RAPPORT DE PROJET

API Libraries Demo

Thibault Vieux
vieuxthiba@eisti.eu

Sommaire

Présentation du projet

Description

Conçu pour les développeurs et les concepteurs, cette application fournit une collection d'exemples de bibliothèques tierces sur Android. Ces exemples sont inspirés des codes sources présentés sur le site Android Developers.

Album photo : https://goo.gl/photos/oNJdrWpF1NcdarG46
Slides : http://slides.com/thibaultvieux/api-librairies-demo

Tester l'application

Dépendances

Android >= 5.1 (API 22)

Installation

Playstore (recommandé) : https://play.google.com/store/apps/details?id=com.melkir.libraries
Fichier APK : https://github.com/melkir/API-Libraries-Demo/releases

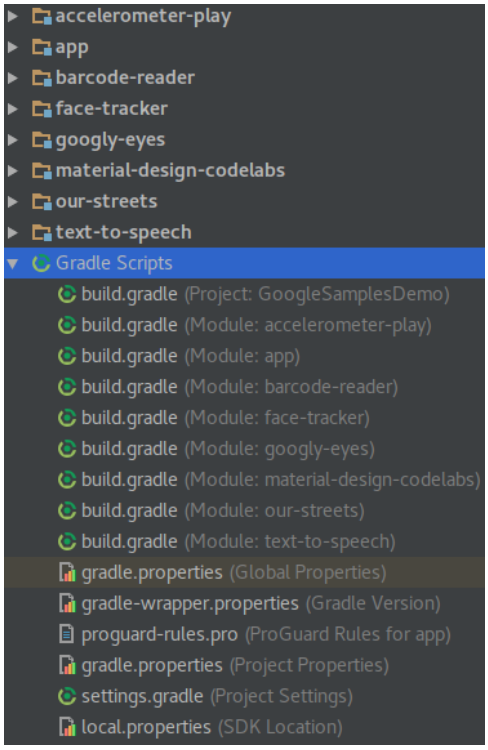
Remarque : Pour la méthode APK il faut autoriser l'installation à partir des sources inconnues dans les paramètres de sécurité du smartphone Android.

Participer au développement

Récupérer le code source du projet :
`git clone https://github.com/melkir/API-Libraries-Demo`

Importer le fichier **`build.gradle`** dans Android Studio
Ajouter un compte Firebase au projet Android : documentation
Ajouter une clé Google Map pour Our Streets : documentation

Architecture de l'application



Le projet étant open-source et ayant pour but d'être réutilisable pour les développeurs, son architecture reprend les principes de la programmation modulaire.

Chaque exemple présent dans l'application est isolé dans un module (appelé Android Library) qui est indépendant du projet dans sa globalité.

Cela m'a permis de grandement faciliter le développement du projet et sa maintenance.

« Il est plus facile de décomposer un problème en ses éléments, forcément plus simples, que de le traiter dans sa totalité » (dixit Descartes)

A noter la programmation modulaire peut s'avérer très utile dans un travail en équipe.

Arborescence des layouts (TreeView) : lien

Les bibliothèques et outils présentés dans l'application sont les suivants :

- Android Vision (barcode-reader, face-tacker, googly-eyes)
- Accéléromètre (accelerometer-play)
- Google Maps (our-streets)
- Firebase UI (app)
- Text-to-Speech (text-to-speech)
- In App Browser (ai-experiments, halloween-doodle)

L'application utilise différent layouts en fonction de l'orientation de l'appareil et les ressources textuelles sont externalisées afin de faciliter la traduction.

Les préférences de l'utilisateur et la session de l'utilisateur sont enregistrées dans l'appareil afin de ne pas être perdu à chaque fermeture de l'application.

La présentation des exemples se fait grâce à un RecyclerView qui s'adapte en fonction des actions de l'utilisateur (lister les exemples, rechercher et filtrer par catégorie).

Points techniques

Les exemples permettent de se faire une idée de ce que l'on pourrait implémenter pour notre future application tout en s'assurant que la bibliothèque que l'on souhaite utiliser fonctionne correctement. Ils permettent également de s'assurer que l'API fournit avec est simple d'utilisation et facile à implémenter.

API Libraries Demo permet de tester et interagir avec ces bibliothèques de manière lucrative. Parfois un simple readme, un GIF ou une vidéo n'est pas suffisant pour convaincre une personne d'utiliser sa bibliothèque et c'est là qu'entre en jeu mon application.

L'application couvre les aspects suivants :

Design
- La présentation des exemples s'adapte en fonction de l'orientation de l'appareil.
- Interface Material Design respectant les lignes directives pour Android.

Performance
- Chargement des images avec la bibliothèque Glide afin d'améliorer la vitesse de chargement et réduire la consommation mémoire.
- Taille de l'application réduite en utilisant Shrink (minify + proguard) et en convertissant les images au format WebP.

