J.ALAOUI

Programmation sous Android

Projet GSB-FRAIS

ECLIPSE

ANDROID SDK

Ressource:

http://developer.android.com/reference

Système d'exploitation Android

Un système d'exploitation pour les appareils mobiles, tablettes et systèmes embarqués avec une interface de programmation Java.

Android est créé par l' Open Handset (qui regroupe plusieurs entreprises) Alliance dirigée par Google.

Open Handset :

Android un OS open source utilisant le noyau Linux.

La dernière version de la plateforme est maintenant *Android 4.4.2* alias *Kit Kat*.

Le développement d'application s'appuie sur le langage Java.

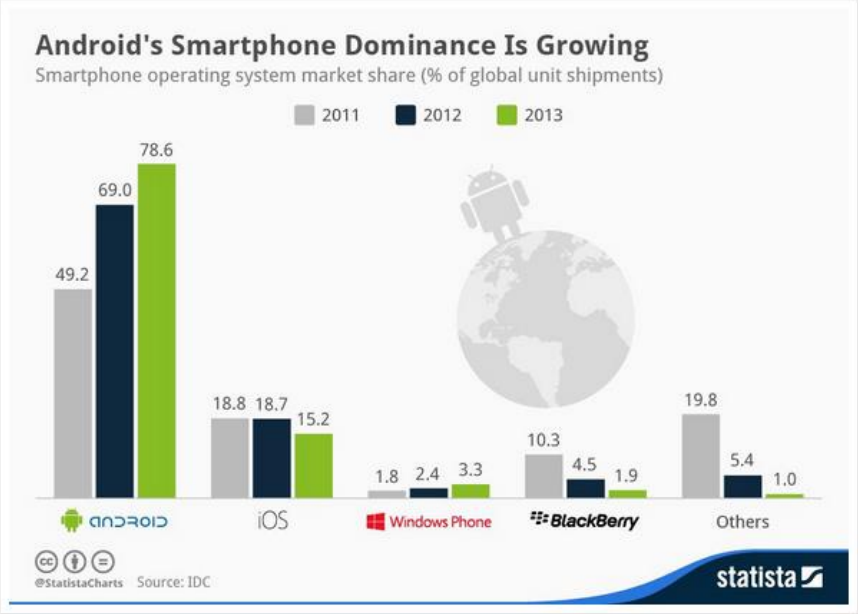
Moteur d'exécution : Machine virtuelle Dalvik

Le kit de développement (SDK) d'Android est un ensemble complet d'outils de développement :

- Compilateur
- Débogueur
- Emulateur
- Machine virtuelle

Android est installé sur un nombre croissant d'appareils mobiles

Android : une part de marché proche de 80%



Installation

SDK Java

Le développement Android nécessite le SDK Java en version 6 ou supérieure, le JRE seul n'est pas suffsant. JRE (Java Runtime Environement) est le kit destiné au client pour pouvoir exécuter un programme Java. Il se compose essentiellement d'une machine virtuelle Java (JVM) capable d'exécuter le bytecode et la bibliothèque standard de Java. SDK (Standard Development Kit), est composé d'un JRE, d'un compilateur, de nombreux programmes utiles, d'exemples de programmes Java et les sources de toutes les classes de l'API.	

Connaitre la version de Java installée sur le poste
Taper sous cmd la commande suivante :

```
C:\>java -version
```

Installer le kit de développement Android : Android SDK

Télécharger le kit SDK ADT Bundle à partir de l'adresse suivante : http://developer.android.com/sdk/index.html Télécharger le SDK ADT Bundle pour Windows	

ADT plugin pour Eclipse

ADT (**Android Development Tools**) est un module additionnel pour Eclipse qui étend les capacités de celui-ci afin de faciliter le développement Android (création, tests et débogage des applications,...). Il permet notamment d'intégrer dans l'EDI la majorité des outils du SDK Android :

• l'Android Project Wizard, assistant de création de projets,
• des éditeurs de manifestes, d'agencements et de ressources,
• la conversion en exécutables Android .dex, la création de fichiers de paquetage .apk et l'installation de ces fichiers dans des machines virtuelles Dalvik,

• le gestionnaire AVD (Android Virtual Device) qui permet de gérer les dispositifs d'hébergement virtuels,
• l'Android Emulator avec contrôle des connexions réseau et possibilité de simuler des appels entrants,
• le DDMS (Dalvik Debug Monitoring Service) comprenant la redirection de ports, la visualisation de la pile, du tas et des threads, le détail des processus et la possibilité de réaliser des captures d'écran,
• … …

Installation

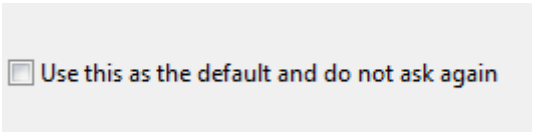
Créer un dossier c:/Android

Décompresser l'archive adt-bundle-windows-x86-20131030 dans un répertoire approprié le dossier Android et renommer le dossier décompressé : android-sdk

