**Documentation**

Projet ANDROID : Gestion Immobilière

Réalisé en Ateliers Professionnels

Table des matières

[Contexte du projet 3](#_Toc133853609)

[Rappel sur les besoins et les fonctionnalités de l’application 4](#_Toc133853610)

[MLD 4](#_Toc133853611)

[Diagramme de cas d’utilisation 6](#_Toc133853612)

[Organisation du code 6](#_Toc133853613)

[Documentation technique 8](#_Toc133853614)

[Les fichiers PHP 8](#_Toc133853615)

[Fichier connexion.php 8](#_Toc133853616)

[Fichier getVisiteurs.php 9](#_Toc133853617)

[Fichier getVisiteurById.php 9](#_Toc133853618)

[Fichier addVisiteur.php 10](#_Toc133853619)

[Fichier modifierVisiteurById.php 11](#_Toc133853620)

[Fonction supprimerVisiteurById.php 12](#_Toc133853621)

[Test des fichiers PHP via FileZilla 13](#_Toc133853622)

[La classe métier : Visiteur 14](#_Toc133853623)

[La classe DAO : VisiteurDAO 16](#_Toc133853624)

[Le fichier HttpPostRequest 21](#_Toc133853625)

[Les contrôleurs 22](#_Toc133853626)

[MainActivity 22](#_Toc133853627)

[PropositionActivity 23](#_Toc133853628)

[AjoutActivity 24](#_Toc133853629)

[ConsultActivity 26](#_Toc133853630)

[DetailsActivity 28](#_Toc133853631)

[ModifierActivity 32](#_Toc133853632)

[Le fichier AndroidManifest 34](#_Toc133853633)

[Mes interfaces 35](#_Toc133853634)

# 

# 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 forma JSON. Cela permettra de stocker les informations structurées. De plus, cela sera utile pour transmettre les données entre mon application web et mon serveur.

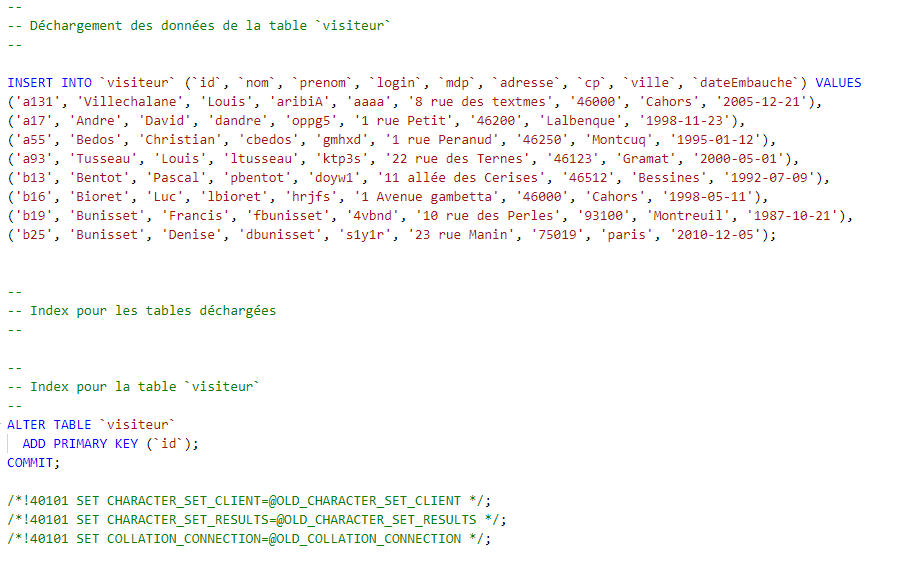
# Rappel sur les besoins et les fonctionnalités de l’application

## MLD

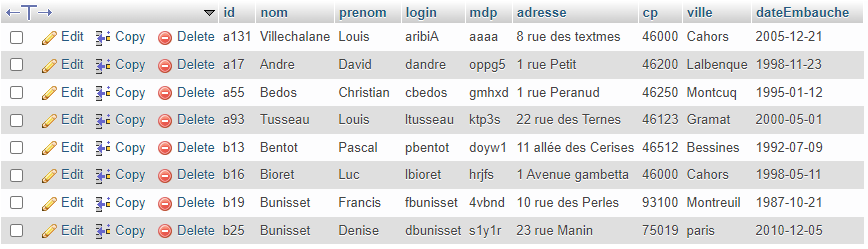
J’ai d’abord dû réaliser un MLD (Modèle Logique de Données) contenant une seule relation (Visiteur).

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





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

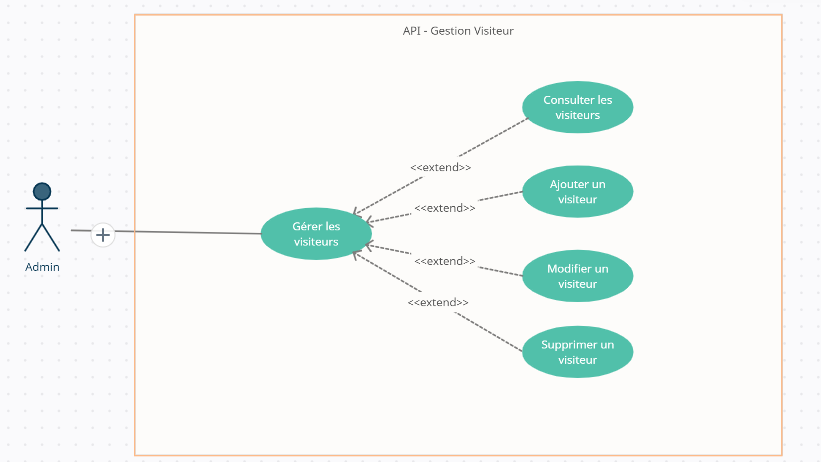


## Diagramme de cas d’utilisation

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.

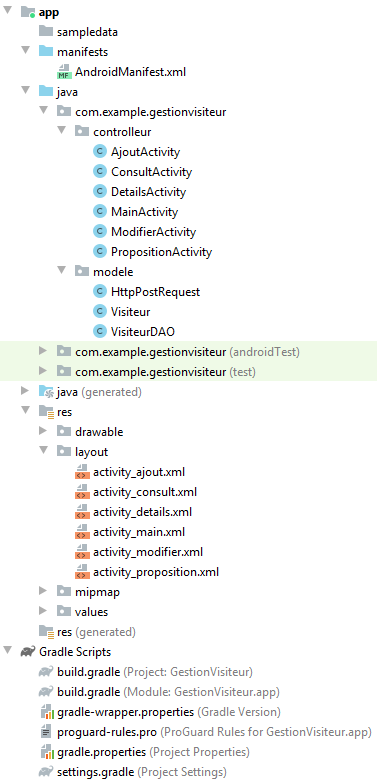
L’administrateur après s’être connecté devra pouvoir gérer :

⦁ 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.



