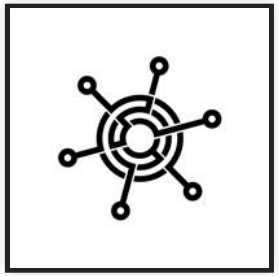
ANDROID, PERFECTIONNEMENT

- Qui je suis
- Qui vous êtes/ce que vous attendez
- Présentation du plan

Tristan SALAÜN Nov-2019

QUI JE SUIS

- Développeur d'applications mobiles Android
- Formateur Android/Java et Kotlin (scolaires et pros)
- Co-fondateur de Startup Marseille 

QUI ÊTES VOUS ?

Tour de table :

- Prénom
- Expérience (Java, Mobile, Android)
- Attentes

ANDROID, PERFECTIONNEMENT

- Introduction
- Outils avancés de développement
- Création d'IHM avancées
- Utilisation des capteurs
- ContentProvider et Services
- Tester une application Android
- Librairies et services utiles pour le développement Android

Tristan SALAÜN Nov-2019

INTRODUCTION

RAPPELS DES PRINCIPES DE BASE ANDROID

Les composants de base :

- Activités
- Fragments
- Service
- Intents
- Bundle
- Broadcast Receiver
- Content Providers

ACTIVITÉS

PRÉSENTATION

L'activité est le composant de base d'Android.

- Point d'entrée dans l'application.
- Plusieurs points d'entrée pour une seule application.
- Faible dépendance entre activités.
- Une interface graphique.
- Souvent en plein écran.

ACTIVITÉS

DÉCLARATION

L'activité est déclarés dans le manifest :

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```

ACTIVITÉS

INTENT FILTERS

Il est possible de définir des filtres qui permettrons de lancer l'activité. Écrivons l'exemple qui va nous permettre de récupérer du texte. Nous allons suivre les étapes suivantes :

- Création d'un bouton qui va partager du texte.
- Création d'une activité qui va afficher le texte reçu.
- Ajout d'un Intent Filter à cette seconde activité.
- Test du résultat.

ACTIVITÉS

CRÉATION DU BOUTON

- On déclare le bouton dans le `Layout`.

- On ajoute un `OnClickListener`.
- On implémente le partage du texte.

Pour partager le texte on va utiliser :

```
1 // Create the text message with a string
2 Intent sendIntent = new Intent();
3 sendIntent.setAction(Intent.ACTION_SEND);
4 sendIntent.setType("text/plain");
5 sendIntent.putExtra(Intent.EXTRA_TEXT, "Un texte à partager");
6 // Start the activity
7 startActivity(sendIntent);
```

ACTIVITÉS

CRÉATION D'UNE NOUVELLE ACTIVITÉ.

- File / New / Activity / Empty Activity (ReceiveTextActivity).
- Ajout du filtre dans le manifest
- Récupération et affichage du texte reçu.

Filtre dans le manifest :

```
<activity android:name=".ReceiveTextActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/plain" />
    </intent-filter>
</activity>
```

Récupération du texte partagé :

```
Bundle extras = getIntent().getExtras();
String receivedText = (extras != null) ? extras.getString(Intent.EXTRA_TEXT) : "";
Log.d(TAG, "onCreate received text = " + receivedText);
```

ACTIVITÉS

VÉRIFICATION DE L'INTENT.

Si vous changez la valeur de l'action par une valeur quelconque (voir ci-dessous), que se passe-t-il, et pourquoi ?

```
1 sendIntent.setAction("aaa");
```

Comment y remédier ?

Il est préférable de vérifier que l'Intent pourra être récupéré avec le code suivant :

```
1 // Verify that the intent will resolve to an activity
2 if (sendIntent.resolveActivity(getPackageManager()) != null) {
3     startActivity(sendIntent);
4 }
```

ACTIVITÉS

CHOOSER.

Lors du choix de l'application à lancer, choisissez : Toujours (Always).

Comment faire si l'on veut proposer le choix à chaque fois ?

On va forcer le lancement du chooser, avec le code suivant :

```
1 // Always use string resources for UI text.
2 // This says something like "Share this photo with"
3 String title = getResources().getString(R.string.chooser_title);
4 // Create intent to show the chooser dialog
5 Intent chooser = Intent.createChooser(sendIntent, title);
6 // Verify the original intent will resolve to at least one activity
7 if (sendIntent.resolveActivity(getPackageManager()) != null) {
8     startActivity(chooser);
9 }
```

ACTIVITÉS

OUVERTURE DEPUIS UN LIEN.

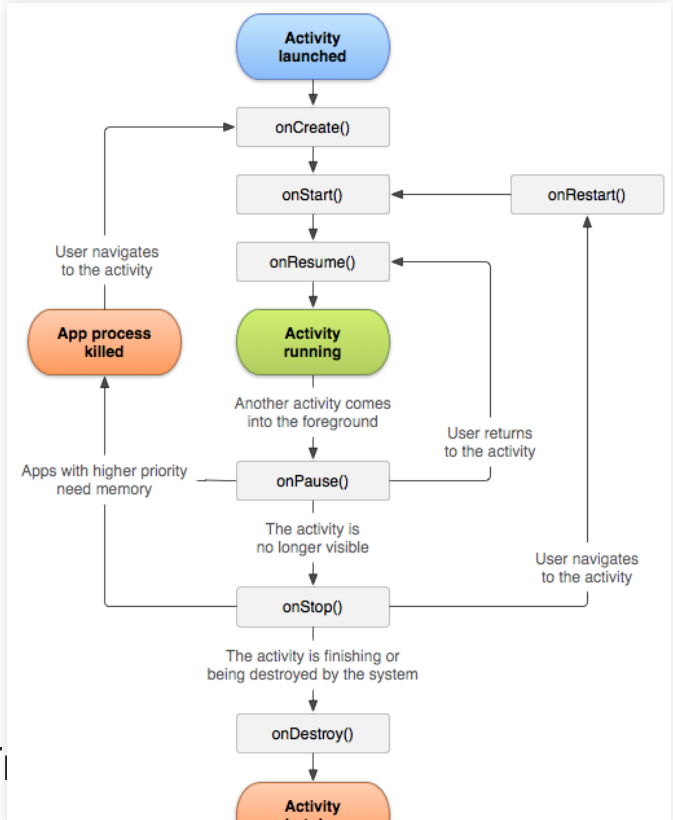
Nous pouvons avoir besoin de lancer notre application depuis un lien sur le web, pour cela nous allons utiliser un Intent Filter :

```
<intent-filter>
    <action android:name="android.intent.action.VIEW"/>

    <category android:name="android.intent.category.DEFAULT"/>
    <category android:name="android.intent.category.BROWSABLE"/>

    <data android:scheme="http"/>
    <data android:host="test.link.com"/>
</intent-filter>
```

Essayons de cliquer sur le lien suivant : Ouvrir application.

ACTIVITÉS

CYCLE DE VIE



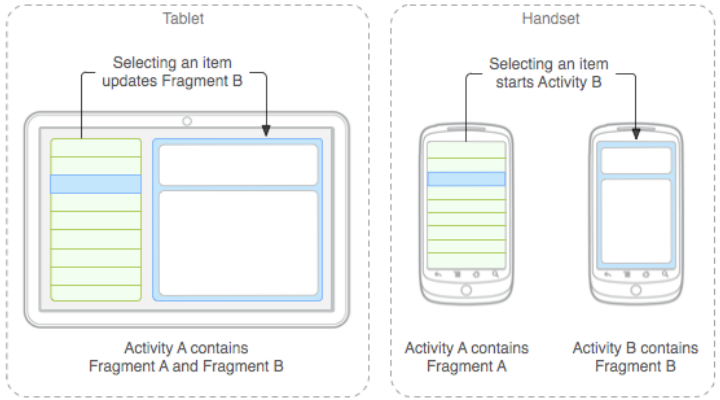
ACTIVITÉS

CYCLE DE VIE

Nous allons utiliser le Live Template : `android_lifecycle_activity`, d'une méthode dans notre classe de l'Activity.
Puis nous utilisons le Live Template : `android_tag`, juste après la déclaration de notre classe, avant la déclaration des méthodes.

FRAGMENTS

Nous allons détailler l'exemple : New / Activity / Master Detail Flow :



FRAGMENTS

Ajout d'un fragment statiquement (dans un XML) :

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
          android:orientation="horizontal"
          android:layout_width="match_parent"
          android:layout_height="match_parent">
<fragment android:name="com.example.news.ArticleListFragment"
          android:id="@+id/list"
          android:layout_weight="1"
          android:layout_width="0dp"
          android:layout_height="match_parent" />
<fragment android:name="com.example.news.ArticleReaderFragment"
          android:id="@+id/viewer"
          android:layout_weight="2"
          android:layout_width="0dp"
          android:layout_height="match_parent" />
```

FRAGMENTS

Ajout d'un fragment dynamiquement, via le `FragmentManager` :

```
1 FragmentManager fragmentManager = getSupportFragmentManager();
2 FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

Qui permet par la suite de gérer dynamiquement les fragments :

```
1 ExampleFragment fragment = new ExampleFragment();
2 fragmentTransaction.add(R.id.fragment_container, fragment);
3 fragmentTransaction.commit();
```

PRÉSENTATION DU DÉVELOPPEMENT NATIF AVEC NDK. JNI.

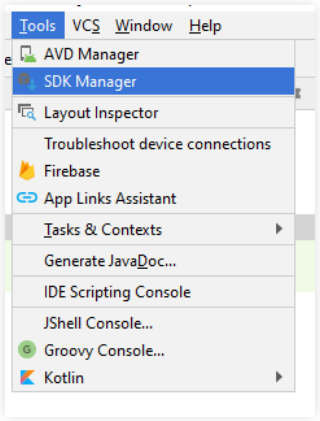
- C/C++.
- Performances.
- Réutilisation de code.

CRÉATION D'UN PROJET.

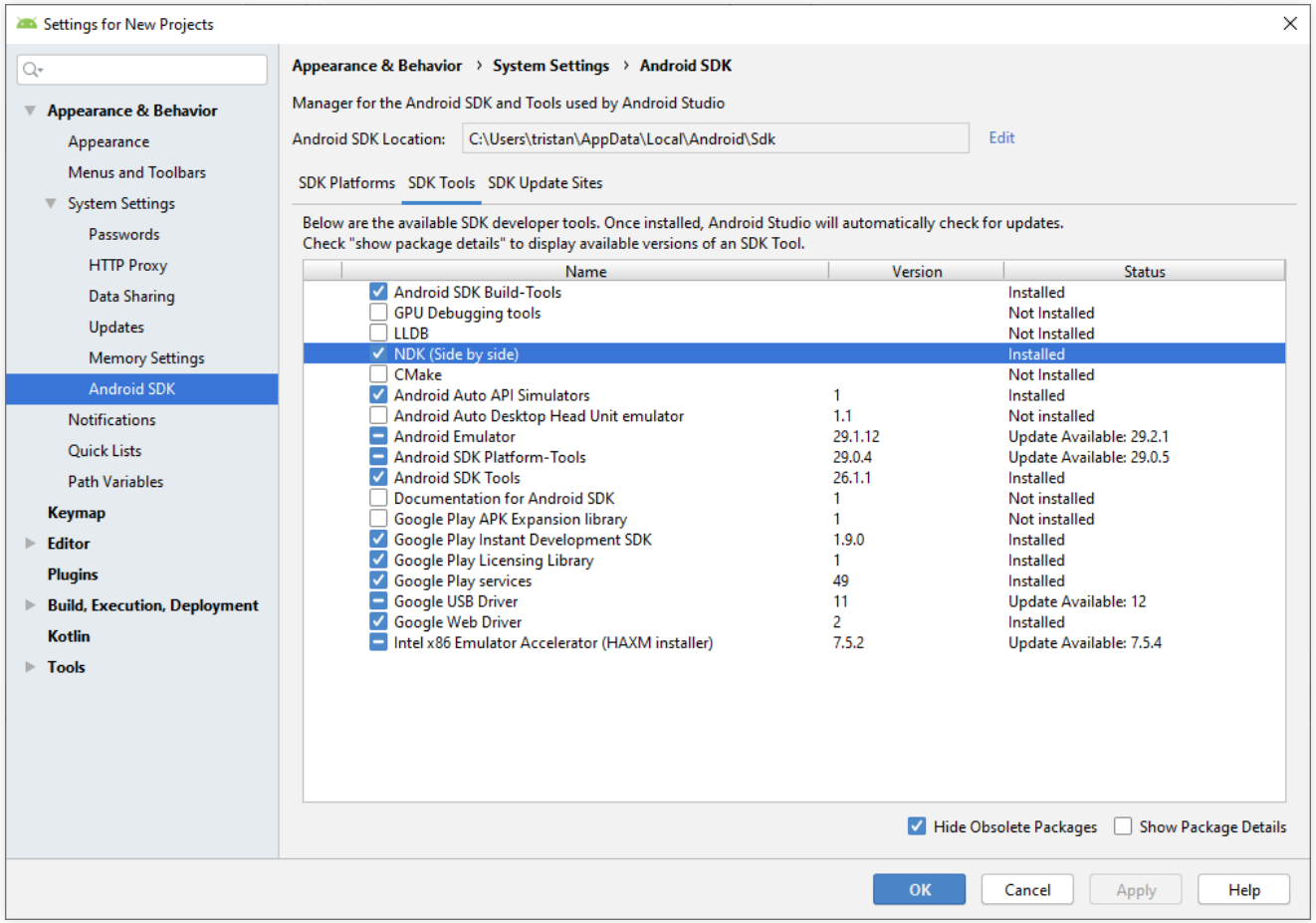
Nous allons étudier le fonctionnement d'une application simple faisant appel à du code C/C++.
Pour cela nous allons suivre plusieurs étapes :

- Installation du NDK.
- Création d'un projet C/C++.
- Test.

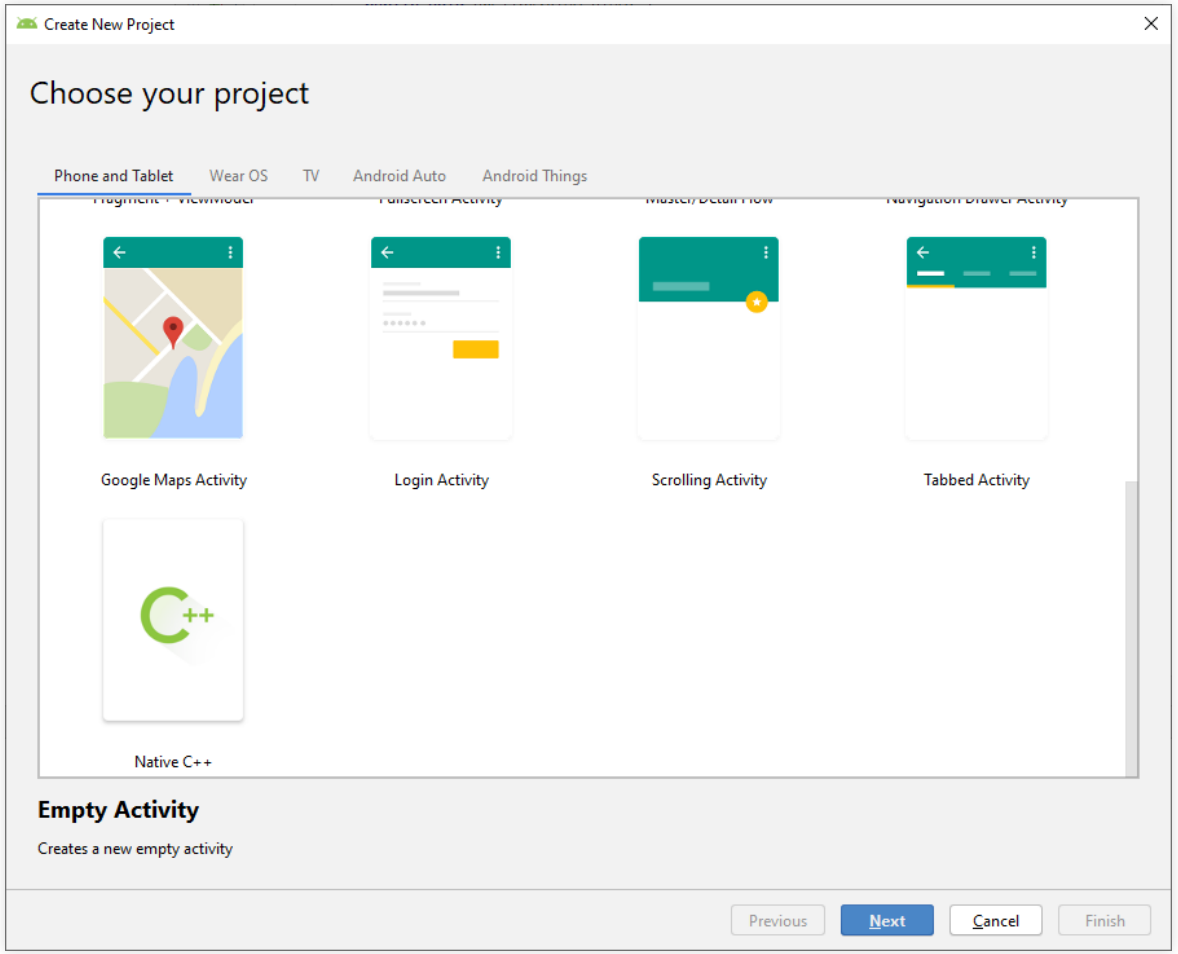
INSTALLATION DU NDK.



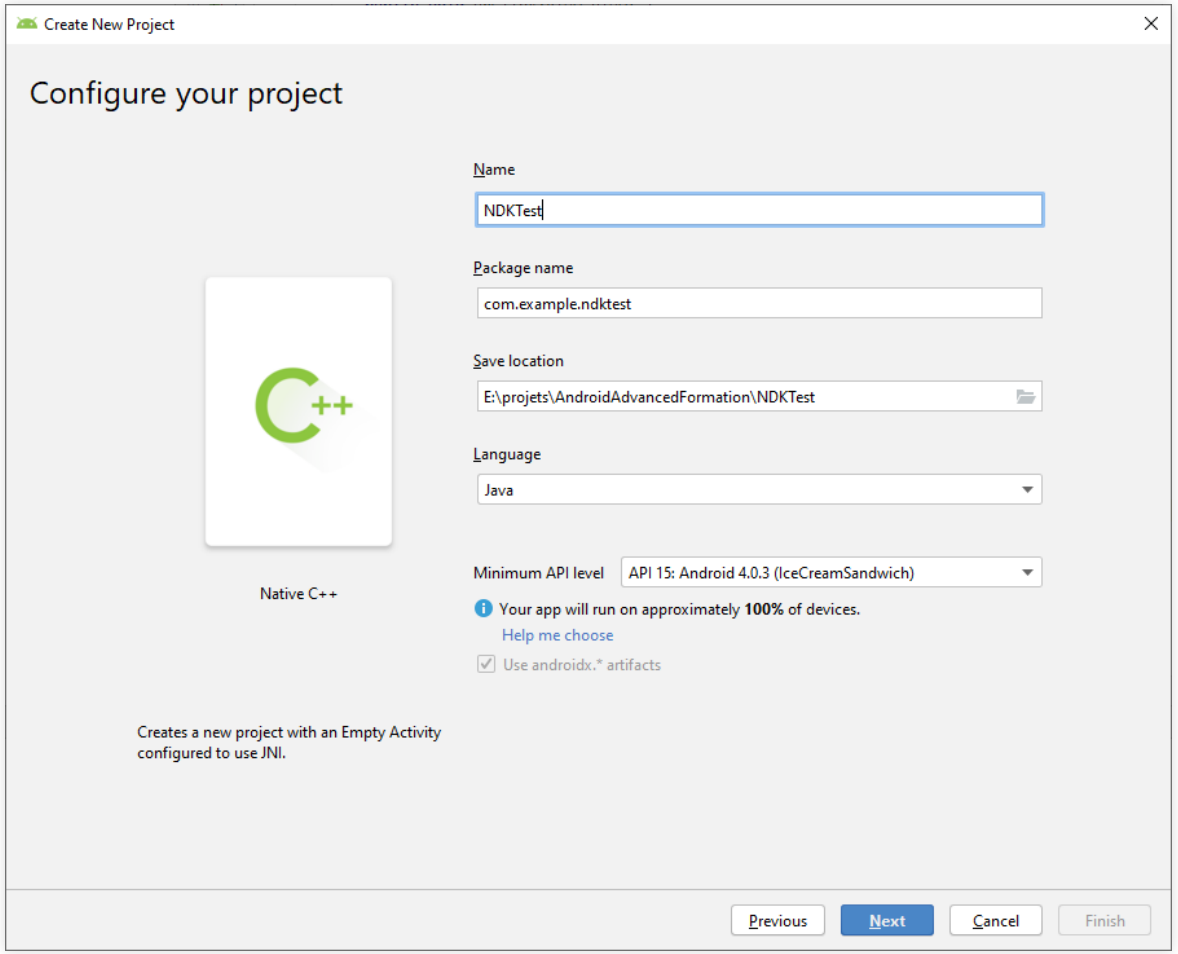
INSTALLATION DU NDK.



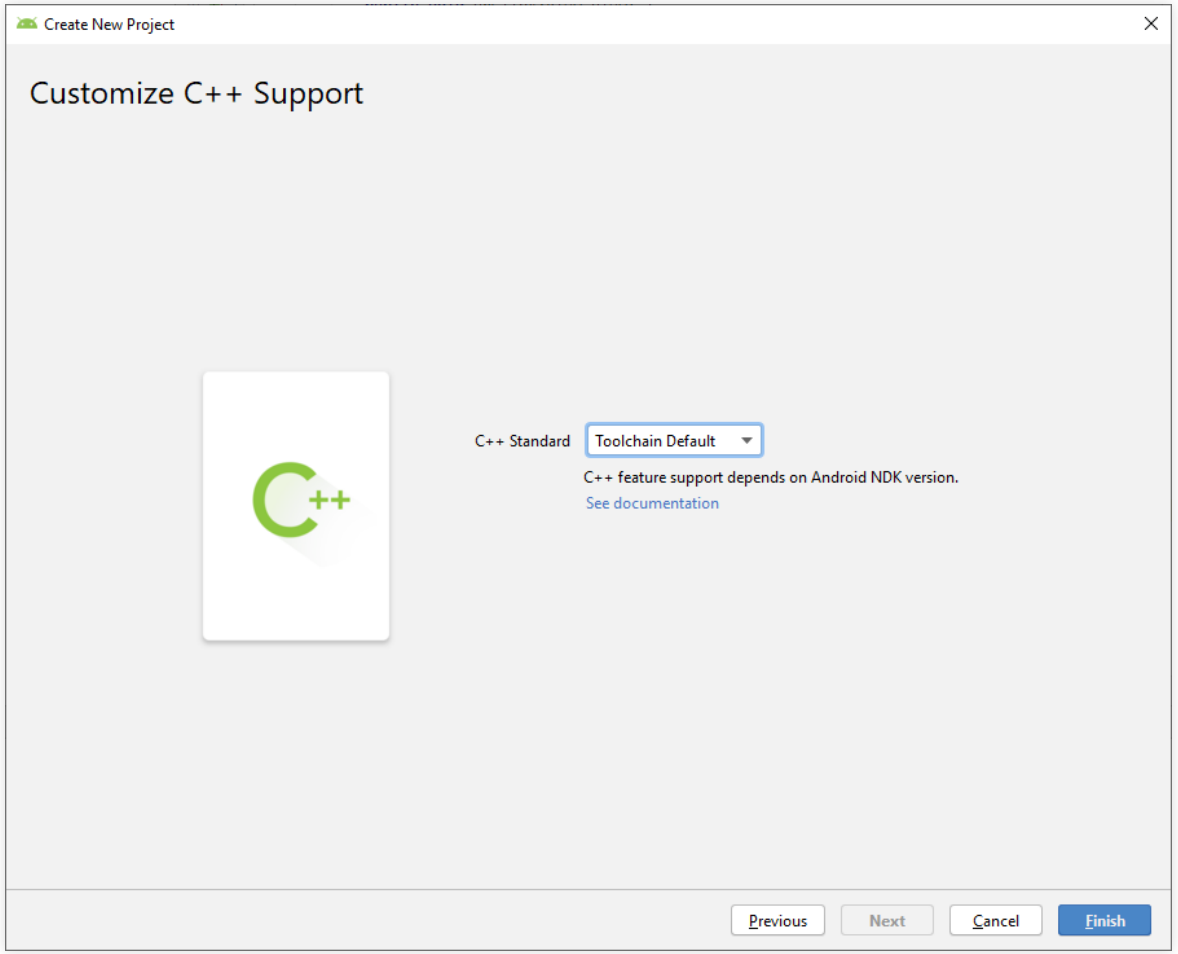
CRÉATION D'UN PROJET.



