BookLIST

An Android APP Project

### Rapport de projet

Par : Cédric Droguet

Matthieu Prudhomme

MSE 2014-2015

Table des MatiÈres

[Le but du projet 3](#_Toc407986497)

[La conception 3](#_Toc407986498)

[La réalisation 4](#_Toc407986499)

[Les problèmes rencontrés 5](#_Toc407986500)

[Mode d’emploi de l’application 7](#_Toc407986501)

[Conclusion 14](#_Toc407986502)

## Le but du projet

Le projet de « book suggestions » consiste à créer une application Android qui permet à l’utilisateur de recevoir des recommandations de lecture en fonctions de ses préférences.

Dans notre cas le programme laisse le choix à l’utilisateur des auteurs ou des titres qu’il souhaite qu’on lui recommande. Une recherche des livres recommandés est alors proposée à l’utilisateur. Il peut ensuite cliquer sur un article pour être redirigé vers la fiche globale du livre pour éventuellement se le procurer.

## La conception

Tout d’abord nous avons recherché ce qui existait dans le Google store. Force est de constater que l’offre concernant les livres est plutôt faible et concernant des outils de recommandations, rien de convaincant n’a été trouvé. Si certaines plateformes commerciales offrent certaines applications ce sont uniquement des porduits développées en interne.

A la suite de ce constat nous avons cherché à comprendre le pourquoi du comment et force a été de constaté que soit il n’existait pas d’API un peu près correcte mais non exhaustive ou alors au contraire il y avait une pléthore de livre mais la consistence des données laissaient à désirer. Peut-être n’avons-nous pas cherché au bon endroit mais c’était le cas pour la plupart des API ouvertes que nous avons testé. On se référera à la section Problèmes Rencontrés pour plus de détails.

Concernant les critères de conception, nous voulions une application rapide d’utilisation simple et non intrusive dans l’écosystème du téléphone. Pour le coup nous voulions laisser à l’utilisateur une grande liberté dans le choix de sa recherche. Nous voulions insérer aussi des propositions par genre mais dans l’API que nous avons choisi ce n’était pas possible – ni dans bon nombre d’autres d’ailleurs mais nous reviendrons sur ce points là dans la section Problèmes rencontrés.

Pour respecter le but du projet nous avons mis en place une liste de fonctionnalités que l’application devait présenter :

* Une recherche des meilleurs livres toutes catégories à l’ouverture de l’application ;
* La possibilité pour l’utilisateur d’émettre des préférences sur la recherche des livres (par titres ou par auteurs) ;
* La possibilité pour l’utilisateur de choisir la langue d’affichage ;
* La possibilité d’être redirigé sur une site d’achat à la selection d’un article.

## La réalisation

Nous avons abordé la réalisation en trois étapes :

* Réalisation d’une interface de base permettant d’afficher une liste de livres ;
* Recherche et mise en place d’une API gérant la recherche des recommandations de livres sur des classements récents ;
* Réalisation d’une première version utilisable du projet.

Pour la réalisation du projet nous avons dans un premier temps développé le programme en utilisant une Android Activity.

Le principe est simple :

1. Une recherche est effectuée pour récupérer les recommandations de livres en utilisant l’URL prédéfini (Google books) ;
2. Les articles sont ensuite classés selon les notes attribuées par les utilisateurs ;
3. Une image est ensuite récupérée pour chaque livre, avec une image par défaut s’il n’y en a pas ;
4. Le tire du livre, l’auteur, une image ainsi qu’une description de chaque article sont alors affichées sur l’écran sous forme de liste d’Adapters ;
5. L’utilisateur peut ensuite cliquer sur l’un d’entre eux pour être redirigé vers une site extérieur, ici une fenêtre de navigateur vers Google Books.

Les trois premiers points sont effectués en arrière plan dans des AsyncTask pour ne pas surcharger l’Activity principale. Les données fournies par GoogleBooks sont en JSON.

A noter qu’un ajout de nouvelles sources de données est possible en dupliquant ces AsyncTask et en choisissant l’URL vers le site de classement en question moyennant évidemment une tâche de désérialisation du JSON pour remplir notre structure de donnée DataBook.