# 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.

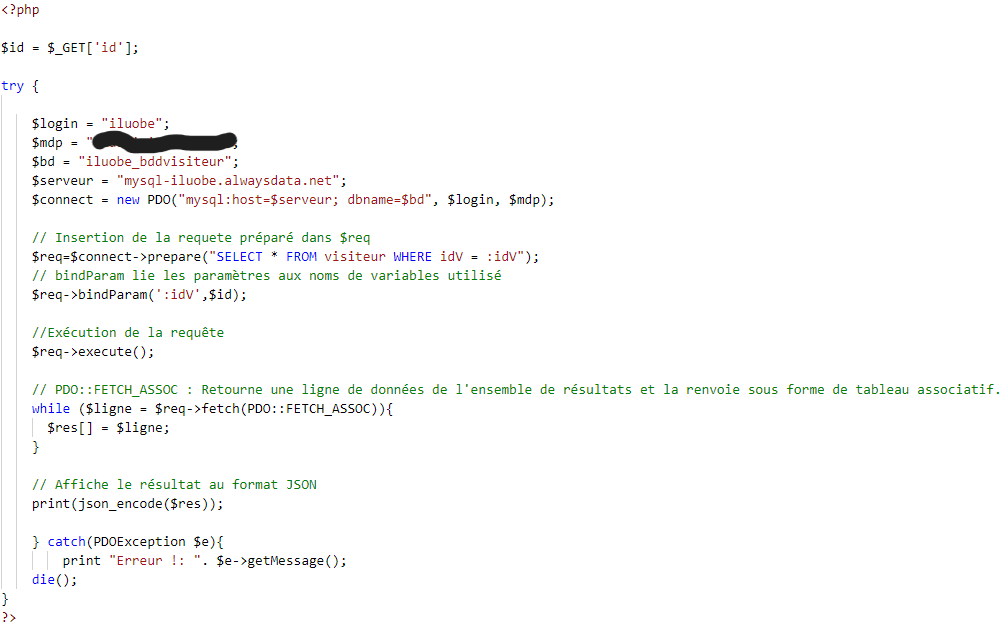
### Fichier connexion.php



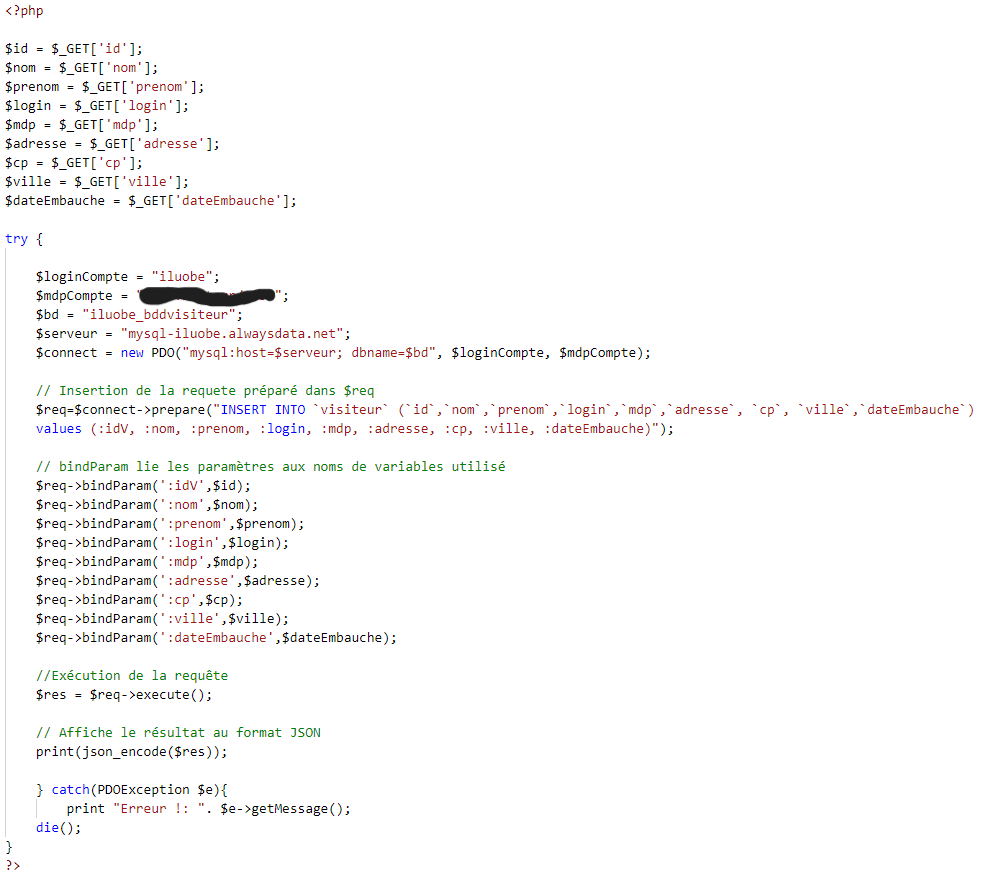
### Fichier getVisiteurs.php



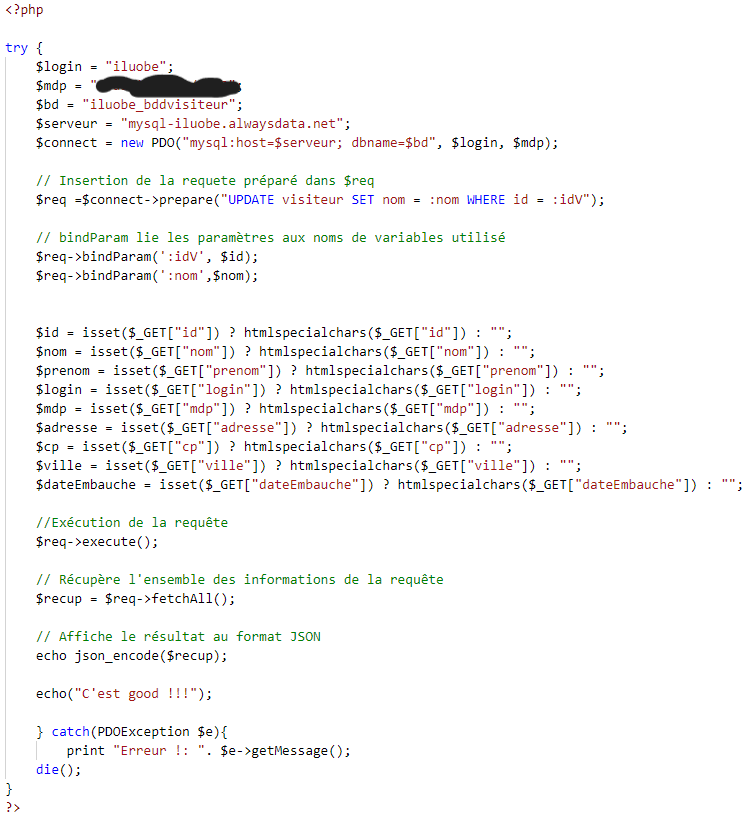
### Fichier getVisiteurById.php



### Fichier addVisiteur.php



### Fichier modifierVisiteurById.php



### Fonction supprimerVisiteurById.php

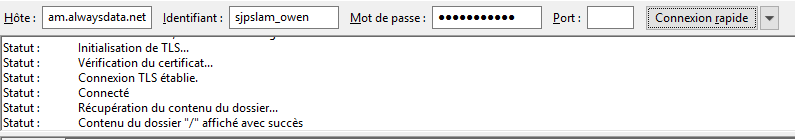


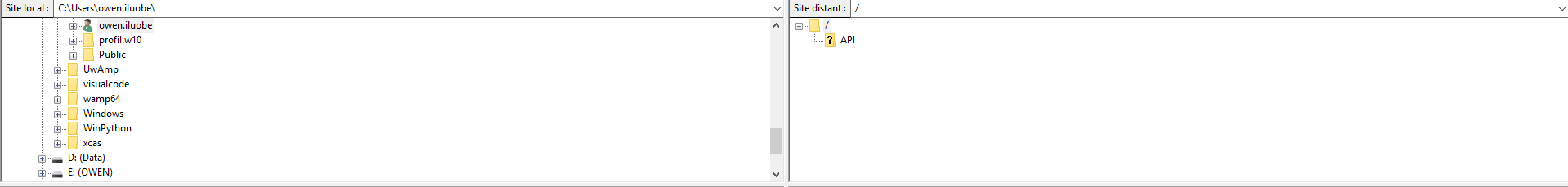
## Test des fichiers PHP via FileZilla

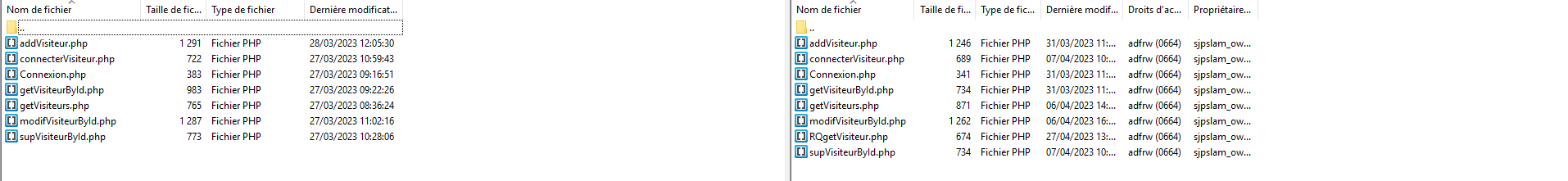
Pour pouvoir tester les fichiers PHP, je devais me connecter au serveur de ma base de données et tranférer les fichiers un par un. A chaque modification apportée, je devais remettre le nouveau fichier mis à jour. Ensuite, sur un navigateur web, je devais rentrer l’url suivante :

[**https://sjpslam.alwaysdata.net/owen/API/nomDuFichierVoulu.php**](https://sjpslam.alwaysdata.net/owen/API/nomDuFichierVoulu.php)

Cela a été possible car j’utilsais la méthode GET sur tous mes fichiers PHP.



