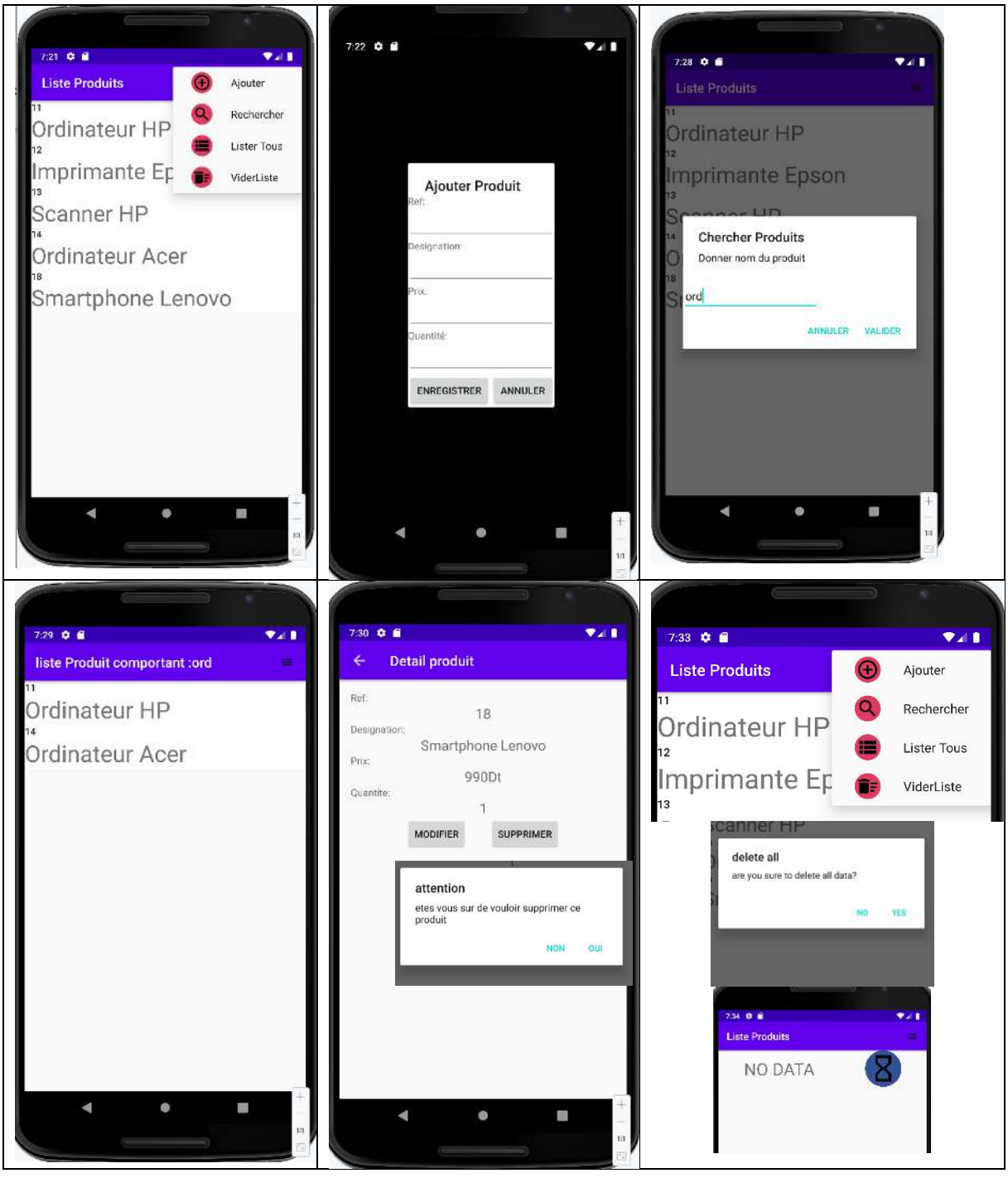
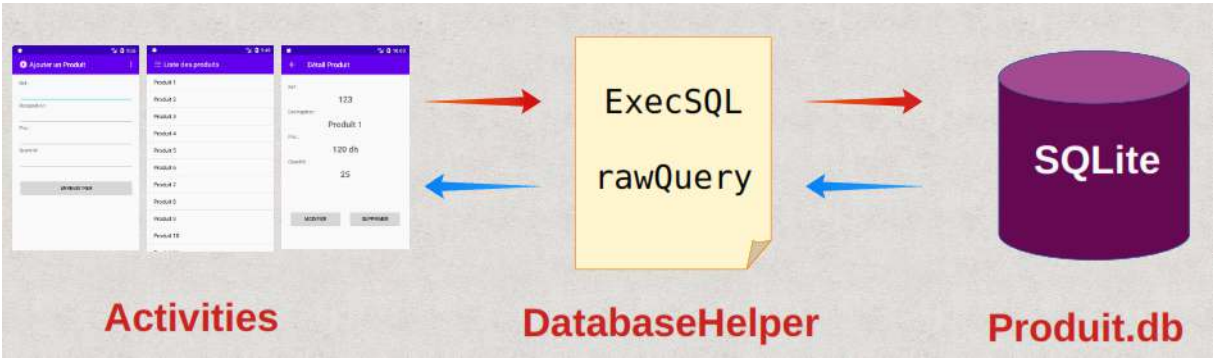
TP 8 Utilisation de base de données SQLite sous Android

