Réalisation d'une application Android

PROJET info0306

Sous la tutelle de Monsieur WILHELM Geoffrey

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Réalisé par

Madiane SAHLI et Raphaël GAREZ

Sommaire

Introduction

L'application PiggyBank est une application Android qui vous permettra de gérer toutes vos dépenses et ce pour une meilleure éducation financière.

Vous pouvez créer des tirelires et y ajouter votre budget du mois. Ainsi, au fil du mois vous pouvez ajouter vos dépenses et vos revenus afin de gérer vos entrées et sorties d'argent à court, moyen et long terme! Les utilisateurs de PiggyBank peuvent créer autant de tirelire et de dépenses qu'ils le souhaitent, sans aucune contrainte ni limites. PiggyBank est une application accessible par tous.

Facile d'utilisation, chaque utilisateur peut gérer son budget en quelques secondes. Fini les tickets de caisses et les révélés bancaires, il y a maintenant PiggyBank.

Prérequis

Voici quelques informations nécessaires à savoir avant d'utiliser PiggyBank :

L'application est à télécharger sur le GitLab :

- **Lien de téléchargement GitLab :**

https://gitlab-mi.univ-reims.fr/gare0008/info0306

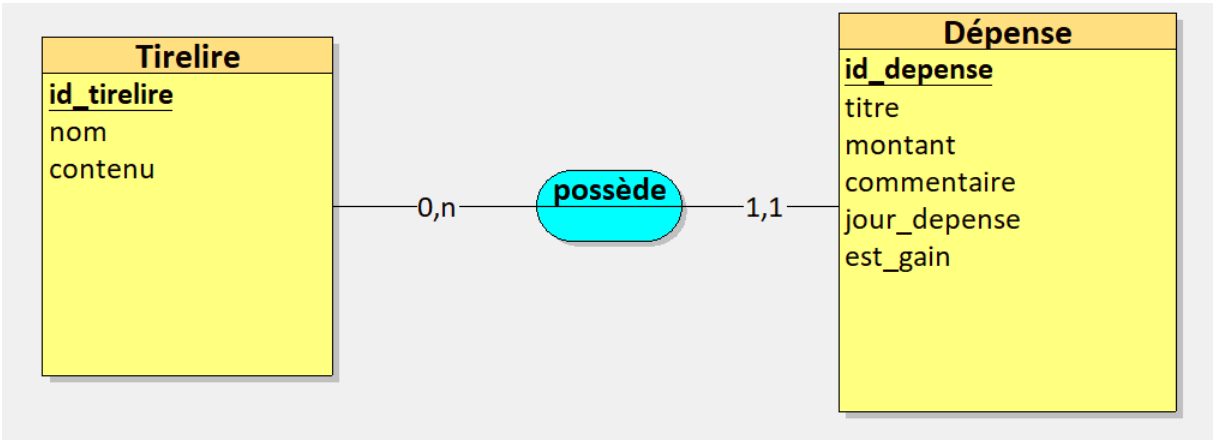
Cahier des charges

Pour commencer, lors de la réception des sujets, nous avions peu d'idée concernant la création d'une application mais après un démarrage traînant, nous avons décider de mettre nos capacités en commun pour réaliser ce chef d'œuvre. Nous avons donc décidé d'axer notre application sur un large public. Nous voulions également créer une application que nous pourrions continuer à utiliser durant notre quotidien. Nous avons donc décider de choisir le thème de l'argent, moyen d'échange le plus fiable aux yeux de tous.

Avant de commencer le développement de l'application, nous avons dû réfléchir à la modélisation de ce dernier. En effet, nous avons réfléchis au modèle conceptuel de donnée (MCD) et au modèle logique de donnée (MLD).

Modèle conceptuel de donnée (M.C.D)

Le MCD est une représentation graphique de haut niveau qui permet facilement et simplement de comprendre comment les différents éléments sont liés entre eux à l'aide de diagrammes codifiés.



Modèle logique de donnée (M.L.D)

Le MLD est un modèle relationnel qui utilise des tables mais pas d'associations, et est convertit de la forme de MCD pour afficher différentes clés primaires et secondaires.

Architecture logicielle

Pour commencer, afin de réaliser l'architecture de PiggyBank nous avons utiliser le modèle MVC.

Le modèle MVC (Model-View-Controller) est un modèle de conception logicielle qui sépare les données de l'application (modèle), l'interface utilisateur (vue) et la logique de contrôle (contrôleur). Le modèle MVC permet de séparer les différentes parties d'une application afin de faciliter le développement et la maintenance du code.

