INFO 306 : programmation mobile

Supervisé par M. Wilhelm

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Project PhotoMapper

Bourdier Clément
Debled Manon
S5O5

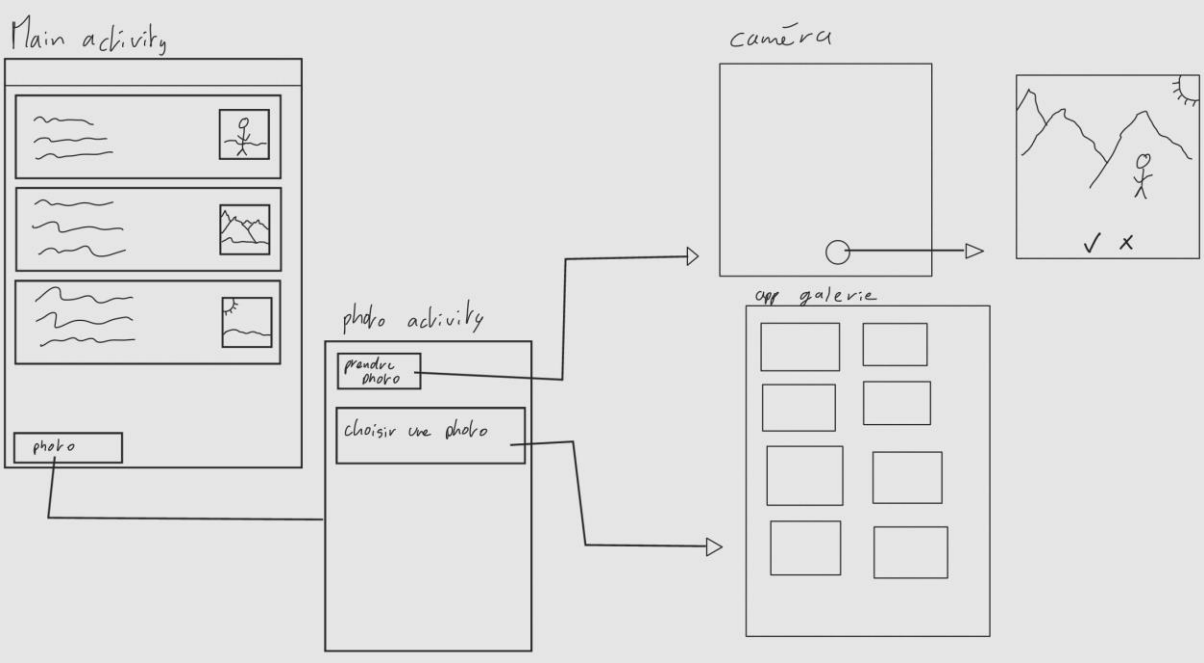
Table des matières

Présentation du projet

Dans le cadre du cours info306, nous avons réalisé ce projet. Il a été demandé de produire une application Android. Le but de l'application est ouvert, le plus important étant la prise en main d'Android Studio au travers de l'utilisation des différentes fonctionnalités disponibles.

Nous nous sommes fixés pour objectif de faire une application appelée PhotoMapper. Elle a pour but d'organiser des photos par voyage et de pouvoir retrouver ou elles ont été prises. Elle permettra de prendre des photos directement depuis l'application ou de les importer depuis la galerie en indiquant à quel dossier elle appartient. Il est ensuite possible d'afficher les photos de chaque voyage pour se « remémorer des souvenirs ».

Diagramme de navigation



Les fonctionnalités

Nous donnons ici un rapide aperçu des fonctionnalités de l'application, l'aspect technique et l'implémentation seront détaillés dans une autre partie.

Les capteurs

Pour son bon fonctionnement l'application a besoin d'accéder à deux capteurs : l'appareil photo et le capteur GPS. On ne gère pas le fonctionnement des capteurs on récupère uniquement les données avec des « intents » c'est-à-dire des objets qui établissent la communication entre les composants d'applications Android.

La géolocalisation

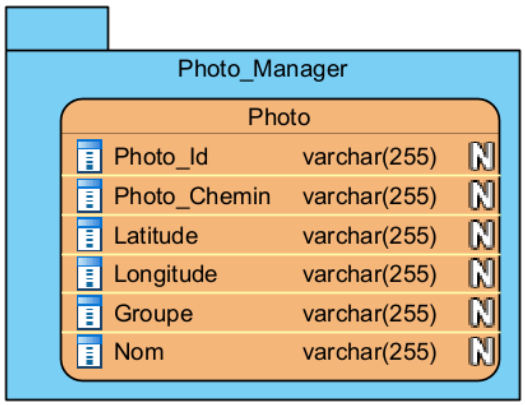
A la première ouverture de l'application il est demandé à l'utilisateur d'autoriser l'accès à la géolocalisation précise de l'appareil. Par la suite, l'accès est réalisé automatiquement lorsque l'application en a besoin, on récupère alors la latitude et la longitude.

