

IN01

# Programmation Android

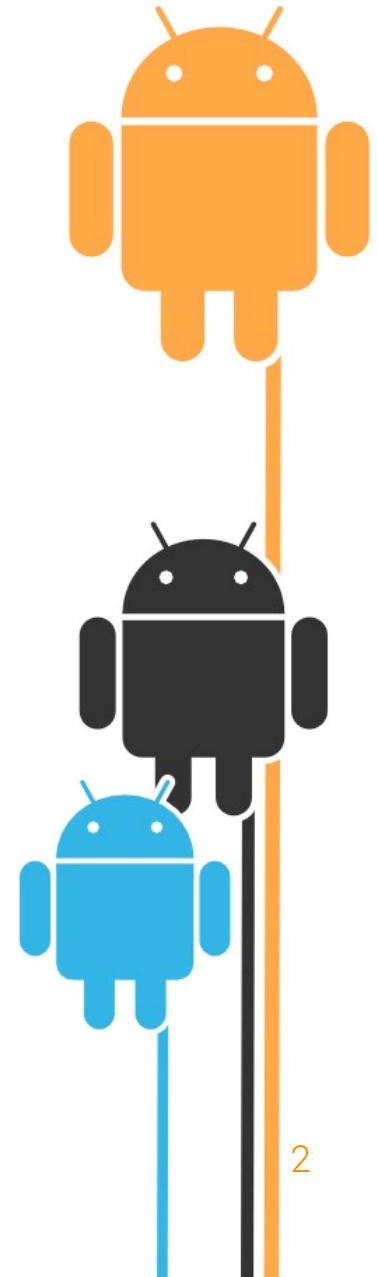
06 – Publication

Yann Caron

le cnam

# Sommaire - Séance 06

- Signature
- Publication
- Tests Bêta et Alpha
- Monétisation
- AdMob
- In-app billing version 3
- Google Analytics



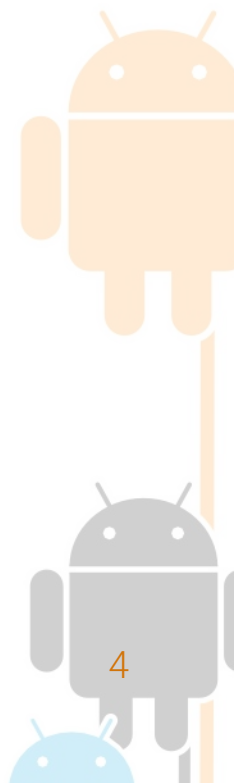
# IN01 – Séance 06

Signature



# Signer son APK

- Qu'est-ce qu'une signature ?
  - C'est une clé (algorithme de cryptage) qui identifie le développeur de l'application sur le store
- Lors de la phase debug, on utilise implicitement une clé par défaut
  - emplacement : `[user home].android/debug.keystore`
  - keystore name : `debug.keystore`
  - keystore password : `android`
  - key alias : `androiddebugkey`
  - key password : `android`
  - CN : `CN=Android Debug,O=Android,C=US`



# Attention à son keystore

- Il ne faut surtout pas perdre sa clé, sinon on ne peut plus publier la même application
- Il ne faut pas la partager, une clé est liée au développeur ou à l'entreprise
- Il faut bien choisir les alias et mots de passe et ne pas les divulguer
- Il faut les retenir
  - ➔ Comme pour la carte bancaire, il ne faut pas mettre le code avec



# Créer une clé

- Avec Android Studio :
  - ➔ Menu build / generate signed APK
  - ➔ Puis create new keystore

**New Key Store**

Key store path:

Password:  Confirm:

Key

Alias:

Password:  Confirm:

Validity (years):

Certificate

First and Last Name:

Organizational Unit:

Organization:

City or Locality:

State or Province:

Country Code (XX):

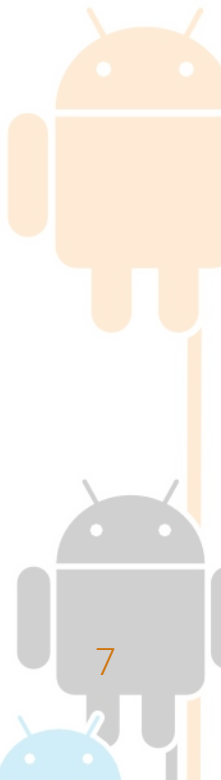
OK Cancel

# En ligne de commande

- Grâce à l'outil keytool (JAVA\_HOME/bin)

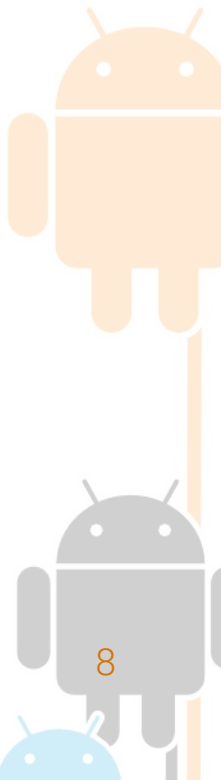
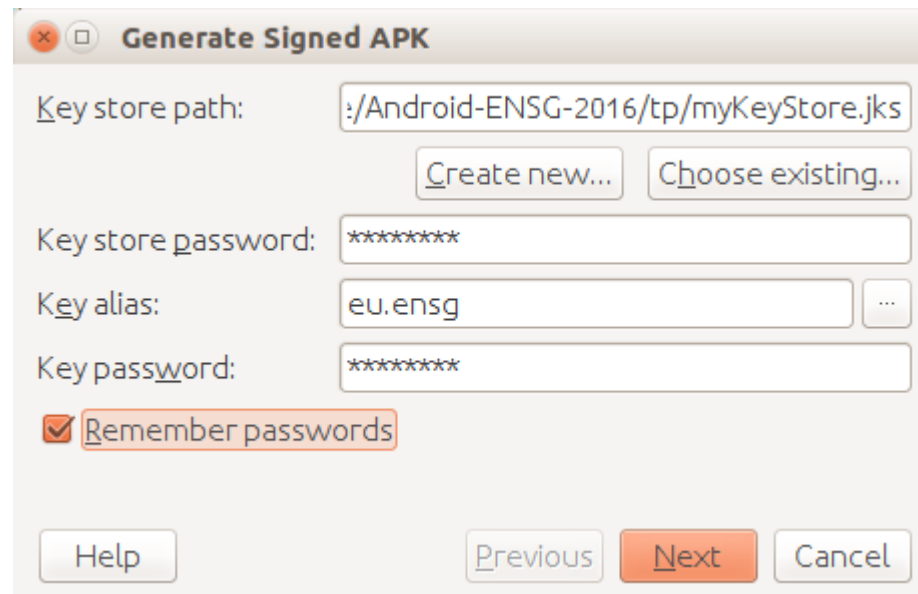
```
keytool -genkeypair -v -keystore myKeystore.keystore -storepass  
myKeystorePassword -alias myKey -keypass myKeyPassWord -keyalg RSA  
-validity 36500
```

- -keystore : le nom et le chemin de la clé
- -storepass : le mot de passe du keystore
- -alias : le nom de la clé
- -keypass : le password de la clé



# Signer l'application

- Une fois le keystore créé, Android studio se charge de tout





# IN01 – Séance 06

Publication



# Création du compte développeur

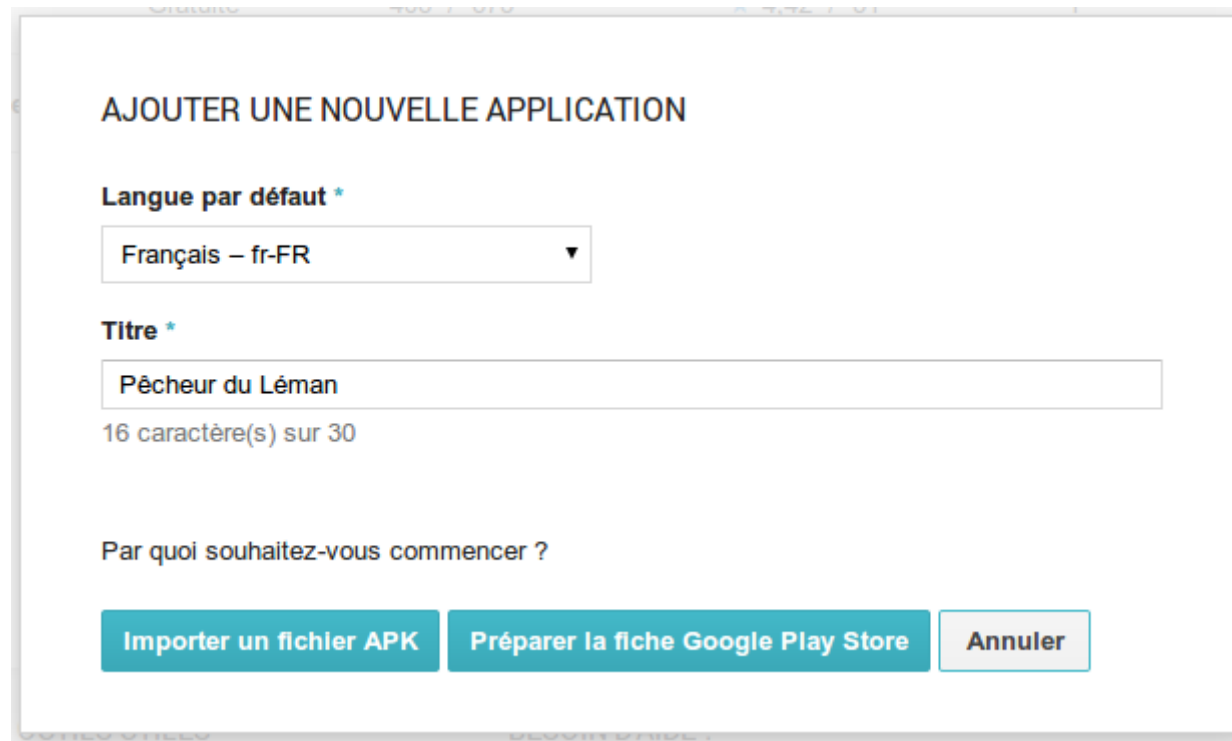
- Une fois l'application signée, elle est prête à être publiée sur un market
- Il existe plusieurs market android :
  - ➔ Officiel : le play store, frais d'inscription 20 € (<https://play.google.com/apps/publish>)
  - ➔ Autres possibilités :
    - Slide me (<http://slideme.org/>)
    - GetJar (<http://www.getjar.mobi/>)
    - YAMM, appoke (ces domaines ne sont plus actifs pour le moment)



# Publication

- Dans l'interface web, on clique sur
- Et on crée la page en important le fichier apk signé

+ Ajouter une nouvelle application



AJOUTER UNE NOUVELLE APPLICATION

Langue par défaut \*

Français – fr-FR ▼

Titre \*

Pêcheur du Léman

16 caractère(s) sur 30

Par quoi souhaitez-vous commencer ?

Importer un fichier APK   Préparer la fiche Google Play Store   Annuler

# Publier sur le play store

The screenshot shows the Google Play Developer Console interface for the application 'Algoid - Programming language'. The app is published and its version 38 is in production. The interface includes a sidebar with navigation options, a top header with the app name and status, and a main content area with sections for APK files, production configuration, and a table of installed versions.

