NELSON MANDELA
Lycée Polyvalent

GestSio

Documentation Technique

Hassani Saïd Chazal

MANUEL Gérald

HIDJAZOU Attoumani

Etablissement : NELSON MANDELA Polyvalent

NELSON MANDELA
Lycée Polyvalent

GestSio

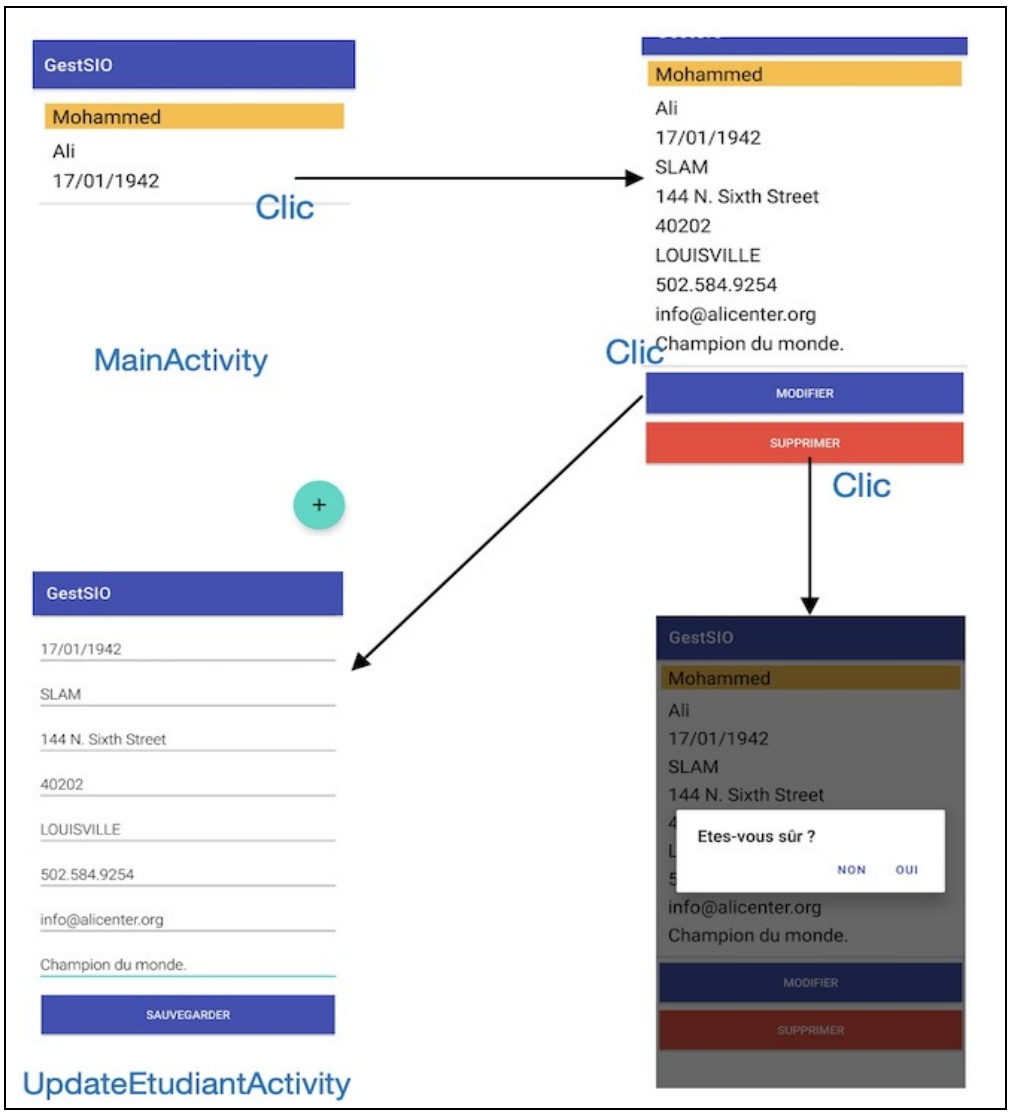
Sommaire

GestSio

1- Fournir par M.Sapin (MDL)

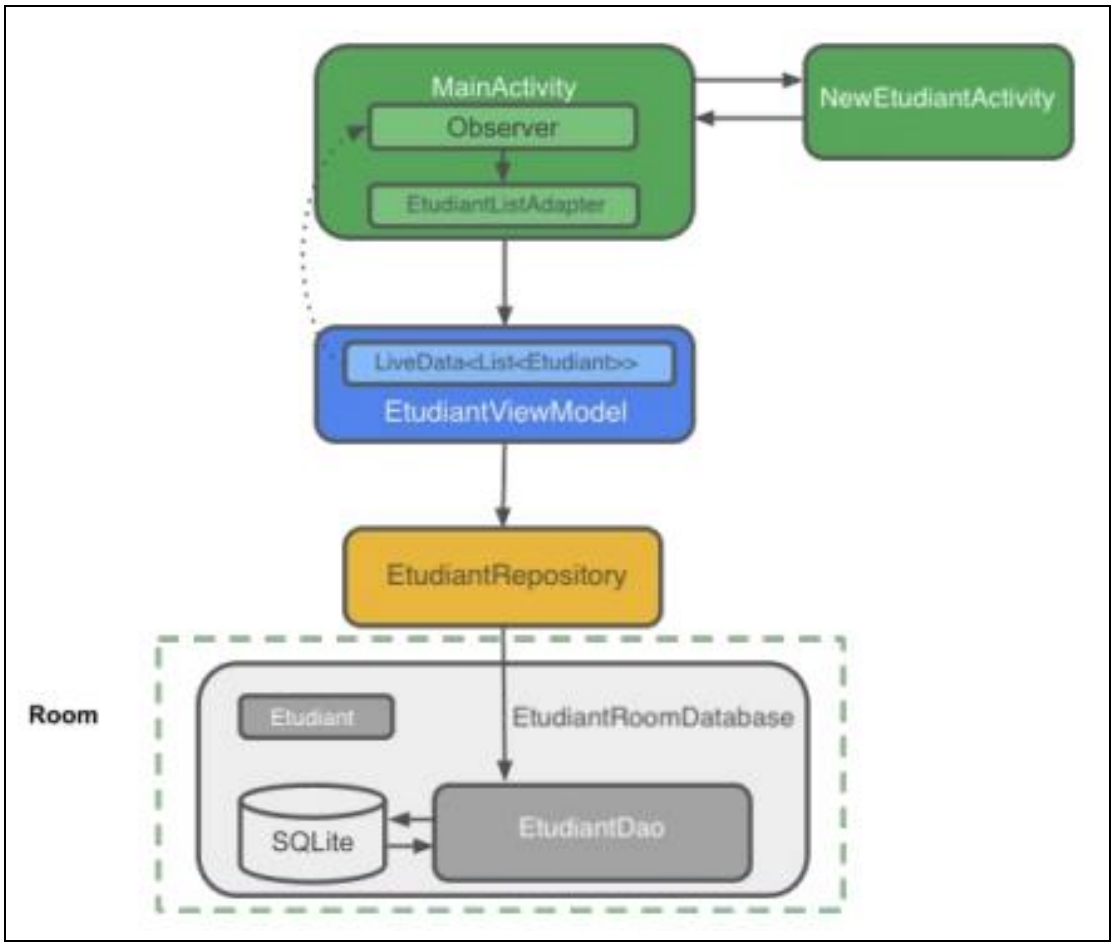
1.1- Présentation succincte de l'application actuelle :

L'application permet au responsable de consulter la liste des étudiants, de modifier un étudiant ou de le supprimer.