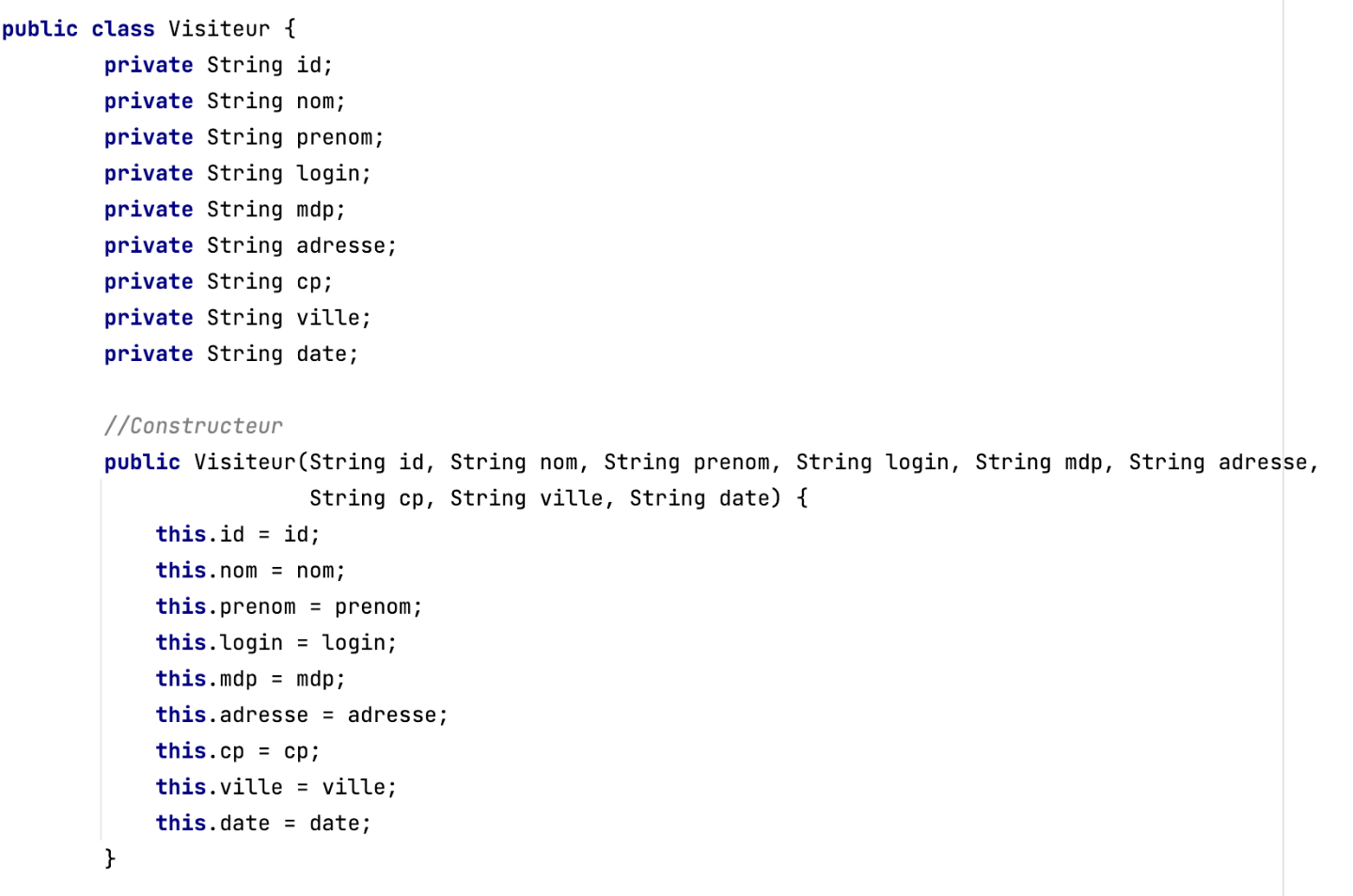


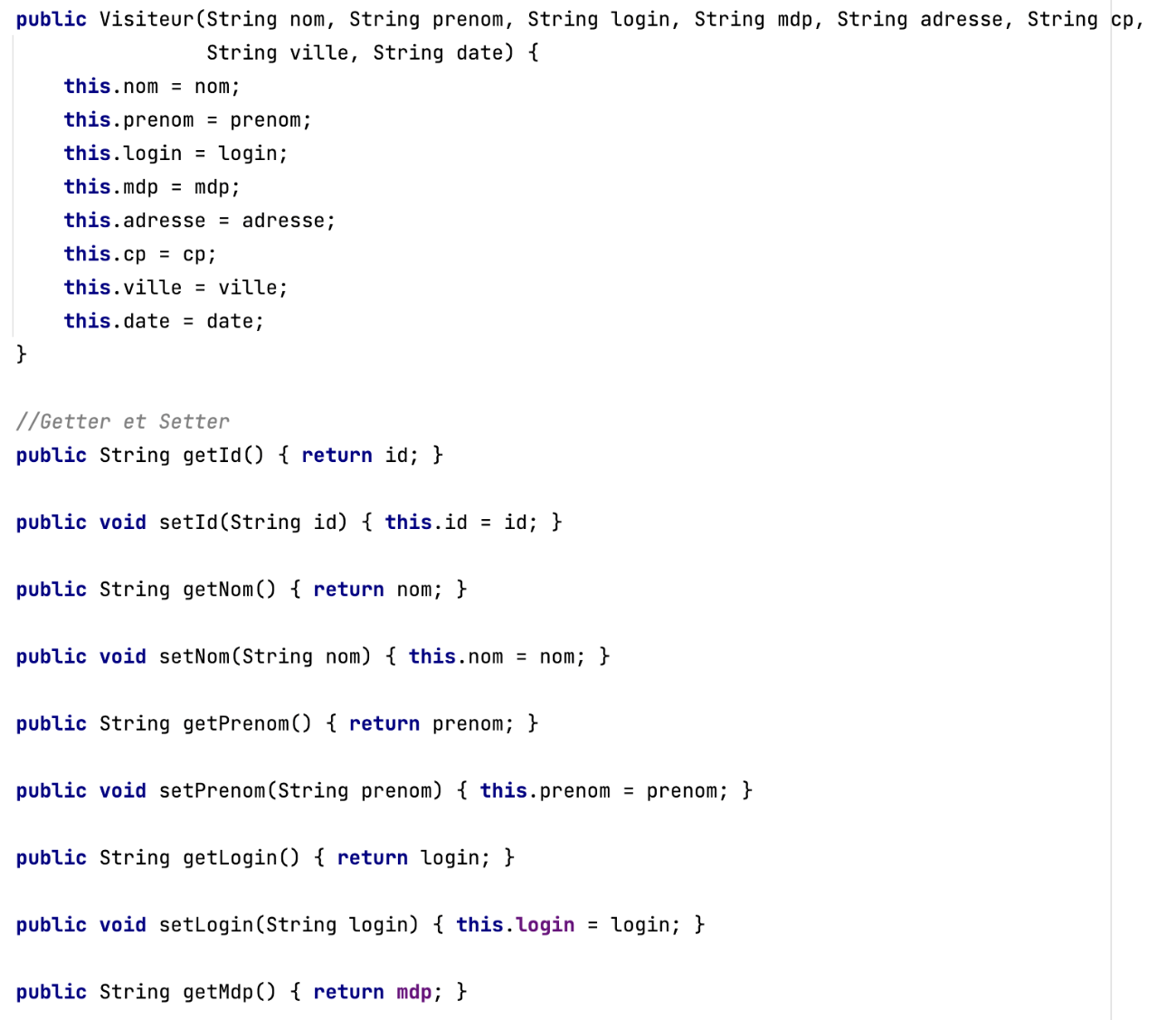


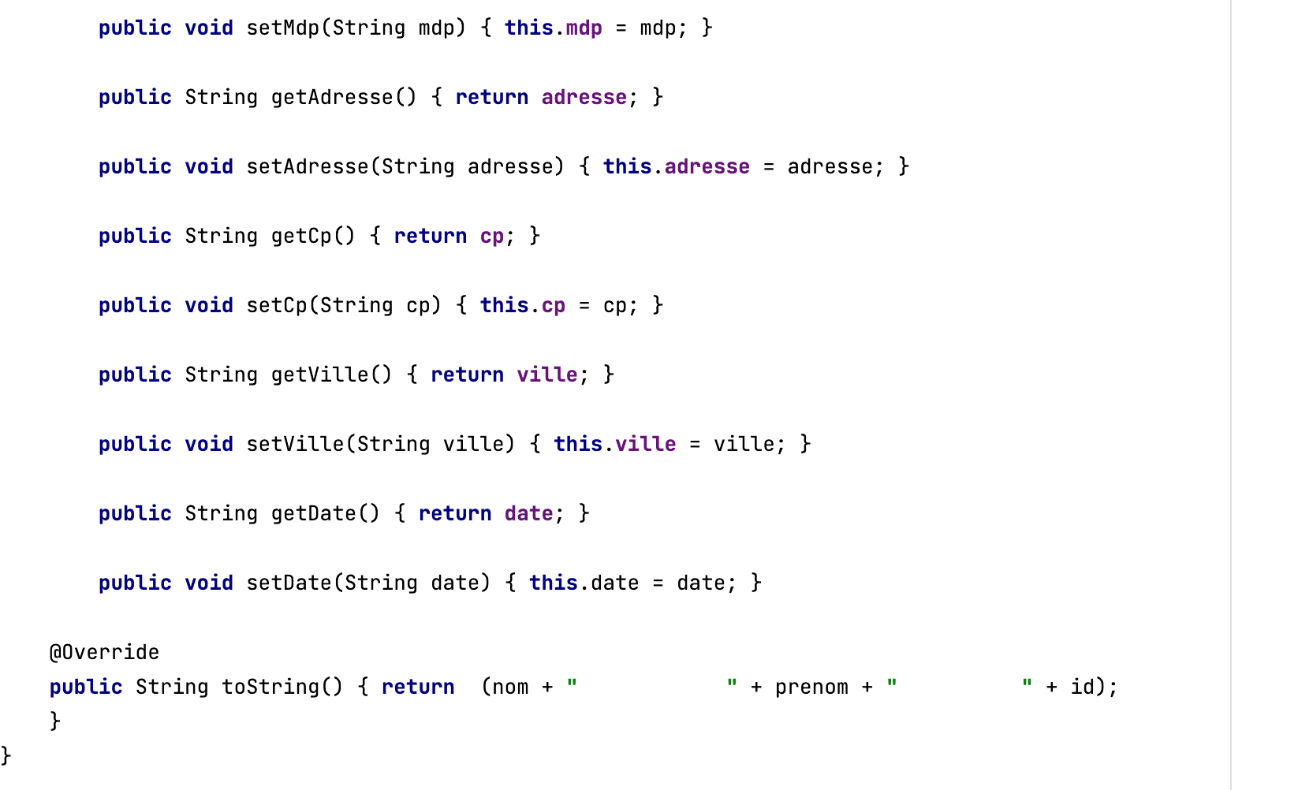


## La classe métier : Visiteur

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



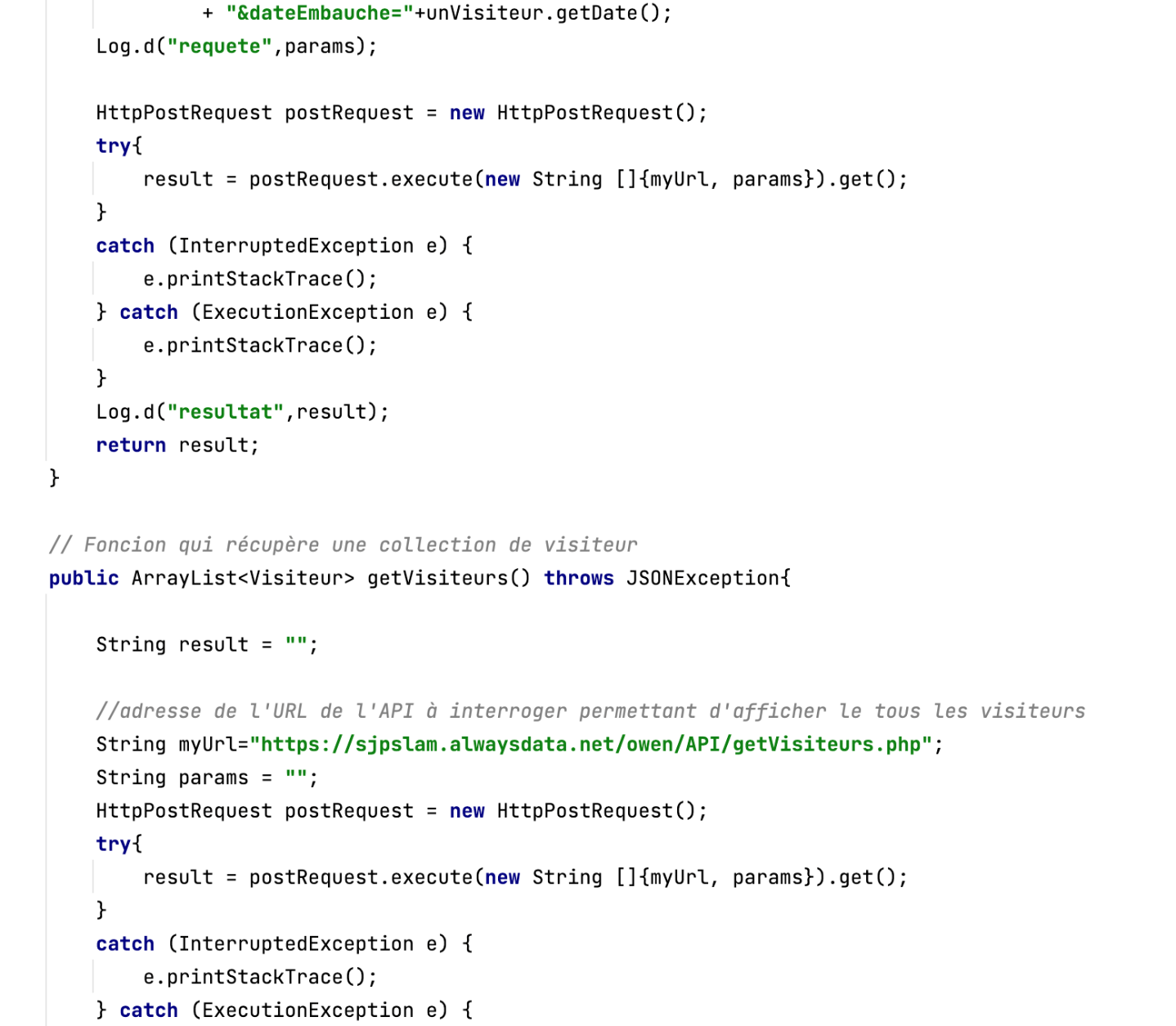


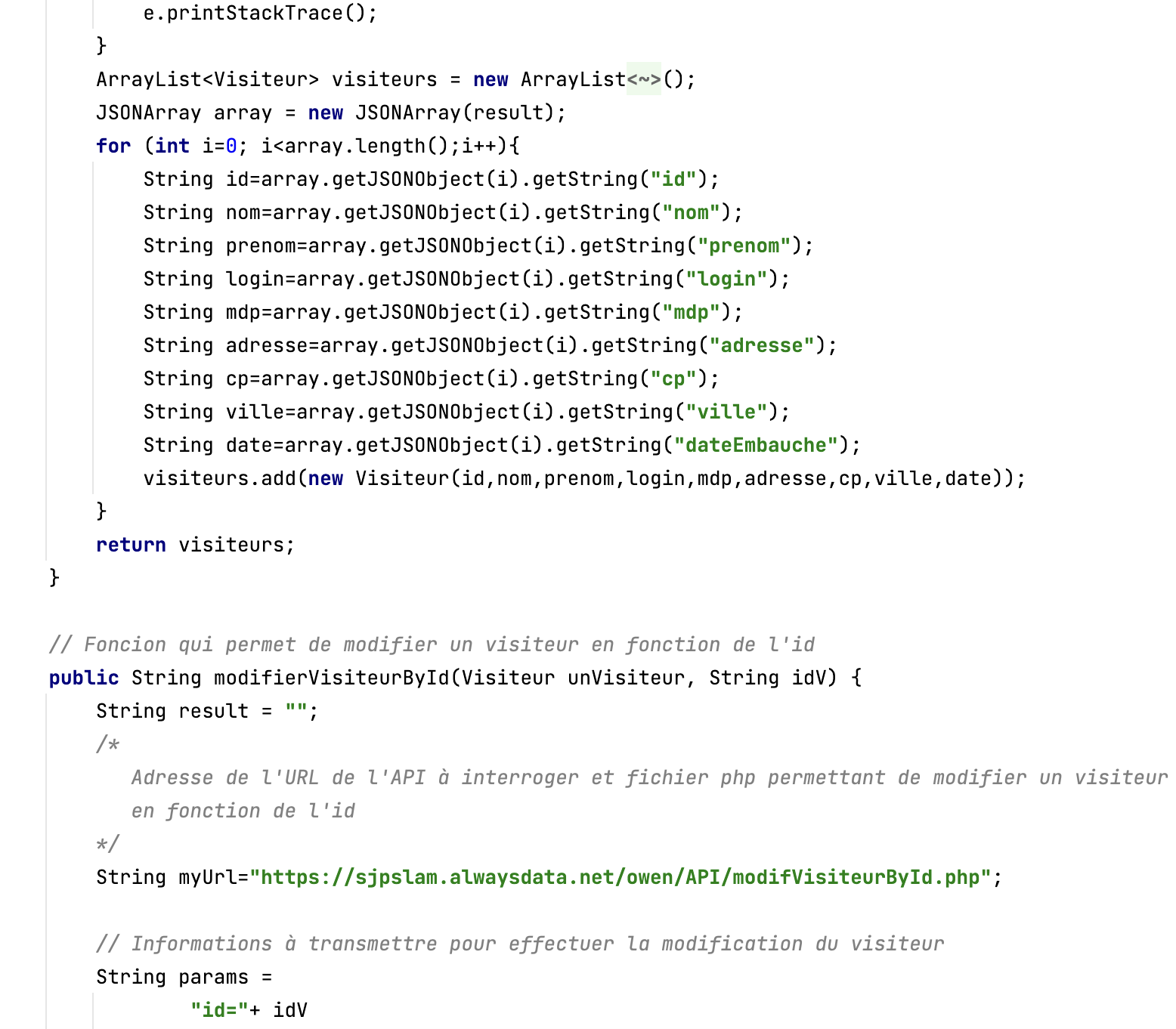


## La classe DAO : VisiteurDAO

Ensuite, 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.