Voici comment cela fonctionne dans Android Studio :

- Le modèle représente les données de l'application et les règles métier qui régissent ces données. Il s'agit souvent de classes Java qui représentent des objets métier, comme des utilisateurs, des produits, etc. Le modèle ne se préoccupe pas de la façon dont les données sont présentées ou de la façon dont l'utilisateur interagit avec elles.
- La vue représente l'interface utilisateur de l'application, c'est-à-dire la façon dont les données du modèle sont présentées à l'utilisateur et comment l'utilisateur peut interagir avec elles. En Android, les vues sont généralement des layouts XML qui décrivent la structure de l'interface utilisateur et des éléments de contrôle tels que des boutons, des champs de texte, etc.
- Le contrôleur gère la logique de l'application et agit comme un intermédiaire entre le modèle et la vue. Lorsque l'utilisateur interagit avec la vue, le contrôleur réagit en modifiant le modèle ou en mettant à jour la vue en conséquence. En Android, le contrôleur est généralement représenté par une activité ou un fragment, qui contient du code Java qui gère la logique de l'application.

En utilisant le modèle MVC, nous pouvons séparer les différentes parties de notre application et travailler sur chacune d'entre elles indépendamment les unes des autres. Cela peut nous aider à rendre notre code plus facile à maintenir et à développer.

Concernant la page principale, nous avons utilisé un adaptateur pour un affichage dynamique. L'utilisation d'un adaptateur permet de séparer les données de l'application de la façon dont elles sont affichées à l'utilisateur. Cela rend le code plus modulaire et facilite la

maintenance et le développement de l'application. De plus, l'utilisation d'un adaptateur permet de réutiliser la même vue pour afficher différents types de données, ce qui peut être utile lorsque vous avez plusieurs écrans qui affichent des données similaires dans l'application.

Un affichage dynamique est un type d'interface utilisateur qui met à jour les données affichées en temps réel en fonction des changements de l'application. Cela peut être utile lorsque l'on veut afficher des informations qui changent fréquemment, ou lorsqu'on veut afficher des données qui dépendent de l'état de l'application.

Choix des technologies utilisées

Notre code utilise diverses technologies telle que 'SQLiteOpenHelper', 'BaseAdapter', 'AppCompatActivity', 'Intent' et 'Bundle', pour créer l'interface utilisateur de l'application et gérer les interactions utilisateur. Il utilise également la classe 'Log' pour les journaux, la classe 'Toast' pour afficher des messages brefs à l'utilisateur et la classe 'ListView' pour afficher des listes d'éléments. On utilise un widget 'EditText' pour l'entrée de données, un widget 'Button' pour les actions de l'utilisateur et un widget 'ListView' pour afficher des listes d'éléments. On utilise également un 'TextWatcher' pour écouter les modifications du widget 'EditText'.

Concernant les méthodes utilisées, la méthode 'onCreate' est une méthode de la classe 'AppCompatActivity' qui est automatiquement appelée lorsque l'activité (ou l'écran) est créée. Dans cette méthode, le code définit le layout (la disposition) à utiliser pour l'activité en appelant la méthode 'setContentView' et en lui passant une référence au fichier de layout XML correspondant.

La méthode 'init' est une méthode personnalisée qui initialise les liens entre les widgets du layout et les variables de la classe. Elle utilise la méthode 'findViewById' pour récupérer une référence à chaque widget à partir de son identifiant défini dans le fichier de layout XML.

Notre code définit une variable de classe appelée 'contrôle' qui est une instance de la classe 'Contrôle'. Cette classe est une classe de contrôleur qui gère les interactions avec les données de l'application.

On utilise également la méthode 'setAdapter' de la liste 'main_listView_tirelire' pour définir un adaptateur (un objet qui permet de lier les données à la liste) pour la liste. L'adaptateur utilisé est obtenu en appelant la méthode 'getAdapterTirelire' de la classe 'Contrôle' et en lui passant une référence à l'activité en cours. La méthode 'getAdapterTirelire' retourne un adaptateur qui contient les noms des instances de la classe 'Tirelire' gérées par la classe 'Contrôle'.

Pour résumé, voilà quelques technologies et méthode que nous avons utilisé pour la création de PiggyBank.

Choix des méthodes de persistance de données

L'utilisation d'une méthode de persistance de données permet de stocker des informations de manière durable, même lorsque l'application est fermée ou que l'appareil est éteint.

Concernant PiggyBank, nous avons choisi d'utiliser une base de données SQLite.

SQLite est une bibliothèque de gestion de base de données relationnelle qui est intégrée à Android et qui permet de stocker des données de manière structurée. Nous avons utilisé SQLite pour plusieurs raisons :

1. Stockage de données structurées: SQLite permet de créer des tables et de définir des schémas de données pour stocker des données de manière structurée.
2. Requêtes SQL: SQLite utilise le langage de requête SQL pour interagir avec la base de données. Cela nous permet d'effectuer des opérations courantes de gestion de données, telles que l'insertion, la mise à jour et la suppression de données, ainsi que des requêtes plus complexes pour extraire des données de la base de données.
3. Facilité d'utilisation: SQLite est intégré à Android et est facile à utiliser.

En résumé, SQLite est une solution de stockage de données efficace et facile à utiliser pour les applications Android qui nécessitent un stockage de données structurées et des requêtes SQL.

