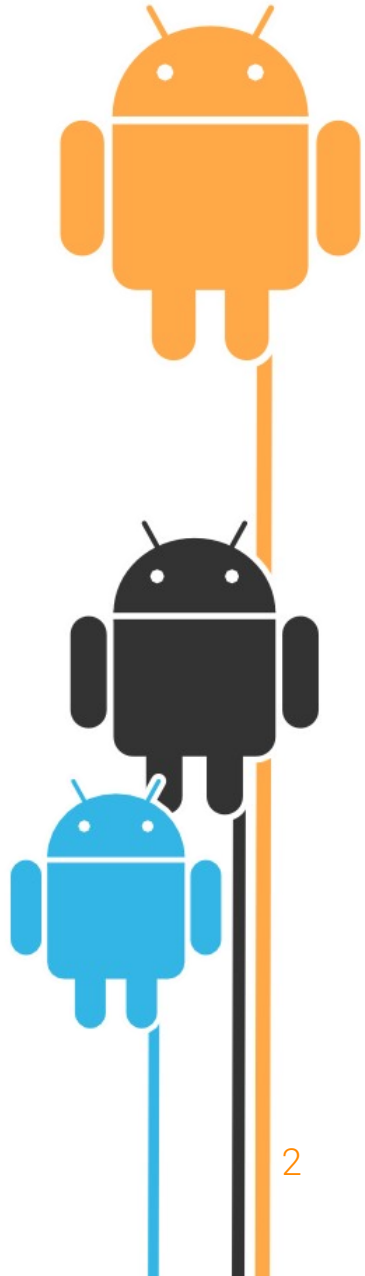
IN01
Programmation Android

06 – Publication

Yann Caron

Sommaire - Séance 06

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

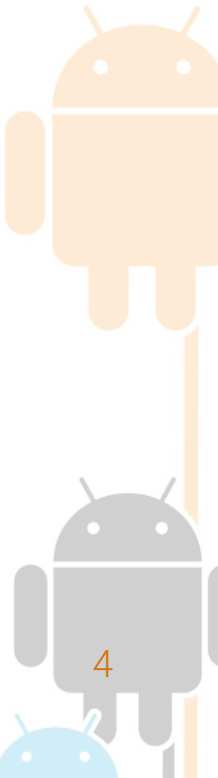


IN01 – Séance 06

Signature

Signer son apk

- Qu'est qu'une signature ?
  - C'est une clé (algorithme d'encryption) 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

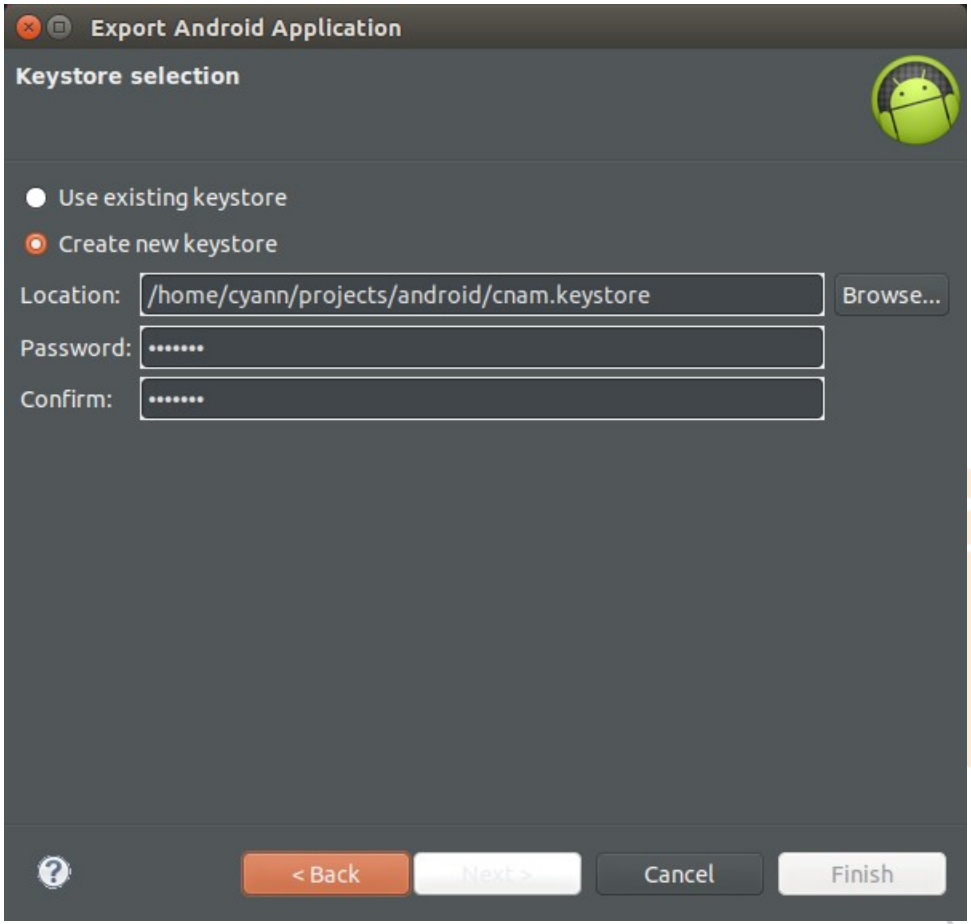


Attention à son keystore

- 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 divulger
- Il faut les retenir
  - Comme pour la carte bancaire, il ne faut pas mettre le code avec

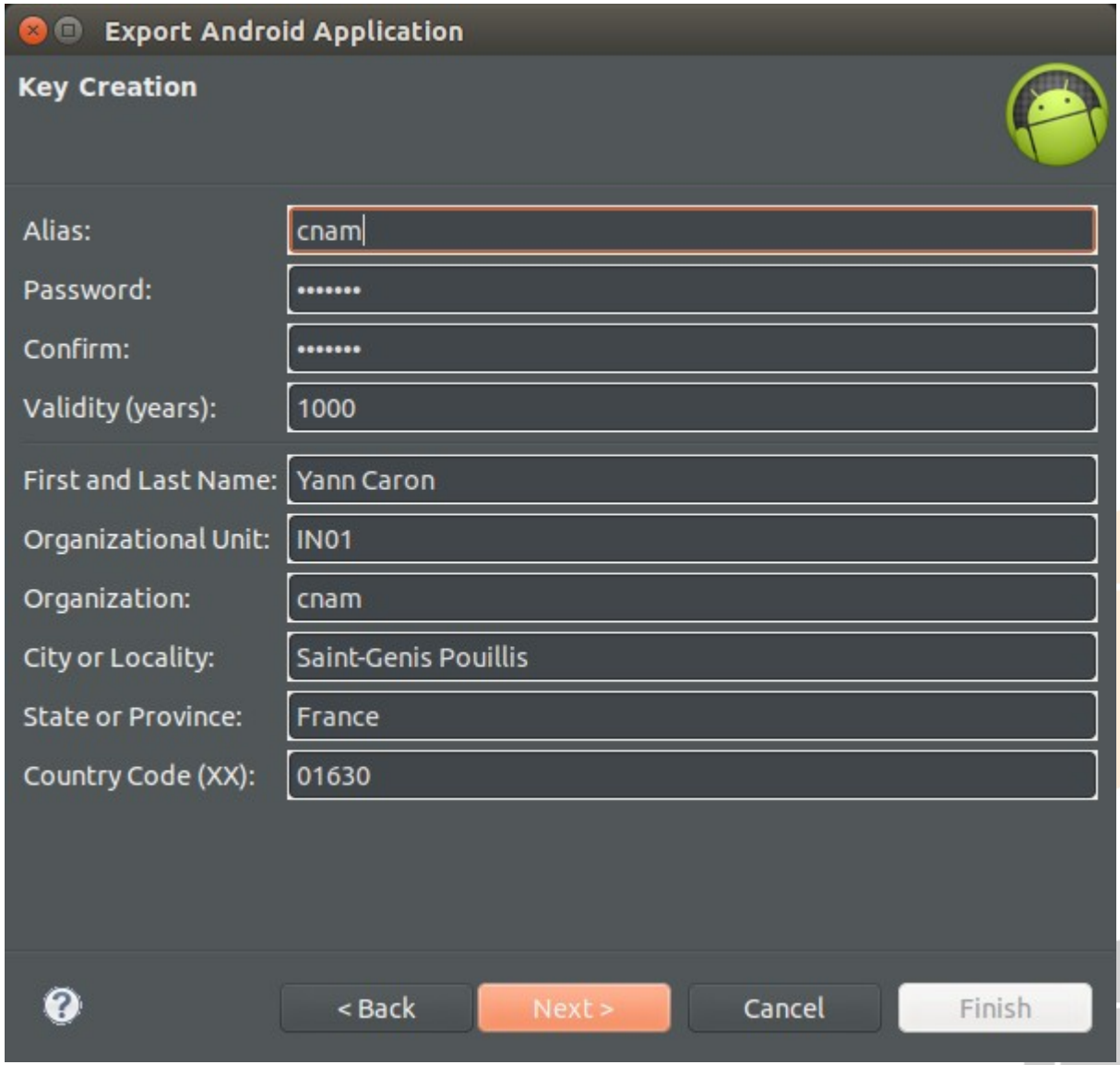
Créer une clé

- Avec Eclipse, rien de plus facile ::
  - clique droit sur le projet / Android tools / Export signed application package



Créer une clé

- Le password, c'est “android”

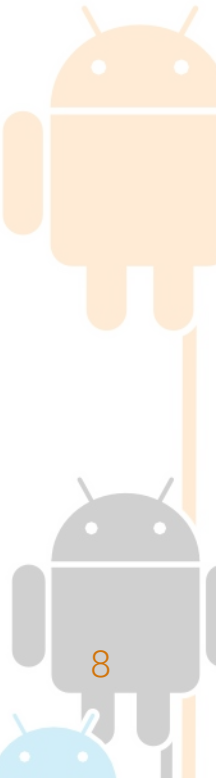


En ligne de commande

- Grâce à l'outil keytool (JAVA_HOME/bin)