GestSio

Le fonctionnement général est basé sur la structure conseillée par Google, selon le schéma suivant :



L'ensemble des informations est stocké dans une base de données SQLite et qui est manipulée grâce à un ORM : ROOM.

Le diagramme de classes est disponible dans le répertoire ressources (sur le NAS).

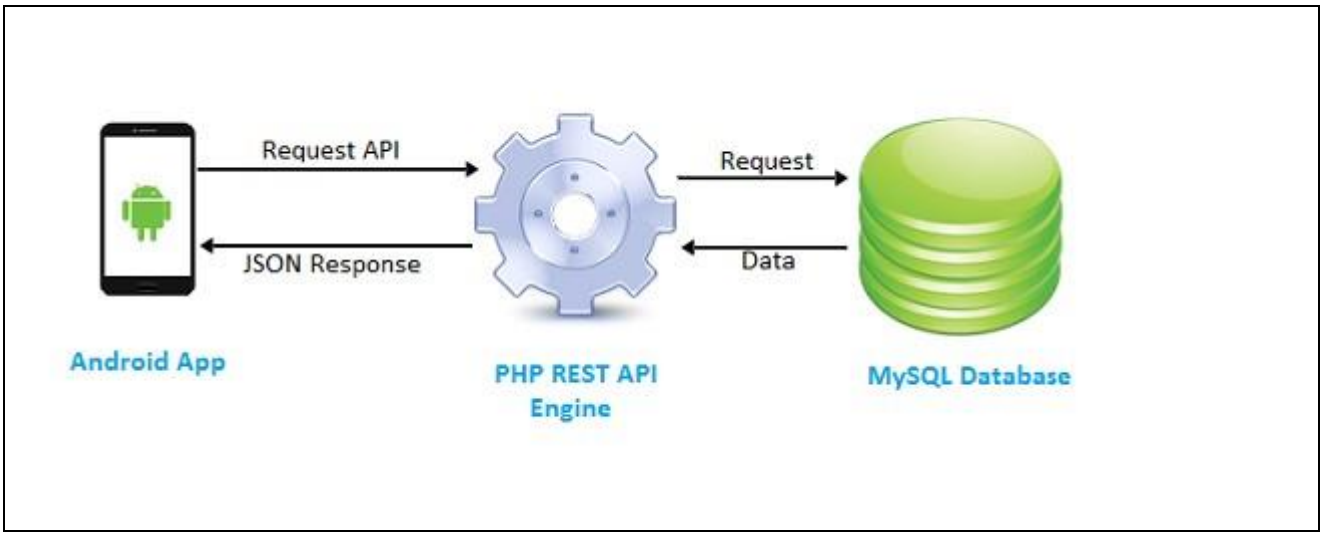
1.2- Evolution envisagée :

M. Sapin souhaiterait que l'application, que l'application soit capable d'afficher, en lecture seule, l'appréciation du tuteur de stage en dessous des autres informations déjà présentes (dans le layout **activity_view_etudiant.xml**).

GestSio

Cette appréciation sera saisie par le tuteur grace à une interface écrite en PHP **que vous ne devez pas développer** et mise à jour à chaque lancement de l'application GestSio.

Pour pouvoir accéder à cette appréciation vous devrez communiquer avec le serveur distant en utilisant une API REST[1] en PHP qui renverra un fichier au format JSON.



Le stagiaire précédent a effectuer des recherches sur le domaine que M. Sapin vous propose de mettre en application. Il vous propose d'utiliser la librairie Volley2 , mais vous pouvez utiliser une autre solution

Quelques pistes :

https://github.com/probelalkhan/android-sqlite-mysql-sync-example

https://www.simplifiedcoding.net/retrieve-data-mysql-database-android/

https://zestedesavoir.com/tutoriels/pdf/1140/communication-entre-android-et-php-mysql.pdf

NELSON MANDELA
Lycée Polyvalent

GestSio

2- Présentation de besoin

Le but de ce projet est de récupérer l'appréciation des étudiants à distance sur une application mobile et chaque id (idEtudiant) il y a son appréciation

3- Description des travaux réalisés

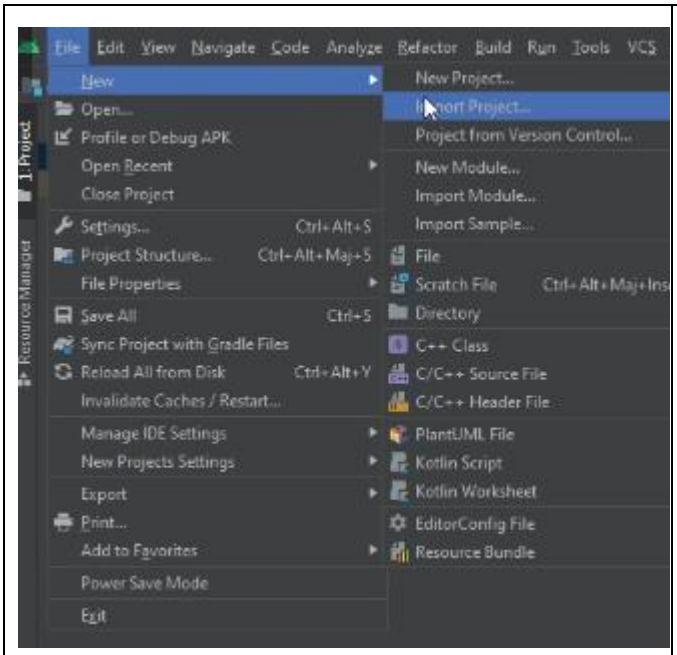
J'ai utilisé le langage php avec mes équipes pour récupérer les données des étudiants dans la base de données au format JSON, en Android, avec la bibliothèque Retrofit V2, sous le langage Java. Les étapes pour récupérer les IDS vers l'application mobile à distance sont :

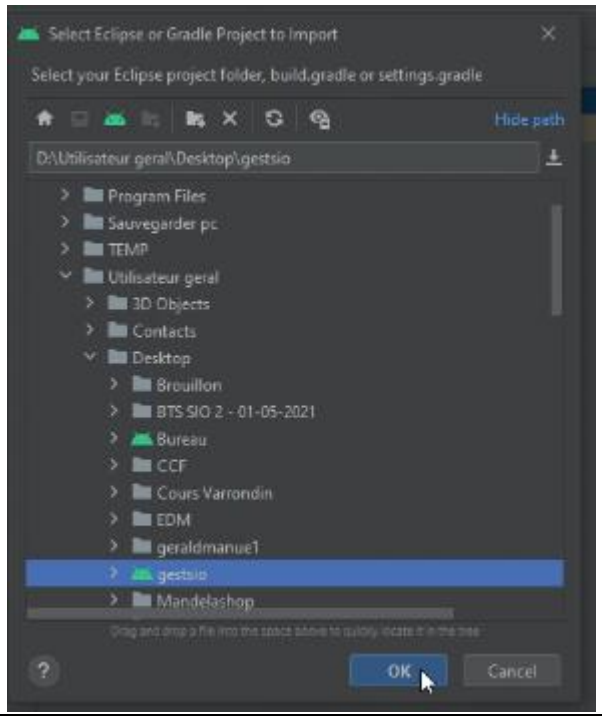
- Récupère le projet existant
- Importer dans Android Studio le projet que l'ancien stagiaire à effectuer
- Importer Retrofit dans le projet Android ▪
- Créer la classe Appreciation sur Java représentant les données (à récupérer du serveur) ▪
- Créer l'interface InterAppreciation sur Java représentant l'API du serveur ▪
- Créer l'instance du client Retrofit (en version 2) ▪
- Créer l'instance du service d'API ▪
- Créer la requête GET ▪
- Exécuter la requête GET ▪
- Récupérer le résultat de la requête

4- Récupérer le projet existant

Exécuter Android Studio en tant qu'administrateur

Ensuite, allez sur **File>New>import Project...** (importe une nouveau projet) et clic sur le projet et clic **Ok** pour finir.





