Formations
Tristan SALAÜN

Pour Kotlin, Android (Java et Kotlin)
premier niveau et avancé

Supports disponibles

- Android premier niveau.
- Android, perfectionnement (Réf. IOD).
- Kotlin.
- Kotlin, développer des applications pour Android (Réf. OTA).
- Kotlin mise en oeuvre (Réf. OTB).

Utilisation de cette présentation

Appuyons sur la touche ?

Android avancé, des bases à des techniques de développement plus avancées

Le support en PDF.

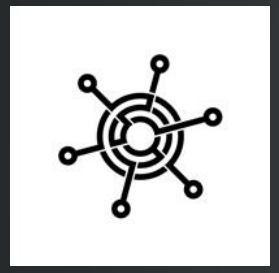
Présentations

- Qui suis-je.
- Qui êtes-vous ? Quelles sont vos attentes.
- Présentation du plan.

Présentations

Qui suis-je

- Développeur d'applications mobiles Android (depuis 2009).
- Formateur Android/Java et Kotlin, et Javacard (scolaires et pros).
- Co-fondateur (1/7) de Startup Marseille.



- Fondateur de Light4Events.

Présentations

Qui êtes-vous ?

Tour de table :

- Prénom.
- Expérience (Java, Kotlin, Mobile Android/iOS).
- Attentes.

Mise en place

Étapes de mise en place :

- Android Studio.
- ADB.
- Téléphone en mode développeur.
- Coloration de logcat.
- Mise en place des templates.
- Mise en place de scrcpy.

Mise en place

Android Studio

Pour commencer, nous allons vérifier que nous avons bien Android Studio fonctionnel avec la dernière version :

Mise en place

ADB

Vérifions que ADB est bien accessible en ligne de commande, cela nous sera utile par la suite.

- Ouvrons une invite de commande (Windows + R, cmd).
- Saisissons : `adb version`.

Si ce n'est pas le cas, ajoutons le au "PATH".

Mise en place

Téléphone en mode développeur

Nous vérifions que notre téléphone est en mode développeur :

- En ligne de commande tapons : `adb devices`
- Vérifions dans Android Studio que le device apparaît bien.

Dans le cas contraire, nous vérifions que le téléphone a bien le mode développeur activé.

Mise en place

Coloration de logcat

Nous allons colorer le logcat afin qu'il soit plus lisible :

- File / Settings
- Cherchons : "logcat"
- Editor / Color Scheme / Android Logcat
- Les couleurs que j'ai définies (à titre d'exemple) :

Assert	C00000
Debug	41BB2E
Error	FF6B68
Info	F5F550
Verbose	BBBBBB
Warning	F4AC40

Mise en place

Les LiveTemplates, sont des raccourcis qui permettent d'écrire rapidement des bouts de code. Je vous fournis une série de LiveTemplates que nous utiliserons dans la suite de la formation. Procédons comme suit :

- Récupération des templates :
  - Projet GitHub LiveTemplates
  - Cliquons sur le bouton Code
  - Download ZIP
- Les copier dans :
  - Windows: `C:\Users<your_user>\AppData\Roaming\Google\AndroidStudio2022.3\templates` (par ex, selon la version installée)
  - Linux: `~/.AndroidStudio"version"/config/templates`
  - macOS: `~/Library/Preferences/AndroidStudio"version"/templates`
- File/Invalidate caches/Restart...
- Just Restart.
- File/Settings.

Mise en place

Utilisation

Pour utiliser les LiveTemplates, il suffit de taper le début de l'abréviation, lancer l'autocomplétion et le LiveTemplate sera exécuté.

- Par exemple tous les lives templates pour compose `and_compose_`.
- Par exemple tous les lives templates pour les tests UIAutomator commencent par `and_uia_`.
- Les autres templates commencent tous par `and_` et sont filtrés par le type de fichier dans lequel on se trouve (XML, Java, Kotlin).

