1 Table des matières

2 Nouveautés Android

Source : http://en.wikipedia.org/wiki/Android_version_history

2.1 Version 3.0

Spécificité importante d'Android 3.0 - Honeycomb

- Disparition du bouton "Menu" remplacé par un équivalent dans l'ActionBar

2.2 Versions 4.0

Spécificités de Android 4.0 - Ice Cream Sandwich

Announced on 19 October 2011 and released on 14 November 2011

- **Unification des interfaces :** La branche 2 était aux smartphones ce que la branche 3 était aux tablettes.
- **Fonctionnalités**
  - Ability to access apps directly from lock screen
  - Face Unlock, a feature that allows users to unlock handsets using facial recognition software
  - A new typeface family for the UI, Roboto
  - Android Beam, a near-field communication feature allowing the rapid short-range exchange of web bookmarks, contact info, directions, YouTube videos and other data
  - WiFi Direct : Anciennement connu sous le nom Wi-Fi P2P, c'est un standard qui permet une connexion Wi-Fi entre appareils sans avoir besoin d'un point d'accès sans fil. Il est alors possible de transférer directement des données entre périphériques.

Plus d'infos : http://developer.android.com/about/versions/android-4.0-highlights.html

2.3 Android 4.1 & 4.2

Spécificités de Android 4.0 - Ice Cream Sandwich - Jelly Bean

4.1 : Announced at the Google I/O conference on 27 June 2012 and released on 9 July 2012

4.2 : Google was expected to announce Jelly Bean 4.2 at an event in New York City on 29 October 2012, but the event was cancelled due to Hurricane Sandy. The new version was announced with a press release instead of rescheduling the live event, under the slogan "A new flavor of Jelly Bean".

- **Fonctionnalités**
  - 4.1
    - Expandable notifications
    - The performance improvement involved "Project Butter" to create a fluid and "buttery-smooth" UI.
    - Shortcuts and widgets can automatically be re-arranged or re-sized to allow new items to fit on home screens
    - Bluetooth data transfer for Android Beam

- Google Wallet
- Google Now
- Stock Android browser is replaced with the Android mobile version of Google Chrome in devices with Android 4.1 preinstalled

o 4.2

- Gestion du mutli-comptes (uniquement sur tablette) + Protection parentale
- Photo Sphere panorama photos
- Keyboard with gesture typing (Swype)

- **SDK Android Tools**
  - o Editeur d'interface multi-config (pour tester et visualiser l'interface dessinée dans plusieurs configurations d'écran)
  - o Nouvelle version de l'UI Automator Test Framework (permet d'automatiser les tests unitaires au niveau des interfaces).

Plus d'infos : http://developer.android.com/about/versions/jelly-bean.html

2.4 Qu'attendre de la prochaine version ?

2.4.1 Nom de code

Puisque la convention veut que les noms de version soient donnés selon des noms de desserts depuis Cupcake, la prochaine version d'Android sera nommée d'après un dessert qui commence par «K». Selon la rumeur, il en sera probablement de "Key Lime Pie".

Source :

http://pocketnow.com/2012/07/03/android-guy-weekly-whats-next-after-android-jelly-bean-key-lime-pie

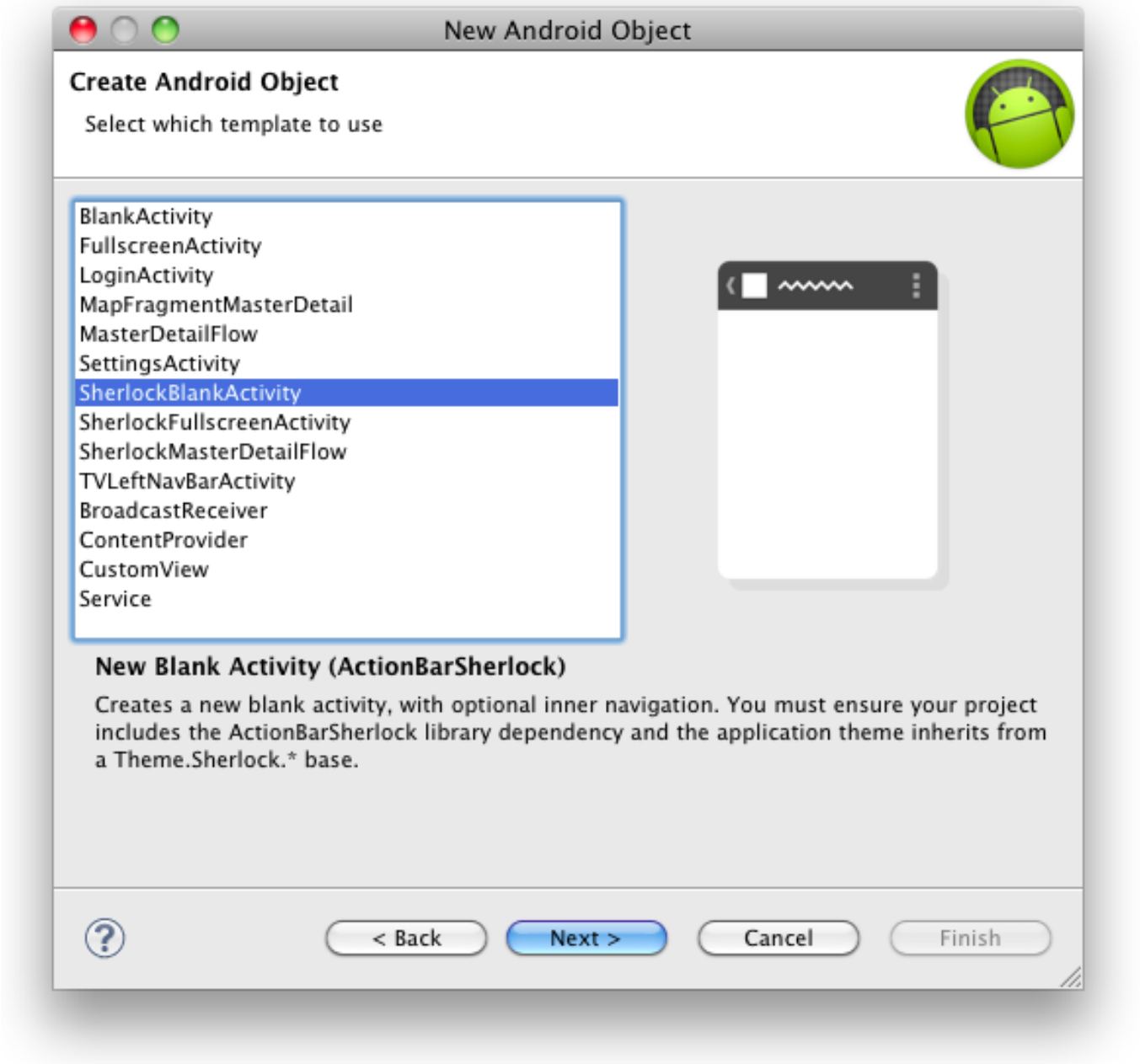
2.4.2 Développement

Concernant les améliorations de développement, il est possible d'activer le SDK Android Preview et l'ADT Preview comme expliqué sur http://tools.android.com/preview-channel.

2.4.3 Templates

Si on utilise l'ADT pour créer de nouveaux projets Android, on aura tous remarqué que l'on avait quelques options disponibles pour choisir le type d'UI avec lequel l'on peut démarrer.

Annoncé par Roman Nurik sur Google+, il est maintenant tout à faire possible de construire ses propres templates et les intégrer au sein du "Wizard" sur eclipse. Vous trouverez toute la documentation officielle sur le format et la syntaxe à respecter - de manière lisible mais temporaire - sur https://dl.dropbox.com/u/231329/android/templatedocs/index.html.



Des projets tels que l'on va découvrir dans la suite de cet exposé peuvent alors fournir des templates permettant aux développeurs à lancer leur projet plus facilement. Il existe d'ailleurs déjà une collection de templates disponibles à https://github.com/jgilfelt/android-adt-templates qu'il suffit d'ajouter au répertoire `<android-sdk-folder>/extras/templates/`.

Remarques : L'ADT ne gère pas les dépendances externes, en dehors de la librairie de support.

Source : http://www.androiduipatterns.com/2012/09/next-version-of-android-tools-will.html

3 Guidelines

3.1 Philosophie

3.1.1 Don'ts

- Recopier les interfaces utilisateurs depuis d'autres plateformes ;
- Sur-utiliser les fenêtres modales de progression et les boites de dialogue de confirmation (intrusif) ;
- Créer des interfaces rigides en utilisant des layouts en position absolues ;
- Utiliser les unités en pixels (px) mais plutôt en dp ou sp pour le texte, qui ne doit pas être trop petit ;
- Utiliser plus de quatre onglets.

3.1.2 Do's

- Créer des versions des ressources adaptées à une majorité des écrans ;
- Fabriquer de large et évidentes cibles, que ce soit des boutons ou des éléments de liste ;
- Gérer convenablement le cycle de vie d'une application ;
- Gérer le changement d'orientation ;
- Utiliser les ressources thèmes/styles, dimensions et couleurs pour réduire la redondance de code (http://developer.android.com/guide/topics/ui/themes.html).

3.2 Principes d'interfaces utilisateurs

3.2.1 Se concentrer sur l'utilisateur

L'application doit être ciblée : "user-first".

3.2.2 Rendre les choses visibles

ActionBar et Dashboard View.

3.2.3 Montrer les effets appropriées

Les effets d'une action doivent être visible soit textuellement en affichant une notification par Toast (de manière non intrusive) soit visuellement en utilisant les 4 états de sélection des éléments graphiques :

```
<selector>
      <item android:drawable=”@drawable/foo_disabled”
            android :state_enabled=”false”>
      <item android:drawable=”@drawable/foo_pressed”
            android :state_pressed=”true”>
      <item android:drawable=”@drawable/foo_focused”
            android :state_focused=”true”>
      <item android:drawable=”@drawable/foo_default”>
</selector>
```

En raison de la nature de l'interface tactile (il n'y pas de survol de souris ou d’info-bulle), les utilisateurs devront compter sur l'indication visuelle de l'état de l’élément d'interface utilisateur.

3.2.4 Être prévisible

Respecter la “pile d’activité” et utiliser l’affordance à bon escient : ce qui peut être cliqué doit avoir l’apparence de quelque chose de cliquable.

