**BTS SIO – Lycée CCI Gard**

**Projet Personnel Encadré (PPE) n°4**

Rapport d’activité

Contexte : GSB

|  |  |
| --- | --- |
| **Intitulé du projet** | Laboratoire GSB - Android |
| **Nom, prénom** | DELAPORTE, Thomas |
| **Année session examen** | 2019 |

**Activités exigées pour l’examen E4**

❒ A1.1.1 - Analyse du cahier des charges d’un service à produire

❒ A1.2.2 - Rédaction des spécifications techniques de la solution retenue (adaptation d’une solution existante ou réalisation d’une nouvelle solution)

❒ A5.2.1 - Exploitation des référentiels, normes et standards adoptés par le prestataire informatique

***Important*** : vous devez présenter des tests vraisemblables et représentatifs pour chaque traitement.

**Sommaire**

[1. Contexte général et technique 3](#_Toc8287574)

[A. Description de l’entreprise 3](#_Toc8287575)

[B. Objectifs du projet 4](#_Toc8287576)

[2. Analyse des données 5](#_Toc8287577)

[A. Règles de gestion 5](#_Toc8287578)

[B. MCD (jMerise) 6](#_Toc8287579)

[C. MLD 7](#_Toc8287580)

[D. Validation de la base et tests 8](#_Toc8287581)

[E. Script de la base de données 8](#_Toc8287582)

[F. Description des traitements 8](#_Toc8287583)

[a. Déclencheurs, vues et héritages 8](#_Toc8287584)

[3. Android 10](#_Toc8287585)

[A. Diagramme des cas d’utilisation 10](#_Toc8287586)

[B. Liste des écrans (« Activity ») 10](#_Toc8287587)

[C. Enchainement des écrans 11](#_Toc8287588)

[D. Utilitaires 12](#_Toc8287589)

[a. Gestion de Session 12](#_Toc8287590)

[b. Gestion cryptage du mot de passe 12](#_Toc8287591)

[C. Widgets personnalisés 13](#_Toc8287592)

[a. « ViewPager » statique 13](#_Toc8287593)

[b. Dialogue avec liste de choix 13](#_Toc8287594)

[4. Conclusion 14](#_Toc8287595)

[A. Éléments restant à développer 14](#_Toc8287596)

[B. Difficultés rencontrées 14](#_Toc8287597)

[C. Notions apprisses et comprises 14](#_Toc8287598)

# Contexte général et technique

## Description de l’entreprise

#### Activité

Le laboratoire Galaxy Swiss Bourdin (GSB) est issu de la fusion entre le géant américain Galaxy (spécialisé dans le secteur des maladies virales dont le SIDA et les hépatites) et le conglomérat européen Swiss Bourdin (travaillant sur des médicaments plus conventionnels), lui-même déjà union de trois petits laboratoires.

En 2009, les deux géants pharmaceutiques ont uni leurs forces pour créer un leader de ce secteur industriel. L'entité Galaxy Swiss Bourdin Europe a établi son siège administratif à Paris.

Le siège social de la multinationale est situé à Philadelphie, Pennsylvanie, aux Etats-Unis.

La France a été choisie comme témoin pour l'amélioration du suivi de l'activité de visite.

#### Organisation

Une conséquence de cette fusion, est la recherche d'une optimisation de l’activité du groupe ainsi constitué en réalisant des économies d’échelle dans la production et la distribution des médicaments (en passant par une nécessaire restructuration et vague de licenciements), tout en prenant le meilleur des deux laboratoires sur les produits concurrents.

L'entreprise compte 480 visiteurs médicaux en France métropolitaine (Corse comprise), et 60 dans les départements et territoires d'outre-mer. Les territoires sont répartis en 7 régions géographiques (Paris-Centre, Sud, Nord, Ouest, Est, DTOM Caraïbes-Amériques, DTOM Asie-Afrique).