L'appareil photo

L'application permet de prendre une photo ou d'aller en chercher une dans la galerie. Avec la prise de photo intégrée, l'ajout de la localisation se fait automatiquement contrairement à l'ajout d'une photo depuis la galerie. La photo peut aussi bien être prise avec la caméra frontale que la caméra arrière. Il est possible de choisir un nom est un groupe qui seront associés à la prochaine prise de vue.

La base de données

La base de données SQLite a pour but de pouvoir rechercher les photos et de retrouver la géolocalisation ainsi que le groupe de photos auquel elle appartient.
Elle a pour but de stocker le chemin de toutes les photos prise dans l'application en y associant un nom, un groupe et la géolocalisation. On implémente la classe « SQLite open helper » qui permet de gérer une base de données SQLite. On en hérite dans une classe que l'on créer nous-même et qui permet de gérer notre base de données c'est à dire les instances de photo manipulées dans l'application.



MCD SQLite 1

L'internationalisation

L'application est internationalisée ainsi elle est en deux langues : anglais et français. La langue est celle qui correspond au langage de l'appareil. On utilise différentes ressources appelées en fonction de la langue.

L'accès au stockage

En plus des photos prises depuis l'application on utilise aussi les photos de la galerie c'est pourquoi l'application a besoin de l'autorisation pour accéder au stockage. Cet accès est

également nécessaire au bon fonctionnement de la base de données afin de charger les photos et les afficher dans une liste.

L’affichage de liste

On utilise un fragment qui va rechercher dans la base de données toutes les photos souhaitées et les afficher ainsi que leurs informations.

Conception de l’application

Dans cette section nous décrivons la méthodologie, l’implémentation des fonctionnalités et les problèmes rencontrés lors de la conception de l’application. Il est aussi évoqué les choix techniques et architecturaux.

Méthodologie

Nous avons implémenté les fonctionnalités importantes de manière indépendantes les unes des autres avant de les regrouper. Voici, dans l’ordre, les étapes du développement de notre application :

Fragment

Nous avons d’abord utilisé une activité par défaut proposée par Android Studio, elle contient déjà deux fragments qui peuvent être affichés un à un dans l’activité. Nous avons ensuite modifié ces fragments pour afficher une liste de photos provenant de la base de données.

Géolocalisation

La géolocalisation a pris du temps à fonctionner. Il est nécessaire de faire appel à deux autorisations de position : la position faible précision, et la position haute précision.
Une fois la géolocalisation récupérée, nous avons pu nous attaquer à la prise de photo.

Prise de photo

Plusieurs manières existent pour prendre une photo dans une application Android, nous avons choisis d’utiliser l’intent «MediaStore.ACTION_IMAGE_CAPTURE ». Cela permet de lancer une capture de photo qui sera stockée sur l’appareil. Cet intent permet de récupérer son emplacement de stockage qui sera utile ensuite pour la base de données. Nous avons choisi de créer un objet photo pour pouvoir gérer les photos au sein de l’application.

SQLite

SQLite permet de gérer une petite base de données en local sans utiliser de serveur. Pour gérer une base de données SQLite, il existe une classe de base permettant de gérer tout plus facilement appelée SQLiteOpenHelper. À ce stade nous avons une position et une photo que nous pouvons stocker dans la base de données.

Liste de photos

Pour afficher le contenu de la base de données, nous affichions d'abord dans la console le contenu renvoyé. Nous avons ensuite affiché directement dans un fragment la liste de photos (nom, groupe, aperçu de l'image) contenu dans la BDD. Pour se faire nous avons construit dynamiquement son contenu en utilisant les évènements onCreateView() et onResume() du fragment. Il y avait beaucoup trop d'images à afficher, nous avons donc fait un affichage sous forme de pages, seul les 5 premières images sont affichées, il faut ensuite cliquer sur un bouton suivant pour nettoyer le fragment et le remplir par les 5 éléments suivants.

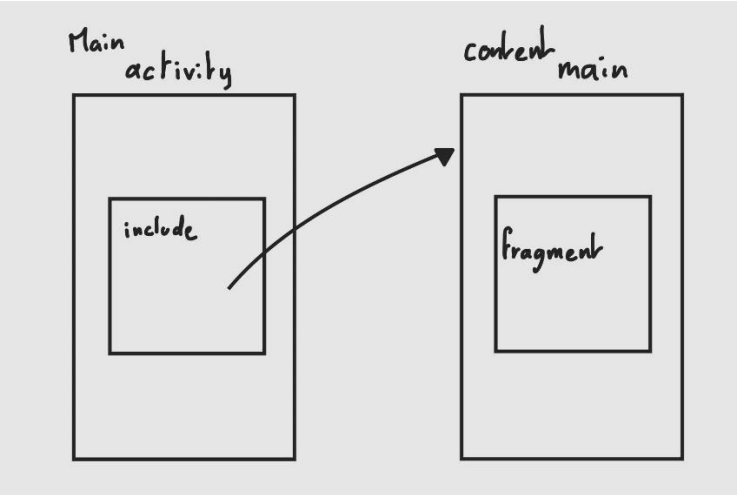
Implémentation

Dans cette section nous décrivons les détails techniques ainsi que les défis rencontrés.

