IUT Lyon 1
l'excellence technologique

DÉPARTEMENT INFORMATIQUE
71 rue Peter Fink 01000
Bourg-en-Bresse
04 74 45 50 59

Diplôme Universitaire de Technologie

Informatique

SAE 3A

EDUKID

LAPIERRE Yohan VAITON Damien GAILLIARD Axel

Promotion 2021/2024

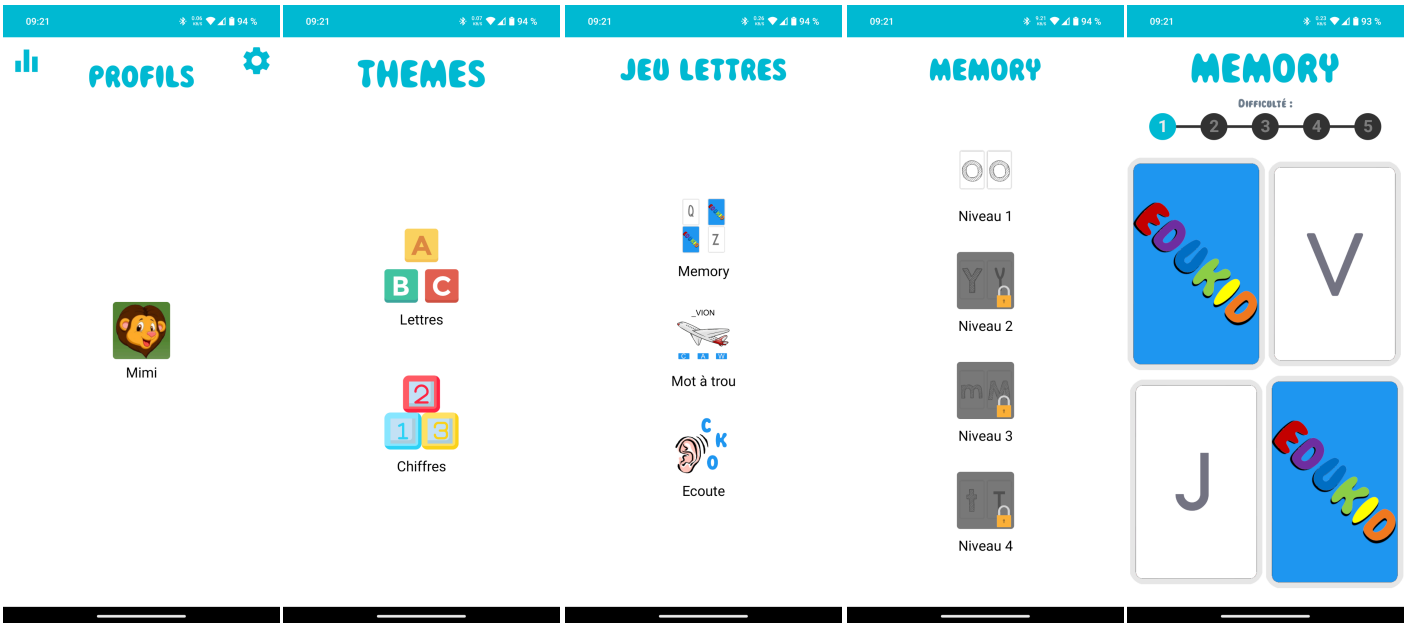
Table des matières

Le sujet de la Situation d'Apprentissage et d'Évaluation (SAE) est de retravailler, améliorer une application déjà existante. Notre groupe a choisi le sujet : EDUKID.

Description de l'existant et des technologies utilisées

Utilisation

EDUKID est une application destinée aux jeunes enfants via des jeux éducatifs. Au lancement, on accède dans un premier temps.
Le projet initial, nommé EDUKID est une application mobile à destination des enfants (autour des 6 ans maximum) et de leurs parents. L'enfant pourra apprendre de manière ludique grâce à de multiples mini-jeux divisés en deux catégories : les chiffres et les lettres. L'application est ergonomique pour des enfants en bas âge. Elle possède des sons et des voix ainsi que différents mini-jeux et niveaux.



L'application mobile de base possède plusieurs utilisateurs cible avec des besoins différents.