## 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 :



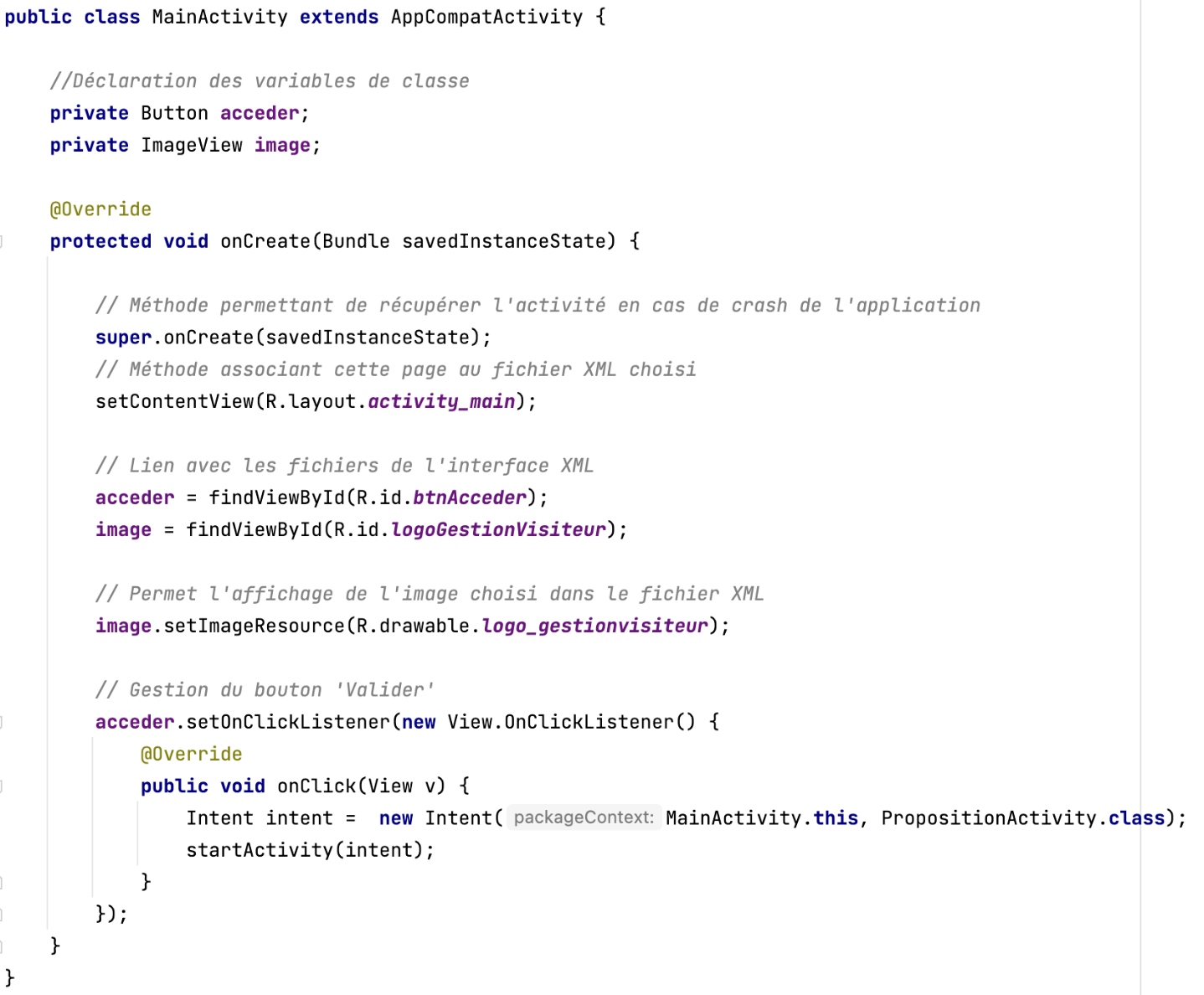


## Les contrôleurs

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

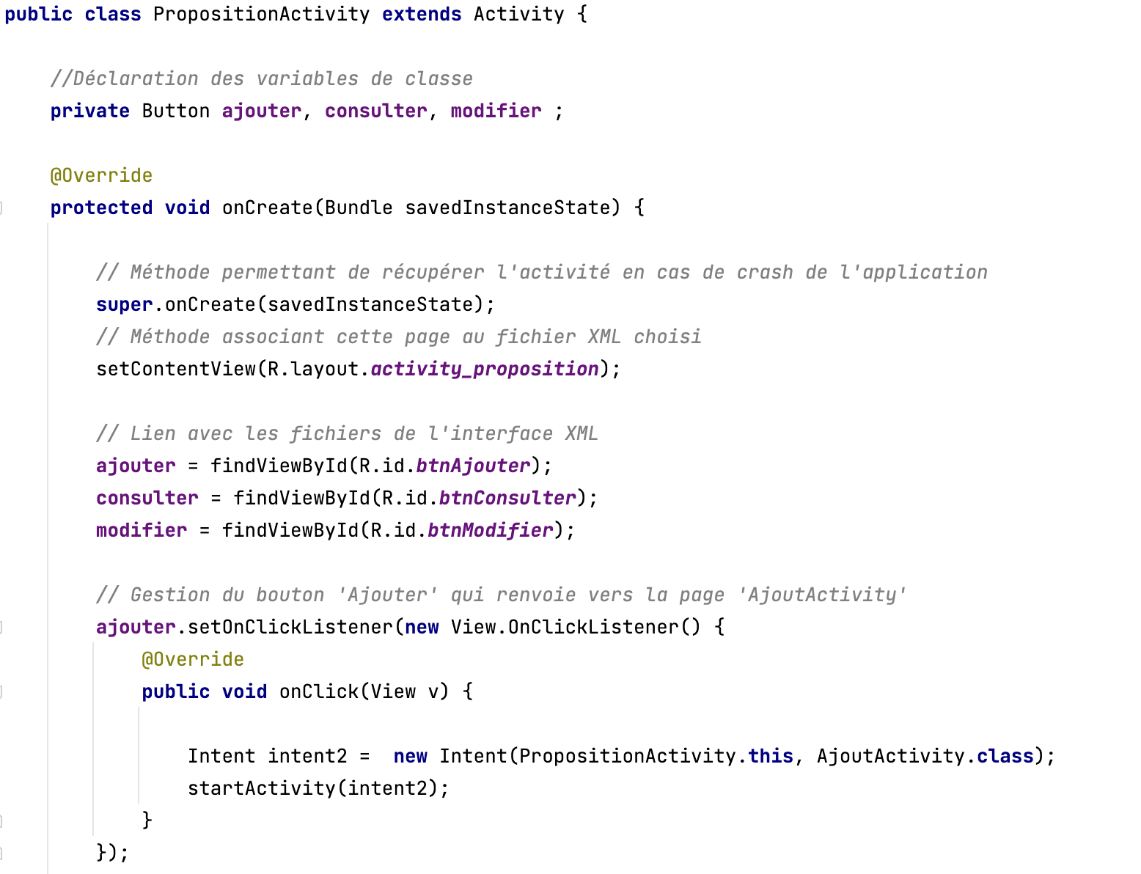
### MainActivity

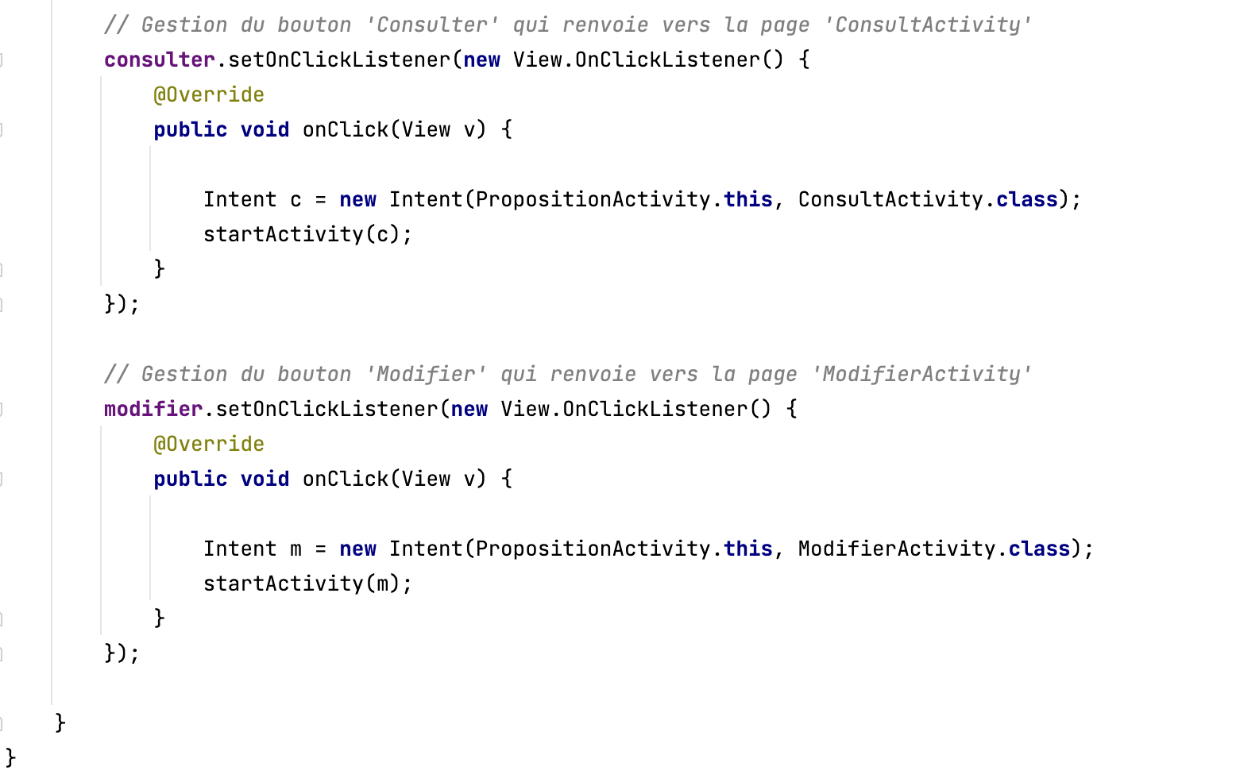
Ce contrôleur est la page d’accueil de mon application.