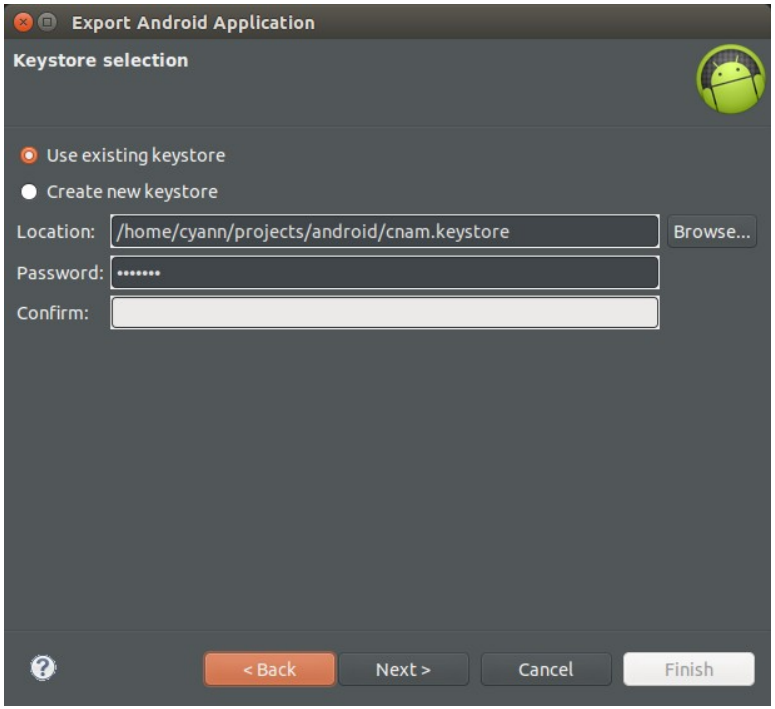
```
keytool -genkeypair -v -keystore myKeystore.keystore -storepass
myKeystorePassword -alias myKey -keypass myKeyPassWord -keyalg RSA
-validity 36500
```

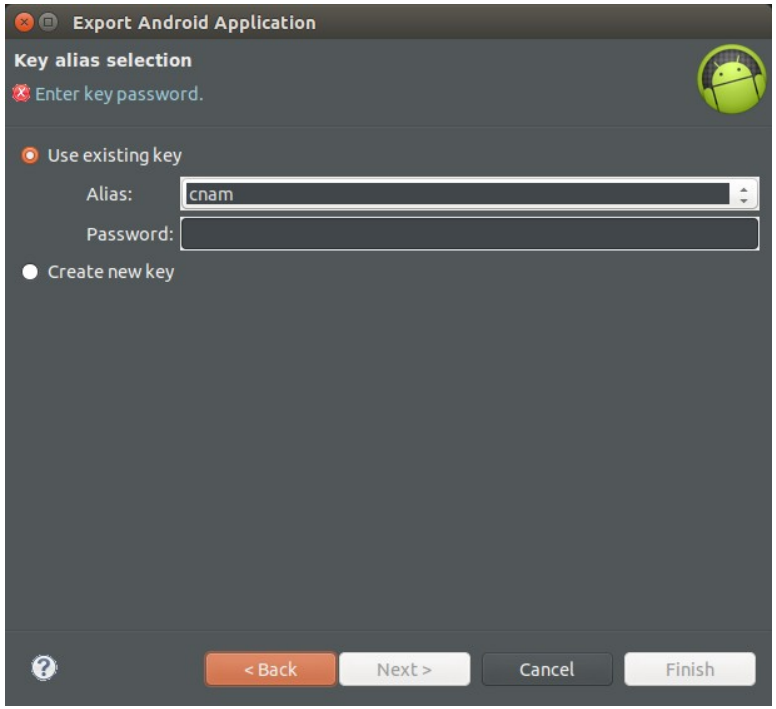
- -keystore :: le nom et le chemin de la clé
- -storepass :: le mot de passe du keystore
- -alias :: le nom de la clé
- -keypass :: le password de la clé



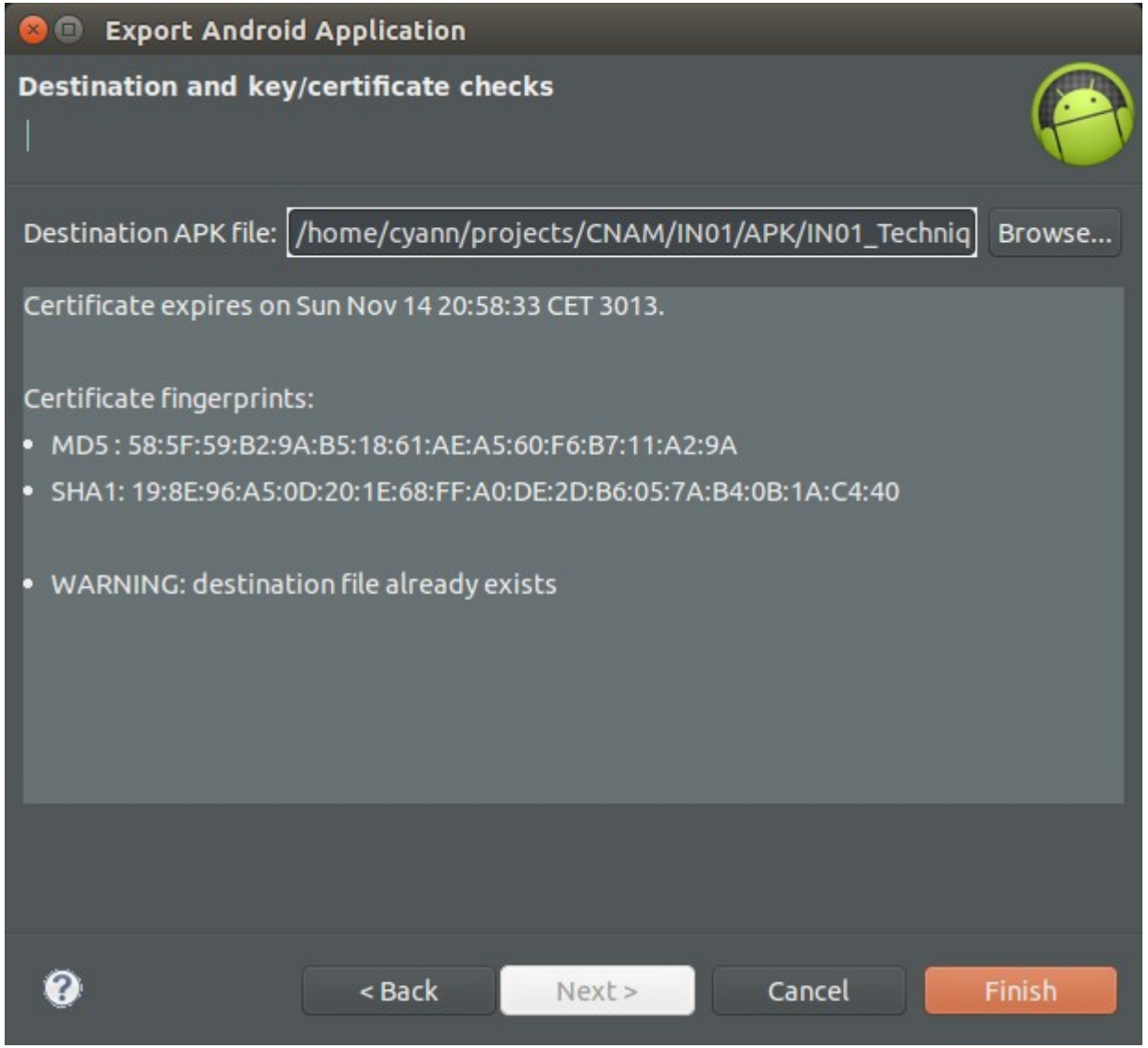
Signer l'application

- Rien de plus simple, eclipse se charge de tout
  - clique droit sur le projet / Android tools / Export signed application package





Signer l'application

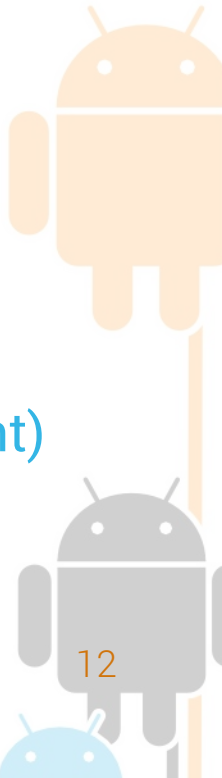


IN01 – Séance 06

Publication

Création du compte developpeur

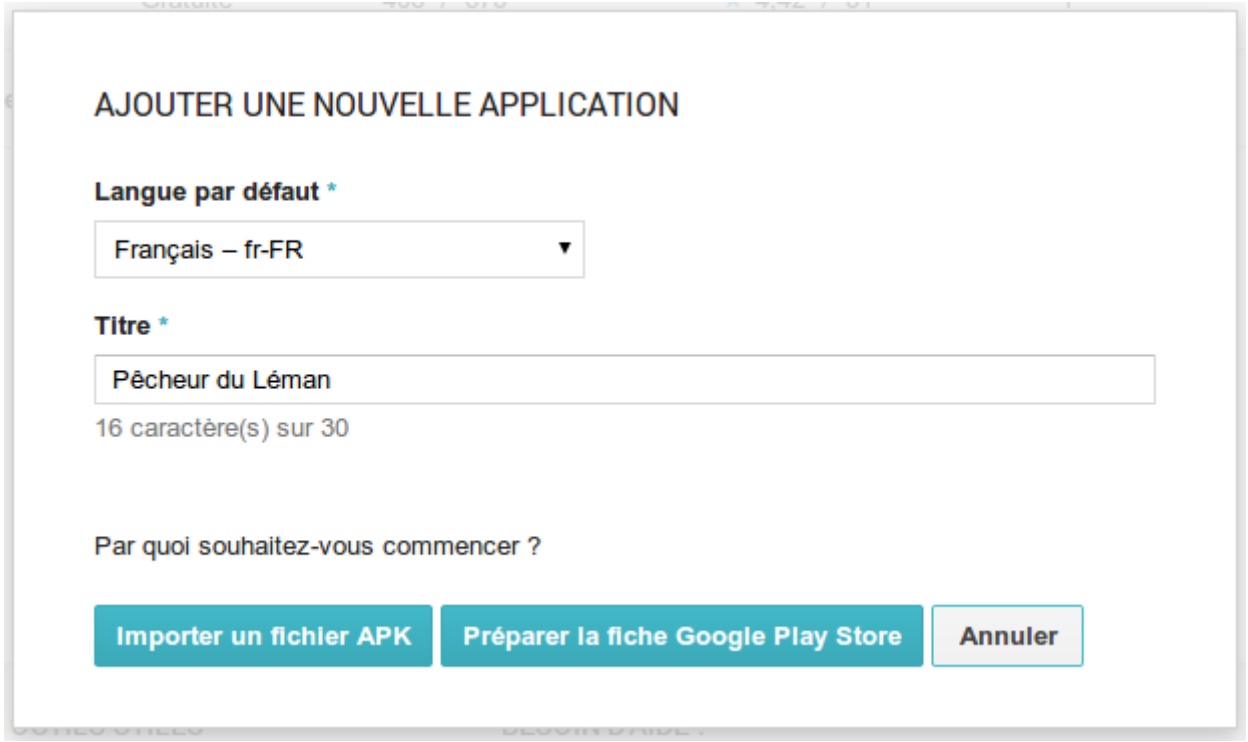
- Une fois l'application signée, elle est prête à être publiée sur un market
- Il existe plusieurs market android ::
  - Officiel :: le play store, frais d'inscription 20€ (https://play.google.com/apps/publish)
  - Alternatifs ::
    - Slide me (http://slideme.org/)
    - GetJar (http://www.getjar.mobi/)
    - 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