Eviter à tout prix la confusion :

- http://androiduiux.com/2012/08/23/android-uiux-tips-1/
- http://androiduiux.com/2012/09/04/android-uiux-tips-2/
- http://androiduiux.com/2012/09/27/android-uiux-tips-3/
- http://androiduiux.com/2012/11/22/android-uiux-tips-4/

3.2.5 Être tolérant aux erreurs

Désactiver les éléments lorsqu’approprié, limiter le nombre d’actions irréversibles et préférer la barre d’undo aux boites de dialogue.

“If an error is possible, someone will make it”

Donald Norman (author of “The Design of Everyday Things”)

3.3 Design patterns

3.3.1 ActionBar

L’ActionBar, que l’on pourrait traduire par “barre d'actions” est une barre placée dans la partie supérieure de chaque écran et est généralement persistante tout au long de l'application.

Elle offre l’accès à plusieurs fonctionnalités primordiales :

- Rend les actions importantes visibles ;
- Prend en charge la navigation cohérente et le changement de vue ;
- Réduit l'encombrement en fournissant une icône de menu pour les actions secondaires.

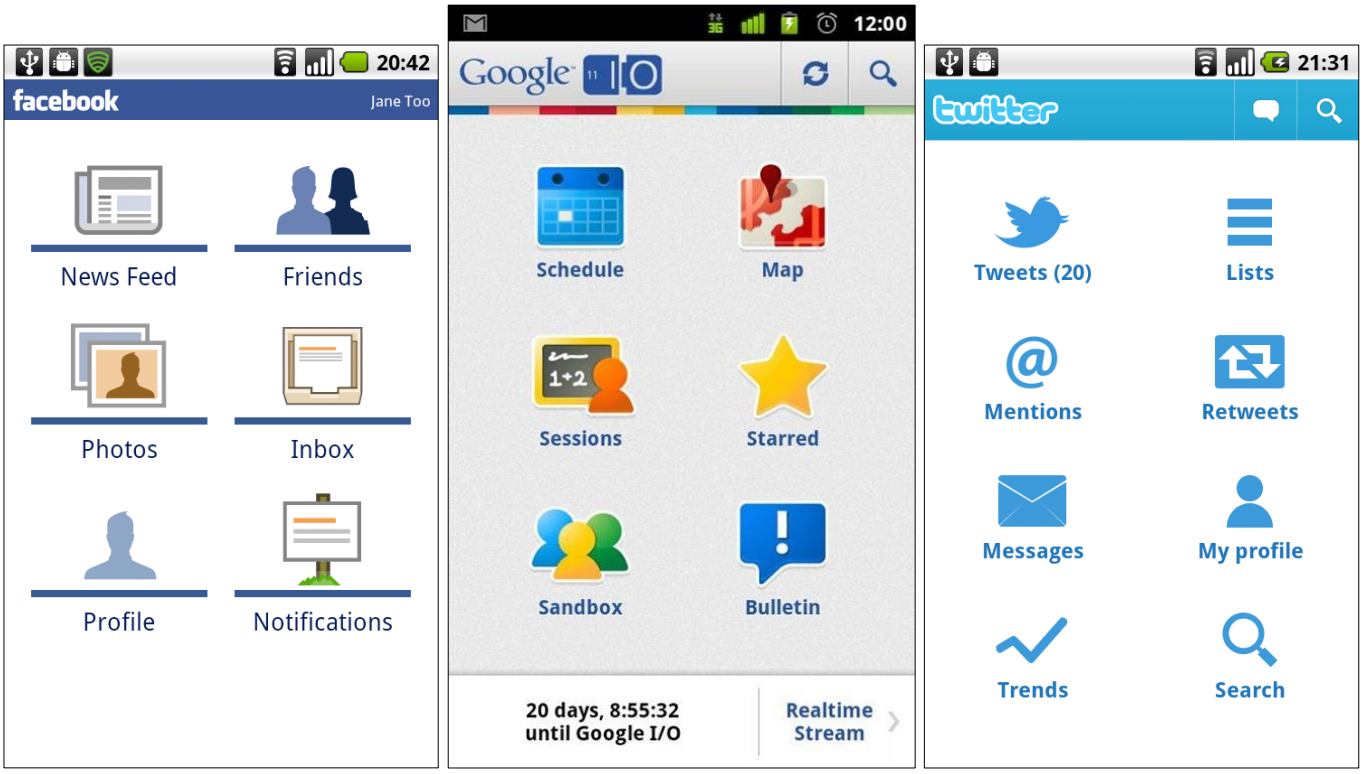
La barre d'action est l'un des éléments de design les plus importants que l'on se doit d'incorporer. Elle est l'endroit idéal pour des actions contextuelles concernant l'écran sur lequel la barre d'action est affichée.

Si un écran est le plus élevé dans une application, il ne devrait pas présenter un bouton "UP". Le bouton de retour est utilisé pour naviguer dans l'ordre chronologique inverse, à travers l'historique d'écrans au travers desquels l'utilisateur a récemment navigué. Il est généralement fondé sur les relations temporelles entre les écrans, plutôt que sur la hiérarchie de l'application comme l'icône "UP".

3.3.2 Dashboard

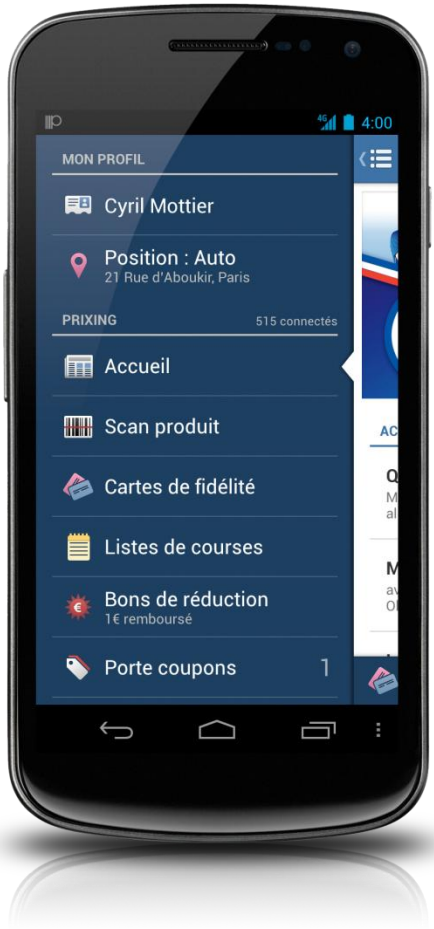
Une fois l'application installée, il faut que l'utilisateur soit opérationnel le plus rapidement possible en évitant un long processus d'inscription car les utilisateurs détestent devoir taper sur leur mobile.

L'activité de départ doit être la plus simple possible et devrait montrer l'ensemble des principales fonctionnalités disponibles en prêtant particulièrement attention à ce que l'interface soit esthétique et cohérente car c'est la première impression, comme le dashboard de l'application Google I/O ou Google+ dont vous trouverez les sources sur http://code.google.com/p/iosched/.



Cependant, il est possible que l'application ne possède qu'une unique interface. Il est alors bon d'envisager de leur proposer de suivre un tutorial graphique permettant de les éduquer sur la façon d'utiliser l'application, comme Winamp ou Google Goggles.

3.3.3 Sliding menu + Bezel swipe



L'activité de Dashboard est maintenant "l'ancienne façon" de réaliser une interface d'accès aux fonctonnalités de l'application. Et si l'on pensait à un menu coulissant : un panneau qui glisse de la gauche de la zone du contenu principal, révélant une vue verticale avec un défilement indépendant.

Soit un pattern UI permettant une navigation entre les différentes vues composant l'application sans devoir retourner à un quelconque dashboard.

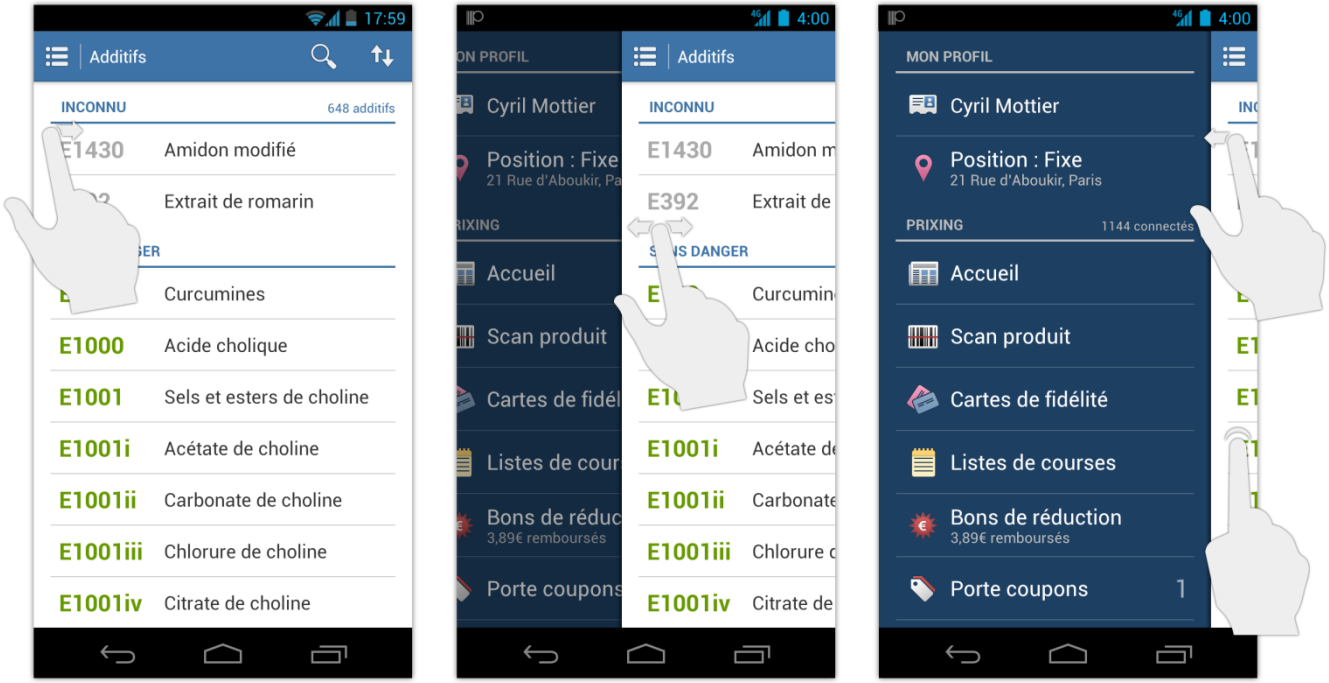
Cependant, il n'existe pas encore de réelle documentation sur le site des guidelines Android :

- Fly-In App Menu
- Slide-out Menu
- Side Navigation
- Slide Menu
- Ribbon Menu
- Sliding Navigation Bar

L'implémentation de chacune des applications est très différente. Le "Look And Feel" est différente, mais l'approche sous-jacente est la même. L'idée est de fournir aux utilisateurs des raccourcis rapides vers la partie la plus importante de l'application sans avoir à quitter l'écran sur lequel ils sont. Les différences visuelles d'implémentations sont très apparentes. L'application Google+ glisse la navigation au dessus de l'interface utilisateur, tandis que les autres déplacent l'interface utilisateur sur le côté.

