Rapport de projet tuteuré Android

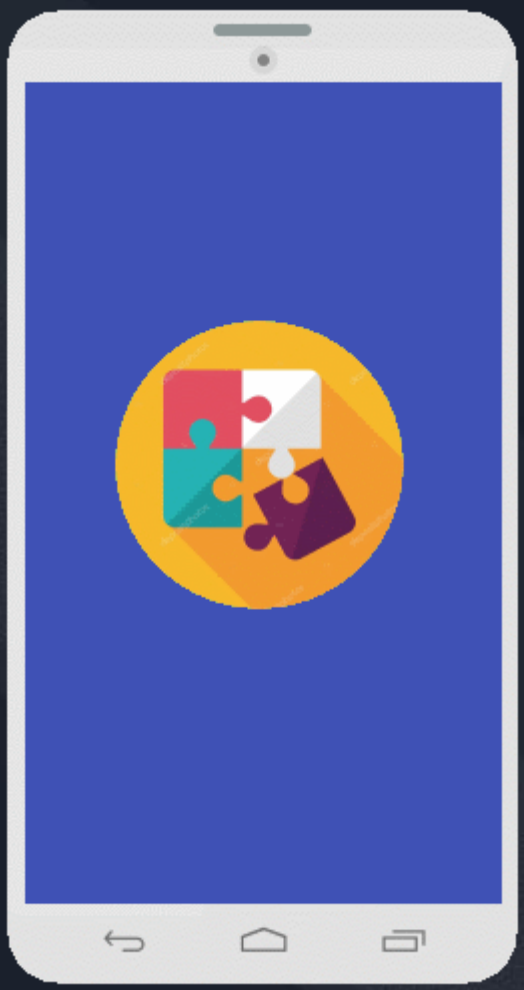
Licence Professionnelle IEM

Année 2017-2018

Application : ComAutis

Client : François Tavel

Par Paul Gronier, Julien Hivert, Antoine Cervo



Google Play

IUT Lyon 1

l'excellence technologique

Introduction

Dans le cadre de notre premier projet tuteuré d'Android, nous avons choisis de travailler sur le projet ComAutis pendant nos 2 semaines, pour plusieurs raisons.

Tout d'abord, il s'adressait à un véritable client qui allait directement utiliser l'application après son développement.

Ensuite, parce qu'il nous a permis de pratiquer les concepts du codage propre en se focalisant uniquement sur du Java Android, sans avoir à passer du temps sur une API externe d'un autre langage.

Enfin, parce qu'il nous a fallu concevoir une architecture extensible et réutilisable pour les étudiants de l'année prochaine qui reprendront notre application pour y ajouter encore plus de fonctionnalités.

L'association ComAutis s'occupe d'enfant atteint d'autisme. Il est plus facile, pour un enfant autiste, de communiquer à l'aide d'images décrivant ses besoins, ses envies. L'association utilise des catalogues remplis d'image afin de dialoguer avec les enfants.

Cependant, ce système atteint vite ses limites, particulièrement lors de sortie avec les enfants. L'association souhaite donc pouvoir transporter ce catalogue, y avoir accès rapidement et facilement, au travers d'une application smartphone.

Table des matières

I – Expression du besoin

L'objectif de l'application est de pouvoir communiquer avec l'enfant en utilisant les images qui sont stockées dans le téléphone. De plus l'association souhaite pouvoir afficher des images pour des besoins précis, pour un enfant précis. L'utilisateur de l'application doit donc pouvoir ajouter un enfant dans l'application, puis lui créer une ou plusieurs pages qui lui seront spécialement dédiées. Ces pages contiendront alors les différentes images représentant des actions à réaliser par l'enfant.

Pour résumer, les fonctionnalités sont :

- Création d'un espace propre à un enfant
- Pour chaque enfant, pouvoir créer un scénario
- Rechercher des images stockées dans la mémoire du téléphone, et les disposer afin de renseigner un scénario
- Paramétrer un timer et le visualiser sur une page de scénario