Mise en place

Shell scripts

Afin d'automatiser certaines tâches, il est possible de les scripter. Pour cela je vous propose une série de scripts que vous pourrez utiliser après la formation :
Récupération des scripts :

- Projet GitHub AndroidBashScripts
- Cliquons sur le bouton Code
- Download ZIP

Pour les lancer sous Windows il faudra installer Git.

Mise en place

scrcpy

Nous allons installer `scrcpy` qui nous permettra de partager l'affichage de notre téléphone.

- Téléchargeons la dernière version sur GitHub.
- Décompressons le fichier ZIP dans un répértoire (`C:\tools` par exemple) ce qui donnera dans notre cas : `C:\tools\scrcpy-win64-v2.3.1` .
- Lançons l'utilitaire avec `scrcpy` .

Une solution si vous avez installé `winget` est d'utiliser la commande suivante :

```
1 winget install --silent --id=Genymobile.scrcpy  -e
```

Copier

Réalisation d'une application Android simple en Kotlin

Nous allons commencer par poser les bases d'une application Android en Kotlin.
Nous allons suivre les étapes suivantes :

- Installer et lancer Android Studio.
- Créer un projet : IMC.
- Mettre en place le view binding.
- Définir notre interface graphique.
- Implémenter le code métier.

Réalisation d'une application Android simple en Kotlin

Installer et lancer Android Studio

Il suffit d'avoir une bonne connexion internet, et d'aller sur https://developer.android.com/studio.

Réalisation d'une application Android simple en Kotlin

Créer un projet : IMC

Réalisation d'une application Android simple en Kotlin

Mettre en place le view binding

Pour cela, nous allons suivre les indications de la page : View Binding.

Réalisation d'une application Android simple en Kotlin

Modifions le build.gradle (Module :app)

```
android {
    ...
    buildFeatures {
        viewBinding true
    }
}
```

Copier

Si l'on est en kotlin (build.gradle.kt) :

```
android {
    ...
    buildFeatures {
        viewBinding = true
    }
}
```

Copier

Modifions la méthode `onCreate` de notre Activity

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ResultProfileBinding.inflate(layoutInflater)
    val view = binding.root
    setContentView(view)
}
```

Copier

Réalisation d'une application Android simple en Kotlin

Usage

Nous pouvons maintenant utiliser nos éléments graphiques directement depuis la classe binding, par exemple :

```
1 binding.name.text = viewModel.name
2 binding.button.setOnClickListener { viewModel.userClicked() }
```

Copier

Réalisation d'une application Android simple en Kotlin

Définir une première interface graphique

Réalisation d'une application Android simple en Kotlin

Solution proposée

Utilisons un ConstraintLayout :

Copier

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
                                                   xmlns:app="http://schemas.android.com/apk/res-auto"
                                                   xmlns:tools="http://schemas.android.com/tools"
                                                   android:layout_width="match_parent"
                                                   android:layout_height="match_parent"
                                                   tools:context=".MainActivity">


    <Button
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="BTN1"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toStartOf="@+id/button2"
            app:layout_constraintStart_toStartOf="parent" />
```

Réalisation d'une application Android simple en Kotlin

Implémenter les méthodes de base

- Ajouter un tag aux boutons avec des valeurs différentes.
- Définit une callback qui sera appelée par le `setOnClickListener`.
- Assigner la callback à nos 3 boutons en utilisant `with`.