Epingler à la barre de tâche l'application Eclipse

SDK Manager

Le SDK Android est constitué de paquets modulaires qui peuvent être téléchargés séparément au moyen de l'outil SDK Manager. Cet outil est particulièrement pratique lorsqu'il est nécessaire de faire une mise à jour liée à une évolution du niveau d'API. Plusieurs niveaux d'API peuvent cohabiter.
Lancer Eclipse, Au démarrage, Eclipse demande de choisir un répertoire de travail (workspace) où seront localisés les projets de développement.
Sélectionner le sous-dossier workspace du dossier Android



Cocher

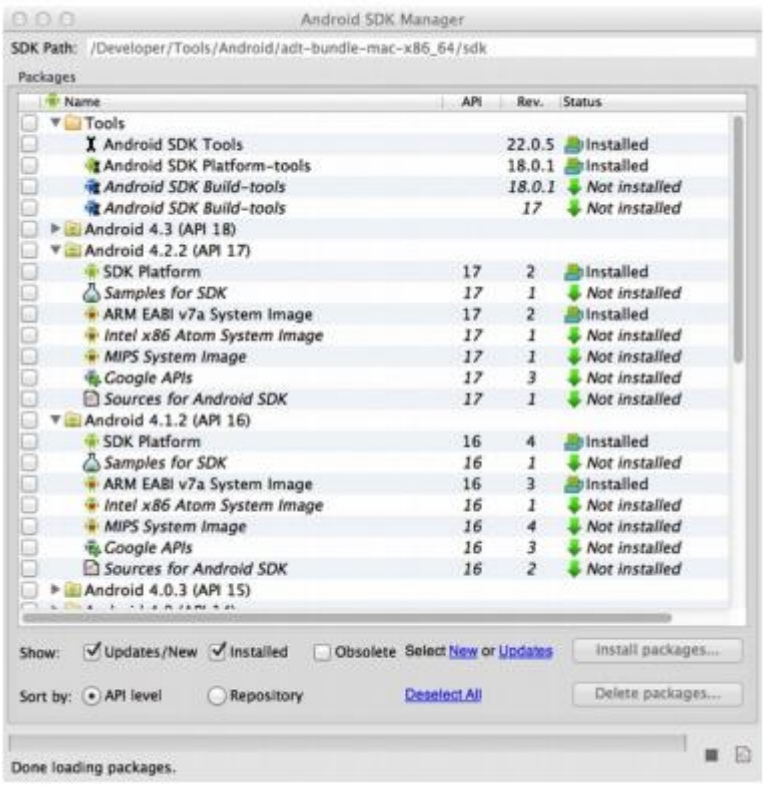
Pour lancer SDK Manager
Cliquer sur l'icône

Ou
Le SDK Manager est accessible depuis le menu principal d'Eclipse :
Window | Android SDK Manager.
Ou
Les paquets disponibles sont tous localisés dans le sous-répertoire :
C:\android\android…

L'installation de la documentation offine et des samples(exemples)

(proposée normalement pour le plus haut

niveau d'API disponible) peut être une bonne chose...