**Google Play Developer Console**

**Algoid - Programming language**  
fr.cyann.algoid [Afficher sur le Play Store](#)   
**PUBLIÉE** 11 décembre 2015 Annuler la publication de l'application

Publication standard ▼  
Traitement de la mise à jour en cours...  
[Envoyer la mise à jour](#)

[Passer en mode avancé](#)

**FICHIERS APK**

**PRODUCTION**  
Version  
**38**

**TESTS BÊTA**  
Configurez des tests bêta pour votre application.

**TESTS ALPHA**  
Configurez des tests alpha pour votre application.

**CONFIGURATION DE LA VERSION EN PRODUCTION** [Importer un nouveau fichier APK en version production](#)

**FICHIER APK ACTUEL** date de publication : **11 déc. 2015 07:42:57**

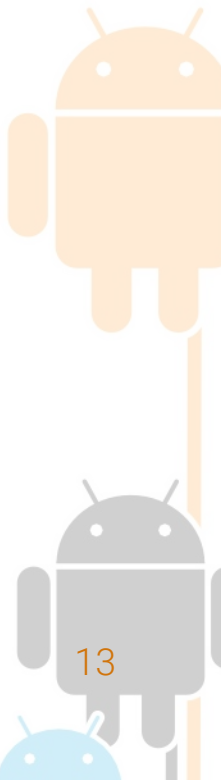
**Appareils compatibles**  
**11376**  
[Afficher la liste](#)

**Appareils exclus**  
**0**  
[Gérer les appareils exclus](#)

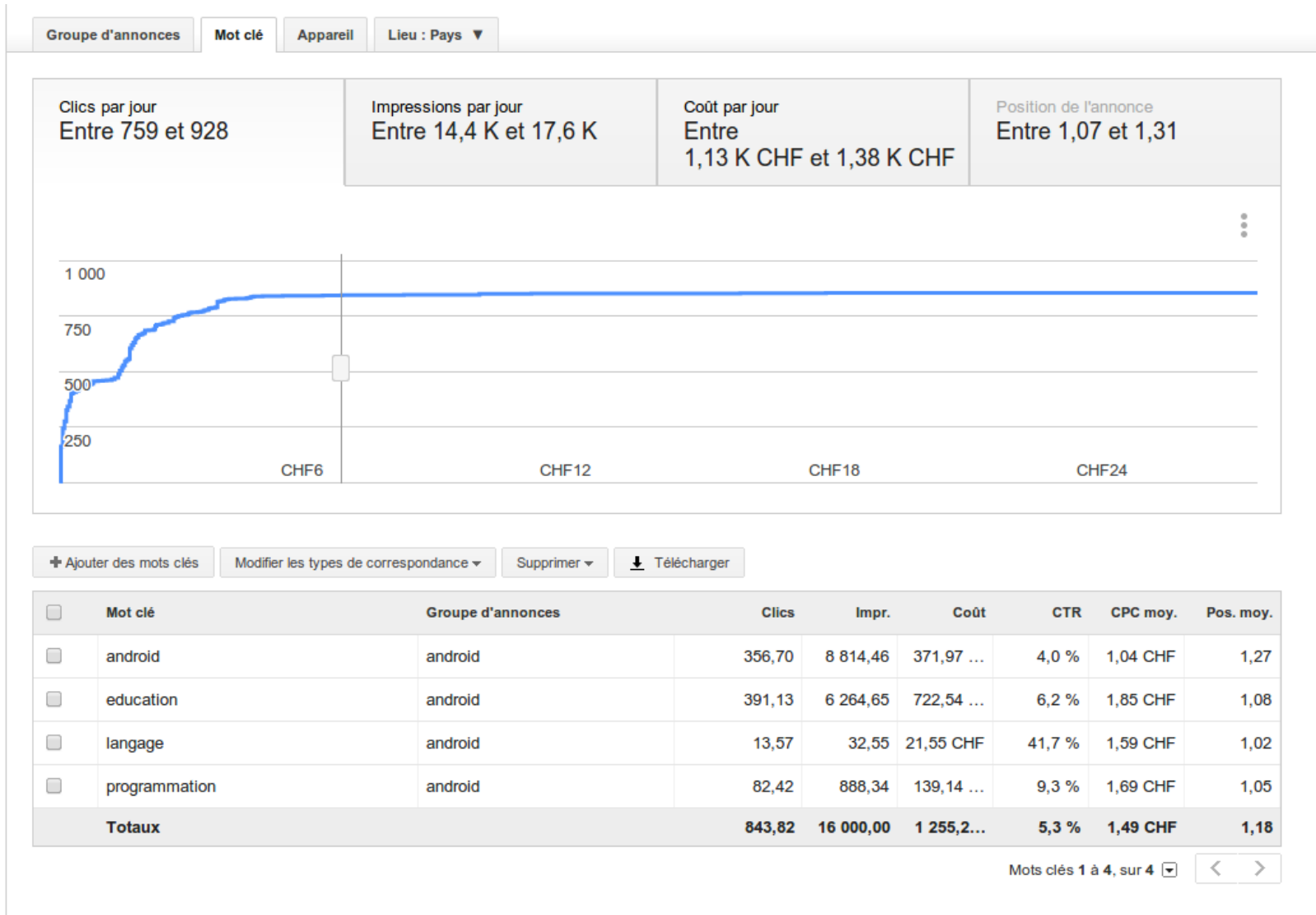
▼ VERSION	IMPORTÉ LE	ÉTAT	ACTIONS
<b>38 (1.3.0)</b>	<b>11 déc. 2015</b>	<b>en production</b>	

# Publication – Titre et description

- Il faut créer une description ni trop courte, ni trop longue (2000 caractères suffisent)
- Bien choisir les mots-clés utilisés et ne pas hésiter à répéter les plus importants
- Pour cela, on fait un audit sémantique
  - <https://adwords.google.com/ko/KeywordPlanner>
- Bien choisir ses mots-clés. Placer les plus importants dans le titre si possible

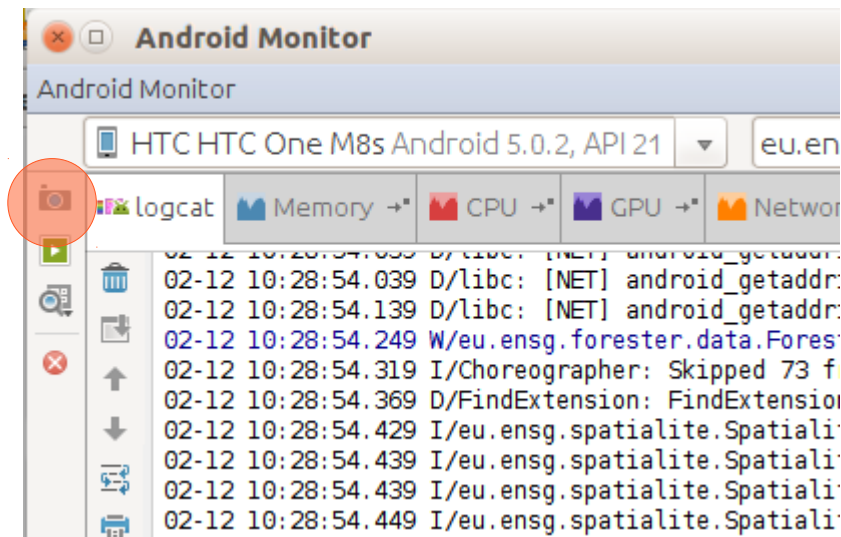


# Mots-clés



# Images

- Il faut ajouter un maximum de captures d'écran les plus attractives possible
- Android studio possède de screenshot dans l'Android Monitor




# Vidéo

- La vidéo de présentation est très importante, c'est souvent elle qui déclenche le téléchargement ou l'achat
- Il faut la publier sur YouTube et référencer la vidéo dans la page du play store
- Des outils permettent de prendre une vidéo :
  - ➔ Depuis l'appareil, mais il faut le rooter
  - ➔ Depuis le pc connecté à l'appareil, le taux de rafraichissement est épouvantable
  - ➔ Le screen capture d'un émulateur (meilleur compromis) ex. : sur Ubuntu, **recordMyDesktop**






# Exemple

 Google play

Rechercher

+Yann



## Algoid - Programming language

CyaNn - 9 mai 2014  
Enseignement

Installée

Cette application est compatible avec l'ensemble de vos appareils.

★★★★☆ (353)


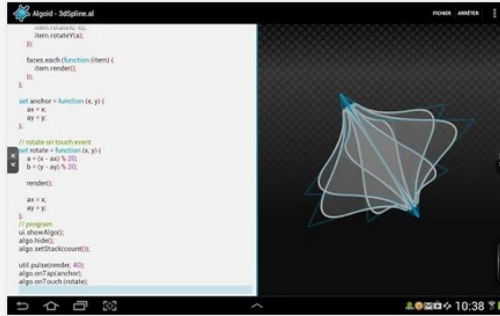
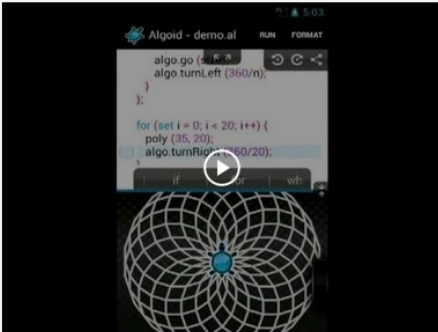
+417 Recommander ce contenu sur Google

Mes applications

Acheter

Jeux

Choix de l'équipe



### Description

Algoid - Langage de programmation éducatif pour enfants et débutants.  
Vous voulez "apprendre à programmer" ? Algoid est fait pour vous !

Algoid est une application éducative pour Android qui apprend aux enfants, aux adolescents et pourquoi pas à leurs parents "comment programmer".  
★★★ Avec Algoid, l'apprentissage de la programmation devient simple et amusant!

Vous souvenez vous du langage de programmation Logo ? Cette tortue (ou robot) amicale qui introduisait étape par étape les notions basiques de la programmation.

★★★ Algoid va plus loin, il embarque un débogueur (debugger) temps-réel (le premier et jusqu'à présent le seul sur Android), un mode d'exécution étape par étape et un explorateur de portée (pour une meilleure compréhension).

Encore plus loin, cachez la tortue et développez votre propre JEU VIDEO directement depuis votre téléphone / tablette (actuellement en phase d'optimisation).

En bref, Algoid va transformer votre téléphone / tablette en une vraie station de développement dédié à l'apprentissage de la programmation. Tout ce dont vous avez besoin pour devenir un futur développeur, que vous soyez enfant, adolescent ou adulte !