Réalisation d'une application Android simple en Kotlin

Solution proposée

Ajoutons un tag aux boutons :

Copier

```
<Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:tag="Salut"
        android:text="btn2"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button3"
        app:layout_constraintStart_toEndOf="@+id/button1" />
```

Réalisation d'une application Android simple en Kotlin

Solution proposée

Définissons la callback : `buttonCallback` :

Copier

```
1 val buttonCallback: (View) -> Unit = { localView ->
2     if (BuildConfig.DEBUG) {
3         Log.d(TAG, "onCreate ${localView.tag}")
4     }
5 }
```

Réalisation d'une application Android simple en Kotlin

Solution proposée

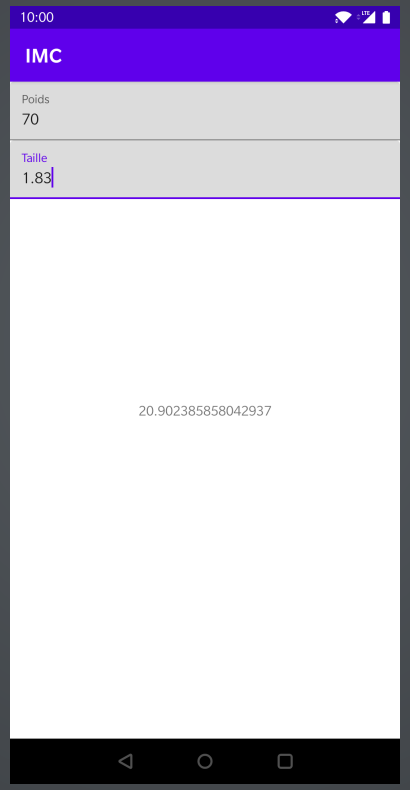
Assignons la callback aux boutons en utilisant le mot clé `with` :

```
1 with(binding) {
2     button1.setOnClickListener(buttonCallback)
3     button2.setOnClickListener(buttonCallback)
4     button3.setOnClickListener(buttonCallback)
5 }
```

Copier

L'application IMC

Définir notre interface graphique



L'application IMC

Solution proposée

Utilisons un ConstraintLayout :

Copier

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
                                                   xmlns:app="http://schemas.android.com/apk/res-auto"
                                                   xmlns:tools="http://schemas.android.com/tools"
                                                   android:layout_width="match_parent"
                                                   android:layout_height="match_parent"
                                                   tools:context=".MainActivity">

    <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/activity_main_editText_weight_layout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            android:hint="Poids">
```

L'application IMC

Implémenter le code métier

- Écouter les changements de valeur des champs textes.
- Calculer et afficher la valeur de l'IMC avec la formule : poids / taille (en m)².
- Afficher des indications supplémentaires.
- Enregistrer la taille.

L'application IMC

Solution proposée

Pour écouter les changements de valeur des champs textes, nous allons utiliser la méthode `addTextChangedListener` :

```
binding.editTextHeight.addTextChangedListener { v ->
    calcIMC()
}
binding.editTextWeight.addTextChangedListener { v ->
    calcIMC()
}
```

Copier

L'application IMC

Solution proposée

Nous allons implémenter la méthode `calcIMC()` :

Copier

