Kévin Trinh
Vincent Lam

Projet “GeoPhoto” dans le cadre du cours Développement Mobile sur Android

Projet “GeoPhoto” dans le cadre du cours Développement Mobile sur Android..... 1
1) Fonctionnalités implémentées.....2
TLTR :.....2
Détails :..... 2
Première fonctionnalité bonus :.....2
Base de donnée interne (SQLite) et deuxième fonctionnalité bonus :..... 3
Base de données externe (MySQL) :..... 3
Traitement d'image (Bonus) :..... 3
Gestion des erreurs & Logging :.....3
2) Fonctionnalités non implémentées.....4
3) Améliorations possibles..... 4
4) Structure de l'application, implémentation et captures d'écran.....4
Vue d’ensemble :.....4
Détails sur l’implémentation :..... 5
MainActivity.java :.....5
GalleryActivity.java :..... 5
PhotoActivity.java :..... 6
MapActivity.java :.....6
database/ImageContract.java & database/ImageDbHelper.java :.....7
Mises en page (fichiers .xml) :.....7
AndroidManifest.xml :..... 7
5) README.md et captures d'écran :..... 7

Kévin Trinh
Vincent Lam

1) Fonctionnalités implémentées

TLTR :

Pour faire court, on a implémenté toutes les fonctionnalités et deux fonctionnalités bonus. Je vous invite à aller dans la section README.md à la fin du rapport pour avoir une vue d'ensemble sur le projet en image.

Détails :

On navigue entre les sections principales de l'application (Galerie, Photo, Carte) via une barre de navigation inférieure dans MainActivity.

Il y a trois écrans : Galerie, Photo, Carte.

Le texte sur l'interface s'adapte à la langue de l'appareil (Français et Anglais via strings.xml et values-fr/strings.xml).

Les mises en page s'adaptent en fonction de l'orientation portrait ou paysage.

On a implémenté la fonctionnalité géolocalisation récupère la latitude et la longitude actuelles (MapActivity).

On peut afficher de la position actuelle avec un marqueur sur Google Maps (MapActivity).

Il y a la fonctionnalité de géocodage. On peut convertir des coordonnées de latitude/longitude en une adresse postale lisible, affichée comme titre du marqueur sur la carte (MapActivity).

On peut capturer des photos à l'aide de l'application caméra de l'appareil (PhotoActivity).

La photo s'affiche sur la page (PhotoActivity).

En cliquant sur le bouton "enregistrer", la photo capturée est sauvegardée dans la galerie dans le téléphone (PhotoActivity).

Première fonctionnalité bonus :

En fonctionnalité bonus, l'application est capable de convertir de Bitmap vers chaîne Base64 et vice-versa, en affichant les résultats (PhotoActivity).

Kévin Trinh
Vincent Lam

Base de donnée interne (SQLite) et deuxième fonctionnalité bonus :

On peut sélectionner une photo depuis la galerie du téléphone (GalleryActivity).

La photo sélectionnée s'affiche sur la page (GalleryActivity).

Ensuite, on peut sauvegarder le titre des images et des données d'image (sous forme de chaîne Base64) dans une base de données SQLite locale (GalleryActivity).

Puis, on peut récupérer et afficher (sous forme textuelle) une liste de tous les détails des photos stockées depuis la base de données SQLite (GalleryActivity).

Et enfin, on efface des champs de saisie et de l'aperçu de l'image après un enregistrement réussi dans SQLite (GalleryActivity).

Base de données externe (MySQL) :

- Stockage de la position actuelle de l'appareil (latitude et longitude) dans une base de données MySQL externe (MapActivity).
- Récupération et affichage de la dernière position enregistrée depuis la base de données MySQL sur la carte (MapActivity).
- Centrage de la vue de la carte sur la dernière position enregistrée récupérée de MySQL (MapActivity).
- Tentative de connexion initiale à MySQL depuis MainActivity avec retour d'information à l'utilisateur (Toast).