Les enfants

Les enfants utilisent l'application dans le but de s'amuser, probablement motivés par les demandes des parents. Ils pourront à la fois s'amuser et apprendre les bases comme les chiffres et les lettres.

Les enfants ne sont pas très attentifs et peuvent vite perdre leur concentration, par conséquent, il faut appuyer sur le design et sur la sensation d'amusement pour que l'enfant reste actif sur l'application. Dû à leur jeune âge, les enfants n'auront probablement pas la faculté de lire, il faudra donc prévoir une interface imagée et des explications orales claires.

Les parents

Les parents sont généralement attentifs à l'utilisation des applications destinées à leurs enfants. Ainsi, il est possible de suivre la progression de leurs enfants directement dans l'application.

Technique

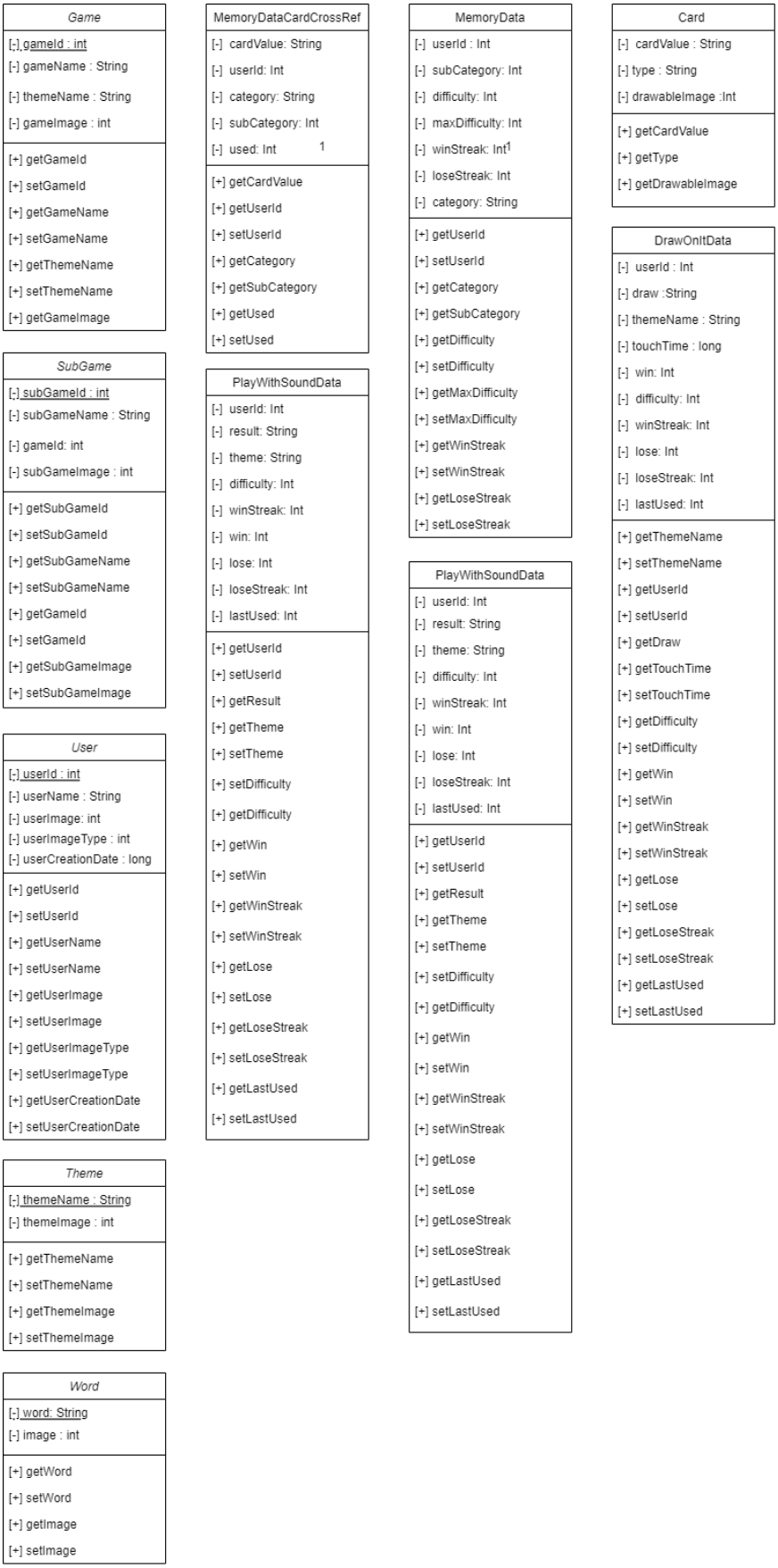
Généralité

EDUKID est une application mobile faite pour la plateforme Android uniquement. Elle est développée en JAVA. Elle ne suit aucune architecture particulière et se compose d'une suite d'activités jusqu'à un jeu avec en cache les sélections du joueur. La base de données est initialisée au lancement de l'application à chaque fois.

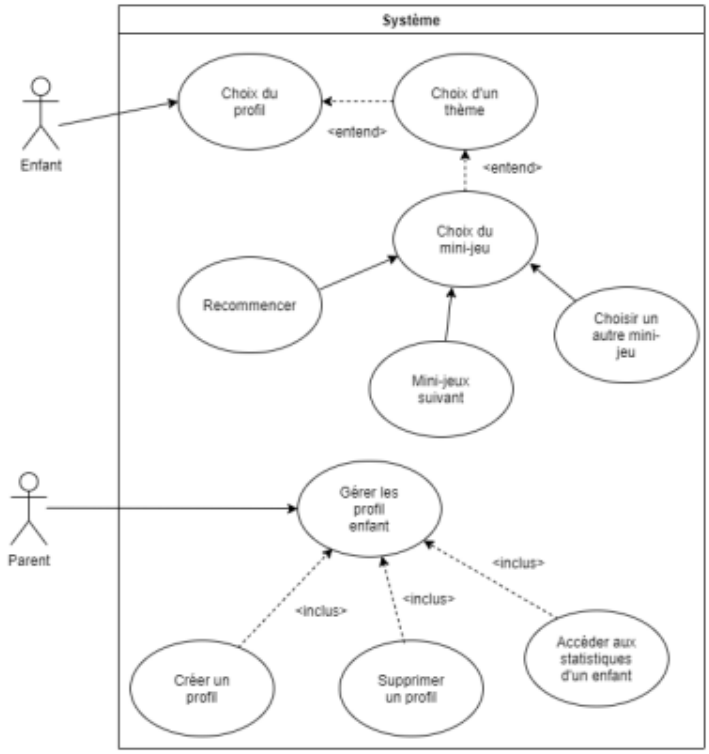
Dépendances utilisé

Room : accès fluide à SqLite
Stateprogressbar : barre de progression avec animations
Roundedimageview : Transformation et manipulation d'images
androidanimations : animations simple