```
fun calcIMC() {
    val weight = try {
        binding.editTextWeight.text.toString().toInt()
    } catch (e: NumberFormatException) {
        binding.textViewResult.text = "ERROR"
        0
    }
    val height: Float = try {
        binding.editTextHeight.text.toString().toFloat()
    } catch (e: NumberFormatException) {
        binding.textViewResult.text = "ERROR"
        0f
    }
    if (weight != 0 && height != 0f) {
        binding.textViewResult.text = (weight / (height * height)).toString()
    }
}
```

L'application IMC

Solution proposée

Pour afficher la page web, nous allons utiliser le LiveTemplate : `and_webview`.

L'application IMC

Solution proposée

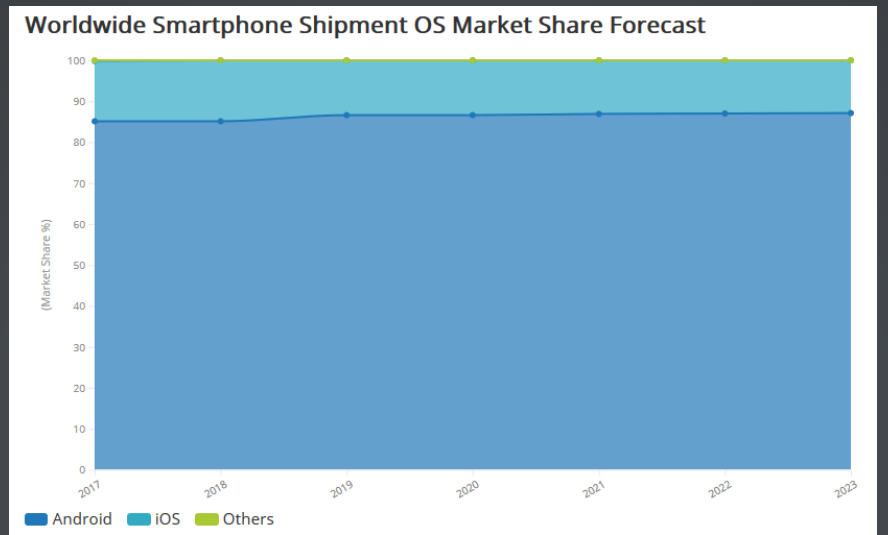
Pour enregistrer et stocker la valeur du poids, nous allons utiliser les 2 LiveTemplates :

- `and_SharedPreferences_load`
- `and_SharedPreferences_save`

La plate-forme Android

- L'architecture Android, Linux. Historiques et fonctionnalités.
- Les terminaux cibles.