# IN01 – Séance 06

Tests Bêta et Alpha



# Tests Bêta

- Avant de publier une application, il est possible de la tester et de la faire tester par des utilisateurs de confiance
- Google met à disposition deux onglets dans la console pour développeur :
  - ➔ TESTS BÊTA : tests avant mise en production
  - ➔ TESTS ALPHA : tests préliminaires



# Gérer les testeurs

- Il faut créer un groupe Google (ou une communauté Google+)
- <https://groups.google.com/forum/#!/creategroup>

**Groupes**

← CRÉER Annuler

Mes groupes  
Accueil  
Favoris

► Favoris

Cliquez sur l'icône en forme d'étoile d'un groupe pour ajouter ce dernier à vos favoris

▼ Consultés récemment

**Nom du groupe** Algoïd testers

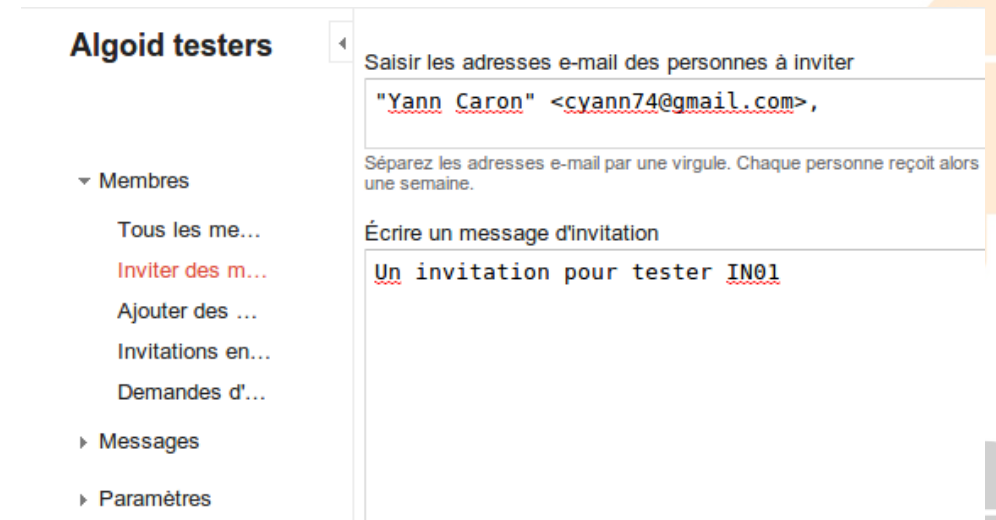
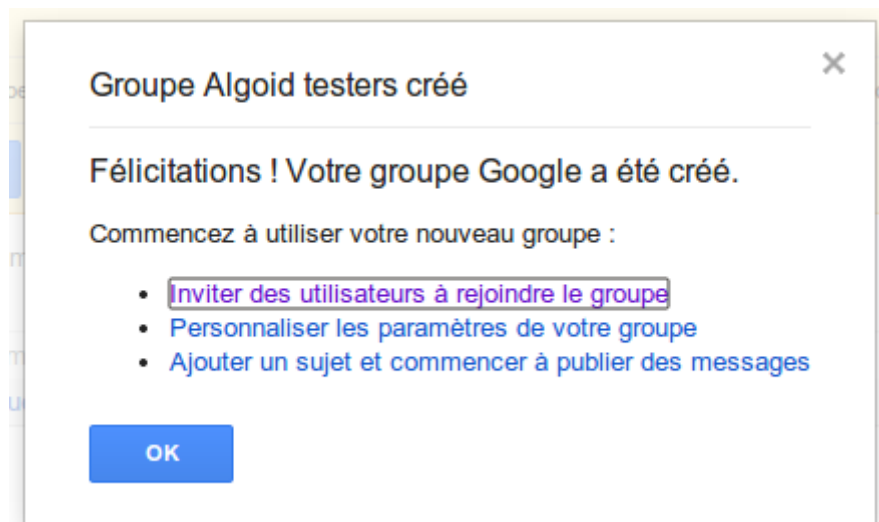
**Adresse e-mail du groupe** algoïd@googlegroups.com  
<https://groups.google.com/d/forum/algoïd>

**Description du groupe** Groupe de test Algoïd !

Adresse du groupe à introduire dans la console développeur

# Envoyer une invitation

- Il faut ensuite envoyer une invitation aux testeurs



# Configurer le groupe

- De retour dans la console pour développeurs Google Play, il faut cliquer sur “Gérer la liste des testeurs” et introduire l'URL du groupe

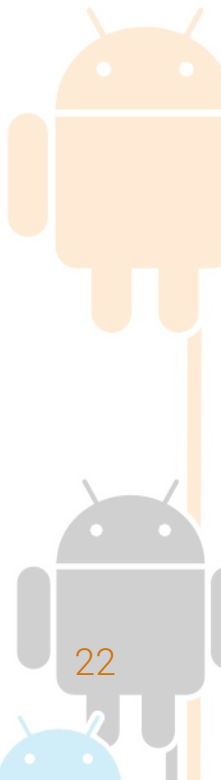
**QUI PEUT TESTER VOTRE APPLICATION EN VERSION ALPHA ?**

Vous pouvez ajouter des groupes Google ou des communautés Google+ pour tester votre application. Après avoir ajouté un groupe, vous devez envoyer le lien d'inscription ci-dessous aux testeurs. Une fois inscrits, ces derniers recevront cette version de test via Google Play.

**Ajoutez des groupes Google ou des communautés Google+**

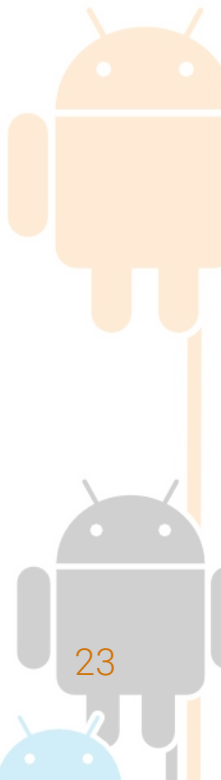
**Inscription des testeurs**

Votre application n'est utilisable par les testeurs que lorsqu'elle est publiée sur Google Play. Si elle ne comporte pas de fichier APK de production, elle n'est visible que par les testeurs des versions alpha et bêta. Le lien que les testeurs utilisent pour s'inscrire s'affiche ici une fois l'application publiée.



# Envoyer aux testeurs

- Pour que le lien soit disponible, il faut publier l'application (si aucune APK en production, l'application ne sera visible que par des testeurs)
- Le lien apparaît dans la fenêtre de gestion des testeurs



# Envoyer aux testeurs

- Après envoi à la liste des testeurs (via le groupe par exemple), il reçoit une invitation
- Il peut télécharger l'APK sur le Play Store





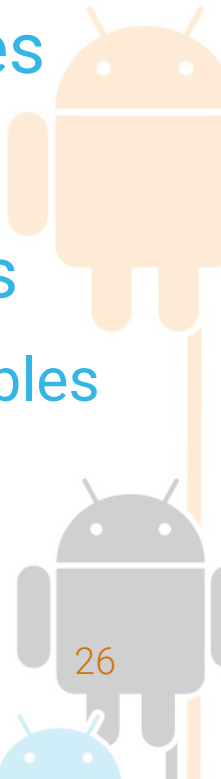
# IN01 – Séance 06

Monétisation



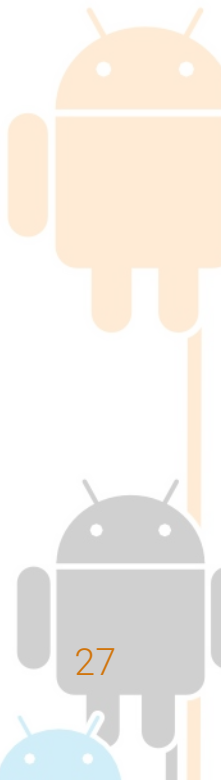
# Monétisation - Direct

- Il existe plusieurs moyens de rentabiliser une application Android
- Modèle direct : application payante
  - ➔ Les utilisateurs Android sont réticents à acheter des applications, même à quelques dizaines de centimes
  - ➔ Peut être un facteur limitant à son succès
  - ➔ Convient à des applications complexes et très utiles
    - Exemple : AIDE, jeux 3D, legacy games, outils indispensables



# Monétisation – Version d'essai

- On publie deux applications sur le store, l'application payante et sa version d'essai
  - ➔ Attention à la gestion des projets ! Le playstore ne supporte qu'un seul nom de package. Il faut créer un project library (le core system) et deux applications (des enveloppes )
  - ➔ Il faut penser à communiquer depuis la version d'essai pour inciter l'achat



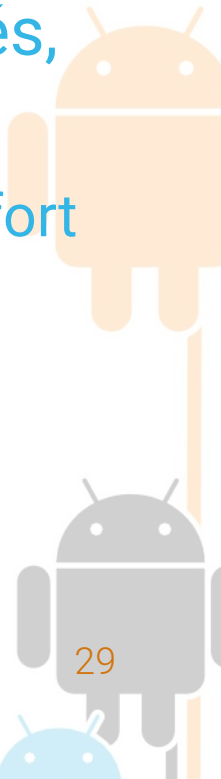
# Monétisation - Donate

- Une application payante libre : donnateware.
  - ➔ Il suffit de mettre une application payante vide sur le Play Store en indiquant que c'est un don
  - ➔ Ne pas oublier de communiquer dans l'application principale à quel endroit effectuer ce don
- PayPal peut également être intégré via une page web et une webview
- Convient aux applications éducatives, *open source*



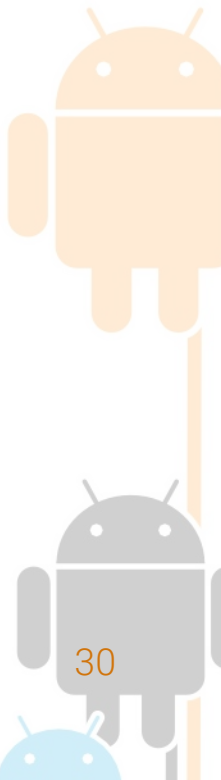
# Monétisation - Freemium

- En général, moins de 5 % de l'audience d'une app est prête à acheter la version payante ou à faire un don (il faut une forte audience)
- Il existe une parade, le freemium, où “free to play”, est une application gratuite qui propose lors de l'utilisation, l'achat de matériel virtuel (plus de vie pour un jeu, des fonctionnalités, etc.)
  - ➔ Ce modèle est plus adapté aux apps à faible audience, mais à fort intérêt (un jeu addictif, une application ayant une audience spécifique)
  - ➔ Voir chapitre sur l'inapp billing



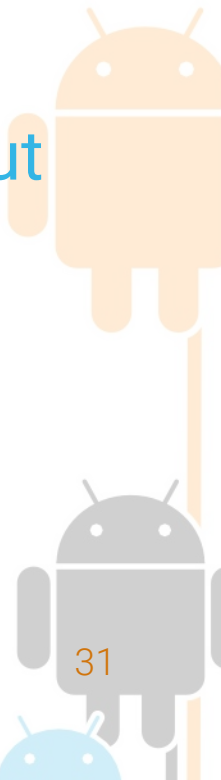
# Monétisation - Publicité

- Il existe deux types de publicité :
  - ➔ Les bannières
    - Une bannière de publicité en bas ou en haut de l'écran
    - Réduit le confort d'utilisation
  - ➔ Les interstitiels
    - Une publicité plein écran entre deux actions
    - Il faut bien gérer le retour à l'application de l'utilisateur
- Les utilisateurs s'en lassent et notent souvent négativement l'application
- Les publicités ne ciblent souvent pas bien le public (se renseigner sur l'Ad Provider)



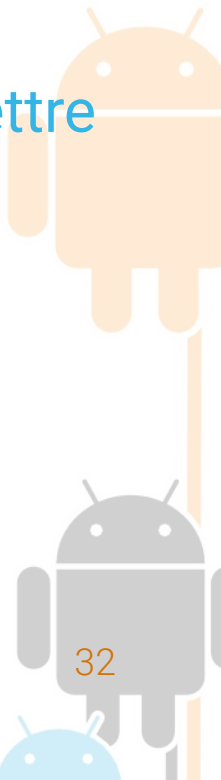
# Monétisation - Autres

- Le sponsoring est un moyen de monétiser son application
  - ➔ Permet de mieux cibler le public
- Utiliser les plateformes d'investissement participatif (Crowdfunding)
  - ➔ Attention, le logiciel est difficilement défendable, il faut une extension matérielle, quelque chose qui sera palpable pour l'investisseur
- Abonnements utilisateurs