NOMMAGE DU PROJET.



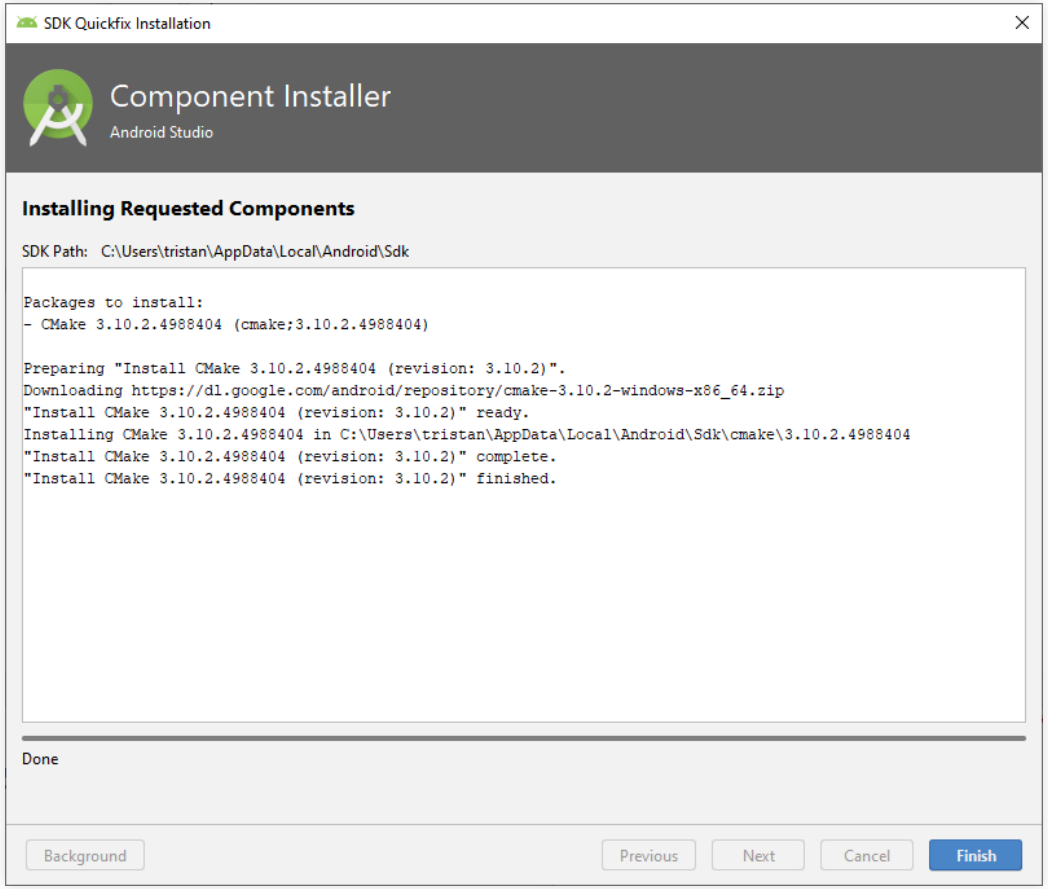
SÉLECTION DE LA TOOLCHAIN.



GESTION ERREUR CMAKE.

```
ERROR: CMake '3.10.2' was not found in PATH or by cmake.dir property.
Install CMake 3.10.2
```

CMAKE INSTALLÉ.



AJOUT CMAKE DANS LE PATH.

```
ERROR: CMake '3.10.2' was not found in PATH or by cmake.dir property.
Set cmake.dir in local.properties to C:\Users\tristan\AppData\Local\Android\Sdk\cmake\3.10.2.4988404
```

TEST.

Il ne reste plus qu'à tester le code, et voir les points les plus importants ensemble.

FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

INSTANT RUN

Instant Run n'est plus disponible sur la version actuelle. Mais il y a deux systèmes de patch.

FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

DÉBUG

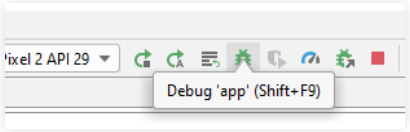
Poser un point d'arrêt :

```
23                  sendIntent.setAction(Intent.ACTION_SEND);
24                  sendIntent.setType("text/plain");
25                  sendIntent.putExtra(Intent.EXTRA_TEXT,  value: "Un texte à partager");
26                  // Start the activity
27              //      if (sendIntent.resolveActivity(getPackageManager()) != null) {
28              //          startActivity(sendIntent);
29              //      }
30
31                  // Always use string resources for UI text.
32                  // This says something like "Share this photo with"
33                  String title = getResources().getString(R.string.chooser_title);
34                  // Create intent to show the chooser dialog
35                  Intent chooser = Intent.createChooser(sendIntent, title);
36                  // Verify the original intent will resolve to at least one activity
37                  if (sendIntent.resolveActivity(getPackageManager()) != null) {
38                      startActivity(chooser);
39                  }
40
41              }
42          });
43      }
44  }
45
```

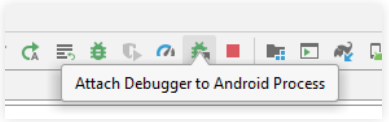
FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

DÉBUG

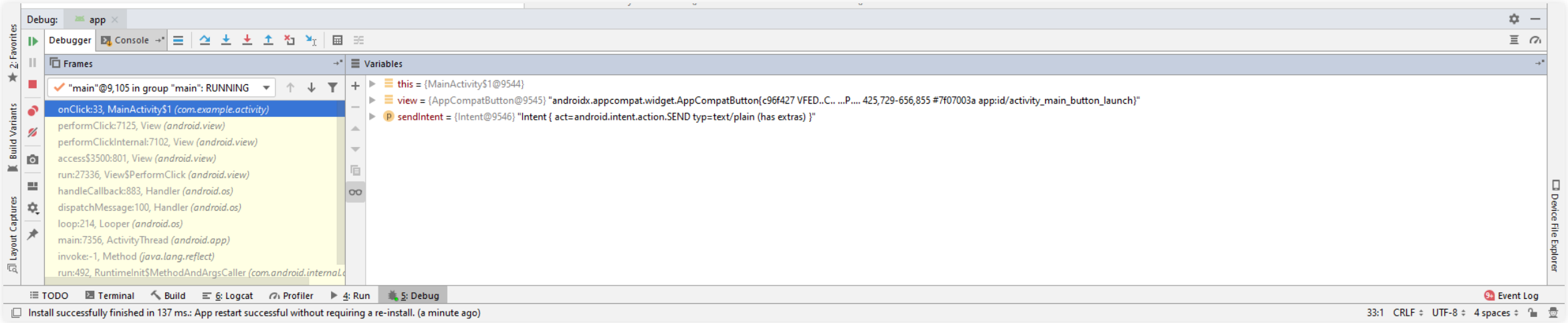
Lancer en mode débug :

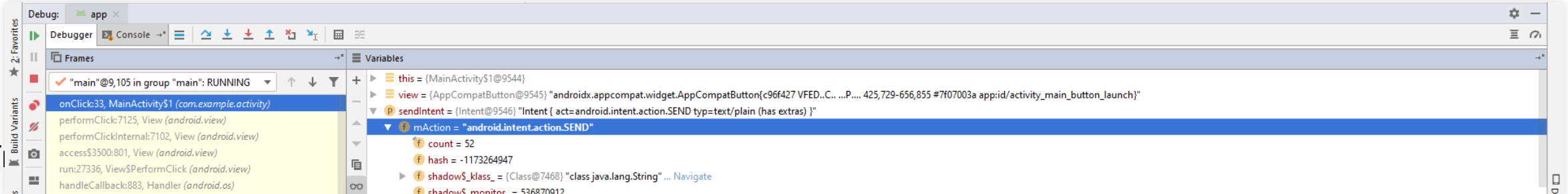


Ou attacher le débugger en cours d'exécution :



DÉBUG FONCTIONNALITÉS AVANCÉS ANDROID STUDIO





FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

DEBUG EN WIFI

Il est possible de se connecter en WiFi pour ADB. En ligne de commande lancer la commande :

```
1 adb tcpip 5555
```

Puis il suffit de se connecter sur l'adresse IP du téléphone, par exemple :

```
1 adb connect 192.168.1.99
```

FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

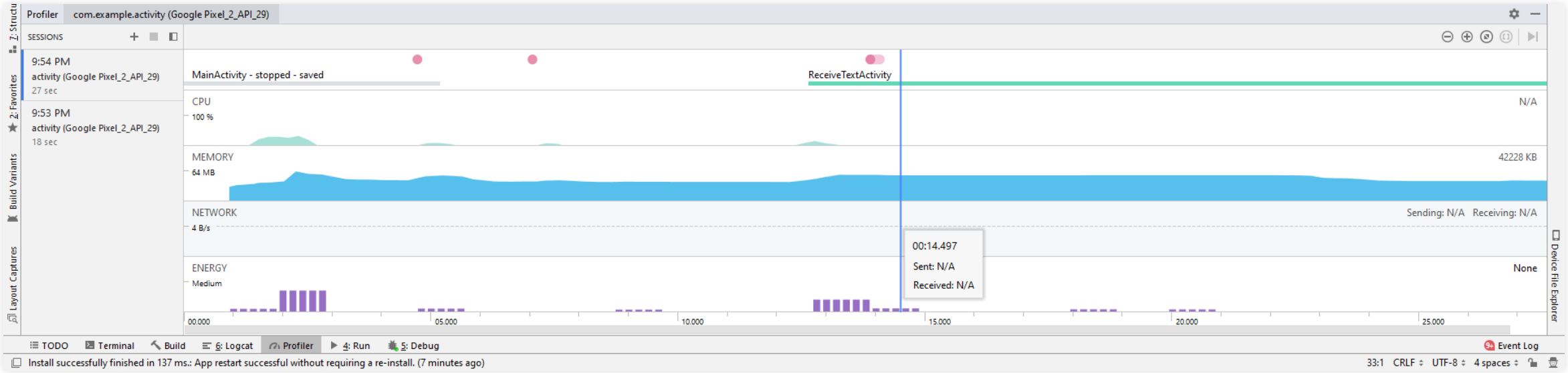
DEBUG EN WIFI AUTOMATIQUE

Nous pouvons utiliser le script suivant (sous Windows dans un fichier .bat) pour nous connecter directement :

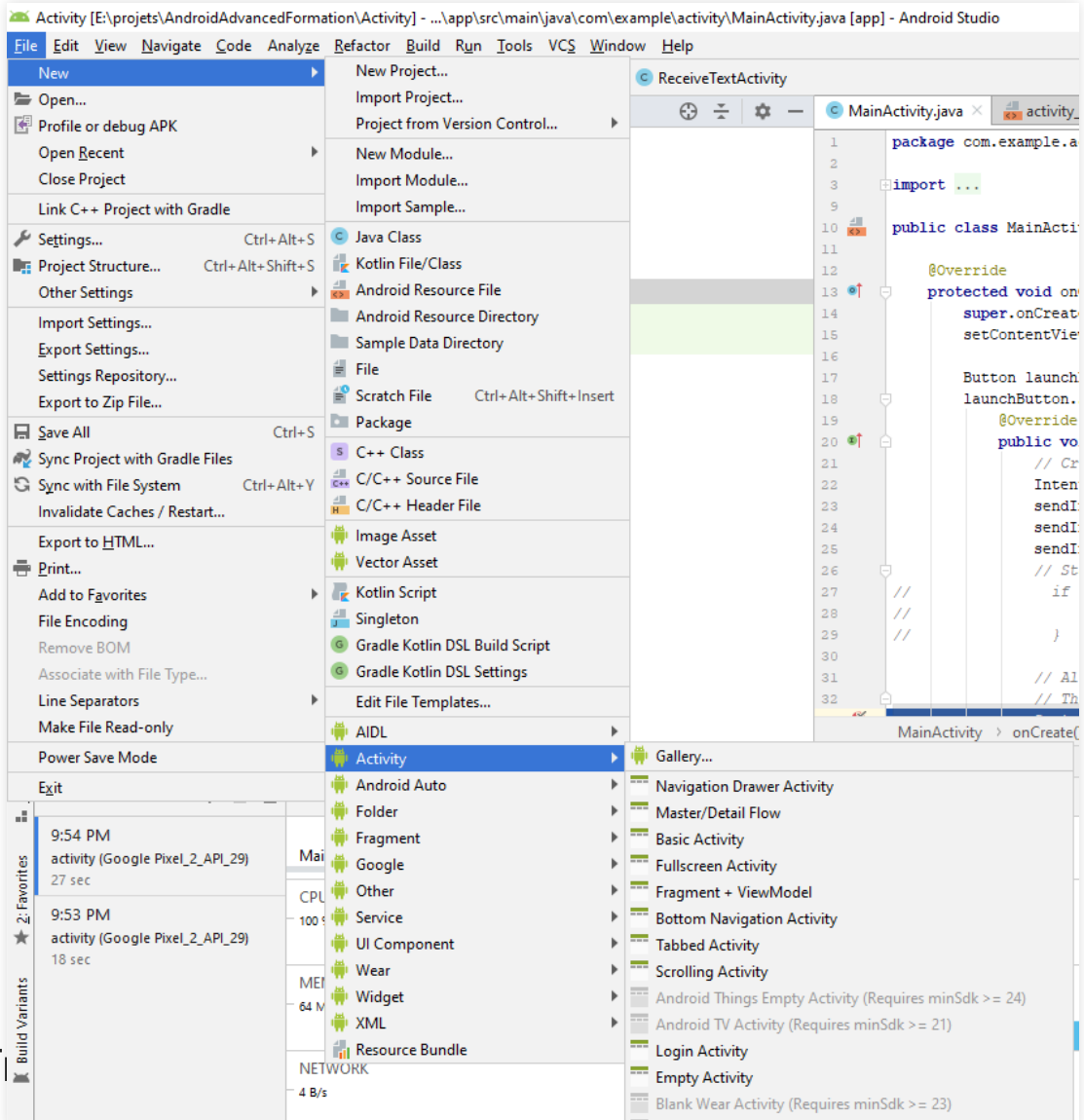
```
@echo off
echo Activate tcp mode
adb -d tcpip 5555

REM Sleep for 1 seconds
echo Wait a bit
ping -n 2 127.0.0.1>nul

echo Find IP
FOR /F "tokens=2 skip=1" %%A IN ('adb -d shell ip -f inet addr show wlan0') DO (
        FOR /F "tokens=1 delims=/" %%B IN ("%%A") DO (
            echo "Connecting to %%B"
                adb connect %%B
        )
        goto :eof
```

FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

PROFILE



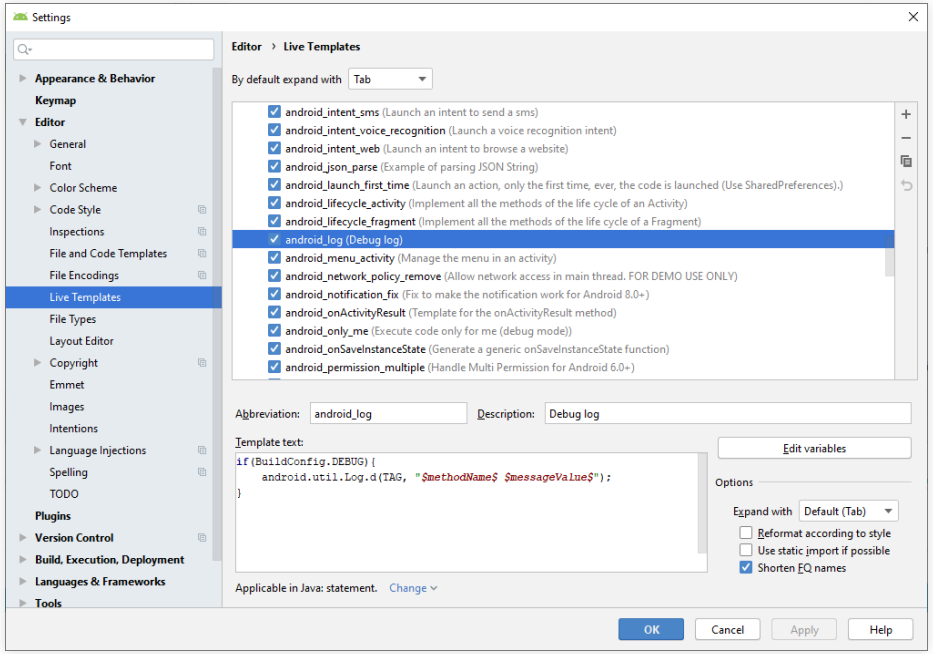
TEMPLATESFONCTIONNALITÉS AVANCÉS ANDROID STUDIO



FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

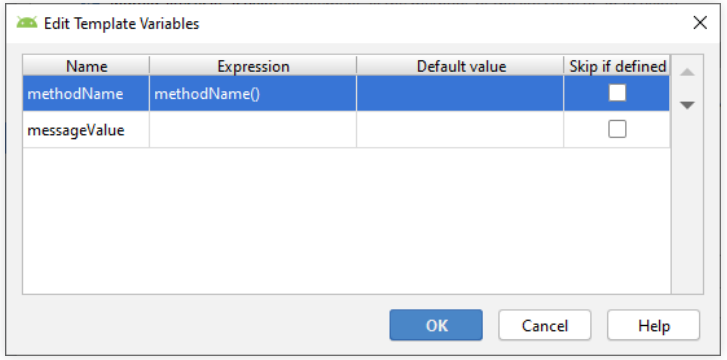
LIVE TEMPLATES

File / Settings
Editor / Live Templates.



FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

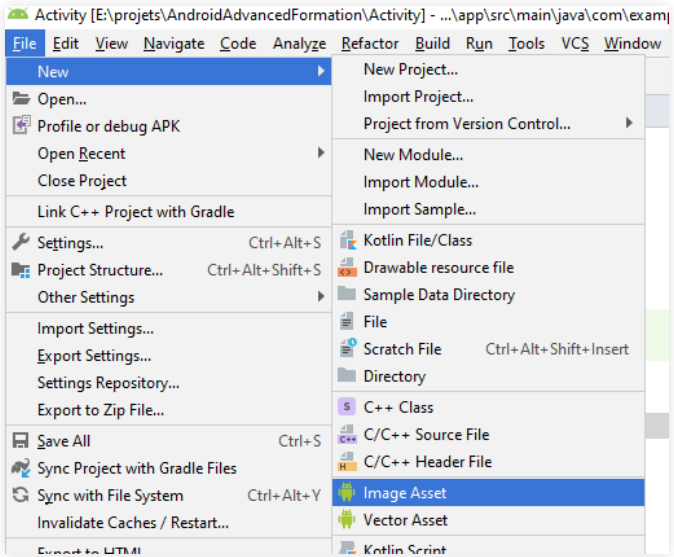
LIVE TEMPLATES



FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

GESTION DES GRAPHIQUES

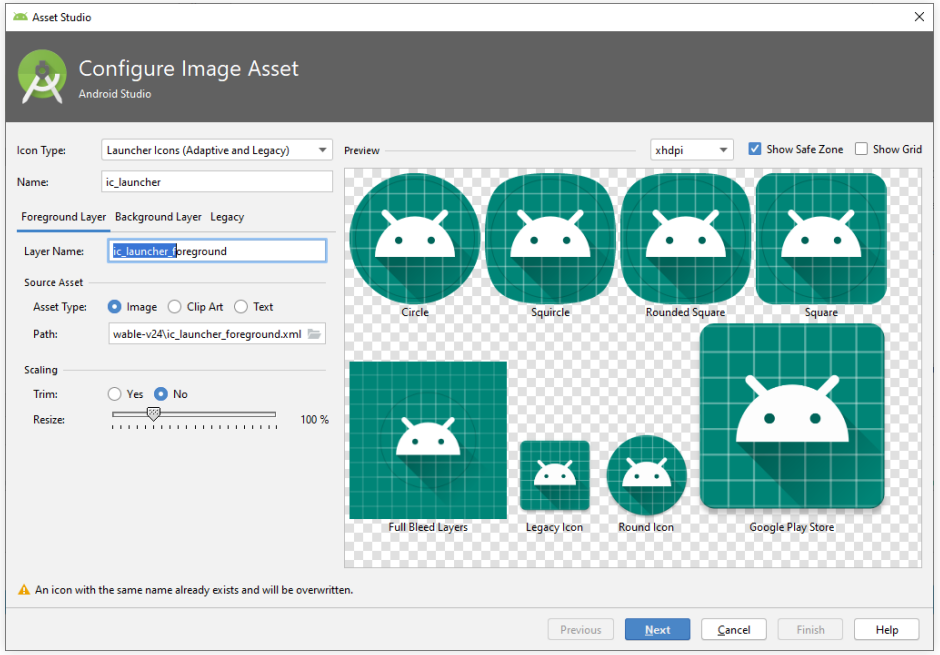
Création d'icônes :



FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

GESTION DES GRAPHIQUES

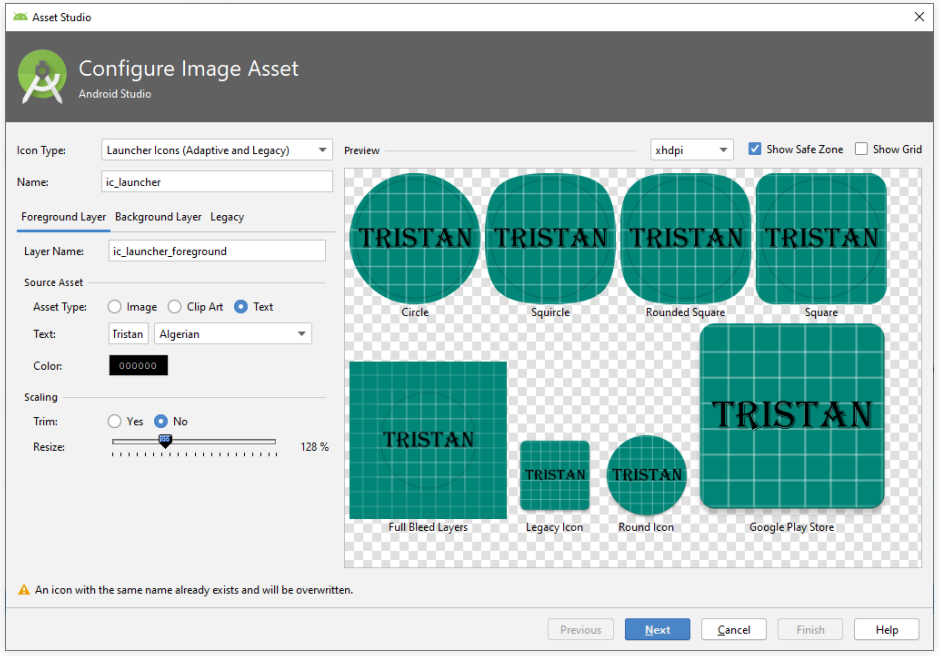
Création d'icônes :



FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

GESTION DES GRAPHIQUES

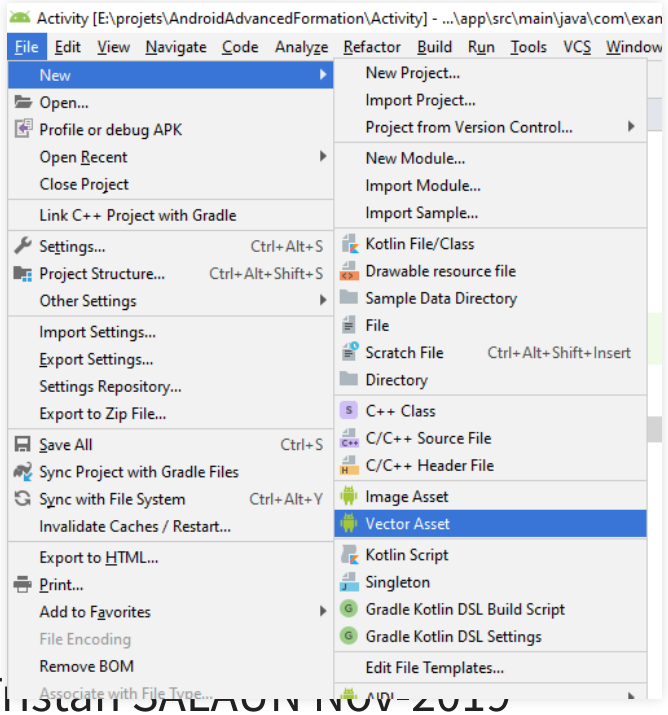
Création d'icônes :



FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

GESTION DES GRAPHIQUES

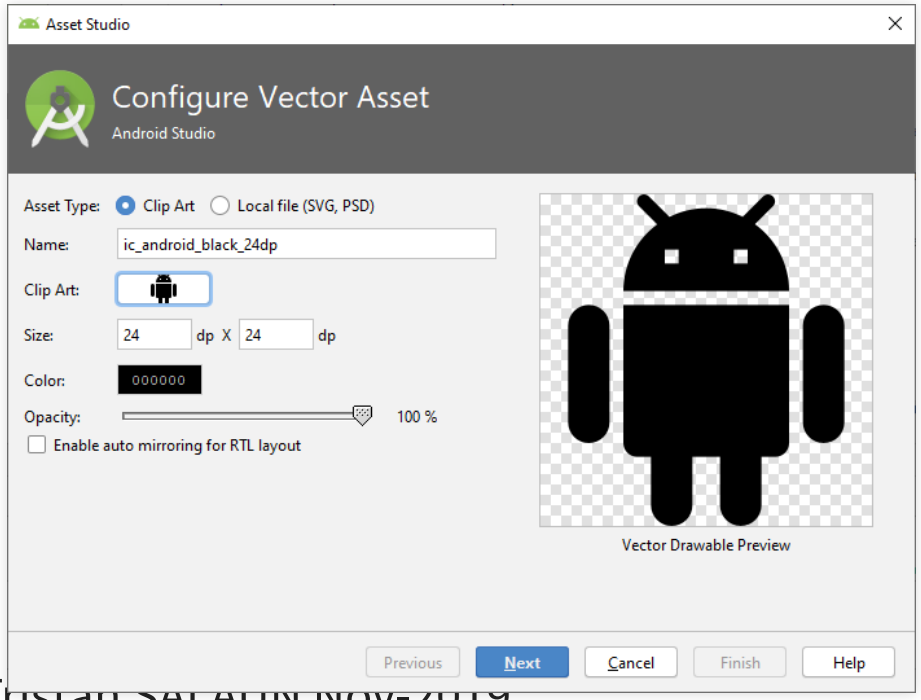
Création de ressources vectorielles :



FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

GESTION DES GRAPHIQUES

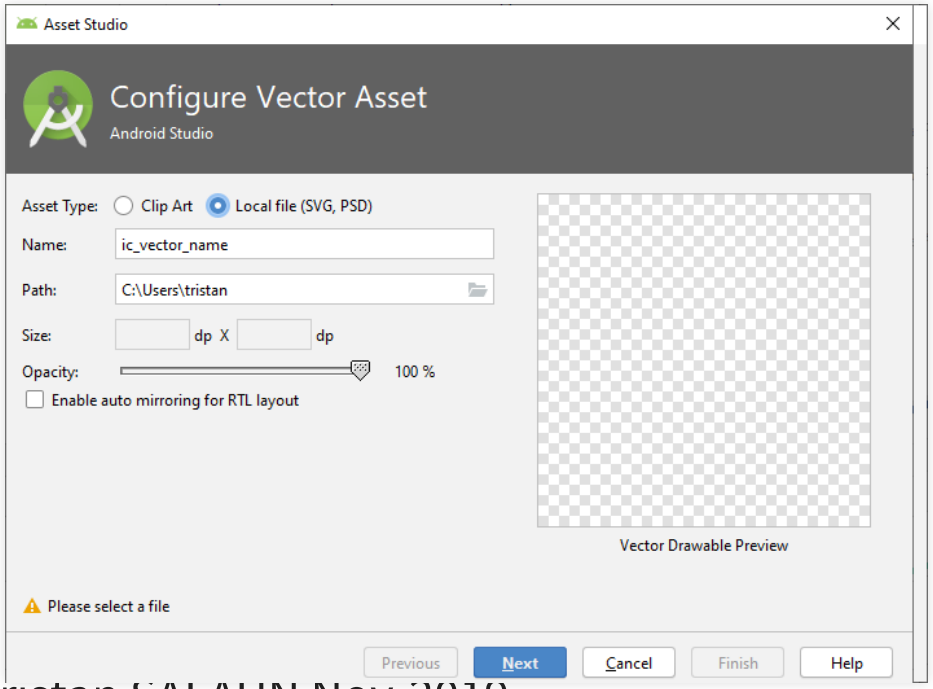
Création de ressources vectorielles :



FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

GESTION DES GRAPHIQUES

Création de ressources vectorielles :



FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

CHANGEMENT DE LA TEINTE

L'image vectorielle à plusieurs avantages :

- Est compacte.
- Être affichée avec différentes tailles sans perte de qualité.
- Changer de couleur facilement.

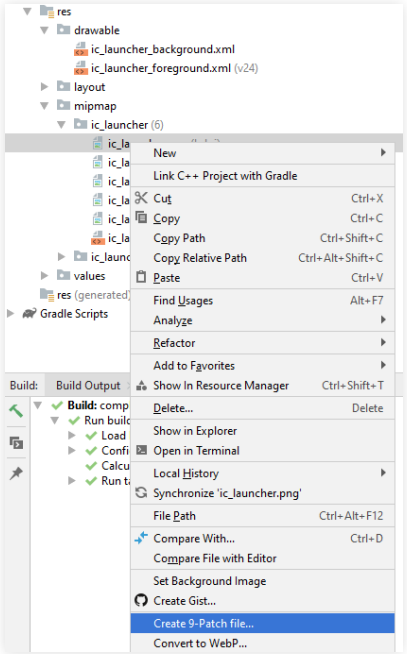
Testons tous cela, par exemple :

```
<ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_cloud"
        android:tint="@color/colorAccent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent" />
```

FONCTIONNALITÉS AVANCÉS ANDROID STUDIO

GESTION DES GRAPHIQUES

Images extensibles (9patch) :



Les ressources graphiques sont disponibles ici.

NOTIFICATIONS

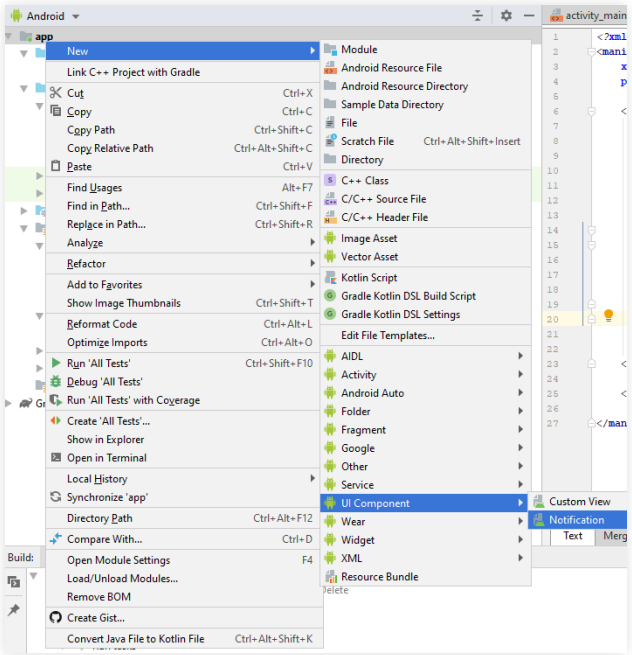
LES TOASTS

Avons nous besoin de revoir les Toasts ?

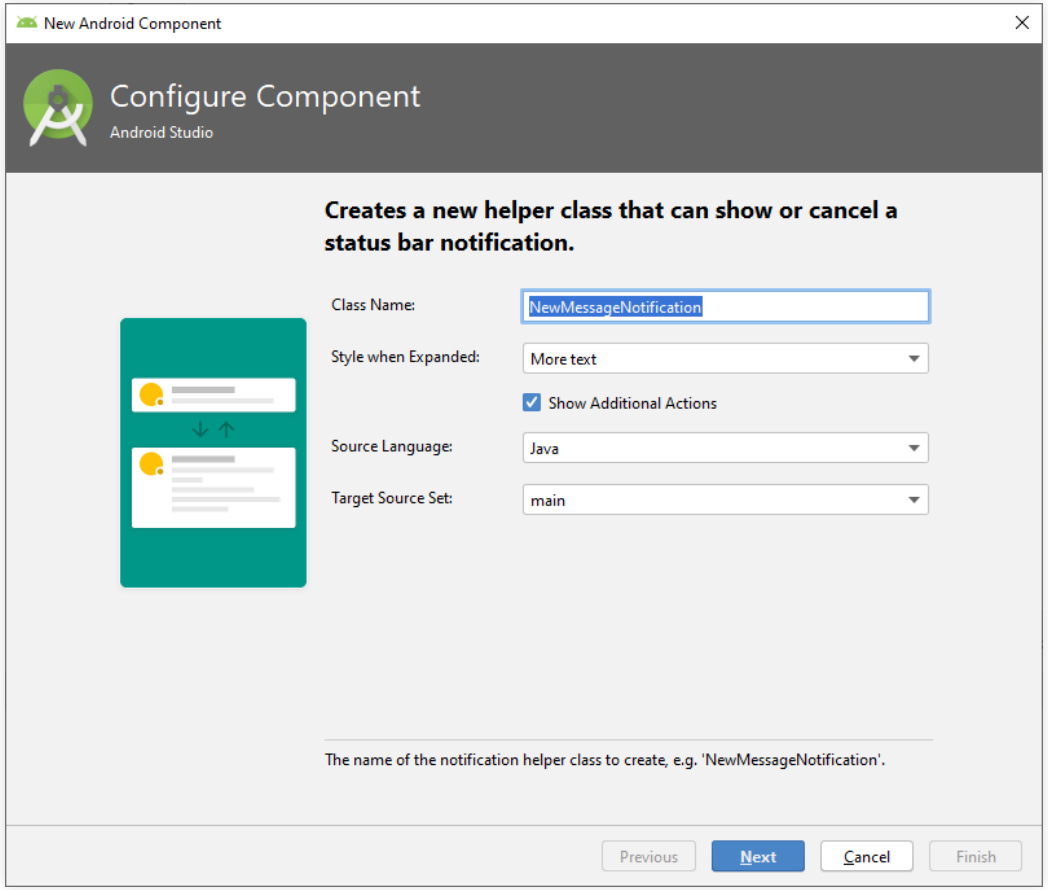
NOTIFICATIONS

NOTIFICATION

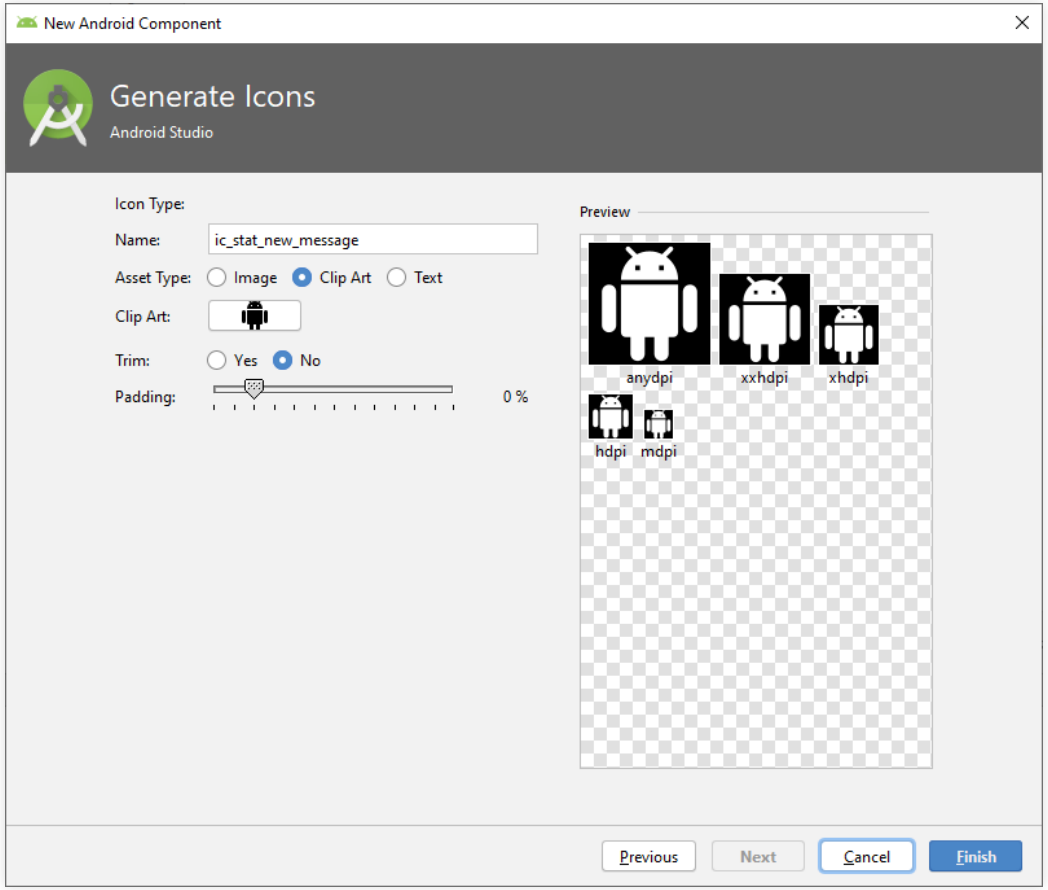
Les notifications, sont bien plus puissantes qu'il n'y parrait, voyons un peu plus en détail leur fonctionnement.



NOTIFICATIONS



NOTIFICATIONS



PERMISSIONS 6.0

APPORTS DU SDK 6.0. LES PERMISSIONS À LA DEMANDE.

- Plus de protection de la vie privée.
- Granularité des permissions.
- Permissions à l'exécution.

PERMISSIONS 6.0

TYPE DE PERMISSIONS

- Normal permissions.
- Signature permissions.
- Dangerous permissions.
- Special permissions (SYSTEM_ALERT_WINDOW et WRITE_SETTINGS).

Les permissions sont regroupées en groupes de permissions. Obtenir une permission d'un groupe permet d'obtenir les autres.

PERMISSIONS 6.0

EXEMPLE D'UTILISATION

Nous voulons pouvoir passer un appel téléphonique, nous allons avoir besoin de la permission `CALL_PHONE`.

Pour cela nous allons passer par plusieurs étapes :

- Ajout dans le manifest.
- Vérification de l'accord.
- Gestion du retour.
- Mise en place rapide.
- Mise en place encore plus rapide.
- Solution alternative.

PERMISSIONS 6.0

AJOUT DANS LE MANIFEST.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example.phonecall">

    <uses-permission android:name="android.permission.CALL_PHONE"/>

    <application ...>
        ...
    </application>
</manifest>
```

PERMISSIONS 6.0

VÉRIFICATION DE LA PERMISSION.

```
1 if (ContextCompat.checkSelfPermission(thisActivity, Manifest.permission.CALL_PHONE)
2         != PackageManager.PERMISSION_GRANTED) {
3     // Permission is not granted
4 }
```

PERMISSIONS 6.0

DEMANDE DE LA PERMISSION.

```
// Here, thisActivity is the current activity
if (ContextCompat.checkSelfPermission(thisActivity,

        Manifest.permission.CALL_PHONE)
        != PackageManager.PERMISSION_GRANTED) {

    // Permission is not granted
    // Should we show an explanation?
    if (ActivityCompat.shouldShowRequestPermissionRationale(thisActivity,
            Manifest.permission.CALL_PHONE)) {
        // Show an explanation to the user *asynchronously* -- don't block
        // this thread waiting for the user's response! After the user
        // sees the explanation, try again to request the permission.
    } else {
        // No explanation needed; request the permission
```

PERMISSIONS 6.0

GESTION DE LA RÉPONSE

```java
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {

    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_CALL_PHONE: {
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // permission was granted, yay! Do the job you need to do.
            } else {
                // permission denied, boo! Disable the functionality that depends on this permission.
            }
            return;
        }
        // other 'case' lines to check for other
```

PERMISSIONS 6.0

MISE EN PLACE RAPIDE.

Afin de mettre en place un code similaire, je vous propose d'utiliser un live template.

- En dehors d'une méthode, utiliser le live template : `android_permission_single`.
- Choisir CALL_PHONE.
- Suivre les instructions dans les commentaires.
- Implémenter le code dans `actionToBeCalled()`.
- Ajouter `@SuppressLint("MissingPermission")` sur cette méthode pour supprimer le warning.
- Appeler la méthode `callAction()`.

PERMISSIONS 6.0

MISE EN PLACE ENCORE PLUS RAPIDE.

Nous pouvons aussi utiliser des librairies externe pour gérer les permissions, par exemple :

- https://github.com/har5hit/PermissionHelper
- https://github.com/jksiezni/permissive

PERMISSIONS 6.0

SOLUTION ALTERNATIVE.

Utiliser un intent qui n'a pas besoin d'une permission particulière, dans notre cas, utiliser :

```
1 String url = "tel:"+"0612345678";
2 Intent callIntent = new Intent(Intent.ACTION_DIAL, Uri.parse(url));
3 startActivity(callIntent);
```

PERMISSIONS 6.0

AUTRES EXEMPLES

Dans d'autres cas nous pouvons nous passer des permissions :

- Accès à Internet (intent, et intent filter pour le retour).
- Accès aux contact (application de SMS, ...) : mode dégradé.
- Enregistrement dans la mémoire privée au lieu de la mémoire externe.

OUTILS AVANCÉS DE DÉVELOPPEMENT

- Paramétrer le build avec Gradle.
- Améliorer son code source avec Lint.
- Mettre au point et profiler/monitorer une application.
- Optimisation de l'APK avec ProGuard.

GRADLE

PARAMÈTRES DE BASE

Détaillons un fichier graddle de base :

```
android {
    compileSdkVersion 29
    buildToolsVersion "29.0.2"
    defaultConfig {
        applicationId "com.example.services"
        minSdkVersion 15
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
```

GRADLE

TYPES DE BUILD

Par défaut nous avons le type `debug` qui est définis. Un second type `release` est aussi définis.

GRADLE

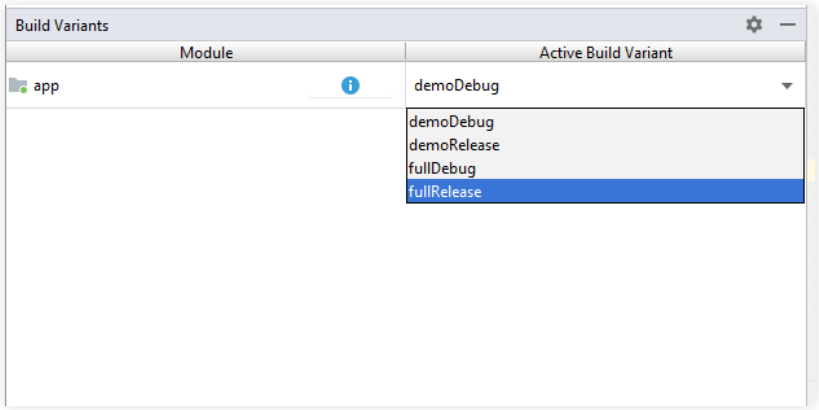
FLAVORS

Nous pouvons définir plusieurs dérivés de notre application principale, via les `flavors`.
Tous les détails ici :
https://developer.android.com/studio/build/build-variants#product-flavors.

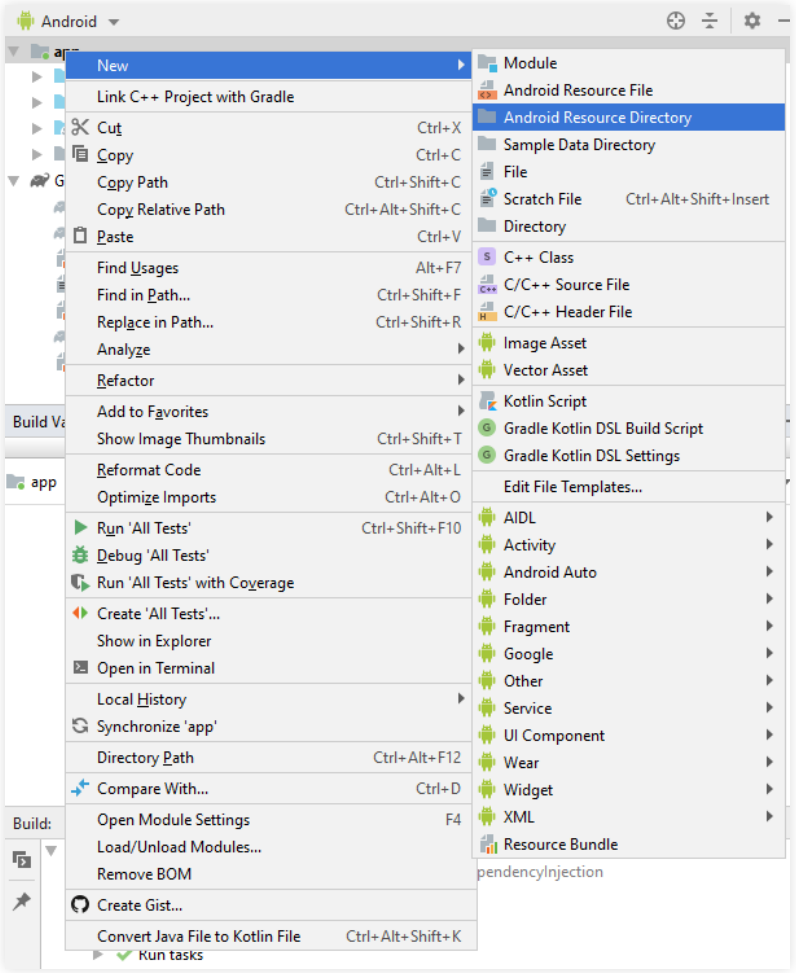
Ajoutons le code suivant :