Il est également possible pour l’utilisateur de choisir la langue d’affichage entre français, allemand et anglais, la langue par défaut étant celle du téléphone ou l’anglais si le téléphones est dans une langue non supportée.

Nous avons ensuite baculé l’affichage dans un Fragment pour permettre les rotations d’écran.

Au niveau des outils, la majeure partie du dévelopmment a été faite dans Android Studio et les tests dans le simulateur Genymotion qui est bien plus performant que l’AVD fourni avec le SDK.

Quelques mots sur l’implémentations

Tout d’abord la requêtes est faites au moyen d’une tâche qui fait la requêtes http sous forme de GET. Elle récupère les data et une autre tâche va ensuite tout mettre en forme et créer une liste d’Adapter. Ensuite on laisse l’affichage aller chercher les images pour les captions au fur et à mesure que l’affichage en a besoin. Là encore, c’est une tâche insynchrone qui s’en charge. Pour le coup nous avons rempli notre objectif d’une interface rapide.

L’avantage aussi de traiter de cette manière est qu’en cas de refactoring, il y a peu d’éléments à changer car l’affichage et la récolte de données sont découplés. De plus la recherche étant assez lente suivant le nombre d’éléments, la réactivité de l’application s’en ressentirait si elle allait chercher les images en même temps. Ce qui est d’autant absurde que dans les bonnes pratiques, il ne faut avoir en mémoire que les données dont on a besoin.

Lorsque l’on regarde le code, nous pouvons voir que notre conteneur de données pour l’affichage impémente l’interface Parelable. Ceci va permettre par la suite de rendre notre objet serialisable ; de même la classe BookDataCreator va servir à déserialiser les objets. Il s’agit de mécanisme qui vont permettre par la suite de passer au travers des Intent, cela permet entre autre de conserver les données lorsque l’on change l’orientation de l’appareil. Nous avons choisi cette marière parmi toutes les façons que l’on connaissait :

* Empêcher l’activité d’être détruite (c’est le mal)
* Empêcher la rotation (oui bien sûr…)
* Relancer une requête en sauvant les paramètres (à l’encontre de notre volonté d’avoir une application efficace)
* Utiliser les transactions avec un FragmentManager. Un peu lourd étant donné qu’il n’y a qu’un seul fragment.
* Rendre serialisable les données pour les transférer via des Intents. Cela nous a semblé propre et efficace. D’autant plus que cela nous permet une grande flexibilité puisqu’il s’agit uniquement des datas et non du contexte d’affichage.

Au niveau du JSON, l’API Google travaille avec des Volumes et un VolumeInfos contenu dans un Volume ce qui a été à la source d’un bug assez évident car portant à confusion mais pas forcément facile à identifier.

Les informations sont stockées dans des BookData qui sont insérer dans un ListArray qui est ensuite triés à l’aide d’un comparateur customisé travaillant sur la note donnée au livres. Et ce en vue de l’affichage pour que les mieux notés apparaiseent en premier.

## Les problèmes rencontrés

Pour réaliser ce projet, nous avions besoin d’une API gérant les recommandations de livres sur internet. Plusieurs API ont été identifiées :

* L’API du journal USA Today, qui traite principalement les best-sellers ;
* L’API du Guardian orienté sur les reviews  mais qui reste une très bonne API;
* L’API de Goodreads, la seule encore en XML et par très user friendly et très fermée ;
* L’API de Google ;
* L’API d’Amazon.

Contrairement aux autres, les résultats de Google sont très fournis mais l’API est difficile d’utilisation. En effet, son implémentation est tentaculaire et pour trouver les bonnes fonctions il faut parfois passer beaucoup de temps à tester les requêtes sur le site de Google Developer, où l’on peut trouver la documentation associée à cette API. Un mot sur la documentation de la Google API : cette dernière est parfois obscure et on ne comprend pas toujours bien comment s’articulent les paramètres. Concernant l’utilisation de recommandation, l’API offre une requête permettant de sortir une liste de recommandation mais il s’agit de celle de Google Play et donc comme il n’est pas possible de faire une recherche directement dedans si l’utilisateur n’a jamais rempli de préférences sur son compte Google les résultats sont généralement non pertinant voir complètement fantaisites. Et cela même lorsque l’on a déjà fait des recherches de livres avec GooglePlay. De plus, cela nécessite d’avoir accès au compte Google de l’utilisateur et nous voulons avoir une application non intrusive.