Traitement d'image (Bonus) :

- Conversion d'images Bitmap en chaînes de caractères encodées en Base64 (PhotoActivity, GalleryActivity).
- Conversion de chaînes encodées en Base64 en images Bitmap pour affichage (PhotoActivity).

Gestion des erreurs & Logging :

- Retours utilisateurs via des Toasts pour des opérations telles que les sauvegardes réussies, les erreurs, le refus de permission.
- Logging et gestion de traces (Log.d, Log.e, Log.i) pour le débogage, les erreurs et les informations de l'application.

Kévin Trinh
Vincent Lam

2) Fonctionnalités non implémentées

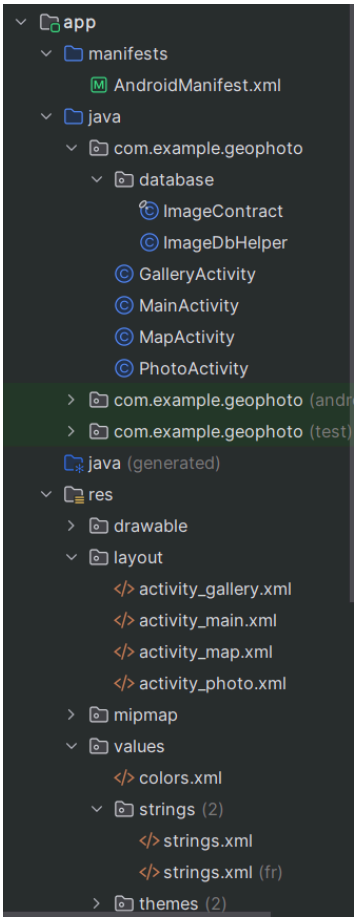
On a implémenté toutes les fonctionnalités demandées.

3) Améliorations possibles

- Pour la base de données MySQL externe, implémenter une API REST pour servir d'intermédiaire entre l'application et la base de données.
- Implémenter des tests unitaires pour la logique telle que la conversion Base64, les interactions avec la base de données (en utilisant Robolectric ou des tests instrumentés pour SQLite).
- Implémenter des tests d'interface utilisateur en utilisant Espresso.

4) Structure de l'application, implémentation et captures d'écran

Vue d'ensemble :



Kévin Trinh
Vincent Lam

ImageDbHelper et ImageContract gèrent le schéma et l'accès à la base de données SQLite locale.

Des appels JDBC dans MapActivity et MainActivity gèrent l'interaction avec la base de données MySQL distante.

On utilise MediaStore pour accéder à la galerie et enregistrer des photos.

Une grande partie de la logique applicative (gestion des clics de bouton, traitement des données, initiation des opérations de base de données, conversion d'images) se trouve dans les Activités.

L'application est structurée autour de multiples composants Activity, chacun responsable d'un écran distinct ou d'une fonctionnalité principale. La navigation entre les Activités est gérée via des Intents.

Les tâches de longue durée (accès base de données, appels réseau pour MySQL) sont effectuées sur des threads d'arrière-plan en utilisant ExecutorService ou new Thread() pour éviter de bloquer le thread UI. Les mises à jour de l'interface utilisateur depuis les threads d'arrière-plan sont gérées à l'aide de runOnUiThread().

Détails sur l'implémentation :

MainActivity.java :

Il sert de point d'entrée et de hub de navigation.

Il utilise des Intent pour lancer GalleryActivity, PhotoActivity, et MapActivity.

On initie une connexion à la base de données MySQL sur un thread d'arrière-plan.

GalleryActivity.java :

On peut choisir une image depuis la galerie de l'appareil en utilisant Intent.ACTION_PICK et startActivityForResult().

On gère le résultat dans onActivityResult() pour obtenir l'URI de l'image sélectionnée.

On décode l'image sélectionnée en un Bitmap et l'affiche.

On peut saisir un titre pour l'image.