Ceci est représenté par notre use case en annexe page 11

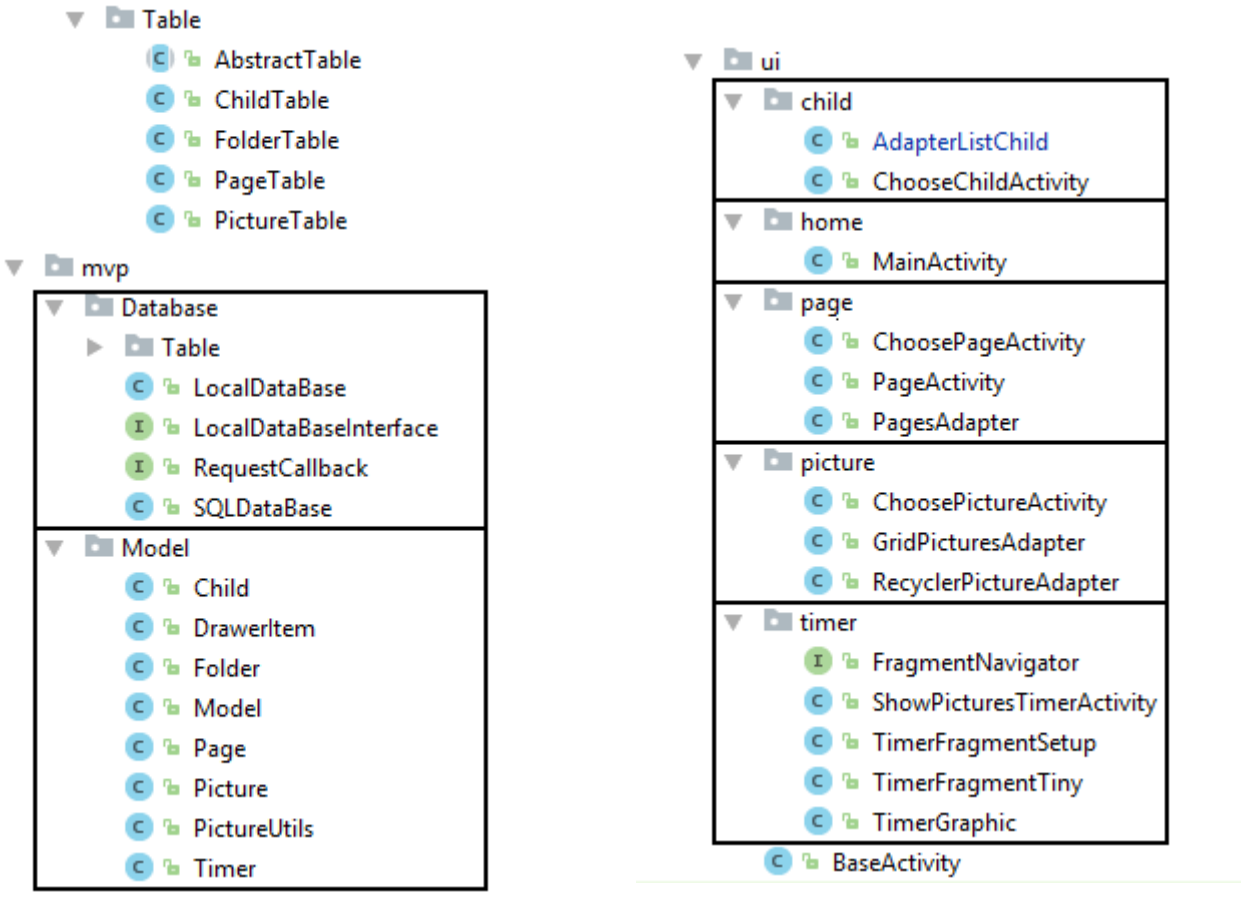
II – Ecriture du code

Après notre échange avec le représentant de l'association, nous avons définit l'ensemble des besoins auxquels il sera possible de répondre sur le délai du projet.

Pour commencer, nous avons écrit entièrement la nouvelle version application. Nous avons respectés les conventions de nommage CamelCase, que ce soit pour les différents packages, identifiants, etc. Sachant que notre code serait réutilisé, il nous a semblé très important de le rendre plus lisible et le plus maintenable possible. Dans cet esprit, nous avons découpés notre code en suivant l'organisation : Model – View – Presenter. Pour nos différentes Layout, nous avons utilisé les « constraint layout » qui sont beaucoup plus puissant que de simple recycler Layout. Les recycler layout ne garantissent pas exactement le même rendu sur tous les téléphones, il existe des différences entre la vue imaginée par l'équipe de design et la vue sur téléphone. Le Constraint layout permet de palier à ce problème, les vues deviennent beaucoup plus simples et beaucoup plus rapide à faire. Il suffit de placer des contraintes et des marges entre les éléments. Cf annexe page 8.