Fonctionnalités
- Présentation de 7 exemples d'utilisation de bibliothèques Android
- Possibilité de filtrer les exemples par catégorie et d'effectuer une recherche.
- Possibilité de rattacher son compte Google afin d'afficher ses informations dans le tiroir de navigation de l'application.
- Possibilité de modifier la langue de l'application depuis les réglages.

Internationalisation
- L'application est traduite en français et en anglais.

Vie privée
- Possibilité de détacher son compte Google depuis les réglages.

Droit d'auteur
- La page à propos mentionne les contributeurs et la licence du projet.
- Chaque exemple contient un lien vers la page de l'auteur dans la vue détaillée.

Limites des fonctionnalités

Sur tablette il se peut que les images de présentations ne soient pas correctement ajustées dû aux dimensions de l'écran. Pour remédier à cela il faudrait définir un layout spécifique pour les écrans larges.

La recherche par titre de module et le filtrage par catégorie sont deux fonctionnalités qui fonctionnement de manière indépendantes, ils ne partagent pas la même liste de résultat ni le même ViewAdapter ce qui provoque un bug d'affichage lorsqu'on les utilise en même temps.

Idées pour de futures améliorations

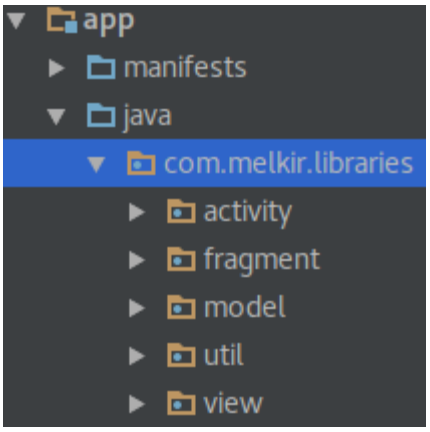
Corriger en priorité les bugs identifiés puis implémenter de nouvelles fonctionnalités en fonction des retours utilisateurs qui seront obtenus sur la page du Playstore, Github et Reddit.

Implémenter une transition Material Design entre l'activité présentant la liste des exemples et l'activité de détail en se basant sur les lignes directives de la documentation officielle (Pesto) : https://material.io/guidelines/motion/material-motion.html#material-motion-what-makes-a-good-transition

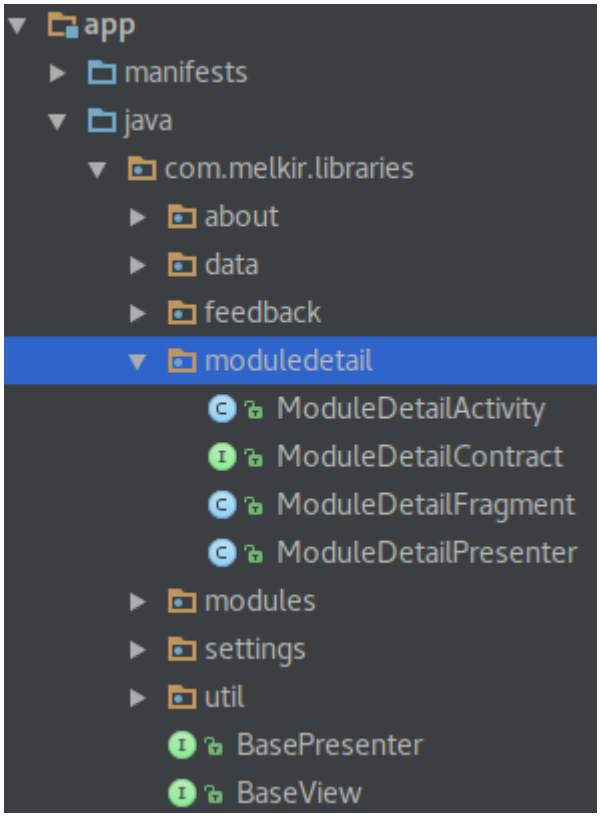
Au lieu de charger les modules au lancement il serait intéressant de les télécharger sur un playstore alternatif et de les charger dynamiquement sans avoir à recompiler l'application. Dans la mesure où il n'y aurait pas de problème de sécurité cela permettrait d'y héberger tous les exemples présents dans les repos officiels. Cela permettrait également aux développeurs Android de présenter leur bibliothèque via mon application.

Tests

Afin de pouvoir effectuer des tests d'intégrations sur l'application le patron d'architecture a été revu sous la forme d'un modèle-vue-présentation (MVP).
Pour cela, je me suis fortement inspiré des guidelines « Android Architecture Blueprints ».