Les fragments

Les fragments utilisés sont encapsulés dans un élément « include ». L'élément include dans Android Studio permet de réutiliser du code XML dans un autre fichier XML. Cela peut être utile pour créer des modèles de mise en page ou des composants réutilisables dans l'application.

Ces fragments utilisent un navgraph pour gérer la navigation entre eux. Le navgraph est un fichier xml qui définit la navigation entre les écrans de l'application. Avant de comprendre ce fonctionnement on a mis beaucoup de temps à comprendre pourquoi le fragment ne se comportait pas comme prévu.



Le manifest

Tous les senseurs pour fonctionner ont besoin des autorisations de la part du système Android. Il faut indiquer dans le manifest qu'on a besoin de ces autorisations. Le fichier AndroidManifest.xml est un fichier de configuration important dans une application Android. Il décrit les éléments suivants de l'application :

- Les permissions requises par l'application pour accéder aux fonctionnalités du système et aux données de l'appareil.
- Les activités, les services, les broadcast receivers et les content providers de l'application, ainsi que leur relation et leur hiérarchie.
- Les dépendances externes de l'application, comme les bibliothèques de code ou les services tiers.
- Les métadonnées de l'application, telles que son nom, sa version et son icône.

Les intents

Les Intents sont des objets utilisés dans Android pour représenter des requêtes d'action ou de données. Ils sont utilisés pour lancer des activités, des services, des broadcast receivers et pour effectuer d'autres actions dans l'application ou entre les applications
Il existe deux types d'Intents : les Intents explicites et les Intents implicites. Les Intents explicites sont utilisés pour lancer une activité ou un service spécifique en utilisant leur nom de classe. Les Intents implicites sont utilisés pour effectuer une action générale, sans spécifier l'activité ou le service qui doit être lancé. Lorsqu'un Intent implicite est envoyé, le système Android recherche les applications qui sont enregistrées pour gérer cet Intent et en lance une. Les Intents sont des éléments clé de l'architecture d'application Android et permettent de créer des applications modulaires et extensibles.

GPS

Pour la position on utilise le manager LocationManager. C'est une classe Android qui permet de gérer les services de localisation de l'appareil, tels que le GPS, le réseau cellulaire et le Wi-Fi. Elle offre des méthodes pour accéder à la position actuelle de l'appareil, ainsi que pour recevoir des notifications lorsque la position de l'appareil change. Il faut penser à déclarer la permission dans le manifest.

L’idée était ensuite d’afficher les photos sur une carte en les positionnant en fonction de leur latitude et longitude. Nous avons fait des essais avec la bibliothèque de carte Google Maps. Il fallait pour ça obtenir une clé d’API pour y accéder, mais l’opération s’est révélée trop compliquée et la bibliothèque aurait été trop limitée pour l’utilisation que nous voulions en faire. Nous n’avons pas eu le temps de construire notre propre système de carte qui aurait été basé sur une carte sous forme d’image, avec une épingle placée dessus pour chaque photo de la base de données.

Le test unitaire avec Junit

Les tests instrumentés sont des tests de code qui s'exécutent sur un appareil Android ou dans un émulateur, plutôt que dans un environnement de développement local comme un ordinateur de bureau. Ils sont utilisés pour tester les fonctionnalités de l'application qui nécessitent l'accès aux fonctionnalités du système Android, telles que l'accès aux données de l'appareil ou à la base de données SQLite.

Ici, le test crée une instance de l’objet Photo, la stocke dans la base de données, puis extrait cette photo depuis la base de données sous la forme d'une autre instance de Photo. On vérifie ensuite si les propriétés sont bien identiques à celle fournies initialement. Cela permet de tester la cohérence de la base de données. Nous avons eu plusieurs problèmes car nous passions par des tests classiques, mais il n’était pas possible de récupérer un contexte d’utilisation. Cela limitait beaucoup les tests possibles, la solution a été de passer à des tests instrumentés qui sont une spécificité d’Android Studio.

Conclusion

Il s’est avéré que la vision finale de l’application était un peu trop ambitieuse. Cependant, la majorité des fonctionnalités primaires de l’applications sont opérationnelles : Il est possible de prendre une photo qui sera stockée, il est possible de lier les photos à des groupes, on peut ensuite les consulter et connaitre la localisation liée à la photo. Il ne manque qu’une carte interactive indiquant les photos prises, et un affinage de l’interface pour pouvoir proposer quelque chose d’exploitable par tout le monde.

Ce projet nous a permis de mieux comprendre Android studio et comment produire une application native. L’utilisation des fragments est maintenant acquise, avec les subtilités des includes et des navgraphs. Nous savons maintenant comment construire un fragment à la volée. Nous savons aussi comment accéder aux différents capteurs de l’appareil, comme le GPS et l’appareil photo aux travers des « intent ».

La version actuelle de l’application PhotoMappeur est un bon prototype montrant toutes ces connaissances en pratique.