La base de données de l'application est stockée sur le téléphone de l'utilisateur, en utilisant la technologie SQLite. Nous avons créé différentes tables : Picture,Page, Chid, Folder. Chaque table hérite d'une classe abstraite « AbstractTable », et redéfinit les méthodes « getContentValue », « fromContentValue », « fromCursor », permettant d'interagir avec les données.

Notre package Model contient tous les objets utilisés dans le projet (Enfant, Page, Dossier, etc).



Pour visualiser la liste des enfants, nous avons utilisés un RecyclerView. La vue affiche la liste des enfants de la table ChildTable dans la base de données.

III – Nouvelles Fonctionnalités

1) Timer

L'association souhaitait avoir un timer à côté des images afin de pouvoir montrer en même temps aux enfants soit la durée d'une tache, soit le temps d'attente avant que l'activité commence.

Pour répondre à ce besoins, nous avons créé un compte à rebours paramétrable par l'utilisateur. Une activité avec 2 fragments a été implémentée : le premier fragment, permet de définir le temps. Le second fragment, affiche le temps restant à côté des photos. Afin que les 2 fragments puissent discuter nous avons mis en place une interface permettant ce dialogue. A chaque seconde, les deux fragments se mettent à jour. C'est le timer paramétrable qui va envoyer au second – dans l'activité page – le temps restant, lui permettant de se redessiner. Dans la première vue, le compte à rebours est affiché à l'aide d'une « progressBar ». Dans le deuxième fragment, nous avons utilisé des canevas afin de dessiner le compte à rebours à côté des images.

2) Recherche simplifiée d'images

Dans l'ancienne version de l'application, l'utilisateur devait allez chercher l'image en réalisant un grand nombre de clic sur l'écran et du temps. Nous avons réduit au maximum le nombre de click permettant d'accédant aux images. Nous avons de choisi de mettre en premier écran, après l'ouverture de l'application, la liste des images qui sont présente sur le téléphone de l'utilisateur dans le dossier « ComAutis ». De ce fait l'utilisateur accède directement aux images.

Nous avons développé un menu de navigation que nous avons placé en bas (Mais il n'est pas dans l'APK fournie pour le moment). Ce menu permet à l'utilisateur d'accéder à la liste des enfants et à leurs pages personnelles. De part ces choix nous avons considérablement réduit le nombre de clique sur l'application pour faire gagner du temps à l'utilisateur. Toujours dans un souci de gagner du temps, nous avons placé dans la Toolbar, un champ de recherche d'image. Les images présentes dans le dossier de l'application sont indexées, de ce fait l'utilisateur saisit une partie du mot, ou le mot entier, correspondant à l'image souhaitée. L'application recherche alors dans le dossier d'images, les images associées à la recherche de l'utilisateur.

Le dossier utilisé pour récupérer se trouve à la racine de la mémoire du téléphone et doit se nommer comAutis. Ainsi, toutes les photos se trouvant à l'intérieur du dossier seront accessibles. Il faut avoir créée le dossier avant de lancer l'application, sans quoi l'application ne le trouvera pas.

IV - Evolution de l'application

Nous aurions étés très heureux de pouvoir continuer ce projet afin d'ajouter plus de fonctionnalités. Cependant, pour une durée de deux semaines, il n'était pas possible de tout concevoir.

Voilà quelques pistes d'améliorations possibles :