### PropositionActivity

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

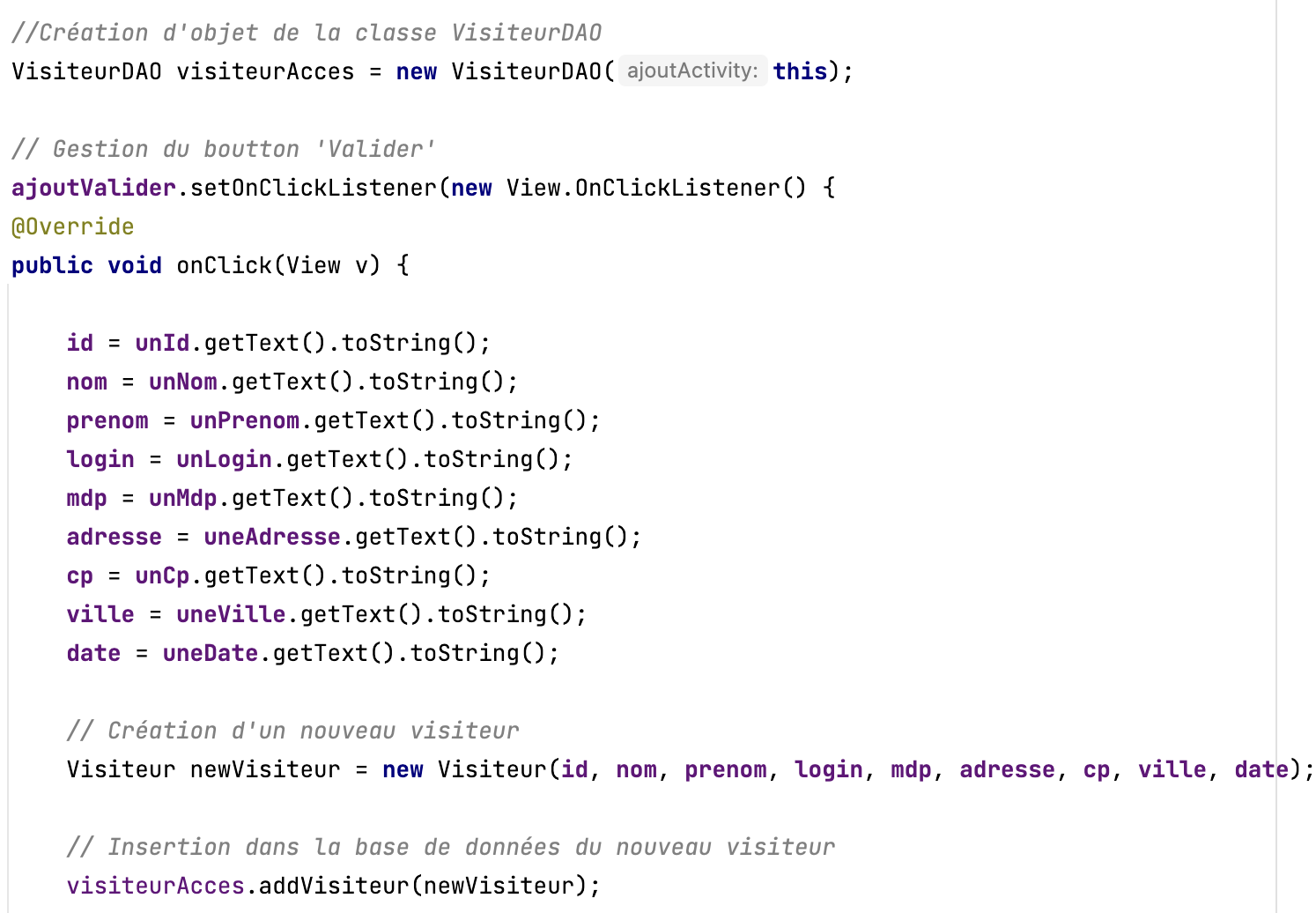


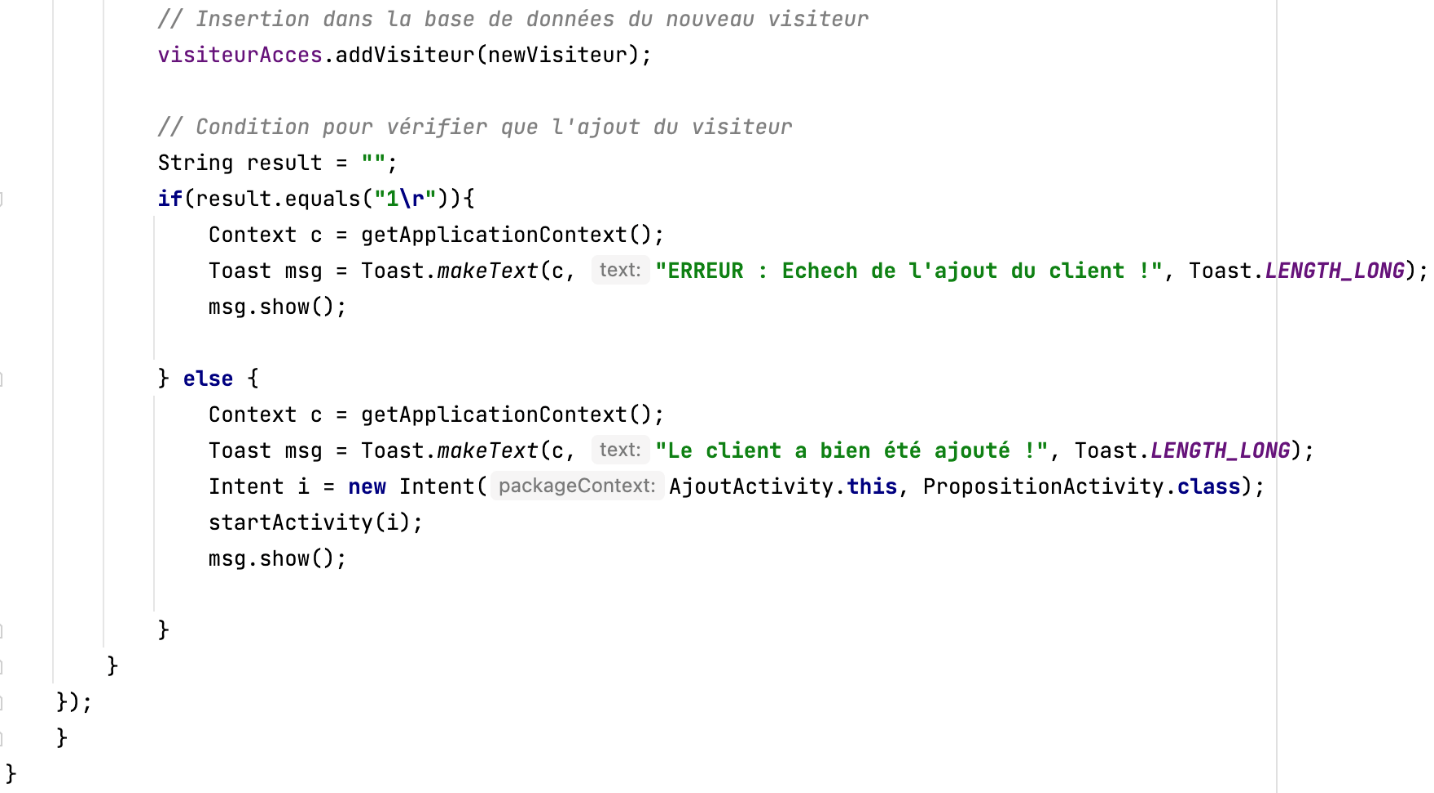


### AjoutActivity

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

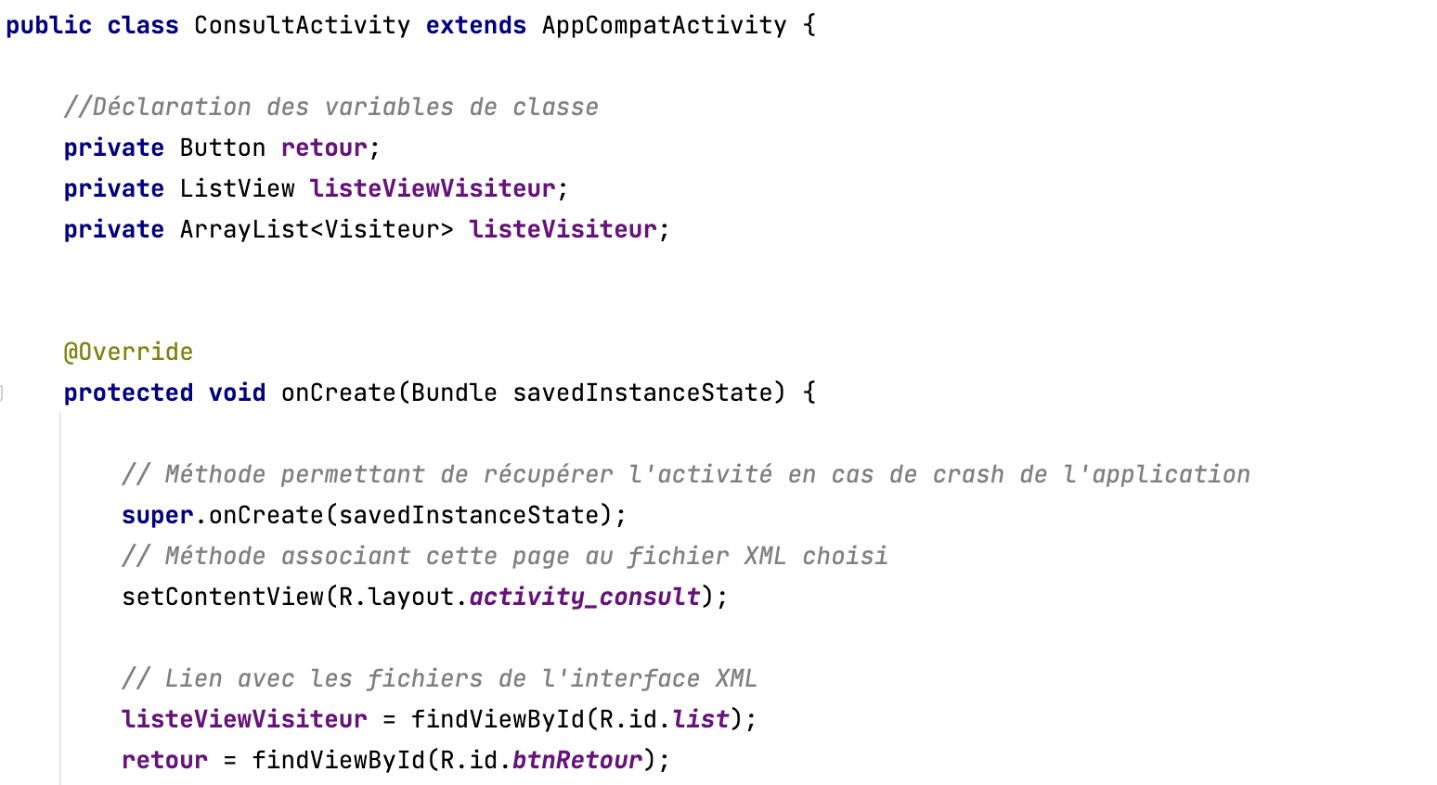