Quoiqu'il en soit, les avantages restent les mêmes :

- **Responsive** : Ce menu volant s'adapte convenablement autant aux smartphones qu'aux tablettes !
- **Dynamique** : Le contenu pouvant défiler, il peut alors grandir indéfiniment !
- **Accès direct et rapide** : S'ouvre plus rapidement qu'un dashboard puisqu'il n'y a aucune transition entre activités nécessaires !

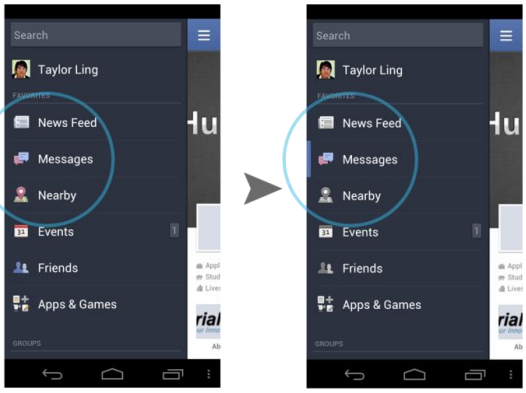


Oui mais attention, ce n'est pas pour autant qu'il faut absolument l'utiliser !

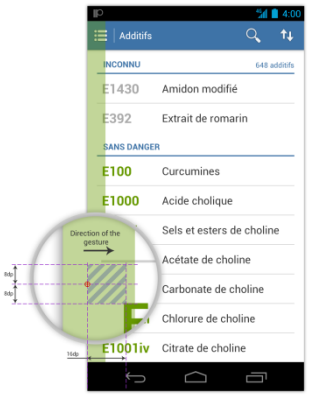
En outre, si vous décidez d'utiliser l'icône "UP", vous devez vous assurer qu'il n'ouvre pas la navigation latérale. En effet, si on utilise la même flèche gauche que la fonctionnalité "UP" pour ouvrir le slide menu, cela brise le guide de style fondamentale de conception de l'interface utilisateur : il doit toujours être évident ce que fait un bouton sur une interface. L'icône "UP" peut soudainement avoir deux fonctions très différentes et l'utilisateur ne peut pas dire de quoi il s'agit, sans essayer le bouton.

Indicateur

Il est important d'afficher clairement dans le menu, la catégorie correspondant à l'activité affichée.



Bezel Swipe



Permet à l'utilisateur de révéler le menu de navigation en glissant ses doigts depuis l'extrême gauche vers la droite et inversément pour le fermer.

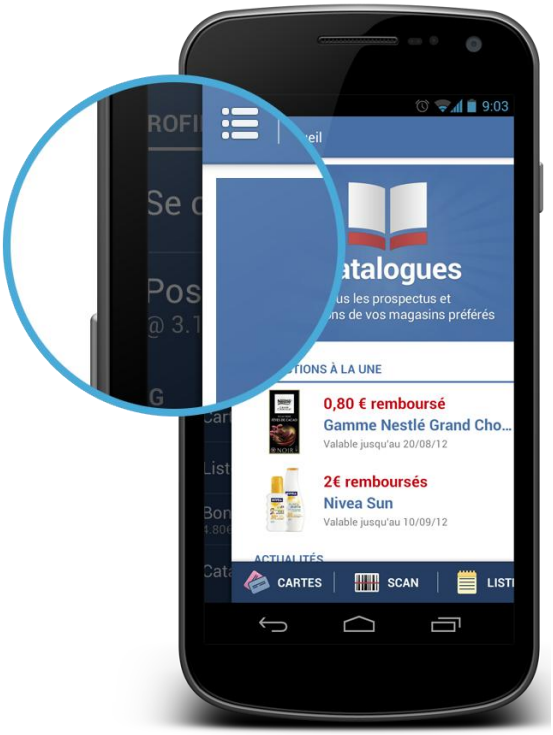
Cela peut provoquer un problème de cohérence puisque le glissement de doigt est un geste déjà réservé pour se déplacer entre des onglets, si mal implémenté.

Améliorer la découverte

Le tableau de bord nécessite aux utilisateurs de revenir à l'écran d'accueil de l'application pour naviguer vers d'autres parties de l'application.

Pour permettre à l'utilisateur de connaître les interactions utilisateurs, il est nécessaire d'améliorer la détectabilité de ces nouveaux modèles :

- L'introduire dans le tutoriel au premier lancement et permettre à l'utilisateur de jouer avec elle ;
- Faire la navigation latérale un peu plus évidente lorsque le premier utilisateur exécutant l'application - un excellent exemple est l'application Prixing qui fait sautiller le panneau de navigation latérale lorsque le premier utilisateur entre dans l'application, créant la curiosité de l'utilisateur ;
- Dans tous les cas, utiliser un bon indicateur pour représenter la navigation latérale.



Il n'y a pas de solution parfaite pour le problème de découverte, mais ce sont des améliorations possibles.

Récapitulons, il existe 3 systèmes de navigation :

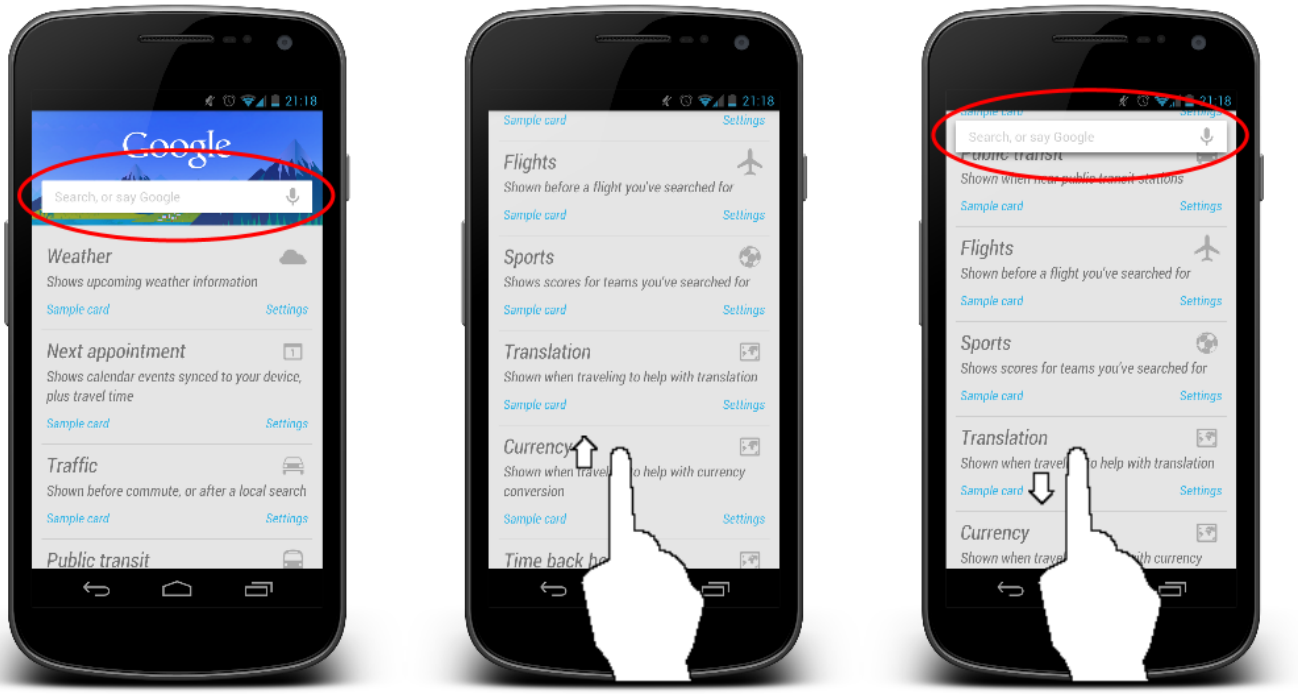
- **Onglets** : simple à mettre en œuvre, accès facile mais pour 4 fonctionnalités max.
- **Dashboard** : visuellement attirant, convenable pour entre 5 à 9 fonctionnalités.
- **Menu latéral** : pas évident à utiliser, contenu dynamique et responsive.

2 options pour le menu latéral :

- Librairie : https://github.com/jfeinstein10/SlidingMenu
- Custom Implementation
  - http://android.cyrilmottier.com/?p=658
  - http://android.cyrilmottier.com/?p=701
  - http://android.cyrilmottier.com/?p=717

3.3.4 Quick Return

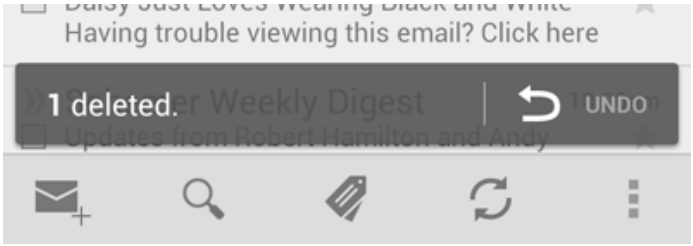
Il s'agit d'un élément de la vue qui se détache de celle-ci et s'affiche indépendamment sur le haut de l'écran lors d'un défilement vertical du haut vers le bas. C'est une technique assez pratique pour les barres de recherche, par exemple.