Les Opérations de base: CREATE ;INSERT; SELECT;UPDATE ;DELETE
Notre application à réaliser apparaîtra comme les images ci-dessus :



Pour ce faire :



Vous devez créer une classe utilitaire pour travailler avec la base de données **SQLite**, cette classe devrait s'étendre de la classe **SQLiteOpenHelper**. Il existe deux méthodes importantes dont vous devez remplacer (override): **onCreate()** et **onUpgrade()**.

1. Créez la classe **DatabaseHelper** s'étend à partir de **SQLiteOpenHelper**.
2. Après avoir étendu votre classe à partir de **SQLiteOpenHelper** vous devez outre passer deux méthodes ***onCreate()*** et ***onUpgrage()***
    - ***onCreate()*** - C'est là où vous devez écrire créer des instructions de table. C'est ce qu'on appelle (called) lorsque la base de données est créée.
    - ***onUpgrade()*** - Cette méthode est appelée lors de la mise à niveau de la base de données, comme la modification de la structure de la table, l'ajout de contraintes à la base de données, etc.

```
DatabaseHelper
public class DatabaseHelper extends SQLiteOpenHelper {
   private static DatabaseHelper instance;

   private DatabaseHelper(Context context) {}
@Override
   public void onCreate(SQLiteDatabase db) {}
@Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){}
}
```

1.Créer un projet et concevoir l'interface principale : Création Base de données

Commençons par l'interface principale ListeProduit.xml

Développer le fichier xml correspondant

…….

Maintenant pour la création de la base de données, nous devons créer une classe java DatabaseHelper qui herite de SQLiteOpenHelper

Avec l'ajout de constructeur, ces deux méthodes onCreate et onUpgrade

……..

Maintenant pour tester ceci

Ajouter l'événement correspondant dans l'interface Principale Option AjouterProduit
Et développer la méthode correspondante

```
SQLiteDatabase database;
DatabaseHelper dbhelper;
if (item.getItemId() == R.id.ajout) {
    dbHelper = new DatabaseHelper(this);
    database = dbHelper.getWritableDatabase();
    return true;

}
//ou bien tester avec un simple bouton action
```

Tester la création de la base de données dans View→Tool Windows→DeviceFileExplorer→Data→Data et chercher la base données correspondante à notre projet.

Puis Database →le fichier Produits.db que vous aller exporter et l'ouvrir avec DB Browser
Ou autre. (lecture fichier *.db)
Ou à travers App Inspection → DataBase Inspection