MPAndroidChart : Schema graphique

Modélisation des Entités manipulées et leurs relations



Accessibilité



Le système de niveau des mini-jeux est mis en place mais est pour le moment compliqué, durant nos tests, nous avons eu des difficultés à atteindre le niveau 2 du *memory*, ce jeu comportant une partie d'aléatoire, les premières difficultés ne posent pas de problème mais dès que le nombre de cartes augmente, la chance de réussir rapidement diminué mais le nombre d'erreur pour ne pas perdre d'étoiles ne bouge pas. On se retrouve donc avec un score d'une étoile ce qui empêche le passage au niveau supérieur.

La gestion du profil est complète et seuls quelques bugs sont à résoudre tel que l'importation de photo de profil non disponible sur certains téléphones.

Problèmes de développement

Ces différentes parties sont présentées pour illustrer quelques exemples de code non conventionnel qui pose des soucis dans la compréhension ainsi que dans le maintien et l'amélioration du code que nous avons dû retravailler.

Adaptabilité des menus

Les menus dans l'application n'ont pas été prévus pour que de nouveaux *thèmes*, *game* ou *subgame/niveaux* soient ajustables facilement. On ne peut pas simplement ajouter des données dans la base de données et laisser l'affichage s'en occuper.

Les menus utilisent des *gridView*, alors que dans ce cas il serait plus simple d'utiliser de simple *ScrollView*, plus adapté à cette situation.