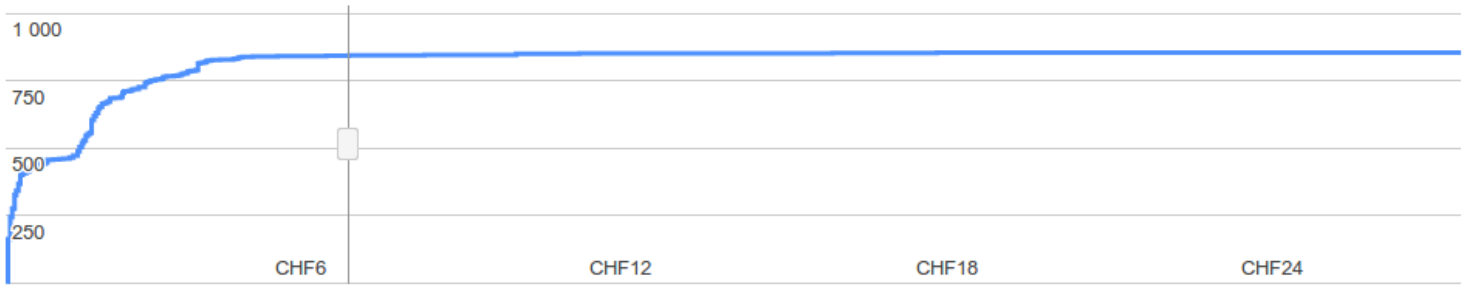
Publication – titre & description

- 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

Groupe d'annonces | **Mot clé** | Appareil | Lieu : Pays ▼

Clics par jour	Impressions par jour	Coût par jour	Position de l'annonce
Entre 759 et 928	Entre 14,4 K et 17,6 K	Entre 1,13 K CHF et 1,38 K CHF	Entre 1,07 et 1,31



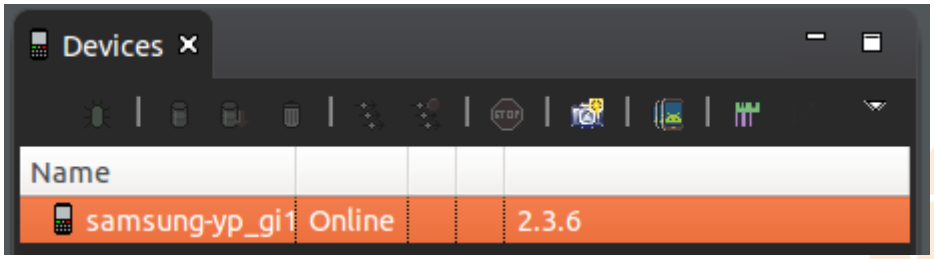
✚ Ajouter des mots clés | Modifier les types de correspondance | Supprimer | Télécharger

Mot clé	Groupe d'annonces	Clics	Impr.	Coût	CTR	CPC moy.	Pos. moy.
android	android	356,70	8 814,46	371,97 …	4,0 %	1,04 CHF	1,27
education	android	391,13	6 264,65	722,54 …	6,2 %	1,85 CHF	1,08
langage	android	13,57	32,55	21,55 CHF	41,7 %	1,59 CHF	1,02
programmation	android	82,42	888,34	139,14 …	9,3 %	1,69 CHF	1,05
Totaux		**843,82**	**16 000,00**	**1 255,2…**	**5,3 %**	**1,49 CHF**	**1,18**

Mots clés **1** à **4**, sur **4**

Images

- Il faut ajouter un maximum de captures d'écran les plus attractives possibles
- Eclipse possède un outil, screen capture dans la perspective DDMS

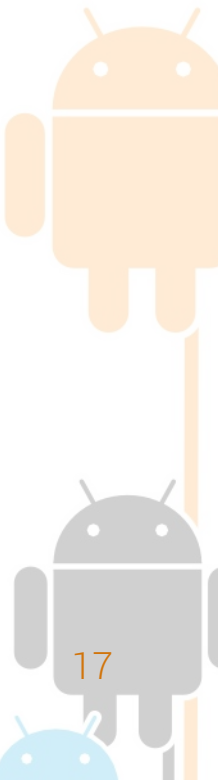


- Ne pas hésiter à

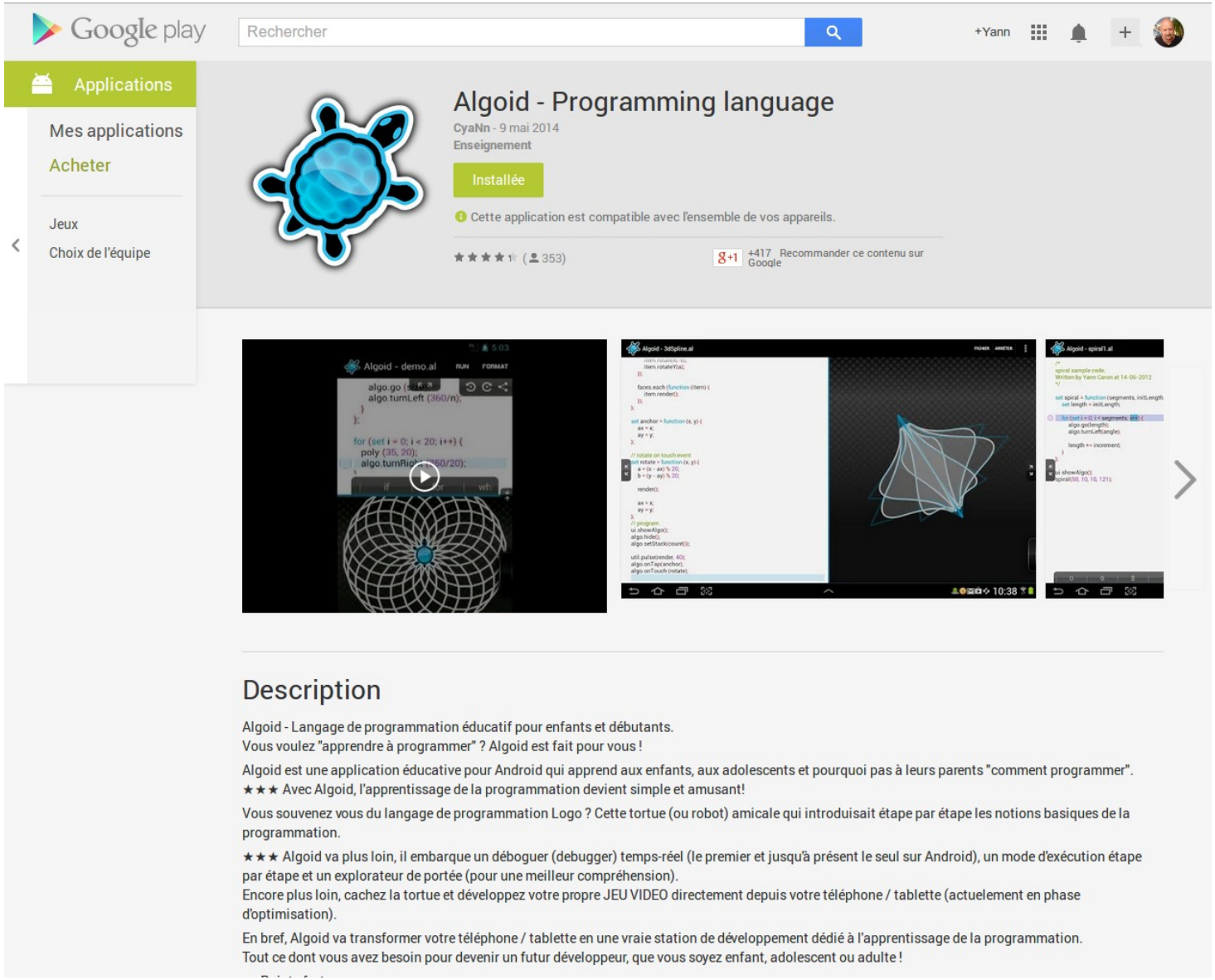
  capturer plusieurs résolutions (téléphone, tablette 7 pouces, 10 pouces.... merci les émulateurs)

Video

- La vidéo de présentation est très importante, c'est souvent elle qui déclence 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'apparail, 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

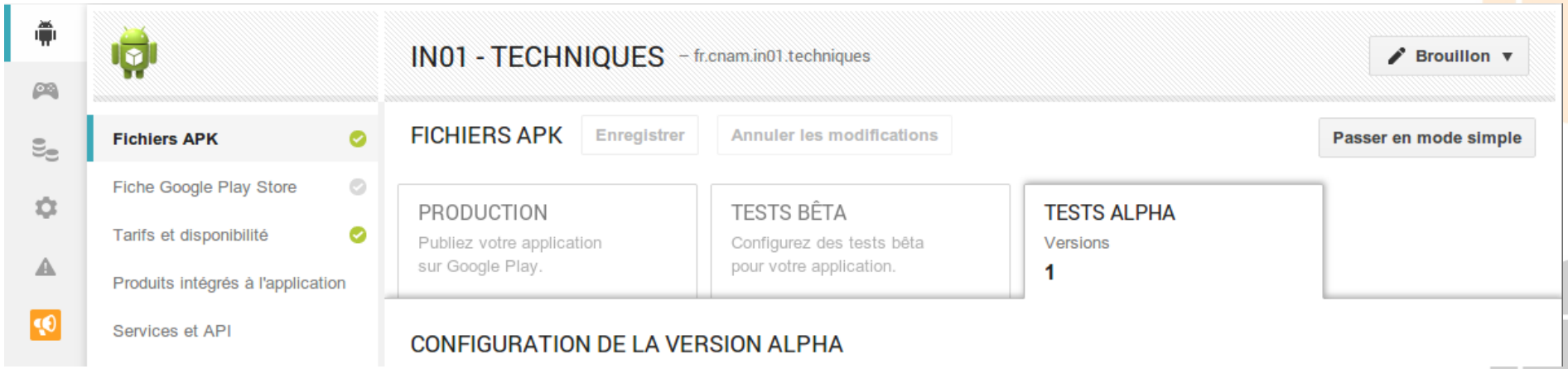

Google play
Rechercher
+Yann
Applications
Mes applications
Acheter
Jeux
Choix de l'équipe
Algoid - Programming language
CyaNn - 9 mai 2014
Enseignement
Installée
Cette application est compatible avec l'ensemble de vos appareils.
(353)
+417 Recommander ce contenu sur Google
Description
Algoid - Langage 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 déboguer (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 meilleur compréhension).
Encore plus loin, cachez la tortue et développez votre propre JEU VIDEO directement depuis votre téléphone / tablette (actuelement 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