#### Organisation du projet

Environnement :

* **SGBD** : SQLite
* **Environnement de développement** : Emulateur Android
  + Gradle
  + Dépendances :
    - DBFlow (ORM pour SQLite)
    - Android Support Library

#### Outils de développement

Logiciels utilisés :

* **Gestion des versions**: Dossier compressé sauvegardé.
* **Gestion base de données** : SQL DB Browser (logiciel portable)
* **Modélisation** : jMerise
* **IDE** : Android Studio

#### Normes et conventions utilisées

Convention de nommage, base de données :

* **Nom de table**: pluriel sans majuscule sauf mot composé
* **Clé primaire** : id*<Nom de table avec majuscule> (sauf exception)*

Structure des fichiers :

* Data *(Ce qui est lié à la base de données)*
  + Converter
  + Models
* Helper *(Des utilitaires : gestion de Session, Cryptage des données)*
* UI *(Interfaces)*
  + Adapter
  + Base
  + Modules *(Toutes les « Activity »)*
  + Widget *(Composants personnalisés)*

## Objectifs du projet

Chaque employé visiteur médical est chargé de rendre visite à des praticiens.

Lors de chaque visite, il peut laisser des échantillons de médicaments. De plus, il doit enregistrer un compte rendu de chaque visite, qui précise des demandes spécifiques du praticien.

Chaque visiteur médical accèdera sur son mobile aux visites à réaliser dans la journée. Il devra saisir les informations concernant ses visites, et renvoyer ces éléments pour mise à jour dans la base centrale.

L’application doit également proposer une authentification sécurisée.

# Analyse des données

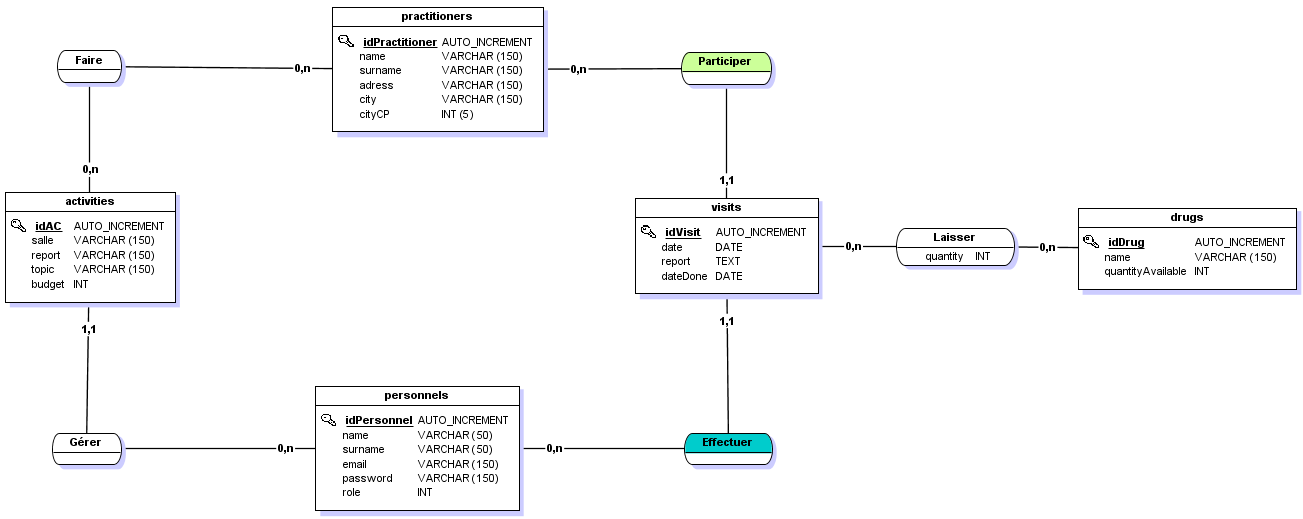
La base de données a été simplifié par rapport au projet « PPE3 » pour correspondre au diagramme de cas d’utilisation présenté page 6.

Les héritages ont été supprimés.

## Règles de gestion

* Personnel
  + Deux rôles : DELEGUE (1) et VISITEUR MEDICAL (0).
  + Possède un nom, prénom, email.
  + Gestion de la connexion avec l’email et un mot de passe.
  + Un visiteur médical peut réaliser des visites.
  + Un délégué peut ajouter des visites.
* Praticiens
  + Possède un nom, prénom, adresse, ville, code postal.
  + Un praticien peut participer à plusieurs visites ainsi qu’a plusieurs activités.
* Médicaments
  + Possède un nom et une quantité disponible en stock.
* Missions des visiteurs :
  + Visites :
    - Réalisé entre un praticien et un visiteur médical.
    - Connaitre la date, le compte rendu (rédigé librement par le visiteur).
    - Les médicaments présentés et le nombre d’échantillons offerts de chaque médicament.
  + Activités complémentaires :
    - Plusieurs praticiens peuvent y participer.
    - Thème de l’activité.
    - Salle utilisée.
    - Compte rendu sous forme de texte-libre.
    - Visiteur responsable de l’Activité.
    - Possède un budget attribué.