```
buildTypes {
    // ...
}
flavorDimensions "version"
productFlavors {
    demo {
        dimension "version"
    }
    full {
        dimension "version"
    }
}
```

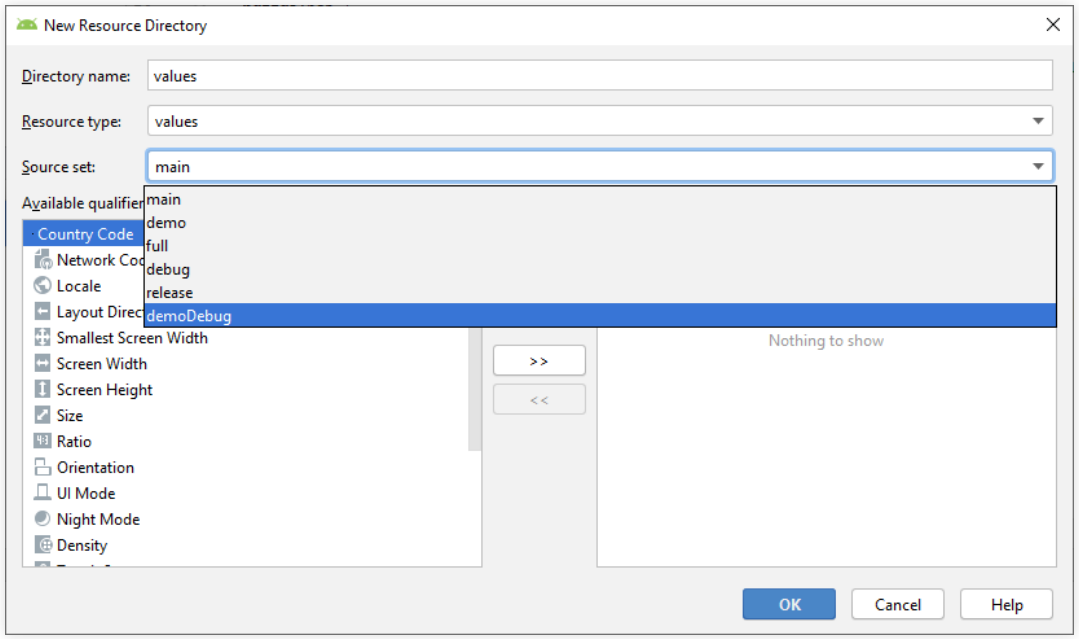
GRADLE


Build Variants
Module
Active Build Variant
app
demoDebug
demoDebug
demoRelease
fullDebug
fullRelease

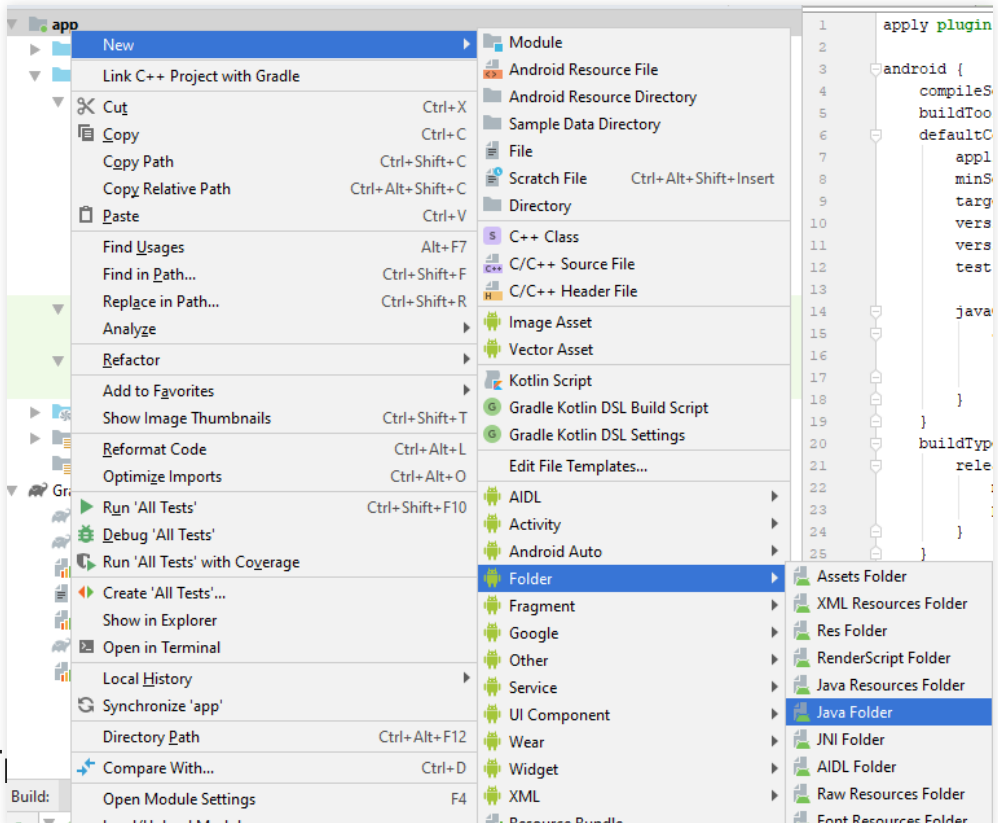
GRADLE


Android
New
Link C++ Project with Gradle
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy Relative Path Ctrl+Alt+Shift+C
Paste Ctrl+V
Find Usages Alt+F7
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Run 'All Tests' Ctrl+Shift+F10
Debug 'All Tests'
Run 'All Tests' with Coverage
Create 'All Tests'...
Show in Explorer
Open in Terminal
Local History
Synchronize 'app'
Directory Path Ctrl+Alt+F12
Compare With... Ctrl+D
Open Module Settings F4
Load/Unload Modules...
Remove BOM
Create Gist...
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Module
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Directory
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Kotlin Script
Gradle Kotlin DSL Build Script
Gradle Kotlin DSL Settings
Edit File Templates...
AIDL
Activity
Android Auto
Folder
Fragment
Google
Other
Service
UI Component
Wear
Widget
XML
Resource Bundle
app
Build:
Run tasks

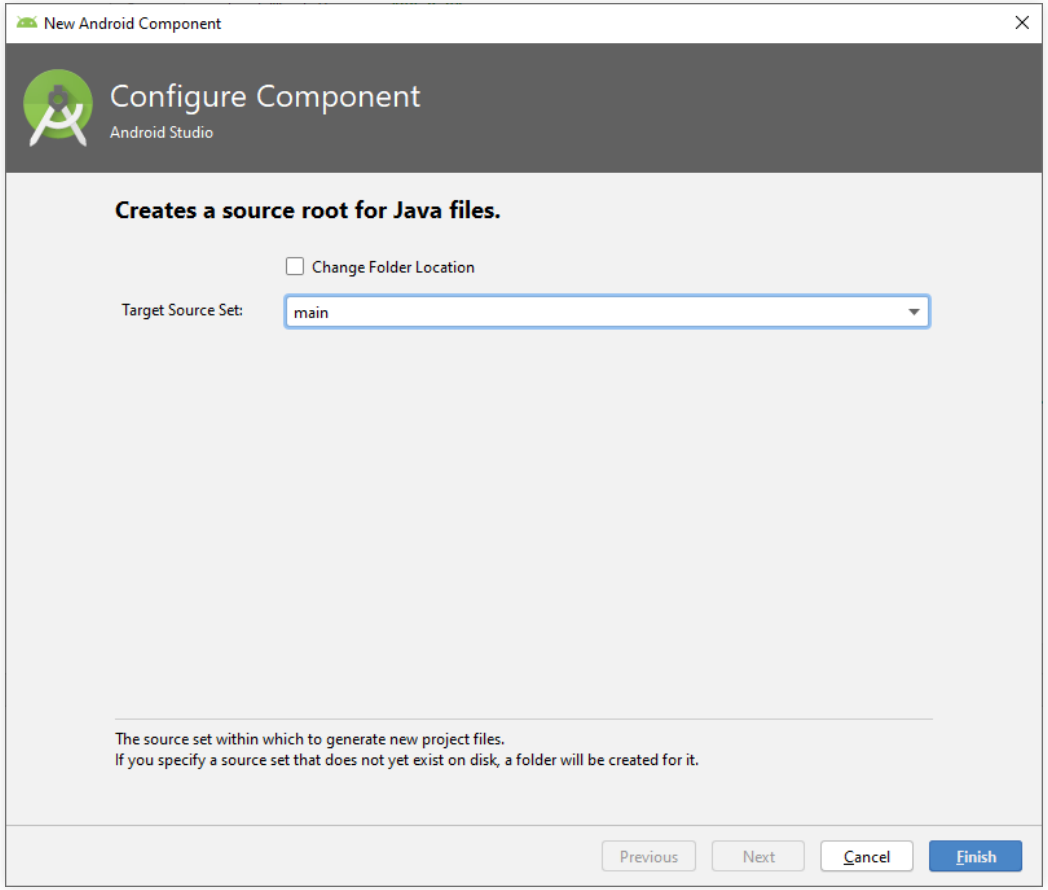
GRADLE


New Resource Directory
Directory name: values
Resource type: values
Source set: main
main
demo
full
debug
release
demoDebug
Available qualifier
Country Code
Network Coc
Locale
Layout Direc
Smallest Screen Width
Screen Width
Screen Height
Size
Ratio
Orientation
UI Mode
Night Mode
Density
>>
<<
Nothing to show
OK
Cancel
Help

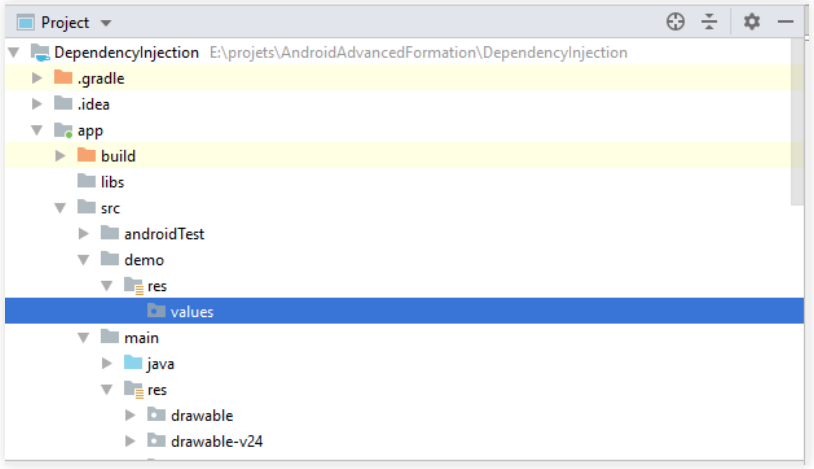
GRADLE


app
New
Link C++ Project with Gradle
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy Relative Path Ctrl+Alt+Shift+C
Paste Ctrl+V
Find Usages Alt+F7
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Run 'All Tests' Ctrl+Shift+F10
Debug 'All Tests'
Run 'All Tests' with Coverage
Create 'All Tests'...
Show in Explorer
Open in Terminal
Local History
Synchronize 'app'
Directory Path Ctrl+Alt+F12
Compare With... Ctrl+D
Open Module Settings F4
Module
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Directory
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Kotlin Script
Gradle Kotlin DSL Build Script
Gradle Kotlin DSL Settings
Edit File Templates...
AIDL
Activity
Android Auto
Folder
Fragment
Google
Other
Service
UI Component
Wear
Widget
XML
Assets Folder
XML Resources Folder
Res Folder
RenderScript Folder
Java Resources Folder
Java Folder
JNI Folder
AIDL Folder
Raw Resources Folder
apply plugin
android {
Build:

GRADLE


New Android Component
Configure Component
Android Studio
Creates a source root for Java files.
Change Folder Location
Target Source Set:
main
The source set within which to generate new project files.
If you specify a source set that does not yet exist on disk, a folder will be created for it.
Previous
Next
Cancel
Finish

GRADLE


Project
DependencyInjection E:\projets\AndroidAdvancedFormation\DependencyInjection
.gradle
.idea
app
build
libs
src
androidTest
demo
res
values
main
java
res
drawable
drawable-v24

GRADLE

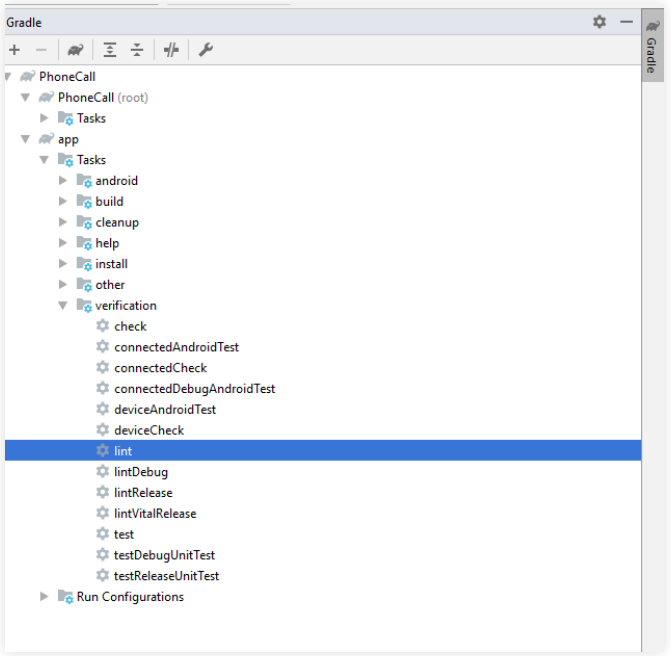
DÉPENDANCES

Nous verrons plus en détails dans la partie ajout des librairies, la gestion des dépendances.

LINT

AMÉLIORER SON CODE SOURCE AVEC LINT

Lancer lint manuellement :

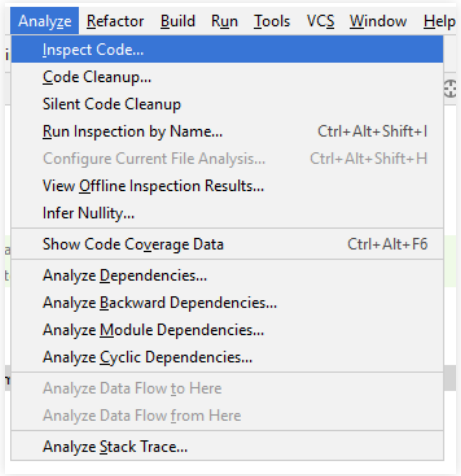


Ouvrez le rapport.

LINT

ANALYSE DU CODE

Nous pouvons aussi lancer l'analyse du code :



LINT

CONFIGURER LINT

Nous avons plusieurs méthodes pour configurer Lint :

- Utiliser l'ampoule jaune ou rouge.
- Créer un fichier `lint.xml` à la racine du projet.
- Utiliser l'annotation `@SuppressLint`.

LINT

EXEMPLE DE FICHIER LINT.XML

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

    <!-- Disable the given check in this project -->
    <issue id="IconMissingDensityFolder" severity="ignore" />

    <!-- Ignore the ObsoleteLayoutParam issue in the specified files -->
    <issue id="ObsoleteLayoutParam">
        <ignore path="res/layout/activation.xml" />
        <ignore path="res/layout-xlarge/activation.xml" />
    </issue>

    <!-- Ignore the UselessLeaf issue in the specified file -->
    <issue id="UselessLeaf">
        <ignore path="res/layout/main.xml" />
```

METTRE AU POINT ET PROFILER/MONITORER UNE APPLICATION.

Nous pouvons étudier la consommation en mémoire, énergie, ... de notre application en utilisant les différents volets du profiler.

- CPU Profiler.
- Memory Profiler.
- Network Profiler.
- Energy Profiler.

Il est aussi possible d'utiliser les outils présents directement sur le téléphone en mode développeur.

OPTIMISATION DE L'APK AVEC PROGUARD.

Nous allons suivre le tutoriel suivant : Getting Started with ProGuard.
Certains ajustement sont à faire :

- Il faut penser à ajouter le repository `maven { url "https://jitpack.io" }` (Page du projet).
- L'erreur `Can't find referenced class org.sl4j` ne semble plus être d'actualité.

CRÉATION D'IHM AVANCÉES

INTERNATIONALISATION

Voyons les fichiers de ressources `strings.xml` et leur utilisation pour internationaliser notre application.

CRÉATION D'IHM AVANCÉES

CONSTRUCTION D'IHM AVANCÉES SUIVANT LES PRÉCONISATIONS MATERIAL DESIGN.

La base du design de nos applications est décrite ici : Material Design
Nous allons voir un peu plus en détails certains éléments.

- Couleurs.
- Buttons.
- Champs texte.
- TextView ajustable
- Résumé des bonnes pratiques

CRÉATION D'IHM AVANCÉES

BOUTONS

Nous allons déclarer un bouton de type `com.google.android.material.button.MaterialButton`, dans notre interface.

CRÉATION D'IHM AVANCÉES

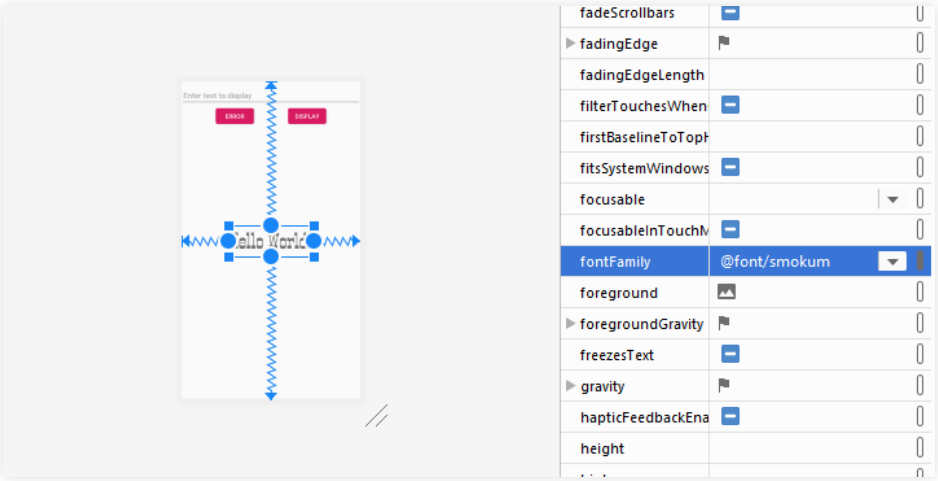
CHAMPS TEXTE

Ajoutons un champ texte en utilisant le materiel design. On pourra s'aider du LiveTemplate `android_xml_material_design_edittext`.

CRÉATION D'IHM AVANCÉES

FONTS EXTERNES

Nous allons utiliser une font externe pour afficher un texte. Pour cela sélectionnez un `TextView`, `All Attributes`, `fontFamilly`, et sélectionnez par exemple `smokum`.



CRÉATION D'IHM AVANCÉES

SEEKBAR

Ajoutons une barre pour sélectionner la taille de notre texte : une `SeekBar`.

CRÉATION D'IHM AVANCÉES

TEXTVIEW AUTOMATIQUE

Ajoutons maintenant un listener sur notre champ texte qui va afficher le texte avec la font, et la taille sera automatique, pour occuper le plus de place possible sur l'écran.

CRÉATION D'IHM AVANCÉES

BOUTON FLOTTANT

Ajoutons un bouton flottant : Référence

CRÉATION D'IHM AVANCÉES

UTILISATION DES STYLES

Nous allons voir comment bien organiser la présentation de ses écrans, à travers les fichiers ressources :

- `colors.xml`
- `dimens.xml`
- `styles.xml`

CRÉATION D'IHM AVANCÉES

DÉCLARATION DES COULEURS

Nous allons déclarer des couleurs dans le fichiers `colors.xml` afin de regrouper ce type de ressources.

CRÉATION D'IHM AVANCÉES

DÉCLARATION DES TAILLES

Le fichier `dimens.xml` nous permet de déclarer les différentes tailles (taille du texte, marge, ...) afin de regrouper ce type de ressources.
Les différentes unités de mesure :

- dp
- sp
- px, in, ...

CRÉATION D'IHM AVANCÉES

DÉCLARATION DES STYLES

Nous allons déclarer des styles, qui feront référence aux couleurs et tailles, définies dans les autres fichiers de ressources, et regrouperons toutes les caractéristiques d'affichage des éléments. Il est même possible de faire des héritages de styles, comme en CSS.

```
<style name="DialogTitle" parent="AppTheme">
    <item name="android:padding">@dimen/dialog_title_text_padding</item>
    <item name="android:textSize">@dimen/dialog_title_text_size</item>
    <item name="android:textStyle">bold</item>
</style>
```

CRÉATION D'IHM AVANCÉES

ALLER PLUS LOIN

Nous allons voir plus en détails les ConstraintLayouts, voir le PDF ici.
Les ressources graphiques sont disponibles ici.

Nous pouvons suivre les tutoriels suivants pour perfectionner notre technique :

- MDC-101 Android: Material Components (MDC) Basics (Java)
- MDC-102 Android: Material Structure and Layout (Java)
- MDC-103 Android: Material theming with Color, Motion and Type (Java)
- MDC-104 Android: Material Advanced Components (Java)

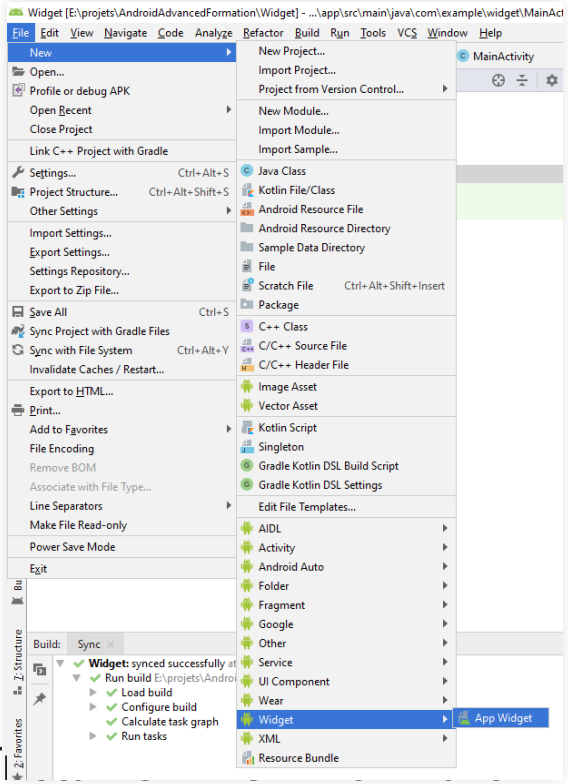
CRÉATION D'IHM AVANCÉES

MÉCANISMES DES WIDGETS

Nous allons développer un widget qui permet d'afficher la date courante.
Créons un nouveau projet `Widget`.

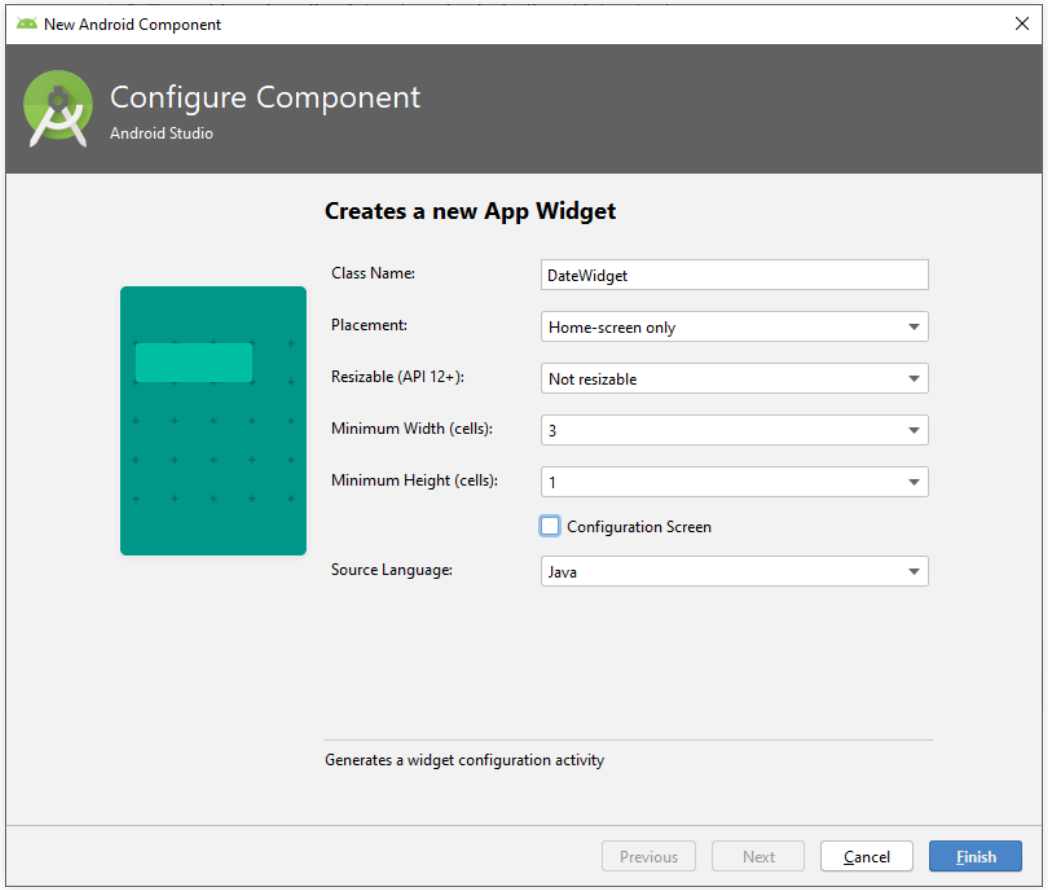
CRÉATION D'IHM AVANCÉES

Ajoutons un widget :



CRÉATION D'IHM AVANCÉES

Paramétrons le template :



CRÉATION D'IHM AVANCÉES

Remplaçons :

```
1 CharSequence widgetText = context.getString(R.string.appwidget_text);
```

Par :

```
1 Date current = new Date();
2 SimpleDateFormat dateFormat = new SimpleDateFormat("dd MMM yyy");
3 String widgetText = dateFormat.format(current);
```

CRÉATION D'IHM AVANCÉES

PRÉSENTATION OPENGL/ES.

Pour réaliser des applications graphiques, 3D, ... nous pouvons utiliser OpenGl.
Pour plus de détails, un tutoriel d'initiation en français, et un exemple un peu plus complexe en anglais.

CRÉATION D'IHM AVANCÉES

AJOUT D'ANIMATIONS

Nous pouvons ajouter facilement une animation à nos layouts, pour cela, créons le projet suivant :

CRÉATION D'IHM AVANCÉES

Définissons l'attribut :

```
1 TextView tvText;
```

Définissons le code du bouton :

```
1 tvText = findViewById(R.id.text);
2
3 findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
4     @Override
5     public void onClick(View v) {
6         tvText.setVisibility(tvText.getVisibility() == View.VISIBLE ? View.GONE : View.VISIBLE );
7     }
8 });
```

Testons.

CRÉATION D'IHM AVANCÉES

Ajoutons un attribut dans le layout parent :

```
1 android:animateLayoutChanges="true"
```

Testons de nouveau.