# Monétisation - Conclusion

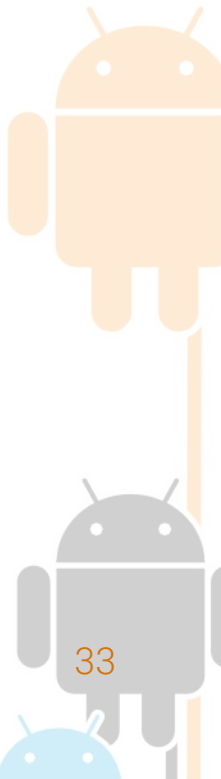
- Pour plus d'informations : <http://monetizepros.com/blog/2013/101-ways-to-make-money-with-your-iphone-android-or-mobile-app/>
- Il faut souvent essayer plusieurs techniques pour trouver la plus adaptée à son application
- Parfois même les mixer, par exemple des bannières, des interstitielles dans une version d'essai
- Attention au message véhiculé. Il est par exemple difficile de mettre de la publicité dans une application destinée à un jeune public
- C'est tout un art !!!! Il faut trouver le bon moyen entre popularité, qualité et rendement





# Monétisation – Quelques questions

- L'app vise-t-elle un marché de niche ?
  - oui : application payante > 1 €
- Quelle valeur lui accorderait le public ?
  - < 0,79 € : publicité semble envisageable
- Sur quelle plateforme ?
  - iOS : les utilisateurs préfèrent payer et être tranquilles
  - Android : les utilisateurs préfèrent la publicité
  - WP : attention, pas d'in-app purchase possible
- L'app est-elle modulaire ?
  - L'in-app purchase semble envisageable
- Y a-t-il des coûts à long terme
  - La publicité peut être une solution pour amortir les coûts
  - L'abonnement semble également envisageable



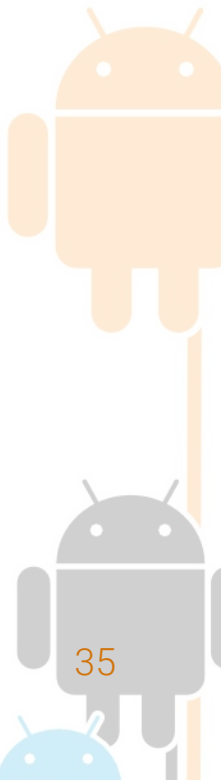
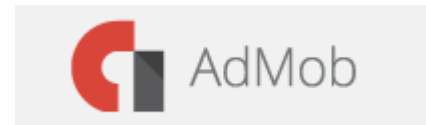
# IN01 – Séance 06

AdMob



# AdMob

- L'Ad Provider de Google
- Fonctionne selon deux modes
  - ➔ Monétisation : être payé pour diffuser de la publicité
  - ➔ Promotion : acheter de l'espace publicitaire pour se faire connaître
- Plusieurs étapes :
  - ➔ S'inscrire sur : <https://apps.admob.com>
    - Le plus simple est d'utiliser le compte Google+
  - ➔ Installer l'API dans l'application
  - ➔ Enregistrer l'application à monétiser



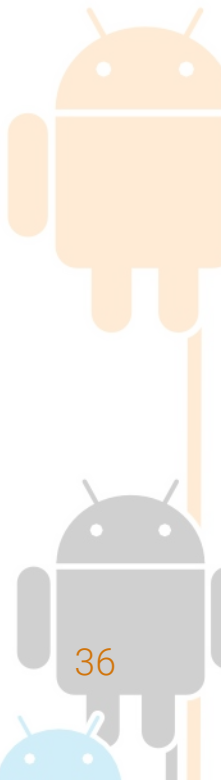
# Créer un bloque d'annonce

- Entrer les données dans le wizard

AdMob Accueil Monétiser Promouvoir Analyser

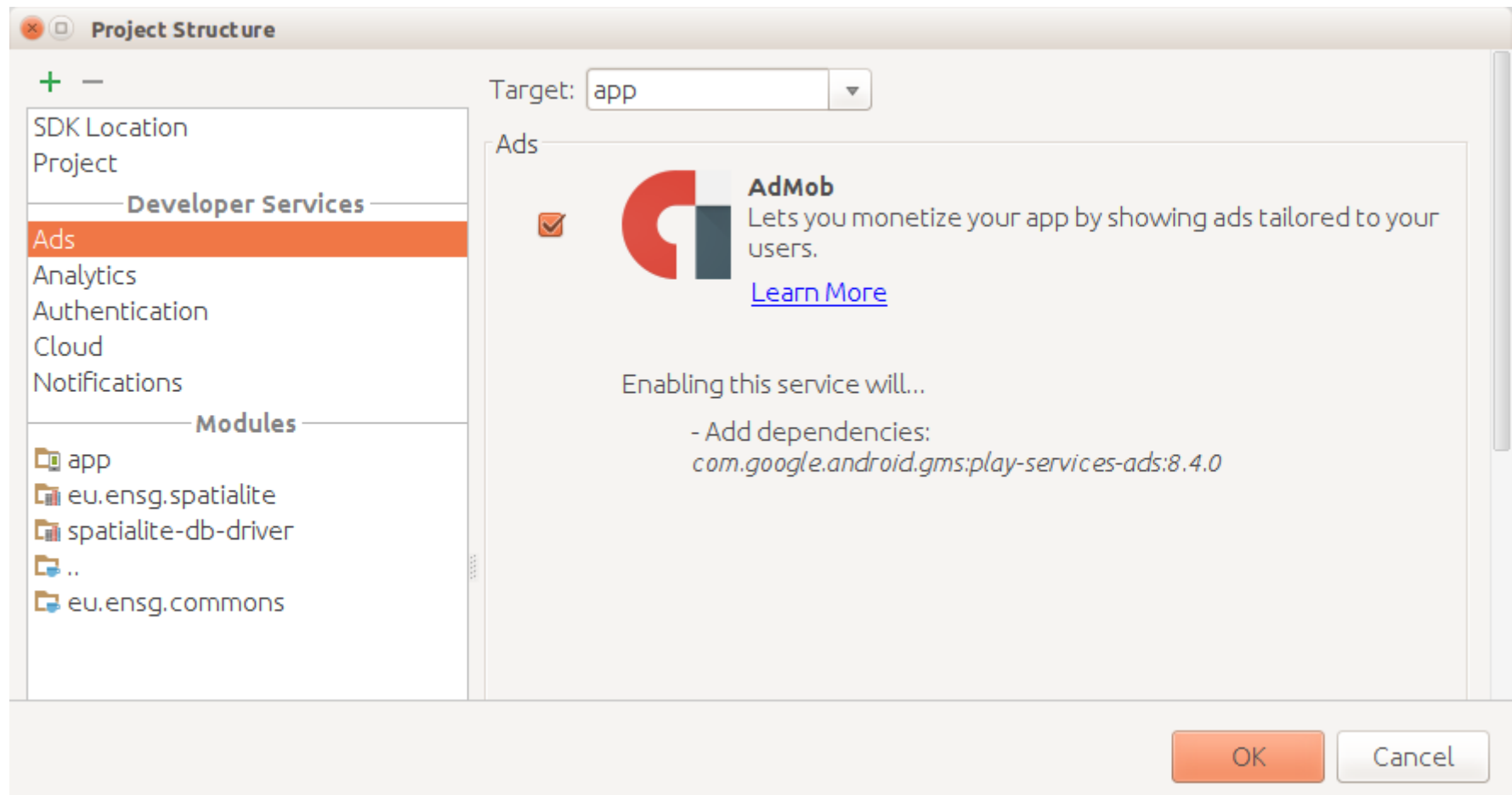
## Monétiser une nouvelle application

- ✓ Sélectionnez une application
  - eu.ensg.forester  
Android
  - ✓ L'application a été ajoutée à AdMob.
- ✓ Configurer Google Analytics pour cette application
  - Vue de création de rapports : eu.ensg.forester | Android > Vue par défaut
  - ID de suivi de l'application : UA-36056995-8
- ✓ Sélectionner le format de l'annonce et attribuer un nom au groupe d'annonces
  - Nom du bloc d'annonces : Bannière
  - ID du bloc d'annonces : ca-app-pub-5359459930746730/2929123072



# Installation

- Dans le menu File / project structure



# Ajouter l'ID

- L'identifiant du bloque annonce est fournit par le site AdMob
- Il doit être enregistré dans le fichier `Strings.xml`

```
<resources>
    ...
    <string name="banner_ad_unit_id">ca-app-pub-5359459930746730/2929123072</string>
</resources>
```

# Ajouter la bannière

- Dans les activities ou les fragments, il faut ajouter la bannière

```
<com.google.android.gms.ads.AdView  
    android:id="@+id/adView"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_centerHorizontal="true"  
    android:layout_alignParentBottom="true"  
    ads:adSize="BANNER"  
    ads:adUnitId="@string/banner_ad_unit_id">  
</com.google.android.gms.ads.AdView>
```

- Plusieurs adSize : BANNER, FULL\_BANNER, LARGE\_BANNER....
  - ➔ À retenir SMART\_BANNER : taille automatique
  - ➔ WIDE\_SKYSCRAPER : vertical

# Charger la bannière

- Dans la méthode onCreate de l'activity ou du fragment, il faut initialiser une requête

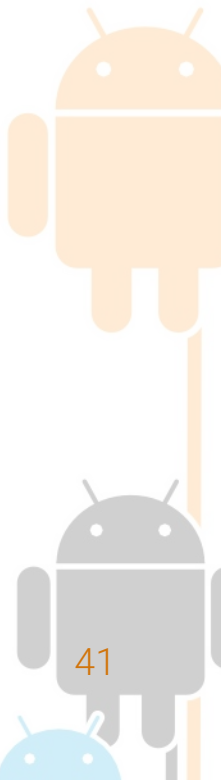
```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_login);  
  
    AdView adView = (AdView) findViewById(R.id.adView);  
    AdRequest adRequest = new AdRequest.Builder().build();  
    adView.loadAd(adRequest);  
}
```