Donc ici création base de données et Table

2.Créer le Menu Option et le recyclerView

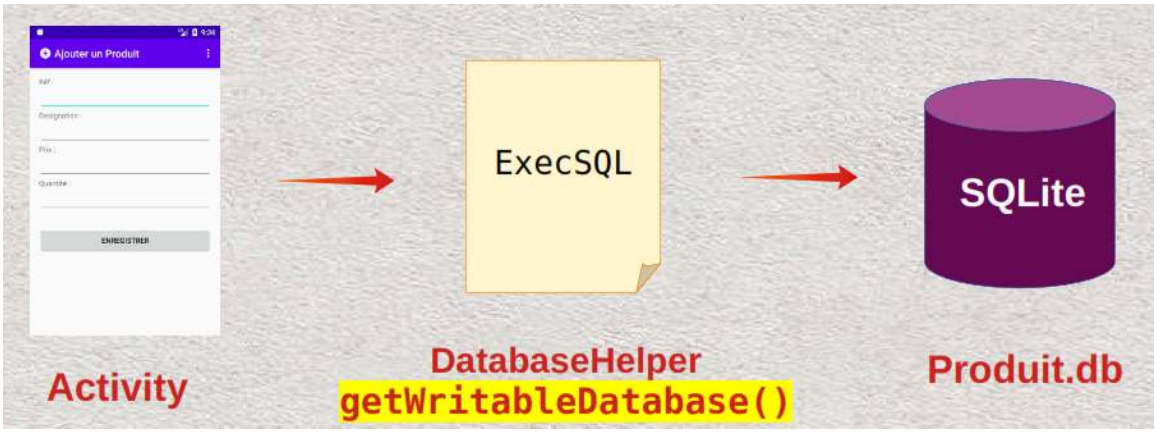
Créer le contenu de la page principale **MenuOption**

- Les actions seront développée au fur et a mesure de la manipulation de notre Base de données CRUD.

Créer votre Adapter afin d'afficher votre **RecyclerView**

3.Insertion de données

Passons maintenant à l'insertion



Accédons maintenant à notre Helper en mode écriture getWritableDatabase afin de pouvoir insérer des données.

Dans notre application :
Ajouter une empty Activity : **AjouterProduitActivity** et générer son Layout

Afin que votre layout apparaîtra comme étant boite de dialogue ajouter et appliquer un thème qui extend d'un DIALOG
Donc Ajoutez sous res/values/style.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="mytheme" parent="Theme.AppCompat.Light.Dialog">
    </style>
</resources>
```

Et dans votre fichier manifest appliquez votre style et n'oublier pas à chaque fois de définir le layout parent afin de permettre de naviguer entre les différentes activités (une flèche de retour sera ajoutée automatiquement dans l'ActionBar:

