrace();
}
return result;
}

// Fonction qui permet de supprimer un visiteur en fonction de l'id
public String supprimerVisiteurById(Visiteur unVisiteur, String idV) {
String result = "";
/*
    Adresse de l'URL de l'API à interroger et fichier php permettant de modifier un visiteur
    en fonction de l'id
*/

```

Projet API : « Gestion Visiteur »

```
String myUrl="https://sjpslam.alwaysdata.net/owen/API/supVisiteurById.php";

// Informations à transmettre pour effectuer la suppression du visiteur
String params =
    "id="+ idV
    +"&nom="+unVisiteur.getNom()+
    "&prenom="+unVisiteur.getPrenom()+
    "&login="+unVisiteur.getLogin()+
    "&mdp="+unVisiteur.getMdp()+
    "&adresse="+unVisiteur.getAdresse()+
    "&cp="+unVisiteur.getCp()+
    "&ville="+unVisiteur.getVille()+
    "&dateEmbauche="+unVisiteur.getDate() ;

Log.d("requete",params);

HttpPostRequest postRequest = new HttpPostRequest();
try{
    result = postRequest.execute(new String []{myUrl, params}).get();

}
catch (InterruptedException e) {
    e.printStackTrace();
} catch (ExecutionException e) {
    e.printStackTrace();
}
return result;
}
}
```

## Le fichier HttpPostRequest

La méthode de requête HTTP POST est utilisée pour envoyer des données au serveur ou pour créer ou mettre à jour ou supprimer des données. Pour ce faire, j'ai donc dû donc mettre à jour mes fichiers PHP en changeant la méthode GET en méthode POST.

Voici donc le fichier HttpPostRequest :

```

public class HttpPostRequest extends AsyncTask<String, Void, String>{
    public static final String REQUEST_METHOD = "POST";
    public static final int READ_TIMEOUT = 15000;
    public static final int CONNECTION_TIMEOUT = 15000;
    private HttpURLConnection connection;

    @Override
    protected String doInBackground(String... params) {
        String targetURL = params[0];
        String urlParameters = params[1];
        Log.d("url",targetURL);
        Log.d("urlParametres",urlParameters);
        URL url;
        connection = null;
        try {
            //Create connection
            url = new URL(targetURL);
            connection = (HttpURLConnection)url.openConnection();
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
            connection.setRequestProperty("Content-Length", "" + Integer.toString(urlParameters.getBytes().length));
            connection.setRequestProperty("Content-Language", "fr-FR");

            connection.setUseCaches (false);
            connection.setDoInput(true);
            connection.setDoOutput(true);

            //Get Response
            InputStream is = connection.getInputStream();
            BufferedReader rd = new BufferedReader(new InputStreamReader(is));
            String line;
            StringBuffer response = new StringBuffer();
            while((line = rd.readLine()) != null) {
                response.append(line);
                response.append('\r');
            }
            rd.close();
            Log.d("reponse",response.toString());
            return response.toString();

        } catch (Exception e) {

            e.printStackTrace();
            return null;
        } finally {

            if(connection != null) {
                connection.disconnect();
            }
        }
    }
}

```

## Les contrôleurs

J'ai dû réaliser 6 contrôleurs pour pouvoir réaliser mon application web.

### MainActivity

Ce contrôleur est la page d'accueil de mon application.

```
public class MainActivity extends AppCompatActivity {

    //Déclaration des variables de classe
    private Button acceder;
    private ImageView image;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        // Méthode permettant de récupérer l'activité en cas de crash de l'application
        super.onCreate(savedInstanceState);
        // Méthode associant cette page au fichier XML choisi
        setContentView(R.layout.activity_main);

        // Lien avec les fichiers de l'interface XML
        acceder = findViewById(R.id.btnAcceder);
        image = findViewById(R.id.logoGestionVisiteur);

        // Permet l'affichage de l'image choisi dans le fichier XML
        image.setImageResource(R.drawable.logo_gestionvisiteur);

        // Gestion du bouton 'Valider'
        acceder.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(packageContext: MainActivity.this, PropositionActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