Structure du projet avant

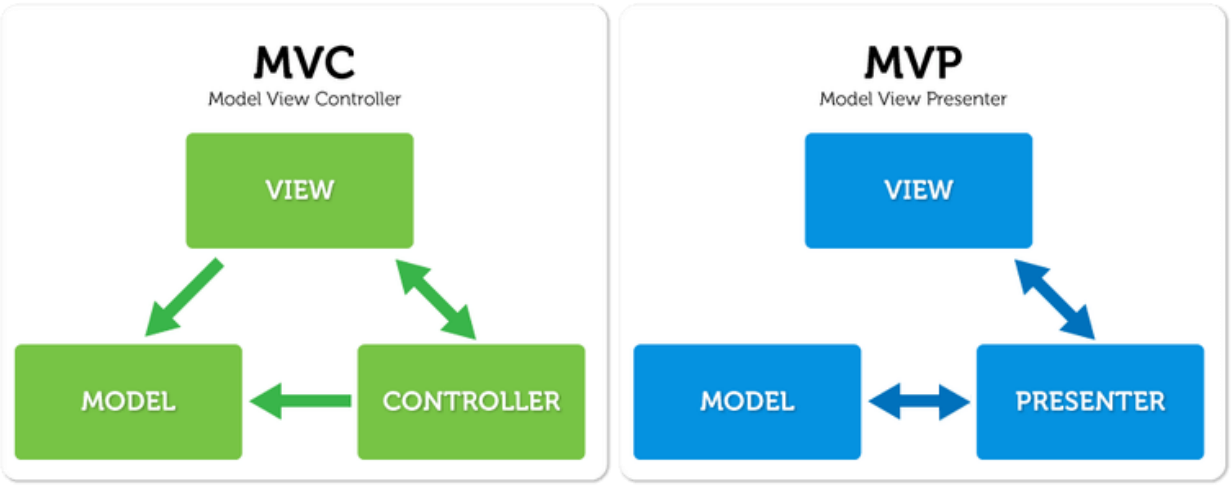


Structure du projet après (MVP)

Tests d'intégrations

La classe ModulesPresenterTest.java permet de tester les méthodes présentes dans le présentateur ModulesPresenter. Ce présentateur a pour rôle d'agir comme un intermédiaire entre la vue ModulesFragment et le modèle ModulesRepository où sont stockés nos données. Il définit la partie logique de l'application et décide de ce qu'il se passe lorsque nous interagissons avec la vue.

Contrairement au modèle MVC classique il n'y a pas de communication directe entre la vue et le modèle.



Le test ligne 49 peut être décrit comme suit :

// arange : Quand la méthode `getModules()` du modèle est appelé, remplacer le résultat par une liste de modules prédéfini.

// act : Appeler la méthode `loadModules()` du présentateur afin d'agir sur le programme.

// assert : Vérifier que la méthode `loadModules()` du présentateur appelle bien la méthode `showModules()` de la vue avec la liste de modules que nous avions prédéfini au début du test.

Le même principe s'applique pour ModuleDetailPresenterTest.java qui permet de tester le présentateur de la classe ModuleDetailPresenter.

Tests fonctionnels

La classe AppNavigationTest.java permet de tester la navigation dans l'application.
Exemple : Cliquer sur le bouton "rechercher", taper "face", vérifier que le premier résultat de la liste est "Face Tracker"

La classe LauncherTest.java permet de lancer tous les modules de l'application afin de vérifier que l'application ne crache pas.

Couverture du code

Package	Class, %	Method, %	Line, %
all classes	18,8% (3/ 16)	16,2% (18/ 111)	13,9% (48/ 345)

Répartition de la couverture

Package	Class, %	Method, %	Line, %
com.melkir.libraries.moduledetail	25% (1/ 4)	25% (5/ 20)	22,6% (14/ 62)
com.melkir.libraries.modules	25% (2/ 8)	20,3% (13/ 64)	15,5% (34/ 219)
com.melkir.libraries.modules.adapters	0% (0/ 4)	0% (0/ 27)	0% (0/ 64)

Rapport détaillé : Coverage-Report.7z

Annexe

Ressources principales

https://github.com/googlesamples
https://github.com/googlecreativelab
https://github.com/googlecodelabs
https://github.com/firebase/quickstart-android
https://github.com/android/platform_development
https://github.com/googlesamples/android-architecture

Code source des projets

https://github.com/googlesamples/android-vision
https://github.com/googlecreativelab/giantemoji
https://github.com/googlesamples/android-AccelerometerPlay
https://github.com/googlesamples/android-OurStreets
https://github.com/googlecodelabs/android-design-library
https://github.com/android/platform_development/.../TextToSpeechActivity.java

Ressources externes

https://www.udacity.com/course/new-android-fundamentals--ud851
https://developer.android.com/guide/topics/ui/themes.html
https://developer.android.com/guide/topics/resources/localization.html
https://developer.android.com/guide/practices/screens_support.html
https://developers.google.com/speed/webp/docs/using
https://developers.google.com/vision
https://aiexperiments.withgoogle.com
https://www.androidexperiments.com
https://www.google.com/doodles/halloween-2016
https://codelabs.developers.google.com/codelabs/material-design-style