- Porter le projet sur iOS

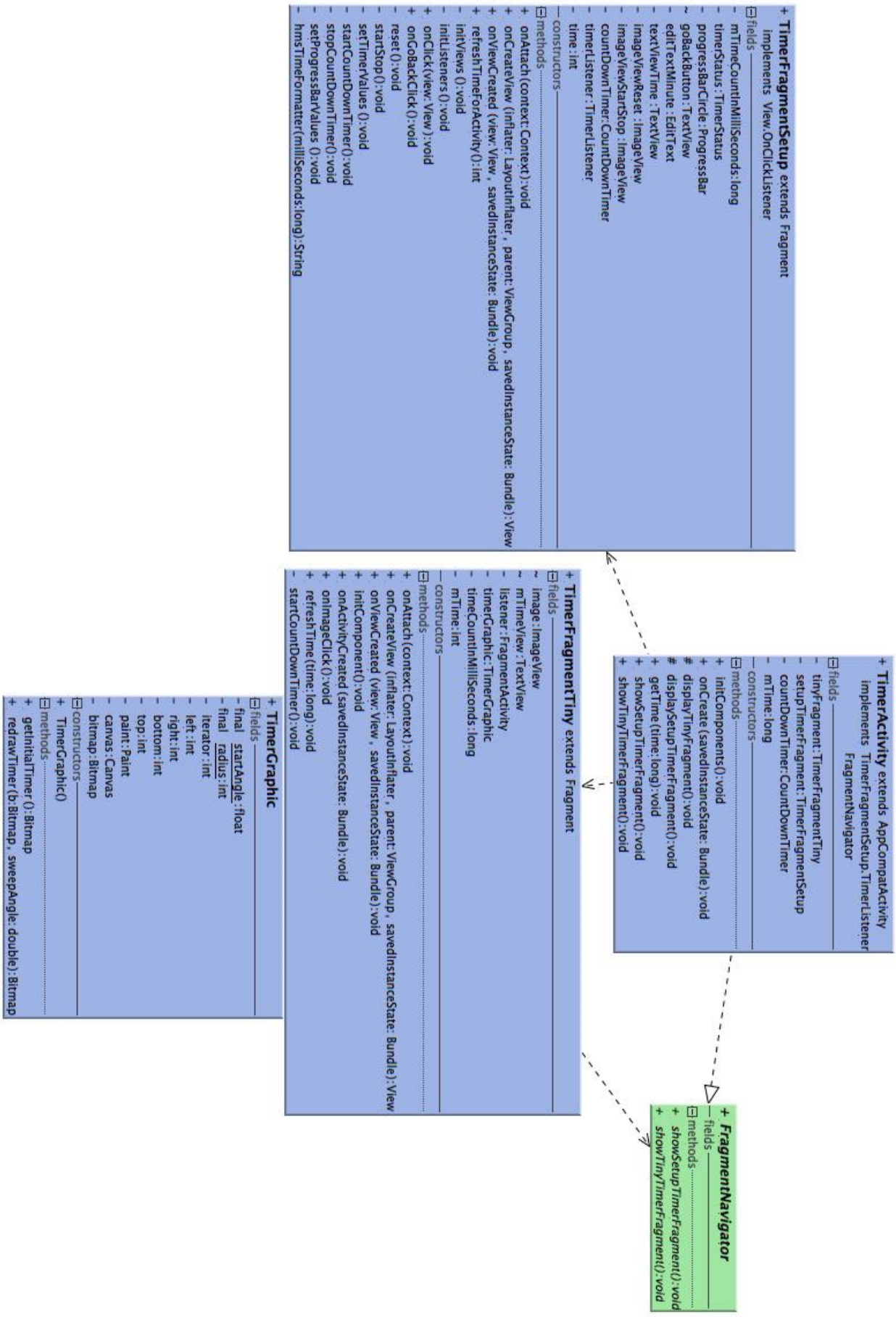
- Etablir un module de communication Accompagnateur – Enfant « Question/Réponse », portant sur la construction de phrase reprenant en partie le système de scénario. Il y aurait alors un mode « Accompagnateur » et un mode « Enfant ». Le mode enfant permettrait de construire une ou plusieurs phrases, auquel l'accompagnateur pourrait répondre.