Complexité des jeux

Le *gridView* semble cohérent dans la situation du jeu *Memory* car les cartes sont disposées en longueur et en largeur, en 2 dimensions et c'est ce qui est utilisé. Il y a 2 façons d'avoir une carte, soit par une image soit par un texte. Le problème majeur est la façon utilisée pour afficher les cartes qui utilise des images. En effet, toutes les images du jeu (l'ensemble de images chiffre) étaient chargées dans chaque ''case'' du *grid* et seule l'image voulant être affichée n'était pas *hidden*. Cette manière de faire ne nous a semblé pas bonne car elle surcharge le code et le rend difficilement compréhensible. Cela empêche aussi de facilement ajouter des images au jeux car en plus de les inscrire en base de données, il faut les ajouter aux autres images déjà présentes qui sont tout entassées au même endroit. La tâche est complexe en plus de ne pas être cohérente, car modifier, ajouter ou supprimer une image touche à la vue, la logique et les données de l'application.

Complexité du code

Le code est très peu documenté et le nom des variables reste assez flou et concis, ne permettant pas de comprendre facilement la logique prise par les précédents développeurs.

Modifications

EDUKID est une application fonctionnelle mais qui ne possède aucune architecture reconnue. Cela rend l'amélioration de l'application très complexe. La maintenabilité se retrouve aussi affectée par ce "désordre interne".
Pour ces raisons, nous avons axé notre SAE sur la restructuration de l'application afin de lui donner une architecture stable, fiable et maintenable dans la durée.
Cette restructuration nécessite le redéveloppement de l'application, alors nous allons aussi changer de langage en passant de Java à Kotlin. Kotlin est basé sur Java et possède certains avantages, notamment sa simplicité (Null-safety par exemple), moderne (coroutines par exemple) et une grande interopérabilité avec Java. Kotlin est aussi un langage utilisé dans nos études et nos projets, nous avons plus d'expérience en application mobile avec ce langage.

Architecture

Afin de résoudre le problème de structuration du code que nous avons soulevé lors de notre analyse, nous avons décidé, en plus du changement de langage, d'adopter une architecture de type "MVVM" (Modèle-Vue-Vue Modèle). Il faut savoir que beaucoup d'applications mobiles sont réalisées avec ce concept parce qu'il s'adapte très bien aux applications mobiles mais il peut également être utilisé pour des sites Web.
Commençons par une explication du fonctionnement :

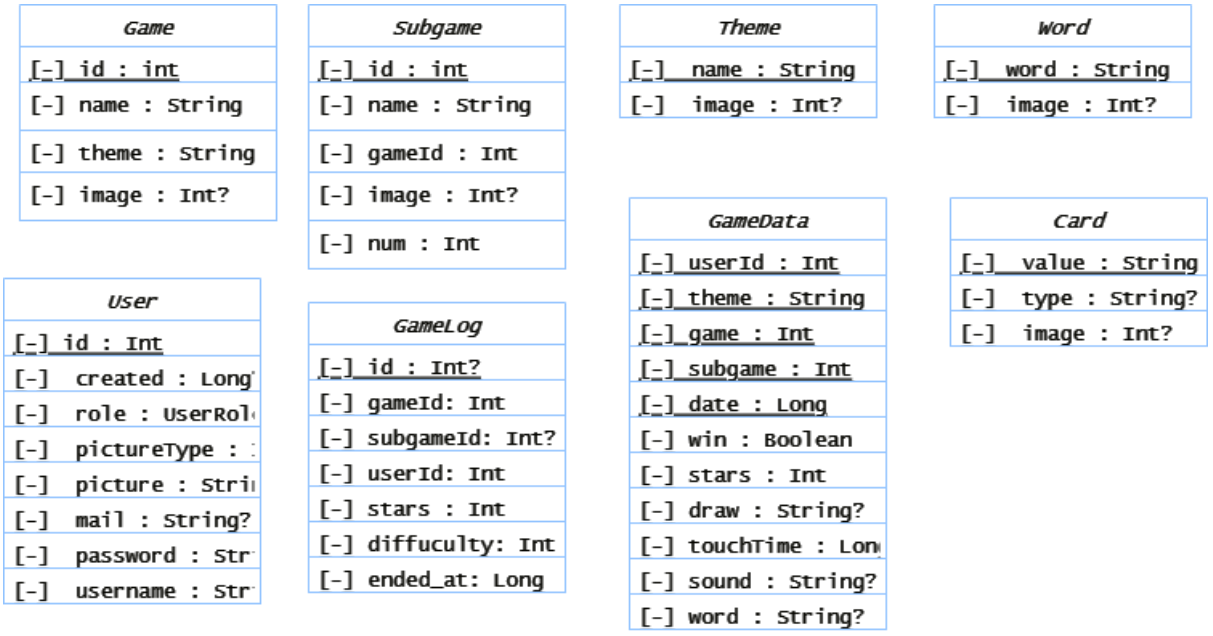
- Modèle :
    - Le modèle représente les données et la logique métier de l'application. Il contient les structures de données, les règles métier, les opérations de