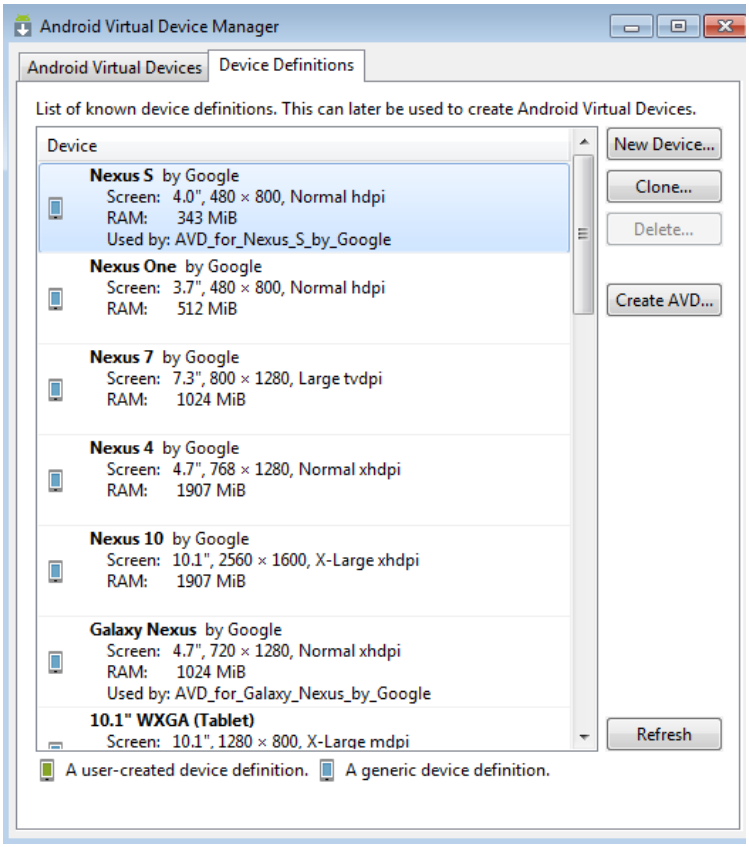


Création d'un AVD

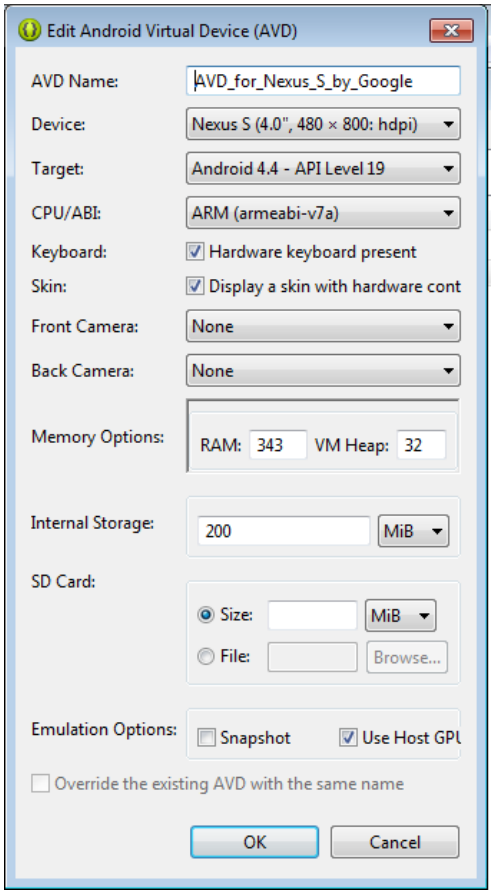
Les applications peuvent (doivent ?) être testées sur un émulateur Android avant d'être réellement installées sur un smartphone ou tablette.
Il est donc nécessaire de créer au moins un AVD (**Android Virtual Device**) afin de définir les caractéristiques d'un émulateur :
• dans le menu principal d'Eclipse,
• sélectionner Window | Android Virtual Device Manager,
• choisir l'onglet Devices Definition. On obtient une liste des Devices prés à l'emploi,

SDK : ***software development kit*** = outils et fichiers pour programmer
AVD : ***Android Virtual Device*** =Smartphone et tablettes virtuelles
ADB : ***Android Debug Bridge. C'est un outil qui permet l'accès à la console du téléphone***
= logiciel de communication
ADT : ***Android Development Tools*** = plugin pour Eclipse

• Choisir Nexus S by Google et cliquer sur **Create AVD**



• renseigner les différents champs de l'appareil à émuler
• cliquer sur Ok



Lancer AVD Manager 
Sélectionner AVD crèer puis cliquer sur Start puis **Lanche**

Les versions

Les APIs

une **interface de programmation** (souvent désignée par le terme **API** pour *Application Programming Interface*) est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

- 2.1 : Eclair (Eclair - Janvier 2010), API level 7
- 2.2 : FroYo (Frozen Yogourt / Yaourt glacé - Mai 2010), API level 8
- 2.3 : Gingerbread (Pain d'épice - Décembre 2010), API level 9
- 3.0 : Honeycomb (Gâteau de Miel – Février 2011), API level 11
- 3.1 : Mars 2011, API level 12
- 3.2 : Juillet 2011, API level 13
- 4.0 : Ice Cream Sandwich (Sandwich à la crème glacée - Octobre 2011), API level 14
- 4.0.3 : Décembre 2011, API level 15
- 4.1 : Jelly Bean (Bonbon à la gelée / Dragée – Juillet 2012), API level 16
- 4.2 : Octobre 2012, API level 17
- 4.3 : Juillet 2013, API level 18
- 4 .4 : Kitkat, fn 2013 API level 19

Accès à la documentation

Comment connaître les attributs et méthodes disponibles pour tel ou tel objet ?
Tout simplement en consultant la documentation du SDK sur le site
http://developer.android.com/reference

Le panneau de gauche donne une liste des points d'entrée de la doc. (espaces de noms) ; la page sélectionnée donne la liste des classes de l'espace, et chaque classe est ensuite détaillée par ses attributs et ses méthodes, héritées ou non. L'outil de recherche situé en haut à droite est bien utile... lorsque l'on connait le nom de ce que l'on cherche.

Eclipse propose également une aide contextuelle très aboutie : dans la vue Éditeur, pour un source java, il suffit de laisser le pointeur de la souris sur un mot (nom de classe, de méthode, …) pour voir surgir une fenêtre contenant la page d'aide appropriée ; glisser ensuite le pointeur souris dans cette fenêtre pour pouvoir naviguer dans la page de documentation... cliquer sur un lien et la documentation apparaît dans un nouvel onglet de l'éditeur.

La documentation Android est aussi accessible hors ligne, voir rubrique SDK Manager...