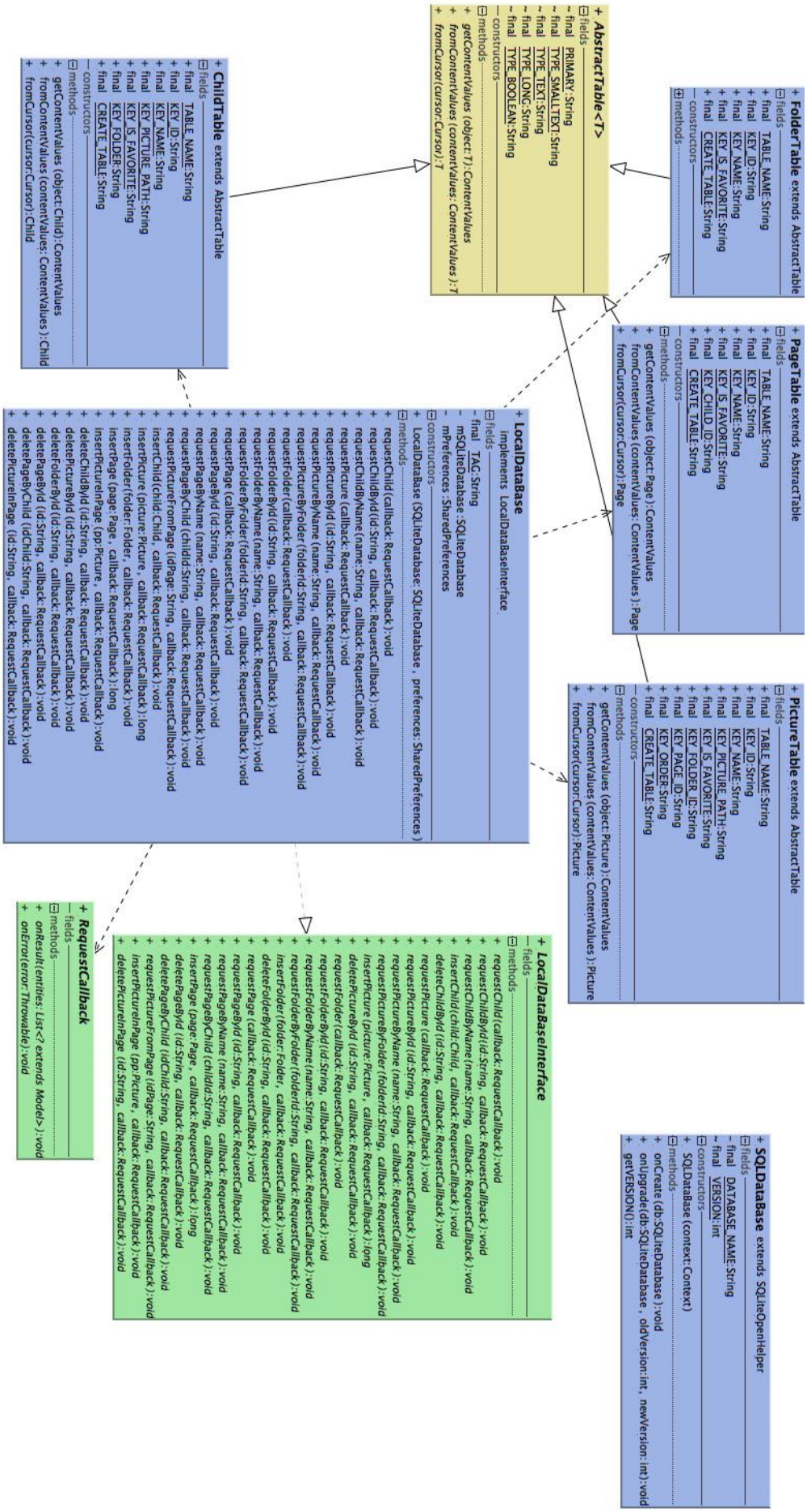
- Un contrôle vocal qui permettrait, en association avec le module Question/Réponse, de générer des phrases pour l'accompagnateur qui serait retransmit en une suite d'images, voir même de trouver des images dans la création de scénario plutôt que de devoir écrire dans une barre de recherche.

