Implémentation d’une base de données SQLite pour du stockage local des données.

**Adapter folder:**

Ce dossier contient un RecyclerView appelé PostAdapter qui lie les données dans une ListView. À partir de cette classe, les utilisateurs peuvent liker/deliker un article.

Un RecyclerView facilite l'affichage de grands ensembles de données. On fournit les données et définit l'apparence de chaque élément, et la bibliothèque RecyclerView crée dynamiquement les éléments lorsqu'ils sont nécessaires.

Comme son nom l'indique, RecyclerView recycle ces éléments individuellement. Lorsqu'un élément défile hors de l'écran, RecyclerView ne détruit pas sa vue. Au lieu de cela, RecyclerView réutilise la vue pour les nouveaux éléments qui ont défilé à l'écran. Cette réutilisation améliore considérablement les performances, améliore la réactivité de l’application et réduit la consommation de batterie.

**Data folder :**

Ce dossier contient 3 appels java appelés **DbConstant.java**, **DbController.java** et **DbHelper.java**. Ces 3 fichiers permettent le contrôle de la base de données SQLite.

Dans le fichier **DbConstant.java**, j'ai déclaré toutes les variables constantes telles que le nom de la table, le nom de la colonne, etc.

Dans le fichier **DbHelper.java**, on crée une base de données SQLite et une table avec le nom de colonne requis.

Dans le fichier **DbController.java**, j'ai ajouté une fonction pour stocker les données dans la base de données, obtenir toutes les données de la table et les fonctions ajouter / supprimer.

**Fragment folder :**

Un fragment est un composant Android indépendant qui peut être utilisé par une activité. Un fragment encapsule des fonctionnalités afin qu'il soit plus facile de les réutiliser dans les activités et les mises en page. Un fragment s'exécute dans le contexte d'une activité, mais a son propre cycle de vie et généralement sa propre interface utilisateur.

J'utilise ici des fragments car j’ai implémenté la bottom navigation pour les éléments de menu. Lorsque je clique sur chaque élément de menu, il remplace simplement le fragment.

Le dossier fragment contient 4 fragments :

● **HomeFragment** - Ce fragment contient un RecyclerView qui est chargé d'afficher les données des postes dans une liste de la base de données SQLite. J'ai ajouté PostAdapter ici pour lier les données de publication avec le RecyclerView. Il y a aussi une fonction appelée **loadData** qui permet d’obtenir toutes les données de la base de données.

● **AddPostFragment** - Ce fragment est responsable du stockage des données dans la base de données. J'ai ajouté 2 EditText et un bouton pour ce faire. Les utilisateurs entreront le titre et le contenu de l'article et, lorsqu'ils appuient sur le bouton, cela envoi les données dans la base de données.

● **LikeFragment** - Dans ce fragment, j'ai implémenté une sorte de page d'accueil de type RecyclerView qui affiche uniquement les données likées. Les autres fonctionnalités sont exactement les mêmes que celles du **HomeFragment**. Je dois charger les données de la base de données en fonction de si les articles sont likés ou non.

● **UserFragment** - Ce fragment montre l'interface utilisateur.

**Listener Folder** :

Dans ce dossier, j'ai créé une interface qui nous aide à interagir entre **PostAdapter**, **HomeFragment** et **LikeFragment**. Lorsque nous cliquons sur l'icône du cœur, cela déclenchera la mise à jour de la base de données.

**Models Folder :**

Dans ce dossier, j'ai créé un modèle appelé **UserPostModel**, ce fichier a des variables qui doivent être stockées dans une base de données, il contient un constructeur et une méthode getter setter (méthode de mutation). Cette méthode permet de lier les données de la base de données avec RecyclerView.

**Utility folder :**

Ce dossier contient un fichier appelé **Apputility.java**. J'ai ajouté deux fonctions appelées **getCurrentTime** et **getUniqueId**. La fonction **getCurrentTime** renvoie l'heure système actuelle qu’on utilise pour stocker dans la colonne created\_at de la base de données. La méthode getUniqueId renvoie un numéro unique aléatoire à 16 chiffres que j'ai utilisé en tant que PostId.

**MainActivity.java :**

C'est l'activité principale qui s'ouvre pour la première fois au lancement de l'application. Cette classe contient la bottom navigation, la gestion des fragments. Lorsqu’on clique sur l'élément de la bottom navigation, cette classe est responsable de la modification du fragment.