## Projet API : « Gestion Visiteur »

### PropositionActivity

Ce contrôleur permet d'accéder à la fonctionnalité concernant les visiteurs (consulter, ajouter, modifier et supprimer).

```
public class PropositionActivity extends Activity {

    //Déclaration des variables de classe
    private Button ajouter, consulter, modifier;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        // Méthode permettant de récupérer l'activité en cas de crash de l'application
        super.onCreate(savedInstanceState);
        // Méthode associant cette page au fichier XML choisi
        setContentView(R.layout.activity_proposition);

        // Lien avec les fichiers de l'interface XML
        ajouter = findViewById(R.id.btnAjouter);
        consulter = findViewById(R.id.btnConsulter);
        modifier = findViewById(R.id.btnModifier);

        // Gestion du bouton 'Ajouter' qui renvoie vers la page 'AjoutActivity'
        ajouter.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent intent2 = new Intent(PropositionActivity.this, AjoutActivity.class);
                startActivity(intent2);
            }
        });

        // Gestion du bouton 'Consulter' qui renvoie vers la page 'ConsultActivity'
        consulter.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent c = new Intent(PropositionActivity.this, ConsultActivity.class);
                startActivity(c);
            }
        });

        // Gestion du bouton 'Modifier' qui renvoie vers la page 'ModifierActivity'
        modifier.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent m = new Intent(PropositionActivity.this, ModifierActivity.class);
                startActivity(m);
            }
        });
    }
}
```

## AjoutActivity

Ce contrôleur permet d'ajouter un visiteur dans ma base de données.

```
public class AjoutActivity extends Activity {

    //Déclaration des variables de classe
    private Button ajoutValider;
    private EditText unId, unNom, unPrenom, unLogin, unMdp, uneAdresse, unCp, uneVille, uneDate;
    private String id, nom, prenom, login, mdp, adresse, cp, ville, date;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        // Méthode permettant de récupérer l'activité en cas de crash de l'application
        super.onCreate(savedInstanceState);
        // Méthode associant cette page au fichier XML choisi
        setContentView(R.layout.activity_ajout);

        // Lien avec les fichiers de l'interface XML
        unId = findViewById(R.id.editTextId);
        unNom = findViewById(R.id.editTextNom);
        unPrenom = findViewById(R.id.editTextPre);
        unLogin = findViewById(R.id.editTextLogin);
        unMdp = findViewById(R.id.editTextMdp);
        uneAdresse = findViewById(R.id.editTextAdRue);
        unCp = findViewById(R.id.editTextCP);
        uneVille = findViewById(R.id.editTextVille);
        uneDate = findViewById(R.id.editTextDate);
        ajoutValider = findViewById(R.id.btnValiderAjout);
    }
}
```

Projet API : « Gestion Visiteur »

```
//Création d'objet de la classe VisiteurDAO
VisiteurDAO visiteurAcces = new VisiteurDAO(ajoutActivity: this);

// Gestion du bouton 'Valider'
ajoutValider.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {

    id = unId.getText().toString();
    nom = unNom.getText().toString();
    prenom = unPrenom.getText().toString();
    login = unLogin.getText().toString();
    mdp = unMdp.getText().toString();
    adresse = uneAdresse.getText().toString();
    cp = unCp.getText().toString();
    ville = uneVille.getText().toString();
    date = uneDate.getText().toString();

    // Création d'un nouveau visiteur
    Visiteur newVisiteur = new Visiteur(id, nom, prenom, login, mdp, adresse, cp, ville, date);

    // Insertion dans la base de données du nouveau visiteur
    visiteurAcces.addVisiteur(newVisiteur);

    // Insertion dans la base de données du nouveau visiteur
    visiteurAcces.addVisiteur(newVisiteur);

    // Condition pour vérifier que l'ajout du visiteur
    String result = "";
    if(result.equals("1\r")){
        Context c = getApplicationContext();
        Toast msg = Toast.makeText(c, text: "ERREUR : Echec de l'ajout du client !", Toast.LENGTH_LONG);
        msg.show();

    } else {
        Context c = getApplicationContext();
        Toast msg = Toast.makeText(c, text: "Le client a bien été ajouté !", Toast.LENGTH_LONG);
        Intent i = new Intent(packageContext: AjoutActivity.this, PropositionActivity.class);
        startActivity(i);
        msg.show();

    }
}
};

});
```