Les API du Guardian et de USA Today sont très bien faites et l’API publique permet de faire déjà un bon nombre de choses. Il serait simple à l’avenir de les inclures dans notre application. Néanmoins nous avons préférer les écarter parce que les résultats sont soit les best sellers soir des reviews. De plus on est limité à une seule langue l’anglais.

L’API ouverte de Goodreads est une catastrophe et peu de développement en vue de l’améliorer ne semlbe être fait vu l’activité sur les forums. Mais le site reste bon car il y a une communauté active et le site fonctionne là-dessus.

L’API d’Amazon à l’air très complète et complexe : cela aurait demandé beaucoup d’effort juste pour maîtriser l’API, d’autant plus qu’il a fallu consacrer beaucoup de temps à l’apprentissage de’Android.

Aucune de ces API ne fournit une recherche par genre. L’API de USA Today fournit les catégories des livres et on aurait pu tout charger sur le téléphone et faire un tri à ‘’la main’’ pour ne sortir que les catégories mais cela aurait été un peu inutile car le calssement se fait par volume de vente et non par la qualité des ouvrages. Si l’API de Google fournit un champ Category mais force est de constater qu’il est quasiment systématiquement vide ou encore il n’y a pas de consitence dans les labels choisis.

Pour des raisons nous avons donc opté pour l’API proposée par Google malgré tous les défauts qu’on peut y trouver mais il remplissait un maximum de nos critères : exhaustivité, différentes langues, résultats rapides et la plupart des ouvrages ont des notes fournies par des utilisateurs.

Nous avons consacré beaucoup de temps et d’energie sur cette partie et tâcher de trouver un outil qui puisse nous fournir les données voulues. Mais comme on peut le voir ceci n’est pas une sinécure et les API publiques ont l’air plutôt limitées.

Concernant Android, il s’agissait surtout d’apprendre à utiliser la plateforme et Eclipse constitue un gros point noir notamment en ce qui concerne la gestion des ressources (package R) et lint ne fonctionne presque jamais correctement. Parfois il perdait la liaison avec le SDK et sans raison refusait de fonctionner. En cours de développement nous sommes passés sur AndroidStudio qui il faut le dire fonctionne très bien ! D’autant plus que la version 1.0 vient de sortir. Une mention sur l’outil de refactoring qui est parfait. Il manquerait plus qu’il répercute aussi les changements entre les fichiers XML et Java. Cela nous a été très utile lorsque l’on est passé de la version Activity à la version Fragment.