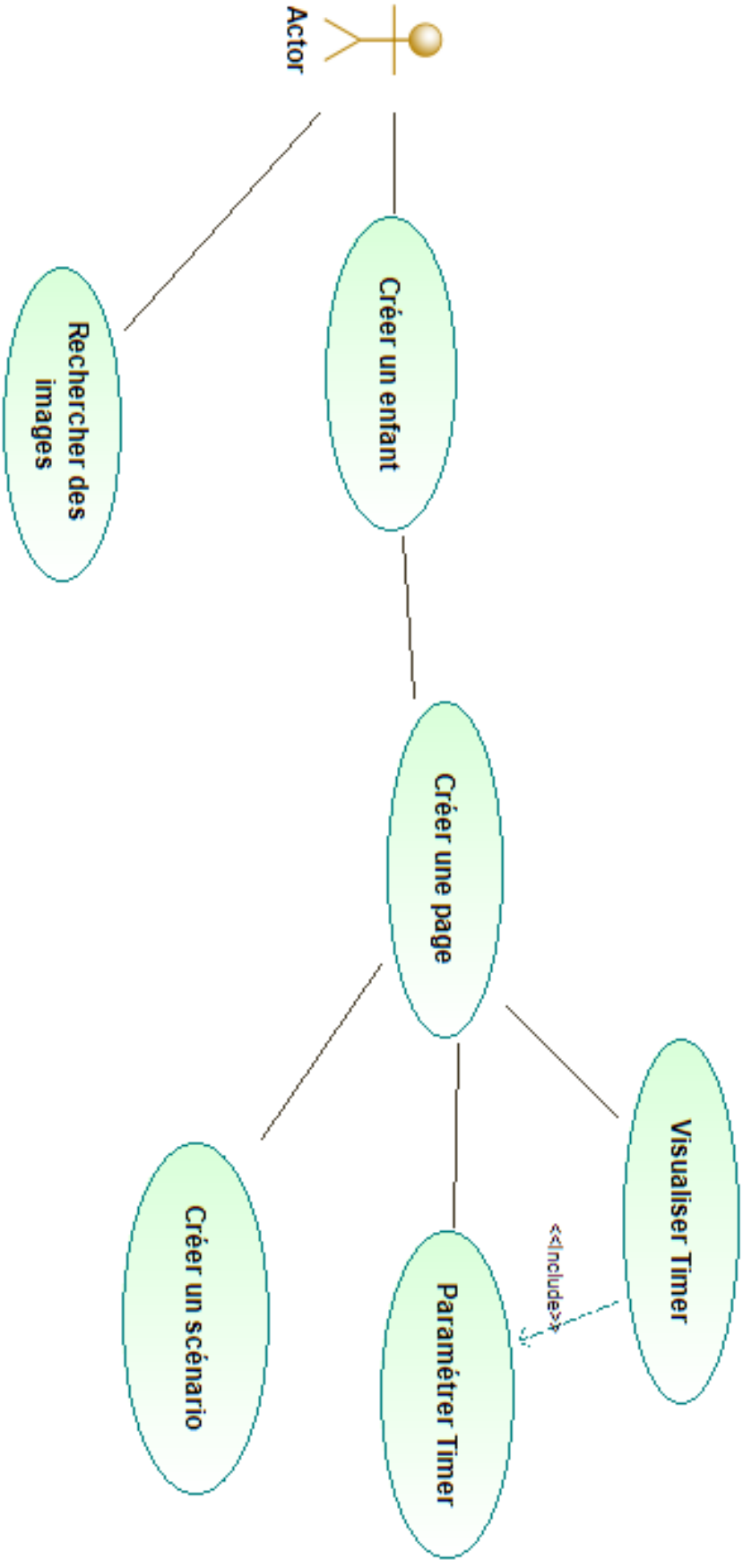
- Un système d'exploration de fichiers ou de création de sous-dossiers qui permettrait de personnaliser et de découper par thème, les images que l'utilisateur peut déposer sur son téléphone.

Annexe

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/ap
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <ImageView
        android:id="@+id/img_child"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:src="@drawable/ic_launcher_background"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />

    <TextView
        android:id="@+id/family_name_child"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:text="Nom de famille"
        app:layout_constraintLeft_toRightOf="@id/img_child"
        app:layout_constraintTop_toTopOf="@id/img_child"
        />

    <TextView
        android:id="@+id/name_child"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Prénom"
        app:layout_constraintTop_toBottomOf="@id/family_name_child"
        app:layout_constraintLeft_toLeftOf="@id/family_name_child"
        />

    <TextView
        android:id="@+id/years_child"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Age"
        app:layout_constraintLeft_toLeftOf="@id/name_child"
        app:layout_constraintTop_toBottomOf="@id/name_child"
        />