# Callbacks

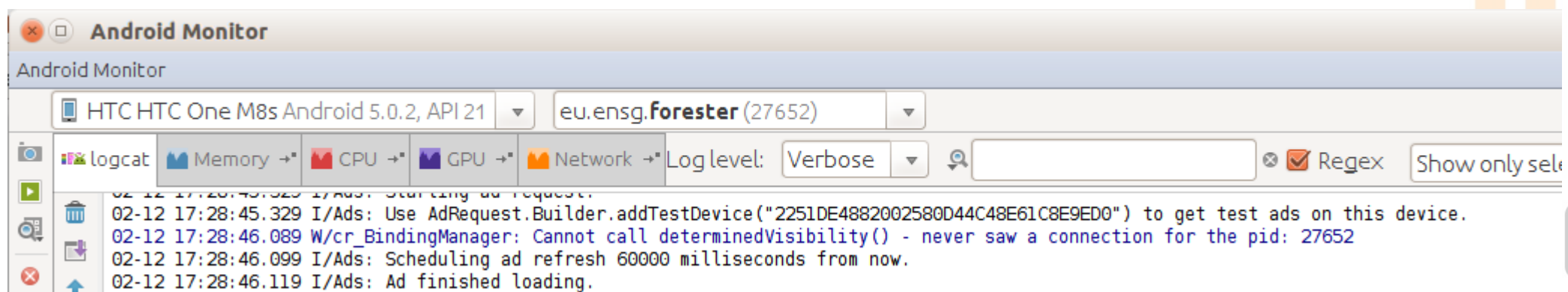
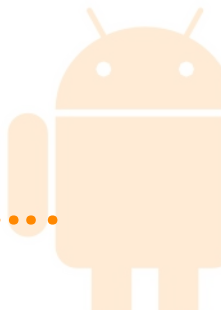
- Récupérer des évènements et effectuer des traitements

```
adView.setAdListener(new AdListener() {  
    @Override  
    public void onReceiveAd(Ad arg0) {}  
  
    @Override  
    public void onPresentScreen(Ad arg0) {}  
  
    @Override  
    public void onLeaveApplication(Ad arg0) {}  
  
    @Override  
    public void onFailedToReceiveAd(Ad arg0, ErrorCode arg1) {}  
  
    @Override  
    public void onDismissScreen(Ad arg0) {}  
});
```



# TestDevice

- Pour éviter que la publicité n'apparaisse lorsque l'on débogue l'application, il existe le mode de test
- Celui-ci fonctionne par appareil
- La clé est indiquée dans le logcat et doit ressembler à : **To get test ads on this device ...**



# TestDevice

- Merci Eclipse
- Il suffit ensuite de recopier la ligne de code indiquée dans le code de l'activity (onCreate en général)

```
adRequest.addTestDevice("F61A3C0A1FC*****");
```

- Et voilà le travail ==>



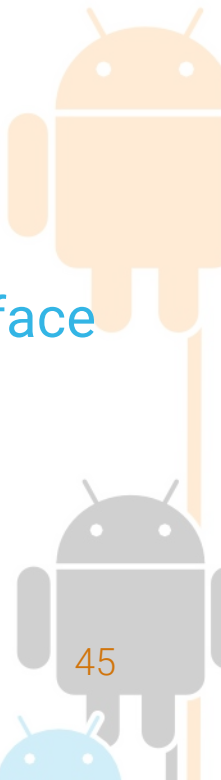
# IN01 – Séance 06

Intégrer In-app billing Version 3



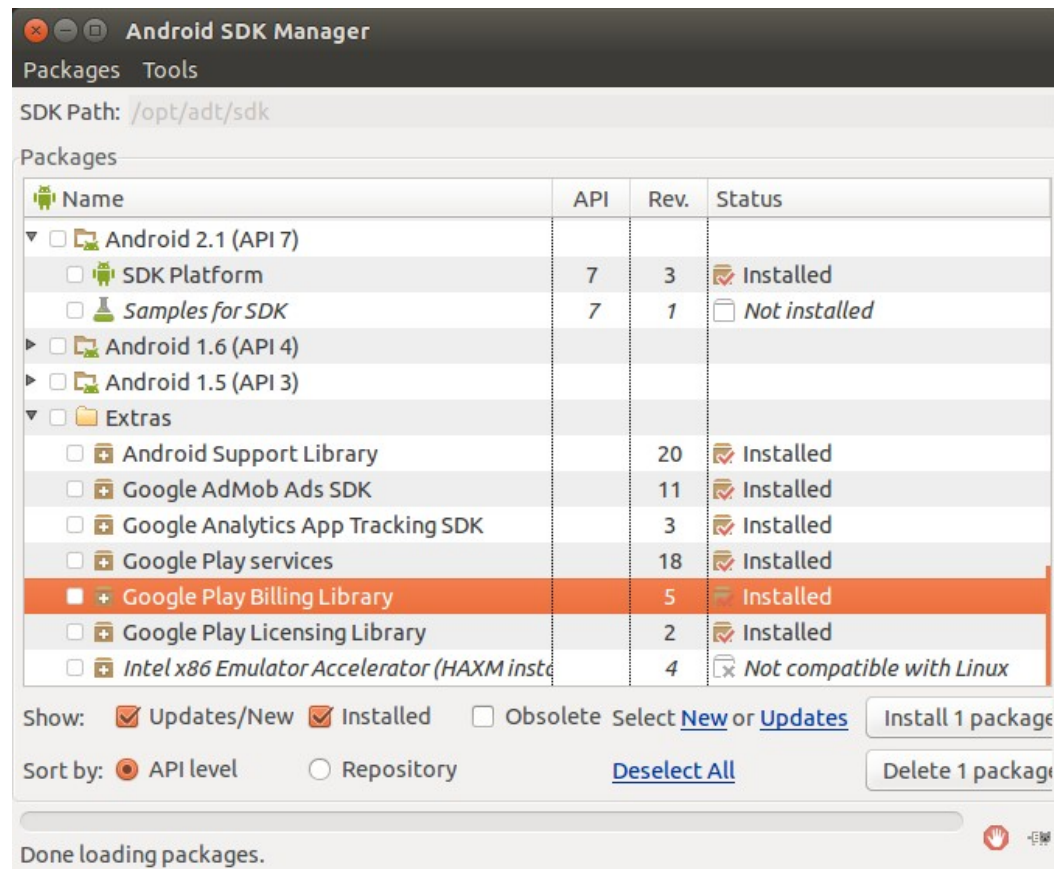
# Vue d'ensemble

- Plusieurs étapes nécessaires à la facturation :
  - ➔ Installer la bibliothèque (fichier AIDL) dans l'application
  - ➔ Mettre à jour le manifest
  - ➔ Créer un **ServiceConnection** et le lier à un **IInAppBillingService**
  - ➔ Créer des éléments à vendre dans le Play Store
- Ensuite, il est possible de :
  - ➔ Obtenir la liste et la description des produits
  - ➔ Envoyer les requêtes de facturation depuis l'application vers l'interface **IInAppBillingService** et récupérer la réponse du service de facturation de Google Play



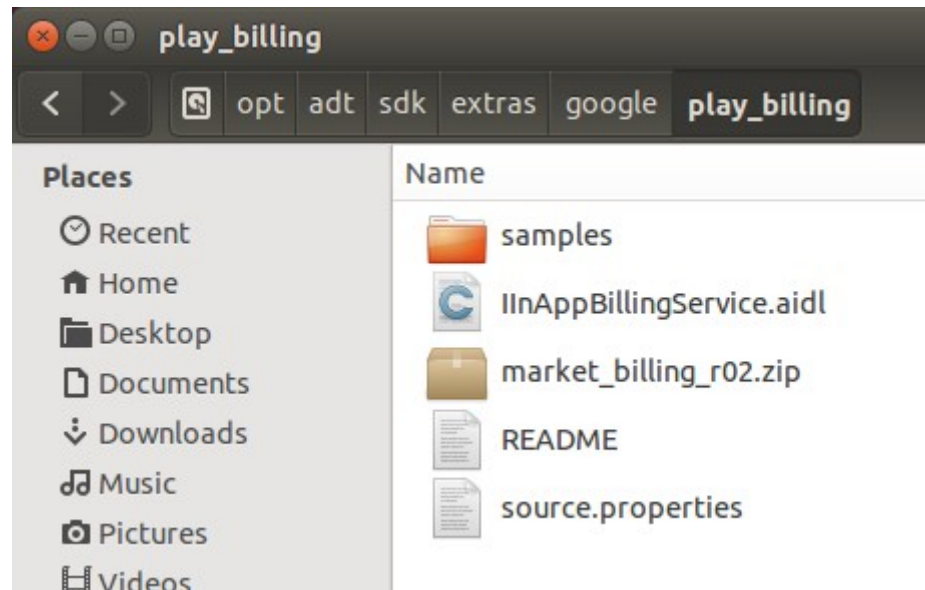
# 1 - Installation

- Télécharger la bibliothèque depuis l'Android SDK Manager



# 1- Installation

- Dans le dossier `[sdk]/extras/google/play_billing`, copier le fichier `IInAppBillingService.aidl` dans le dossier `src` du projet



# 1 - Installation

- Dans le projet, créer un package `com.android.vending.billing`
- Y copier le fichier `IInAppBillingService.aidl`
- Un fichier AIDL (Android Interface Definition Language) est une interface pour échanger des données via l'Interprocess Communication (IPC)
- C'est une bibliothèque qui facilite la communication interprocessus
- Le fichier AIDL indique au compilateur de générer le fichier `gen/com.android.vending.billing/IInAppBillingService.java` qu'on pourra utiliser par la suite

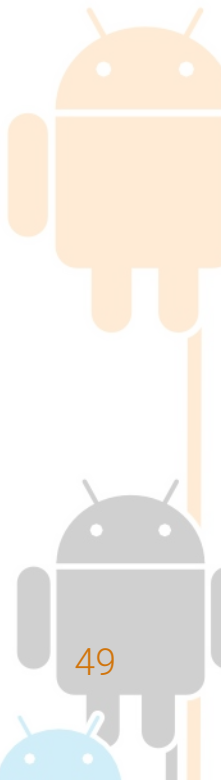


## 2 - Manifest

- Ajoute la permission BILLING dans le manifest

```
<uses-permission  
android:name="com.android.vending.BILLING" />
```

- On commence à en avoir l'habitude
- Il faut penser à ajouter la permission internet



# 3 – Créer le ServiceConnection

- Il faut créer un objet qui implémente l'interface ServiceConnection

```
IInAppBillingService billingService; ◀ Variable d'instance

ServiceConnection serviceConn = new ServiceConnection() {
    @Override
    public void onServiceDisconnected(ComponentName name) {
        billingService = null;
    }

    @Override
    public void onServiceConnected(ComponentName name,
        IBinder service) {
        billingService = IInAppBillingService.Stub.asInterface(service);
    }
};
```

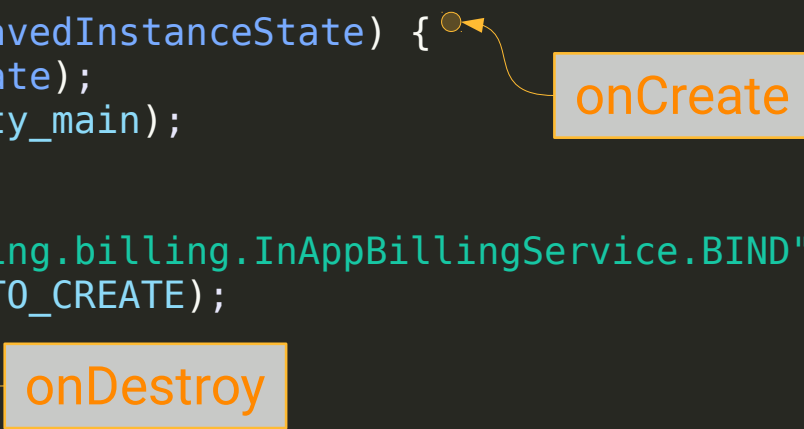
# 3 – Binder le ServiceConnection

- Pour lier (binder), on utilise un Intent
- Il faut penser à délier ce qui a été lié