5- Importer Retrofit V2 dans le projet Android Studio

Par la suite, il faut au préalable importer, dans le projet Android Studio, la dépendance **Retrofit V2**, ainsi qu'un convertisseur de requête **JSON** afin d'assurer la rétrocompatibilité dans **build.gradle** :

```
//retrofit
Implementation "com.squareup.retrofit2:retrofit:2.9.0"
```

```
// gson
Implementation "com.squareup.retrofit2:converter-gson:2.9.0"
```

Il s'agit aussi d'ajouter la permission internet dans **AndroidManifest.xml** :

```
<uses-permission android:name="android.permission.INTERNET" />
```

Dans la suite, nous nous intéressons à réaliser une requête HTTP GET et à récupérer une liste d'objets au format JSON depuis un serveur distant.

6- Créer la classe JAVA représentant les données

Il s'agit de créer la classe JAVA représentant l'objet à récupérer sur notre serveur.
Ici, nous allons créer une API REST simple à l'aide du langage PHP pour permettre que le format JSON récupère les données dans la base de données (BDD) MySQL et transfert sur application mobile à distance.

Le codage de langage php :

```
<?php
try {
$bdd = new PDO("mysql:host=localhost;dbname=table_etudiant;charset=utf8", "sio",
"sio", array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
}
catch (Exception $e) {
die('Erreur fatale : ' . $e->getMessage());
}
```

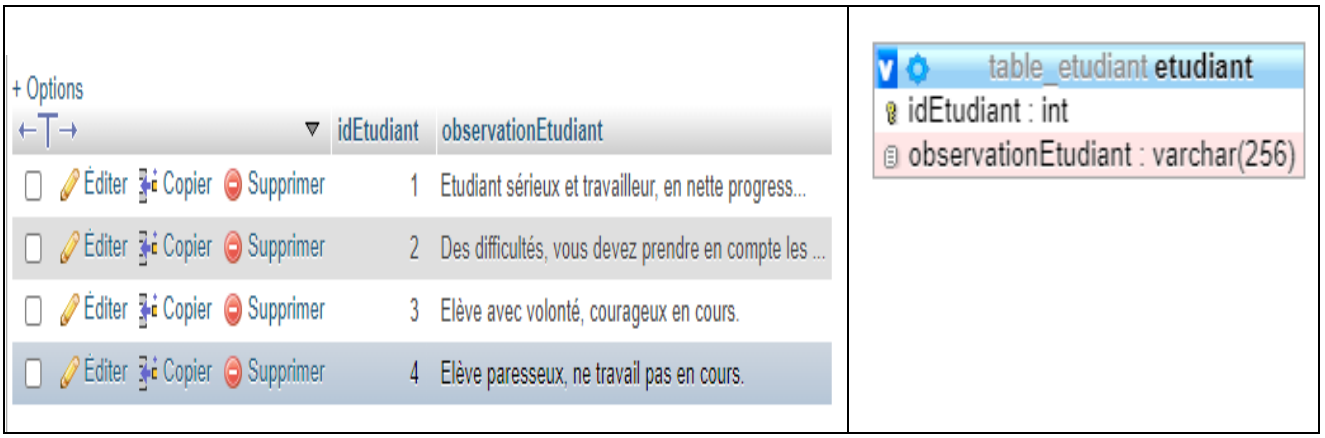
GestSio

```php
// Préparation de la requête : le ? correspond au paramètre attendu
$req = $bdd->prepare("SELECT `idEtudiant`, `observationEtudiant` FROM `etudiant` WHERE idEtudiant = ?");
// Exécution de la requête en lui passant le tableau des arguments
// (ici un seul élément : le code du idEtudiant)
$req->execute(array($_GET['idEtudiant']));
$idEtudiant = $req->fetchAll(PDO::FETCH_ASSOC);
echo json_encode($idEtudiant);
?>
```

Voici le code de BDD pour récupérer les IDS de chaque étudiant pour affiche dans l'application mobile :

```sql
CREATE TABLE `etudiant` (
`idEtudiant` int NOT NULL AUTO_INCREMENT,
`observationEtudiant` varchar(256) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
primary key (`idEtudiant`)
) ENGINE=MyISAM AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb3;
```

Voici la Modélisation BDD et sa table qui nomme étudiant :



Sur cette table en peut ajouter plusieurs appréciations pour chaque id (idEtudiant) lors de récupérations sur application mobile car on n'a pas la main sur ajouter dans application mobile pour mettre à jour les appréciations sur le BDD.

La requête permettant d'obtenir la liste de l'appréciation de chaque id d'étudiant est dans cette url : http://165.169.241.28:31195/MyApi/getEtudiant.php?idEtudiant=1 chaque id (idEtudiant), il y a son appréciation.

Par exemple, idEtudiant=1 on remplace idEtudiant=2 par la suite idEtudiant=3, etc...

idEtudiant=1

[{"idEtudiant":"1","observationEtudiant":"Etudiant s\u00e9rieux et travailleur, en nette progression. Poursuivez vos efforts."}]

idEtudiant=2

[{"idEtudiant":"2","observationEtudiant":"Des difficult\u00e9s, vous devez prendre en compte les conseils et mettre en place les strat\u00e9gies de m\u00e9thodologie."}]

idEtudiant=3

[{"idEtudiant":"2","observationEtudiant":"El\u00e8ve avec volont\u00e9, courageux en cours."}]

Ainsi le serveur, renvoie une liste de cours, contenant chacun :

- Un identifiant (idEtudiant) pour récupérer chaque id, ici id = idEtudiant
- Une observation (observationEtudiant) pour observer chaque appréciation de chaque étudiant

Afin de modéliser un tel objet, de chaque appréciation étudiant, soit Appreciation, il faut créer la classe JAVA représentant l'objet. Par exemple, la classe Appréciation représente les appréciations qu'on récupère pour l'application mobile, selon les données du serveur :

```
public class Appreciation {
   private int idEtudiant;
   private String observationEtudiant ;

    //GETTERS
   public int getIdEtudiant() {
      return idEtudiant;
   }
   public String getObservationEtudiant() {
      return observationEtudiant;
  }
}
```

Les attributs du constructeur doivent porter les mêmes noms que ceux du fichier JSON renvoyé par le serveur : idEtudiant et observationEtudiant.

7- Créer l'interface JAVA représentant l'API

Il s'agit de créer l'interface JAVA qui va contenir la déclaration de toutes les requêtes disponibles sur notre serveur. Pour le moment, nous avons une seule requête GET, ne prenant aucun paramètre et renvoyant une liste qu'on a créée. Par exemple, l'interface InterAppreciation représente l'API de notre serveur :

```
public interface InterAppreciation {

  @GET("getEtudiant.php?")
  Call<List<Appreciation>> getAppreciation(@Query("idEtudiant") int idEtudiant);
}
```

Avec les imports suivants :

```
import java.util.List;
import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Query;
```

L'annotation @GET de Retrofit V2 indique la déclaration d'une requête **GET**. Ensuite, en paramètre, il est placé le chemin de la requête `getEtudiant.php?`.