</android.support.constraint.ConstraintLayout>
```


+ TimerFragmentSetup extends Fragment
implements View.OnClickListener
fields
- mTimeCountInMilliSeconds:long
- timerStatus:TimerStatus
- progressBarCircle:ProgressBar
~ goBackButton:TextView
- editTextMinute:EditText
- textViewTime:TextView
- imageViewReset:ImageView
- imageViewStartStop:ImageView
- countDownTimer:CountDownTimer
- timerListener:TimerListener
- time:int
constructors
methods
+ onAttach(context:Context):void
+ onCreateView(inflater:LayoutInflater, parent:ViewGroup, savedInstanceState:Bundle):View
+ onViewCreated(view:View, savedInstanceState:Bundle):void
+ refreshTimeForActivity():int
- initViews():void
- initListeners():void
+ onClick(view:View):void
+ onGoBackClick():void
- reset():void
- startStop():void
- setTimerValues():void
- startCountDownTimer():void
- stopCountDownTimer():void
- setProgressBarValues():void
- hmsTimeFormatter(milliSeconds:long):String
+ TimerActivity extends AppCompatActivity
implements TimerFragmentSetup.TimerListener
FragmentNavigator
fields
- tinyFragment:TimerFragmentTiny
- setupTimerFragment:TimerFragmentSetup
- countDownTimer:CountDownTimer
- mTime:long
constructors
methods
+ initComponents():void
+ onCreate(savedInstanceState:Bundle):void
# displayTinyFragment():void
# displaySetupTimerFragment():void
+ getTime(time:long):void
+ showSetupTimerFragment():void
+ showTinyTimerFragment():void
+ FragmentNavigator
fields
methods
+ showSetupTimerFragment():void
+ showTinyTimerFragment():void
+ TimerFragmentTiny extends Fragment
fields
~ image:ImageView
~ mTimeView:TextView
- listener:FragmentActivity
- timerGraphic:TimerGraphic
- timeCountInMilliSeconds:long
- mTime:int
constructors
methods
+ onAttach(context:Context):void
+ onCreateView(inflater:LayoutInflater, parent:ViewGroup, savedInstanceState:Bundle):View
+ onViewCreated(view:View, savedInstanceState:Bundle):void
+ initComponent():void
+ onActivityCreated(savedInstanceState:Bundle):void
+ onImageClick():void
+ refreshTime(time:long):void
- startCountDownTimer():void
+ TimerGraphic
fields
- final startAngle:float
- final radius:int
- iterator:int
- left:int
- right:int
- bottom:int
- top:int
- paint:Paint
- canvas:Canvas
- bitmap:Bitmap
constructors
+ TimerGraphic()
methods
+ getInitialTimer():Bitmap
+ redrawTimer(b:Bitmap, sweepAngle:double):Bitmap


+ FolderTable extends AbstractTable
fields
+ final TABLE_NAME:String
+ final KEY_ID:String
+ final KEY_NAME:String
+ final KEY_IS_FAVORITE:String
+ final CREATE_TABLE:String
constructors
methods
+ AbstractTable<T>
fields
~ final PRIMARY:String
~ final TYPE_SMALLTEXT:String
~ final TYPE_TEXT:String
~ final TYPE_LONG:String
~ final TYPE_BOOLEAN:String
constructors
methods
+ getContentValues (object:T):ContentValues
+ fromContentValues (contentValues: ContentValues):T
+ fromCursor(cursor:Curso):T
+ PageTable extends AbstractTable
fields
+ final TABLE_NAME:String
+ final KEY_ID:String
+ final KEY_NAME:String
+ final KEY_IS_FAVORITE:String
+ final KEY_CHILD_ID:String
+ final CREATE_TABLE:String
constructors
methods
+ getContentValues (object:Page):ContentValues
+ fromContentValues (contentValues: ContentValues):Page
+ fromCursor(cursor:Cursor):Page
+ PictureTable extends AbstractTable
fields
+ final TABLE_NAME:String
+ final KEY_ID:String
+ final KEY_NAME:String
+ final KEY_PICTURE_PATH:String
+ final KEY_IS_FAVORITE:String
+ final KEY_FOLDER_ID:String
+ final KEY_PAGE_ID:String
+ final KEY_ORDER:String
+ final CREATE_TABLE:String
constructors
methods
+ getContentValues (object:Picture):ContentValues
+ fromContentValues (contentValues: ContentValues):Picture
+ fromCursor(cursor:Cursor):Picture
+ ChildTable extends AbstractTable
fields
+ final TABLE_NAME:String
+ final KEY_ID:String
+ final KEY_NAME:String
+ final KEY_PICTURE_PATH:String
+ final KEY_IS_FAVORITE:String
+ final KEY_FOLDER:String
+ final CREATE_TABLE:String
constructors
methods
+ getContentValues (object:Child):ContentValues
+ fromContentValues (contentValues: ContentValues):Child
+ fromCursor(cursor:Cursor):Child
+ LocalDataBase
implements LocalDataBaseInterface
fields
- final TAG:String