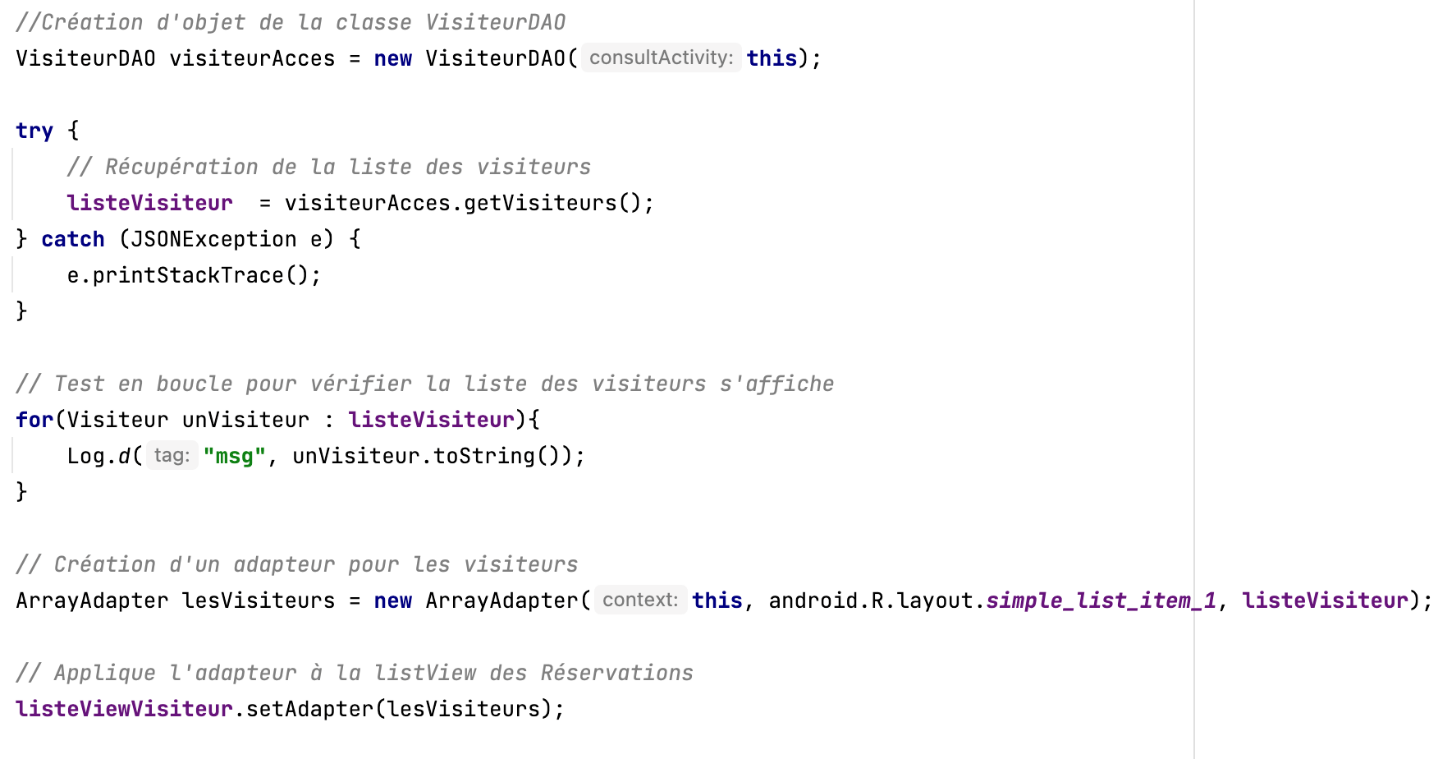


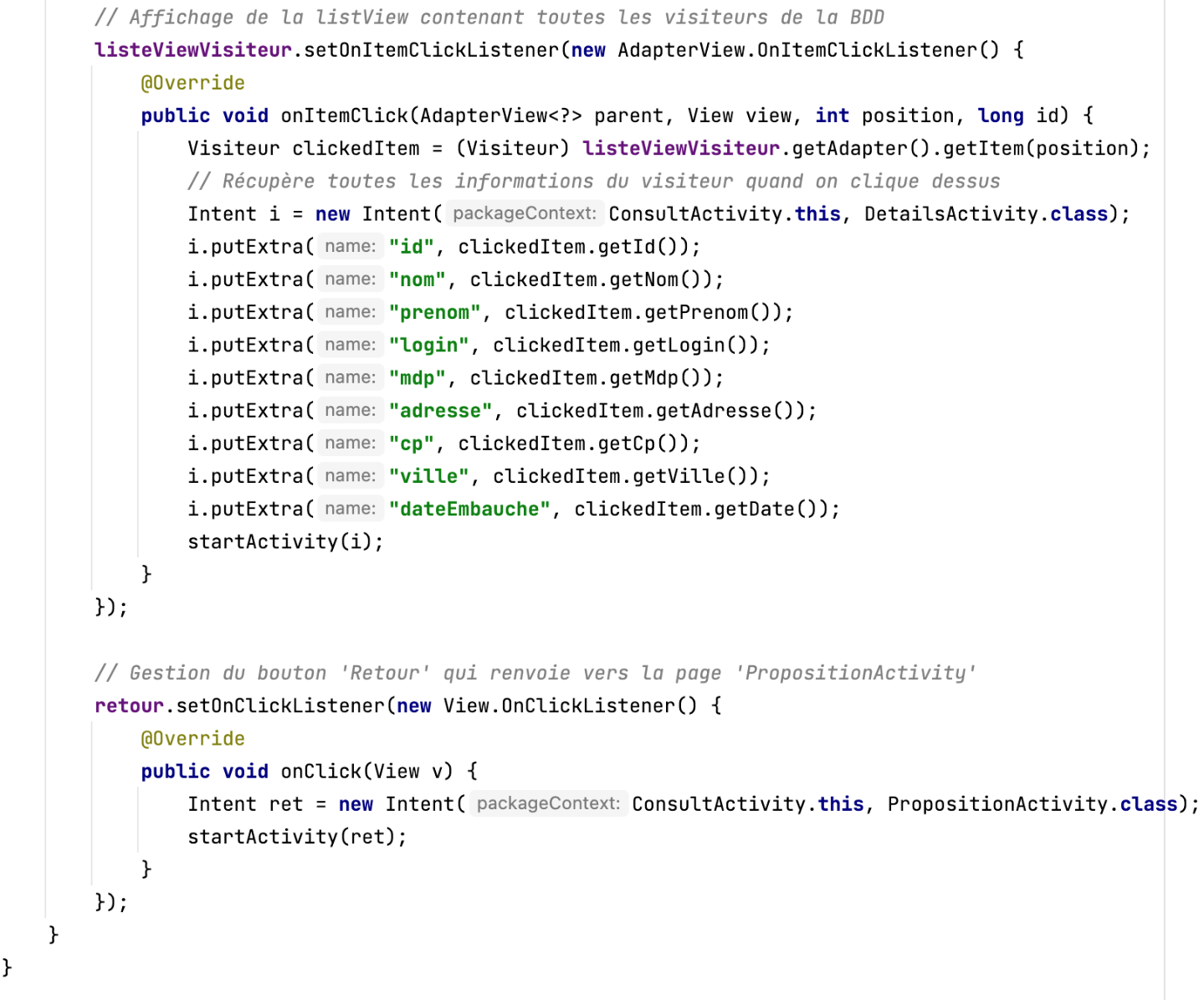


### ConsultActivity

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





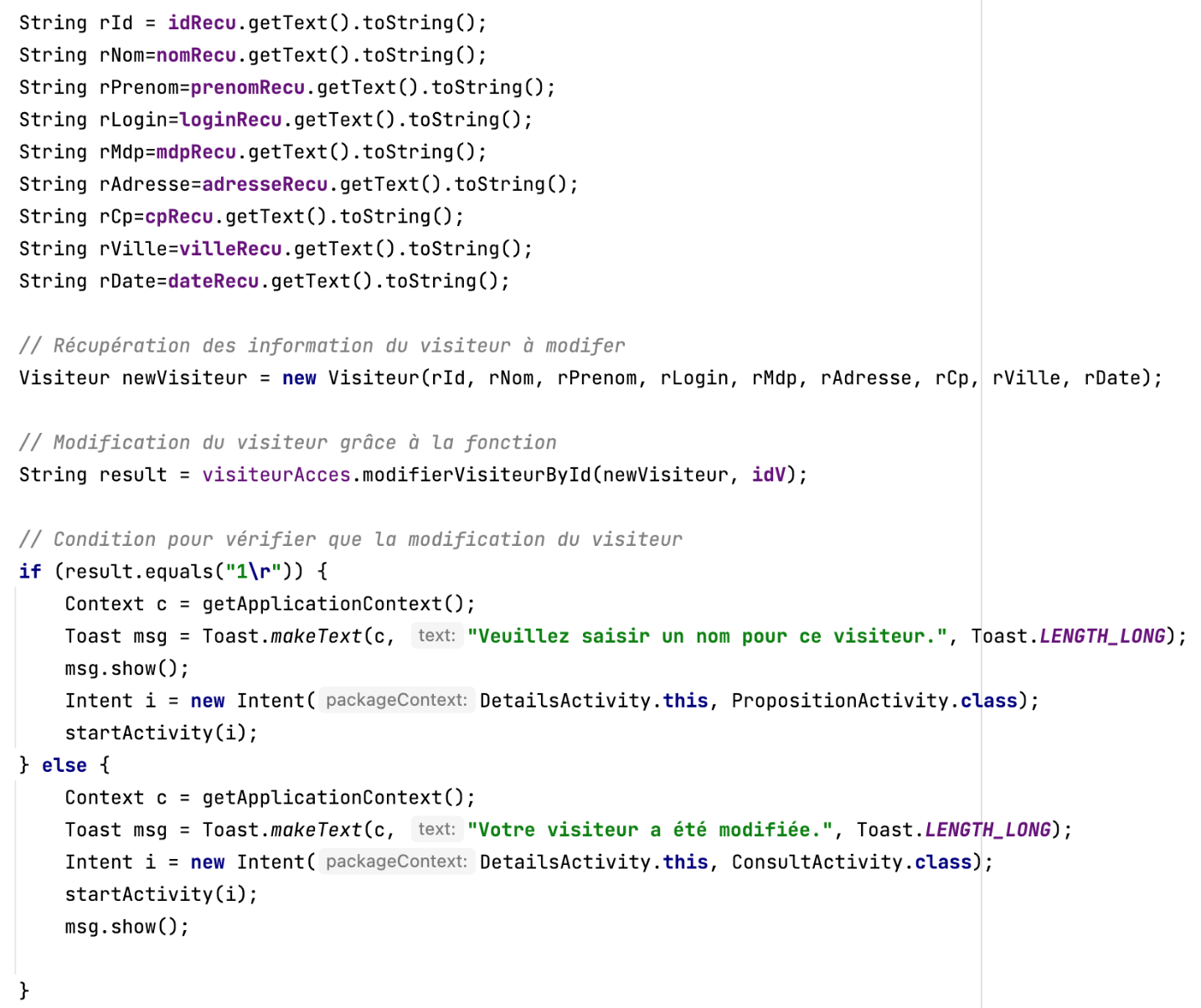


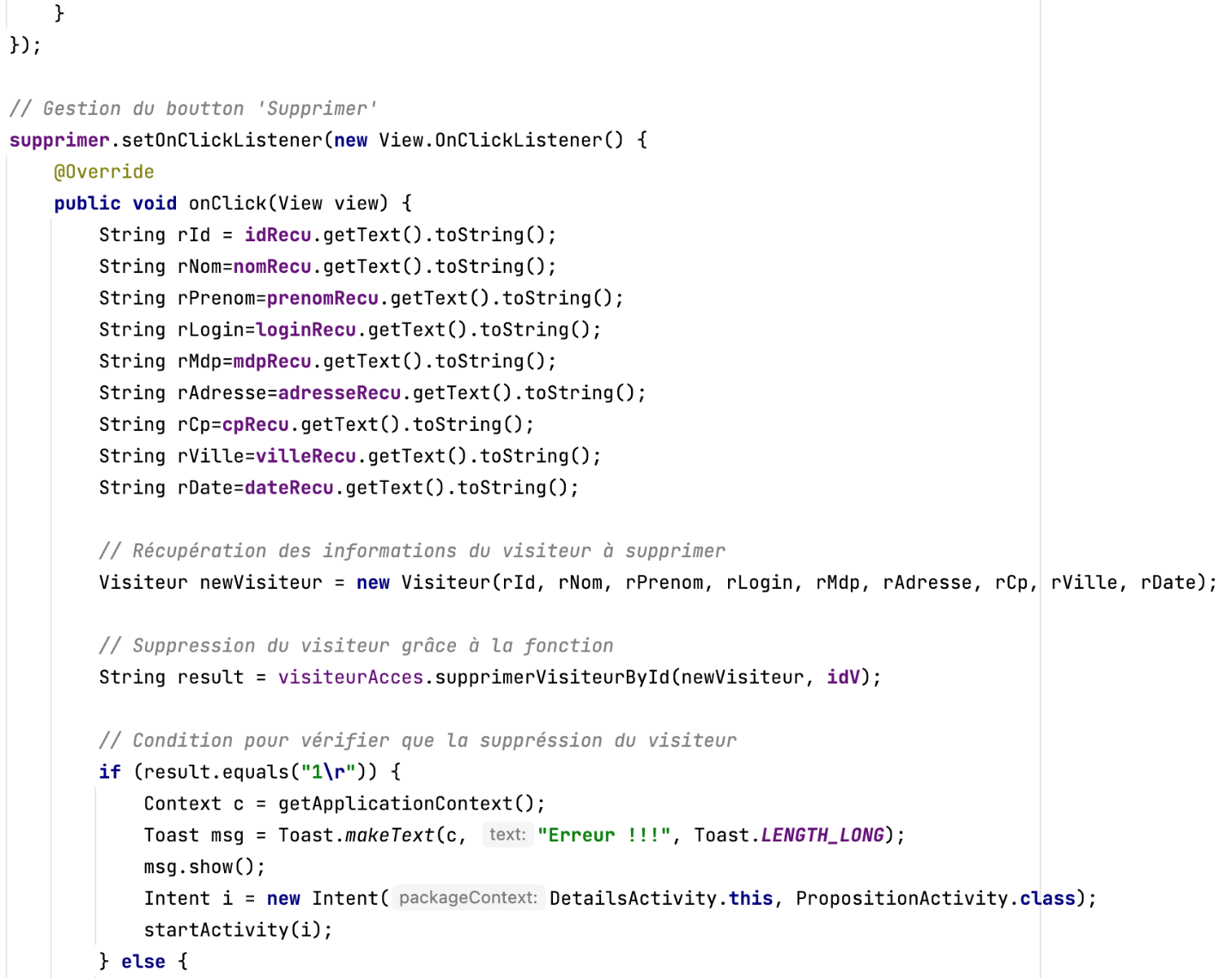