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

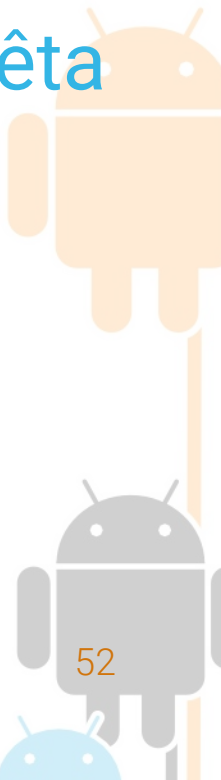
    bindService(
        new Intent("com.android.vending.billing.InAppBillingService.BIND"),
        serviceConn, Context.BIND_AUTO_CREATE);
}

@Override
public void onDestroy() {
    super.onDestroy();
    if (billingService != null) {
        unbindService(serviceConn);
    }
}
```

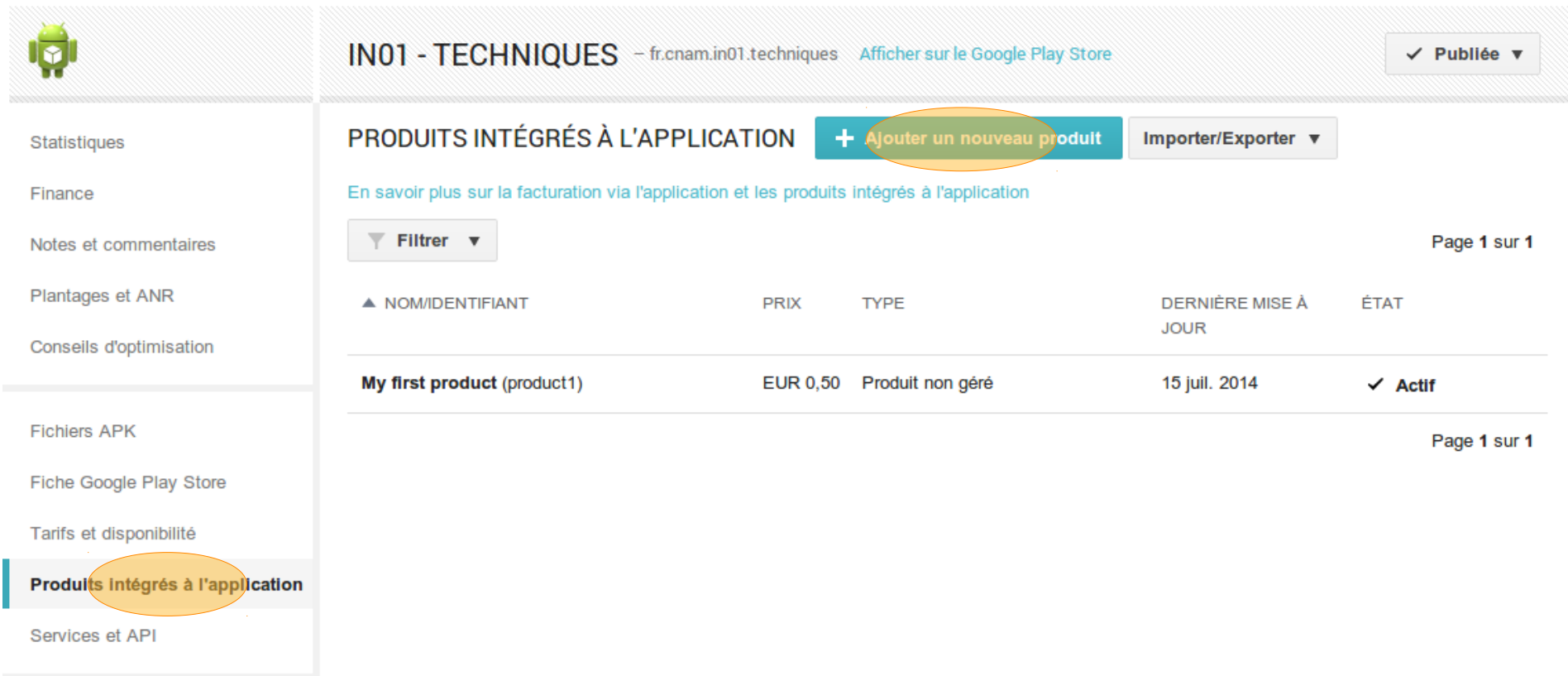


# 4 – Créer des produits intégrés

- Maintenant que notre service est lié, il faut créer des éléments à vendre... pour tester...
- De retour dans la console pour développeur de Google Play
- Dans notre application en mode test Alpha ou Bêta
- Attention !! L'application doit être publiée
  - ➔ Mais s'il n'existe pas d'APK de production, elle n'apparaît pas dans le Play Store



# 4 – Créer des produits intégrés



The screenshot displays the 'IN01 - TECHNIQUES' application interface. The left sidebar contains a menu with the following items: 'Statistiques', 'Finance', 'Notes et commentaires', 'Plantages et ANR', 'Conseils d'optimisation', 'Fichiers APK', 'Fiche Google Play Store', 'Tarifs et disponibilité', 'Produits Intégrés à l'application' (highlighted with an orange circle), and 'Services et API'. The main content area is titled 'PRODUITS INTÉGRÉS À L'APPLICATION' and includes a '+ Ajouter un nouveau produit' button (also highlighted with an orange circle) and an 'Importer/Exporter' dropdown menu. Below this, there is a 'Filtrer' button and a table of integrated products. The table has five columns: 'NOM/IDENTIFIANT', 'PRIX', 'TYPE', 'DERNIÈRE MISE À JOUR', and 'ÉTAT'. A single product is listed: 'My first product (product1)' with a price of 'EUR 0,50', type 'Produit non géré', last update '15 juil. 2014', and status '✓ Actif'. The page is marked 'Page 1 sur 1'.

IN01 - TECHNIQUES – fr.cnam.in01.techniques [Afficher sur le Google Play Store](#) ✓ Publiée ▼

PRODUITS INTÉGRÉS À L'APPLICATION + Ajouter un nouveau produit Importer/Exporter ▼

[En savoir plus sur la facturation via l'application et les produits intégrés à l'application](#)

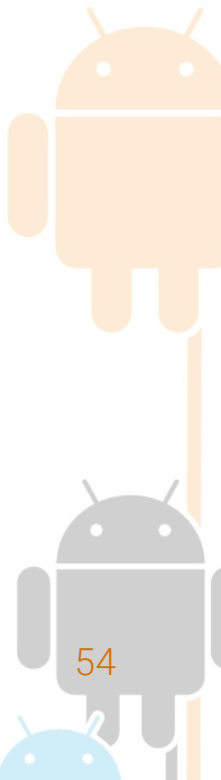
Filtrer ▼ Page 1 sur 1

▲ NOM/IDENTIFIANT	PRIX	TYPE	DERNIÈRE MISE À JOUR	ÉTAT
<b>My first product</b> (product1)	EUR 0,50	Produit non géré	15 juil. 2014	✓ Actif

Page 1 sur 1

# 4 – Créer des produits intégrés

- Il existe trois types de produits :
  - ➔ Produit géré : achat possible qu'une seule fois. Une fois acheté, la plateforme gère la possibilité de télécharger à nouveau le produit
  - ➔ Produit non géré : le produit peut être acheté plusieurs fois
  - ➔ Abonnement : facturation récurrente



# 4 – Créer des produits intégrés

## AJOUTER UN NOUVEAU PRODUIT

Quel type de produit voulez-vous ajouter ? \*

**Produit géré** Produit non géré Abonnement

Les articles gérés ne peuvent être achetés qu'une seule fois par compte utilisateur sur Google Play. Le site enregistre définitivement les informations de transaction pour chaque article et pour chaque utilisateur. [En savoir plus](#)

**ID de produit \***

8 caractère(s) sur 100

Veuillez noter que vous ne pourrez PAS modifier le type ni l'identifiant du produit par la suite, ni réutiliser celui-ci. [En savoir plus](#)

**Continuer** Annuler

ID du produit

# Obtenir la liste des produits

- Plusieurs étapes sont nécessaires
- On crée une liste des produits dont on veut la description

```
ArrayList<String> skuList = new ArrayList<String>();  
skuList.add("product1");  
skuList.add("product2");  
  
Bundle querySkus = new Bundle();  
querySkus.putStringArrayList("ITEM_ID_LIST", skuList);
```

Création de la liste

Contener à envoyer



# Obtenir la liste des produits

- On envoie la requête et on traite la réponse

```
Bundle skuDetails = billingService.getSkuDetails(  
    3, getPackageName(), "inapp", querySkus);  
  
int response = skuDetails.getInt("RESPONSE_CODE");  
if (response == 0) {  
    ArrayList<String> responseList = skuDetails.getStringArrayList("DETAILS_LIST");  
  
    for (String thisResponse : responseList) {  
        JSONObject object = new JSONObject(thisResponse);  
  
        String sku = object.getString("productId");  
        String price = object.getString("price");  
  
        Log.i(this.getClass().toString(),  
            "Product: " + sku + ", price: " + price + "\n");  
    }  
}
```

Envoie la requête

Traite le code réponse

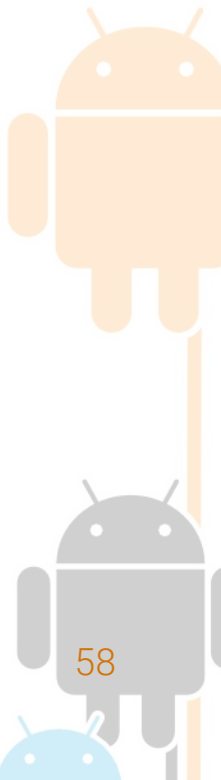
Traite la liste

C'est du JSON

# Codes réponses

- La valeur `RESPONSE_CODE` peut signifier plusieurs choses :

Response Code	Value	Description
<code>BILLING_RESPONSE_RESULT_OK</code>	0	Success
<code>BILLING_RESPONSE_RESULT_USER_CANCELED</code>	1	User pressed back or canceled a dialog
<code>BILLING_RESPONSE_RESULT_BILLING_UNAVAILABLE</code>	3	Billing API version is not supported for the type requested
<code>BILLING_RESPONSE_RESULT_ITEM_UNAVAILABLE</code>	4	Requested product is not available for purchase
<code>BILLING_RESPONSE_RESULT_DEVELOPER_ERROR</code>	5	Invalid arguments provided to the API. This error can also indicate that the application was not correctly signed or properly set up for In-app Billing in Google Play, or does not have the necessary permissions in its manifest
<code>BILLING_RESPONSE_RESULT_ERROR</code>	6	Fatal error during the API action
<code>BILLING_RESPONSE_RESULT_ITEM_ALREADY_OWNED</code>	7	Failure to purchase since item is already owned
<code>BILLING_RESPONSE_RESULT_ITEM_NOT_OWNED</code>	8	Failure to consume since item is not owned

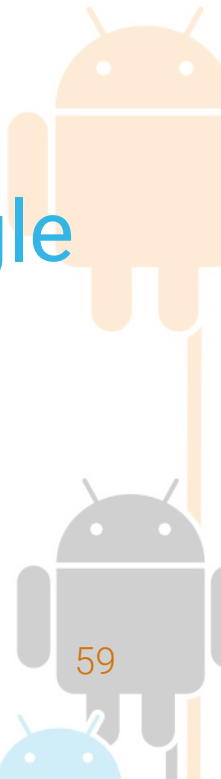


# Résultat

- Remarque, sur Android, on utilise souvent le JSON (dans LibGdx par exemple)