Plus d'infos : http://code.google.com/p/romannurik-code/source/browse/misc/scrolltricks

3.3.5 UndoBar

Il s'agit d'une barre qui apparaît après qu'une action potentiellement destructrice ou lourde soit effectuée (par exemple, l'archivage ou la suppression d'un e-mail) et vous propose brièvement la possibilité d'annuler la dernière action.



Plus d'infos : http://code.google.com/p/romannurik-code/source/browse/misc/undobar

3.4 Outils dédiés

Plus d'infos sur les guidelines à respecter : http://developer.android.com/design/index.html

- http://www.androiduipatterns.com/
- Android UI Patterns App
- http://www.androidviews.net/
- 9-patch

3.4.1 Prototyping tools

- Crayon + feuille
- Prototyping stencils for Pencil
- Android Asset Studio - Générateur de ressources graphiques

3.5 Sources

- http://fr.slideshare.net/AndroidDev/android-ui-design-tips
- http://fr.slideshare.net/mobilegui/excellence-in-the-android-user-experience
- https://speakerdeck.com/electryc/ux-design-best-practices
- http://androiduiux.com/2012/06/15/side-navigation-ui-pattern-in-android/
- http://androiduiux.com/2012/07/27/side-navigation-ui-pattern-in-android-part-2/

4 Développement

4.1 Fragments

Un Fragment est comme une mini-Activity (portion d'une activité) qui ne peut pas vivre en dehors d'une Activity. Un des aspects pratiques du Fragment est qu'il peut être facilement réutilisé d'une Activity à l'autre.

Pour créer une interface utilisateur dynamique (multi-fenêtre) sur Android, on a besoin d'encapsuler les composants d'interface utilisateur et les comportements d'activité en modules que l'on peut échanger parmi différentes activités. On peut créer ces modules avec la classe Fragment, qui se comporte un peu comme une activité imbriquée qui peut définir son propre layout et gérer son propre cycle de vie.

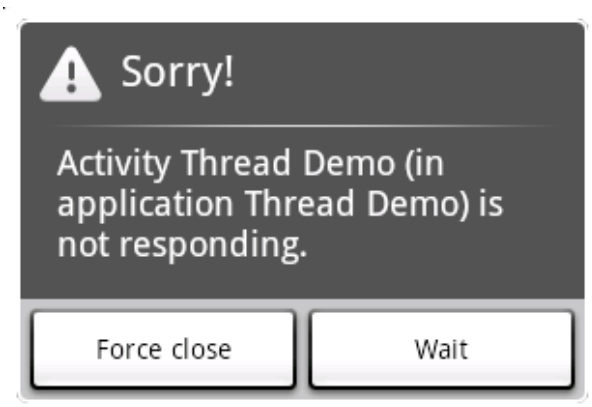
Bref, on peut combiner de multiples fragments dans une seule activité pour construire une interface multi-fenêtre et réutiliser un fragment dans de multiples activités. Un fragment doit toujours être intégré dans une activité et le cycle de vie du fragment est directement affecté par le cycle de vie de l'activité l'hébergeant. On peut réutiliser des fragments dans des configurations de layout différents pour optimiser l'expérience utilisateur en fonction de l'espace disponible à l'écran.

Plus d'infos : http://developer.android.com/training/basics/fragments/creating.html

4.2 AsyncTask

Comme son nom l'indique, AsyncTask est une classe qui permet de créer des tâches qui s'exécutent de manière asynchrone et en arrière-plan.

Ce procédé est encore plus important, car Android affiche un message d'erreur quand le thread principal (UI Thread car il gère également l'affichage des graphismes) est bloqué depuis trop longtemps.



Cette classe rend vraiment facile ce procédé, et il n'est pas nécessaire de s'occuper de threads. Elle doit être utilisée pour des traitements qui durent quelques secondes tout au plus, d'autres API's sont disponibles pour les tâches qui durent une très longue période.

4.2.1 Fonctionnement

Afin de mettre en place une AsyncTask, il faut créer une classe qui l'étendra. 4 méthodes peuvent dès lors être overridées :

[optionnelle] *`onPreExecute()`* : Appelé avant la tâche (initialisations nécessaires) par le UI Thread (principal).

[obligatoire] `doInBackground (Param... params)` : Appelé par le thread dédié à la tâche juste après que *onPreExecute()* ait fini. Les paramètres passés à new MyAsyncTask(param1, param2, param3)

[optionnelle] *`onPogressUpdate(Progress... progress)`* : Appelé par l'UI Thread sur demande, permet d'afficher une forme de

[optionnelle] *`onPostExecute(Result result)`* : Appelé par le UI Thread après la fin du thread dédié, le résultat est passé en paramètre.

Attention aux 3 méthodes Pre, progress, Post car elles s'exécutent sur l'UIThread et doivent donc être légères.

Android nous assure que les appels de ces méthodes sont synchrones, il n'est donc pas nécessaire de faire des vérifications dans la méthode doInBackground pour voir si tous les éléments sont bien initialisés.

Cette classe est générique et doit être spécifiée au moment de l'extend

```
private class MyTask extends AsyncTask<Void, Void, Void> { ... }
```

- Params : Type de paramètre envoyé à la tâche ;
- Progress : L'unité de progression de la tâche ;
- Result : Le type de résultat renvoyé par la tâche.

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
     protected Long doInBackground(URL... urls) {
         int count = urls.length;
         long totalSize = 0;
         for (int i = 0; i < count; i++) {
             totalSize += Downloader.downloadFile(urls[i]);
             publishProgress((int) ((i / (float) count) * 100));
             // Escape early if cancel() is called
             if (isCancelled()) break;
         }
         return totalSize;
     }

     protected void onProgressUpdate(Integer... progress) {
```

```
        setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}
```

4.2.2 Évolution

Les tâches asynchrones ont subi pas mal de changements au fil des versions d'Android. Les asynctasks sont arrivées à la version 1.5 (Cupcake), dans laquelle les tâches asynchrones étaient sérialisées, c'est à dire exécutées les unes après les autres. Dans la version 1.6, les tâches asynchrones étaient exécutées par un pool de threads, permettant ainsi l'exécution parallèle de plusieurs tâches. L'ordre d'exécution de ceux-ci était dès lors impossible à prédire. Dans la version 3 (HoneyComb), les développeurs d'Android on fait marche arrière et sont revenu à un seul et unique thread exécutant les tâches les unes après les autres. L'explication donnée pour ce retour en arrière est que la programmation parallèle était très compliquée et provoquait la plupart des erreurs d'exécution.

Une méthode spécifique de la classe AsyncTask permet une exécution parallèle, mais l'utilisation de celle-ci est fortement déconseillée pour les problèmes cités ci-dessus (executeOnExecutor dans lequel un objet de type Executor est passé, c'est-à-dire un thread indépendant).

```
new MyAsyncTask(....).executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, ....);
```

Remarque : Lors d'une rotation d'écran, une application est tuée, puis redémarrée "dans le bon sens", si une AsyncTask était en cours, elle est également stoppée. Il est donc nécessaire de tenir compte de cela et de sauvegarder son état dans le ????

4.3 ListView : conseils de performance

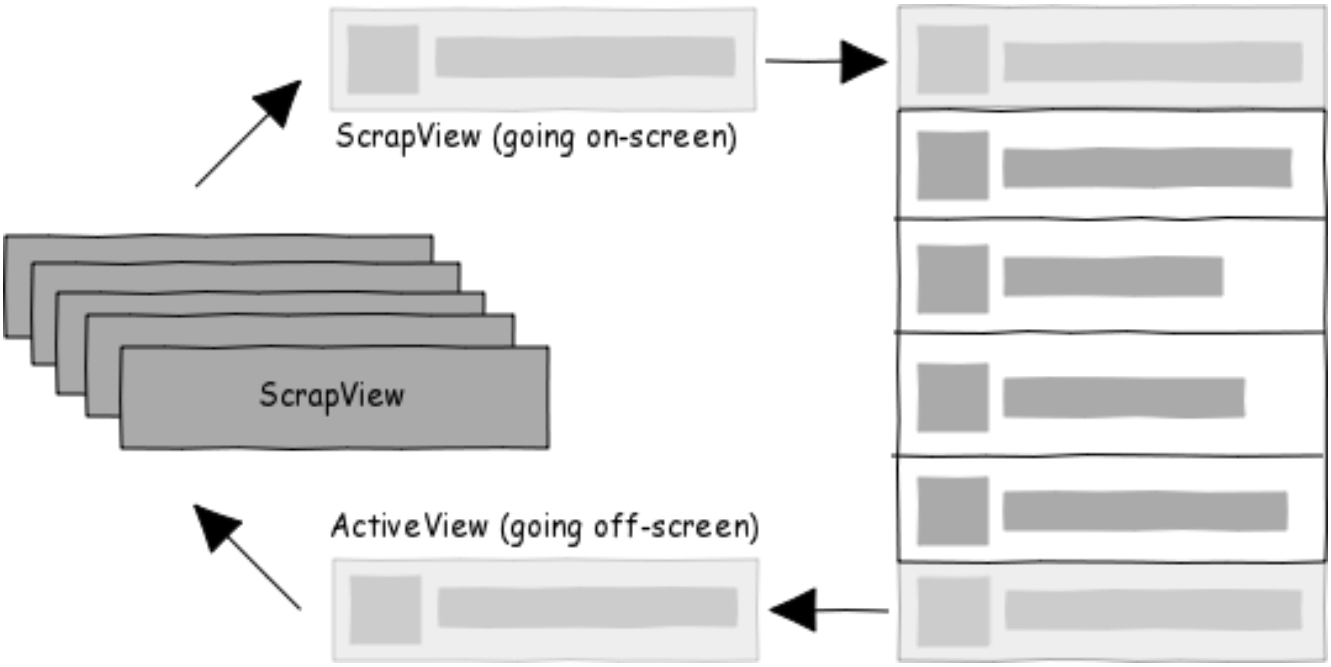
Une ListView est conçu pour être évolutive et performante. En pratique, cela signifie essentiellement :

- Le moins d'"inflation" possible ;
- Exposer uniquement les lignes qui sont (ou sont sur le point de devenir) visible à l'écran.

Les inflations de layout sont des opérations coûteuses car cela implique de passer dans un arbre de blocs XML. Une ListView résout ce problème en recyclant les vues non visibles dites "ScrapViews".

Une ListView utilise un recycleur de vue pour continuer à ajouter des vue recyclées en-dessous ou au-dessus de la vue courante et déplace les vues actives dans un pool recyclables lorsqu'elles se déplacent hors de l'écran pendant le défilement.

De cette façon, une ListView a uniquement besoin de garder en mémoire suffisamment de vues que pour remplir l'espace alloué au layout.



ListView “inflate” dynamiquement et recycle les vues lors du défilement. Il est donc essentiel de coder la méthode getView() aussi légère que possible.

Plus d’infos : http://lucasr.org/2012/04/05/performance-tips-for-androids-listview/

4.3.1 View recycling

Chaque fois qu’une ListView doit montrer une nouvelle ligne à l'écran, elle appelle la méthode getView () de l'adaptateur. Comme vous le savez, getView () prend trois arguments: la position de la ligne, une convertView et la ViewGroup parent.

L'argument convertView est essentiellement une "ScrapView" comme décrit précédemment. Elle aura une valeur non nulle lorsque ListView vous demande de recycler le layout de la ligne. On doit alors simplement mettre à jour son contenu au lieu d’inflate un nouveau layout pour la ligne.

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView = mInflater.inflate(R.layout.your_layout, null);
    }

    TextView text = (TextView) convertView.findViewById(R.id.text);
    text.setText("Position " + position);

    return convertView;
}
```