traitement des données, etc. Le modèle n'a aucune connaissance de l'interface utilisateur et fonctionne indépendamment de celle-ci.

- Vue :
    - La vue est responsable de l'interface utilisateur de l'application. Elle affiche les données à l'utilisateur et capture les interactions de l'utilisateur, telles que les clics de boutons, les entrées de formulaire, etc. La vue ne contient quasiment jamais de logique métier.
- VueModèle :
    - Le vue-modèle agit comme un intermédiaire entre la vue et le modèle. Il expose les données et le comportement nécessaires à la vue, généralement sous une forme plus adaptée à l'interface utilisateur. Le vue-modèle traduit les données du modèle en quelque chose que la vue peut afficher et interpréter, et il convertit également les actions de l'utilisateur en commandes pour le modèle.

Cela permet de séparer les responsabilités entre les différentes parties de l'application. En découle le principal avantage d'avoir un patron d'architecture dans son projet est la maintenabilité. Avec un MVVM, modifier une partie de l'application a beaucoup moins de chances d'impacter le reste de l'application. Cela permet également de faciliter la compréhension du code pour quelqu'un qui arriverait sur le projet puisque tout est ordonné dans les différents fichiers.

Gestion des données



Grâce au passage vers le langage Kotlin, nous avons eu accès au concept de DataClass qui permet de simplifier les fichiers de base de données en se passant des méthodes *getter* et *setter*. On en a profité également pour simplifier un peu la base de données initiale.

Dépendances

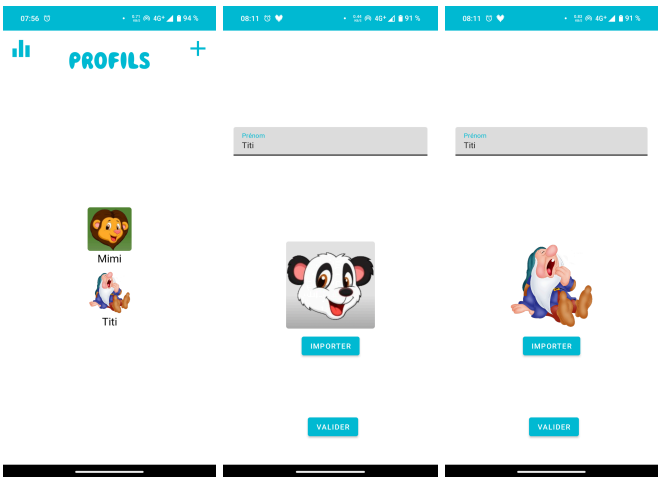
Room : accès fluide à SqLite
Timber : Débogage
Glide : gestion de l'affichage des images
SplashScreen : Gestion du SplashScreen
Daimaija : Dessin
Retrofit : Appels API

Résultat

La *V1* qui nous a été fourni lors du début du projet n'était pas du tout modulable.
Nous avons pu dans les délais du projet ajouter seulement un jeu, mais ce jeu est très personnalisable. En effet, il est possible en ajoutant une simple ligne en base de données de créer un *thème*. On peut ajouter des *game* juste en précisant à quel *thème* il appartient, et de la même façon pour les *subgames* ainsi que des images. Il est donc très simple de personnaliser son jeu, en seulement quelques lignes et quelques images, pour le *memory* par exemple.