```
{ "title": "My first product (IN01 - Techniques)", "price": "0,50 €", "type":  
": "inapp", "description": "Description here", "price_amount_micros": 50000  
0, "price_currency_code": "EUR", "productId": "product1" }  
Product: product1, price: 0,50 €
```

- Pour le moment product2 n'est pas encore chargé. Il faut attendre l'approbation de Google



# Acheter un article

- On crée une requête grâce à un Intent

```
Bundle buyIntentBundle = billingService.getBuyIntent(
    3, getPackageName(), "product1", "inapp", "developer payload");

int response = buyIntentBundle.getInt("RESPONSE_CODE");

if (response == 0) {
    PendingIntent pendingIntent =
        buyIntentBundle.getParcelable("BUY_INTENT");

    startIntentSenderForResult(
        pendingIntent.getIntentSender(), 1001,
        new Intent(), Integer.valueOf(0),
        Integer.valueOf(0), Integer.valueOf(0));
}
```

Utile pour sécurité

On crée la requête

Un code d'identification

# Réponse dans l'évènement

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    if (requestCode == 1001) {
        int responseCode = data.getIntExtra("RESPONSE_CODE", 0);
        String purchaseData = data.getStringExtra("INAPP_PURCHASE_DATA");
        String dataSignature = data.getStringExtra("INAPP_DATA_SIGNATURE");

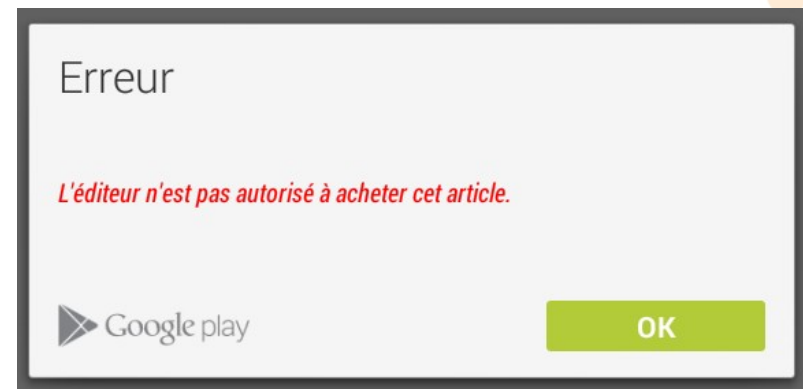
        if (resultCode == RESULT_OK) {
            JSONObject object = new JSONObject(purchaseData);
            String sku = object.getString("productId");
        }
    }
}
```

Notre code d'identification

On traite la réponse  
On remercie etc.

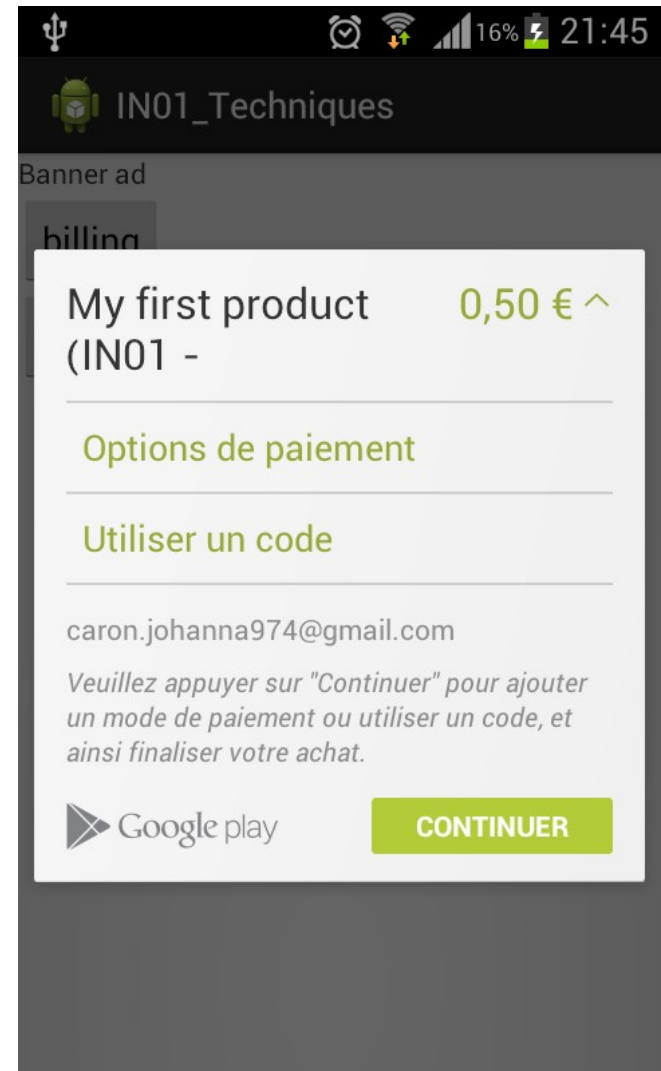
# Tester

- Pour tester, il faut une application signée et publiée en Alpha/Bêta
- Sinon on a ça :
- Il faut aussi tester avec un autre compte que celui du développeur
- Sinon on a ça aussi :



# Tester

- Il faut utiliser un autre compte
- Il est possible de changer de compte sur le même appareil
- Il faut restaurer les paramètres constructeurs



# Sécurité

- La transaction JSON est signée par Google
- Par sécurité, on peut comparer cette clé
- Pour cela il faut la calculer :
  - ➔ Il faut appliquer un algorithme de cryptage RSA 64 bits
  - ➔ Sur la clé de licence de l'application qui se trouve dans la console pour développeur

Produits intégrés à l'application

**Services et API**

### LICENCE ET FACTURATION VIA L'APPLICATION

Les licences vous permettent d'empêcher toute distribution non autorisée de votre application. Elles peuvent également être utilisées pour valider les achats facturés via l'application. [En savoir plus sur les licences](#)

### VOTRE CLÉ DE LICENCE POUR CETTE APPLICATION

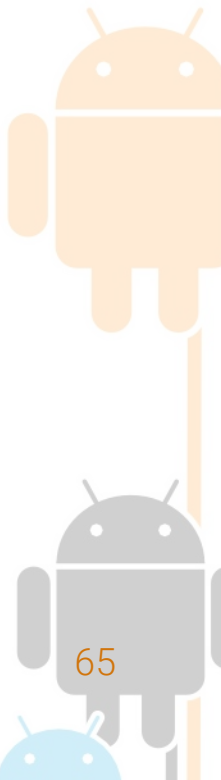
Clé publique RSA codée en Base64, à inclure dans votre fichier binaire. Veuillez supprimer les espaces.

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtG5ckrvtzgY4tjsMOKh8U+a0ZJYDt11nCWLm+dvfCJkcJ2fijdcNRQwgzQ/V1U  
Ytx2oLGZ4ejtmlUaYrotBxkS10NkwV2SYjOYwjyMP60TVAWplvIte6XjRbeXIwxLVNMeiMSz12rXuwK42AxxR4ql86NmLUKCnbmu2SDWUK  
5ahGxHvaLbAgXS4bbP09c1QyMaNCPlxfA2RsQKmN+B+yngaZsar8hEElPk6nwX7AzpaI6iokYyM25/szcZI7N641zCbLO0edeBQLmcANX/  
WwwZ5DsOpu0VJENA5MA00MvvZsoXSPXeavALWfCdcJ6ZpVGUF87s4SiE/kKm0H9MjyPwIDAQAB
```



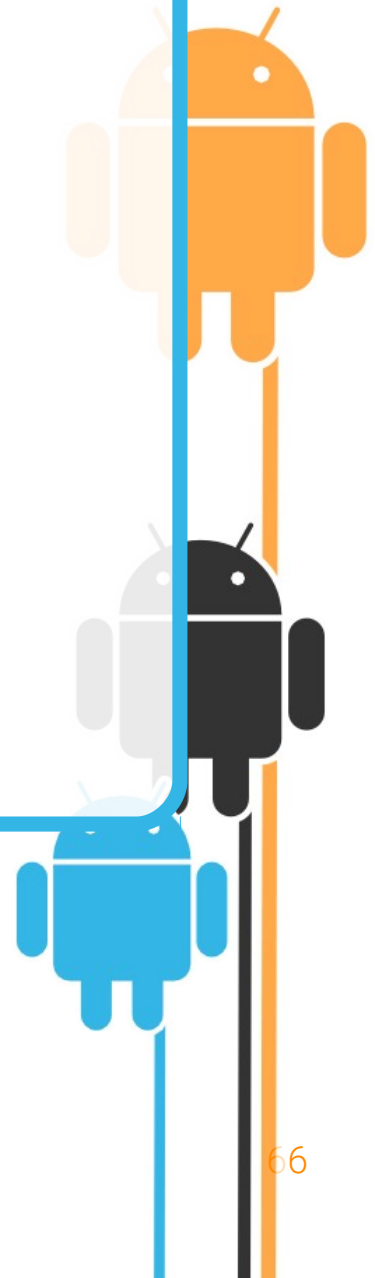
# Conclusion

- Pour plus d'informations consulter :  
[http://developer.android.com/google/play/billing/billing\\_integrate.html](http://developer.android.com/google/play/billing/billing_integrate.html)
- Sur le même principe, il est possible :
  - ➔ D'obtenir une liste des éléments achetés
  - ➔ de retélécharger un élément acheté



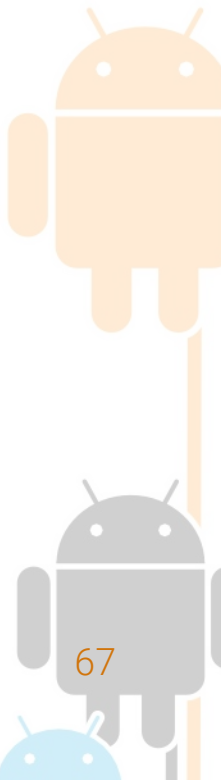
# IN01 – Séance 06

Google Analytics



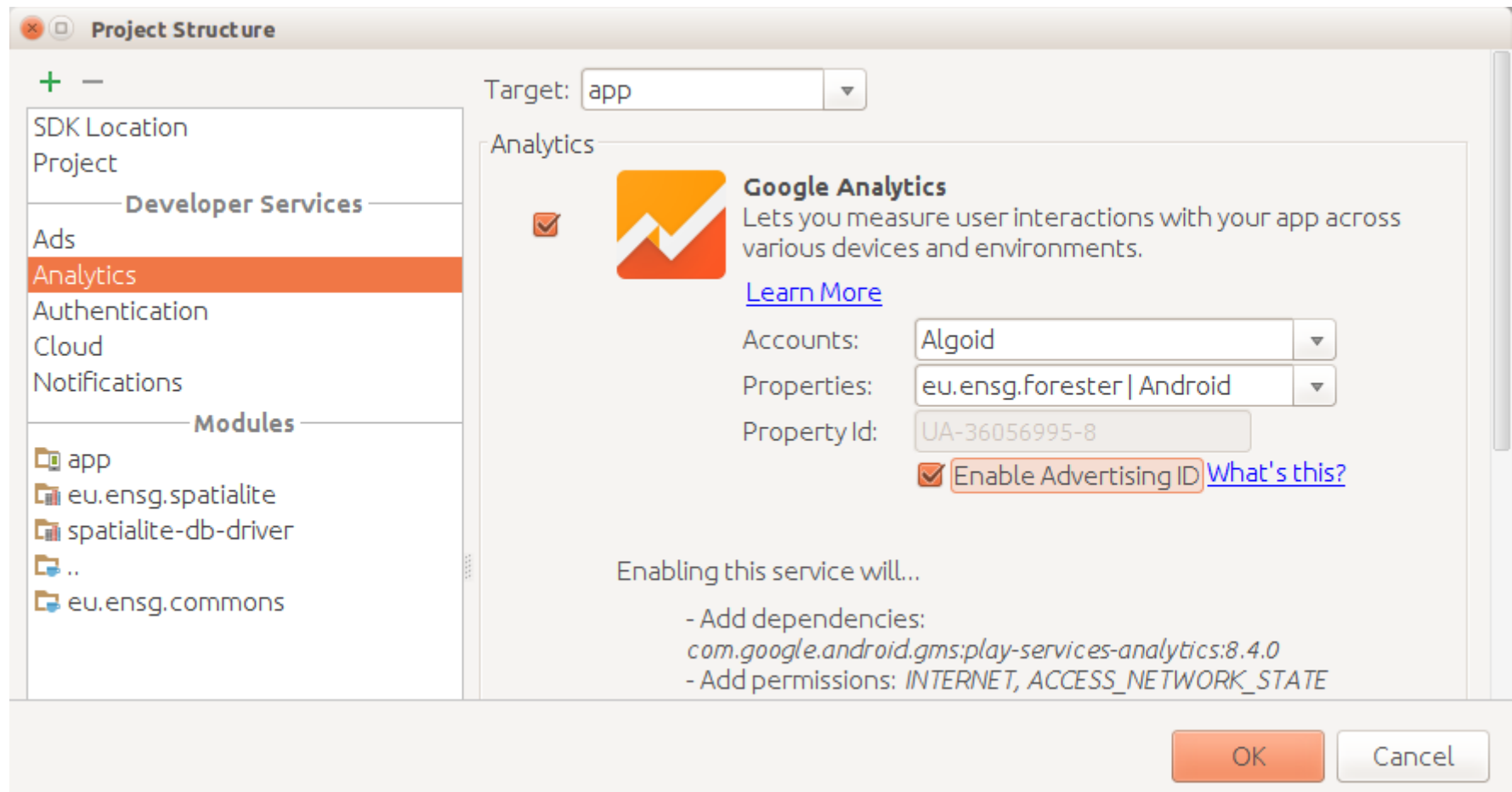
# Vue d'ensemble

- But : obtenir des métriques de l'application
- Répond aux questions sur l'utilisation de l'application : Combien d'utilisateurs ? Combien de nouveaux ? D'où ? Quoi ? Quand ? Comment ? Combien de temps ?
- Et même des sondages, pourquoi pas ?



# Installation

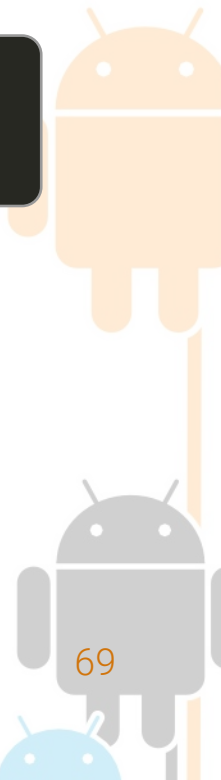
## ■ Dans android Studio : Project Structure



# Autorisations

- Il faut les bonnes autorisations dans le Manifest

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```



# Démarrage/Arrêt

- On doit se servir des évènements onStart et onStop du cycle de vie de l'activity