Les développeurs peuvent simplement mettre à jour le contenu de vues recyclés au lieu d’inflate toutes les lignes.

4.3.2 View Holder pattern

Le pattern ViewHolder consiste à réduire le nombre d’appels à findViewById() dans getView().

```
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;

    if (convertView == null) {
        convertView = mInflater.inflate(R.layout.your_layout, null);

        holder = new ViewHolder();
        holder.text = (TextView) convertView.findViewById(R.id.text);

        convertView.setTag(holder);
    } else {
        holder = convertView.getTag();
    }

    holder.text.setText("Position " + position);
    return convertView;
}

private static class ViewHolder {
    public TextView text;
}
```

4.3.3 Async loading

Il est possible d'afficher des images sur chaque ligne d'une ListView. Lorsque celles-ci se situe au sein de l'application, il n'y a pas de problème puisqu'elles sont en cache. Par contre, si on désire les charger depuis la carte SD ou depuis Internet, il est préférable de ne pas les charger toutes directement car cela bloquera le thread chargé de l'affichage.

Plus d'infos : http://lucasr.org/2012/04/05/performance-tips-for-androids-listview/

4.3.4 Pour aller plus loin

Plus d'infos : http://www.youtube.com/watch?v=wDBM6wVEO70

GreenDroid :

- ListView Tips & Tricks #1: Handling emptiness
- ListView Tips & Tricks #2: Sectioning your ListView
- ListView Tips & Tricks #3: Create fancy ListViews
- ListView Tips & Tricks #4: Add several clickable areas
- ListView Tips & Tricks #5: Enlarged touchable areas

4.4 Divers

4.4.1 Passage de business object entre activités

- Passage de l'id d'un object en base de données : c'est bête puisqu'on a déjà chargé l'objet !
- La classe pourrait implémenter l'interface `Serializable`. Il s'agit d'une solution qui fonctionne parfaitement mais ce n'est pas une solution optimale.

- Enfin utiliser l'interface Android dédiée `Parcelable` :

```
public class ExampleParcelable implements Parcelable {
   private String stringValue;
   private Integer integerValue;
   private char charValue;
   private boolean boolValue;
   private Date dateValue;

   @Override
   public int describeContents() {
       return 0; // 99.9% of the time you can just ignore this
   }

   @Override
   public void writeToParcel(Parcel dest, int flags) {
       dest.writeString(stringValue);
       dest.writeInt(integerValue);
       dest.writeInt(charValue);
       dest.writeInt(boolValue ? 1 : 0);
       dest.writeLong(dateValue.getTime())
   }

   public ExampleParcelable(Parcel parcel) {
       // The only important thing is to read them in the same
       // order as you wrote them (take a look at writeToParcel)
       stringValue = parcel.readString();
       integerValue = parcel.readInt();
       charValue = (char) parcel.readInt();
            boolValue = parcel.readInt();
       dateValue = new Date(parcel.readLong());
   }

   public static final Parcelable.Creator<ExampleParcelable> CREATOR = new
Parcelable.Creator<ExampleParcelable >() {

      @Override
      public ExampleParcelable createFromParcel(Parcel source) {
         return new ExampleParcelable(source);
      }

      @Override
      public ExampleParcelable[] newArray(int size) {
         return new ExampleParcelable[size];
      }
   };
}
```

http://devk.it/proj/parcelabler/ permet de générer le code nécessaire pour vous !

Plus d'infos :

http://vuknikolic.wordpress.com/2012/06/28/passing-objects-from-one-activity-to-another/

4.4.2 Où stocker correctement les fichiers

Les développeurs ont pour mauvaise habitude de stocker des fichiers directement à la racine de la carte SD.

```
File getExternalFilesDir(String type)
```

- Si vous passez une valeur nulle, l'objet File pointe vers le répertoire des fichiers ;
- Si vous ajoutez unn'importe quel constantes de répertoires de la classe Environment, vous obtiendrez un objet File pointant vers un sous-répertoire de votre répertoire de fichiers ;
- Si le répertoire n'existe pas encore, Android le crée pour vous ;
- Si le support externe n'est pas monté, la méthode renvoie null.

Remarque : Cette méthode n'a été introduite qu'avec l'API de niveau 8 (Android 2.2 Froyo).

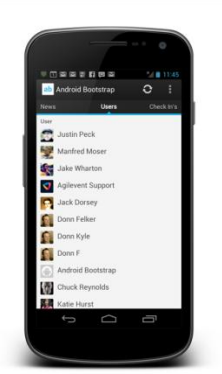
Plus d'infos : http://www.grokkingandroid.com/how-to-correctly-store-app-specific-files-in-android

5 Bootstrapping

5.1 Android Bootstrap

Comme son nom l'indique, Android Bootsrap vous permet de lancer rapidement une application Android basique sans devoir passer par une étape lourde de configuration. Le code de cette application, évidemment OpenSource, est disponible sur Github et met en place quelques-unes des fonctionnalités et des librairies les plus utilisées.

5.1.1 Installation



Afin de mettre en place rapidement le projet, elle tire parti des repository Maven (Package *Maven Integration for Eclipse*). Le connecteur Maven pour Android *(Android Configurator for M2E)* est également requis.

Relativement simple, l'utilisation d'Android Bootstrap est facilitée par l'utilisation de Maven (et de son plugin Android for Maven). Une erreur assez fréquente également est l'oubli de setter la variable d'environnement $ANDROID_HOME.

Après import, vous possédez donc une application basique totalement fonctionnelle. Il ne vous reste plus alors qu'à vous concentrer sur la logique de votre application. Nous verrons quelques-unes de ces librairies plus en détail plus tard, mais nous pouvons déjà vous citer : Fragments (mini activité réutilisable), Fragment Pager (Slider d'un Fragment à un autre), Account Manager (Connexion pour page protégée), android-maven-plugin, RoboGuice 2, ActionBarSherlock 4, ViewPagerIndicator (Permet d'indiquer plus clairement avec des onglets qu'il y a moyen de slider entre fragments), http-request, GSON, Robotium, ainsi que la consommation simplifiée d'API WebServices.

5.1.2 L'application résultante

L'application résultante et exécutable immédiatement sur votre téléphone est bien sûr basique, mais propose des fonctionnements fort intéressants tels que

- L'accès aux données protégées par un mot de passe ;
- L'authentification via un webservice ;
- Les requêtes et le chargement asynchrone de données ;
- Une structure de base propre ;
- Des listes riches et personnalisées (affichage d'une image) ;

L'application est téléchargeable sur le Play Store.

Le site propose même (en version *très beta*) la génération d'une application préconfigurée avec le nom et le nom de package désigné.

Ce projet vous permettra donc de mettre en place très rapidement une application basique avec des librairies fonctionnelles et configurées parmi les plus populaires et qui vous permettront de continuer votre application sur de bonnes bases.

5.2 AndroidKickstartR

Android KickStartr reprend le même concept. Mais là où AndroidBootstrap impose une multitude de librairies (dont certaines que vous n'utiliserez peut-être pas), Android KickstartR pousse la personnalisation et la sélection de librairies au niveau au-dessus. Il est possible de choisir d'inclure ou non @ndroidAnnotations, Spring RestTemplate, ActionBar Sherlock, NineOldAndroids (Android 3.0 Animations API), Support LIbrary 4, ACRA (Library de CrashReport). D'autres librairies telles que ViewPAgerIndicator, RoboGuice 2 et ViewPager Sample sont annoncées, mais pour le moment pas encore disponibles.

Des présélections correspondantes au profil Basic, REST Application (Application qui a besoin de se connecter à un webService) et Full sont disponibles pour une configuration rapide.

Enfin la personnalisation automatique dans l'application résultante du nom de l'application, du nom de package, et de la main activity sont également disponibles et enfin la configuration du projet pour utiliser Maven et / ou Eclipse est présent.

Contrairement à AndroidBootstrap, l'application résultante sera un simple "Hello World", mais le projet est déjà configuré et permet de se lancer très rapidement dans le code sans passer par l'étape configuration.

Le code source de ce générateur d'applications est également open-source et disponible sur GitHub.

6 Annotations et injections

6.1 @ndroid @nnotations

Certains mécanismes de base sont toujours nécessaires et parfois répétitifs dans la création d'application Android. De plus, certains bouts de code alourdissent les classes, les rendant moins lisibles et donc plus difficiles à maintenir.

Cette librairie rajoute des annotations supplémentaires qui permettent de remplacer du code. Rien n'est magique ! Android Annotations se décompose en deux modules :

- Un pré-compilateur : Celui-ci va analyser le code lors de la création de l'APK. Les annotations seront alors converties en code.
- Une librairie qui fera partie de l'APK final.