## MCD (jMerise)



## MLD

personnels (idPersonnel, name, surname, email, password, role)

practitioners (idPractitioner, name, surname, adress, city, cityCP)

visits (idVisit, date, report, dateDone)

drugs (idDrug, name, quantityAvailable)

drugsleft (idDrugLeft, #idDrug, #idVisit, quantity)

activities (idAC, #idPersonnel, salle, report, theme, state, budget)

activityParticipants (idAcPar, #idActivities, #idPractitioner)

## Validation de la base et tests

Points à traiter :

* Vérification des types de données
* Respect des conventions de nommage
* Contraintes sur champs [PK, index, NULL interdit, …]
* Relations et leur contrainte [FK, contrainte de référence, effacement en cascade]
* Jeux d’essai représentatifs et vraisemblables

## Script de la base de données

Générer par DBFlow : *« LaboratoireGSB\_creation.sql »*

## Description des traitements

### Déclencheurs, vues et héritages

#### Déclencheurs standards

#### Mise à jour du stock de médicament lors d’ajout ou de modification de médicaments laissés lors d’une visite.

Lorsque on insère, on enlève le montant d’échantillon laissé au stock total du médicament.

1. **CREATE** **TRIGGER** drugleft\_insert
2. BEFORE **INSERT** **ON** drugsleft
3. **BEGIN**
5. **SELECT** CASE **WHEN** NEW.quantity <= 0 **THEN**
6. RAISE(ABORT, 'Quantité invalide <= 0')
7. **END**;
9. **UPDATE** drugs **SET** quantityAvailable = quantityAvailable - NEW.quantity **WHERE** idMedicament = NEW.drug\_idMedicament;
10. **END**

Lorsque on met a jour des médicaments laissés et que le nouveau résultat est différent de 0 on ajoute ou enlève les médicaments du stock total.

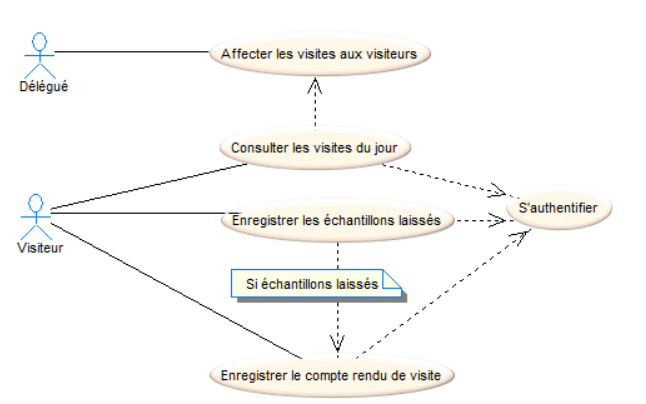
1. **CREATE** **TRIGGER** drugleft\_update
2. BEFORE **UPDATE** **ON** drugsleft
3. **WHEN** NEW.quantity <> 0
4. **BEGIN**
5. **UPDATE** drugs **SET** quantityAvailable = quantityAvailable + (OLD.quantity - NEW.quantity) **WHERE** idMedicament = NEW.drug\_idMedicament;
6. **END**

Lorsque on met à jour des médicaments laissés et que le nouveau résultat est égal à 0, on supprime le médicament laissés de la base de données.

1. **CREATE** **TRIGGER** drugleft\_update\_delete
2. BEFORE **UPDATE** **ON** drugsleft
3. **WHEN** NEW.quantity = 0
4. **BEGIN**
5. **UPDATE** drugs **SET** quantityAvailable = OLD.quantity **WHERE** idMedicament = NEW.drug\_idMedicament;
6. **DELETE** **FROM** drugsleft **WHERE** idDrugLeft = NEW.idDrugLeft;
7. **END**