- mSQLiteDatabase :SQLiteDatabase
- mPreferences :SharedPreferences
constructors
+ LocalDataBase (SQLiteDatabase: SQLiteDatabase , preferences: SharedPreferences)
methods
+ requestChild (callback: RequestCallback):void
+ requestChildById(id:String, callback: RequestCallback):void
+ requestChildByName(name:String, callback: RequestCallback):void
+ requestPicture (callback: RequestCallback):void
+ requestPictureById (id:String, callback: RequestCallback):void
+ requestPictureByName (name:String, callback: RequestCallback):void
+ requestPictureByFolder (folderId:String, callback: RequestCallback):void
+ requestFolder(callback: RequestCallback):void
+ requestFolderById(id:String, callback: RequestCallback):void
+ requestFolderByName(name:String, callback: RequestCallback):void
+ requestFolderByFolder(folderId:String, callback: RequestCallback):void
+ requestPage (callback: RequestCallback):void
+ requestPageById (id:String, callback: RequestCallback):void
+ requestPageByName (name:String, callback: RequestCallback):void
+ requestPageByChild (childId:String, callback: RequestCallback):void
+ requestPictureFromPage (idPage:String, callback: RequestCallback):void
+ insertChild(child:Child, callback: RequestCallback):void
+ insertPicture (picture:Picture , callback: RequestCallback):long
+ insertFolder(folder:Folder, callback: RequestCallback):void
+ insertPage (page:Page , callback: RequestCallback):long
+ insertPictureInPage (pp:Picture , callback: RequestCallback):void
+ deleteChildById (id:String, callback: RequestCallback):void
+ deletePictureById (id:String, callback: RequestCallback):void
+ deleteFolderById (id:String, callback: RequestCallback):void
+ deletePageById (id:String, callback: RequestCallback):void
+ deletePageByChild (idChild:String, callback: RequestCallback):void
+ deletePictureInPage (id:String, callback: RequestCallback):void
+ LocalDataBaseInterface
fields
methods
+ requestChild (callback: RequestCallback):void
+ requestChildById (id:String, callback: RequestCallback):void
+ requestChildByName (name:String, callback: RequestCallback):void
+ insertChild(child:Child, callback: RequestCallback):void
+ deleteChildById (id:String, callback: RequestCallback):void
+ requestPicture (callback: RequestCallback):void
+ requestPictureById (id:String, callback: RequestCallback):void
+ requestPictureByName (name:String, callback: RequestCallback):void
+ requestPictureByFolder (folderId:String, callback: RequestCallback):void
+ insertPicture (picture:Picture , callback: RequestCallback):long
+ deletePictureById (id:String, callback: RequestCallback):void
+ requestFolder(callback: RequestCallback):void
+ requestFolderById(id:String, callback: RequestCallback):void
+ requestFolderByName(name:String, callback: RequestCallback):void
+ requestFolderByFolder(folderId:String, callback: RequestCallback):void
+ insertFolder(folder:Folder, callback: RequestCallback):void
+ deleteFolderById (id:String, callback: RequestCallback):void
+ requestPage (callback: RequestCallback):void
+ requestPageById (id:String, callback: RequestCallback):void
+ requestPageByName (name:String, callback: RequestCallback):void
+ requestPageByChild (childId:String, callback: RequestCallback):void
+ insertPage (page:Page , callback: RequestCallback):long
+ deletePageById (id:String, callback: RequestCallback):void
+ deletePageByChild (idChild:String, callback: RequestCallback):void
+ requestPictureFromPage (idPage:String, callback: RequestCallback):void
+ insertPictureInPage (pp:Picture , callback: RequestCallback):void
+ deletePictureInPage (id:String, callback: RequestCallback):void
+ RequestCallback
fields
methods
+ onResult(entities: List<? extends Model>):void
+ onError(error:Throwable):void
+ SQLiteDatabase extends SQLiteOpenHelper
fields
- final DATABASE_NAME:String
~ final VERSION:int
constructors
+ SQLiteDatabase (context:Context)
methods
+ onCreate (db:SQLiteDatabase):void
+ onUpgrade(db:SQLiteDatabase , oldVersion:int, newVersion: int):void
+ getVERSION():int


Actor
Rechercher des images
Créer un enfant
Créer une page
Créer un scénario
Paramétrer Timer
<<Include>>
Visualiser Timer