### ConsultActivity

Ce contrôleur permet de consulter la liste des visiteurs dans ma base de données.

```
public class ConsultActivity extends AppCompatActivity {

    //Déclaration des variables de classe
    private Button retour;
    private ListView listeViewVisiteur;
    private ArrayList<Visiteur> listeVisiteur;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        // Méthode permettant de récupérer l'activité en cas de crash de l'application
        super.onCreate(savedInstanceState);
        // Méthode associant cette page au fichier XML choisi
        setContentView(R.layout.activity_consult);

        // Lien avec les fichiers de l'interface XML
        listeViewVisiteur = findViewById(R.id.list);
        retour = findViewById(R.id.btnRetour);

        //Création d'objet de la classe VisiteurDAO
        VisiteurDAO visiteurAcces = new VisiteurDAO(consultActivity: this);

        try {
            // Récupération de la liste des visiteurs
            listeVisiteur = visiteurAcces.getVisiteurs();
        } catch (JSONException e) {
            e.printStackTrace();
        }

        // Test en boucle pour vérifier la liste des visiteurs s'affiche
        for(Visiteur unVisiteur : listeVisiteur){
            Log.d(tag: "msg", unVisiteur.toString());
        }

        // Création d'un adaptateur pour les visiteurs
        ArrayAdapter lesVisiteurs = new ArrayAdapter(context: this, android.R.layout.simple_list_item_1, listeVisiteur);

        // Applique l'adaptateur à la ListView des Réservations
        listeViewVisiteur.setAdapter(lesVisiteurs);
    }
}
```

Projet API : « Gestion Visiteur »

```
// Affichage de la listView contenant toutes les visiteurs de la BDD
listeViewVisiteur.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Visiteur clickedItem = (Visiteur) listeViewVisiteur.getAdapter().getItem(position);
        // Récupère toutes les informations du visiteur quand on clique dessus
        Intent i = new Intent( mContext: ConsultActivity.this, DetailsActivity.class);
        i.putExtra( name: "id", clickedItem.getId());
        i.putExtra( name: "nom", clickedItem.getNom());
        i.putExtra( name: "prenom", clickedItem.getPrenom());
        i.putExtra( name: "login", clickedItem.getLogin());
        i.putExtra( name: "mdp", clickedItem.getMdp());
        i.putExtra( name: "adresse", clickedItem.getAdresse());
        i.putExtra( name: "cp", clickedItem.getCp());
        i.putExtra( name: "ville", clickedItem.getVille());
        i.putExtra( name: "dateEmbauche", clickedItem.getDate());
        startActivity(i);
    }
});

// Gestion du bouton 'Retour' qui renvoie vers la page 'PropositionActivity'
retour.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent ret = new Intent( mContext: ConsultActivity.this, PropositionActivity.class);
        startActivity(ret);
    }
});
}
}
```

## DetailsActivity

Ce contrôleur permet de modifier et supprimer un visiteur dans ma base de données.

```
public class DetailsActivity extends AppCompatActivity {

    //Déclaration des variables de classe
    private Button valider, supprimer;
    private VisiteurDAO recuperVisiteur;
    private Visiteur leVisiteur;
    private String idV, nom, prenom, login, mdp, adresse, cp, ville, date;
    private EditText idRecu, nomRecu, prenomRecu, loginRecu, mdpRecu, adresseRecu, cpRecu, villeRecu, dateRecu;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        // Méthode permettant de récupérer l'activité en cas de crash de l'application
        super.onCreate(savedInstanceState);
        // Méthode associant cette page au fichier XML choisi
        setContentView(R.layout.activity_details);

        // Lien avec les fichiers de l'interface XML
        idRecu = findViewById(R.id.editTextId);
        nomRecu = findViewById(R.id.editTextNom);
        prenomRecu = findViewById(R.id.editTextPre);
        loginRecu = findViewById(R.id.editTextLogin);
        mdpRecu = findViewById(R.id.editTextMdp);
        adresseRecu = findViewById(R.id.editTextAdRue);
        cpRecu = findViewById(R.id.editTextCP);
        villeRecu = findViewById(R.id.editTextVille);
        dateRecu = findViewById(R.id.editTextDate);
        valider = findViewById(R.id.btnValiderAjout);
        supprimer = findViewById(R.id.btnSupprimer);
```

Projet API : « Gestion Visiteur »

```
// Récupération des informations du visiteur cliquée au préalable
idV = getIntent().getStringExtra( name: "id");
nom = getIntent().getStringExtra( name: "nom");
prenom = getIntent().getStringExtra( name: "prenom");
login = getIntent().getStringExtra( name: "login");
mdp = getIntent().getStringExtra( name: "mdp");
adresse = getIntent().getStringExtra( name: "adresse");
cp = getIntent().getStringExtra( name: "cp");
ville = getIntent().getStringExtra( name: "ville");
date = getIntent().getStringExtra( name: "dateEmbauche");

// Modification des données du visiteur
idReçu.setText(idV);
nomReçu.setText(nom);
prenomReçu.setText(prenom);
loginReçu.setText(login);
mdpReçu.setText(mdp);
adresseReçu.setText(adresse);
cpReçu.setText(cp);
villeReçu.setText(ville);
dateReçu.setText(date);

//Création d'objet de la classe VisiteurDAO
VisiteurDAO visiteurAcces = new VisiteurDAO( detailsActivity: this);

// Gestion du bouton 'Valider'
valider.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

Projet API : « Gestion Visiteur »