**Drawable folder :**

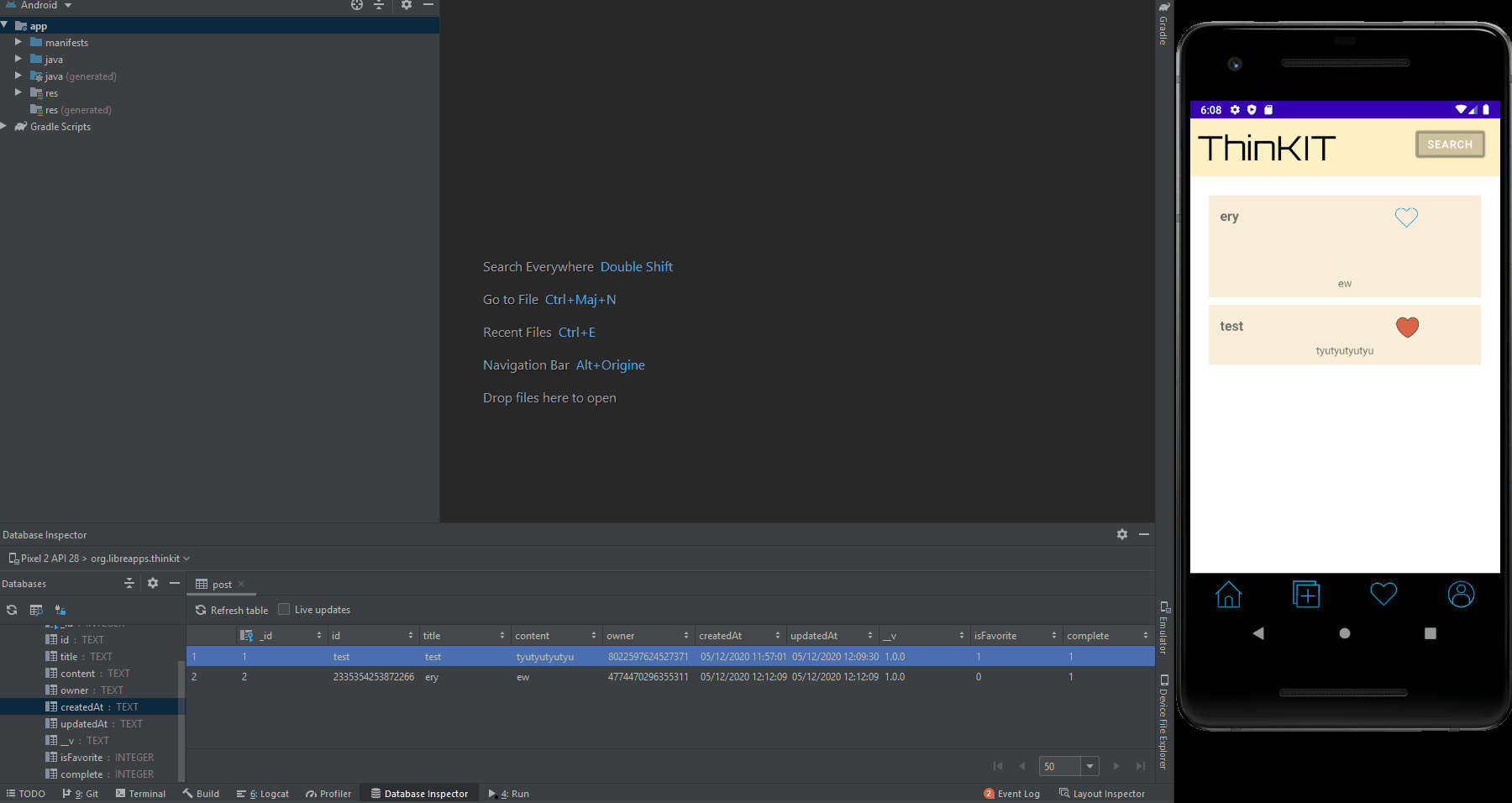
Ce dossier contient toutes les images, icônes et données graphiques.

* **Font folder** : Ce dossier contient une police personnalisée.
* **Layout folder** : Ce dossier contient toute l'interface utilisateur.
* **Menu folder** : Ce dossier contient l'élément du menu du bas.
* **Mimap folder** : Ce dossier contient une icône de lancement, une image, etc.
* **Values folder** : Ce dossier contient un fichier de style, un fichier content des chaines de valeurs, un fichier contenant toutes les couleurs liées à cette application.