Application saisie des frais des visiteurs GSB

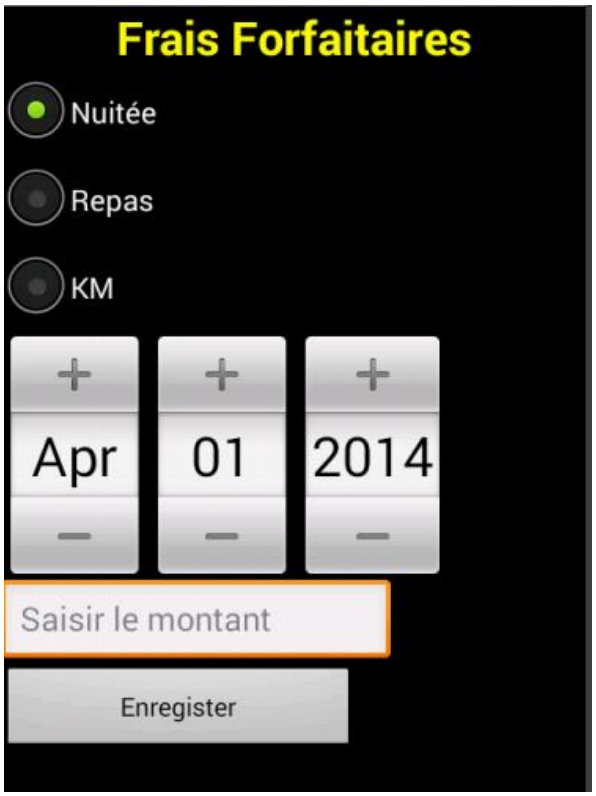
Voir cahier de charge PPE 1

Mission 1 : Application de saisie des frais au forfait des visiteurs

Cas d'utilisation

Vue 1 : Menu principal

Vue 2 : Gestion des frais au forfait



Mission 2 : Application de saisie des frais hors forfait des visiteurs

Les données saisies dans le cadre de cette application sont stockées dans le SGBD SQLite.

Mission 3 : GSB – Exportation des données sqLite vers la BDD Mysql

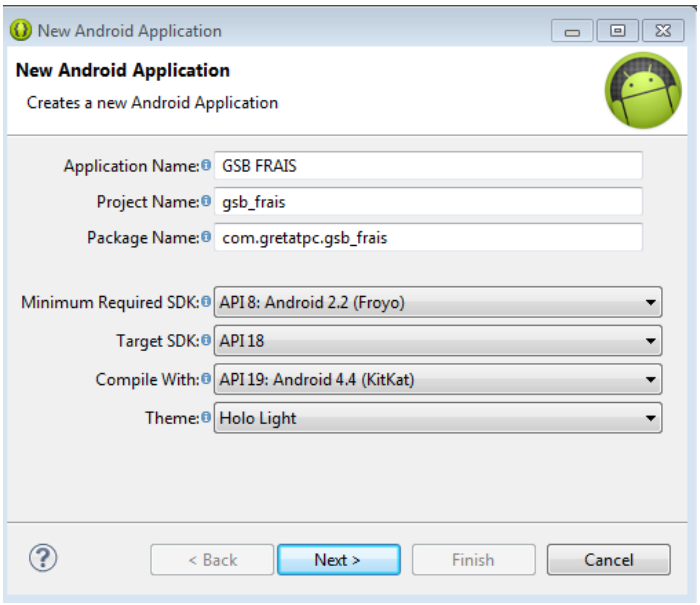
Exportation des données saisies vers la base de données Mysql distante pour validation des frais par le comptable (à chaque fin de mois)

Création du projet : GSB FRAIS

Lancer Eclipse

Démarrer Eclipse et lancer File | New | Android Application Project...
Remplir les champs de la boite de dialogue :

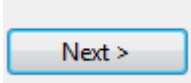
Application name : ***GSB FRAIS***
nom « convivial » de l'application, celui qui apparaitra sur le smartphone... :
• Project name : **gsb_frais**
nom du projet Eclipse (répertoire dans le dossier workspace défni lors du démarrage de l'EDI)
• Package name : com.gretatpc.gsb_frais
nom de package Java qui doit être unique pour la publication sur Google Play.
On peut utiliser le namespace par défaut pour les exemples de la documentation (com.example), mais cet espace ne permettra pas une publication sur Google Play

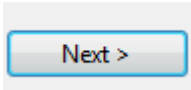


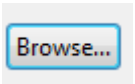
NB : Eclipse est un IDE, Integrated Development Environment (EDI environnement de développement intégré en français), c'est-à-dire un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation. Il est développé par IBM, est gratuit et disponible pour la plupart des systèmes d'exploitation.

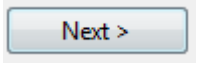
• Minimum required SDK : version minimale de SDK (API level) qui pourra être utilisée par l'application
• Target SDK : version la plus récente du SDK sur laquelle l'application a été testée
• Compile with : version de la librairie avec laquelle compiler l'application, choisir la version la plus élevée disponible
• Theme : style de l'interface utilisateur
(note : pour ces 4 derniers champs, les valeurs proposées par défaut doivent convenir...)