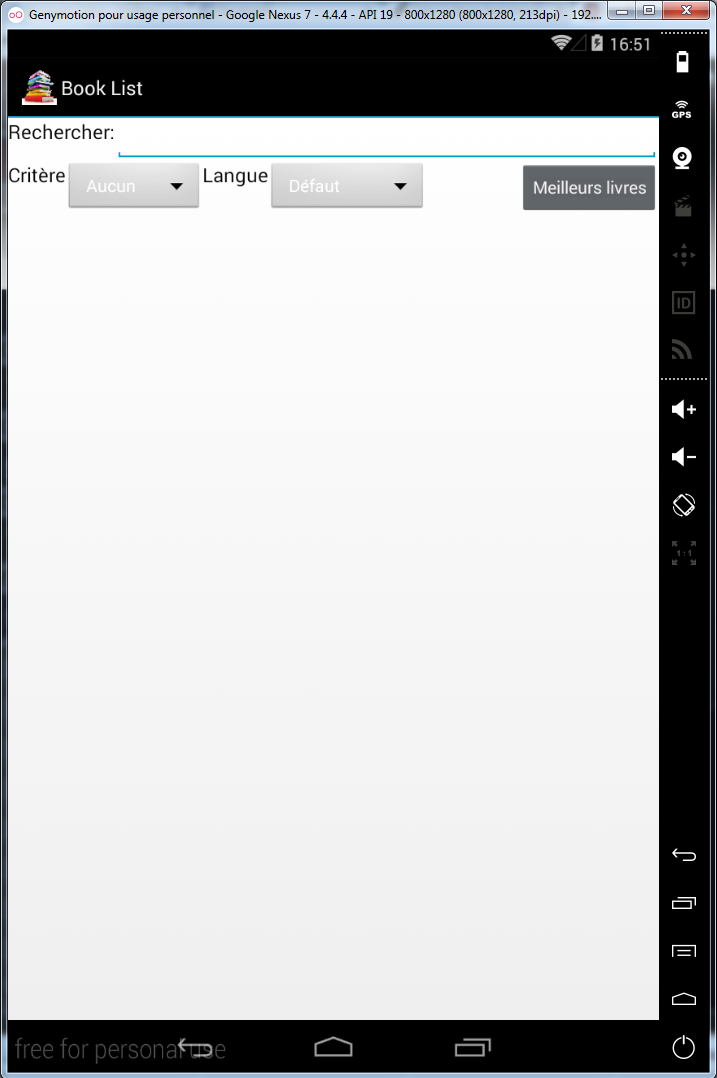
De plus, en passant de l’un à l’autre, le style a complètement changé et il y a encore quelques soucis de ce point de vue là notemment en ce qui concerne les styles.

Une autre difficulté est venue de la Legacy d’Android : en effet, il y a eu beaucoup de changement entre certaines versions du SDK qui font que lorsque l’on cherche à effectuer un type d’opération (par exemple la persistance des objects lorsque l’Activity ou le Fragment est détruit), les développeurs avaient pour habbitude de traiter le problème d’une façon et que par la suite cette possiblité a disparu. Et dans ce cas la documentation d’Android n’est pas capable de nous indiquer ce qui a changé et on passe alors beaucoup de temps à parcourir les exemples pour ce faire.

En parlant de cela, les tutoriaux sont pour la plupart dépassés ou pas à jour et se limitent à des choses extrêmement simples à partir desquelles il est bien souvent impossible de faire évoluer l’application.

## Mode d’emploi de l’application

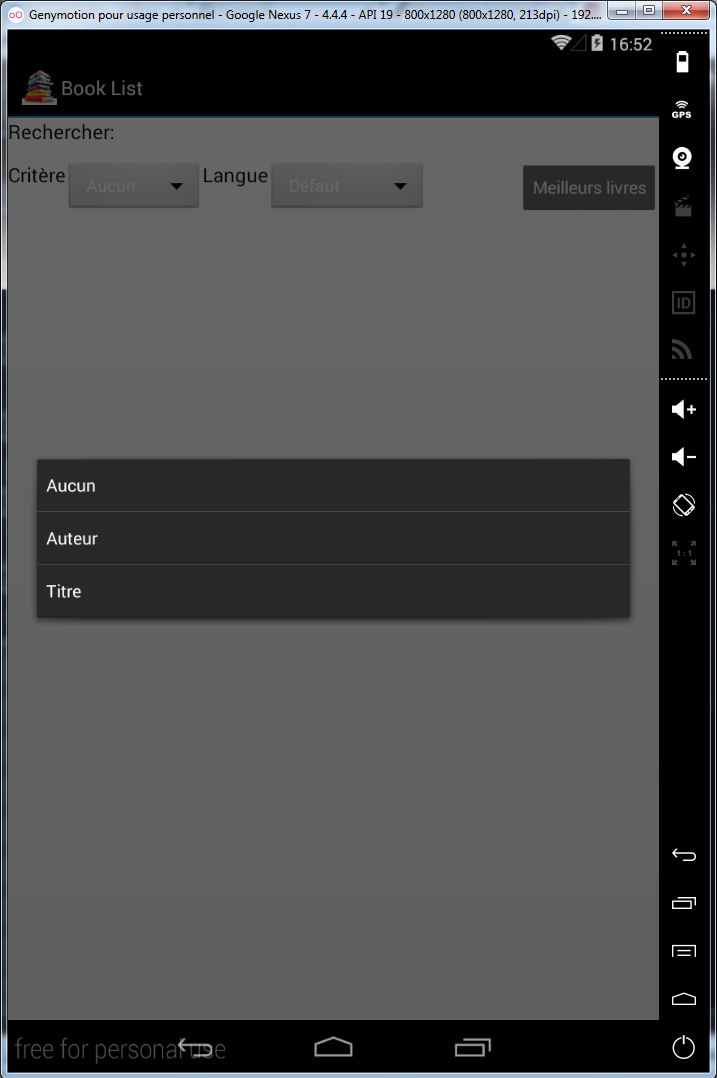
A l’ouverture de l’application nous avons la fenêtre suivante :