CRÉATION D'IHM AVANCÉES

AJOUT D'UN FOND EN DÉGRADÉ

Créons un drawable : `background.xml` :

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient
            android:angle="90"
            android:endColor="#FFF"
            android:startColor="#CCC" />
</shape>
```

Ajoutons le en background de notre layout principal.

CRÉATION D'IHM AVANCÉES

TRANSITION ENTRE LES ACTIVITÉS

Il est possible d'ajouter des animations lors du lancement d'une activité, mais il est encore plus parlant pour l'utilisateur d'avoir des animations faisant correspondre les éléments d'un écran, vers le second. Voyons cela en détail ici.

CRÉATION D'IHM AVANCÉES

HAUT NIVEAU D'ANIMATION

Un exemple d'interface animée : Application de pizza.

CRÉATION D'IHM AVANCÉES

`tools`

Dans un layout, il est possible de changer le prefix `android` par le préfix `tools` ce qui impactera seulement l'affichage dans l'IDE, mais sera ignoré dans l'application. Testons par exemple en changeant la visibilité d'un élément :

```
1 tools:visibility="gone"
```

LIBRAIRIES ET SERVICES

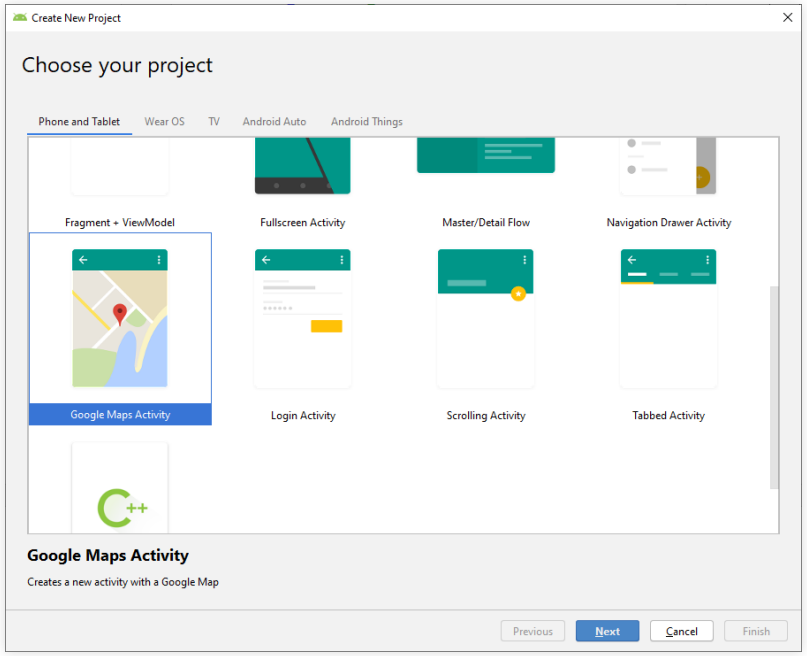
Nous allons voir plusieurs services proposés par la librairie Google Play Services :

- Google Maps
- Géolocalisation
- LeaderBoard

LIBRAIRIES ET SERVICES

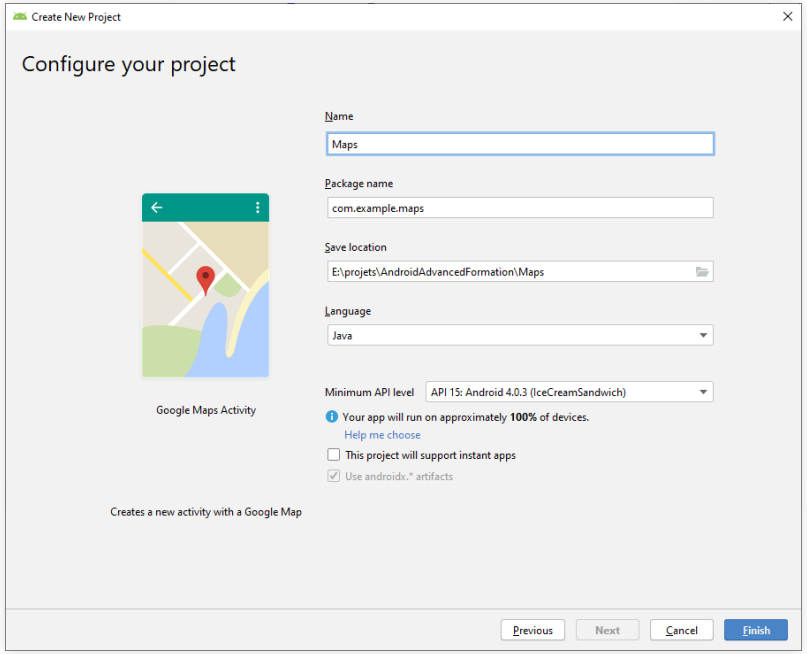
UTILISER LES GOOGLE PLAY SERVICES.

Nous allons par exemple mettre en place une Google Maps.



LIBRAIRIES ET SERVICES

Donnez un nom à l'application :



LIBRAIRIES ET SERVICES

Suivez les instructions pour obtenir une clé API :

LIBRAIRIES ET SERVICES

Modifiez le code pour afficher un point sur notre localisation actuelle (la récupérer sur Google Maps par exemple), exemple d'usage :

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    // Add a marker in Startup Marseille and move the camera
    LatLng startupMarseille = new LatLng(43.291590, 5.378210);
    mMap.addMarker(new MarkerOptions().position(startupMarseille).title("Marker in Startup Marseille"));
    mMap.moveCamera(CameraUpdateFactory.zoomTo(19));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(startupMarseille));

}
```

LIBRAIRIES ET SERVICES

AFFICHAGE POSITION COURANTE

Nous allons maintenant afficher la position courante de l'utilisateur en utilisant le GPS. Pour cela nous allons suivre les étapes suivantes :
Ajouter la librairie à notre graddle :

```
1 implementation 'com.google.android.gms:play-services-location:17.0.0'
```

Définir un attribut de type `Marker` pour garder une référence sur le marker de la position courante :

```
1 private Marker currentPosition;
```

LIBRAIRIES ET SERVICES

Récupérer la position du smartphone, pour pouvoir l'afficher :

```
1 FusedLocationProviderClient client = LocationServices.getFusedLocationProviderClient(this);
2
3 // Get the last known location
4 client.getLastLocation().addOnCompleteListener(this, new OnCompleteListener<Location>() {
5     @Override
6     public void onComplete(@NonNull Task<Location> task) {
7         // TODO
8     }
9 });
```

LIBRAIRIES ET SERVICES

AFFICHAGE DU MARKER

Complétons le code :

```
1 LatLng currentLocation = new LatLng(task.getResult().getLatitude(), task.getResult().getLongitude());
2 currentPosition = mMap.addMarker(new MarkerOptions().position(currentLocation).title("Current position").icon(BitmapDescr
```

LIBRAIRIES ET SERVICES

GESTION DES AUTORISATIONS

Nous avons l'erreur suivante :

```
1 com.google.android.gms.tasks.RuntimeExecutionException: java.lang.SecurityException: Client must have ACCESS_COARSE_LOCAT
```

Comment la corriger ?

- Activer directement dans les paramètres de l'application (pour tester).
- Demander la permission dynamiquement.

LIBRAIRIES ET SERVICES

ERREUR DE POSITION SUR L'ÉMULATEUR

Si vous rencontrez un problème avec la localisation, lancez l'application Maps, et le problème devrait être corrigé.

LIBRAIRIES ET SERVICES

LEADERBOARD

Nous pouvons utiliser le système de gestion des scores, des accomplissements proposé par Google, tous les détails : ICI.

LIBRAIRIES ET SERVICES

RÉCAPITULATIF DES SERVICES

La liste des services disponibles est accessible ICI et ICI.

LIBRAIRIES ET SERVICES

INTÉGRER DES BIBLIOTHÈQUES TIERCES À UN PROJET ANDROID.

Nous allons installer et utiliser une bibliothèque tierce dans un projet. Par exemple un sélecteur de couleur.
Suivez les consignes de la librairie, avec les instructions supplémentaires suivantes :

- Déclenchez le sélecteur sur le click d'un bouton.
- Nous modifierons la couleur de fond de l'application.
- La couleur initiale doit être la couleur du fond.

Récupération du code couleur du fond de l'application :

```
1 int color = Color.RED;
2 Drawable background = rootView.getBackground();
3 if (background instanceof ColorDrawable)
4     color = ((ColorDrawable) background).getColor();
```

LIBRAIRIES ET SERVICES

INTÉGRATION D'UN SCANNER DE QRCODE

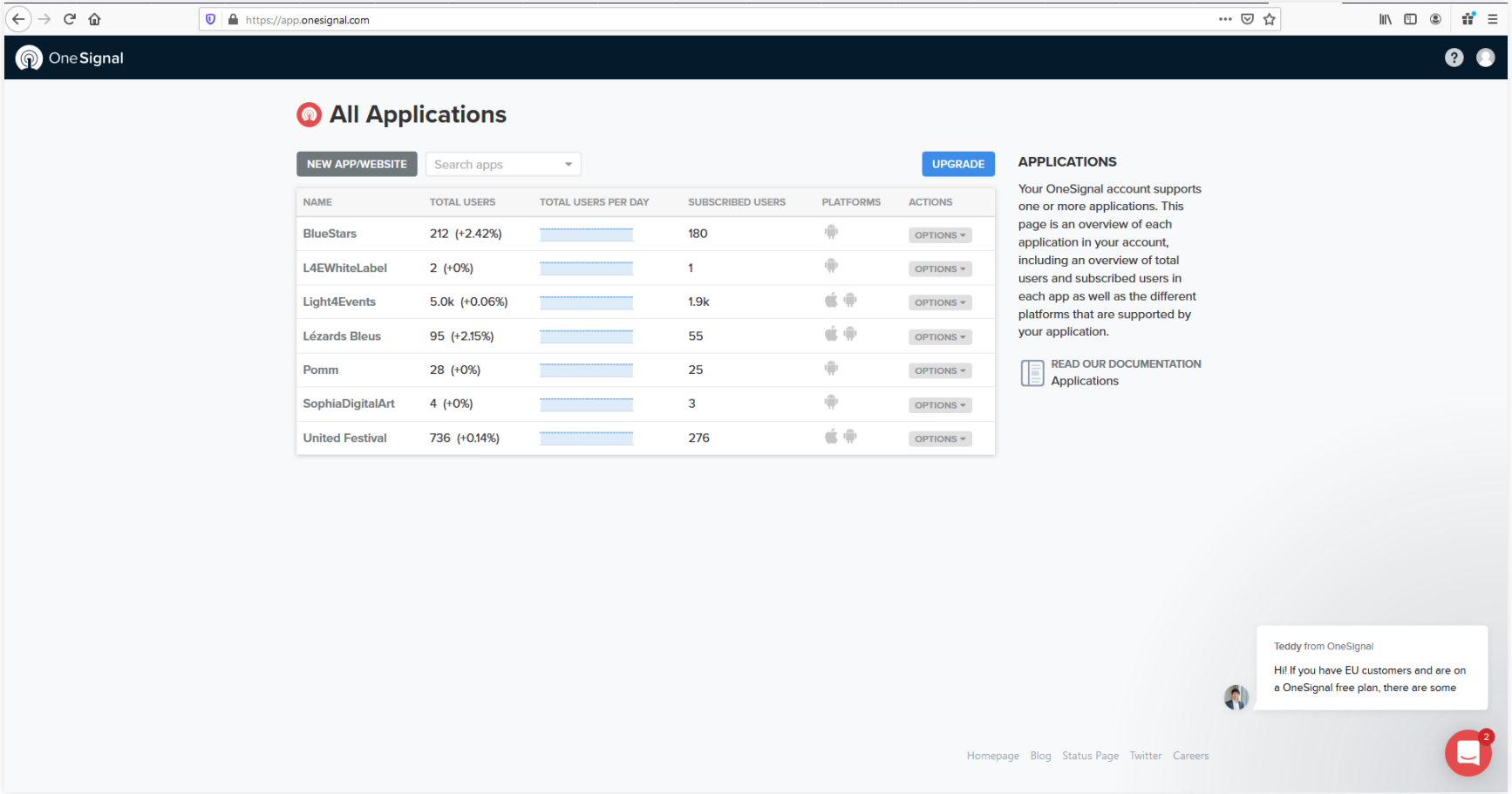
Voir le PDF ici.

LIBRAIRIES ET SERVICES

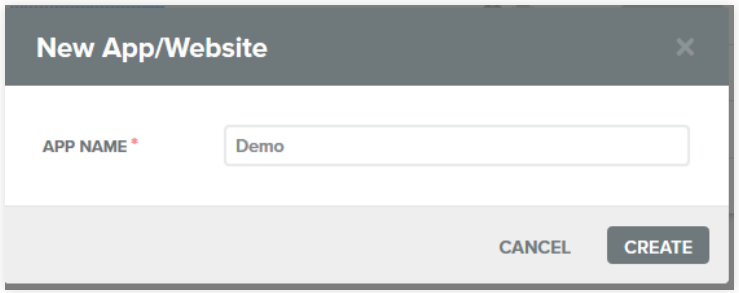
INTÉGRATION D'UNE PUSH NOTIFICATION

Nous allons utiliser les services de OneSignal, pour cela nous allons créer un compte sur OneSignal, et suivre les étapes suivantes ...