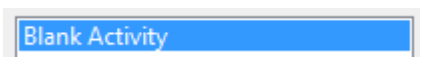
Ensuite

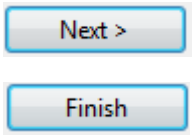




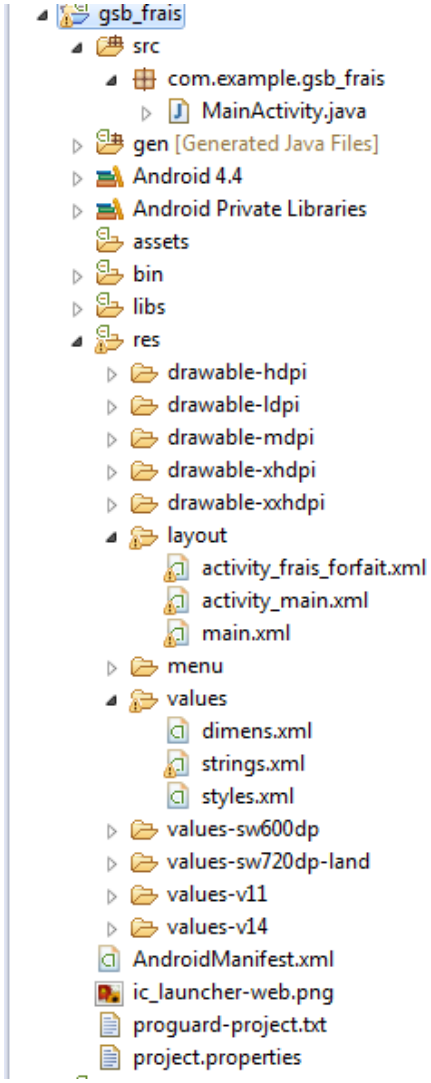
Choisir votre Image File avec 



Choisir 



Structures du Project Androïde



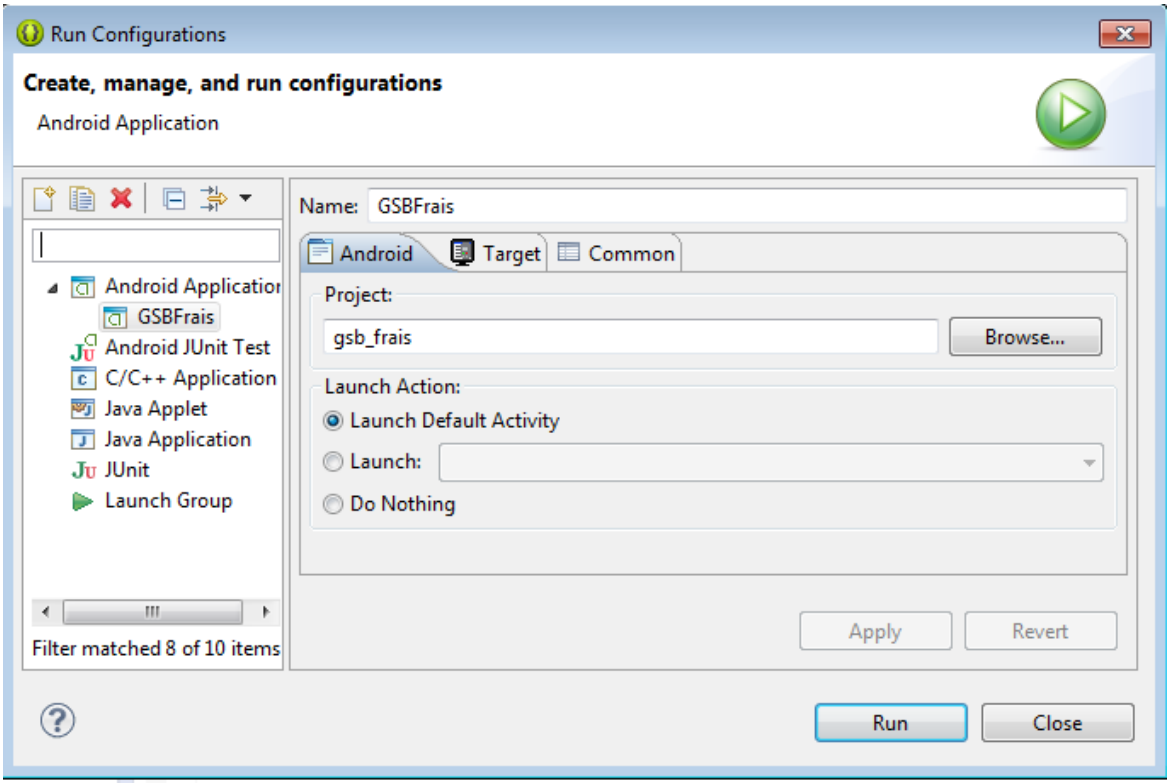
- src : Ce dossier contient les sources de votre application (code JAVA) et les packages. Le paquetage de notre application est com.gretatpc.gsb_frais, bien sûr vous pouvez avoir plusieurs.
- gen : Dossier qui contiendra le fichier R.java (le code source produit par les outils de compilation d'Android).
- assets : Contient les fichiers statiques fournis avec l'application pour son déploiement sur le terminal.
- res : C'est le dossier qui contiendra les ressources de votre application (images, vidéos, styles), c'est-à-dire les fichiers statiques fournis avec l'application, soit sous leur forme initiale, parfois sous une forme prétraitée.
  - drawable-hpdi : Contient toutes images, bitmaps dont vous avez besoin pour votre application en haute résolution.
  - drawable-ldpi : Contient toutes images, bitmaps dont vous avez besoin pour votre application en basse résolution.
  - drawable-mdpi : Contient toutes images, bitmaps dont vous avez besoin pour votre application en moyenne résolution.
  - layout : Le SDK Android offre une technique de création d'interfaces graphiques à l'aide de fichiers XML .C'est dans ce dossier que vous inclurez l'ensemble des fichiers décrivant vos interfaces.
  - values : Ce dossier contient un ensemble de fichiers décrivant les valeurs utilisées par votre application .On peut, par exemple, y mettre des chaines de caractères, des entiers, des couleurs (strings.xml), des tableaux (arrays.xml),

etc… .