Le premier élément tout en haut correspond à une barre de recherche pour indiquer au programme un titre ou un auteur qui souhaite rechercher.

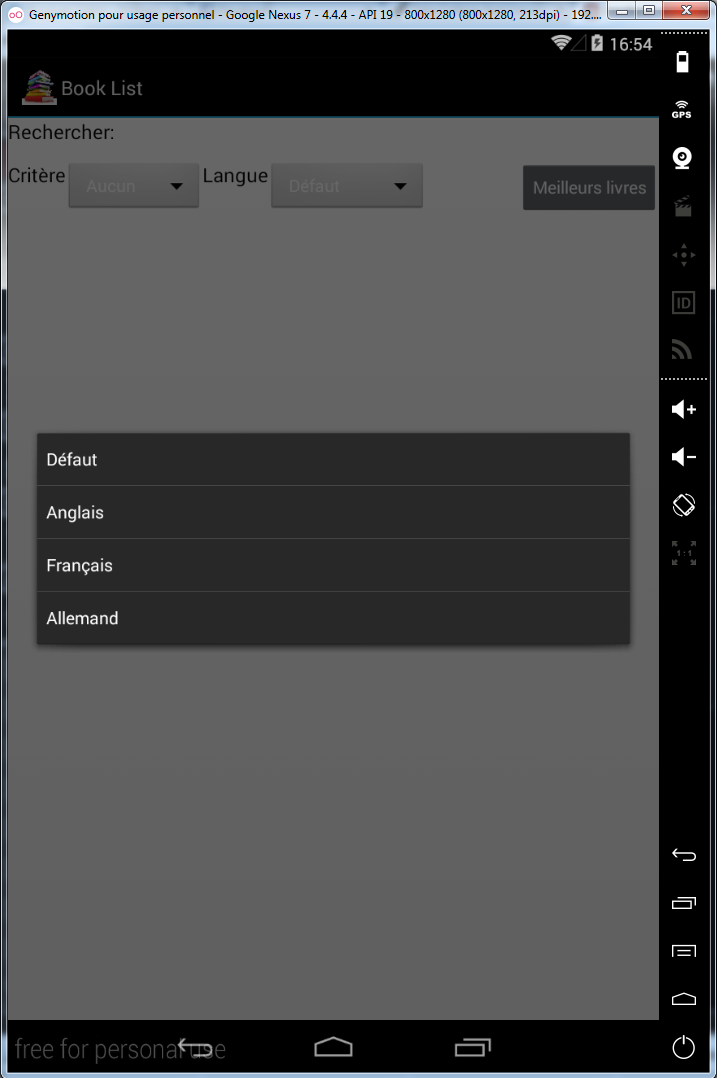
On peut distinguer également les deux menus déroulants en haut à gauche qui représentent les préférences utilisateurs, le plus à gauche correspondant au tri (par auteurs ou par titres), l’autre à la langue d’affichage. On remarque que ceux-ci sont positionnés sur Aucun et Défaut les paramètres par défaut du programme.