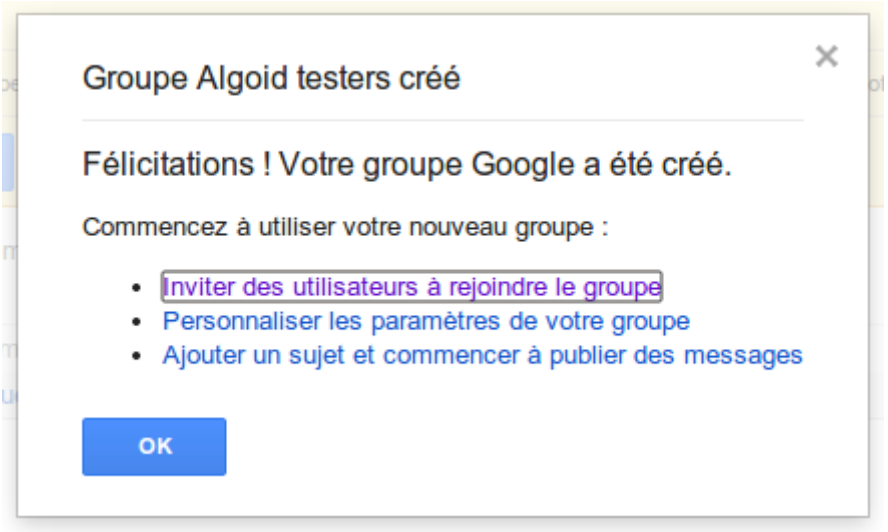


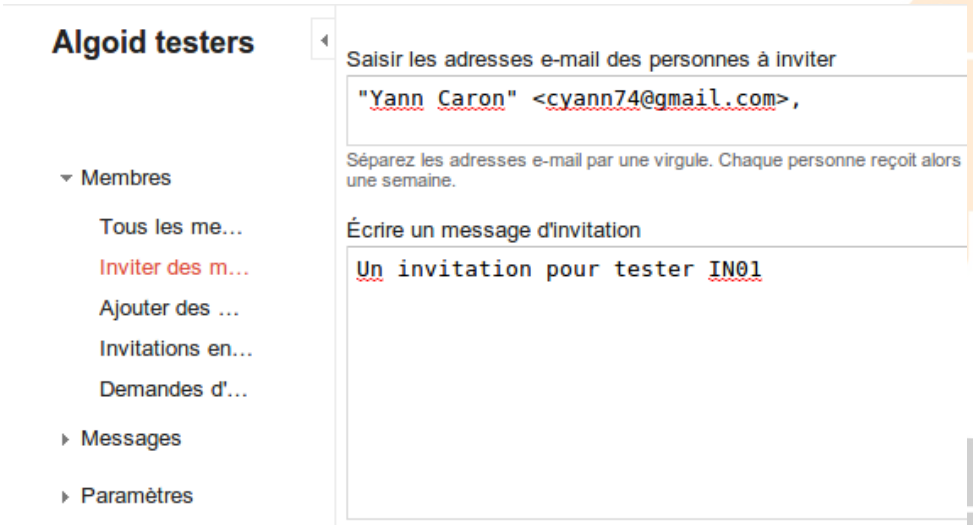
Gérer les testeurs

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

Envoyer une invitation

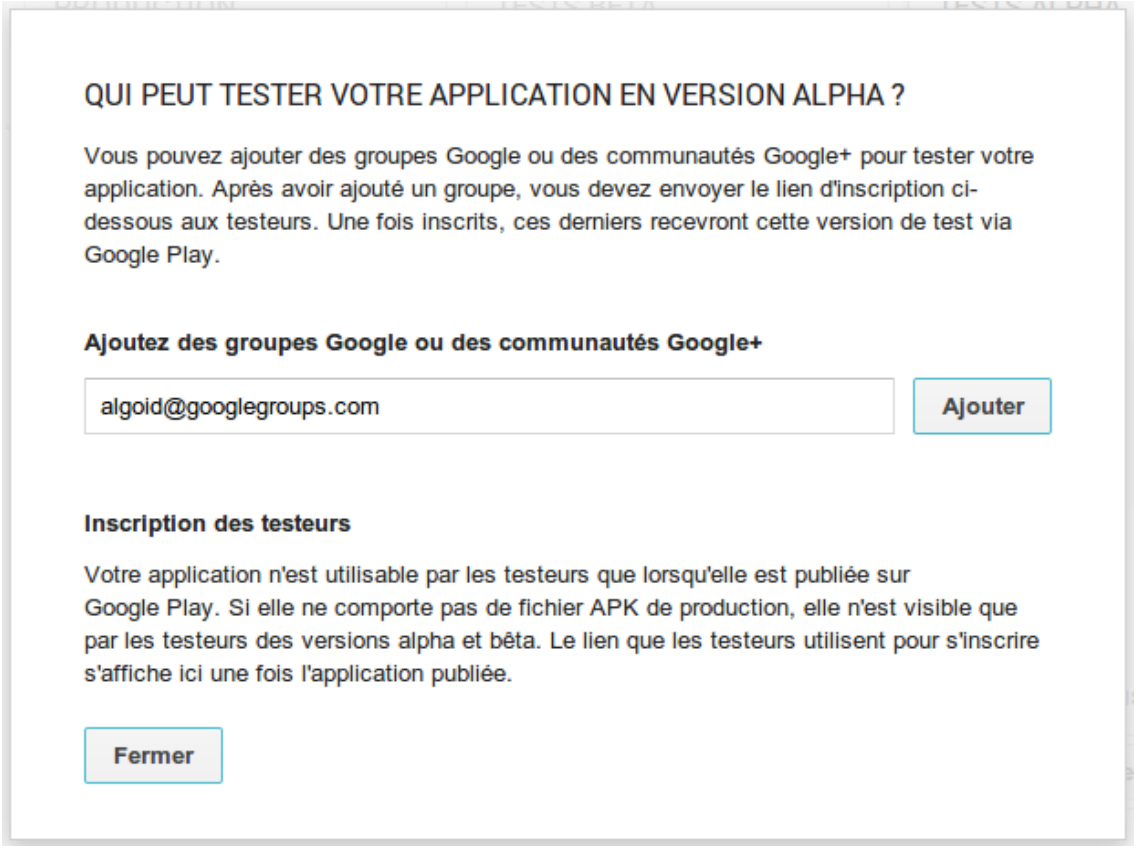
- Il faut ensuite envoyer une invitation aux testeurs





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



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 des testeurs)
- Le lien apparait dans le fenêtre de gestion des testeurs

Envoyer aux testeurs

- Après envoie 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

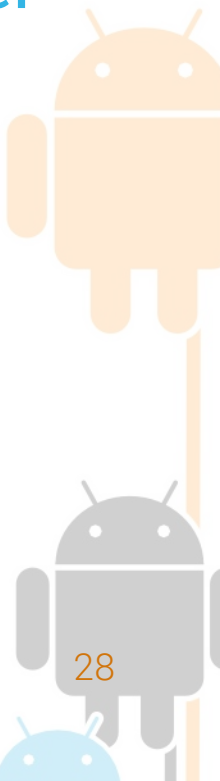
Monétisation

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'essaie

- 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 insciter l'achat



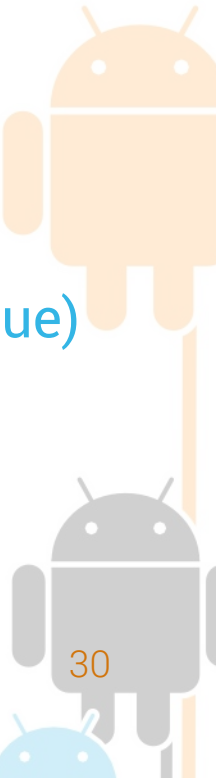
Monétisation - donate

- 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 sources

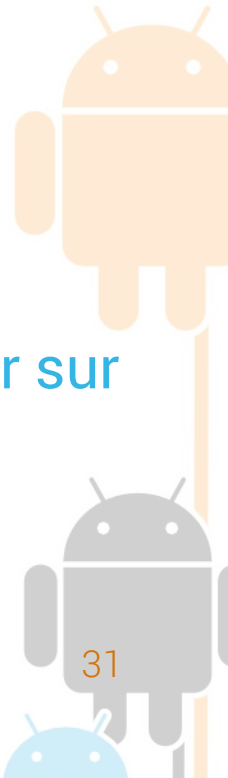
Monétisation - Freemium

- En générale, 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 materiel 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



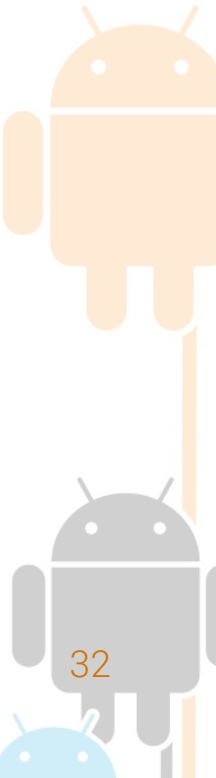
Monétisation - publicité

- 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 intersticiels
    - 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)



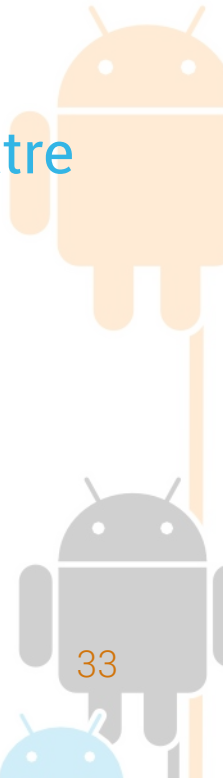
Monétisation - Autres

- 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 defendable, il faut une extension materiel, quelque chose qui sera palpable pour l'investisseur
- Abonnements utilisateurs



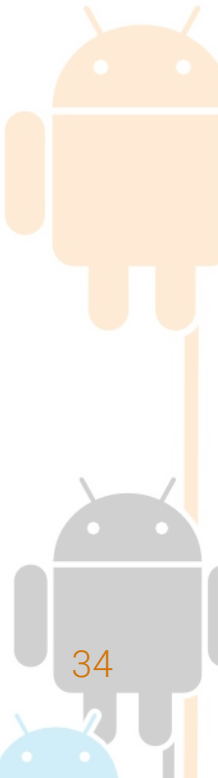
Monétisation - conclusion

- 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 intersticielles dans une version d'essai
- Attention au message véhiculé. Il est par exemple difficile de mettre de la publicité dans un 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