On enregistre l'image (convertie en chaîne Base64) et son titre dans une base de données SQLite en utilisant ImageDbHelper et ContentValues.

On charge et affiche (sous forme de texte) tous les enregistrements de photos depuis la base de données SQLite.

On utilise SQLiteOpenHelper pour la création et la gestion des versions de la base de données.

PhotoActivity.java :

On lance l'application caméra en utilisant MediaStore.ACTION_IMAGE_CAPTURE et FileProvider pour obtenir une URI pour enregistrer l'image.

On reçoit le chemin de l'image dans onActivityResult() et affiche la photo capturée.

On permet d'enregistrer la photo capturée dans la galerie du téléphone via MediaStore.Images.Media.insertImage().

En fonctionnalité bonus, l'application est capable de convertir de Bitmap vers chaîne Base64 et vice-versa, en affichant les résultats.

MapActivity.java :

On a implémenté Google Maps en utilisant SupportMapFragment et OnMapReadyCallback.

On demande et gère les permissions de localisation.

On utilise FusedLocationProviderClient pour obtenir la position actuelle/dernière.

On affiche la position sur la carte avec un marqueur.

On utilise Geocoder pour convertir les coordonnées en une adresse postale pour le marqueur.

On peut se connecter à une base de données MySQL distante via JDBC sur un thread en arrière-plan.

On sauvegarde la latitude et la longitude dans la base de données MySQL.

On récupère la dernière position enregistrée depuis MySQL et l'affiche/la centre sur la carte.

database/ImageContract.java & database/ImageDbHelper.java :

https://developer.android.com/training/data-storage/sqlite?hl=fr

En suivant la documentation, on a implémenté ImageContract et ImageDbHelper :

- ImageContract définit le schéma de la table SQLite (nom de la table, noms des colonnes).
- ImageDbHelper étend SQLiteOpenHelper pour gérer la création (SQL_CREATE_ENTRIES) et les montées de versions possibles de la base de données.

Mises en page (fichiers .xml) :

- Définies en utilisant LinearLayout.
- ScrollView est utilisé pour le contenu qui pourrait dépasser la hauteur de l'écran.
- Des éléments UI comme Button, ImageView, EditText, TextView sont utilisés.
- strings.xml et values-fr/strings.xml sont utilisés pour les chaînes de caractères.

AndroidManifest.xml :

- Déclare toutes les Activités.
- Déclare les permissions nécessaires (INTERNET, ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION, READ_EXTERNAL_STORAGE/READ_MEDIA_IMAGES).
- Déclare le FileProvider.
- Configure la clé API Google Maps.

5) README.md et captures d'écran :

On a fait des gifs au fur et à mesure du développement de l'application. Je vous invite à cliquer sur les liens pour voir le contenu. Vous pouvez également explorer notre README.md.

- Bonus Feature : Find Photo From Gallery

https://github.com/user-attachments/assets/8f1cb838-7a2c-45cb-932c-fba2f127baf1

- Take Photo
- Display Photo
- Save Photo

Kévin Trinh
Vincent Lam

https://github.com/user-attachments/assets/f8151e6e-093b-4f1f-b20d-1c97501ed92c

- Bonus Feature : ByteArray in Base64 and vice versa

https://github.com/user-attachments/assets/b1beaf3b-7aed-4f10-b130-1e71bbcb5217

- Feature : save image in SQLite

https://github.com/user-attachments/assets/44699216-bd09-4174-ad24-de0e6ae806f3

- Feature : save in MySQL and display location from the database

https://github.com/user-attachments/assets/7e49b426-c8be-41ec-b3fc-1a4ac3b7cbc5

- Feature : focus back to last location

https://github.com/user-attachments/assets/39573e2b-5d8e-4e48-9098-99e2d447201f

- Feature : language changes according to Region Preferences fr eng

https://github.com/user-attachments/assets/62f88db7-ad1c-4f93-b1b1-587a8aa9a8b4