En utilisant le premier menu déroulant on accède au type de recherche, par auteur ou par titre :



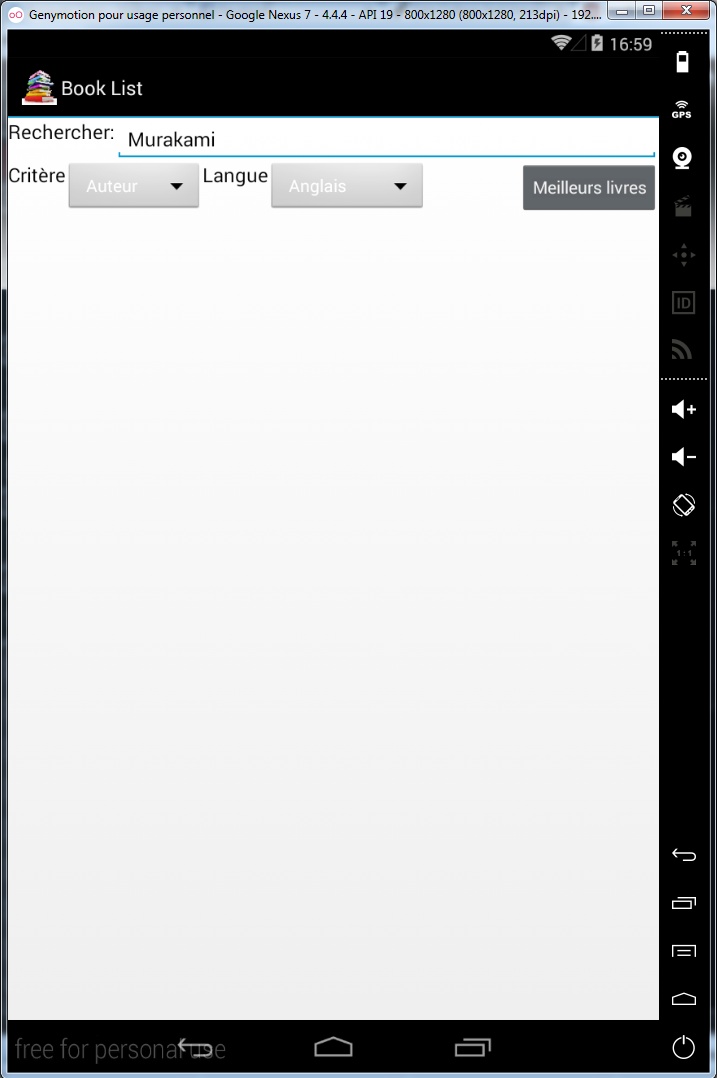
La valeur Aucun indique à la recherche de ne faire aucune discrimination. L’API ne fournit aucune information sur ce qui est vraiment recherché dans ce cas-là.

En utilisant le deuxième menu déroulant on accède à la selection de la langue, Anglais, Français ou Allemand :