# Android

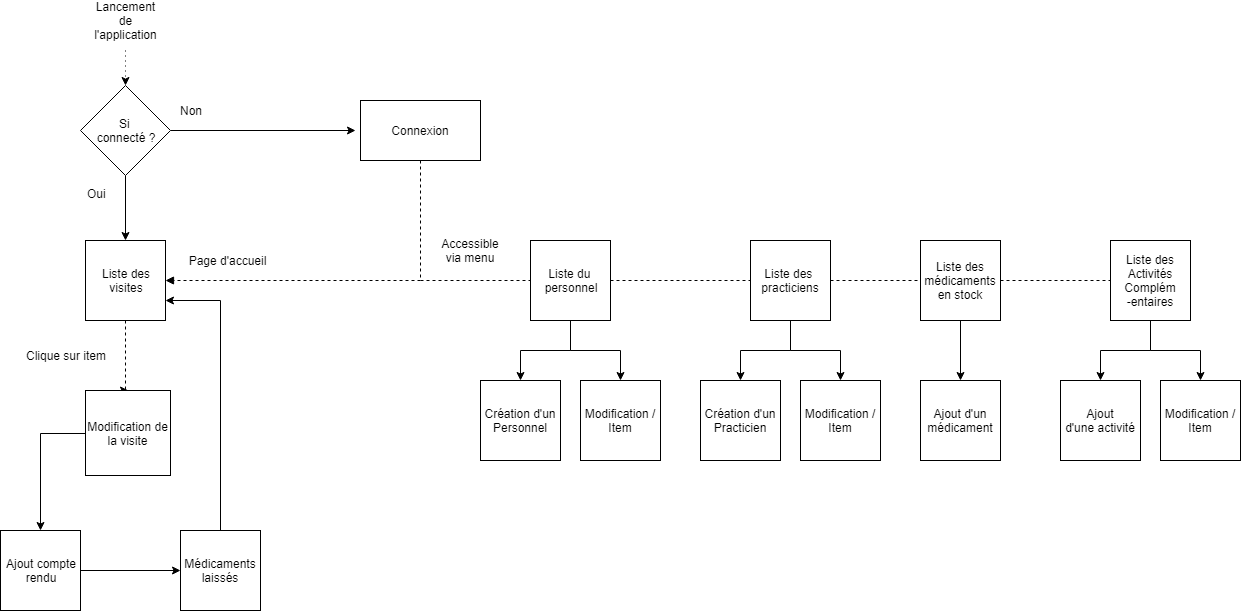
### Diagramme des cas d’utilisation



### Liste des écrans (« Activity »)

* Connexion
* Visites
  + Listes des visites
  + Modification (sous forme d’étapes)
    - Compte rendu
    - Médicaments laissés
* Personnels
  + Liste (avec suppression)
  + Création
  + Modification
* Praticiens
  + Liste (avec suppression)
  + Création
  + Modification
* Médicaments en stock
  + Liste
  + Création
* Activités complémentaires
  + Liste
  + Création
  + Modification

### Enchainement des écrans



## Utilitaires

### Gestion de Session

Les données de l’utilisateur après la connexion sont stockées dans un fichier : « data/data/delaporte.fr.ppe4\_delaporte/shared\_prefs/fr.delaporte.account.xml »

Sous la forme suivante :

1. **<?xml** version='1.0' encoding='utf-8' standalone='yes' **?>**
2. **<map>**
3. **<boolean** name="isLogin" value="true" **/>**
4. **<int** name="role" value="1" **/>**
5. **<string** name="email"**>**delegue@cci.fr**</string>**
6. **<int** name="idPersonnel" value="1" **/>**
7. **</map>**

### Gestion cryptage du mot de passe

Le mot de passe doit être crypté dans la base de données.

Utilisation de « SHA-512 », ainsi qu’un « salt ».

1. **public** **class** Encryption {
3. **private** **static** **byte**[] SALT = ("laboGSB").getBytes();
5. **public** **static** String encryptSHA512(String passwordToHash){
7. String generatedPassword = **null**;
9. **try** {
10. MessageDigest md = MessageDigest.getInstance("SHA-512");
11. md.update(SALT);
13. **byte**[] bytes = md.digest(passwordToHash.getBytes());
14. StringBuilder sb = **new** StringBuilder();
16. **for**(**int** i= 0; i< bytes.length; i++) {
17. sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).substring(1));
18. }
20. generatedPassword = sb.toString();
21. } **catch** (NoSuchAlgorithmException e) {
22. e.printStackTrace();
23. }
25. **return** generatedPassword;
26. }
27. }

## Widgets personnalisés

### « ViewPager » statique

Pour le système d’étape, j’utilise un widget « ViewPager » qui me permet de mettre sur un même écran plusieurs sous écran (« Fragment »). Seulement un « ViewPager » classique peut être contrôlé avec le mouvement du doigt, pour changer de vue.