Ici en récupère appréciation de chaque id (idEtudiant) de l'étudiant avec **getAppreciation** en exécutant des requêtes qui nomment **@QUERY** de valeur entière **(int).** Elle ne prend pas de paramètre puisque la requête n'en demande pas. Et elle renvoie un Call<...>, c'est la réponse du serveur, prenant entre chevron List<Appreciation>, c'est-à-dire la liste de chaque appréciation d'un étudiant renvoyée par le serveur au format JSON.

Toutes fonctions associées à une requête, retourne un objet Call<>, ensuite le paramètre entre chevron varie selon ce que renvoie le serveur (une liste ou un objet) et ce que l'on souhaite récupérer. Par ailleurs, si la requête requiert un paramètre tel qu'un nom d'utilisateur alors il est spécifié en paramètre de la fonction.

8- Créer l'instance du client Retrofit

D'abord, ajouter une nouvelle **TextView** dans le dossier layout qui nomme **activity_view_etudiant.xml** pour permettre affiche de l'appréciation.

La classe XML de **activity_view_etudiant.xml** :

```
<TextView
    android:id="@+id/textViewAppreciation"
```

```
        style="@style/text_view_style"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
```

Depuis, une Activity qui nomme **ViewEtudiantActivity.class** ou bien un Fragment, nous allons créer une instance de client Retrofit. Par exemple, dans un Fragment NetworkFragment :

```
//retrofilt builder
    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("http://165.169.241.28:31195/MyApi/")
            .addConverterFactory(GsonConverterFactory.create())
            .build();
```

N'oublie pas les imports concernés sont celui de **Retrofit V2** et le convertisseur de **JSON** :

```
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;
```

Le client Retrofit est à configurer avec une url de serveur ainsi qu'un convertisseur de requête car il permet de convertir du JSON en objet JAVA.

9- Créer l'instance du service

Après avoir créé l'instance du client Retrofit, à la suite dans NetworkFragment, il s'agit de créer l'instance du service comme suit :

```
InterAppreciation interAppreciation = retrofit.create(InterAppreciation.class);
```

et d'importer la classe InterAppreciation.

Ce service est créé à partir du client Retrofit, lequel contient l'url du serveur, et à partir du **.class** de l'interface, laquelle contient toutes les requêtes possibles avec le serveur.

GestSio

10- Créer la requête GET

Toujours à la suite dans NetworkFragment, il s'agit de créer la requête GET :

```
Call<List<Appreciation>> call = interAppreciation.getAppreciation(etudiant.getIdEtudiant());
```

11- Exécuter la requête GET

Enfin, il est possible d'exécuter la requête venant d'être créée via la fonction enqueue(object: Callback<...>) de Retrofit :

```
call.enqueue(new Callback<List<Appreciation>>() {
    @Override
    public void onResponse(Call<List<Appreciation>> call, Response<List<Appreciation>> response) {
        //verifier la reponse
        if(response.code() != 200){
            textreponse.setText("verifie la connetion" +response.code());
            return;
        }
        if (!response.isSuccessful()){
            textreponse.setText("verifie la connetion" +response.code());
        }
        List<Appreciation> Appreciations = response.body();
        for (Appreciation Appreciation : Appreciations){
            String responseTest = "";
            responseTest += Appreciation.getObservationEtudiant() + "\n";
            textreponse.append(responseTest);
        }
      //  String json = "ID=" + response.body().getIdEtudiant() +
        //        "\n Appreciation= " + response.body().getObservationEtudiant();
    }

    @Override
    public void onFailure(Call<List<Appreciation>> call, Throwable t) {
```

```
        textreponse.setText(t.getMessage());
    }
});
```

les imports associés sont les suivants :

```
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
```

ainsi que l'import de la classe Appreciation.

La fonction enqueue(new Callback<…>) créer une nouvelle paramètre l'objet Callback, cet objet intercepte la réponse du serveur. Cette dernière est traitée dans la sur-implémentation des fonctions onResponse(...) et onFailure(...). Les paramètres du Callback ainsi que des fonctions onResponse(...) et onFailure(...) dépendent directement du type de retour de la fonction représentant la requête. Dans notre cas, le type de retour List<Appreciation> (définit par ce que renvoie le serveur)..

12- Récupérer le résultat de la requête

Enfin, la liste des appréciation pour récupérer chaque id (idEtudiant) n'oublie pas cette code textreponse=findViewById(R.id.textViewAppreciation); et textreponse.append(responseTest);.

NELSON MANDELA
Lycée Polyvalent

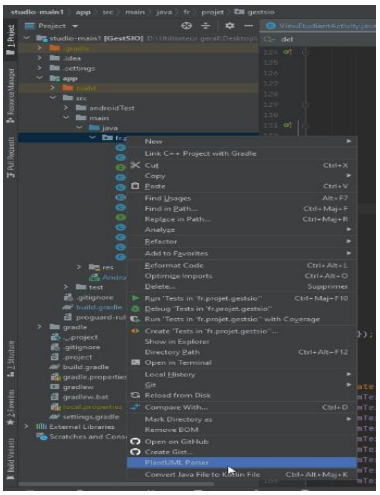
GestSio

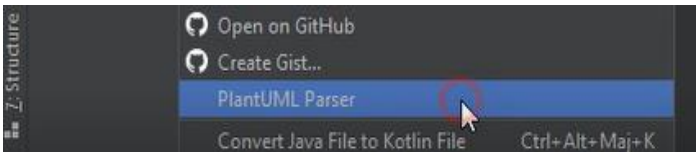
13- Digramme de classe mise à jour par notre équipe

Dans le projet de l'ancien stagiaire comme il y a déjà télécharger l'appication de PlantUML Parser

Le PlantUML Parser permet de convertissez le code source Java en Plantuml.

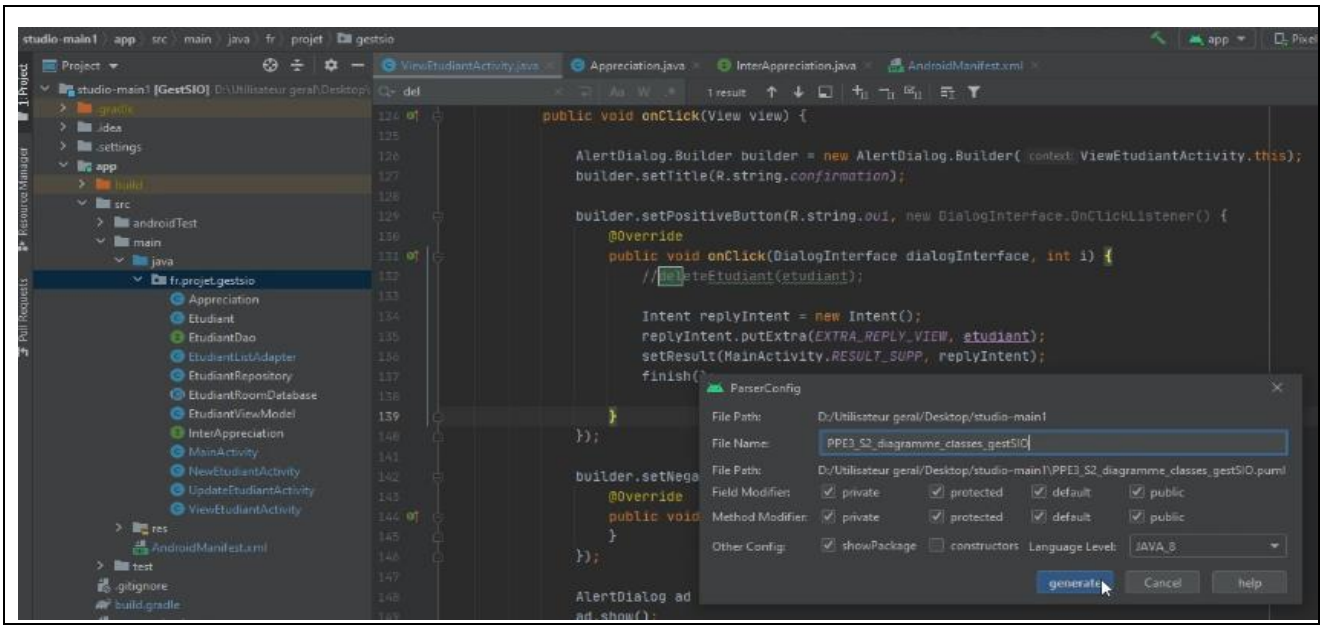
Pour le digramme de classe cliquez droite et cliquez sur **PlantUML Parser** :