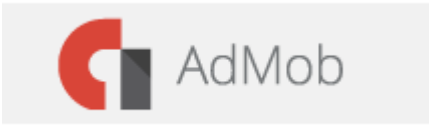
- L'app vise-t-elle un marché de niche ?
  - oui :: application payante > 1€
- Quelle valeur lui accorderait le public ?
  - < 0.79 € :: publicité semble envisageable
- Sur quelle plateforme ?
  - iOS :: les utilisateurs préfèrent payer et être tranquile
  - Android :: les utilisateurs préfèrent la publicité
  - WP :: Attention, pas d'in-app purchase possible
- L'app est elle modulaire ?
  - L'in-app purchase semble envisageable
- Y a-t-il des coût à long terme
  - La publicité peut-être une solution pour amortir les coûts
  - L'abonnement semble également envisageable



IN01 – Séance 06

AdMob

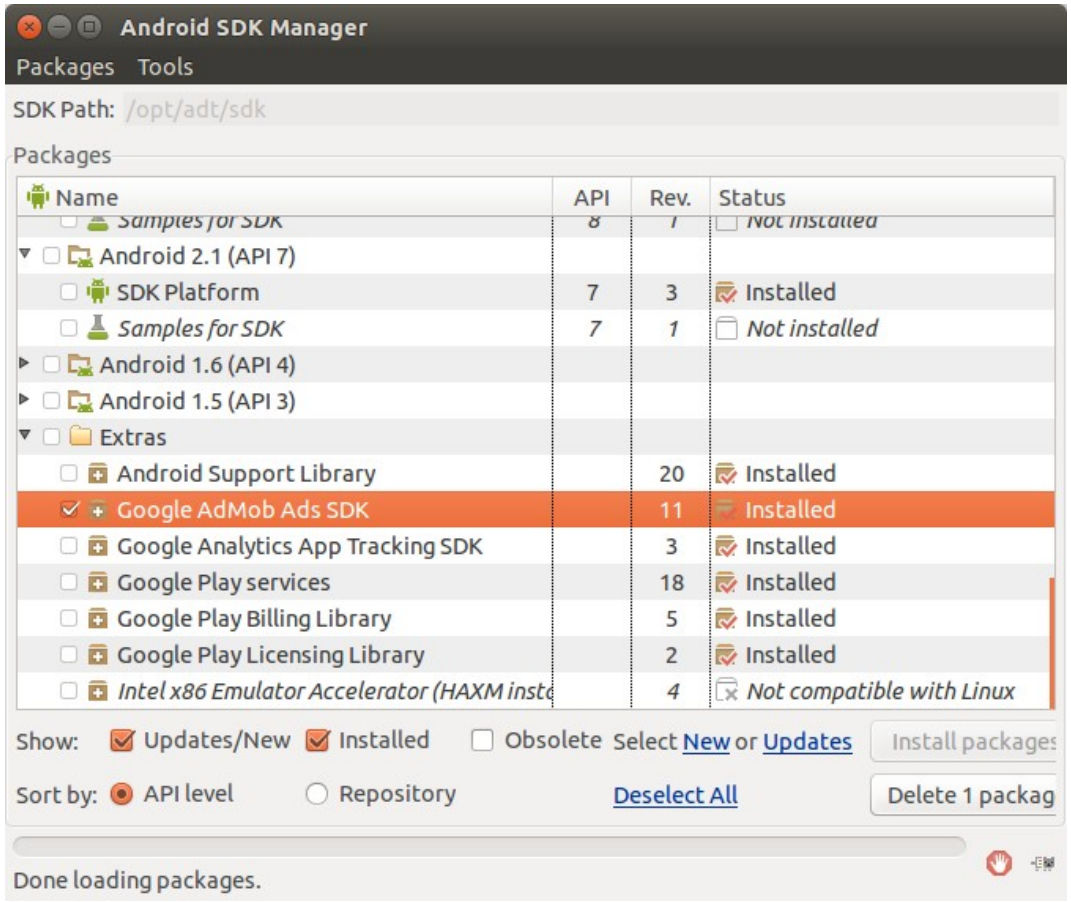
AdMob



- 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 connaitre
- 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

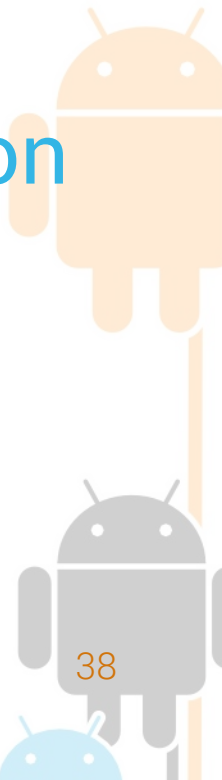
Installation

- Disponnible dans l'android SDK manager

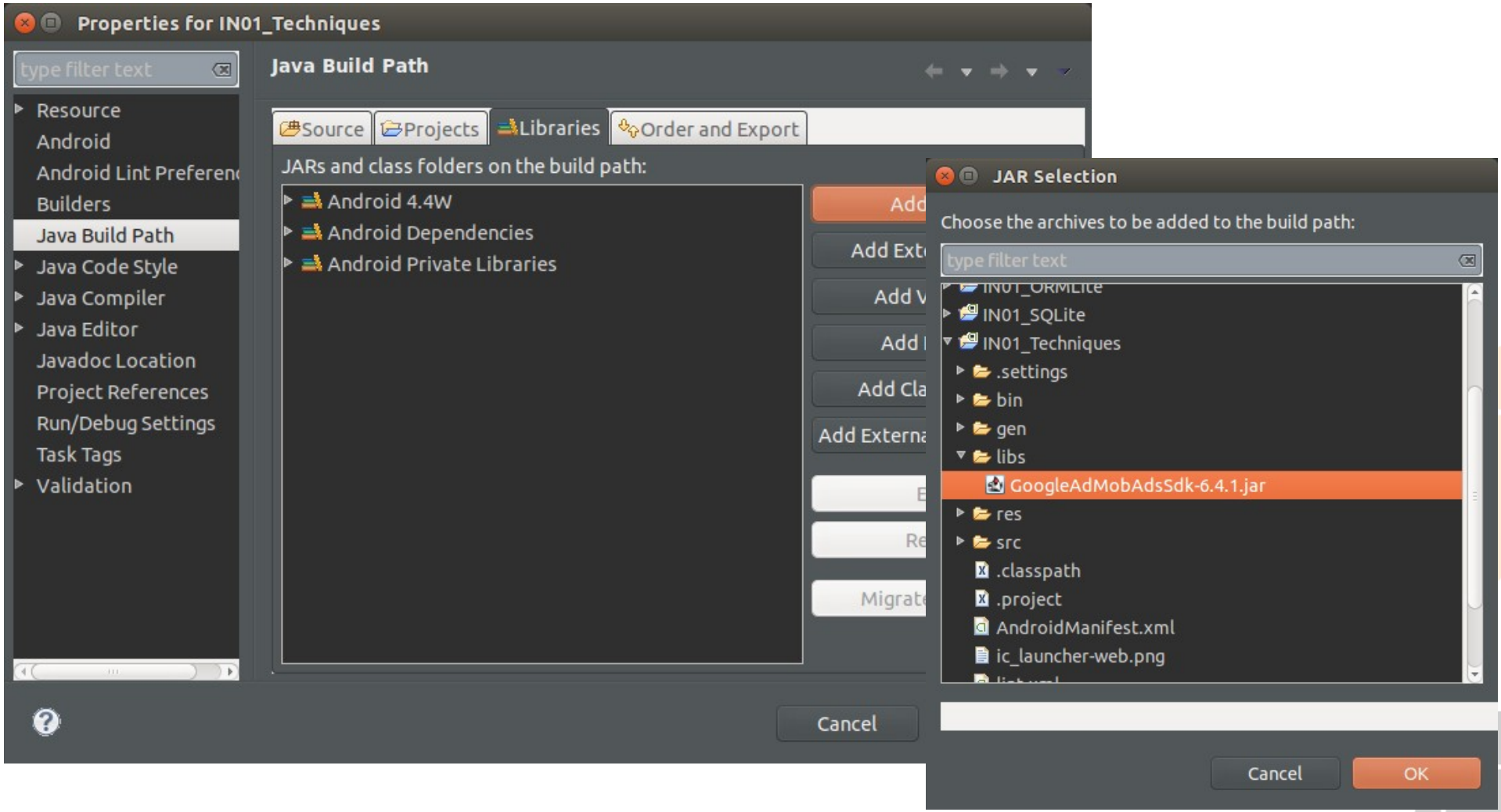


Installation

- Copier le jar GoogleAdMobAdsSdk-x.x.x.jar dans le repertoire libs du projet
- Le jar se situe dans l'adk :: [sdk]/extras/google/admob_ads_sdk/
- Ajouter la librairie dans la path de l'application



Installation



Manifest

- Il faut ajouter la permission et l'AdActivity

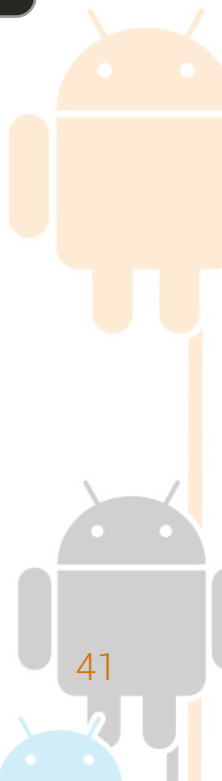
```
<manifest>
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission
    android:name="android.permission.ACCESS_NETWORK_STATE" />
  ...
  <application>
    <activity
      android:name="com.google.ads.AdActivity"
      android:configChanges="keyboard|keyboardHidden|
        orientation|screenLayout|uiMode|
        screenSize|smallestScreenSize" />
  </application>
</manifest>
```

Ajouter la bannière

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

```
<com.google.ads.AdView
        xmlns:googleads="http://schemas.android.com/apk/lib/com.google.ads"
        android:id="@+id/ad"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        googleads:adSize="BANNER"
        googleads:adUnitId="@string/admob_id" />
```

- Plusieurs adSize :: BANNER, FULL_BANNER, LARGE_BANNER....
  - A retenir SMART_BANNER :: taille automatique
  - WIDE_SKYSCRAPER :: vertical



Charger la bannière

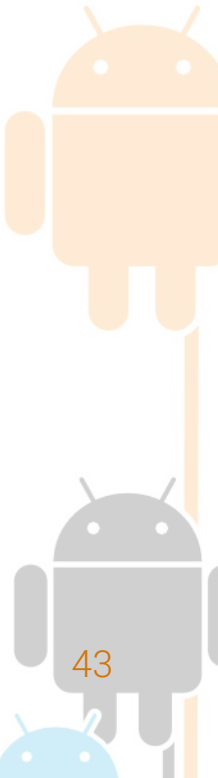
- Dans la méthode onCreate de l'activity ou du fragment il faut initialiser une requête