Dans notre cas, notre classe « MySQLiteOpenHelper » hérite de la classe SQLiteOpenHelper et qui permet de gérer une base de données SQLite sur Android. Elle contient les propriétés creationTableTirelire et creationTableDepense, qui sont des chaînes de caractères contenant des requêtes SQL pour créer des tables tirelire et depense dans la base de données.

La classe MySQLiteOpenHelper définit également deux méthodes : onCreate() et onUpgrade(). La méthode onCreate() est appelée lorsque la base de données est créée pour la première fois. La méthode onUpgrade() est appelée lorsque la version de la base de données est modifiée.

Liste des fonctionnalités implémentées

Voila la liste des fonctionnalités implémentées dans PiggyBank :

- Interface utilisateur: Nous avons utilisé des layouts XML et des éléments de contrôle tels que des boutons, des champs de texte et des listes déroulantes pour créer l'interface utilisateur de notre application, l'application possède au moins 2 activités et elle est capable de s'adapter à l'orientation de l'écran que ce soit sur téléphone ou tablette. PiggyBank possède un menu dans la barre des statuts.
- Persistance de données: Nous avons utiliser des méthodes de persistance de données telle que SQLite comme expliqué précédemment.
- Notifications et service: Nous avons utilisé les notifications et les services pour informer l'utilisateur d'ajouter ses dépenses lors du premier lancement de l'application puis quotidiennement.
- Prise en charge de plusieurs langues : Nous avons utilisé des fichiers de ressources et des classes de la bibliothèque Android pour prendre en charge plusieurs langues (français et anglais).
- Style de l'application : PiggyBank suit au maximum les conventions et bonnes pratiques de style de programmation pour assurer une qualité et une cohérence dans le développement de l'application.
- Intentions: Nous utilisons des 'Intent' lors de chaque création d'une tirelire. Ces intentions interagissent directement avec la base de données de notre application afin de créer modifier ou encore supprimer des données d'une tirelire ou d'une dépense.
- Boite d'alerte de dialogue : Une boîte de dialogue d'alerte est un type de boîte de dialogue qui s'affiche sur l'écran de l'utilisateur et qui demande une confirmation ou une action de sa part. Dans notre cas nous en utilisons pour plusieurs actions tel que la modification du nom d'une tirelire, ou encore lorsque l'on demande à l'utilisateur s'il souhaite vraiment supprimer une tirelire.

Pour résumé, notre application possède de multiples fonctionnalités inscrites sur le sujet tel que l'intégration de plusieurs activité, l'inclusion d'une base de données et bien plus encore.

Problème rencontré

La réalisation de ce projet a été précipité. Nous avons rencontré de nombreux problèmes de taille qui nous ont fait perdre beaucoup de temps. A chaque difficulté, il nous a fallu tester et chercher des solutions par nous-même.

Parfois la réalisation de l'application peut différer de ce que nous avions imaginé au démarrage. Nous étions parfois dans l'obligation de faire marche arrière sur certaines fonctionnalités ce qui nous a fait perdre pas mal de temps.

Nous avions prévu de donner la possibilité à l'utilisateur d'associer une image qui provient de sa galerie ou de son appareil photo. Néanmoins, la gestion des images avec SQLite nous a posé de gros problème et nous avons donc dû renoncer à cette idée.

Le travail en binôme ne s'est pas déroulé comme prévu, nous avons dû nous adapter aux codes de l'un et l'autre et cela ne s'est pas fait sans efforts des deux parties.

Regret

Malheureusement, il y a de nombreuses fonctionnalités que nous aurions voulu mettre en œuvre, mais en raison de la réalisation tardive et de la difficulté de ces dernières, elles n'ont pas été créé.

Conclusion

Pour conclure, tout au long de la réalisation de notre application, nous avons essayé de mettre en pratique les connaissances acquises durant notre cursus universitaire.

Au cours de la création de PiggyBank, nous avons étudier et implémenté les différentes notions vues au cours des TPs.

En effet, nous ne nous rendions pas compte de la complexité de la réalisation d'un tel projet. Cependant, c'est notre première application avec Android Studio et nous sommes assez fiers du résultat. Bien entendu, l'application n'est pas parfaite mais elle est fonctionnelle et nous pensons que c'est le plus important. Etant donné que Raphael a un téléphone Android, nous pensons que l'utilisation de PiggyBank va être un réel plaisir au quotidien ainsi il va pouvoir gérer ses comptes de la meilleure des manières possibles.

Finalement, nous avons pris plus de plaisir à réaliser l'application que ce que nous avions imaginé durant les TPs.

Webographie

https://cours.univ-reims.fr/course/view.php?id=819

https://stackoverflow.com/

https://www.youtube.com/watch?v=TCJFoExOBUE&list=PLBNheBxhHLQxmCCiHGkXBAIsC1VKpZkSe