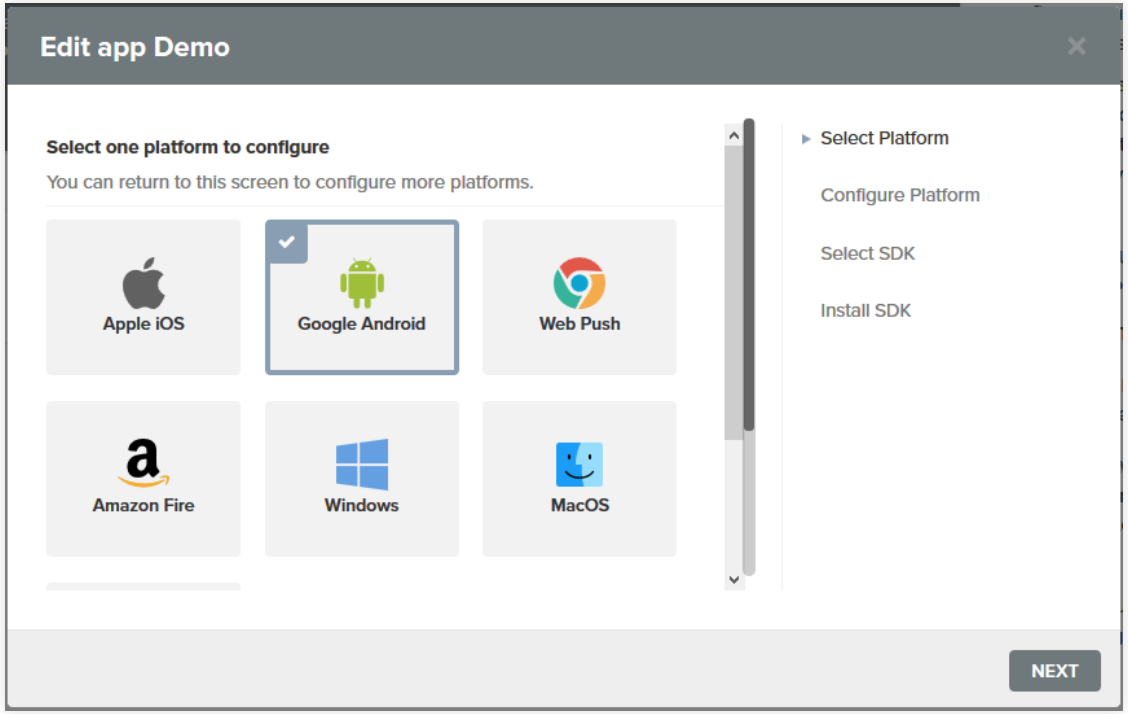
LIBRAIRIES ET SERVICES



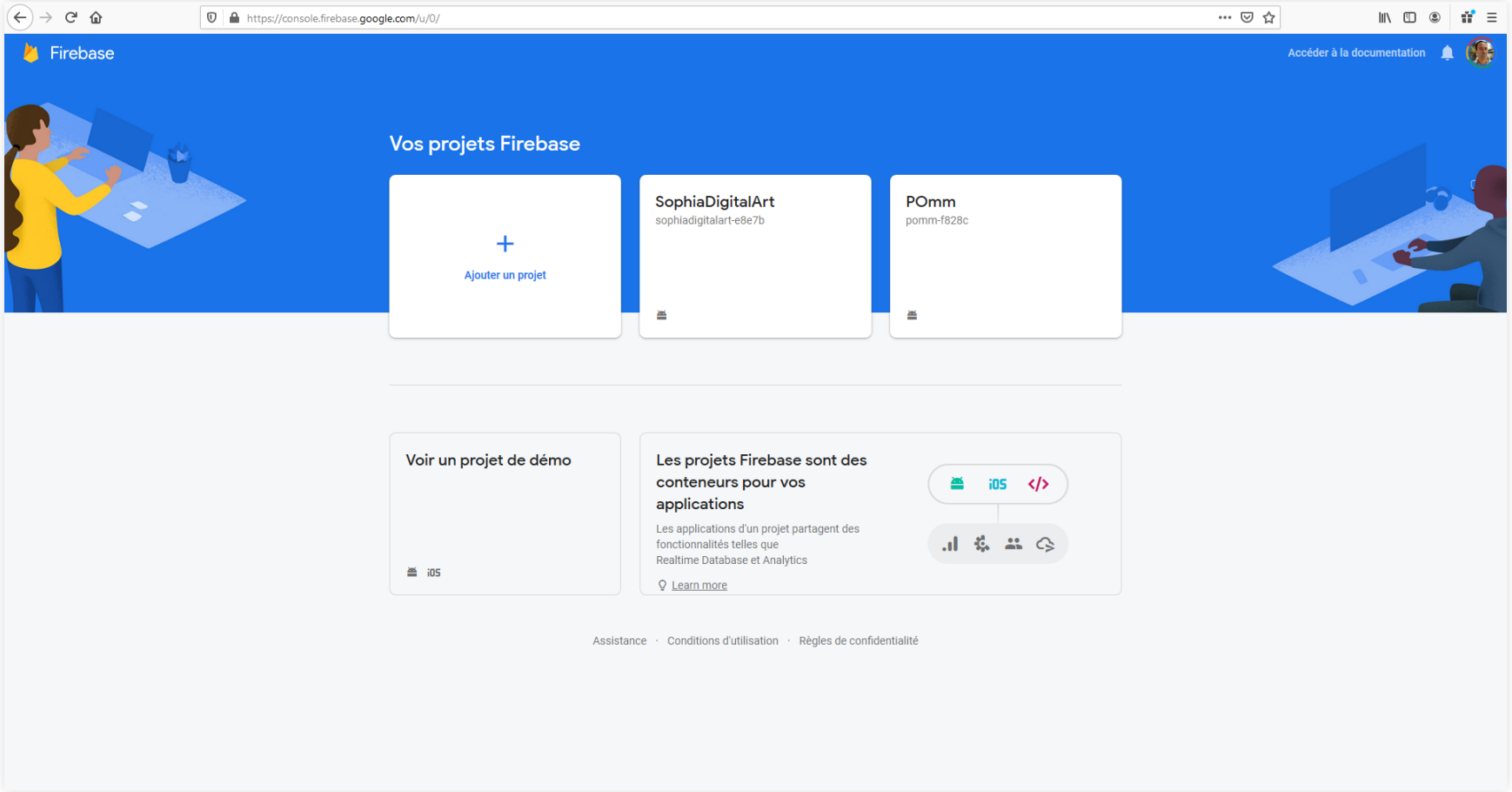
LIBRAIRIES ET SERVICES



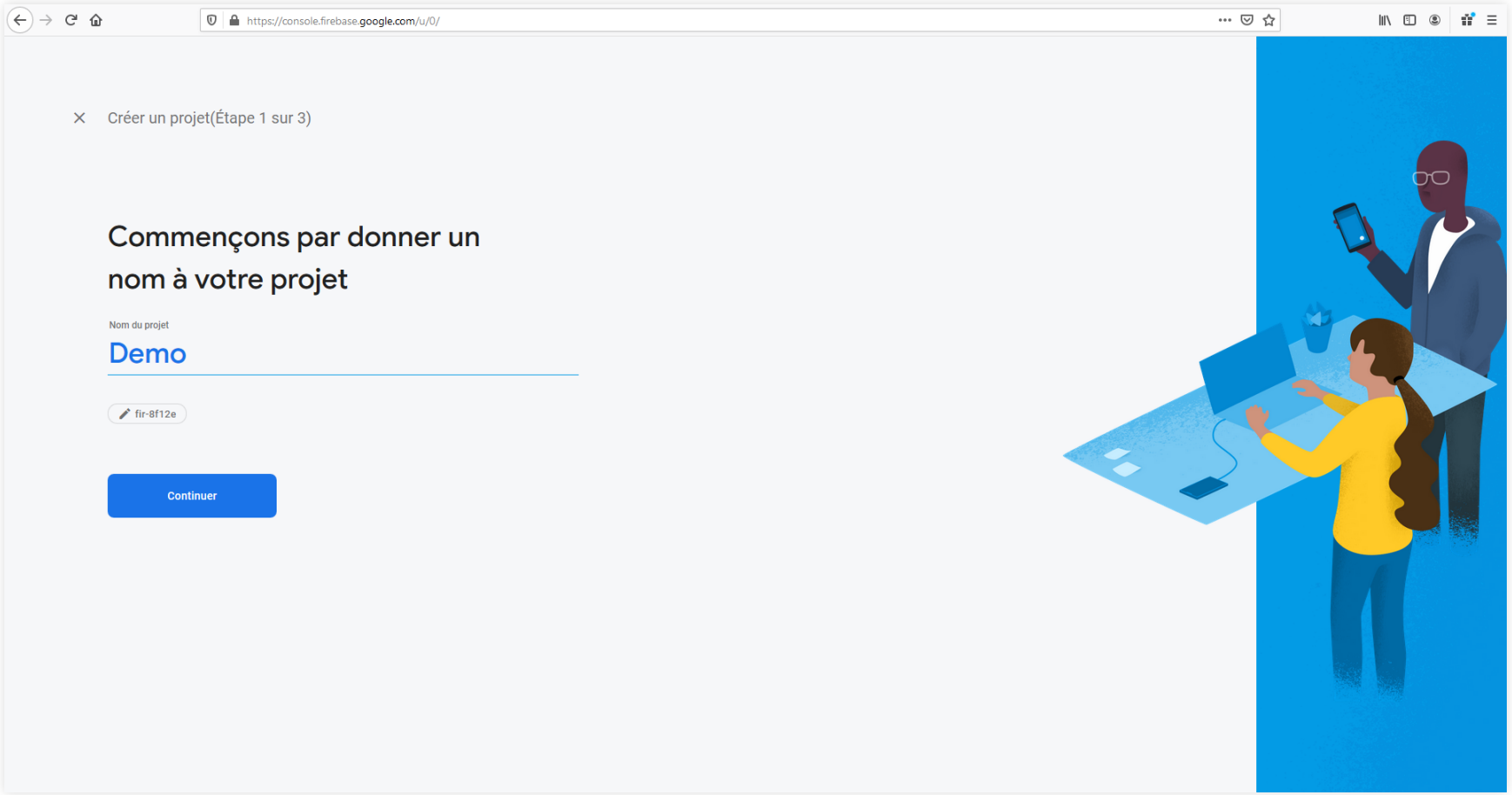
LIBRAIRIES ET SERVICES



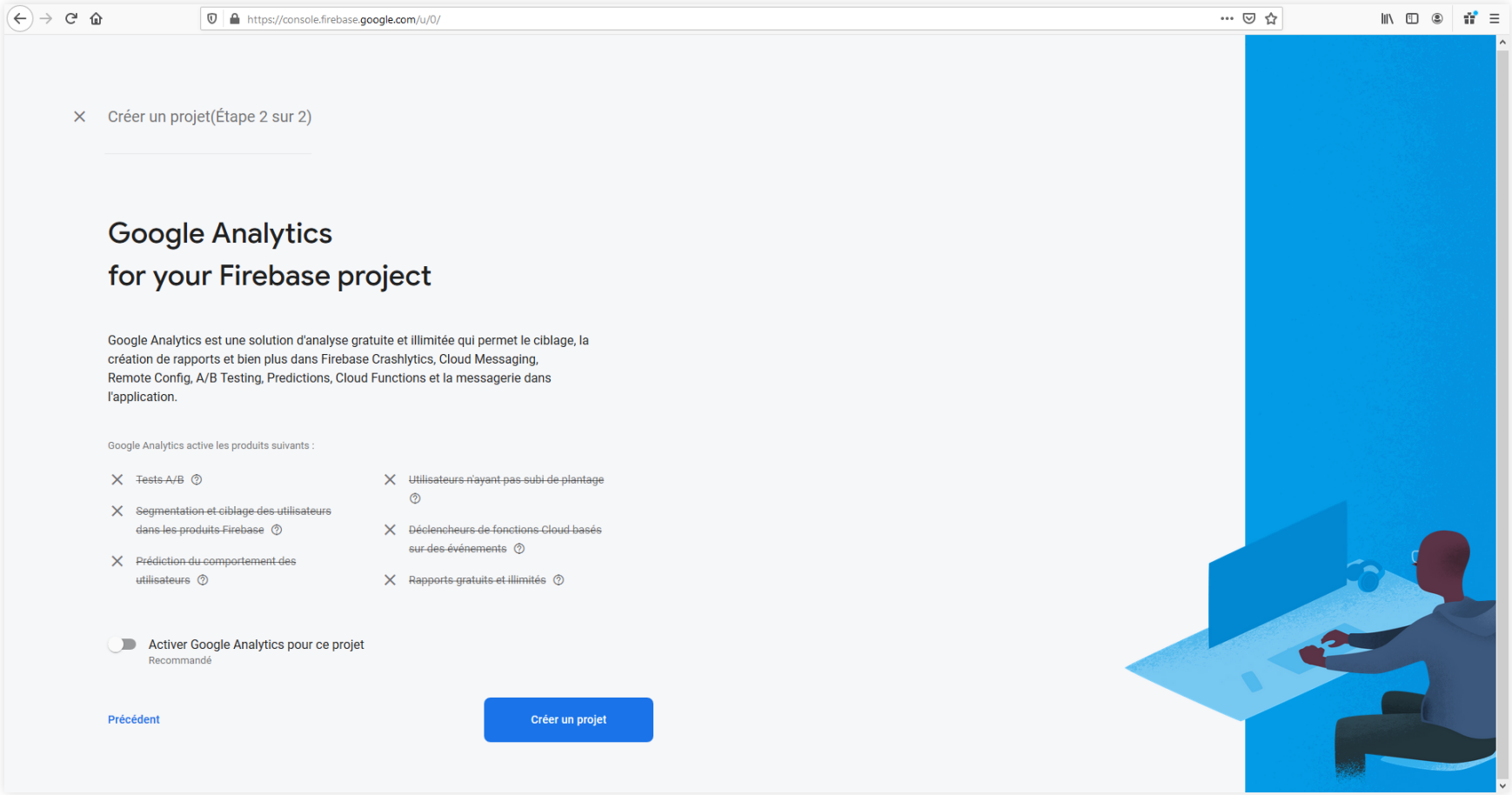
LIBRAIRIES ET SERVICES



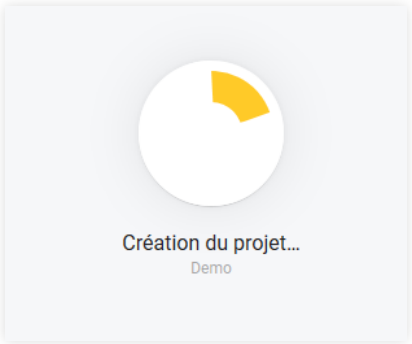
LIBRAIRIES ET SERVICES



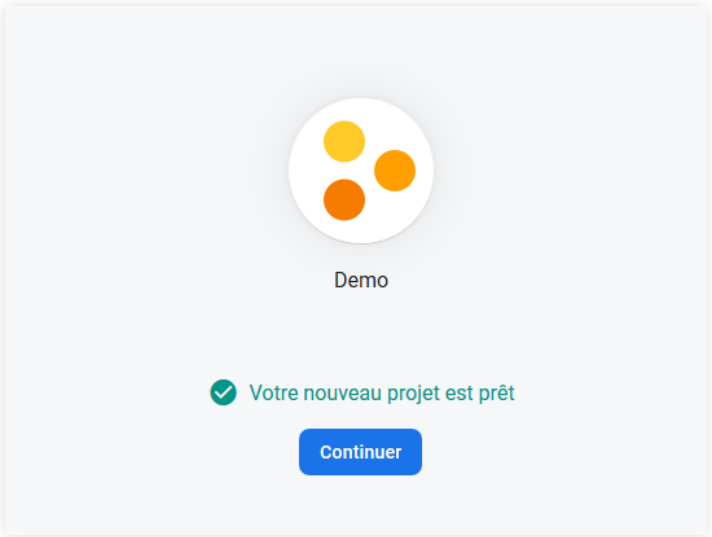
LIBRAIRIES ET SERVICES



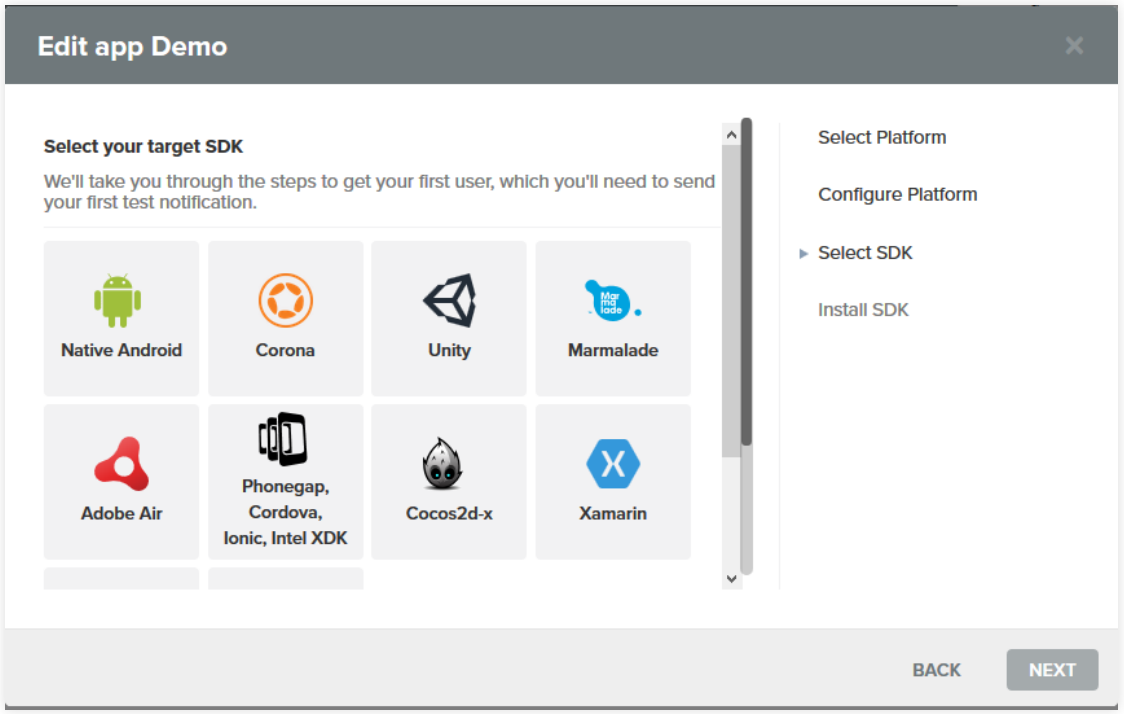
LIBRAIRIES ET SERVICES



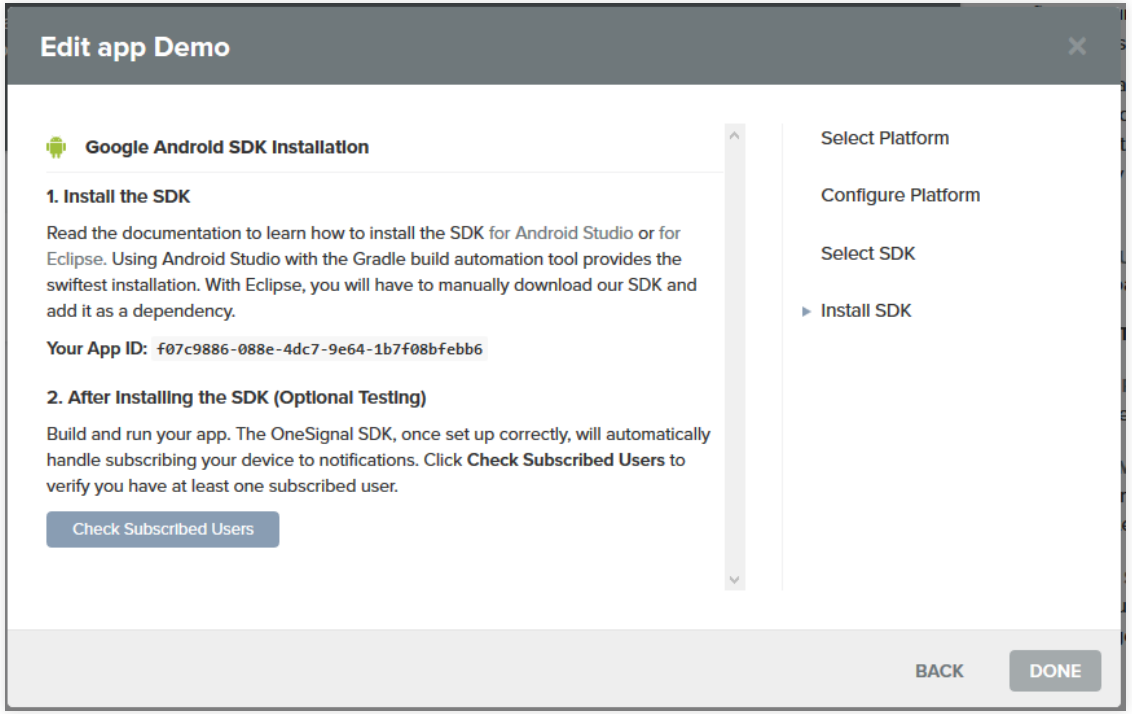
LIBRAIRIES ET SERVICES



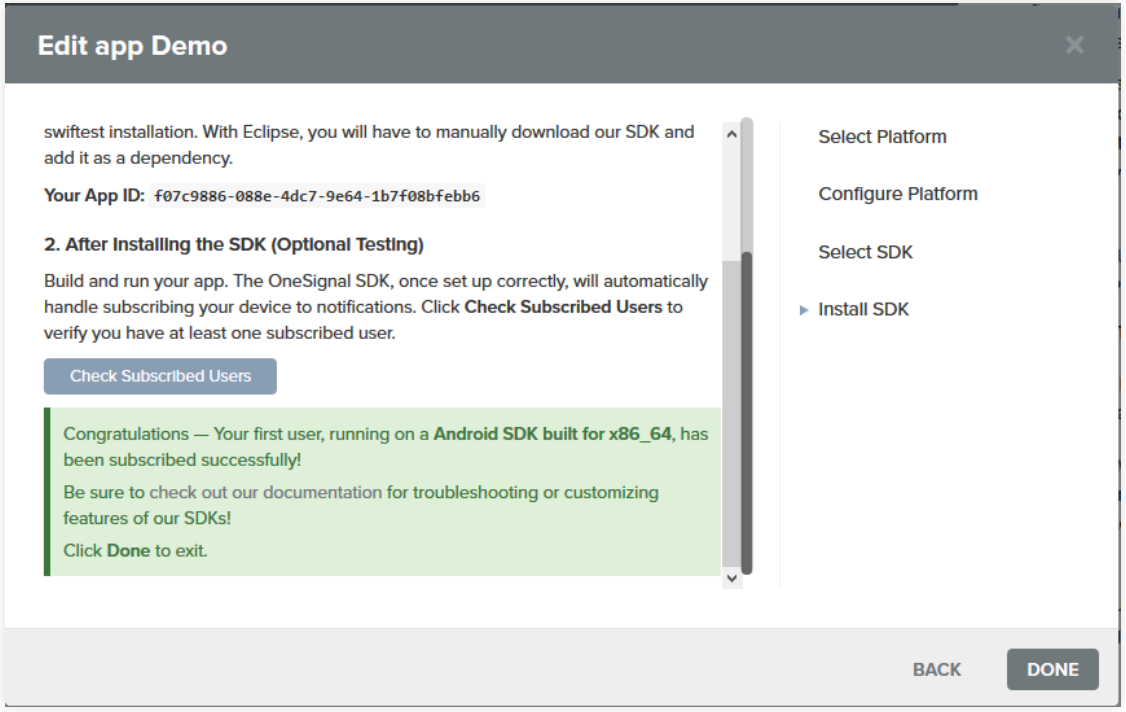
LIBRAIRIES ET SERVICES



LIBRAIRIES ET SERVICES



LIBRAIRIES ET SERVICES



LIBRAIRIES ET SERVICES

SIMPLIFIER L'ACCÈS À DES RESSOURCES REST AVEC RETROFIT.

Nous allons suivre le tutoriel suivant : Retrofit2 – webservices REST.

LIBRAIRIES ET SERVICES

MAÎTRISER LE CHARGEMENT DES IMAGES AVEC PICASSO.

Nous allons intégrer la librairie Picasso à notre projet.

- Ajoutons une `ImageView` à notre projet.
- Ajoutons la librairie dans le gradle.
- Ajoutons la permission INTERNET.
- Utilisons la librairie comme indiqué dans la documentation.

LIBRAIRIES ET SERVICES

Ce qui nous donnera le code suivant (avec debug activé) :

```
1 ImageView imageView = findViewById(R.id.image);
2 Picasso.get().setLoggingEnabled(true);
3 Picasso.get().setIndicatorsEnabled(true);
4 Picasso.get().load("https://upload.wikimedia.org/wikipedia/commons/d/da/Internet2.jpg").into(imageView);
```

LIBRAIRIES ET SERVICES

GLIDE

Procédons de la même manière avec la librairie Glide.
Ce qui nous donnera le code suivant :

```
1 Glide.with(this).load("https://upload.wikimedia.org/wikipedia/commons/d/da/Internet2.jpg").into(imageView);
```

LIBRAIRIES ET SERVICES

L'INJECTION DE DÉPENDANCES (DAGGER).

Nous allons mettre en place une librairie permettant l'injection de dépendances : Dagger.

Dans un premier temps, pour bien comprendre l'injection de dépendances, nous allonssuivre le tutoriel suivant :
https://developer.android.com/training/dependency-injection.

Dans un second temps nous allons suivre le tutoriel suivant :
https://www.supinfo.com/articles/single/2972-dagger-2-android-guide-etre-rapidement-pieds.

CUSTOM VIEW

DÉFINITION

Nous pouvons définir des vues que nous allons dessiner sur un `Canvas`. Pour cela nous allons suivre les étapes suivantes :

- Création d'une custom view à partir du template.
- Personnalisation du modèle.

CUSTOM VIEW

CRÉATION DU MODÈLE

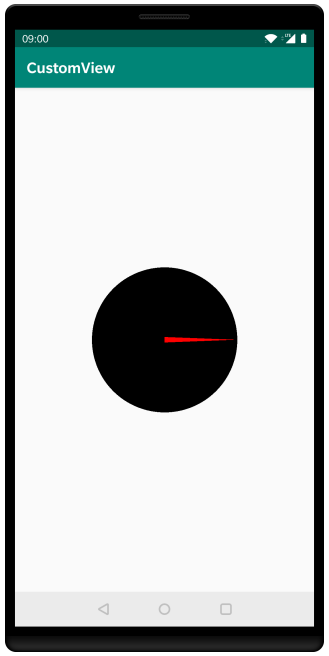
Pour créér le modèle, nous allons utiliser le menu :

`File / New / Ui Component / Custom View`

CUSTOM VIEW

LA VUE À OBTENIR

Nous souhaitons obtenir la vue suivante :



CUSTOM VIEW

EXERCICE

Écrivez le code qui permet de dessiner la vue.
Nous utiliserons les éléments suivants :

- `Paint`.
- `translate`.
- `rotate`.
- `setColor`.
- `Path`.
- `drawArc`.

UTILISATION DES CAPTEURS

MISE EN ŒUVRE DE CAPTEURS.

Nous allons développer une application qui va éteindre l'écran quand on est proche de l'écran, et le ralumer quand on est à distance. Pour cela plusieurs étapes :

- Nous allons lister les capteurs.
- Récupérer le capteur de proximité.
- S'enregistrer sur les changements.
- Réagir aux changements.

UTILISATION DES CAPTEURS

LISTE DES CAPTEURS.

Pour lister tous les capteurs disponibles sur notre smartphone, utilisons le code suivant :

```
1 SensorManager sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
2 List<Sensor> liste = sensorManager.getSensorList(Sensor.TYPE_ALL);
3 if (BuildConfig.DEBUG) {
4     for (Sensor currentSensor : liste) {
5         Log.d(TAG, "onCreate sensor " + currentSensor);
6     }
7 }
```

UTILISATION DES CAPTEURS

RÉCUPÉRONS LE CAPTEUR DE PROXIMITÉ

```
1 private Sensor mProximitySensor = null;
2 mProximitySensor = sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
```

UTILISATION DES CAPTEURS

ENREGISTRONS NOUS, SUR LES CHANGEMENTS

Dans un premier temps nous récupérons le manager des capteurs :

```
1 private SensorManager mSensorManager = null;
2 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

Puis nous nous enregistrons sur les changements uniquement quand l'application est active :

```
 1 @Override
 2 protected void onResume() {
 3     super.onResume();
 4     mSensorManager.registerListener(mSensorEventListener, mProximitySensor, SensorManager.SENSOR_DELAY_NORMAL);
 5 }
 6
 7 @Override
 8 protected void onPause() {
 9     super.onPause();
10     mSensorManager.unregisterListener(mSensorEventListener, mProximitySensor);
11 }
```

UTILISATION DES CAPTEURS

LE LISTENER

Le listener qui va réagir aux changements est définis par :

```
1  final SensorEventListener mSensorEventListener = new SensorEventListener() {
2      public void onAccuracyChanged(Sensor sensor, int accuracy) {
3          // Que faire en cas de changement de précision ?
4      }
5
6      public void onSensorChanged(SensorEvent sensorEvent) {
7          if(BuildConfig.DEBUG){
8              Log.d(TAG, "onSensorChanged " + sensorEvent.values.length);
9              Log.d(TAG, "onSensorChanged " + sensorEvent.values[0]);
10         }
11     }
12 };
```

Il ne nous reste plus qu'à implémenter notre code fonctionnel.
Exemple plus poussé d'utilisation du capteur de proximité pour éteindre l'écran.

UTILISATION DES CAPTEURS

EXERCICE

Utiliser le capteur magnétique pour afficher le nord, en utilisant la vue custom précédemment développée.

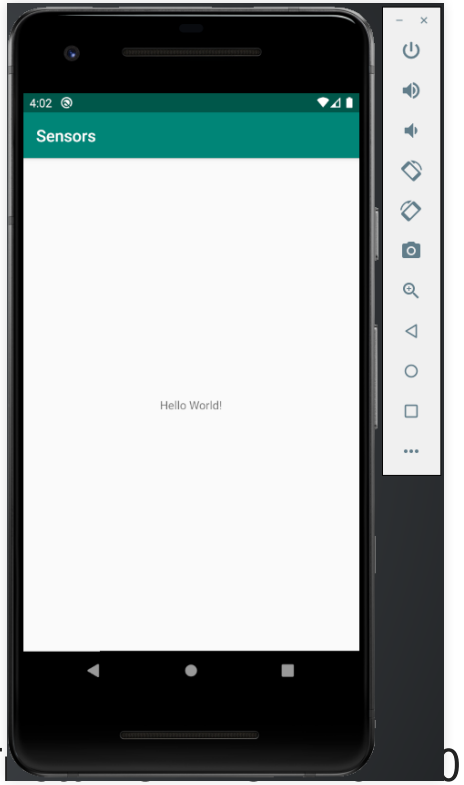
UTILISATION DES CAPTEURS

EXEMPLE COMPLET DE JEUX

Un exemple plus complet de jeux utilisant les capteurs est disponible ici.
Un exemple reprenant la base de ce que l'on à vu ici.
Un exemple simple pour afficher les valeurs de l'accéléromètre ici.

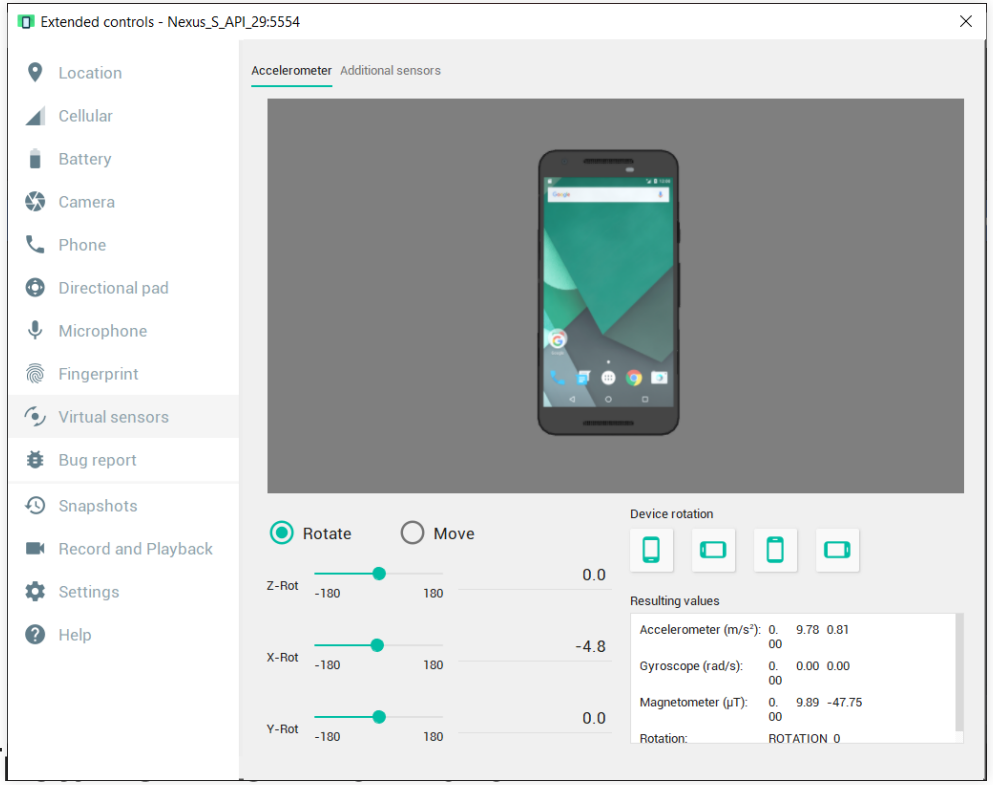
UTILISATION DES CAPTEURS

PARAMÉTRAGE DANS LE SIMULATEUR DES CAPTEURS.



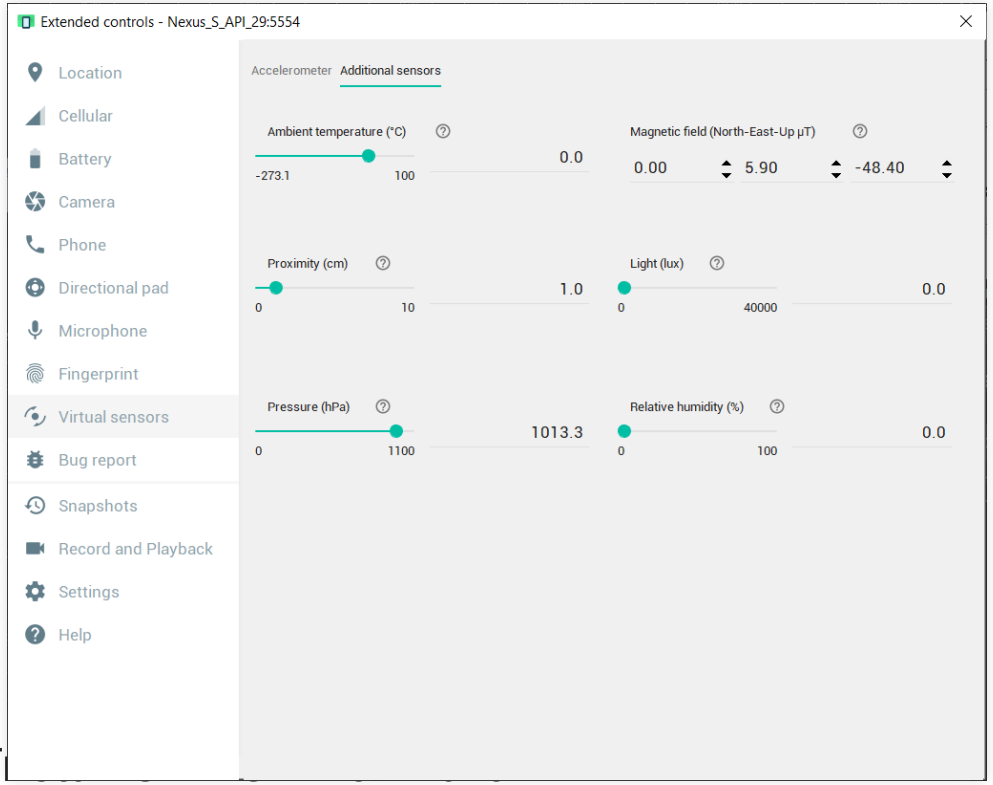
UTILISATION DES CAPTEURS

PARAMÉTRAGE DANS LE SIMULATEUR DES CAPTEURS.



UTILISATION DES CAPTEURS

PARAMÉTRAGE DANS LE SIMULATEUR DES CAPTEURS.



CONTENTPROVIDER ET SERVICES

DÉFINITION

Le content provider permet de partager avec d'autres applications des données, de manière standardisée.

CONTENTPROVIDER ET SERVICES

UTILISATION D'UN CONTENT PROVIDER

Nous allons utiliser un content provider, par exemple le MediaStore. Créons un nouveau projet ContentProviderClient, et listons les sons disponibles sur notre téléphone :