User

Comme dans la *V1* il est possible de créer des profils avec des images déjà présentes dans l'application ou avec des images importées de l'appareil de l'utilisateur. Ces différents utilisateurs auront leur propre statistique.
Le choix de profil se fait sur la première page de l'application, si aucun utilisateur existe, on est automatiquement dirigés vers la création d'un profil.
La première image est la liste des profils déjà créés, la deuxième image est la page de création d'un profil, et la dernière image est la page de création de profil avec une image importée localement de l'appareil.



Thèmes

Dans l'application, en plus des *thèmes* des lettres et des chiffres, nous avons récemment ajouté des contenus sur les fruits et légumes, les couleurs, les animaux et les formes géométriques. Ces nouveaux *thèmes* élargissent les possibilités d'apprentissage et offrent aux enfants une expérience plus enrichissante et diversifiée.

Les nouveaux thèmes sont plus riches que les 2 premiers. Une centaine d'images ont été ajoutées.

Jeux

Memory

Concernant l'avancement du redéveloppement du *memory*, le jeu en lui-même est entièrement terminé et fonctionne correctement. Cependant, la gestion du déblocage des niveaux en fonction de la réussite ou de l'échec n'est pas encore entièrement mise en place. Actuellement, tous les niveaux sont accessibles même si les niveaux précédents n'ont pas été réussis. Toutefois, tous les éléments nécessaires pour implémenter cette fonctionnalité sont prêts, il reste simplement à finaliser la logique de progression dans le jeu afin de rendre l'expérience plus cohérente et gratifiante pour les joueurs.

Autres

Les autres jeux n'ont pas pu être retravaillés par manque de temps, notre application possède par contre de bonnes bases pour un meilleur développement de ces jeux.

Intégration continue

L'application possède maintenant un système d'intégration continu. Nous avons mis en place un système pour gérer les différentes versions de l'application, à savoir le mode de développement (dev), le mode de publication (recette) et le mode de production (prod). Cela permet de gérer chaque étape du cycle de vie de EDUKID.
Tout d'abord, le mode de développement est l'environnement où nous, les développeurs, travaillons sur les nouvelles fonctionnalités et effectuons des modifications de code. Ensuite, le mode de publication (release) est une étape intermédiaire où les fonctionnalités développées sont regroupées pour être testées dans un environnement plus proche de la production. Enfin, le mode de production est l'environnement où l'application est déployée pour les utilisateurs finaux.

Conclusion / Bilan personnel

Axel

Travailler sur ce projet m'a permis de renforcer mes compétences en développement d'applications mobiles. Le passage de Java à Kotlin, ainsi que l'adoption de l'architecture MVVM m'ont permis d'approfondir mes connaissances de l'organisation de code.

Yohan

Ce projet Android en Kotlin a été super enrichissant. La maintenance de l'appli m'a vraiment fait réaliser à quel point la qualité du code et l'architecture MVVM sont essentielles. J'ai aussi amélioré mes compétences avec ROOM pour la base de données. Gérer l'équipe a été une expérience cool qui a ajouté une autre corde à mon arc. En résumé, ce projet a été bien plus qu'une simple mission informatique, c'était une vraie leçon pratique.

Damien

Ce projet m'a permis d'approfondir mes connaissances dans la gestion d'un projet informatique qui n'est pas le mien. La compétence principale que j'ai travaillée est l'analyse, notamment la recherche de points à améliorer comme l'architecture ou encore la propreté et donc par conséquent, la recherche de solutions pour les résoudre.

Aides

- *Thèmes* est composé d’un ou plusieurs *game lui-même pouvant être composé d’un ou plusieurs subgame/niveaux*
- *scrollView* est adapté pour des affichages en liste comme pour les *thèmes*, *game et subgame/niveaux* où le nombre d'élément à afficher n’est pas déterminé à l’avance et est donc par définition très modulable
- *gridView* est semblable à une *scrollView* mais en 2 dimension, une sorte de tableau pratique pour afficher un nombre d'élément important à l'écran de manière organisé, régulier et adaptable car en effet ici aussi le nombre d'élément n’est pas déterminé à l’avance et est donc très modulable
- *hidden* est un attribut qui quand il est activé permet de cacher un élément
- *V1* est la version du code que nous avons reçu au début du projet.