### DetailsActivity

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





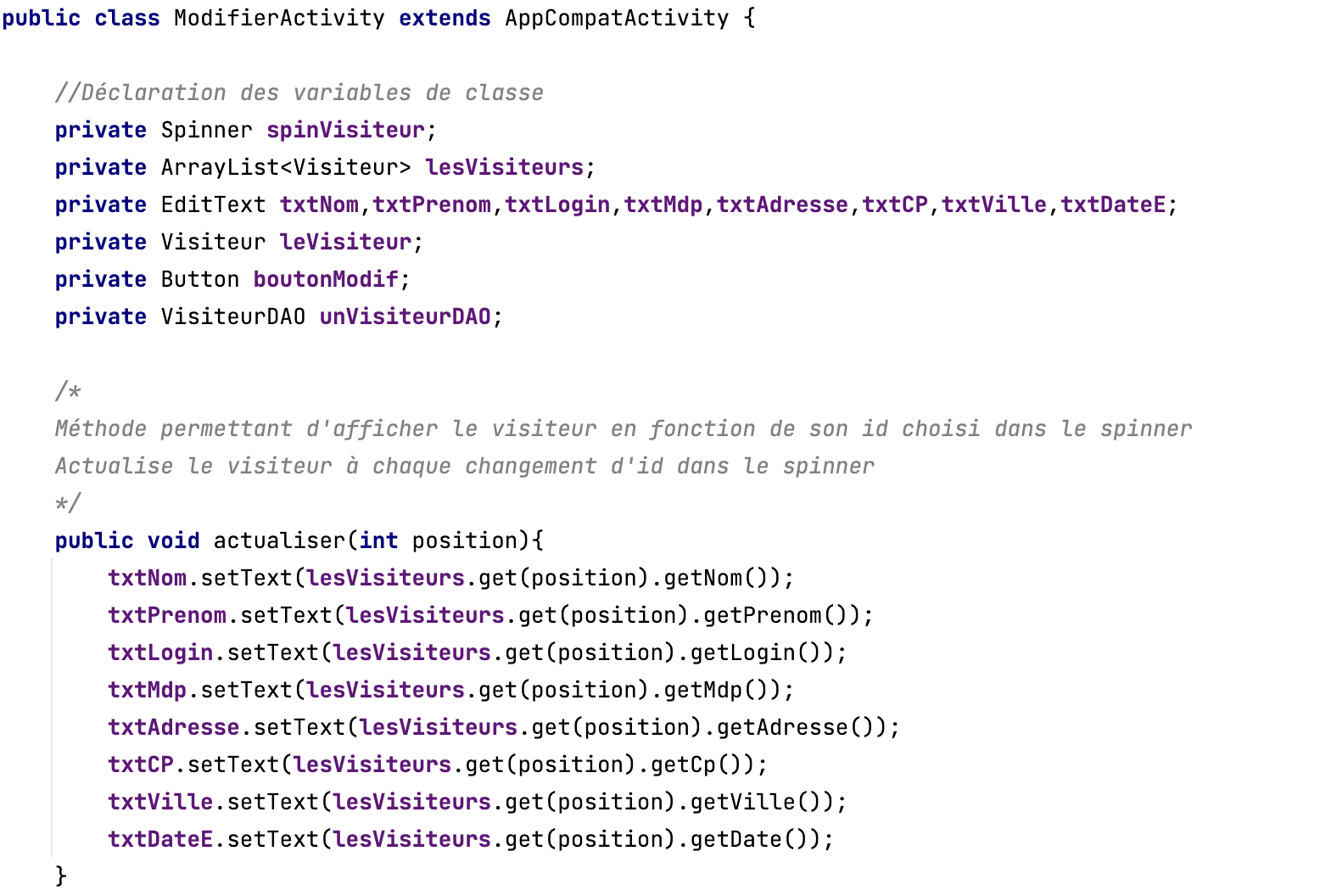


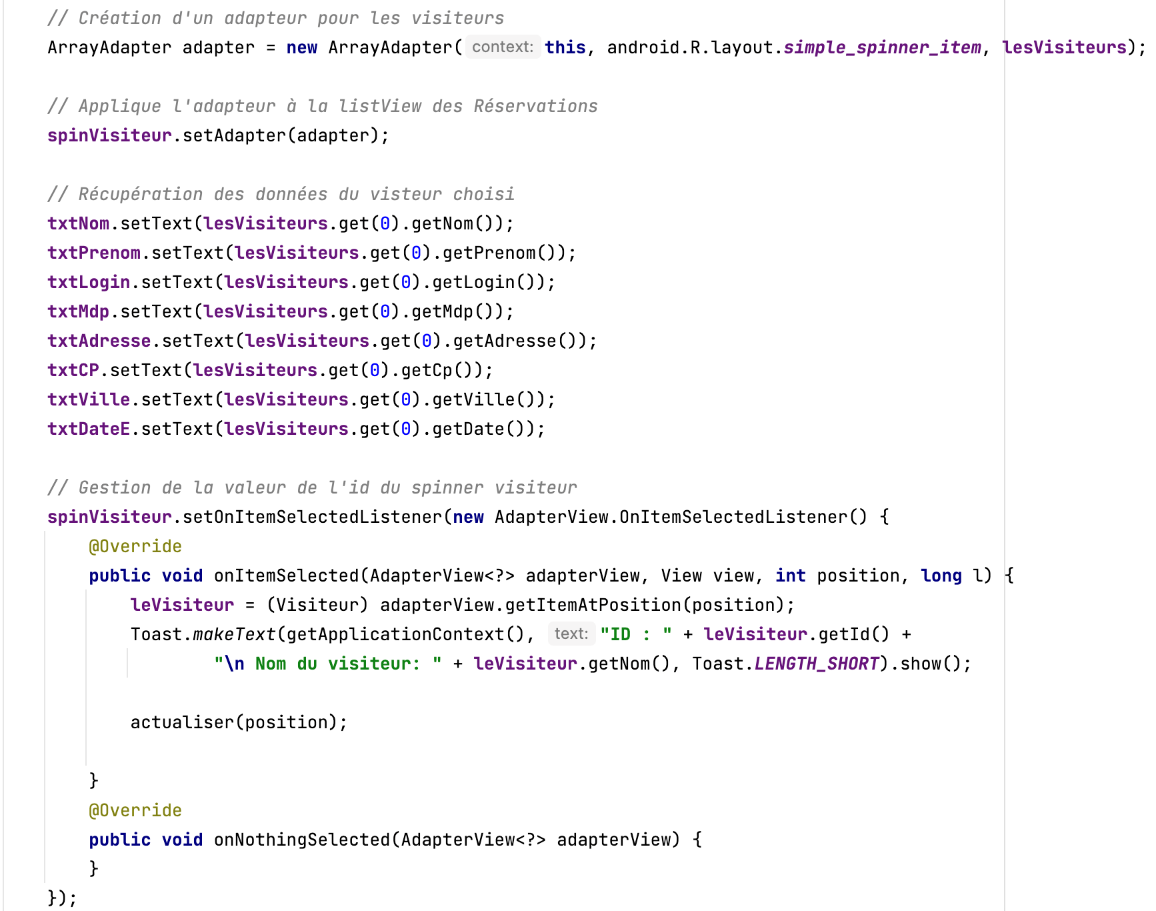
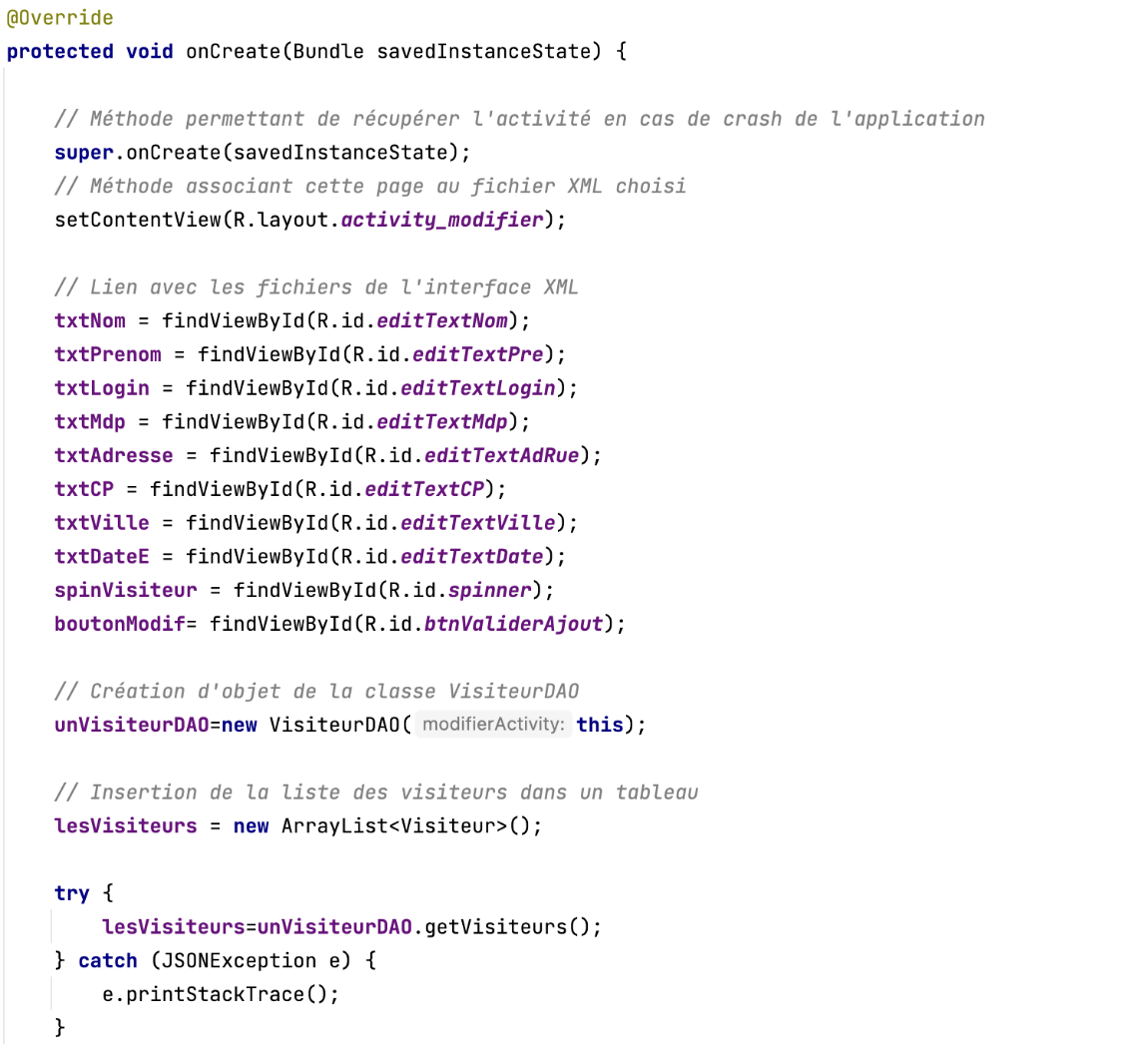




### ModifierActivity

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.