Android dans le monde



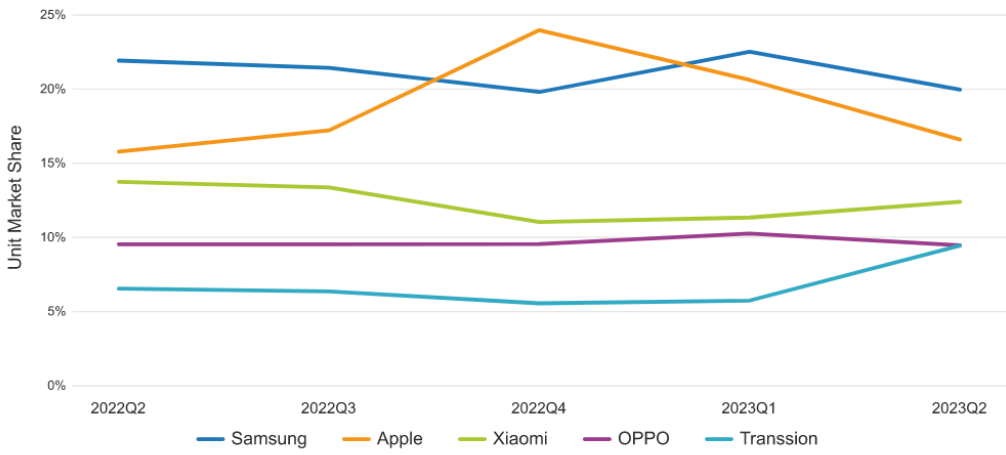
Year	2017	2018	2019	2020	2021	2022	2023
Android	85.1%	85.1%	86.6%	86.6%	86.9%	87.0%	87.1%
iOS	14.7%	14.9%	13.4%	13.4%	13.1%	13.0%	12.9%
Others	0.2%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
TOTAL	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**

Source : https://www.idc.com/promo/smartphone-market-share/os

Android dans le monde

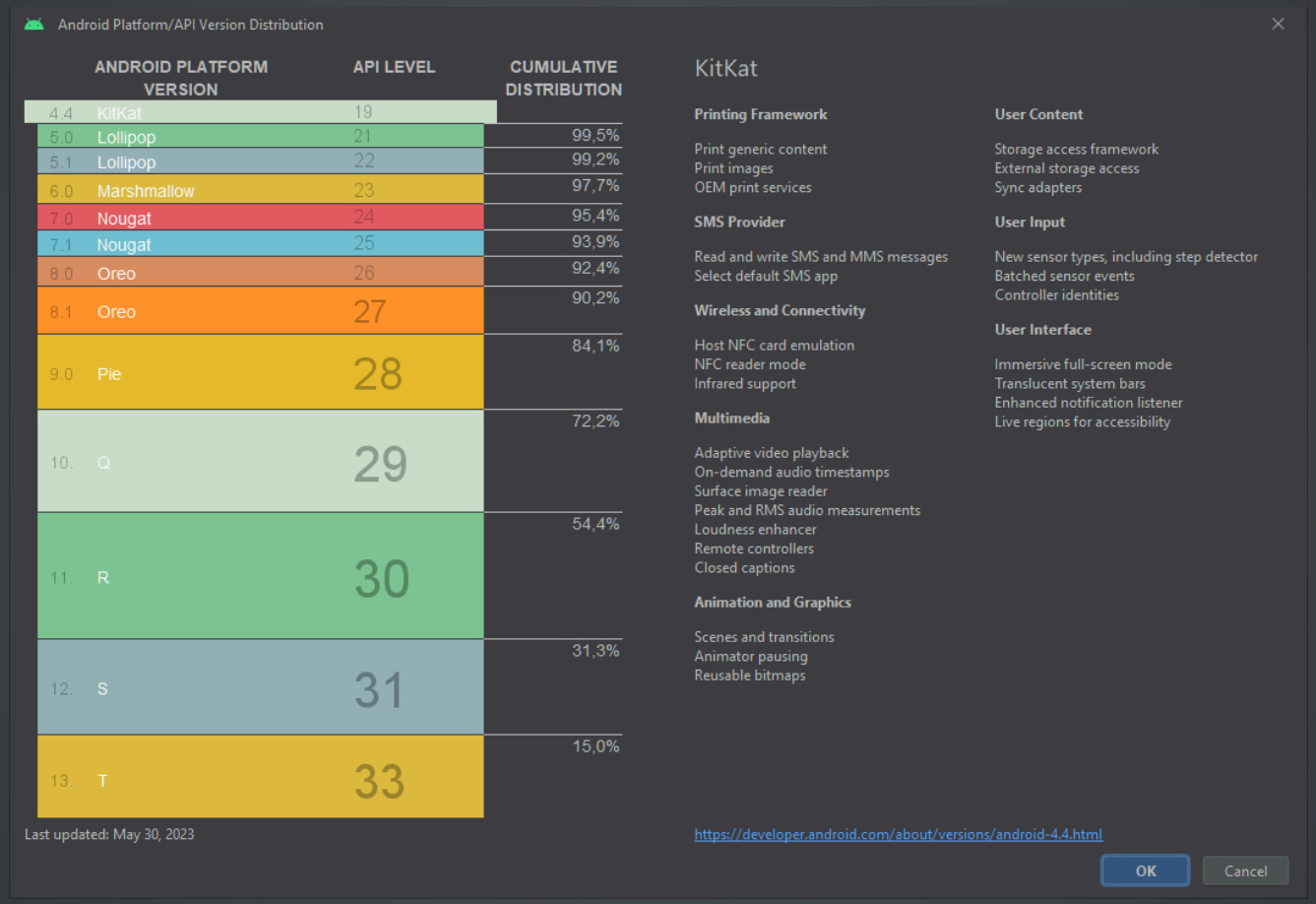