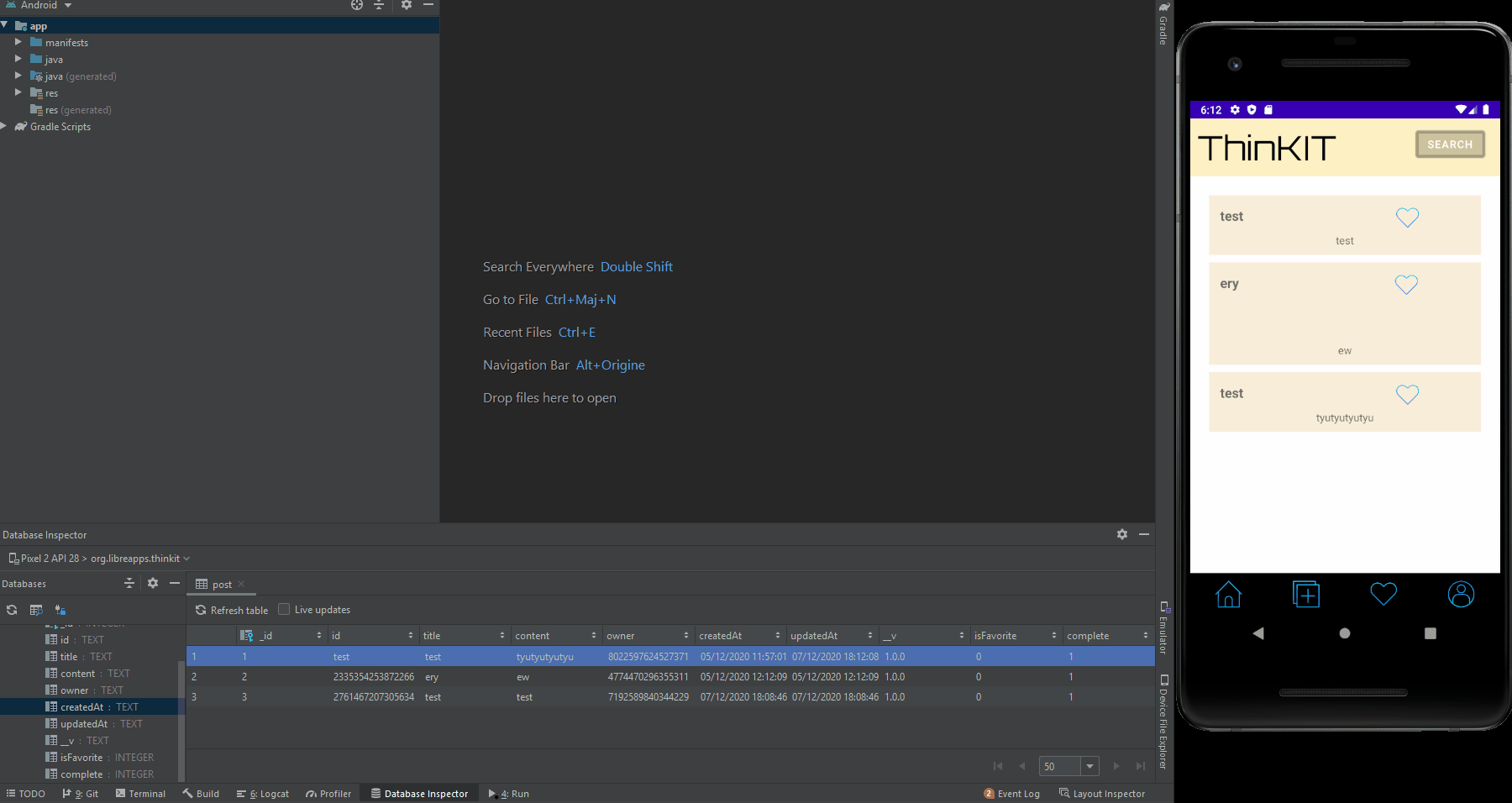
**Demo base de données SQLite**

Il existe un outil intégré à Android Studio appelé Database Inspector permettant d’inspecter, modifier et faire des requêtes avec les bases de données de l’application qui est en train de run.

**Création d’un article dans l’application :**



**Like d’un article**

****

**Modifications de la database**