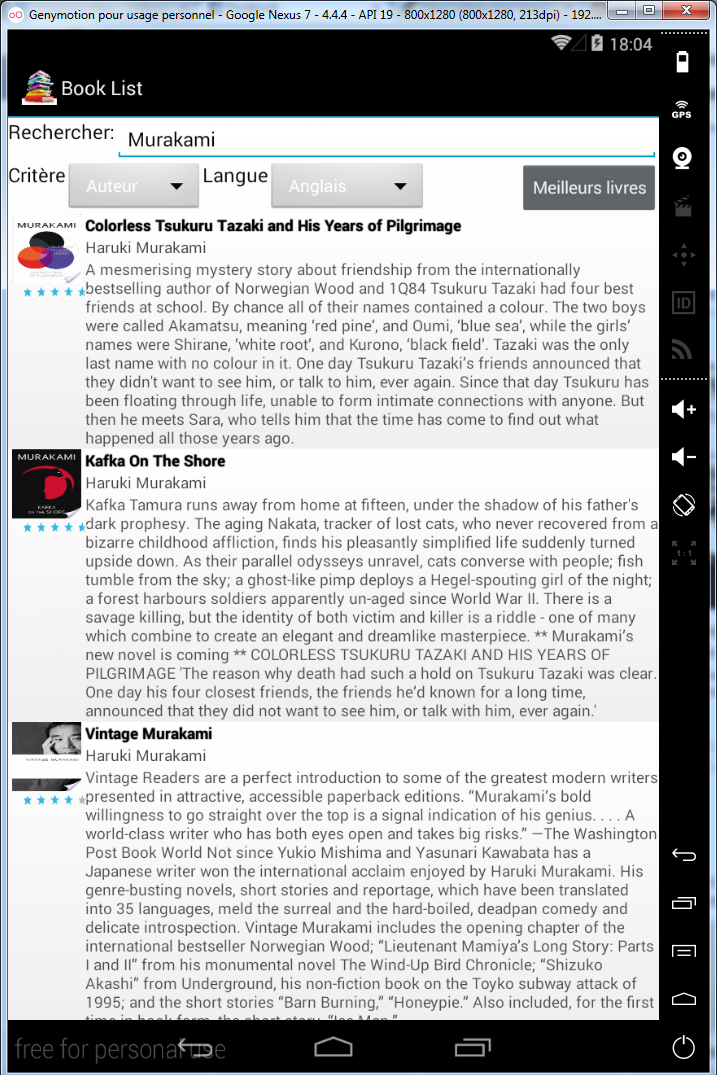


La valeur Défaut correspond ici à la langue du téléphone.

Prenons donc un exemple : nous recherchons l’auteur Murakami parmi les livres en Anglais :

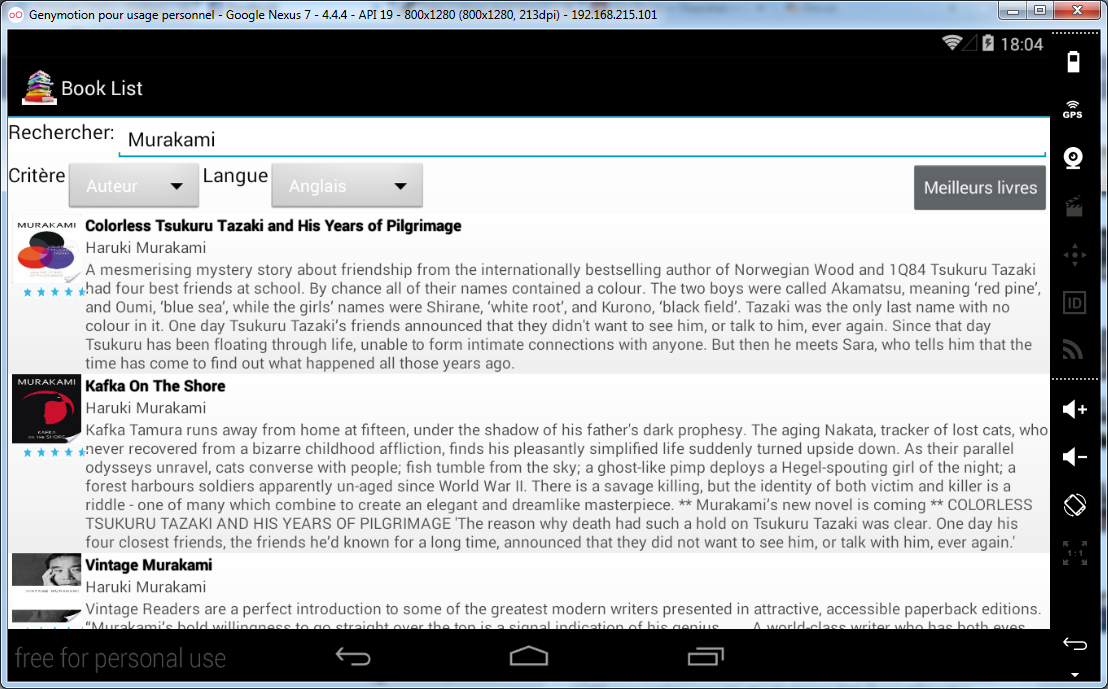


On obtient ainsi les recommandations suivantes :

