Compte rendu application de todo list v3
Luca Jakomulski et Tristant Buscemi

Introduction
L'application est similaire à la deuxième version, à la différence notable que les listes sont enregistrées en local suite à leur consultation, ce qui permet à l'utilisateur d'y accéder même hors ligne.

Il est à noter que si une liste n'est pas parcourue au préalable par l'utilisateur quand celui-ci dispose d'une connexion, les listes ne seront pas enregistrées dans la base de donnée.

Analyse
Afin de réussir cette transformation, un nouveau package a été créé : le package database. Celui-ci contient des classes DatabaseEntity qui correspondent à la structure des données dans la base de données. Il contient également les interfaces dans lesquelles on va définir les différentes requête que l'on peut effectuer à la BDD, ainsi que les paramètres de ces dernières. Et enfin, il contient une classe qui va représenter la base de donnée dans le reste du code.

Exemple d'une classe DatabaseEntity

```java
@Entity
public class DatabaseItem {

    @PrimaryKey
    public long id;

    @ColumnInfo(name = "idList")
    public long idList;

    @ColumnInfo(name = "label")
    public String description;

    @ColumnInfo(name = "checked")
    public int done;
}
```

Et le DAO associé (les interfaces mentionnés plus haut)

```java
@Dao
public interface ItemDAO {

    @Query("SELECT * FROM databaseitem WHERE idList = :idList")
    List<DatabaseItem> getItems(long idList);

    @Query("UPDATE databaseitem SET checked = :checked WHERE id = :idItem")
    void updateStatus(int checked, long idItem);

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insertItems(List<DatabaseItem> itemList);

}
```

Maintenant, on peut modifier sereinement les classes activités. A chaque action de récupération des données, on va différentier le cas où l'on est connecté à internet ou non avec la méthode `connectedToNetwork()` codée lors de la version précédente. Si on est connecté à internet, on récupère les données et on les enregistre directement dans la base de donnée. Si on n'est pas connecté à internet, on récupère les données dans la base de données si ces dernières existent.

Les accès à la base de données sont exécutées dans un thread différents du main thread.

Enfin on a créé une classe Convert qui va nous permettre de convertir les données reçu par l'API à des données compréhensible par Room et inversement.

Exemple de récupération de données dans la base de données :

```
(Thread) run() → {
        mDb = Room.databaseBuilder(getApplicationContext(), AppDatabase.class,  name: "db").fallbackToDestructiveMigration().build();
        List<DatabaseTodoList> databaseTodoList = mDb.TodoListDAO().getTodoLists(mIdUser);

        if (databaseTodoList != null) {
            Convert c = new Convert();
            mListTodoList.addAll(c.todoListListDB(databaseTodoList));
            mListsAdapter.notifyDataSetChanged();
        } else {
            Message message = mHandler.obtainMessage();
            message.sendToTarget();
        }
}.start();
```

Exemple d'écriture de données dans la base de données suite à leur récupération avec l'API :

```
public void onResponse(Call<ResponseTodoLists> call, final Response<ResponseTodoLists> response) {
    if (response.isSuccessful() && response.body().success) {
        mListTodoList.addAll(response.body().lists);
        mListsAdapter.notifyDataSetChanged();

        // on enregistre les données en local
        (Thread) run() → {
                mDb = Room.databaseBuilder(getApplicationContext(), AppDatabase.class,  name: "db").fallbackToDestructiveMigration().build();
                Convert c = new Convert();
                mDb.TodoListDAO().insertTodoLists(c.todoListList(response.body(), mIdUser));
        }.start();
```

Conclusion
On a réussi à mettre en place une BDD accessible avec Room

Perspective
La modification du statut d'un item (check ou uncheck) n'a pas été mis en place. En effet, il est aisé d'enregistrer une modification de statut, mais au moment d'envoyer ces modifications via l'API il faut être capable de savoir quels items ont été modifiés afin de faire les requêtes correspondantes. Cependant, suite à un manque de temps ces modifications n'ont pas été misesn en place.

Sitographie
Slack et Android Developper et surtout on s'est inspiré du code vu en cours