- AndroidManifest.xml :C'est le point de départ de toute application Android. C'est dans ce fichier que l'on déclare ce que contiendra l'application- les activités, les services, …).

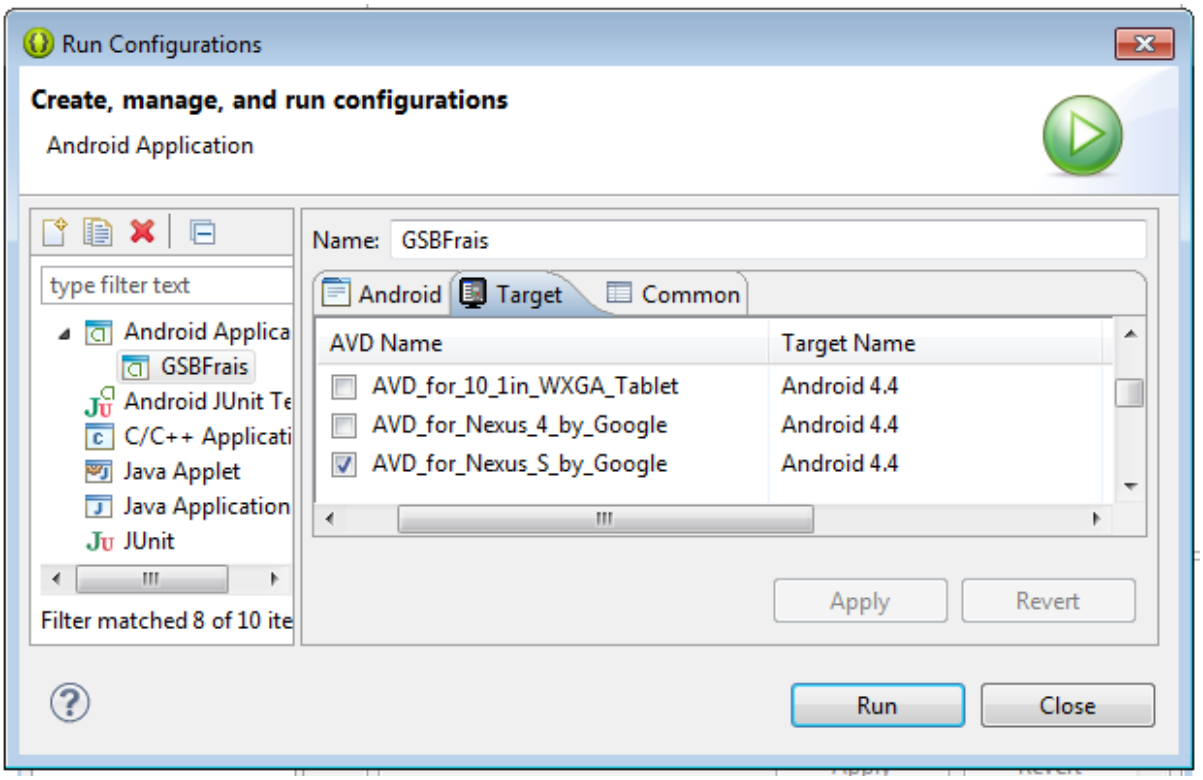
Tester l'application de base

Choisir dans le menu principal Run -> Run configuration …

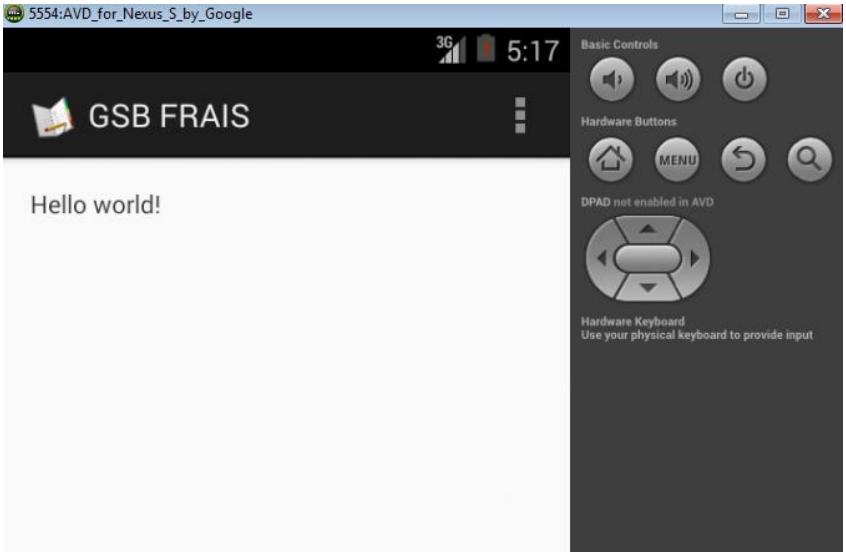


En utilisant le bouton **Browse**.. choisir l'application gsb_frais

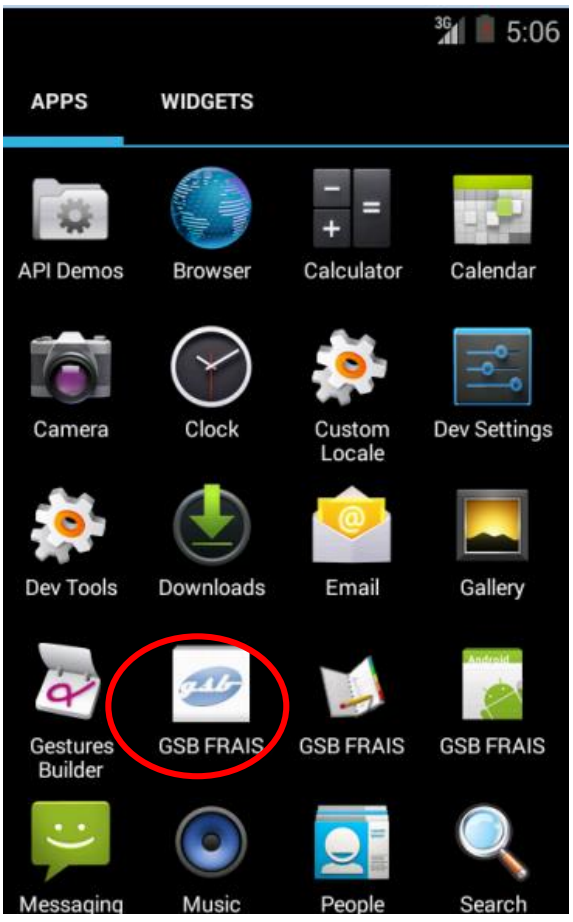
Choisir l'onglet Target pour choisir AVD



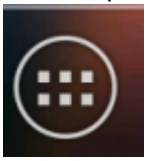
Cliquer sur Run



L'application est installée dans AVD



Sur l'emulateur (AVD) cliquer sur 

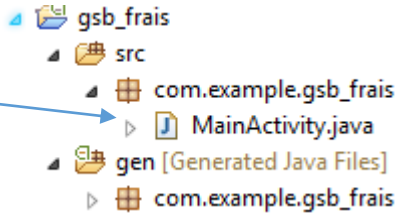
Puis sur 

On obtient toutes les applications installées sur AVD

Cliquer sur 

Comprendre le code

Éditer le source principal de l'application, MainActivity.java



Activity est la classe de base du composant visuel interactif de l'application.
La classe MainActivity étend cette classe et redéfnit la méthode onCreate(). L'appel setContentView() présente l'interface
utilisateur en désérialisant une ressource de layout (arrangement géométrique).
La deuxième surcharge prépare la mise en place d'un menu déroulant d'options pour l'application.