```
<activity android:name=".AjouterProduitActivity"
    android:parentActivityName=".MainActivity"
    android:theme="@style/mytheme"/>
```

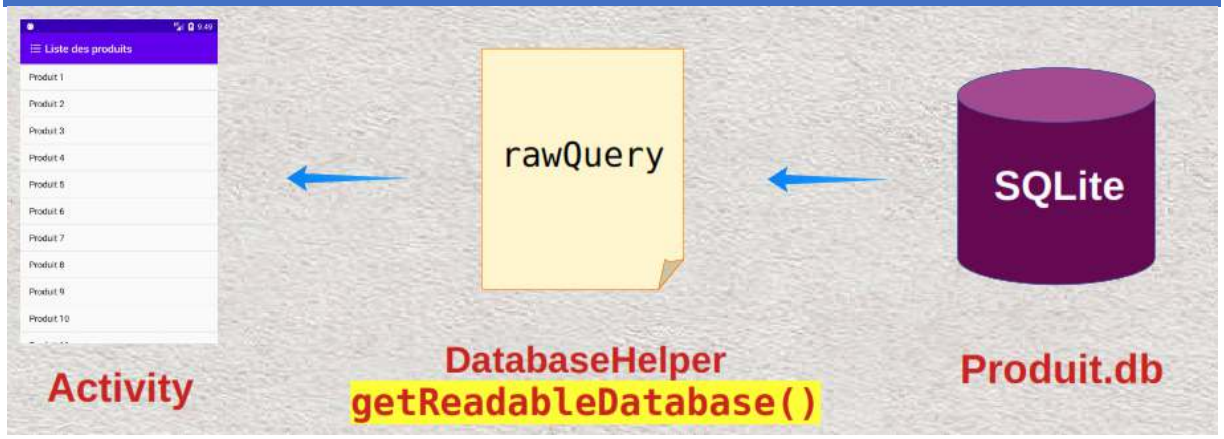
Pour la classe Java **AjouterProduitActivity,** récupérer les Editexts correspondant, développer la fonction **Enregistrer** produits et y mettre la **requête d'insertion correspondante**.

Pour la page principale, il faut implémenter l'événement ajout produit correspondant (ITEM OPTION) et développer l'intent correspondant pour l'appel a notre formulaire.

Tester et vérifier l'ajout correspondant dans la base de données

Passons maintenant a la lecture des données insérée : la liste des enregistrements

4.Selection de données



Les données sont insérée dans un RecyclerView, pour la lecture nous allons utiliser **getReadableDatabase**

Pour **execSQL** retourne int, nbre ligne affectée selon le nbre de lignes insérées

Ici, la sélection c'est un résultat récupérée sous forme de lignes, nous allons utiliser un curseur en utilisant **rawQuery**(**req**, **param**)

Et puis lecture et parcours des différentes lignes par index de lignes, 0 pour le premier champ….

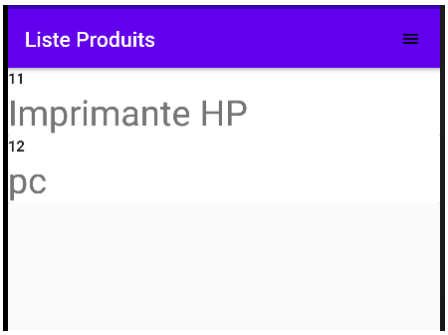
Ou bien par nom de colonne **getColumnIndex**("column1")

Modifier la classe Java **ListProduitsActivity**

Récupérer les données Produits et l'affecter a notre recyclerview

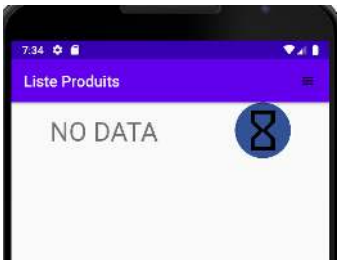
Les produits doivent être enregistrées dans une classe **Java Bean Produits**

Item choisi doit être personnalisé, ainsi créer un layout pour chaque item : produit_item.xml comportant la référence et la désignation du produit.



Tester maintenant la liste de produits

Traiter le cas ou la liste est Vide



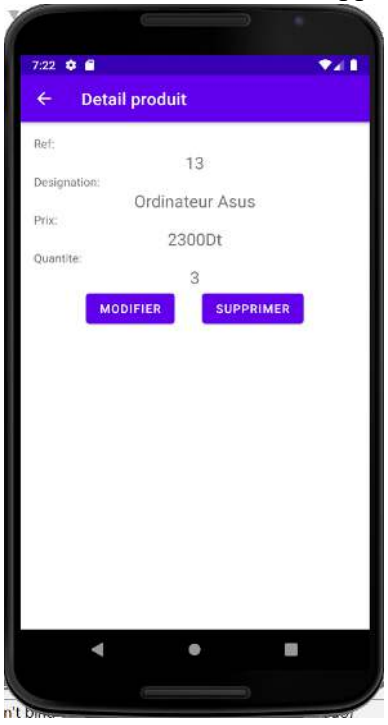
Passons maintenant a la suppression ou mise ajour d'un produit existant

5.Mise à Jour de données

Ici on aura deux possibilités pour un produit ou bien le modifier ou le supprimer, la méthode utilisé sera **getWritableDatabase**

Dans notre exemple, l'objectif est de choisir un élément de la liste, lorsqu'on clique dessus, on affiche son détail et nous allons s'orienter vers modification ou suppression

Pour ce faire
Créer empty activity **DetailProduitActivity**
et le layout correspondant avec deux bouton modifier et supprimer



Pour le code Java
Modifier notre liste **ListProduitActivity** en modifiant la classe `CustumAdapterProduit` afin d'ajouter l'evenement correspondant au click sur un élément (item) de la liste et afficher donc le layout **detail_produits**
L'objet que nous devons passer doit être **sérialisable**, pour ce faire il faut implémenter la méthode **sérialisable**, il suffit que notre **bean Produits implements serialisable** et donc ajouter attribut id, car toute donnée de l'objet doit etre ajoutée.