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

  AdView adView = (AdView) findViewById(R.id.ad);

  AdRequest adRequest = new AdRequest();
  adRequest.addKeyword("sporting goods, fishing");
  adView.loadAd(adRequest);
}
```

Callbacks

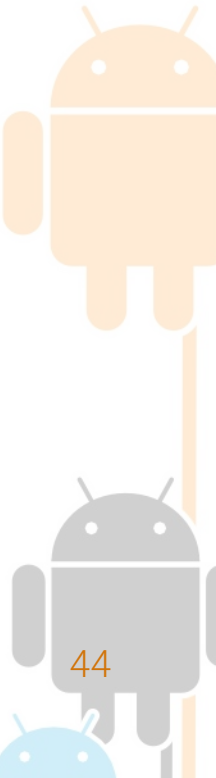
- Récupérer des évènements et effectuer des traitements

```
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) {}
});
```

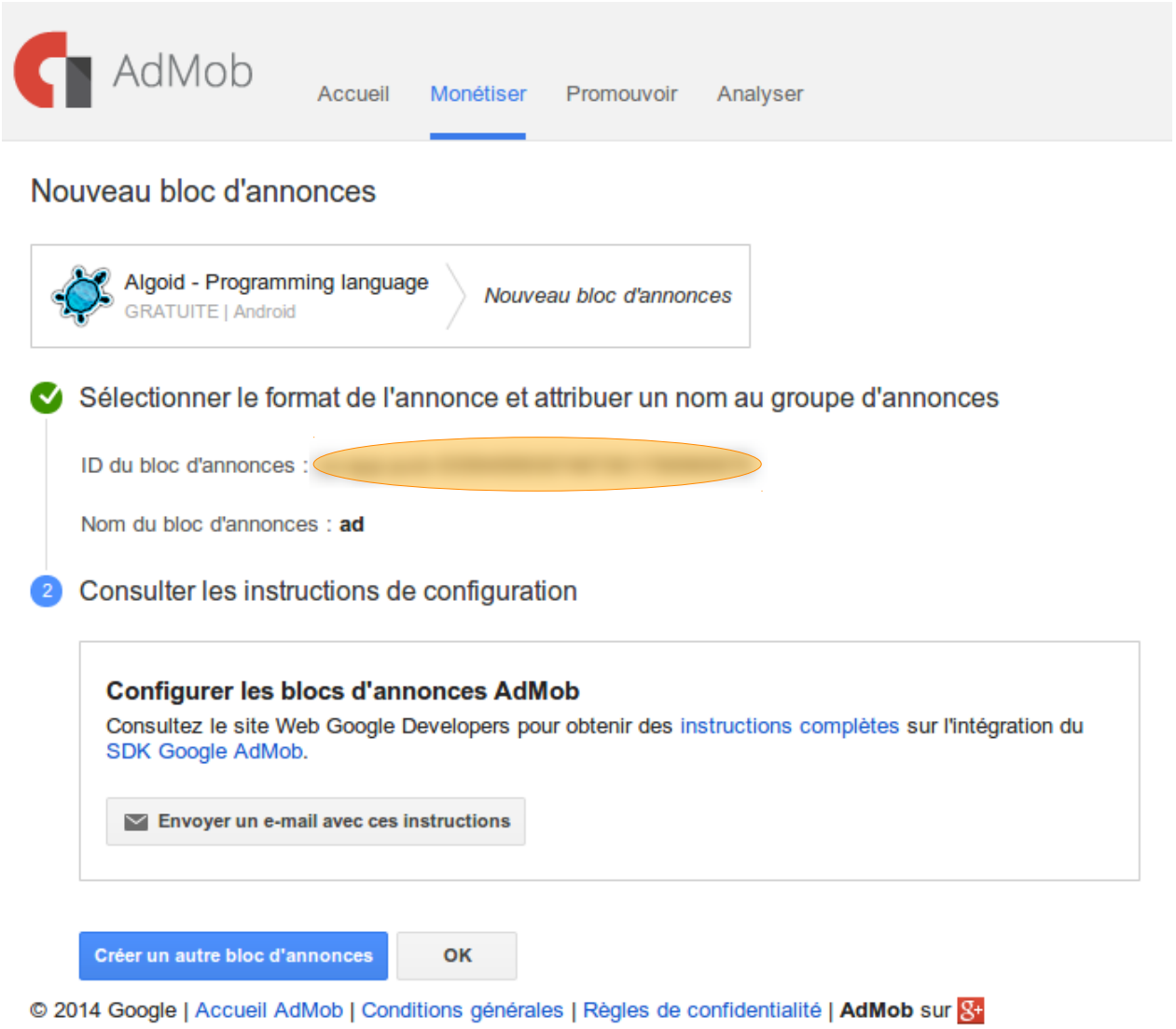


Obtenir un code

- Un fois enregistré sur le site admob :: https://apps.admob.com
- Il faut enregistrer l'application pour obtenir un code, mais pour cela elle doit d'abord être publiée sur le play store
- Ensuite il faut créer un bloc d'annonce
- Et récupérer le code du bloc d'annonce



Obtenir un code



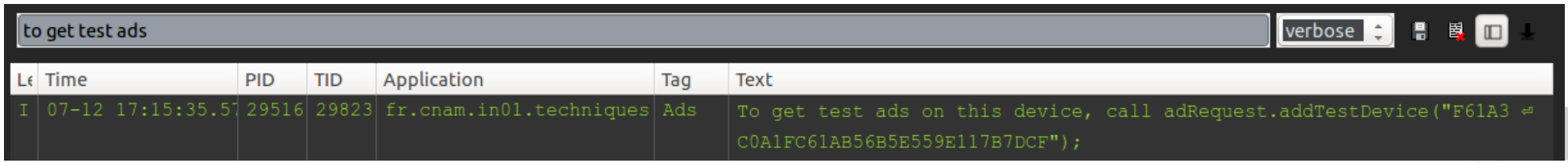
adUnitId

- Il faut renseigner la propriété de l'adView avec le code obtenue
- Celui ci fait référence à une string adMod_id (le nom est arbitraire)

```
<string name="admob_id">ca-app-pub-5359459930746730/3377299077</string>
```

TestDevice

- Pour éviter que la publicité n'apparaisse lorsque l'on debug l'application, il existe le mode de test
- Celui-ci fonctionne par appareil
- La clé est indiquée dans le logcap 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**********************");
```

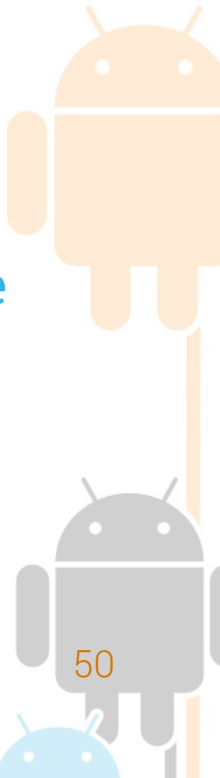
- Et voilà le travail ==>

IN01 – Séance 06

Intégrer In-app billing Version 3

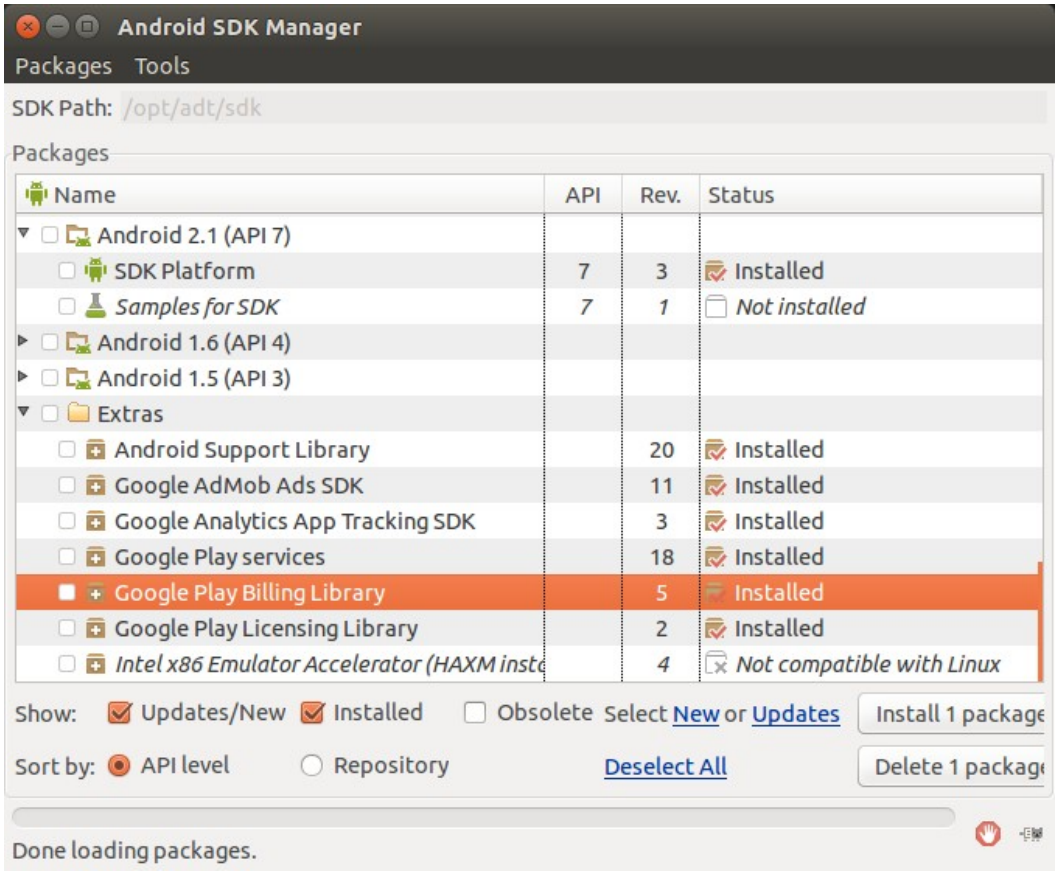
Vue d'ensemble

- Plusieurs étapes nécessaires à la facturation ::
  - Installer la bibliothèque (fichier AIDL) à 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 requête de facturation depuis l'application vers l'interface `IinAppBillingService` et récupérer la réponse du service de facturation de Google Play