```
String rId = idRecu.getText().toString();
String rNom=nomRecu.getText().toString();
String rPrenom=prenomRecu.getText().toString();
String rLogin=loginRecu.getText().toString();
String rMdp=mdpRecu.getText().toString();
String rAdresse=adresseRecu.getText().toString();
String rCp=cpRecu.getText().toString();
String rVille=villeRecu.getText().toString();
String rDate=dateRecu.getText().toString();

// Récupération des information du visiteur à modifier
Visiteur newVisiteur = new Visiteur(rId, rNom, rPrenom, rLogin, rMdp, rAdresse, rCp, rVille, rDate);

// Modification du visiteur grâce à la fonction
String result = visiteurAcces.modifierVisiteurById(newVisiteur, idV);

// Condition pour vérifier que la modification du visiteur
if (result.equals("1\r")) {
    Context c = getApplicationContext();
    Toast msg = Toast.makeText(c, text: "Veuillez saisir un nom pour ce visiteur.", Toast.LENGTH_LONG);
    msg.show();
    Intent i = new Intent( packageContext: DetailsActivity.this, PropositionActivity.class);
    startActivity(i);
} else {
    Context c = getApplicationContext();
    Toast msg = Toast.makeText(c, text: "Votre visiteur a été modifiée.", Toast.LENGTH_LONG);
    Intent i = new Intent( packageContext: DetailsActivity.this, ConsultActivity.class);
    startActivity(i);
    msg.show();
}

}
```

Projet API : « Gestion Visiteur »

```
    }
});

// Gestion du bouton 'Supprimer'
supprimer.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String rId = idRecu.getText().toString();
        String rNom=nomRecu.getText().toString();
        String rPrenom=prenomRecu.getText().toString();
        String rLogin=loginRecu.getText().toString();
        String rMdp=mdpRecu.getText().toString();
        String rAdresse=adresseRecu.getText().toString();
        String rCp=cpRecu.getText().toString();
        String rVille=villeRecu.getText().toString();
        String rDate=dateRecu.getText().toString();

        // Récupération des informations du visiteur à supprimer
        Visiteur newVisiteur = new Visiteur(rId, rNom, rPrenom, rLogin, rMdp, rAdresse, rCp, rVille, rDate);

        // Suppression du visiteur grâce à la fonction
        String result = visiteurAcces.supprimerVisiteurById(newVisiteur, idV);

        // Condition pour vérifier que la suppression du visiteur
        if (result.equals("1|r")) {
            Context c = getApplicationContext();
            Toast msg = Toast.makeText(c, text: "Erreur !!!", Toast.LENGTH_LONG);
            msg.show();
            Intent i = new Intent( packageContext: DetailsActivity.this, PropositionActivity.class);
            startActivity(i);
        } else {
            Context c = getApplicationContext();
            Toast msg = Toast.makeText(c, text: "Votre visiteur a été supprimé.", Toast.LENGTH_LONG);
            Intent i = new Intent( packageContext: DetailsActivity.this, ConsultActivity.class);
            startActivity(i);
            msg.show();
        }
    }
});
```

## ModifierActivity

Ce contrôleur permet aussi de modifier un visiteur en fonction de son id. La différence avec le contrôleur « DetailsActivity » est que ce contrôleur possède un spinner qui nous permet de choisir directement le visiteur qu'on veut modifier.