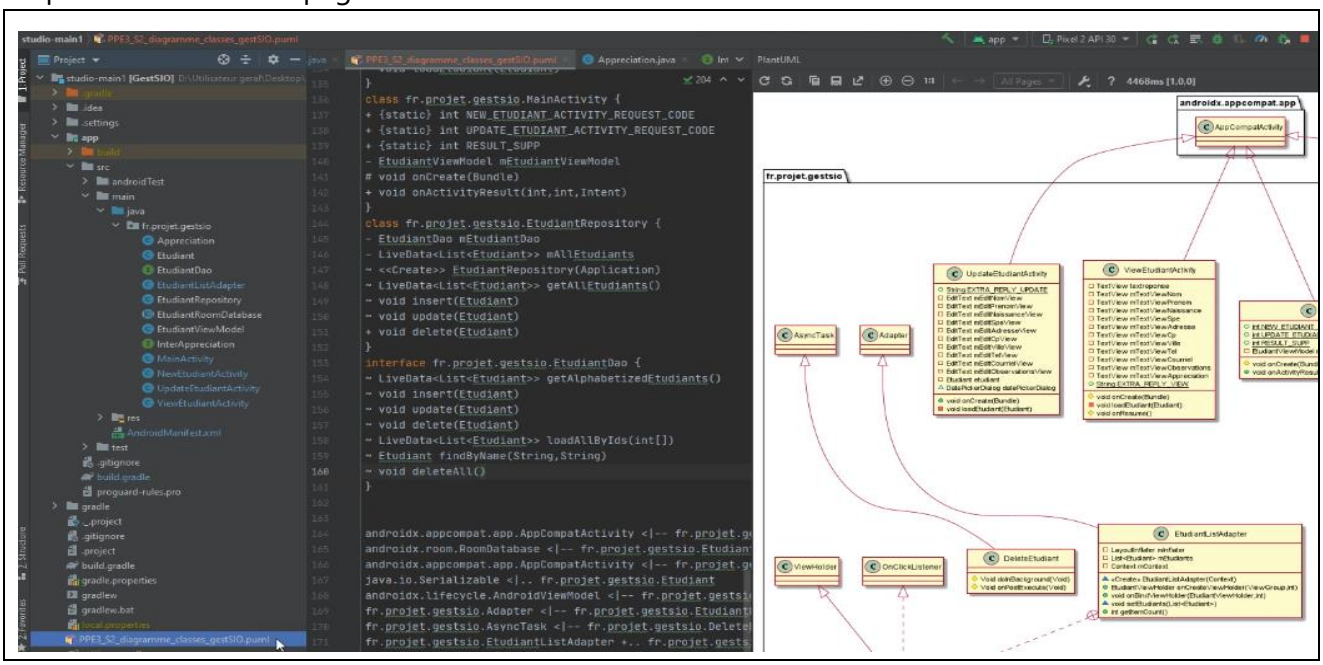


Page ouvre et choisissez le nom comme vous voulez :

GestSio



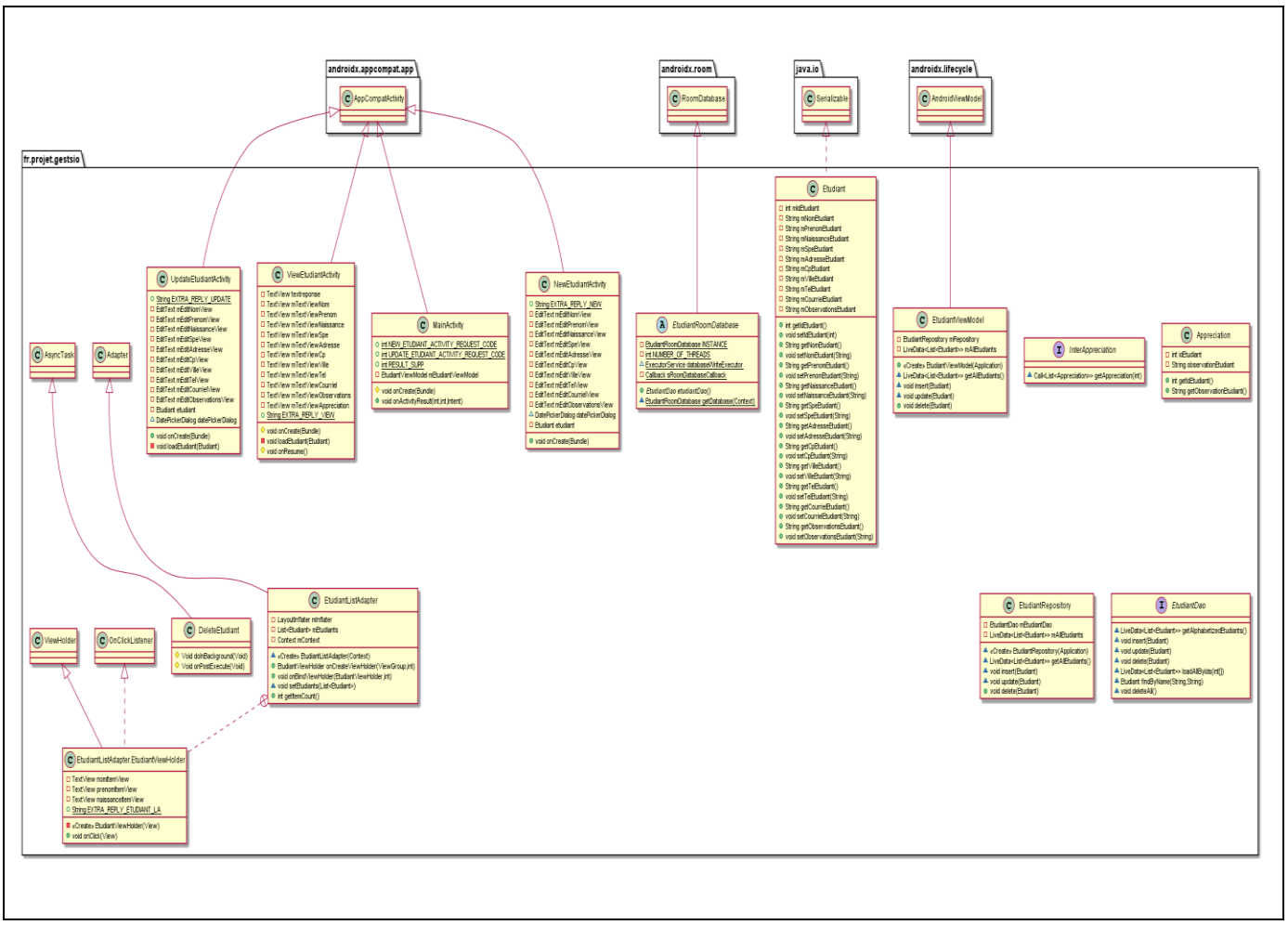
Cliquez **Ok** et un autre page ouvre :