Il a fallu de ce fait bloquer toutes interactions de l’utilisateur.

**Class** : « NonSwipeableViewPager »

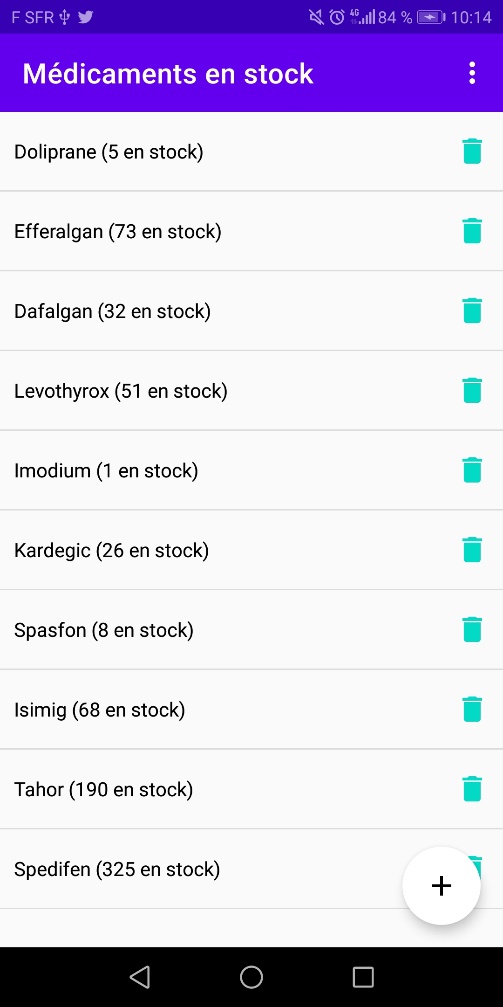