```
@Override
public void onStart() {
    super.onStart();
    EasyTracker.getInstance(this).activityStart(this);
}

@Override
public void onStop() {
    super.onStop();
    EasyTracker.getInstance(this).activityStop(this);
}
```

# Configuration

- On doit créer un fichier de configuration `res/values/analytics.xml`

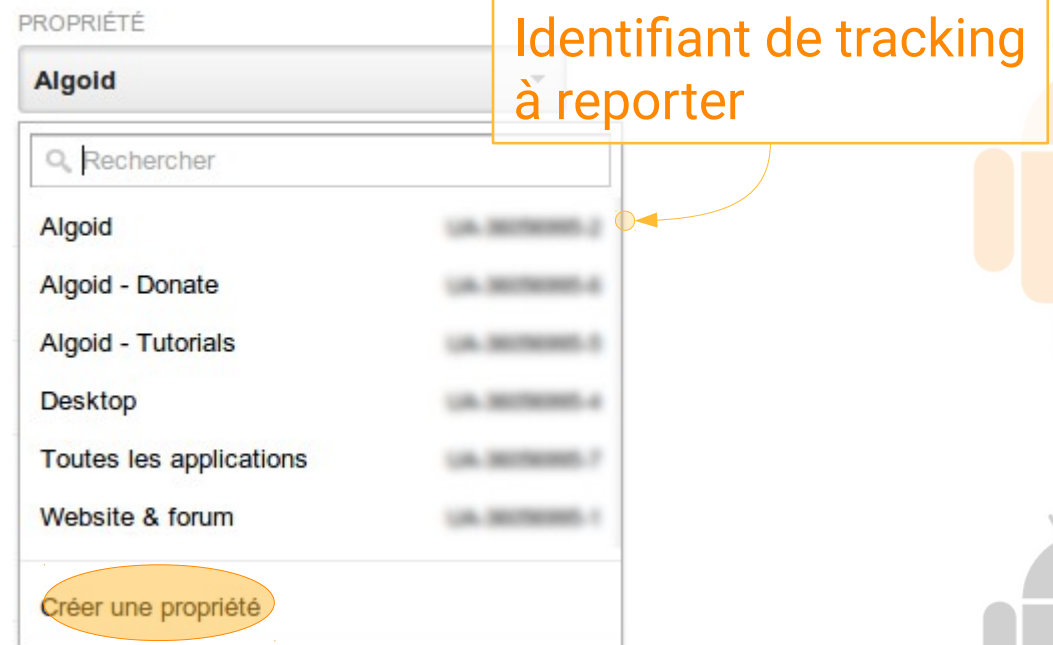
```
<?xml version="1.0" encoding="utf-8" ?>

<resources>
  <!--Replace placeholder ID with your tracking ID-->
  <string name="ga_trackingId">UA-XXXX-Y</string>
  <!--Enable automatic activity tracking-->
  <bool name="ga_autoActivityTracking">true</bool>
  <!--Enable automatic exception tracking-->
  <bool name="ga_reportUncaughtExceptions">true</bool>
</resources>
```

Identifiant de tracking

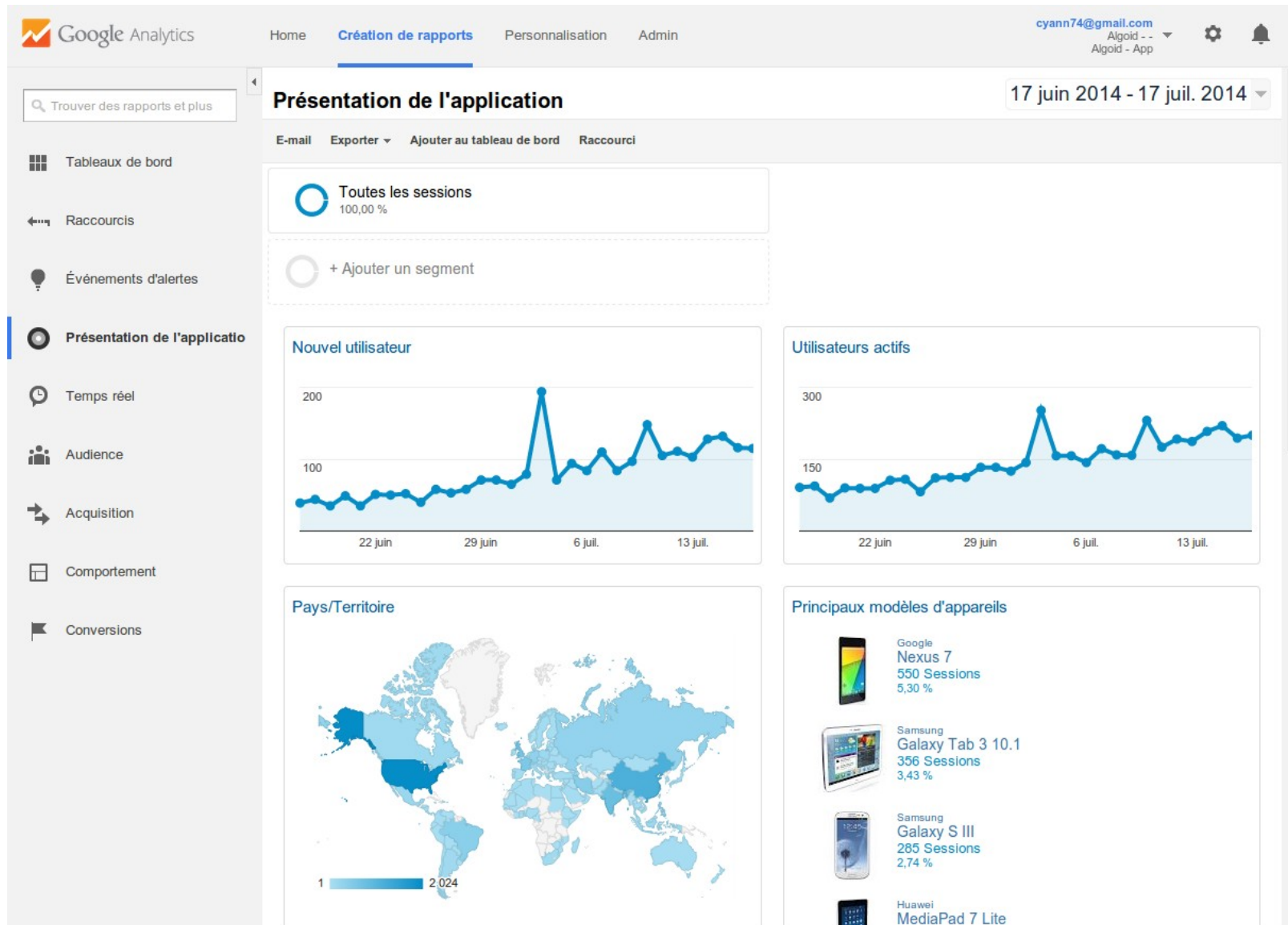
# Analytics – Inscription

- Il faut créer un compte sur Google Analytics : <https://www.google.com/analytics>
- Puis créer une “propriété” dans la section Analytics du site :





# Analytics



# Évènements

- Selon les actions, il est possible d'envoyer des évènements

```
EasyTracker.getInstance(this).send(MapBuilder  
    .createEvent("ui_action",    // Event category (required)  
                "button_press",  // Event action (required)  
                "play_button",   // Event label  
                null)            // Event value  
    .build());
```

# Conclusion

- Google propose, en plus de l'OS Android, du framework ADK des plugins ADT, un grand nombre de services
- La documentation de leur mise en place se trouve ici :  
<http://developer.android.com/google/index.html>
- Entre autres : Games, Location, Google+, Maps, Drive, Cast (clé TV-HDMI), Ads, Wallet, Google Play, In-app billing, Messaging, Cloud Save

# Fin

- Merci de votre attention
- Des questions ?