```
1  Uri media = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
2  String[] projection = { MediaStore.Audio.Media._ID,           // 0
3          MediaStore.Audio.Media.ARTIST,         // 1
4          MediaStore.Audio.Media.TITLE,          // 2
5          MediaStore.Audio.Media.ALBUM_ID,       // 3
6          MediaStore.Audio.Media.ALBUM,          // 4
7          MediaStore.Audio.Media.DATA,           // 5
8          MediaStore.Audio.Media.DISPLAY_NAME,   // 6
9          MediaStore.Audio.Media.DURATION };     // 7
10
11 String selection = MediaStore.Audio.Media.IS_MUSIC + " != 0";
12
13 Cursor cursor = getContentResolver().query(media,projection,selection,null,null);
14 while(cursor.moveToNext()){
15     if(BuildConfig.DEBUG){
```

CONTENTPROVIDER ET SERVICES

UN AUTRE CONTENT PROVIDER

Nous allons maintenant utiliser le content provider du dictionnaire afin de nous familiariser avec les actions disponibles (CRUD : Create Read, Update et Delete). Pour cela nous allons passer par plusieurs étapes :

- Créer un AVD en API 22.
- Ajouter des entrées au dictionnaire.
- Utiliser le projet ContentProviderClient.
- Demander la permission.
- Interagir avec le content provider.

CONTENTPROVIDER ET SERVICES

CRÉER UN AVD EN API 22

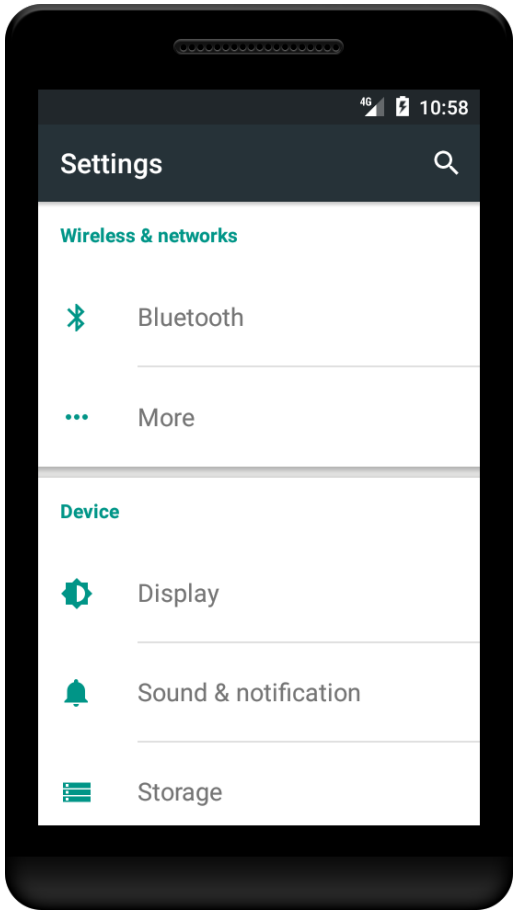
Le content provider du UserDictionary n'étant pas disponible à partir de l'API 23, nous allons créér une AVD en API 22.

CONTENTPROVIDER ET SERVICES

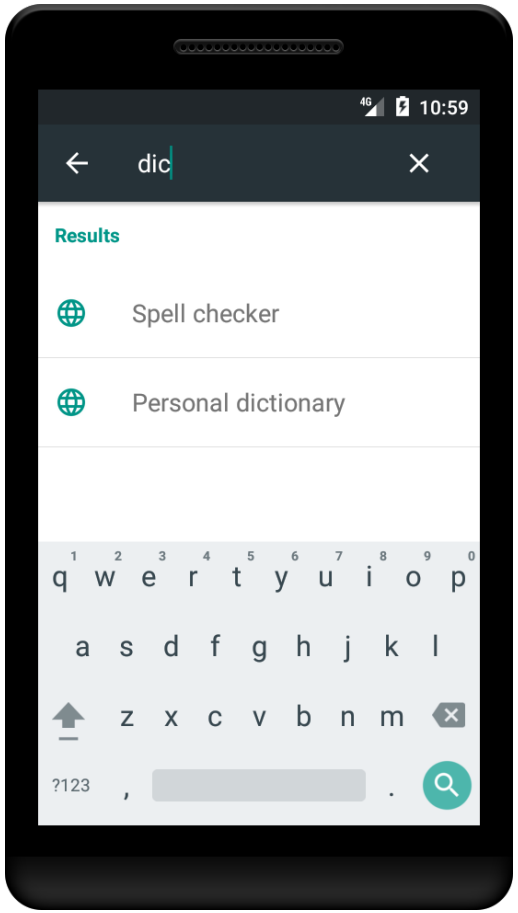
AJOUTER DES ENTRÉES AU DICTIONNAIRE

Afin d'avoir des données à lister, nous allons ajouter des entrées au dictionnaire personnel.

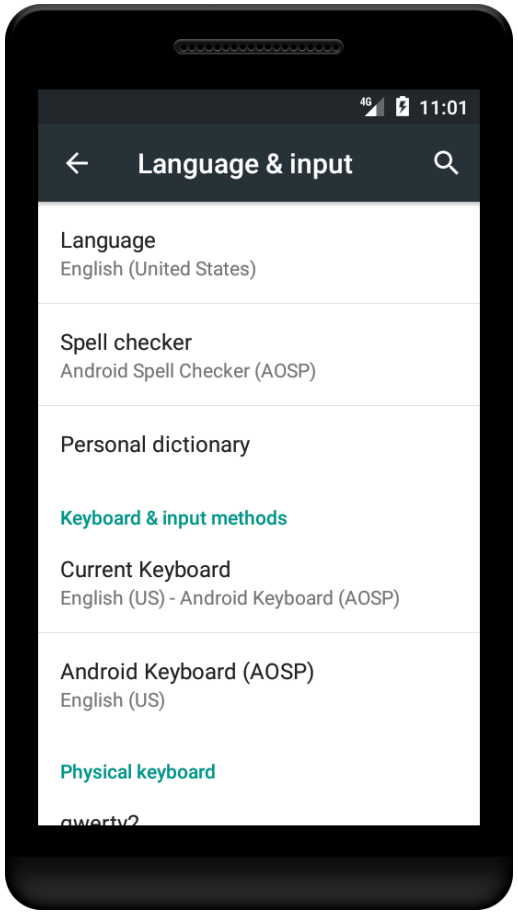
CONTENTPROVIDER ET SERVICES



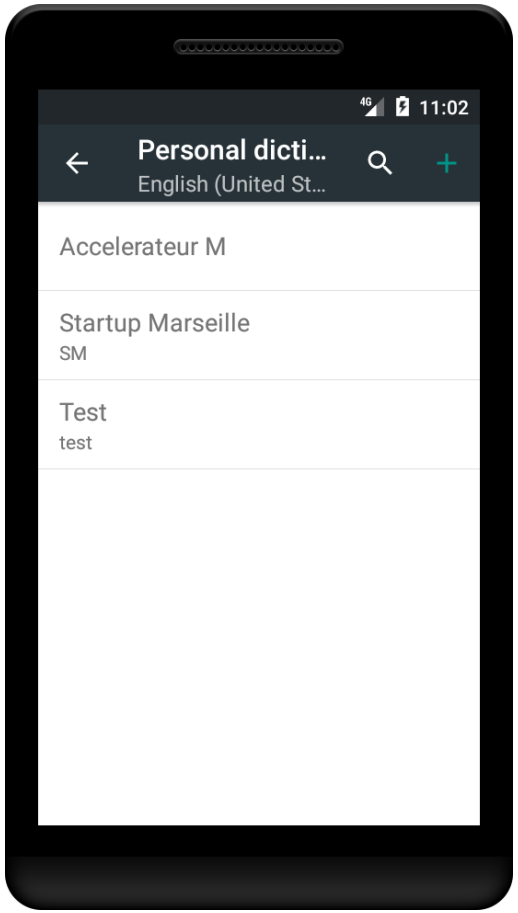
CONTENTPROVIDER ET SERVICES



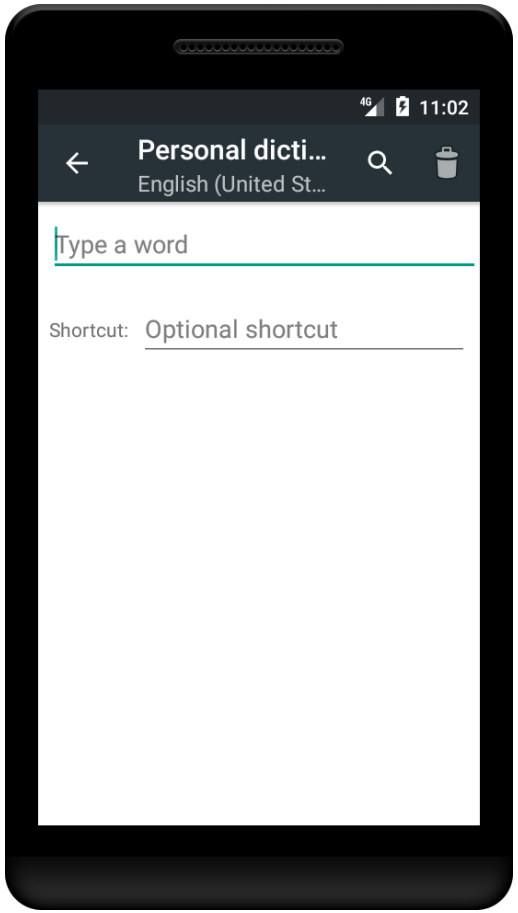
CONTENTPROVIDER ET SERVICES



CONTENTPROVIDER ET SERVICES



CONTENTPROVIDER ET SERVICES



CONTENTPROVIDER ET SERVICES

UTILISER LE PROJET CONTENTPROVIDERCLIENT

Nous allons réutiliser le précédent projet, tout simplement.

CONTENTPROVIDER ET SERVICES

DEMANDER LA PERMISSION

Ajoutons à notre `AndroidManifest.xml` la permission :

```
1 <uses-permission android:name="android.permission.READ_USER_DICTIONARY"/>
```

CONTENTPROVIDER ET SERVICES

INTERAGIR AVEC LE CONTENT PROVIDER

Nous allons pouvoir utiliser le content provider `UserDictionary`.
Commençons par lister les entrées (LiveTemplate android_content_provider_user_dictionary_read) :

```
// A "projection" defines the columns that will be returned for each row
String[] projection = {
        UserDictionary.Words._ID,    // Contract class constant for the _ID column name
        UserDictionary.Words.WORD,   // Contract class constant for the word column name
        UserDictionary.Words.LOCALE  // Contract class constant for the locale column name
};

// Defines a string to contain the selection clause
String selectionClause = null;

// Initializes an array to contain selection arguments
String[] selectionArgs = null;

// Defines a string to order the result
String sortOrder = null;
```

CONTENTPROVIDER ET SERVICES

ECRIRE DANS LE CONTENT PROVIDER

Maintenant écrivons dans le content provider :

```
// Defines a new Uri object that receives the result of the insertion
Uri newUri;

// Defines an object to contain the new values to insert
ContentValues newValues = new ContentValues();

/*
* Sets the values of each column and inserts the word. The arguments to the "put"
* method are "column name" and "value"
*/
newValues.put(UserDictionary.Words.APP_ID, "example.user");
newValues.put(UserDictionary.Words.LOCALE, "en_US");
newValues.put(UserDictionary.Words.WORD, "insert");
newValues.put(UserDictionary.Words.FREQUENCY, "100");
```

CONTENTPROVIDER ET SERVICES

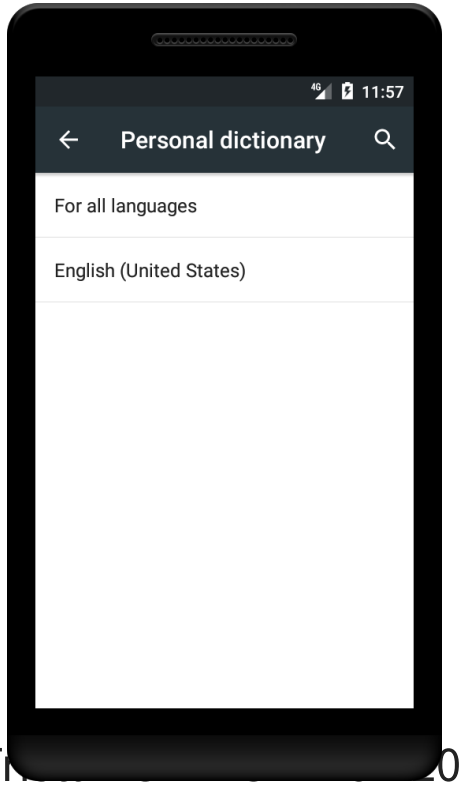
MISE À JOUR

La mise à jour des informations est similaire :

```
// Defines an object to contain the updated values
ContentValues updateValues = new ContentValues();

// Defines selection criteria for the rows you want to update
String selectionClause = UserDictionary.Words.LOCALE +  " LIKE ?";
String[] selectionArgs = {"en_%"};

// Defines a variable to contain the number of updated rows
int rowsUpdated = 0;

/*
* Sets the updated value and updates the selected words.
*/
updateValues.putNull(UserDictionary.Words.LOCALE);
```

CONTENTPROVIDER ET SERVICES

Quand nous regardons dans le dictionnaire, nous voyons que maintenant nos mots sont disponibles pour toutes les langues :



CONTENTPROVIDER ET SERVICES

EFFACER DES DONNÉES

Nous allons maintenant supprimer des données :

```
// Defines selection criteria for the rows you want to delete
String selectionClause = UserDictionary.Words.WORD + " LIKE ?";
String[] selectionArgs = {"%insert%"};

// Defines a variable to contain the number of rows deleted
int rowsDeleted = 0;

// Deletes the words that match the selection criteria
rowsDeleted = getContentResolver().delete(
    UserDictionary.Words.CONTENT_URI,   // the user dictionary content URI
    selectionClause,                    // the column to select on
    selectionArgs                       // the value to compare to
);
```

CONTENTPROVIDER ET SERVICES

AUTRES CONTENTS PROVIDERS

De nombreux contents providers sont disponibles tels que :

- La liste des appels.
- La liste des SMS.
- Le répertoire des contacts.
- Les dictionnaires de l'utilisateur.
- La liste des favoris.
- ...

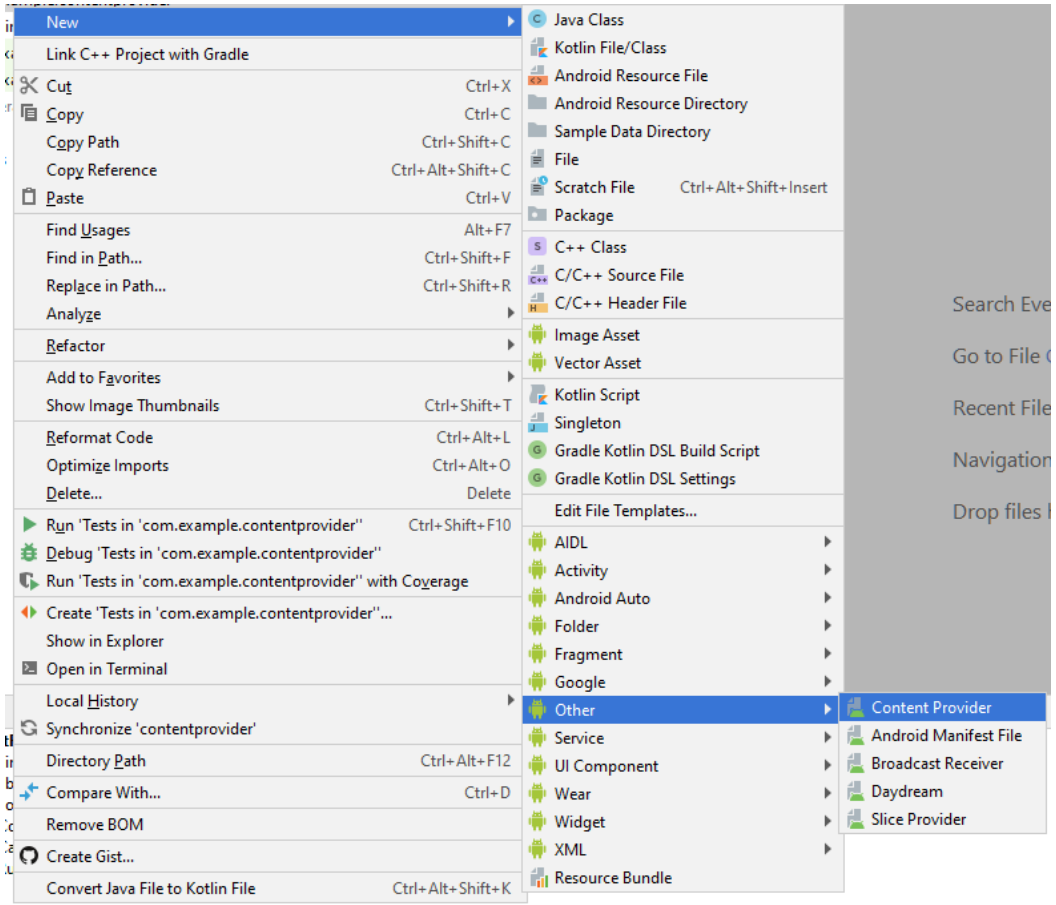
CRÉATION D'UN CONTENT PROVIDER

LES ÉTAPES

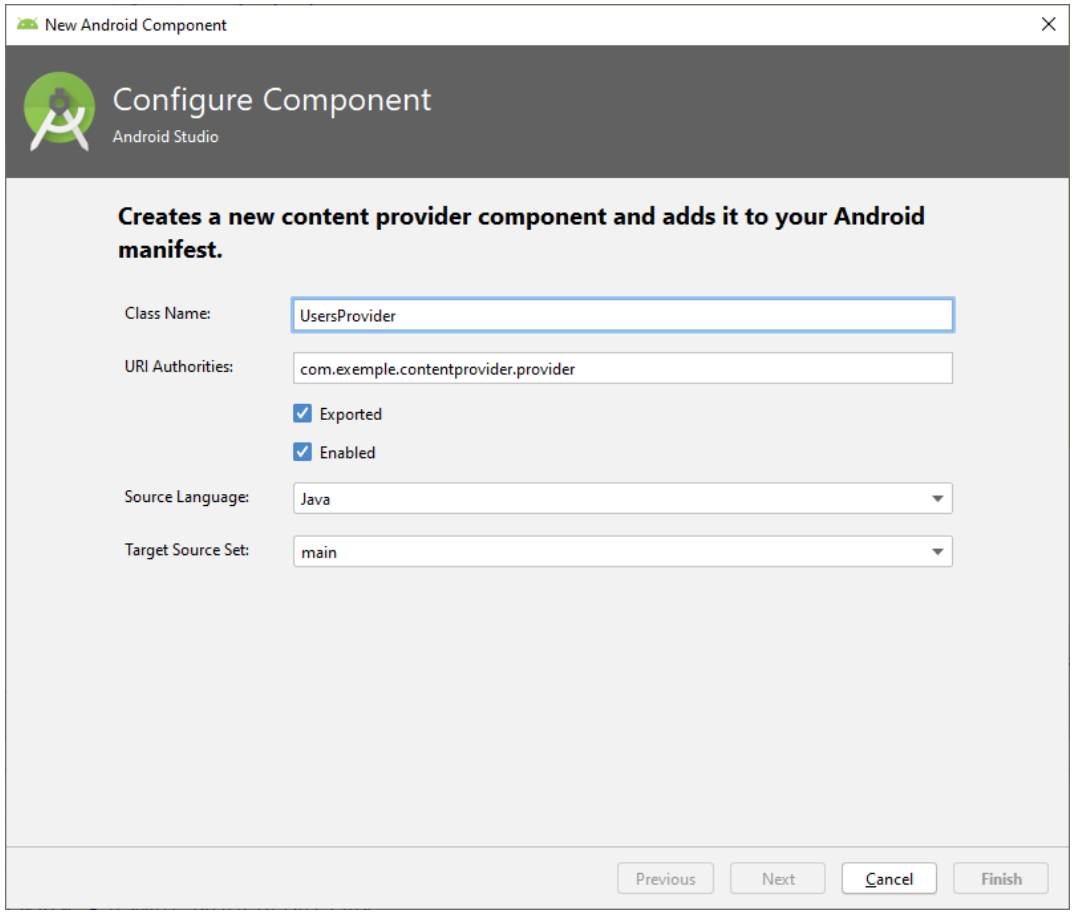
Nous allons maintenant créer un content provider. Pour cela nous allons suivre plusieurs étapes :

- Créer un projet `ContentProvider`.
- Implémenter rapidement la méthode `query`.
- Tester dans le projet `ContentProviderClient` la bonne réaction.
- Implémenter une base de donnée pour le Content Provider.

CRÉATION D'UN CONTENT PROVIDER



CRÉATION D'UN CONTENT PROVIDER



CRÉATION D'UN CONTENT PROVIDER

IMPLÉMENTATION RAPIDE

La méthode `query` doit retourner un `Cursor`, implémentons rapidement la création d'un `Cursor` contenant des `Users`.