```
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class MainActivity extends Activity {

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the
action bar if it is present.
		getMenuInflater().inflate(R.menu.main, menu);
		return true;
	}

}
```

Les ressources du projet sont stockées dans le dossier **res** avec notamment les sous-dossiers drawable-xxx, layout et values. Elles sont référencées dans le code par la construction ***R.<resourcefolder>.<resourcename>***.

L'icône de l'application est disponible sous différents formats pour s'adapter aux différentes résolutions d'appareils (avec en plus une version Web placée directement à la racine du dossier projet) :

ldpi : 36 x 36 pixels (obsolète)
mdpi : 48 x 48 pixels
hdpi : 72 x 72 pixels
xhdpi : 96 x 96 pixels
xxhdpi : 144 x 144 pixels

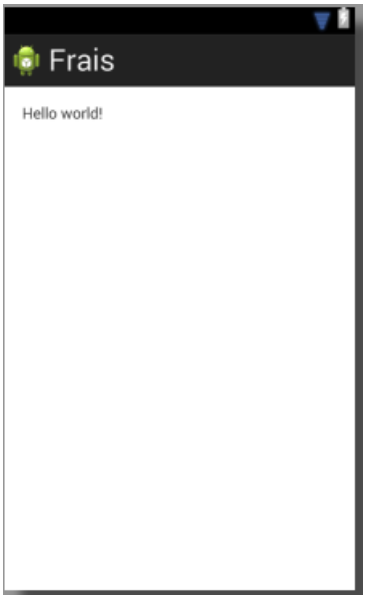
Le fchier res/layout/activity_main.xml fournit les éléments de construction de l'interface utilisateur (UI) de l'application

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
     android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

Les fichiers XML peuvent être ouverts en mode texte ou en mode graphique grâce aux onglets situés en bas de a fenêtre d'édition.