Voici quelques exemples très parlants :

- Création d'une activité en FullScreen

```
public class MyActi extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(FLAG_FULLSCREEN, FLAG_FULLSCREEN);
    }
}
```

```
@NoTitle
@Fullscreen
public class MyActi extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
    }
}
```

- Action exécutée lors d'un click sur un bouton

```
View updateBookmarksButton1 = findViewById(R.id.updateBookmarksButton1);
updateBookmarksButton1.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        updateBookmarksClicked();
    }
});

View updateBookmarksButton2 = findViewById(R.id.updateBookmarksButton2);
updateBookmarksButton2.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
@Click({R.id.updateBookmarksButton1, R.id.updateBookmarksButton2})
void updateBookmarksClicked() {
    …
}
```

```
        updateBookmarksClicked();
    }
});

void updateBookmarksClicked() {
    …
}
```

- Création d'une tâche asynchrone

```
class UpdateBookmarksTask extends
AsyncTask<String, Void, Bookmarks> {

    @Override
    protected Bookmarks
doInBackground(String... params) {
        …
    }

    @Override
    protected void onPostExecute(Bookmarks
result) {
        …
    }
}
```

```
@Background
void searchAsync(String searchString, String
userId) {
    Bookmarks bookmarks =
restClient.getBookmarks(searchString,
userId);
    updateBookmarks(bookmarks);
}
```

De nombreuses fonctionnalités sont déjà supportées et de nouvelles sont rajoutées à chaque version. Nous pouvons déjà citer :

(En choisir quelques-unes des plus intéressantes en fonction de ce qui aura été vu au cours.)

La liste complète des annotations disponibles, ainsi que leur documentation respective est disponible sur https://github.com/excilys/androidannotations/wiki/AvailableAnnotations.

6.1.1 Comment cela fonctionne ?

C'est très simple : pour chaque classe annotée de l'application, le précomplateur va générer une classe dans le même package, nommé MaClasse_ et étendant MaClasse. Cette classe générée va rajouter des comportements à la classe de base en overridant certaines méthodes.

Certaines modifications dans le code sont donc nécessaires pour prendre en compte ce principe:

Dans le manifest : `<activity android:name=".MaClasse_" />`

Pour lancer une activité : `startActivity(this, MyListActivity_.class);`

Outre le temps que cette librairie fait gagner, elle évite des oublis ou des erreurs et empêche donc des bugs qui auraient pu être évités.

Qu'en est-il des performances ? La bonne nouvelle est qu'Android Annotation n'impacte pas du tout les performances ! Enfin, ce n'est pas tout à fait vrai : à la compilation, la précompilation faite par

Android Annotation demande un peu plus de temps, mais l'application APK produite elle reste du code Android classique, il n'y a aucune réflexion ou chargement supplémentaire nécessaire à l'exécution.

6.2 RoboGuice

RoboGuice est une autre librairie similaire. Celle-ci est basée sur Google Guice qui est une bibliothèque d'injection de dépendance.

La différence majeure qu'il y a vis-à-vis d'Android Annotation est que cette injection est effectuée au running time (quand l'application est exécutée). Il y a donc un impact sur la vitesse d'exécution de l'application.

6.2.1 Mise en place

Pour utiliser l'injection de dépendance proposée par RoboGuice, il suffit de configurer le projet Eclipse en rajoutant les .jar de RoboGuice et de Guice 3.0 (No AOP version). Pour une plus grande simplicité, il est évidemment possible d'utiliser Maven pour gérer ces dépendances.

6.2.2 Injection !

Sans RoboGuice

```
class AndroidWay extends Activity {
    TextView name;
    ImageView thumbnail;
    LocationManager loc;
    Drawable icon;
    String myName;

    public void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        name      = (TextView)
findViewById(R.id.name);
        thumbnail = (ImageView)
findViewById(R.id.thumbnail);
        loc       = (LocationManager)
getSystemService(Activity.LOCATION_SERVICE);
        icon      =
getResources().getDrawable(R.drawable.icon);
        myName    =
getString(R.string.app_name);
        name.setText( "Hello, " + myName );
    }
}
```

Avec RoboGuice

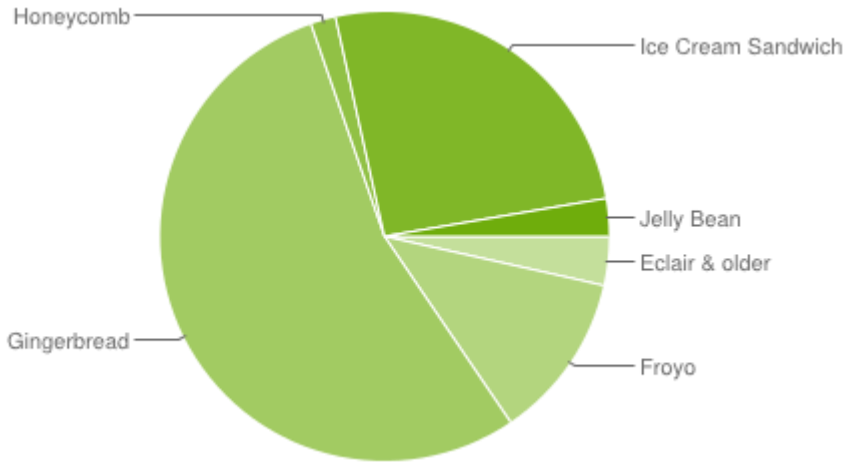
```
class RoboWay extends RoboActivity {
    @InjectView(R.id.name)
TextView name;
    @InjectView(R.id.thumbnail)
ImageView thumbnail;
    @InjectResource(R.drawable.icon)
Drawable icon;
    @InjectResource(R.string.app_name)
String myName;
    @Inject
LocationManager loc;

    public void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        name.setText( "Hello, " + myName );
    }
}
```

7 Autres librairies

7.1 Support Library

La Support Library (ou Compatibility Library jusqu'à sa version 3) est une bibliothèque additionnelle, développée par Google qui permet l'utilisation d'API qui n'étaient pas encore disponibles dans les versions anciennes d'Android ou qui rajoutent des fonctionnalités supplémentaires qui ne font pas partie du Framework de base. Le but est réellement de simplifier le développement d'application, surtout au niveau de la gestion des différentes versions d'Android.

Il s'agit en quelque sorte d'une réponse à l'éclatement des versions d'Android.



7.1.1 Versions

Il en existe 2 versions différentes : v4 et v13, requérant respectivement l'API Android de niveau 4 et de niveau 13.

La version 4 permet l'accès à des fonctionnalités tirées d'Android 3.0 et supérieures :

- **Fragment**
- **FragmentManager**
- **FragmentTransaction**
- **ListFragment**
- **DialogFragment**
- **LoaderManager**
- **Loader**
- **AsyncTaskLoader**
- **CursorLoader**
- et d'autres ...

L'utilisation de ces classes de compatibilité est la même (à l'une ou l'autre exception près) que celle des versions 3 et supérieures.

L'ActionBar n'est pas disponible dans cette librairie, mais nous avons vu (verrons) comment la remplacer.

7.1.2 Inclusion dans un projet

Si vous utilisez Eclipse, son installation est très simple : Clic doit sur le projet)> Build Path)> Add to Build Path. Si non, le package est téléchargeable via le SDK Manager (il est alors téléchargé dans `<sdk>/extras/android/support/v4/android-support-v4.jar` ou `<sdk>/extras/android/support/v13/android-support-v13.jar`). Il suffira alors de l'ajouter en tant que librairie au projet Android.

7.2 ActionBarSherlock

ActionBarSherlock est une extension de la bibliothèque de support conçu pour faciliter l'utilisation de l'ActionBar dans toutes les versions d'Android avec une seule API.

La bibliothèque utilise automatiquement la barre d'action native ou, le cas échéant enveloppe automatiquement une implémentation personnalisée autour des layout. Cela permet de facilement développer une application avec une barre d'action pour chaque version à partir d'Android 2.

Plus d'infos : http://actionbarsherlock.com/

7.3 GreenDroid(Light)

GreenDroid est une bibliothèque de développement pour la plate-forme Android développée par Cyril Mottier2. Elle a pour but d'aider dans la conception d'interfaces utilisateur (en simplifiant la mise en place de certains widgets notamment) en gardant une certaine cohérence en termes de charte graphique.

Les principaux objectifs de cette librairie sont nombreux :

- Éviter de perdre du temps à copier les mêmes extraits de code encore et encore ;
- Tenter de coder des applications Android au design cohérent ;
- Aider les développeurs à coder des applications hautement fonctionnelles ;
- Exploiter la puissance du framework Android ;
- Utiliser autant que possible du XML.

7.3.1 GDCatalog

La bibliothèque comprend un projet appelé GDCatalog qui utilise la bibliothèque GreenDroid. Il a été développé pour aider les développeurs à comprendre comment utiliser GreenDroid dans leurs applications Android.

Plus d'infos : http://greendroid.cyrilmottier.com/

7.3.2 GreenDroidLight

Cette version offre une version allégée de la bibliothèque originale :