```
public class ModifierActivity extends AppCompatActivity {

    //Déclaration des variables de classe
    private Spinner spinVisiteur;
    private ArrayList<Visiteur> lesVisiteurs;
    private EditText txtNom,txtPrenom,txtLogin,txtMdp,txtAdresse,txtCP,txtVille,txtDateE;
    private Visiteur leVisiteur;
    private Button boutonModif;
    private VisiteurDAO unVisiteurDAO;

    /*
    Méthode permettant d'afficher le visiteur en fonction de son id choisi dans le spinner
    Actualise le visiteur à chaque changement d'id dans le spinner
    */
    public void actualiser(int position){
        txtNom.setText(lesVisiteurs.get(position).getNom());
        txtPrenom.setText(lesVisiteurs.get(position).getPrenom());
        txtLogin.setText(lesVisiteurs.get(position).getLogin());
        txtMdp.setText(lesVisiteurs.get(position).getMdp());
        txtAdresse.setText(lesVisiteurs.get(position).getAdresse());
        txtCP.setText(lesVisiteurs.get(position).getCp());
        txtVille.setText(lesVisiteurs.get(position).getVille());
        txtDateE.setText(lesVisiteurs.get(position).getDate());
    }
}
```

## Projet API : « Gestion Visiteur »

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    // Méthode permettant de récupérer l'activité en cas de crash de l'application
    super.onCreate(savedInstanceState);
    // Méthode associant cette page au fichier XML choisi
    setContentView(R.layout.activity_modifier);

    // Lien avec les fichiers de l'interface XML
    txtNom = findViewById(R.id.editTextNom);
    txtPrenom = findViewById(R.id.editTextPre);
    txtLogin = findViewById(R.id.editTextLogin);
    txtMdp = findViewById(R.id.editTextMdp);
    txtAdresse = findViewById(R.id.editTextAdRue);
    txtCP = findViewById(R.id.editTextCP);
    txtVille = findViewById(R.id.editTextVille);
    txtDateE = findViewById(R.id.editTextDate);
    spinVisiteur = findViewById(R.id.spinner);
    boutonModif= findViewById(R.id.btnValiderAjout);

    // Création d'objet de la classe VisiteurDAO
    unVisiteurDAO=new VisiteurDAO( modifierActivity: this);

    // Insertion de la liste des visiteurs dans un tableau
    lesVisiteurs = new ArrayList<Visiteur>();

    try {
        lesVisiteurs=unVisiteurDAO.getVisiteurs();
    } catch (JSONException e) {
        e.printStackTrace();
    }
    // Création d'un adaptateur pour les visiteurs
    ArrayAdapter adapter = new ArrayAdapter( context: this, android.R.layout.simple_spinner_item, lesVisiteurs);

    // Applique l'adaptateur à la listView des Réservations
    spinVisiteur.setAdapter(adapter);

    // Récupération des données du visiteur choisi
    txtNom.setText(lesVisiteurs.get(0).getNom());
    txtPrenom.setText(lesVisiteurs.get(0).getPrenom());
    txtLogin.setText(lesVisiteurs.get(0).getLogin());
    txtMdp.setText(lesVisiteurs.get(0).getMdp());
    txtAdresse.setText(lesVisiteurs.get(0).getAdresse());
    txtCP.setText(lesVisiteurs.get(0).getCP());
    txtVille.setText(lesVisiteurs.get(0).getVille());
    txtDateE.setText(lesVisiteurs.get(0).getDate());

    // Gestion de la valeur de l'id du spinner visiteur
    spinVisiteur.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> adapterView, View view, int position, long l) {
            leVisiteur = (Visiteur) adapterView.getItemAtPosition(position);
            Toast.makeText(getApplicationContext(), text: "ID : " + leVisiteur.getId() +
                "\n Nom du visiteur: " + leVisiteur.getNom(), Toast.LENGTH_SHORT).show();

            actualiser(position);
        }
        @Override
        public void onNothingSelected(AdapterView<?> adapterView) {
        }
    });
}
```

## Le fichier AndroidManifest

Le fichier manifeste fournit des informations essentielles sur mon application. De plus, ce fichier permet aussi de déclarer les autorisations que mon application doit avoir pour accéder aux données de ma base de données.

Dans mon fichier AndroidManifest.xml, j'ai dû rajouter 2 autorisations :