NELSON MANDELA
Lycée Polyvalent

GestSio

Voici le nouveau diagramme de classes de Android studio :



Plus détails : [Android Studio - Google Drive]

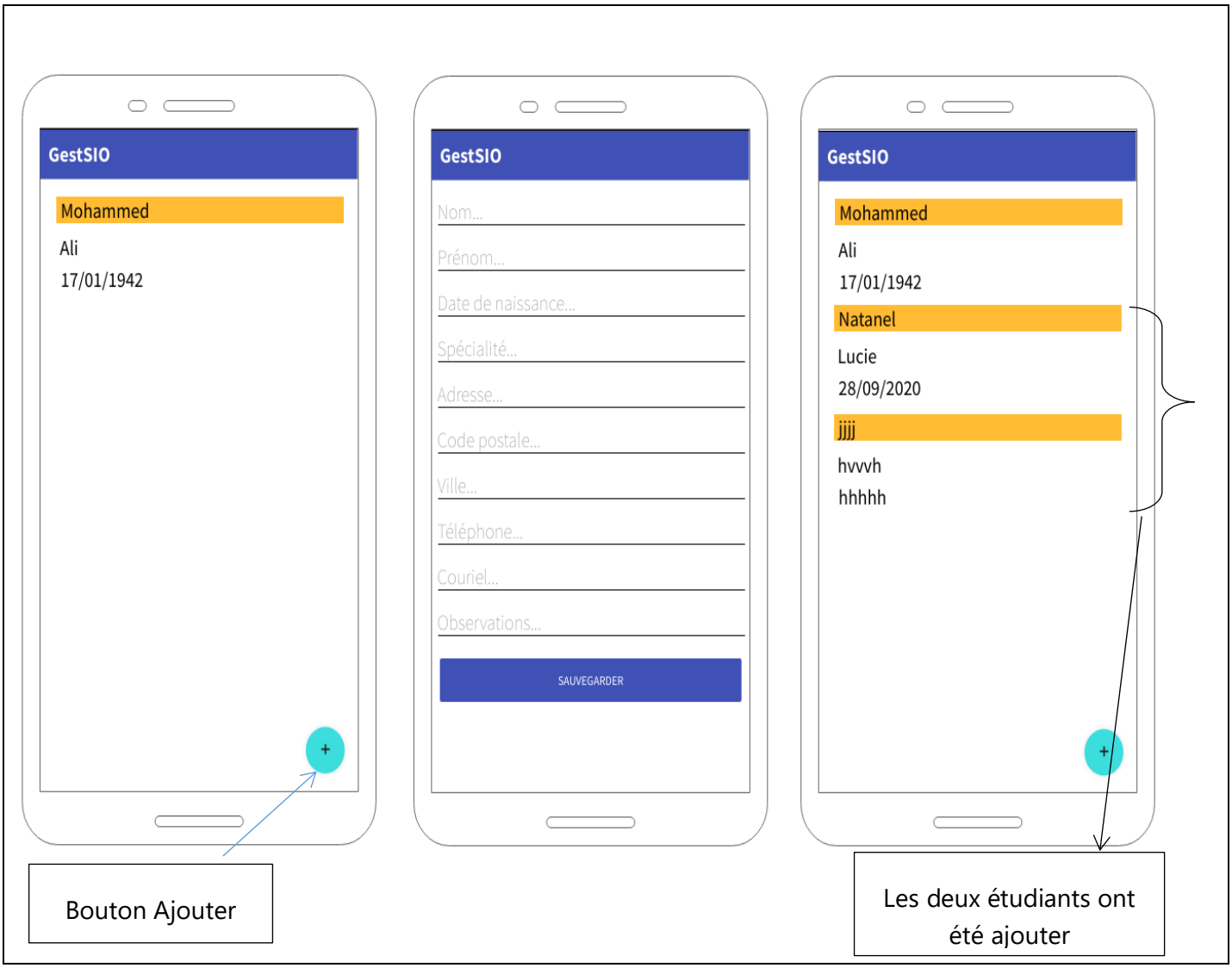
GestSio

14- Maquette application mobile

Pour faire le maquette on a utiliser un logiciel en ligne comme WireframePro

Voici le lien du site : WireframePro - GestSIO / 1_maquette_ajouter (mockflow.com)

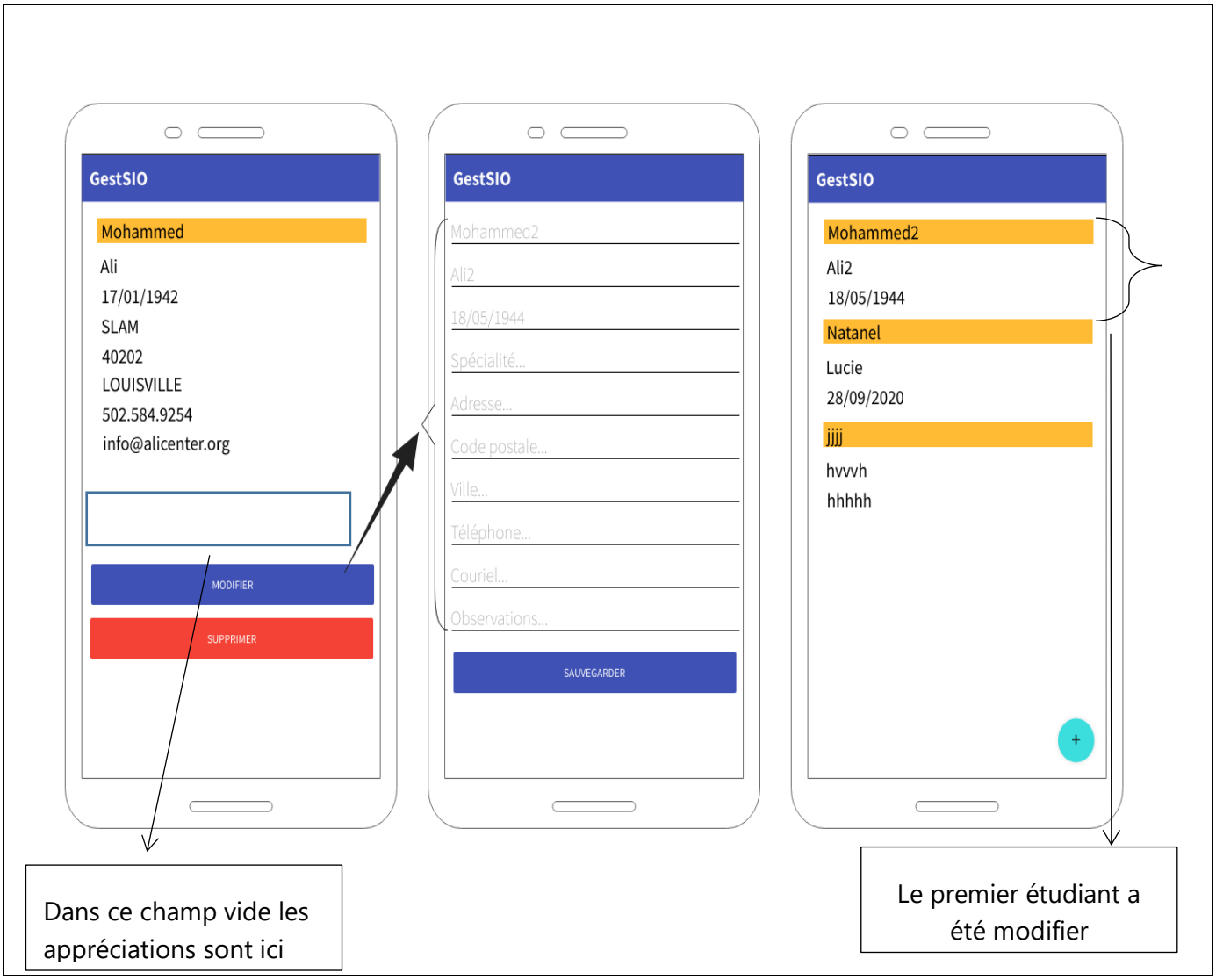
Ajouter un étudiant :