```
//passer un objet prd dans l'INTENT
 intent.putExtra("data",prd );
```

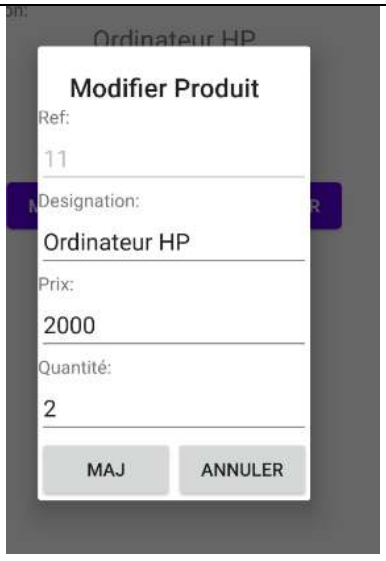
```
  //recuperer produit qui est serialisable
prd=(Produits)intent.getSerializableExtra("data");
```

Modifier **DetailProduitActivity**, pour réceptionner les paramètres passés dans le onCreate et récupérer l'intent.

Dans **AjouterProduitActivity** dans le onCreate récupérer l'intent, vérifier les données et fixer le mode modifier ou ajouter.

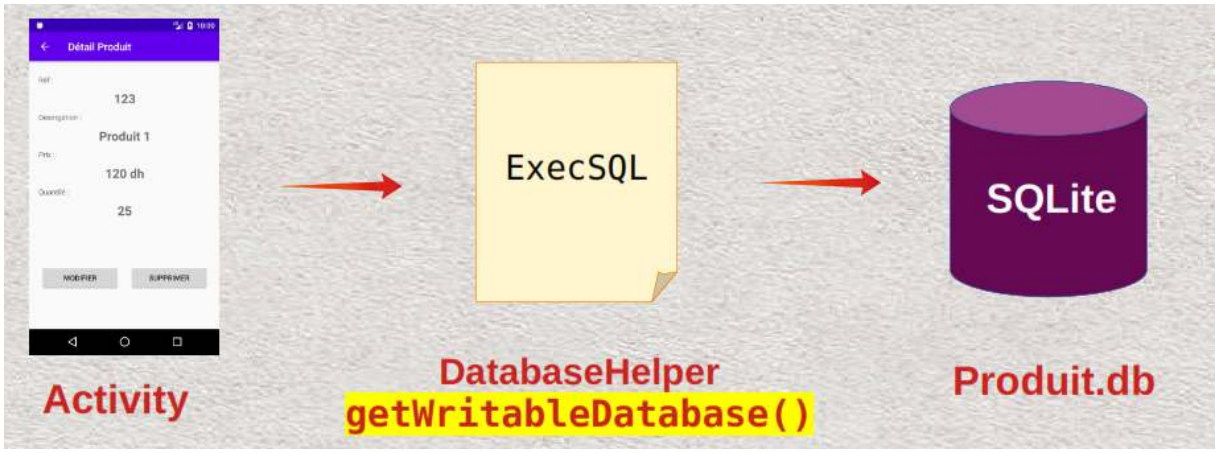
Voir les figures ci-dessus afin de voir la différence entre le mode Ajout ou Modification correspondant au même layout ajoutProduit

Réaliser les modifications nécessaire (Title, Bouton Enregistrer/MAJ, Ref disabled pour la modification)

	
Ajout Produit	Modifier un produit

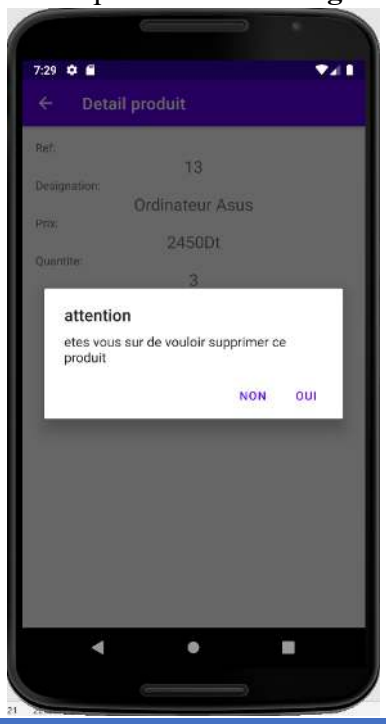
Pour la suppression

6.Suppression de données



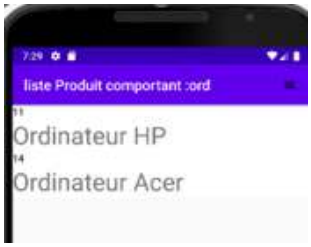
Le mode est getWritableDatabase et la requete delete avec le parametre column id

Dans notre exemple :
Supprimer le produit affiché actuel et **retourner** vers ma liste
Sans oublier de **verifier** la suppression par un **AlertDialog**



8. Recherche de données

La recherche sera selon la **désignation**



En cas de succès se rediriger vers la page `ListProduitsActiity` avec les produits recherchés (selon le critère de recherche : dans la figure cidessus les produits comportant « **ord** » dans leur désignation .
Sinon afficher un Toast avec un message « **Produit not found** »