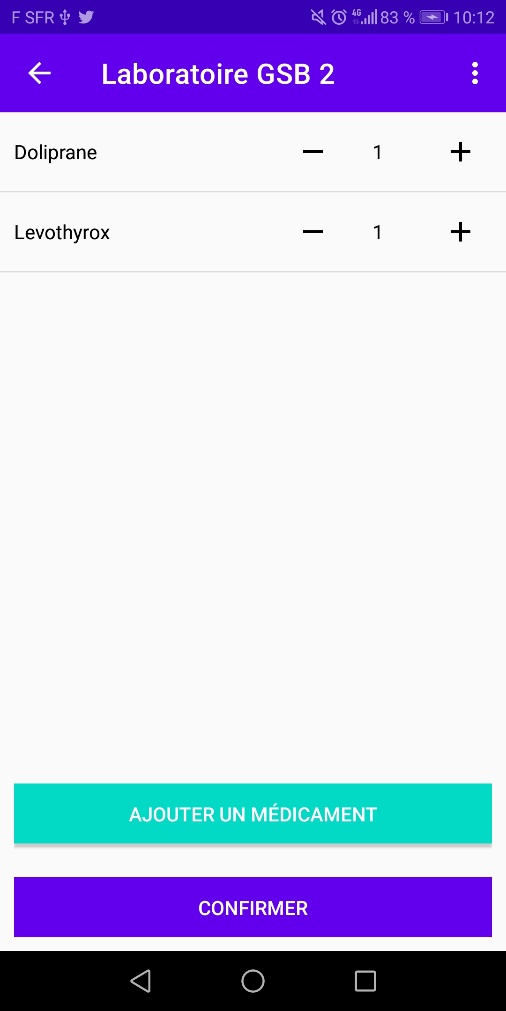
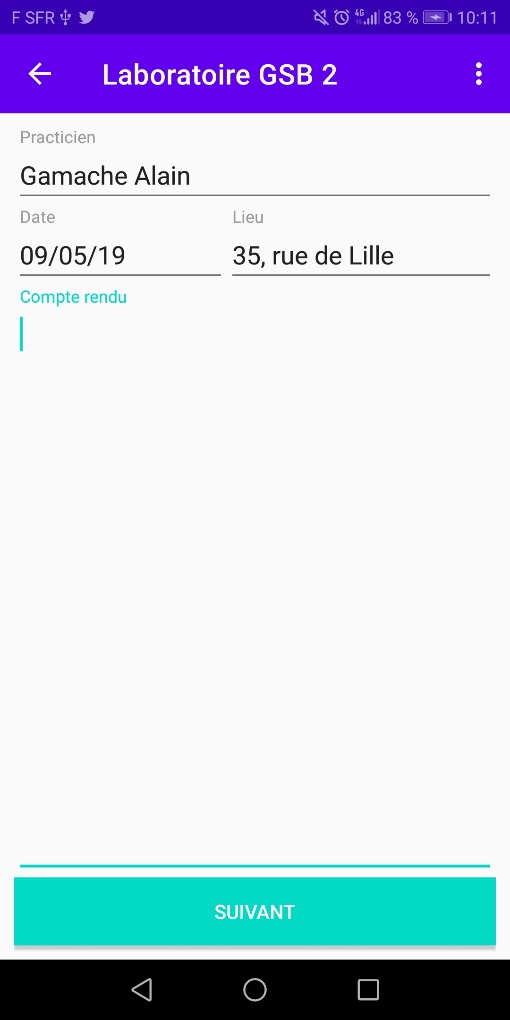
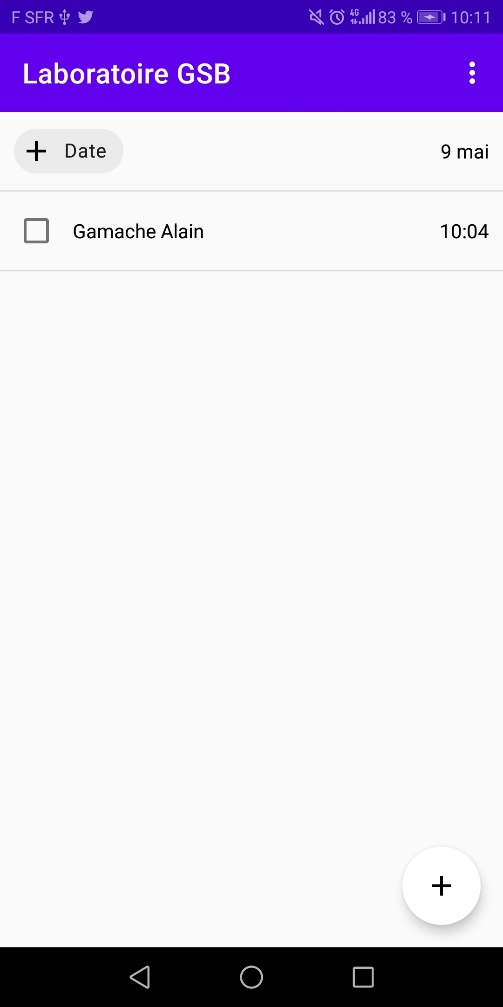
### Dialogue avec liste de choix