- android.permission.INTERNET → permet aux applications d'ouvrir des sockets réseau.
- android.permission.ACCESS\_NETWORK\_STATE → permet aux applications d'accéder à des informations sur les réseaux.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="GestionVisiteur"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.GestionVisiteur">
        <activity android:name=".controlleur.MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".controlleur.PropositionActivity"></activity>
        <activity android:name=".controlleur.AjoutActivity"></activity>
        <activity android:name=".controlleur.ConsultActivity"></activity>
        <activity android:name=".controlleur.DetailsActivity"></activity>
        <activity android:name=".controlleur.ModifierActivity"></activity>
    </application>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
</manifest>
```

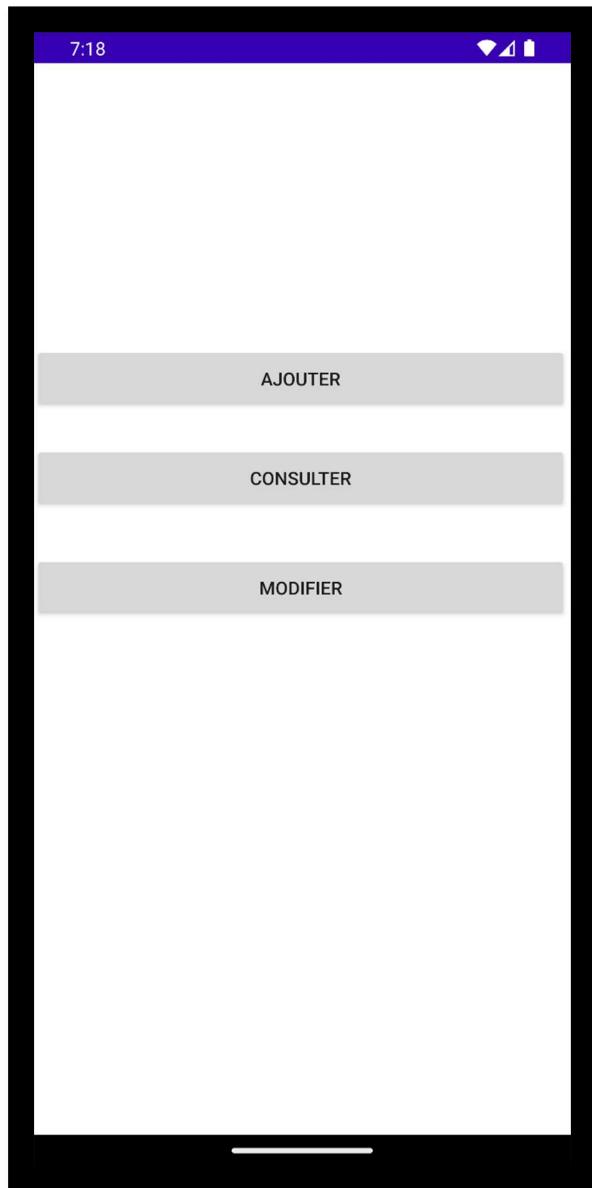
## Mes interfaces

Voici le résultat final de mes interfaces réalisés.

L'interface de la page d'accueil :