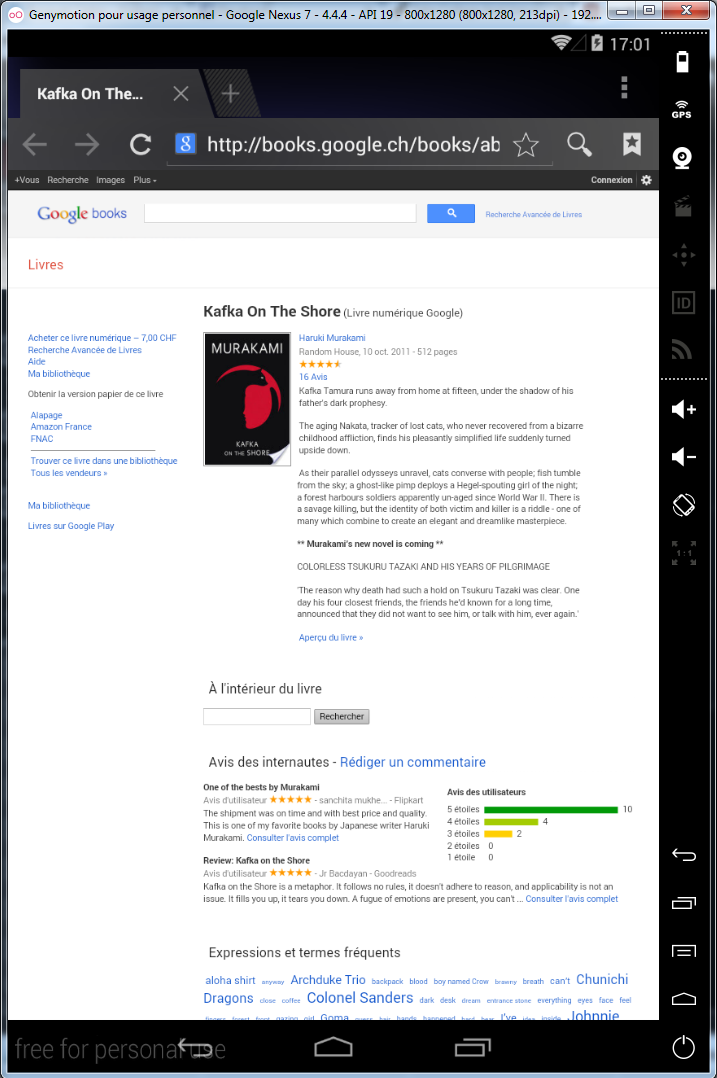


On remarque ici qu’il s’agit bien des livres écrits par Haruki Murakami, Ryu Murakami et et qu’ils sont bien en Anglais. On peut lire également la petite description fournie par le site (Google books).

Egalement, en cas de rotation de l’appareil on obtient le résultat suivant :



Enfin, en cliquant sur une des recommandation, pour l’exemple « Kafka on the Shore », on accède au site pour plus d’informations :



## Conclusion

Dans le cadre de ce projet, la conception et la recherche d’API s’est faite à deux. La partie concernant l’interface et la présentation des données a été réalisée par Matthieu et la partie communication avec l’API Google ainsi que la persistence des données a été réalisée par Cédric. La documentation a été rédigée par les deux.

Il a été intéressant de mettre en œuvre des éléments concrets pas vu ou alors survolés en cours afin d’avoir de bonnes fondations quant au développement Android.

Néanmoins, beaucoup de temps a été perdu à trouver une bonne API pour au final en conserver une qui est un peu boîteuse et non consistante. Typiquement certaines informations, comme les images, ne sont pas fournies par le JSON alors que si on se rend sur le site de GoogleBooks elles existent.

Nous aurions voulu proposer plus de fonctionnalités comme l’affichage automatique d’une sléection mais sans criètere de recherche il n’est pas possible d’avoir une liste de livre de la part de Google. Au final, il s’agit ici plus de mettre en pratique des éléments propres à Android que de commencer à se battre avec des formats de données. En effet, de ce point de vue la leçon apprise est que ce n’est pas toujours simple de trouver ce dont on a besoin et que parfois il faut faire avec.

Il aurait été possible de faire des recommandations bidons comme le font la plupart des applications mais nous nous sommes tenus à nos choix à savoir permettre à l’utilisateur d’avoir des informations précises à une demande précise qui soi rapide, légère, simple et peu intrusive dans son appareil.