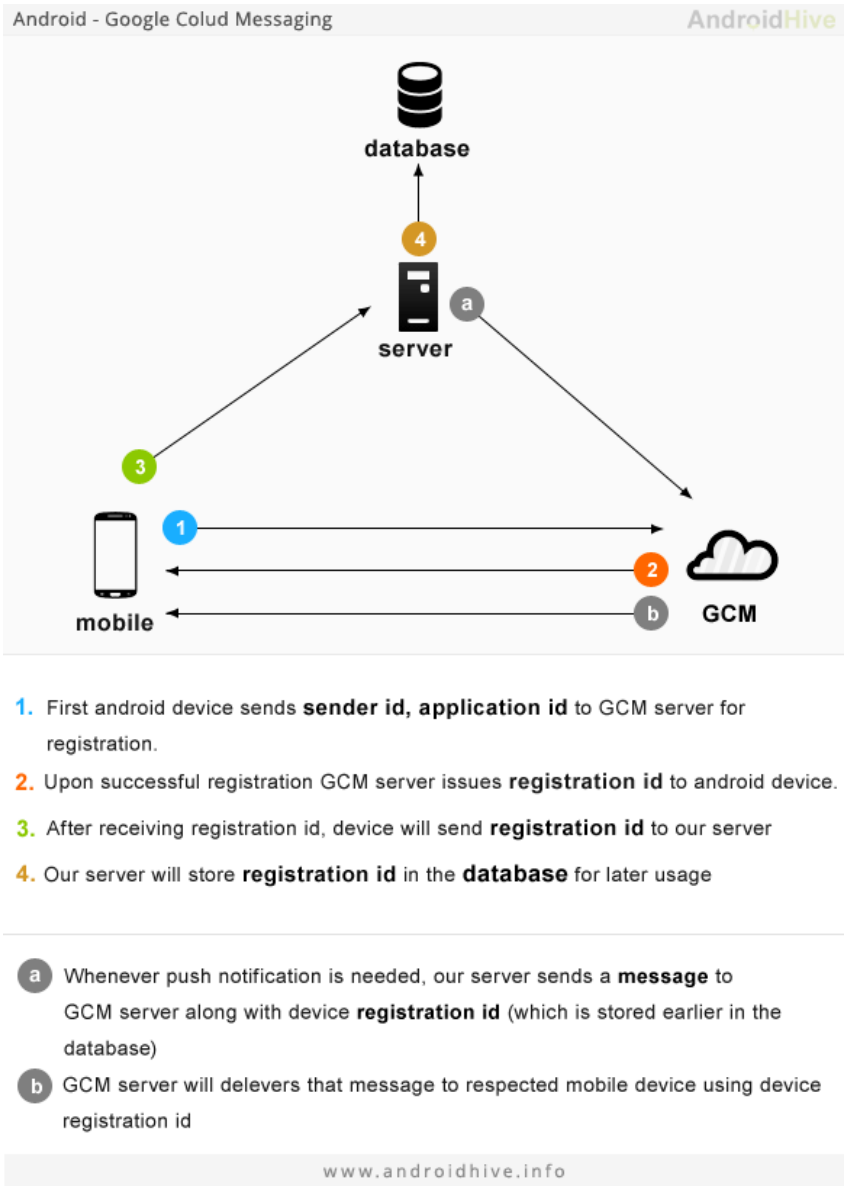
- Tout le code correspondant à ActionBar a été retiré comme il entre en conflit avec l'implémentation du SDK à partir de HoneyComb. Pour le support de l'ActionBar, il suffit de s'intéresser à ActionBarSherlock.
- Certaines contraintes de la bibliothèque d'origine ont été modifiées.

Plus d'infos : https://github.com/foxykeep/GreenDroidLight

7.4 Google Cloud Messaging

Google Cloud Messaging pour Android (GCM) est un service qui permet aux développeurs d'envoyer des données à partir de serveurs aux applications Android sur les appareils. Le service gère tous les aspects de file d'attente de messages et leur diffusion à l'application cible Android fonctionnant sur le périphérique visé.

En utilisant ce service, on peut envoyer des données à une application chaque fois que de nouvelles données sont disponibles au lieu de faire des requêtes au serveur en boucle. Cela améliore l'expérience utilisateur et permet d'économiser la batterie.



Plus d'infos :

- http://developer.android.com/guide/google/gcm/index.html
- http://www.androidhive.info/2012/10/android-push-notifications-using-google-cloud-messaging-gcm-php-and-mysql/

7.5 Spring for Android

Spring pour Android est une extension du Framework Spring (sur lequel il est basé) qui permet de simplifier le développement d'applications Android natives.

Il apporte de base les fonctionnalités d'un **client REST** et **un système d'authentification (OAuth)** qui permet l'accès sécurisé à des webservices.

La librairie en est à sa version 1.0 et cette première version stable date du 30/05/2012.

L'avantage de Spring for Android est que la plupart des composants sont repris de Spring. Outre le fait qu'il s'agit d'un gage de qualité, un utilisateur habitué à Spring s'y retrouvera donc très rapidement.

Ces fonctionnalités sont séparées en .jar différents et il est donc possible de profiter de certaines fonctionnalités séparément.

7.5.1 Rest Client

Il est basé sur le client REST de Spring fait en Java et se base sur une classe RestTemplate qui offre une couche d'abstraction pour les requêtes HTTP (elle est basée sur le client HTTP Android de base).

Il permet également de marshaller les objets pour les transmettre en HTTP et d'unmarshaller les réponses en objet pour les exploiter au sein d'Android. Pour prendre en charge cette (dé)compression vers du JSON, Spring pour Android se base sur des applications tierces. Elle en supporte 2 actuellement : Jackson (La plus connue) et Google Gson (La plus légère). Il est également possible de convertir vers du XML et là se base sur Simple XML Serialization.

Enfin, il propose un support pour l'agrégation de Flux RSS ou Atom, basé à nouveau sur une librairie tierce Android Rome Feed Reader.

Toutes les librairies citées précédemment sont open-source.

Il supporte également la compression des données en GZip, ce qui est plutôt important pour un téléphone souvent connecté au data.

7.5.2 Un peu de code ?

```
// Recherche sur Google
String url = "https://ajax.googleapis.com/ajax/services/search/web?v=1.0&q={query}"
// Create a new RestTemplate instance
RestTemplate restTemplate = new RestTemplate();

// Add the String message converter
restTemplate.getMessageConverters().add(new StringHttpMessageConverter());

// Make the HTTP GET request, marshaling the response to a String
String result = restTemplate.getForObject(url, String.class, "Ma recherche");
```

D'autres exemples :

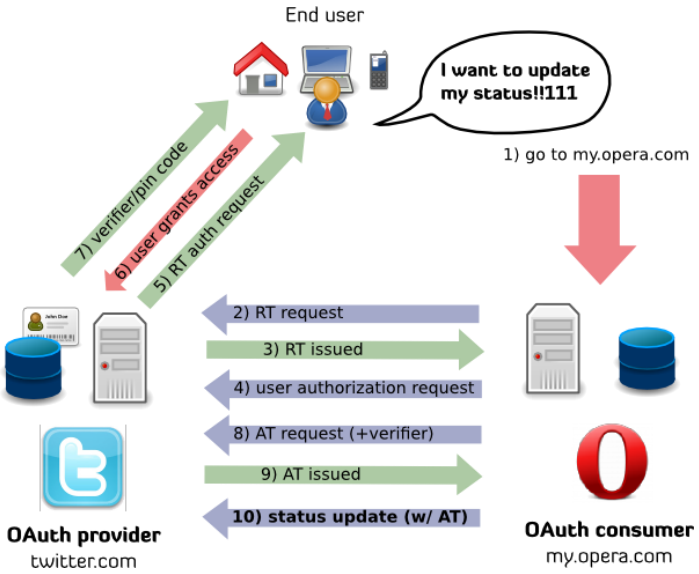
http://static.springsource.org/spring-android/docs/1.0.x/reference/html/rest-template.html

7.5.3 Système d'authentification

De plus en plus d'applications requièrent un compte afin d'accéder à des fonctionnalités en étant identifiées. Pour mettre cela en place, l'application doit se connecter aux webservices afin de vérifier la connexion est les droits à allouer à l'utilisateur. Un des standards qui s'est rapidement imposé est OAuth.

Les applications peuvent également tirer parti des réseaux sociaux pour certaines actions, il est donc également nécessaire que les utilisateurs soient connectés sur ceux-ci.

Rappel du fonctionnement de base de OAuth



L'encryption des données envoyées pour la connexion est évidemment disponible via le composant Spring Security et adaptée pour Android.

7.5.4 Intégration de Spring Social

Une des forces du framework Spring est la possibilité de rajouter des fonctionnalités en rajoutant des plugin au système de base. L'un des plus adulés pour le moment est le composant Spring Social qui permet de rajouter rapidement une connexion à Facebook et à Twitter. Il permet également la possibilité aux utilisateurs de l'application de s'authentifier localement via un webservice distant.

Un peu de code ?

```
FacebookConnectionFactory connectionFactory;
connectionFactory = (FacebookConnectionFactory)
connectionFactoryRegistry.getConnectionFactory(Facebook.class);
String redirectUri = "https://www.facebook.com/connect/login_success.html";
String scope = "publish_stream,offline_access,read_stream,user_about_me"; // Permissions
MultiValueMap<String, String> additionalParameters = new LinkedMultiValueMap<String,
String>();
additionalParameters.add("display", "touch"); //Configurer pour affichage un mobile
OAuth2Parameters parameters = new OAuth2Parameters(redirectUri, scope, null,
additionalParameters);
OAuth2Operations oauth = connectionFactory.getOAuthOperations();
String authorizeUrl = oauth.buildAuthorizeUrl(GrantType.IMPLICIT_GRANT, parameters);
AccessGrant accessGrant = new AccessGrant(accessToken);
Connection<FacebookApi> connection = connectionFactory.createConnection(accessGrant);
connectionRepository.addConnection(connection);
```

7.5.5 Applications d'exemples

Plusieurs applications de tests et de démo sont disponibles sur GitHub et reprennent les fonctionnalités décrites précédemment :

- Connexion et recherche à Twitter ;
- Connexion à Facebook ;
- Connexion à un webservice custom ;
- Agrégateur de flux RSS.

7.6 Polaris

7.6.1 Leitmotiv

Cela fait presque trois ans que Google a mis à jour la librairie Google Maps pour Android. Comme vous le savez tous, cette bibliothèque externe peut être utilisé pour intégrer les fonctionnalités liées à la cartographie dans les applications. Malheureusement, en plus de vieillir, cette librairie souffre également d'un grand nombre de limitations.

La philosophie de Polaris est basée sur 3 règles principales :

- Rendre la vie de l'utilisateur plus facile ;
- Rendre la vie du développeur plus facile ;
- Faire que la carte semble "polie" et naturelle.

Plus d'infos :

- https://speakerdeck.com/cyrilmottier/polaris-simple-mapping-library-for-android
- http://android.cyrilmottier.com/?p=824

8 Outils et testing de développement

8.1 Maven for Android

Plus d'infos :

- http://code.google.com/p/maven-android-plugin/
- http://rgladwell.github.com/m2e-android/

8.2 Robotium

Après avoir conçu une application, il est important de la tester. Je ne vous apprendrai pas qu'une application bugguée n'est pas aimée, et se construit vite une mauvaise réputation (encore plus avec le système de scores et de top du Google Play). C'est dans ce contexte qu'entre Robotium. Comme le dit leur slogan, Robotium c'est comme Selenium, mais pour Android.

Robotium peut être utilisé pour tester tant des applications dont on a le code source, comme des applications dont on ne possède que l'APK.