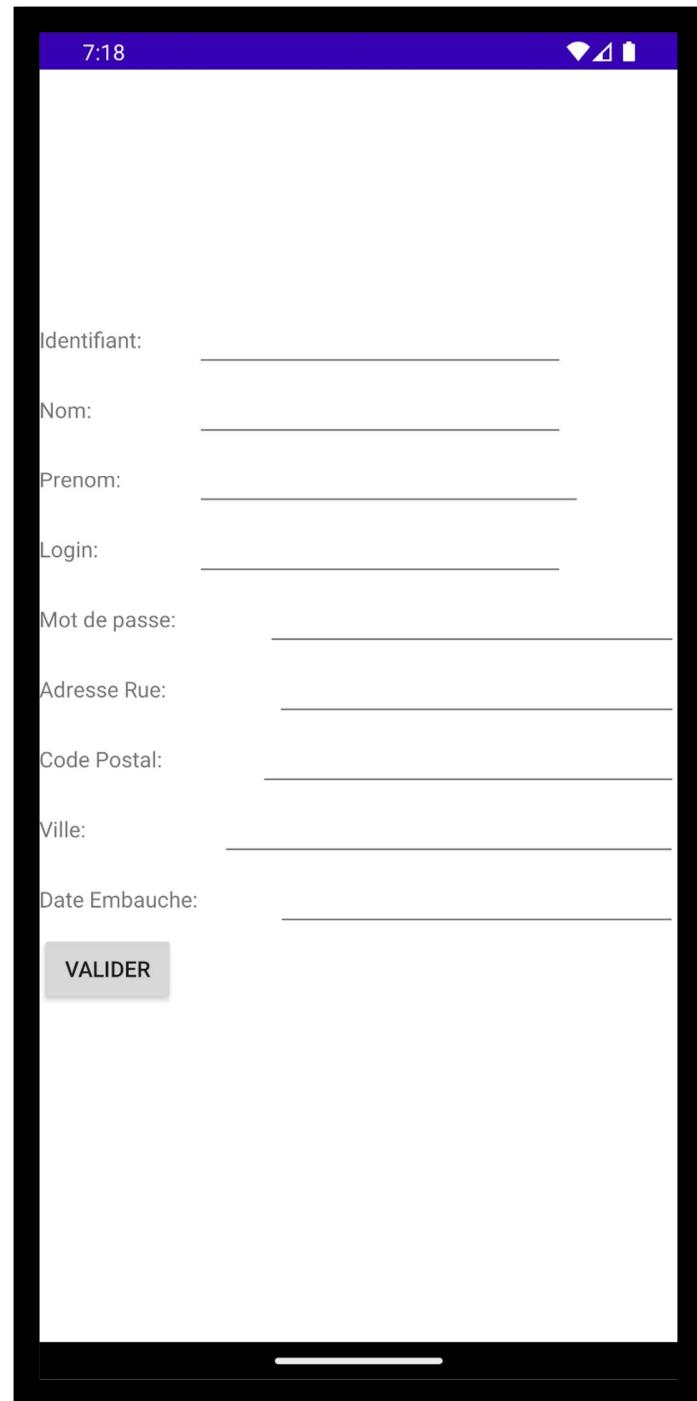


L'interface de ma page de proposition :

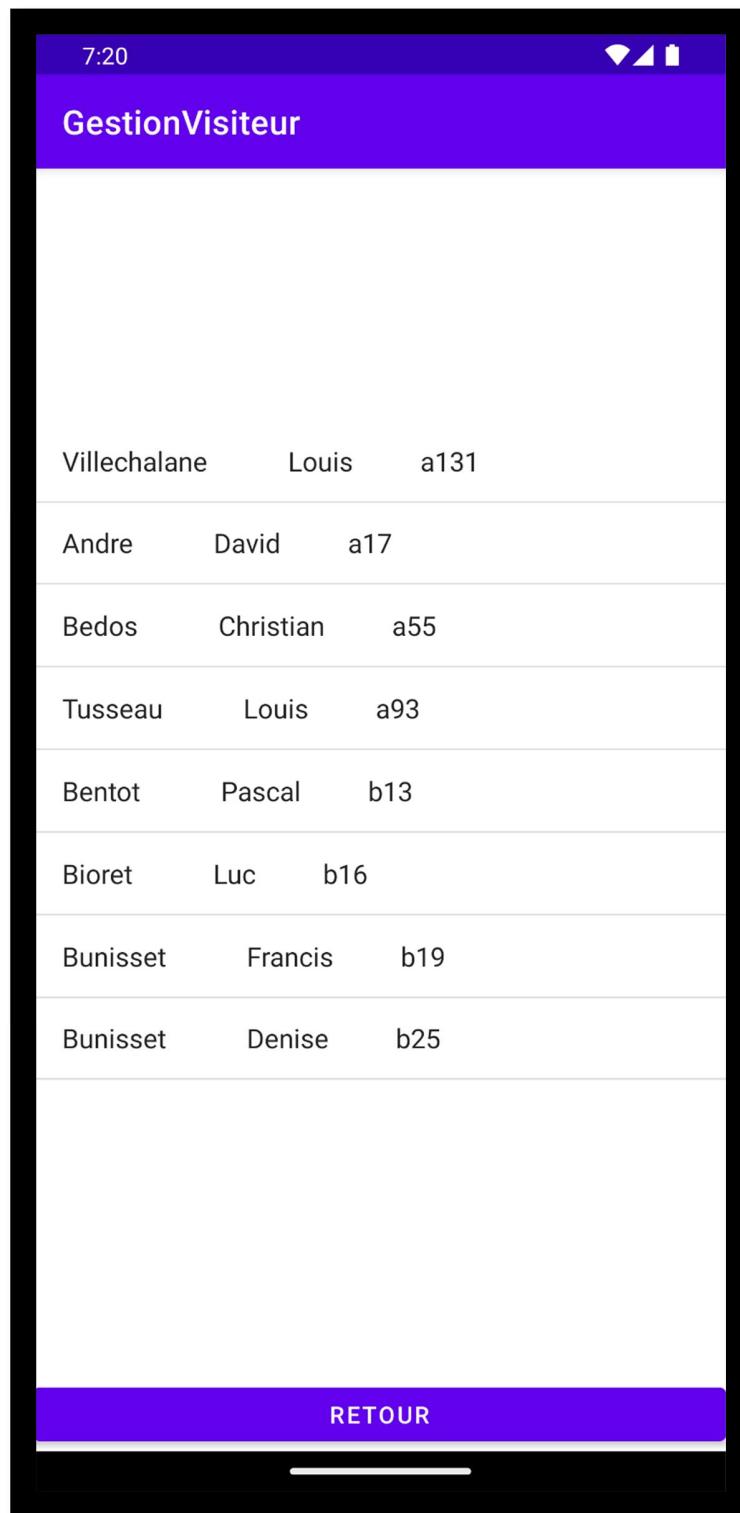


Projet API : « Gestion Visiteur »

L'interface pour l'ajout d'un visiteur :

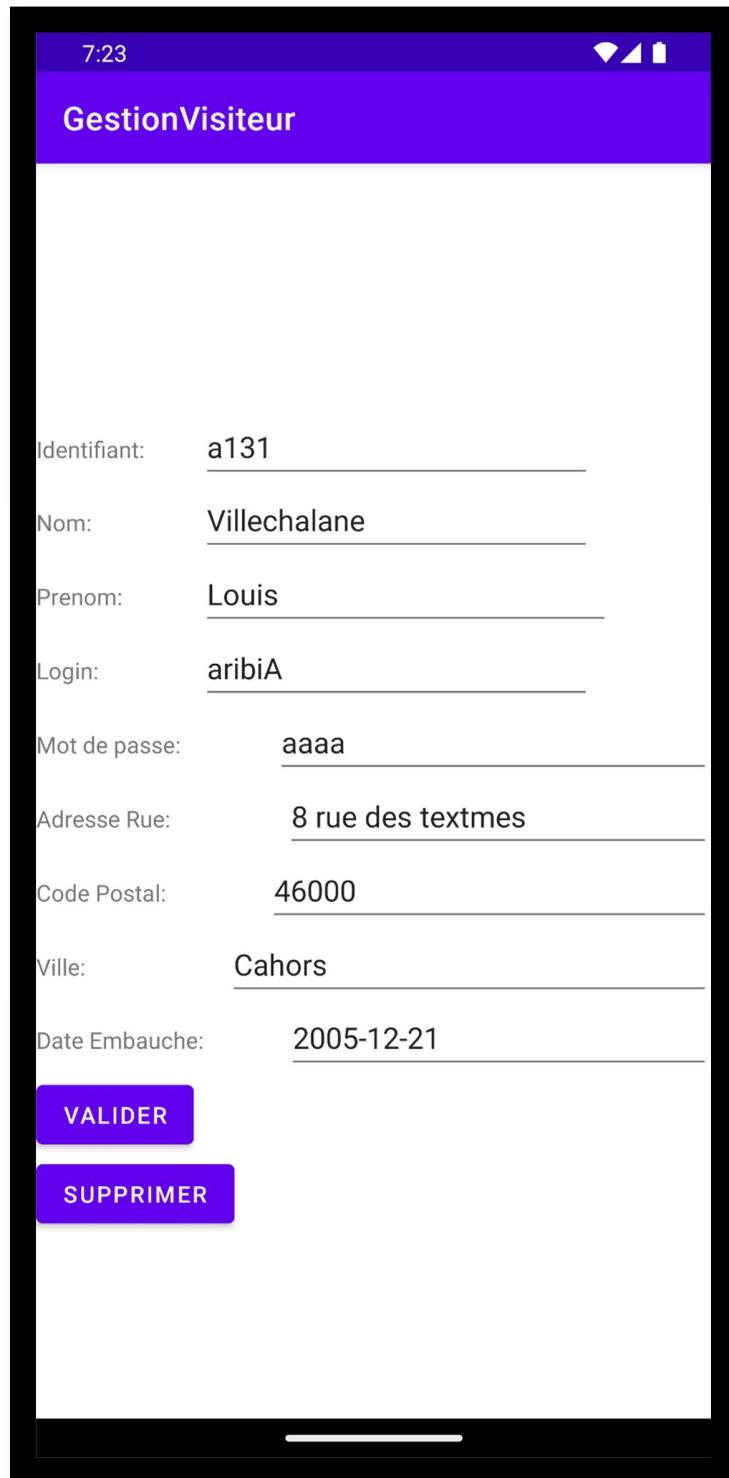


L'interface pour la consultation des visiteurs :



Projet API : « Gestion Visiteur »

L'interface pour la modification et la suppression d'un visiteur :



Projet API : « Gestion Visiteur »

L'interface pour la modification d'un visiteur en fonction de son id :