## 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.



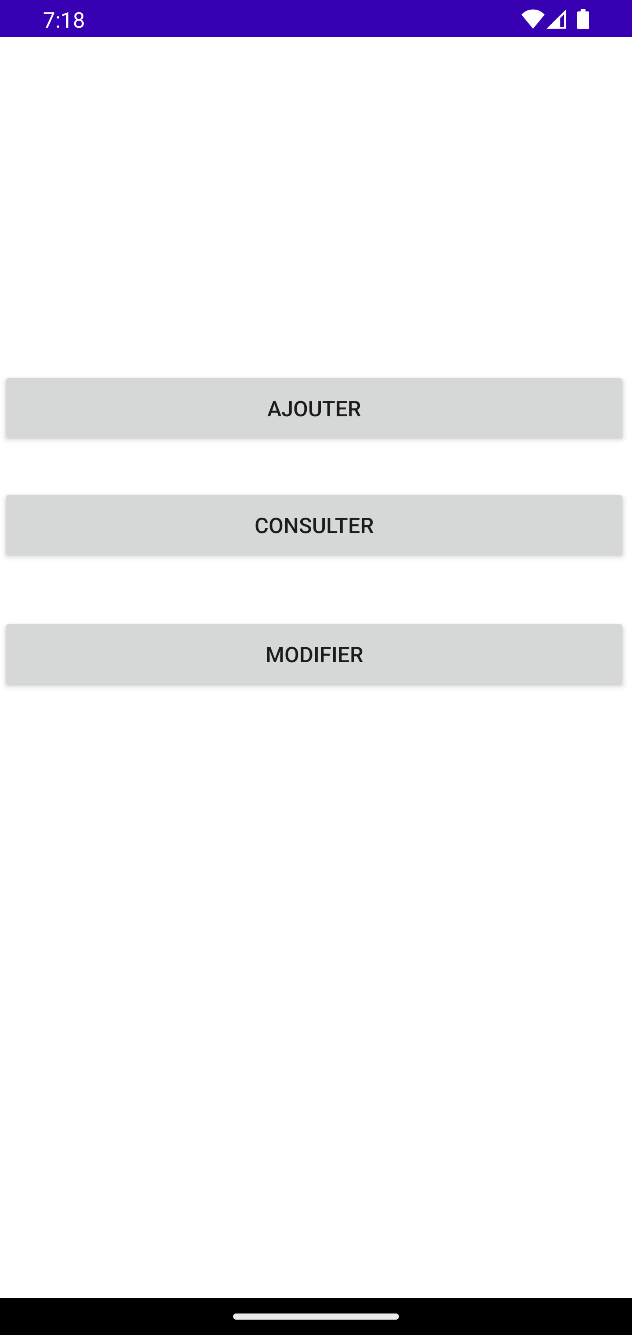
# Mes interfaces

Voici le résultat final de mes interfaces réalisés.

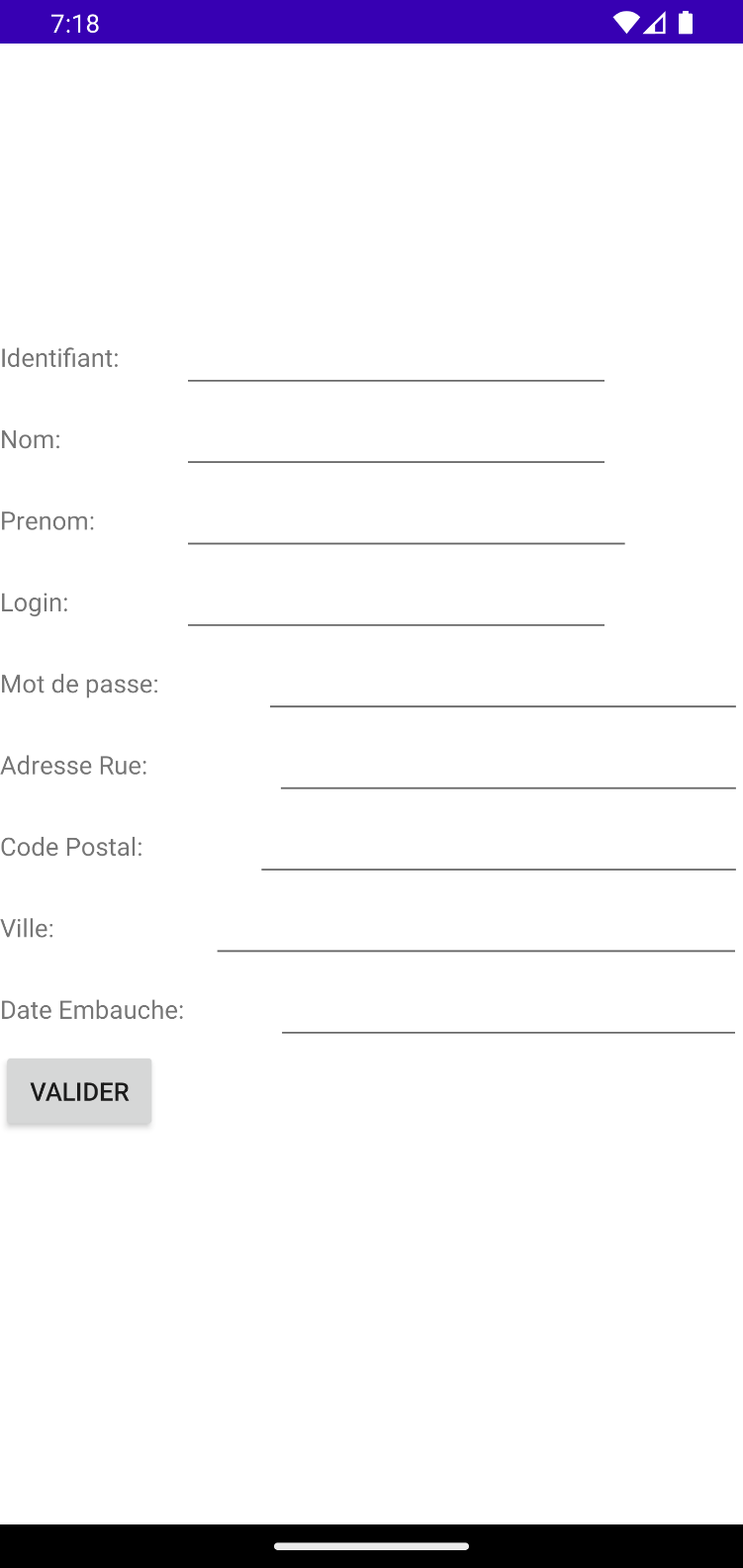
L’interface de la page d’accueil :



L’interface de ma page de proposition :



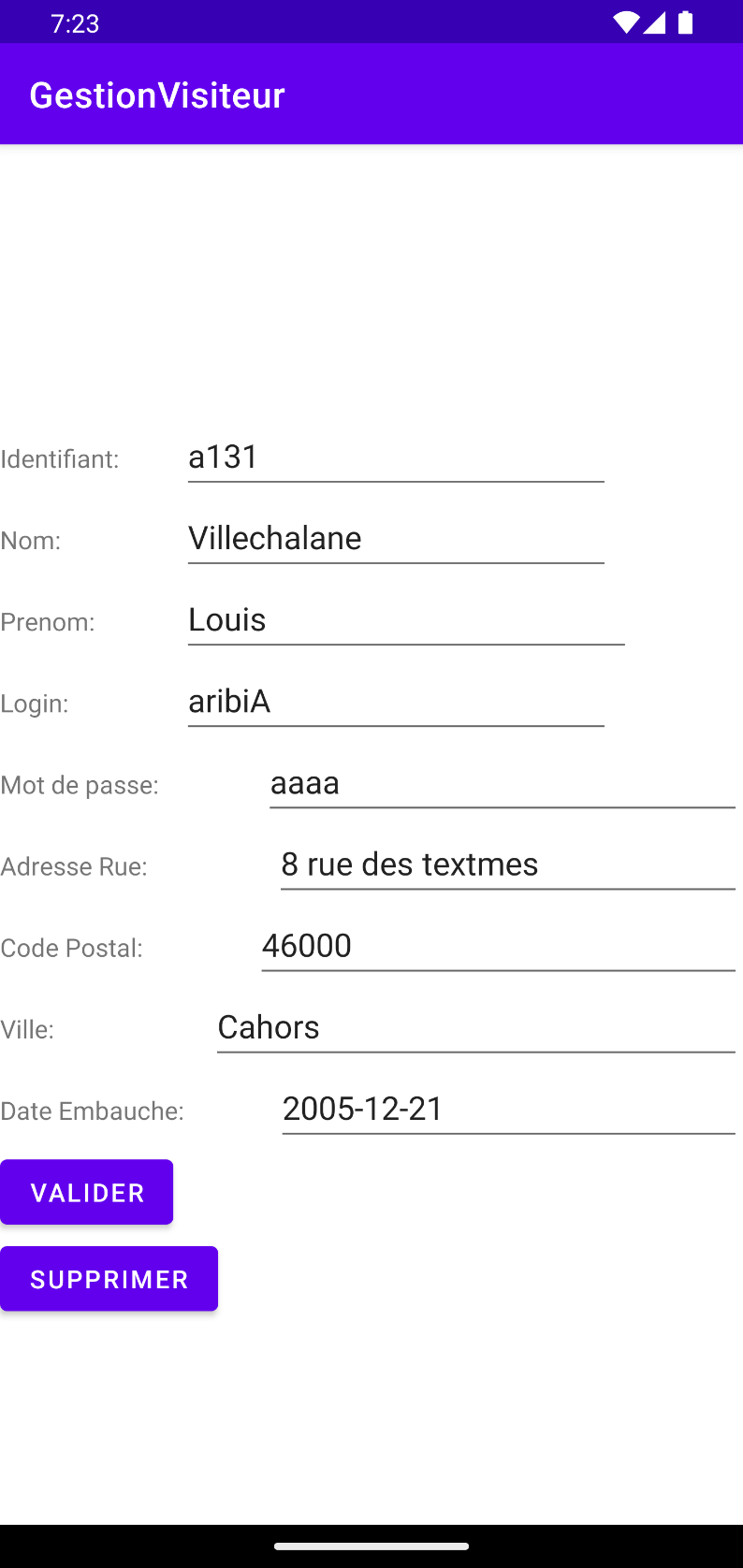
L’interface pour l’ajout d’un visiteur :



L’interface pour la consultation des visiteurs :



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



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