```
1  @Override
2  public Cursor query(Uri uri, String[] projection, String selection,
3                      String[] selectionArgs, String sortOrder) {
4
5      MatrixCursor cursor = new MatrixCursor(new String[]{"name", "firstname"});
6      cursor.newRow()
7              .add("name", "SALAUN")
8              .add("firstname", "Tristan");
9
10     cursor.newRow()
11             .add("name", "SNOW")
12             .add("firstname", "John");
13     return cursor;
14     }
```

CRÉATION D'UN CONTENT PROVIDER

TEST DANS CONTENTPROVIDERCLIENT

Ouvrons le projet `ContentProviderClient`, et ajoutons le code pour interroger notre nouveau `ContentProvider` :

```
1  Cursor cursor = getContentResolver().query(Uri.parse("content://com.exemple.contentprovider.provider"), null, null, null
2  if(cursor.moveToFirst()) {
3      StringBuilder strBuild=new StringBuilder();
4      while (!cursor.isAfterLast()) {
5          strBuild.append("\n"+cursor.getString(0)+ "-"+ cursor.getString(1));
6          cursor.moveToNext();
7      }
8      if(BuildConfig.DEBUG){
9          Log.d(TAG, "onCreate " + strBuild);
10     }
11 }
```

CRÉATION D'UN CONTENT PROVIDER

BASE DE DONNÉE AVEC ROOM

Implémentons maintenant une vrai base de donnée en utilisant Room.
Pour cela je vais utiliser le Live Template `android_db_room_00_documentation` et écrire les classes pour gérer ma base de donnée. La base de notre Content Provider sera l'entity suivante :

```
1 public class UserEntity {
2     public String name;
3     public String firstname;
4 }
```

CRÉATION D'UN CONTENT PROVIDER

CLASSE DE CONTRAT

Afin de rendre notre content provider plus accessible à des développeurs tiers, nous allons écrire une classe de contrat définissant toutes les informations nécessaires pour accéder à notre content provider :

```java
import android.net.Uri;
import android.provider.BaseColumns;

public final class UserContract {
    // The Authority
    public static final String AUTHORITY = "com.exemple.contentprovider.provider";

    // The path to the data… and explain
    public static final String PATH_TO_DATA = "users"; //Vous pouvez déclarer plusieurs paths (les paths utilisent les /


    public interface MyDatas extends BaseColumns {

        // The URI and explain, with example if you want
        public static final Uri CONTENT_URI = Uri.parse("content://" + UserContract.AUTHORITY + "/" + UserContract.PATH
```

CRÉATION D'UN CONTENT PROVIDER

INTÉGRATION AU CONTENT PROVIDER

Nous allons intégrer la base de donnée à notre content provider.
Implémentation de la méthode afin de récupérer une instance de la base de donnée :

```
1 public boolean onCreate() {
2     Context context = getContext();
3     userDao = AppDatabase.getDatabase(context).userDao();
4     if (userDao != null) {
5         return true;
6     }
7     return false;
8 }
```

CRÉATION D'UN CONTENT PROVIDER

QUERY

Dans une première étape nous retournerons l'entièreté de la base de donnée avec le code suivant :

```
1 public Cursor query(Uri uri, String[] projection, String selection,
2                     String[] selectionArgs, String sortOrder) {
3     return userDao.getCursorAll();
4 }
```

CRÉATION D'UN CONTENT PROVIDER

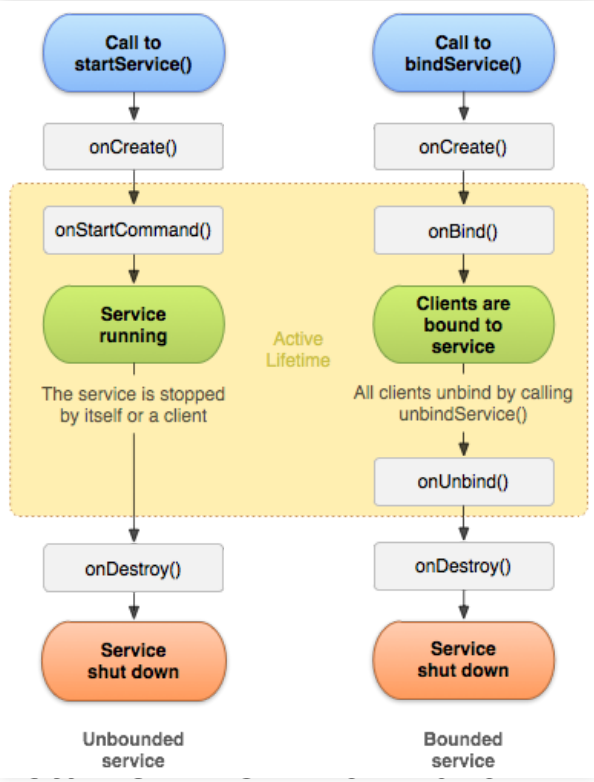
INSERT

Une implémentation possible serait :

```
1  public Uri insert(Uri uri, ContentValues values) {
2      UserEntity currentUserEntity = new UserEntity();
3      if (values.containsKey(UserContract.MyDatas.KEY_COL_FIRSTNAME)) {
4          currentUserEntity.firstname = values.getAsString(UserContract.MyDatas.KEY_COL_FIRSTNAME);
5      }
6      if (values.containsKey(UserContract.MyDatas.KEY_COL_NAME)) {
7          currentUserEntity.name = values.getAsString(UserContract.MyDatas.KEY_COL_NAME);
8      }
9      long rowID = userDao.insert(currentUserEntity);
10     if (rowID > 0) {
11         Uri _uri = ContentUris.withAppendedId(UserContract.MyDatas.CONTENT_URI, rowID);
12         getContext().getContentResolver().notifyChange(_uri, null);
13         return _uri;
14     }
15     throw new SQLiteException("Failed to add a record into " + uri);
```

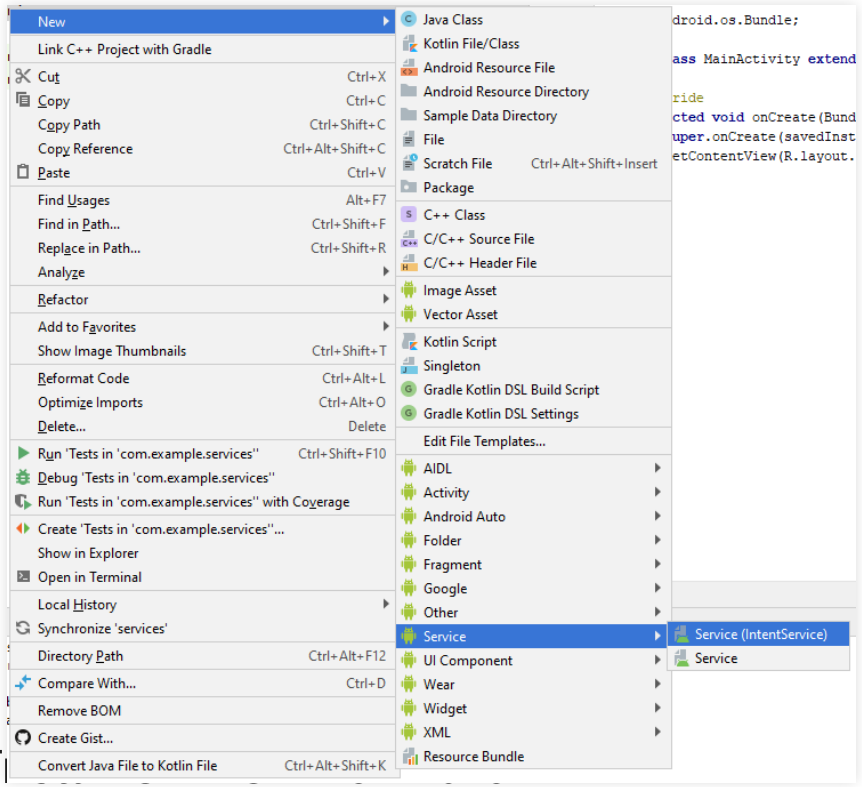
LES SERVICES

CYCLE DE VIE DES SERVICES.



LES SERVICES

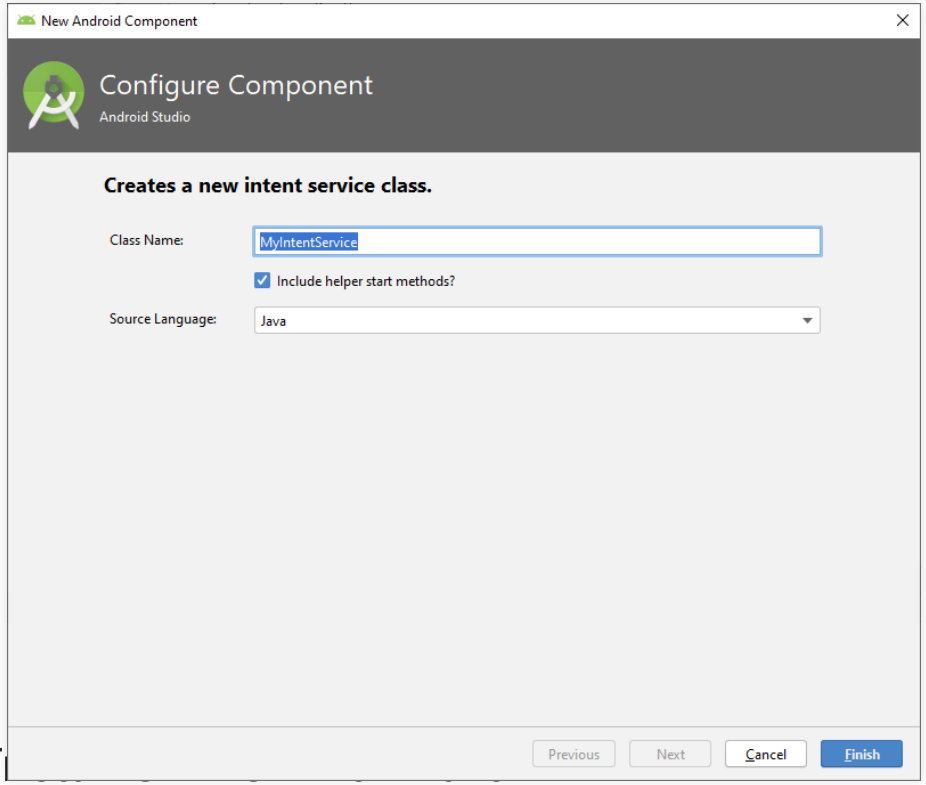
INTENT SERVICE.



IOD

LES SERVICES

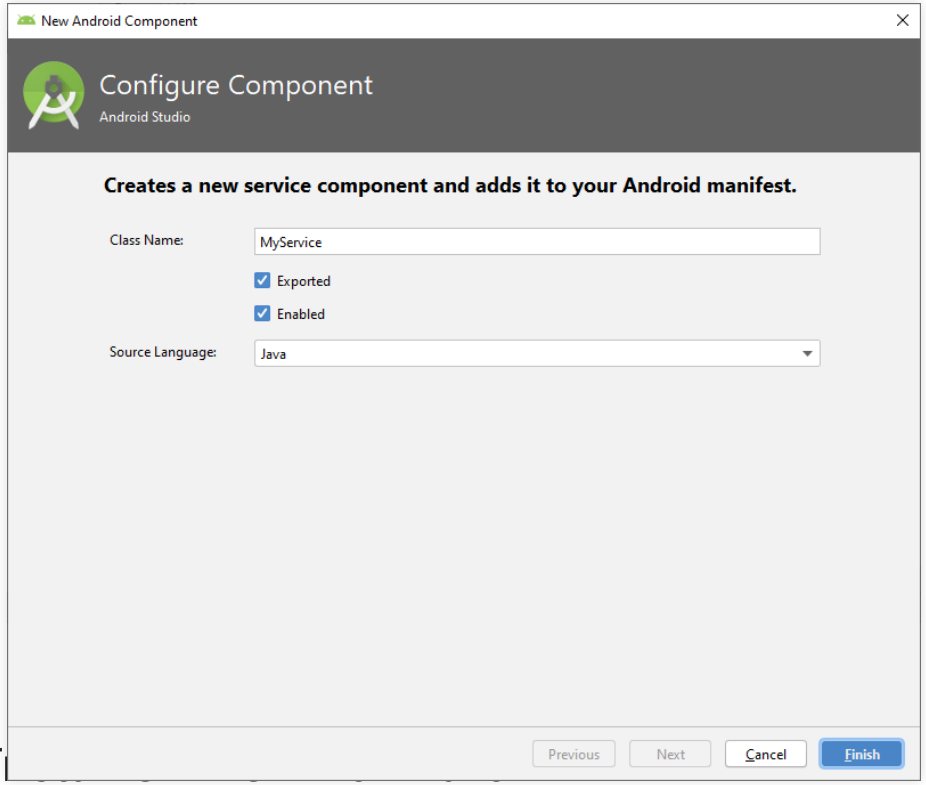
INTENT SERVICE.



T

LES SERVICES

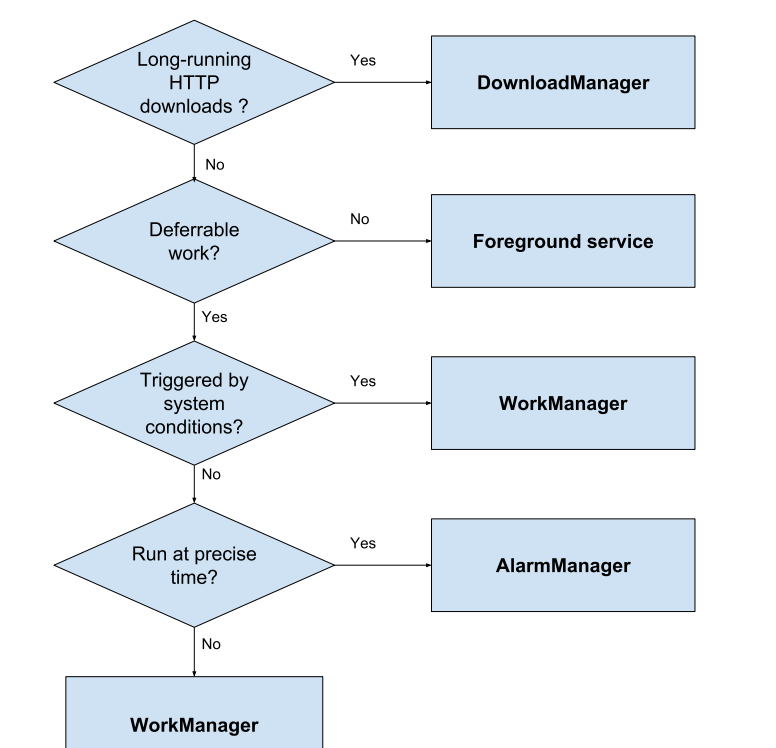
SERVICE BINDÉ.



ARRIÈRE-PLAN ET PREMIER PLAN

LE CHOIX

Pour choisir le bon mode pour lancer une tâche de fond, suivons les indications de ce schéma :



ARRIÈRE-PLAN ET PREMIER PLAN

LIER SERVICES ET ACTIVITÉS.

Nous allons utiliser EventBus pour faire communiquer nos services avec notre Activité.
Dans un premier temps commençons par un tutoriel assez simple, mais regroupant tous les concepts, ici.
Nous trouverons un peu plus de détails sur EventBus ici.

ARRIÈRE-PLAN ET PREMIER PLAN

UTILISER DES THREADS DEPUIS UN SERVICE.

Commençons par utiliser le code suivant dans notre `MainActivity` :

```
for (int i = 0; i < 10; i++) {
    if (BuildConfig.DEBUG) {

        Log.d(TAG, "run background " + i);
    }
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

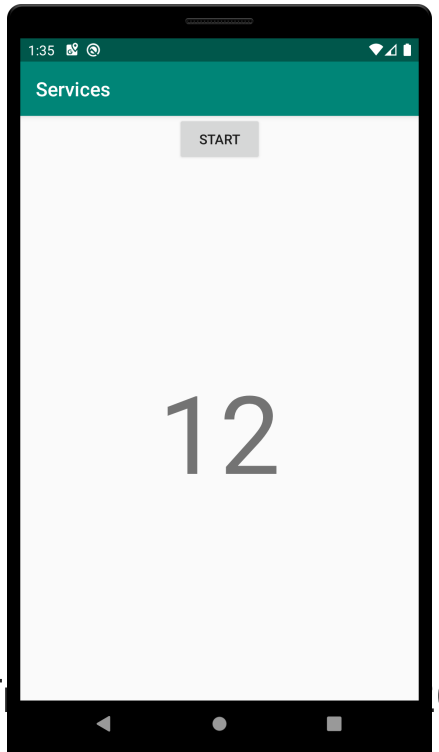
Que remarquons nous ?

Plaçons maintenant le code dans notre service, que remarquons nous ?

ARRIÈRE-PLAN ET PREMIER PLAN

COUNTDOWN SIMPLE

Utilisons EventBus, et un service intent pour lancer notre countdown avec l'interface graphique suivante :



ARRIÈRE-PLAN ET PREMIER PLAN

DÉFINIR DES ALARMES.

Nous pouvons aussi déclencher des actions à un temps (heure/jour) donné.
Un exemple ici.
Modifions l'exemple pour afficher une notification, qui sera plus facile à visualiser. En utilisant le template, cela ne devrais pas être long.

La référence se trouve ici et nous trouverons un autre exemple ici.

ARRIÈRE-PLAN ET PREMIER PLAN

TRAVAUX PRATIQUES

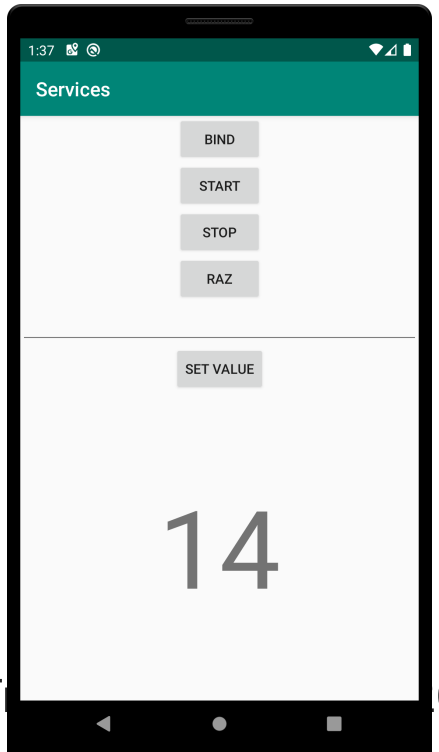
Réalisons maintenant une application qui va nous permettre de gérer un décompte avec les fonctionalités suivante :

- Initialiser la valeur.
- Lancer.
- Arrêter.
- Remettre à zéro.

ARRIÈRE-PLAN ET PREMIER PLAN

COUNTDOWN ÉVOLUÉ

Utilisons maintenant un service bindé pour pouvoir interagir avec notre service via l'interface suivante :



TESTER UNE APPLICATION ANDROID

PRÉSENTATION DES OUTILS

Il existe plusieurs outils adaptés aux différents types de tests. Les différents types de tests :

- Test monkey.
- `monkeyrunner`.
- Test unitaire.
- Test d'intégration.
- Test fonctionnels de bout en bout.

TESTER UNE APPLICATION ANDROID

TESTS MONKEY

Un des test les plus simple à mettre en oeuvre et qui remonte souvent des bugs est le test du signe (monkey).
On le lance de la manière suivante :

```
1 adb shell monkey -p your.package.name -v 500
```

TESTER UNE APPLICATION ANDROID

`monkeyrunner`

Nous pouvons écrire des tests simples, par exemple pour automatiser la génération de captures d'écrans avec un script `monkeyrunner` qui va installer l'application, réaliser des actions (tout en effectuant des captures d'écran), pour enfin effacer l'application testée. Voyons tout cela en détail dans le script partagé.

TESTER UNE APPLICATION ANDROID

TESTS UNITAIRES

Ils permettent de tester les méthodes de chaque classe, séparément.

TESTER UNE APPLICATION ANDROID

FONCTIONS SPÉCIALES

- setUp appelée avant chaque test
- tearDown appelée après chaque test

TESTER UNE APPLICATION ANDROID

ASSERTIONS

- assertEquals(a,b)
- assertTrue(a) et assertFalse(a)
- assertSame(a,b) et assertNotSame(a,b)
- assertNull(a) et assertNotNull(a)
- fail (message)

TESTER UNE APPLICATION ANDROID

BONNES PRATIQUES

- Écrire les test en même temps que le code.
- Tester uniquement ce qui peut vraiment provoquer des erreurs.
- Exécuter ses tests aussi souvent que possible, idéalement après chaque changement de code.
- Écrire un test pour tout bogue signalé (même s'il est corrigé).
- Ne pas tester plusieurs méthodes dans un même test : JUnit s'arrête à la première erreur.
- Attention, les méthodes privées ne peuvent pas être testées !

TESTER UNE APPLICATION ANDROID

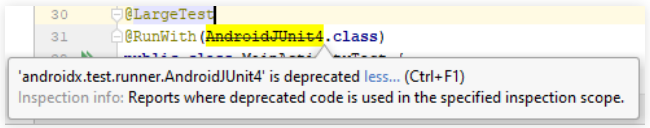
SIMULATION D'INTERACTIONS UTILISATEUR AVEC ESPRESSO.

Nous allons voir plus en détails la mise en place des tests.

TESTER UNE APPLICATION ANDROID

`AndroidJUnit4` DÉPRÉCIÉ

Pour corriger le message :



Ajoutons la librairie :

```
1 androidTestImplementation 'androidx.test.ext:junit:1.1.1'
```

Et modifions l'import correspondant.

TESTER UNE APPLICATION ANDROID

TEST CONTENT PROVIDER

Regardons plus en détail le projet `ContentProvider` et mettons en place des tests relatifs à notre base de donnée.

TESTER UNE APPLICATION ANDROID

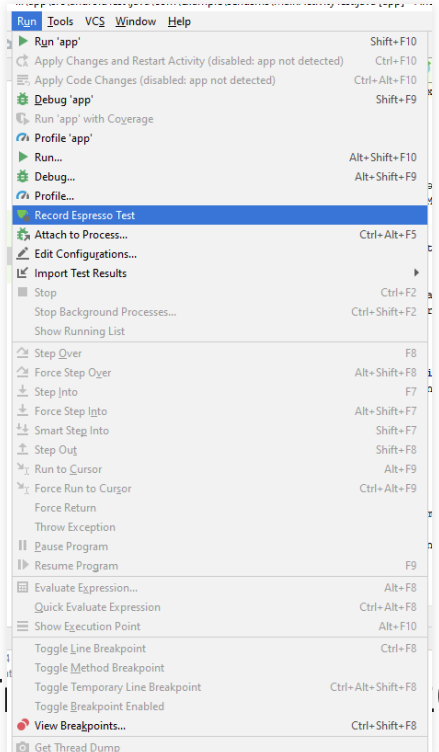
TEST SUITE

Regardons maintenant plus en détail le projet `TestSuite` nous mettrons en place des tests sur notre application d'envoi de SMS.

TESTER UNE APPLICATION ANDROID

TESTS ENREGISTRÉS

Nous pouvons mettre en place des tests de manière graphique sans avoir besoin d'écrire de code.



TESTER UNE APPLICATION ANDROID

UTILISATION DE CLOUD TEST LAB.

Nous allons suivre le tutoriel ici

La référence est disponible ici : https://firebase.google.com/docs/test-lab.

SEND SMS

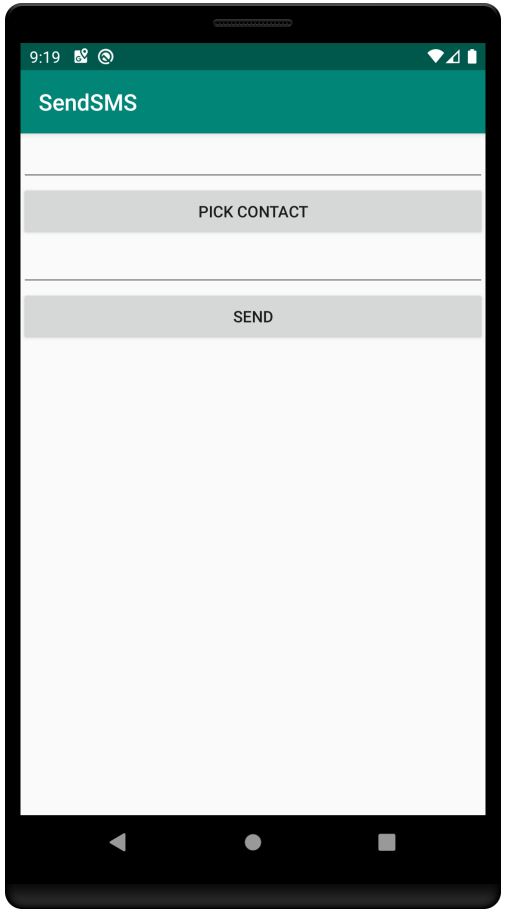
PRÉSENTATION

Revoyons certains principes vu dans la formation, nous allons réaliser une application qui permet d'envoyer des SMS, et informer l'utilisateur de l'état d'avancée des envois/réceptions. Nous allons encore une fois grandement nous aider des Live Templates mis à disposition.

SEND SMS

GUI

Mettons en place au minimum, l'interface suivante :



SEND SMS

CONTACT PICKER

Implémentons la sélection des contacts, en deux étapes :

- L'implémentation du click pour sélectionner le contact.
- La récupération du contact lors du retour de la sélection.

Implémentons le callback pour le click du bouton, et utilisons le Live Template : `android_intent_select_contact` pour implémenter la méthode.
Suivons les instructions.

SEND SMS

SEND SMS

Implémentons l'envoi des SMS, en suivant les étapes suivantes :

- Récupération des références vers les `EditText`.
- Implémentation du click pour envoyer le SMS.
- Ajoutons la permission pour envoyer les SMS.
- Implémentation des classes de BroadcastReceiver.

SEND SMS

RÉCUPÉRATION DES `EditText`

Déclarons nos deux variables d'instances de classe :

```
1 private EditText editTextNumber, editTextText;
```

Récupérons leur valeur :

```
1 editTextNumber = findViewById(R.id.activity_main_editText_number);
2 editTextText = findViewById(R.id.activity_main_editText_message);
```

SEND SMS

IMPLÉMENTATION DU CODE

Implémentons le code pour envoyer les SMS : utilisons, hors d'une méthode, le Live Template : `android_send_sms_01_method`.
Suivons les instructions.
Implémentons le callback pour le click du bouton avec :

```
1 sendSMS(getApplicationContext(), editTextNumber.getText().toString(), editTextText.getText().toString());
```

SEND SMS

PERMISSIONS

Utilisons le Live Template `android_permission_single`.
Déplaçons le code situé dans le click du bouton, vers la méthode `actionToBeCalled()`, et remplaçons le par l'appel à la méthode `actionToBeCalled()`.
Testons notre code.

SEND SMS

CORRECTION DES ERREURS

Si nous avons l'erreur" :

```
1 java.lang.SecurityException: Sending SMS message: uid XXXXX does not have android.permission.SEND_SMS.
```

Nous avons :

- Soit oublié d'ajouter la permission dans le `Manifest.xml`.
- Soit oublié de demander la permission dynamiquement (Android 6.0+).

SEND SMS

ALLONS PLUS LOIN

Améliorons l'UX de notre application :

- Passons au Material design sur les `EditText` et les `Buttons`.
- Effectuons des tests sur les valeurs des champs.
- Mémorisons les dernières valeurs utilisées.
- Mettons en place un EventBus pour communiquer des `BroadcastReceiver`s vers la `MainActivity`.

ANNEXES

LIVESTEMPLATES

Nous avons à notre disposition plusieurs LiveTemplates, tels que :

- `android_tts` : pour activer une synthèse vocale.
- `android_intent_voice_recognition` : pour activer une reconnaissance vocale.

ANNEXES

RÉFÉRENCES UTILES

- Liste de nombreuses librairies
- JetPack
- Liste de cour intéressants.
- GitLab