GestSio

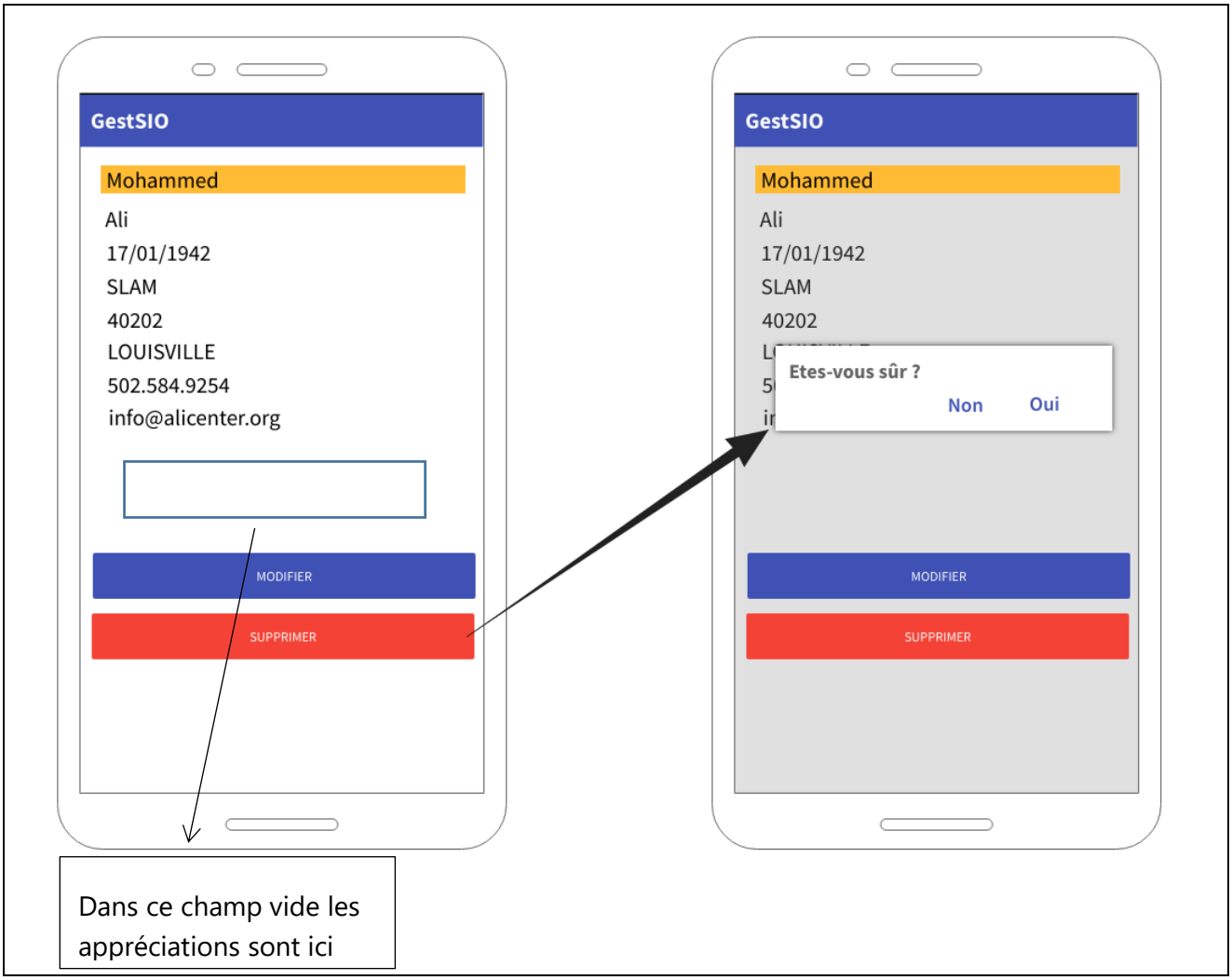
Attention dans cette maquette on ne peut pas modifier l'appréciation que juste seulement dans la base de données en peut le modifier.

Modifier un étudiant :



GestSio

Supprimer un étudiant :



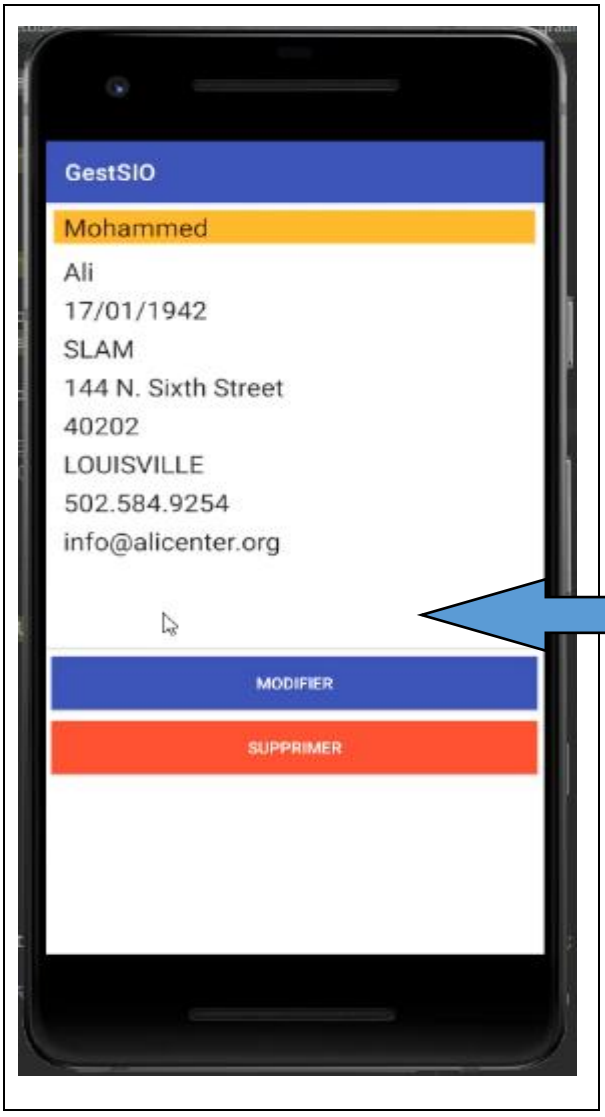
Soit tu choisis non pour annuler ou oui pour supprimer

GestSio

15- Résultat des appréciations

On fait un test dans l'application Android Studio, mais on a vue qu'au début est vide.