Création d’une boite de dialogue avec un « Spinner » à l’intérieur.

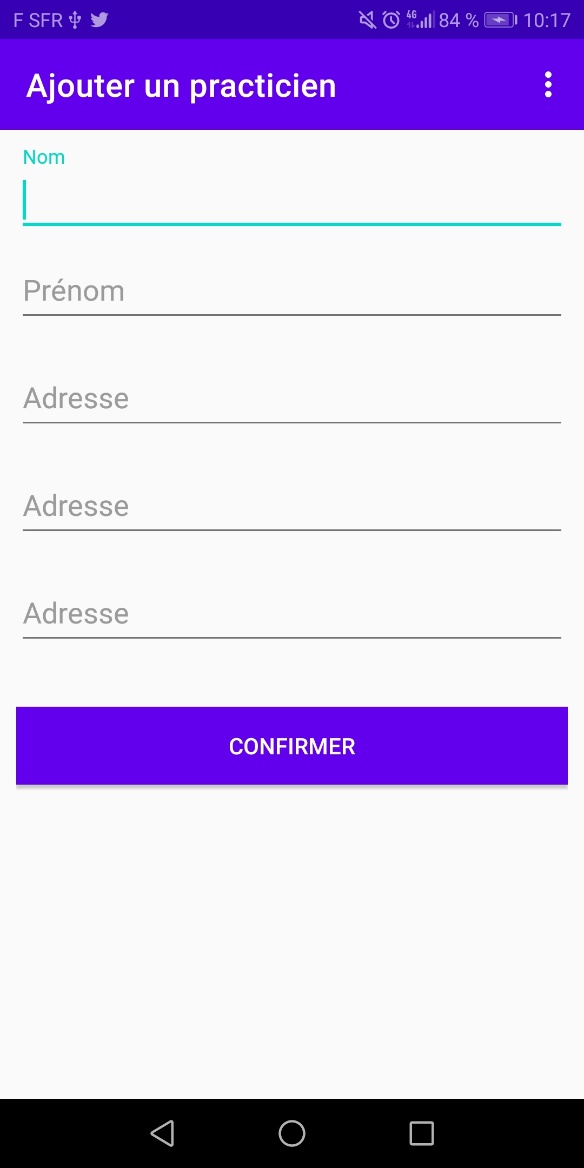
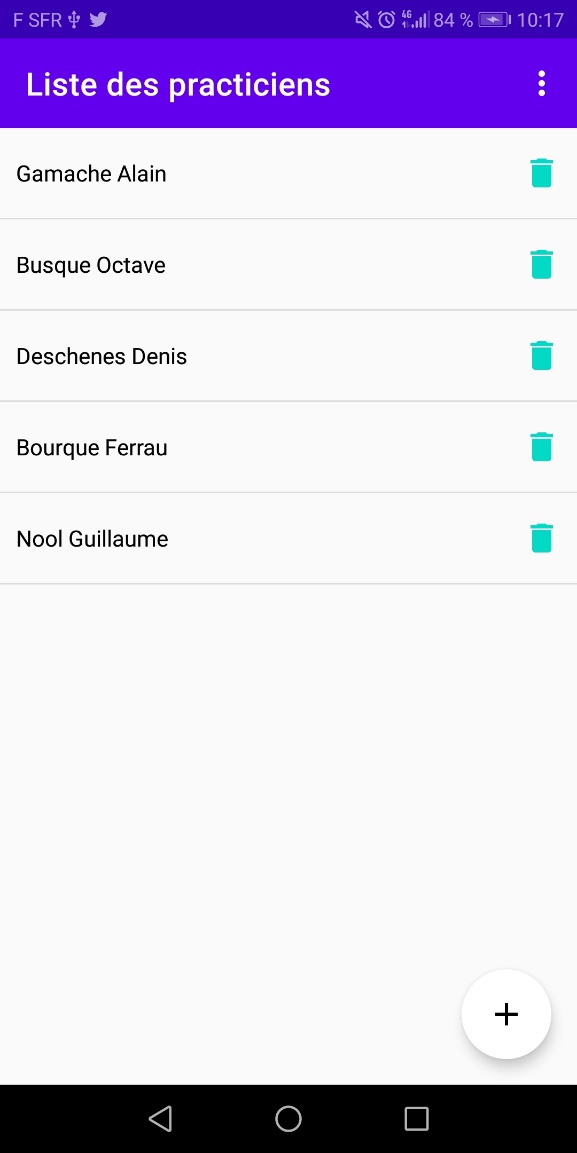
**Class** : « SpinnerDialog »

1. SpinnerDialog dialog = **new** SpinnerDialog(getContext());
2. dialog.setTitle("Compte rendu prédéfini");
4. ArrayAdapter<String> spinnerAdapter = **new** ArrayAdapter<>(Objects.requireNonNull(getContext()), android.R.layout.simple\_spinner\_item, **new** String[]{"a", "b", "c"});
6. dialog.setAdapter(spinnerAdapter);

# Gestion des visites



# Gestion des praticiens (Exemple de CRUD)



# Conclusion

## Éléments restant à développer

Filtre pour lister les visites d’un Visiteur Médical pour les délégués.

Mode connexion et hors connexion.

## Difficultés rencontrées

Triggers dans SQLite différents de ceux sous SQL Server.

## Notions apprisses et comprises

Utilisation et création d’une base de données SQLite

Création d’une application **Android** :

* Gestion des « Activity » (interfaces)
* Création et utilisation d’« Adapter »