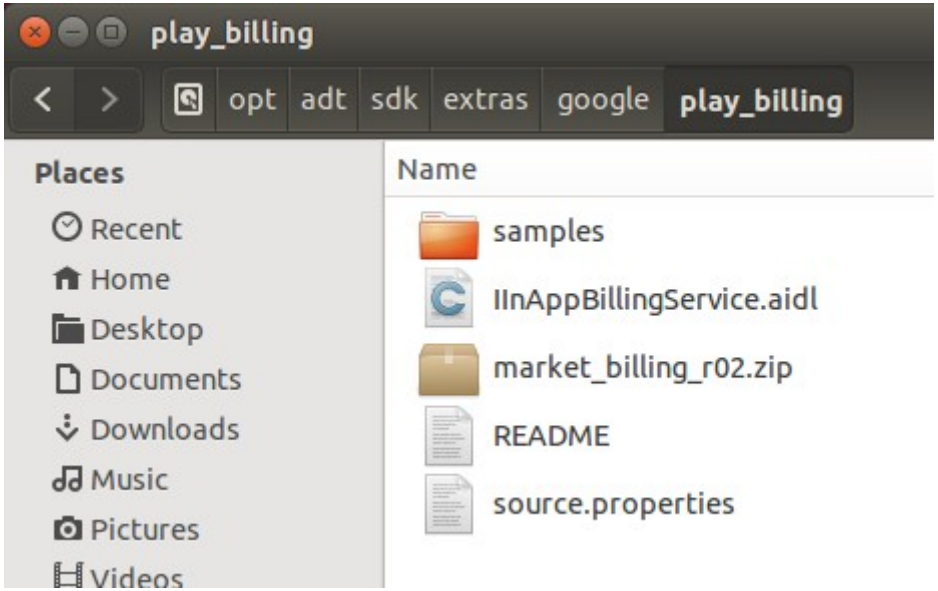
1 - Installation

- Télécharger la bibliothèque depuis l'Android SDK Manager



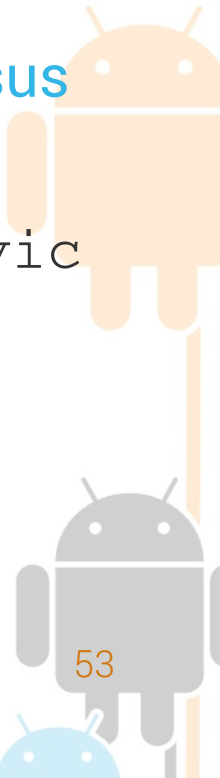
1- Installation

- 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 inter processus
- Le fichier AIDL indique au compilateur de générer le fichier `gen/com.android.vending.billing/IinAppBillingService.java` qu'on pourra utiliser par la suite

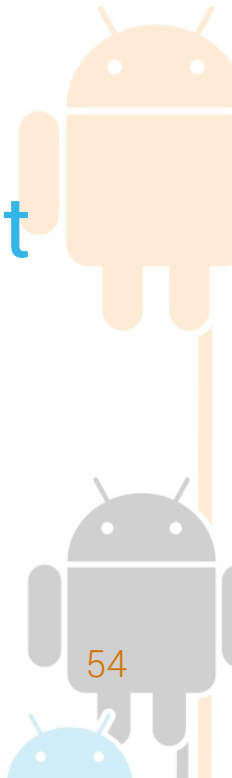


2 - Manifest

- 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;

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);
   }
};
```

Variable d'instance

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);
  }
}
```

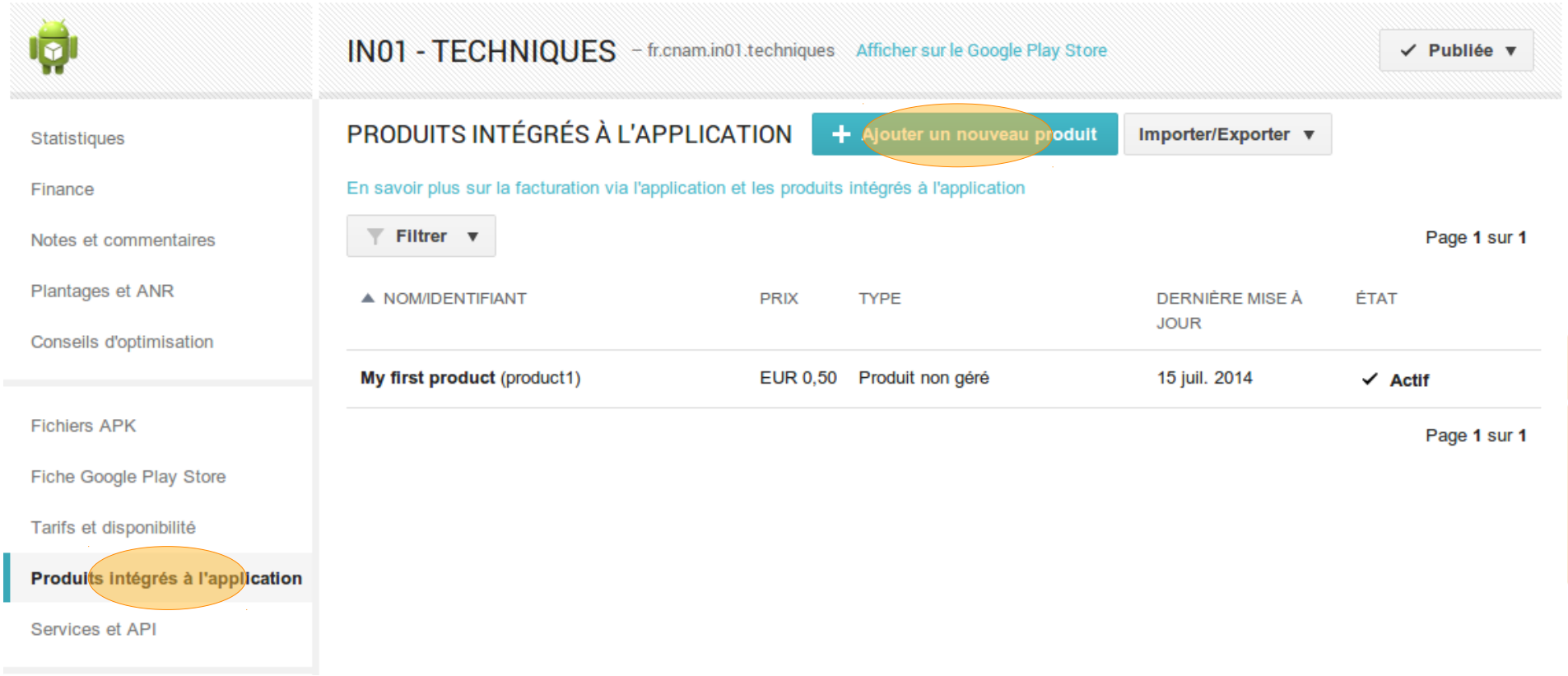
onCreate

onDestroy

4 – Créer des produits intégrés

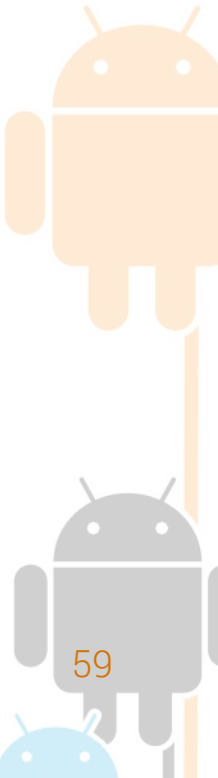
- Maintenant que notre service est lié il faut créer des éléments à vendre.... pour tester....
- De retour dans la console pour développeur de Google play
- Dans notre application en mode test Alpha ou Bêta
- Attention !! L'application doit être publiée
  - Mais s'il n'existe pas d'APK de production,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

4 – Créer des produits intégrés



4 – Créer des produits intégrés

- Il existe trois types de produit ::
  - Produit géré :: achat possible qu'une seule fois. Une fois achetée, 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

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

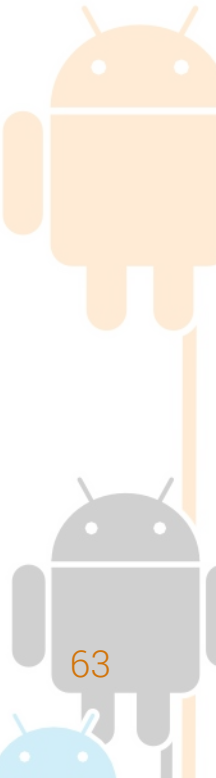
Traite la liste

C'est du JSON

Codes réponses

- La valeur RESPONSE_CODE peut signifier plusieurs choses ::

Response Code	Value	Description
BILLING_RESPONSE_RESULT_OK	0	Success
BILLING_RESPONSE_RESULT_USER_CANCELED	1	User pressed back or canceled a dialog
BILLING_RESPONSE_RESULT_BILLING_UNAVAILABLE	3	Billing API version is not supported for the type requested
BILLING_RESPONSE_RESULT_ITEM_UNAVAILABLE	4	Requested product is not available for purchase
BILLING_RESPONSE_RESULT_DEVELOPER_ERROR	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
BILLING_RESPONSE_RESULT_ERROR	6	Fatal error during the API action
BILLING_RESPONSE_RESULT_ITEM_ALREADY_OWNED	7	Failure to purchase since item is already owned
BILLING_RESPONSE_RESULT_ITEM_NOT_OWNED	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", "developper 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));
}
```

Util pour sécurité

On crée la requête

Un code d'identification

Réponse dans l'évènement

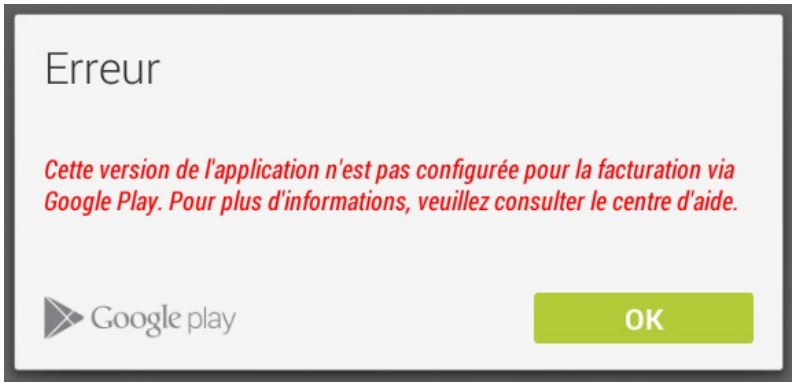
```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {


  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

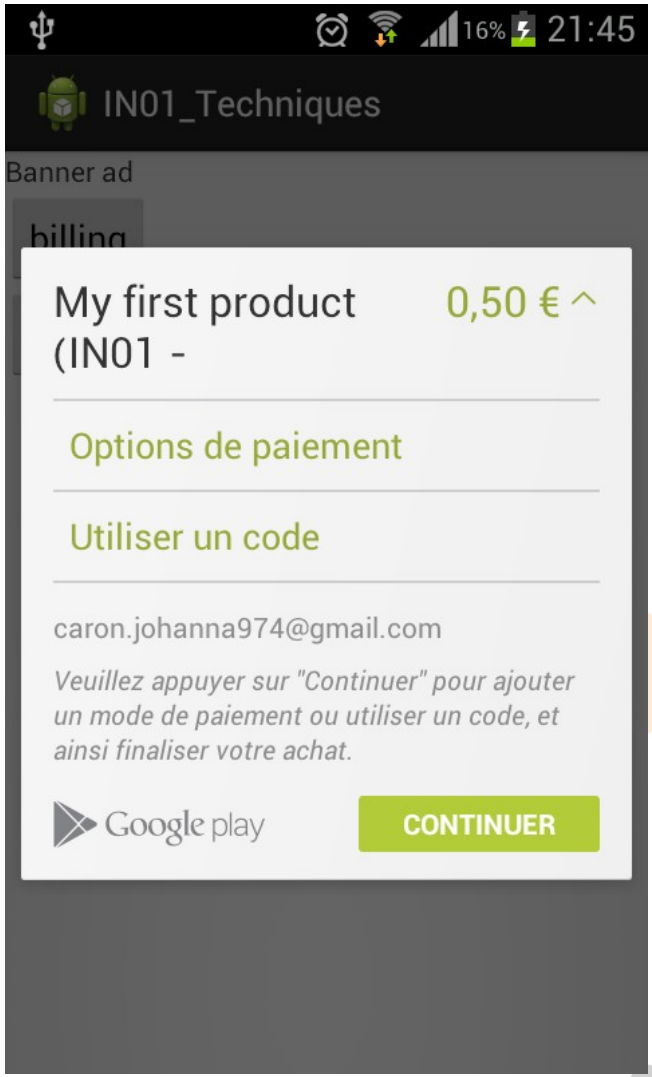
- Pour tester, il faut une application signée et publiée en Alpha / Bêta
- Sinon on a ça ::



- 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

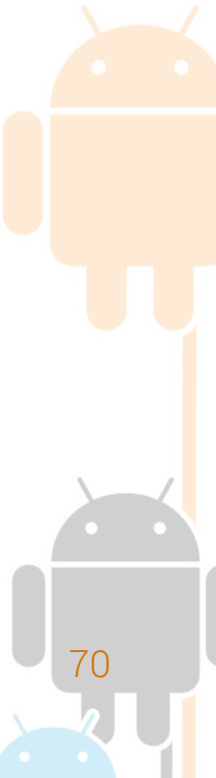


Sécurité

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

Conclusion

- Pour plus d'informations consulter :: 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 re-télécharger un élément acheté

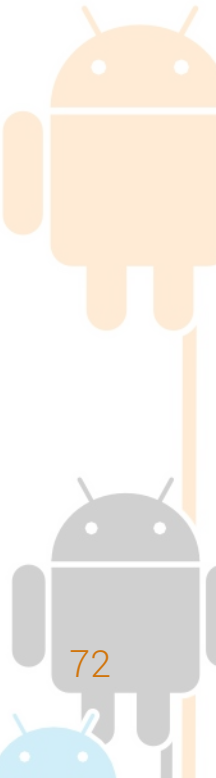


IN01 – Séance 06

Google Analytics

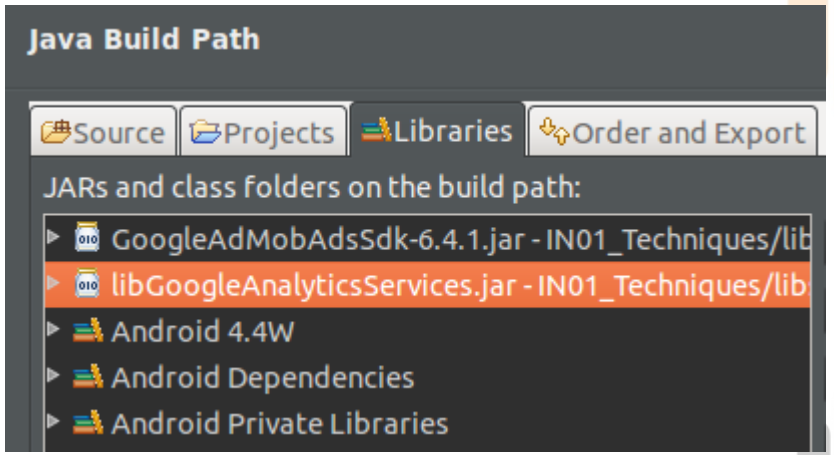
Vue d'ensemble

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



Installation

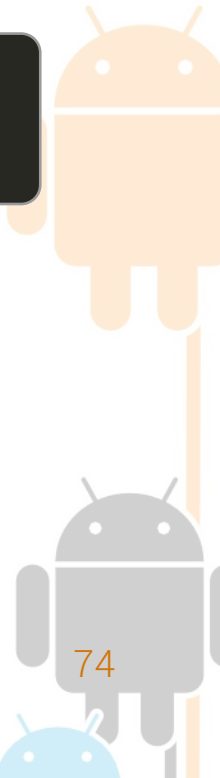
- Classique
- Télécharger depuis ici :: https://developers.google.com/analytics/devguides/collection/android/resources?hl=fr
- Placer libGoogleAnalyticsServices.jar dans le repertoire /libs du projet
- L'ajouter dans le classpath du projet



Autorisations

- On a l'habitude maintenant !! 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

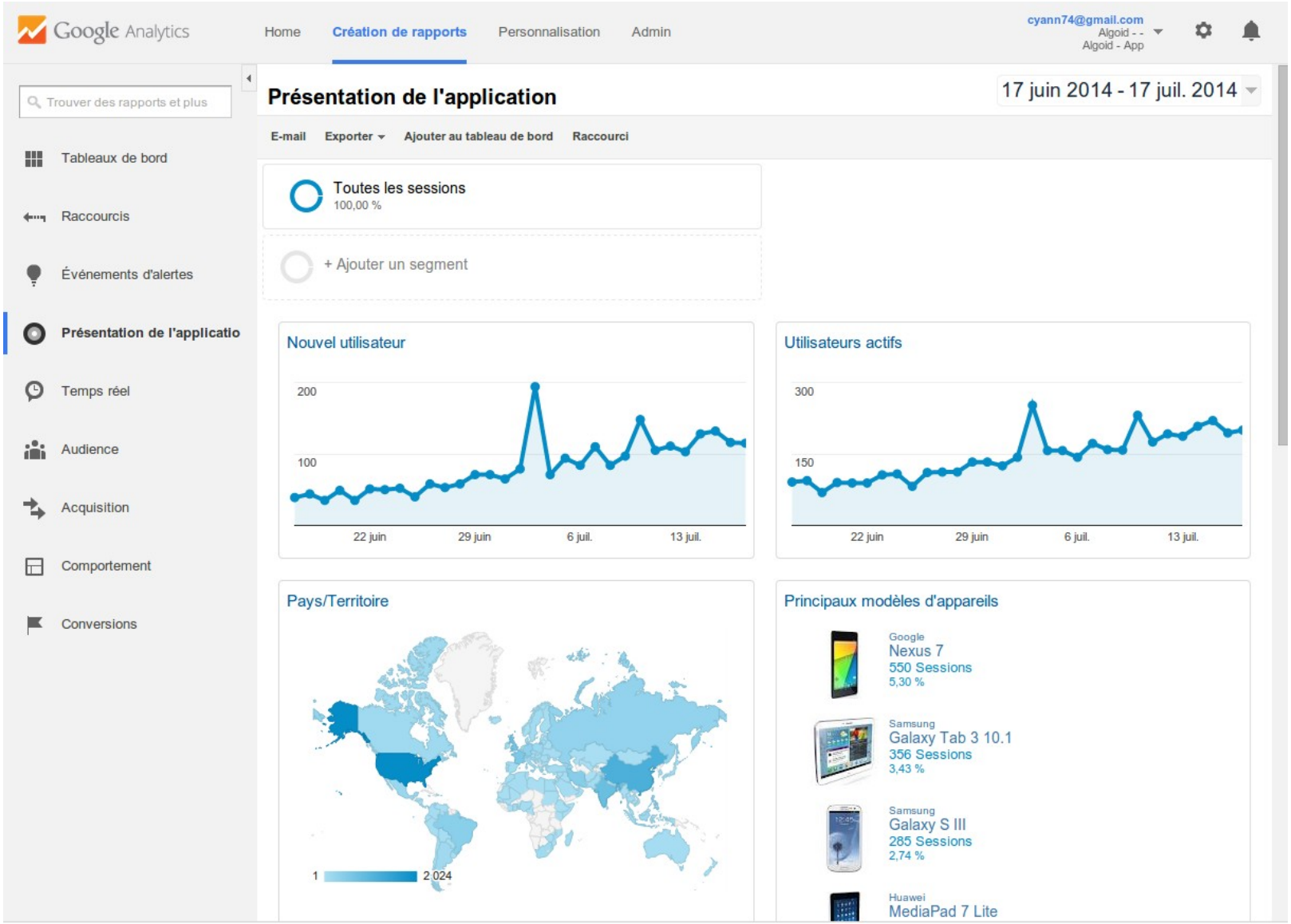
```xml
<?xml version="1.0" encoding="utf-8" ?>

<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


Google Analytics
Home
Création de rapports
Personnalisation
Admin
cyann74@gmail.com
Algoid - -
Algoid - App
Trouver des rapports et plus
Tableaux de bord
Raccourcis
Événements d'alertes
Présentation de l'applicatio
Temps réel
Audience
Acquisition
Comportement
Conversions
Présentation de l'application
17 juin 2014 - 17 juil. 2014
E-mail
Exporter
Ajouter au tableau de bord
Raccourci
Toutes les sessions
100,00 %
+ Ajouter un segment
Nouvel utilisateur
200
100
22 juin
29 juin
6 juil.
13 juil.
Utilisateurs actifs
300
150
22 juin
29 juin
6 juil.
13 juil.
Pays/Territoire
1
2 024
Principaux modèles d'appareils
Google
Nexus 7
550 Sessions
5,30 %
Samsung
Galaxy Tab 3 10.1
356 Sessions
3,43 %
Samsung
Galaxy S III
285 Sessions
2,74 %
Huawei
MediaPad 7 Lite

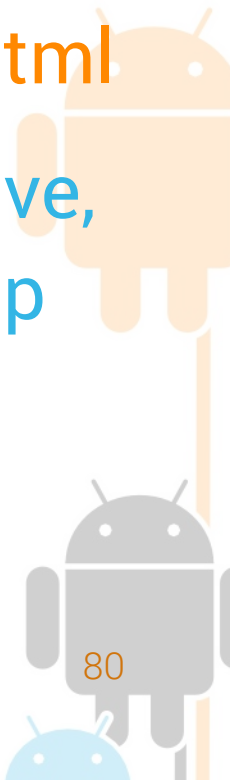
Évenements

- Selon des 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 autre :: 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 ?