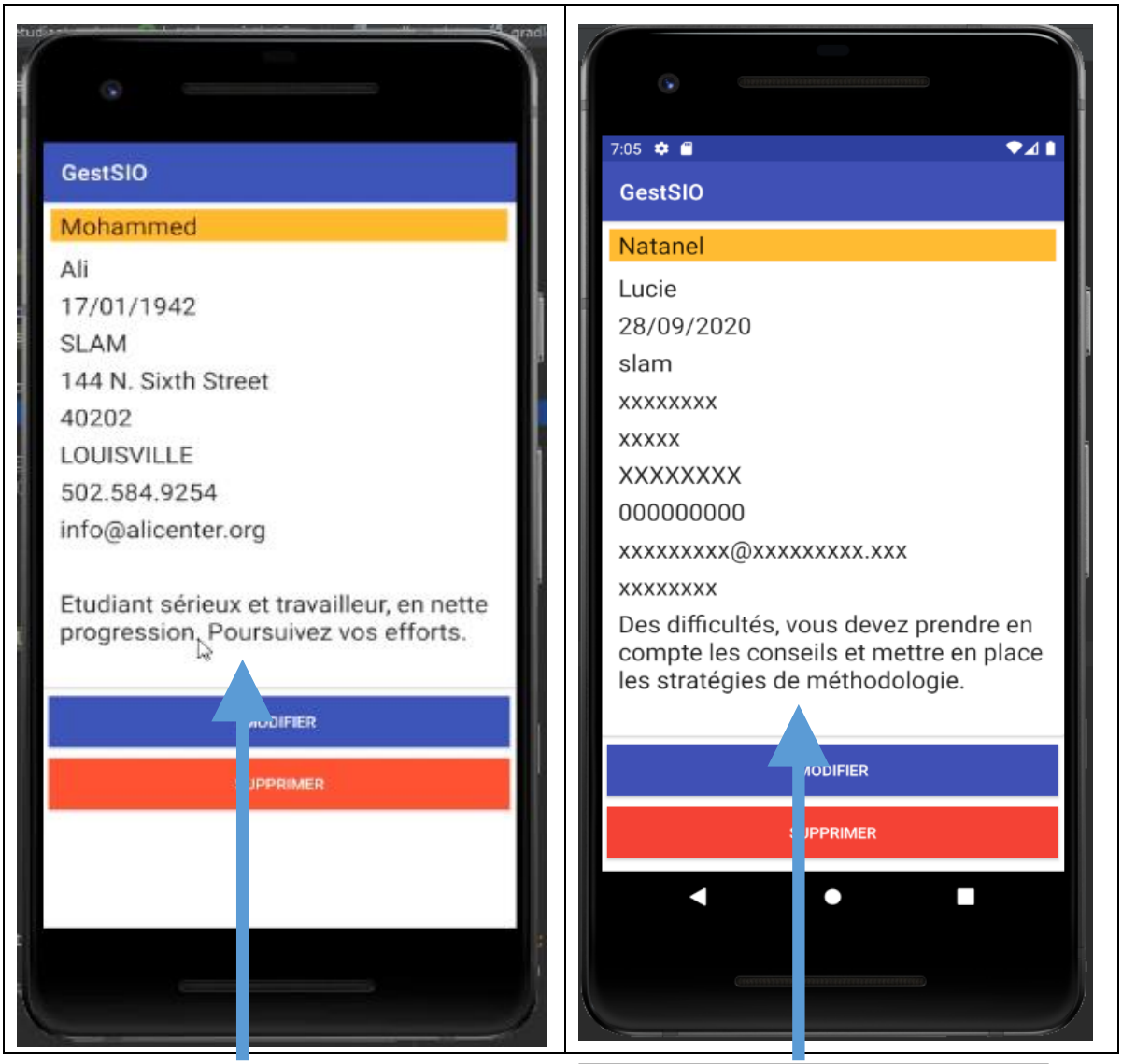
Voir ci-dessous :



GestSio

Et ont ajouté trois étudiants dans l'application mobile et obtient chaque étudiant à son appréciation qu'on a récupérer à distance. À chaque fois ont modifié les appréciations à distance et ça change sur l'application mobile.

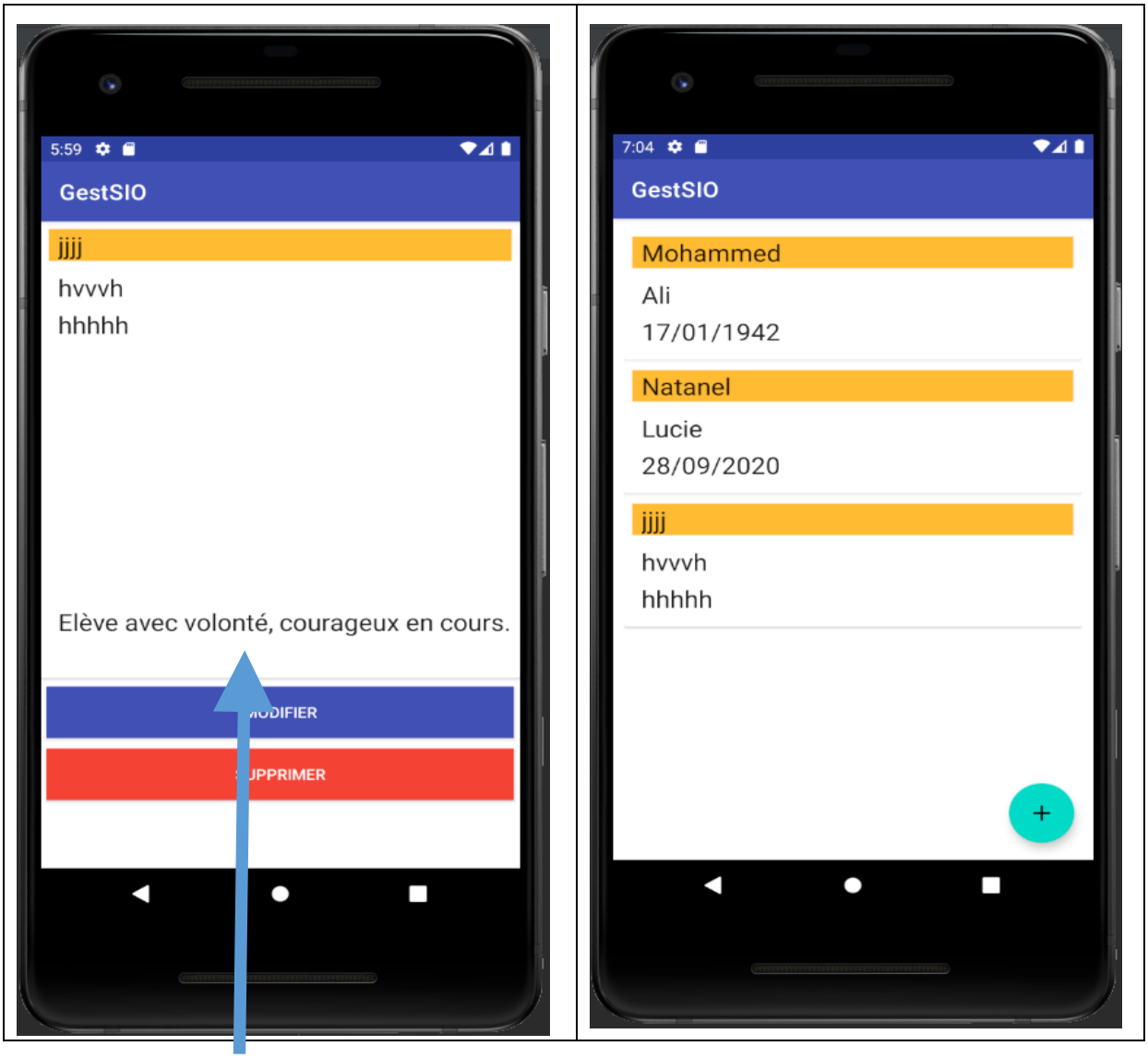
Voici les appréciations qu'on récupère à distance pour chaque id :



La première appréciation qui s'affiche c'est-à-dire idEtudiant = 1

La deuxième appréciation qui s'affiche c'est-à-dire idEtudiant = 2

GestSio



La troisième appréciation qui s'affiche c'est-à-dire idEtudiant = 3