Le fichier **res/values/strings.xml** contient les définitions des chaines de l'application

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">My Hello</string>
<string name="action_settings">Settings</string>
<string name="hello_world">Hello world!</string>
</resources>
```

Même si l'architecture de l'application semble un peu compliquée pour afficher un simple message, c'est la bonne démarche !

La structure proposée présente plusieurs avantages :

- la logique de l'application et sa présentation sont parfaitement découplées,
- le placement des éléments d'interface est beaucoup plus facile à traiter avec une arborescence XML que par de code,
- tous les textes sont centralisés, ce qui facilite l'internationalisation de l'application

Construction de l'interface utilisateur : Vues

La vue est la partie visuelle d'une activité

Les vues sont stockées dans le dossier l**ayout** de **res**

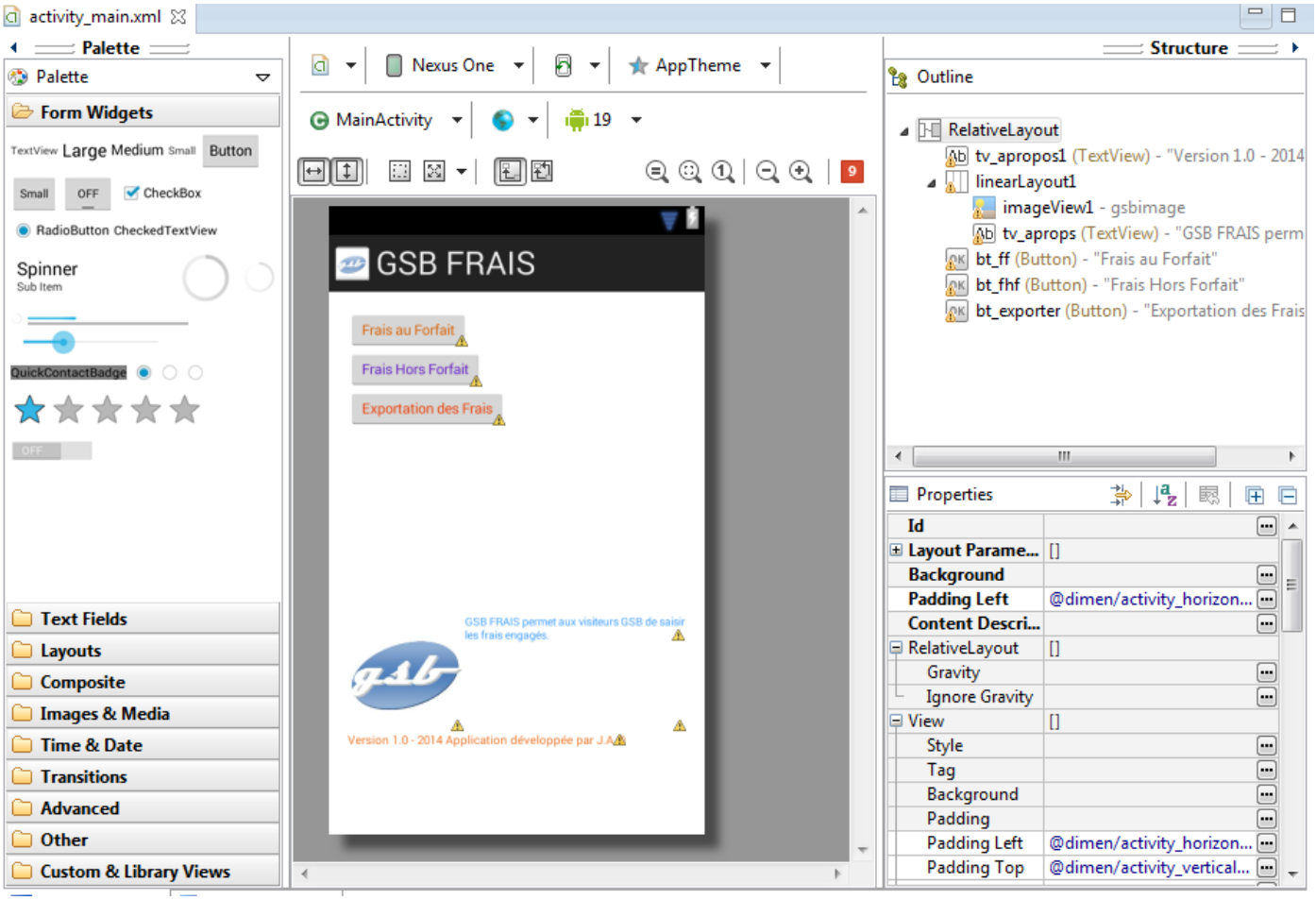
Ici la vue créer par défaut est : activity_main.xml

Une application Android est composée d'activités
Une activité = 1 page, avec des contrôles (TextView, EditText, Button...)

Cliquer sur activity_main.xml, on obtient l'interface graphique pour construire notre première vue

Une interface utilisateur Android est constituée d'une hiérarchie d'objets (bouton, image, label,...), chacun de ces objets correspond à une sous-classe de la classe View ; comme par exemple TextView pour les textes.

Les objets sont paramétrés par des propriétés facilement modifiables à partir de la vue graphique. La liste des propriétés de l'objet sélectionné apparaît dans la partie gauche :



Un élément défni en XML peut être facilement retrouvé dans le code, il sufft pour cela de lui attribuer un attribut id →

TextView myTxt = (TextView)findViewById(R.id.myTxt)

```
… …
<TextView
android:id="@+id/myTxt"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/hello_world"
/>
… …
```