Le projet Robotium a débuté en 2010 (soit aux alentours de la version 2.1 d'Android). Ils assurent un support d'Android 1.6 et supérieur. Robotium peut tester une application tant sur l'émulateur que sur un device réel. Il permet également de prendre des screenshots pour capturer l'état de l'application quand on le souhaite (mais requiert évidemment une permission supplémentaire dans le manifest *android.permission.WRITE_EXTERNAL_STORAGE*, cela peut poser problème pour des applications dont on a que l'APK). Le code source du projet est disponible sur GitHub : https://github.com/jayway/robotium (et non sur Google Code, où se trouve la page principale du projet, car il ne supporte pas Git).

8.2.1 Utilisation / Intégration dans un projet

Afin de tester son application, avec Robotium (et même sans), il faut créer un projet de test. Robotium doit être intégré en tant que librairie au sein du projet **de test**.

Pour lancer la suite de Tests, il suffira de faire comme une suite de tests normale : Clic droit → Run As JUnit Test.

8.2.2 Classe de tests

```
public class EditorTest extends ActivityInstrumentationTestCase2<EditorActivity> {
  private Solo solo;
  public EditorTest() {
                super("com.test.editor", EditorActivity.class); //Classe à tester
  }
  public void setUp() throws Exception {
        solo = new Solo(getInstrumentation(), getActivity());
  }
  public void testPreferenceIsSaved() throws Exception {
                solo.sendKey(Solo.MENU); //Simule un clic sur la touche menu.
                solo.clickOnText("More"); // Clique sur “More”.
           //Roboto permet l’utilisation de strings localisée :
solo.getString(“id.resources”)
                solo.clickOnText("Preferences");
                solo.clickOnText("Edit File Extensions");
                Assert.assertTrue(solo.searchText("rtf")); //Chercher le texte

                solo.clickOnText("txt"); // Cliquer sur le texte
                solo.clearEditText(2); // Vider la 2e textBox
                solo.enterText(2, "robotium"); // Entrer le texte “robotium” dans la 2e
textBox
                solo.clickOnButton("Save"); //Clicker sur sauver
                solo.goBack(); // Emule le bouton back.
                solo.clickOnText("Edit File Extensions");
                Assert.assertTrue(solo.searchText("application/robotium")); //Vérifie que
la modification ait bien été faite.
  }
   @Override
   public void tearDown() throws Exception {
        solo.finishOpenedActivities(); //Kill les activités ouvertes.
  }
}
```

8.3 UI Automator Test Framework

Depuis sa version 21 (c’est-à-dire celle sortie il y tout juste une dizaine de jours), le SDK apporte un nouvel outil de test : UI Automator Test Framework. Celui-ci s’articule autour de 3 nouveaux outils :

Un outil graphique pour scanner et ananlyser les différents composants de l’interface (UIAUtomatorViewer) ;

Une librairie qui contient des API pour créer des tests fonctionnels d’UI (UIAutomator) ;

Et enfin un moteur d’exécution pour ces tests sur plusieurs plateformes.

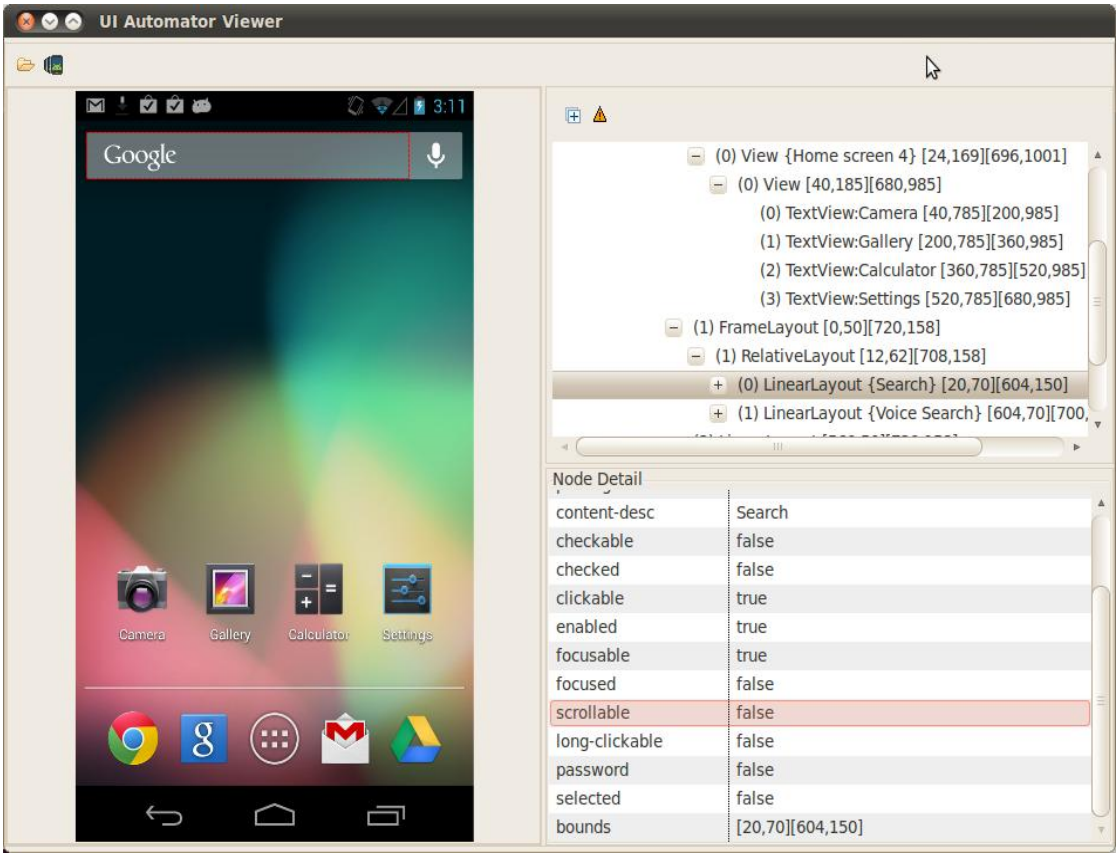
Cet outil requiert Android 4.1 (API 16) ou supérieur. Ce système requiert également l'utilisation d'un appareil physique, il ne fonctionne pas directement sur l'émulateur.

8.3.1 Fonctionnement

Le fonctionnement est assez similaire à Robotium. Il n'est pas nécessaire de connaître le code source afin de tester l'application. Cela permet de bien séparer les tests du code de l'application (Ou permet de travailler en équipe : développeur =/= testeur).

L'UIAutomatorViewer permet de visualiser ce qu'il se passe à l'écran de l'appareil de test et de prendre des screenshots.

Il permet également d'observer la structure de l'écran qui est présenté sur le device de test :



C'est à partir de ces informations qu'il est alors possible de rédiger les tests qui seront exécutés par UIAutomator : on peut facilement pointer un composant de l'interface.

8.3.2 Création des tests

La création des tests via Eclipse est facilitée grâce à l'intégration de la librairie. Pour ce faire, il suffit de créer un nouveau projet, puis d'éditer de JavaBuildPath pour rajouter les librairies JUnit 3 et les .jar uiautomator.jar et android.jar contenu dans le répertoire du <SDK>/platforms/.

Afin de créer une classe de test, il faut créer une classe qui étend UIAutomatorTestCase. Cette classe étend elle-même la classe junit.framework.TestCase, ce qui permet d'utiliser le fameux Assert() de JUnit.

Afin de lancer les tests, l'application testée doit bien sûr être installée sur l'appareil en question.

8.3.3 Un peu de code ?

```
public class LaunchSettings extends UiAutomatorTestCase {

   public void testDemo() throws UiObjectNotFoundException {

      // getUIDevice() : récupère le device de test.
      // pressHome() : Simule une pression simple sur le bouton home.
      getUiDevice().pressHome();

      // We're now in the home screen. Next, we want to simulate
      // a user bringing up the All Apps screen.
      // If you use the uiautomatorviewer tool to capture a snapshot
      // of the Home screen, notice that the All Apps button's
      // content-description property has the value "Apps".  We can
      // use this property to create a UiSelector to find the button.
      UiObject allAppsButton = new UiObject(new UiSelector()
         .description("Apps"));

      // Simulate a click to bring up the All Apps screen.
      allAppsButton.clickAndWaitForNewWindow();

      // In the All Apps screen, the Settings app is located in
      // the Apps tab. To simulate the user bringing up the Apps tab,
      // we create a UiSelector to find a tab with the text
      // label "Apps".
      UiObject appsTab = new UiObject(new UiSelector()
         .text("Apps"));

      // Simulate a click to enter the Apps tab.
      appsTab.click();

      // Next, in the apps tabs, we can simulate a user swiping until
      // they come to the Settings app icon.  Since the container view
      // is scrollable, we can use a UiScrollable object.
      UiScrollable appViews = new UiScrollable(new UiSelector()
         .scrollable(true));

      // Set the swiping mode to horizontal (the default is vertical)
      appViews.setAsHorizontalList();

      // Create a UiSelector to find the Settings app and simulate
      // a user click to launch the app.
      UiObject settingsApp = appViews.getChildByText(new UiSelector()
         .className(android.widget.TextView.class.getName()),
         "Settings");
      settingsApp.clickAndWaitForNewWindow();

      // Validate that the package name is the expected one
      UiObject settingsValidation = new UiObject(new UiSelector()
```

```
            .packageName("com.android.settings"));
        assertTrue("Unable to detect Settings",
            settingsValidation.exists());
    }
}
```

8.3.4 Exécution des tests

Contrairement à Roboto UI, le fichier de tests sera ici un .jar généré par la commande :

```
<android-sdk>/tools/android uitest-project -n <name> -t 1 -p <path>
```

- Envoi sur l'appareil :

```
adb push <path_to_output_jar> /data/local/tmp/
```

- Exécution :

```
adb push <path_to_output_jar> /data/local/tmp/
```

Plus d'infos *:* http://developer.android.com/tools/testing/testing_ui.html

9 Ressources pertinentes

- FrAndroid DevTips
- Cyril Mottier's blog
- Videos Google I/O
- Romain Nurik
- http://android-developers.blogspot.fr/
- Particulièrement http://developer.android.com/training/index.html
- http://www.theultimateandroidlibrary.com/
- http://www.androidhive.info/