Data table for "Worldwide Top 5 Smartphone Companies, 2023Q2 Unit Market Share" chart

Company	2022Q2	2022Q3	2022Q4	2023Q1	2023Q2
Samsung	21.9%	21.4%	19.8%	22.5%	20.0%
Apple	15.8%	17.2%	24.0%	20.6%	16.6%
Xiaomi	13.8%	13.4%	11.0%	11.4%	12.4%
OPPO	9.5%	9.5%	9.6%	10.3%	9.5%
Transsion	6.5%	6.3%	5.6%	5.7%	9.4%
Others	32.4%	32.1%	30.0%	29.5%	32.1%

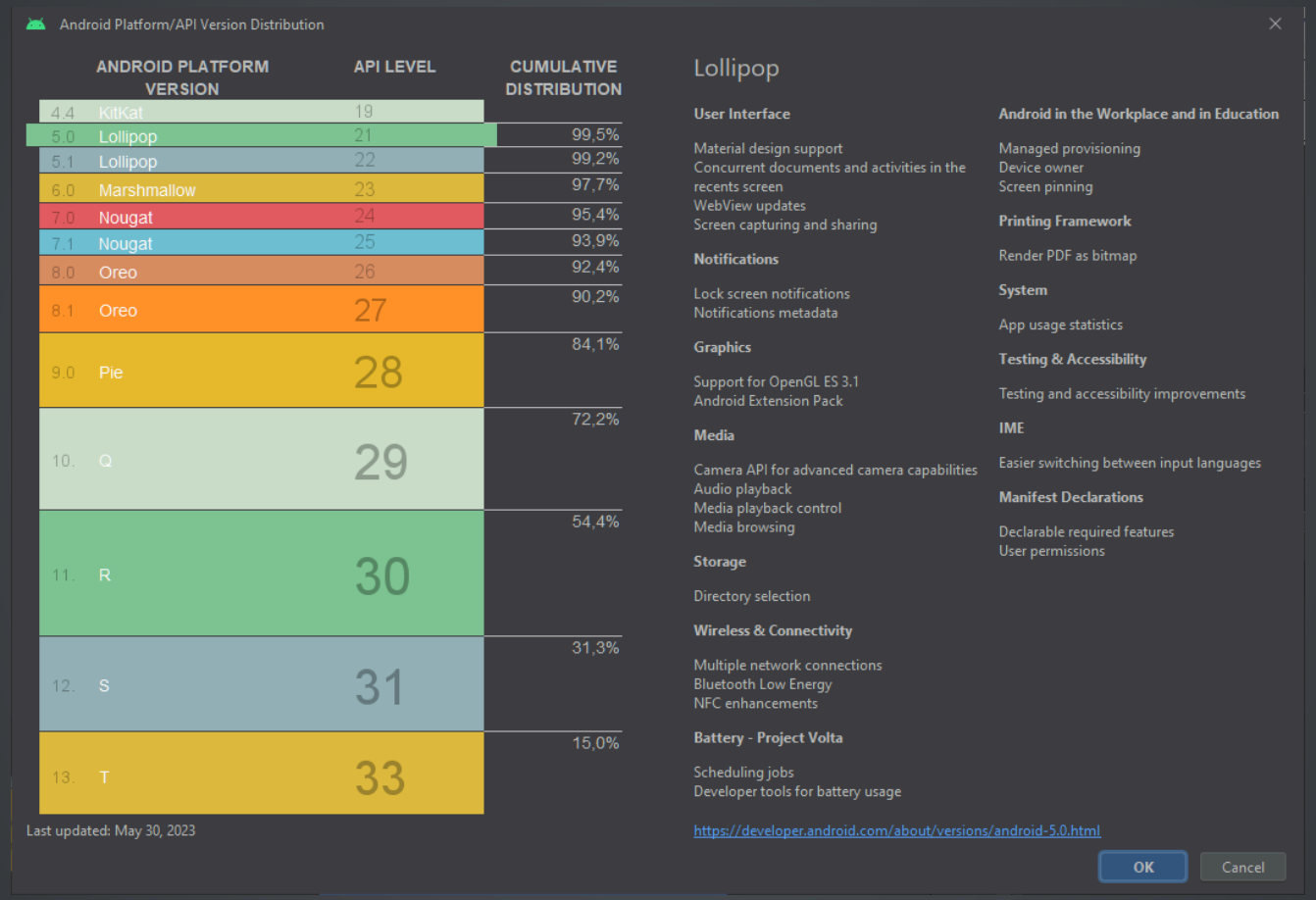


Source : https://www.idc.com/promo/smartphone-market-share/vendor

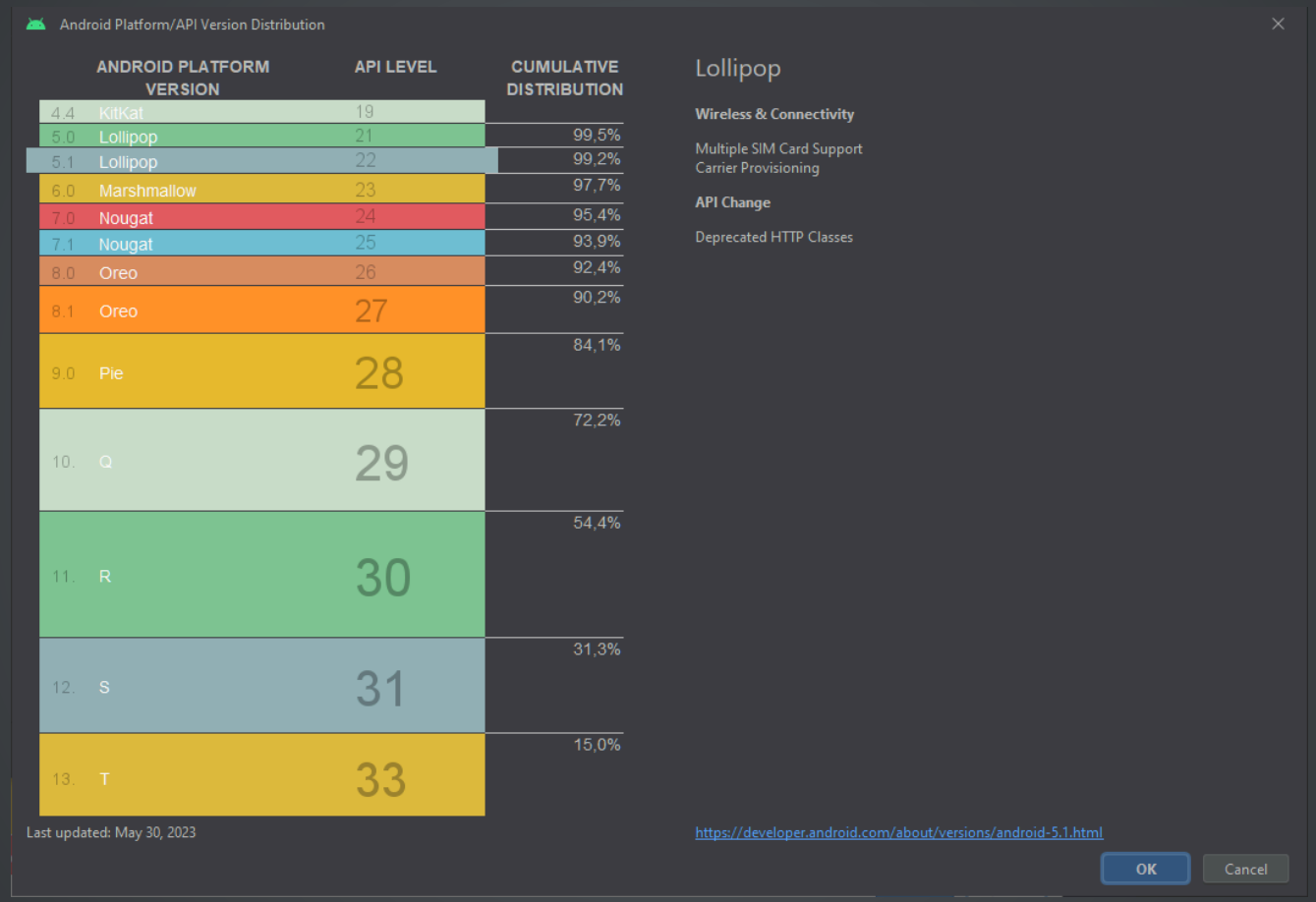
L'histoire d'Android



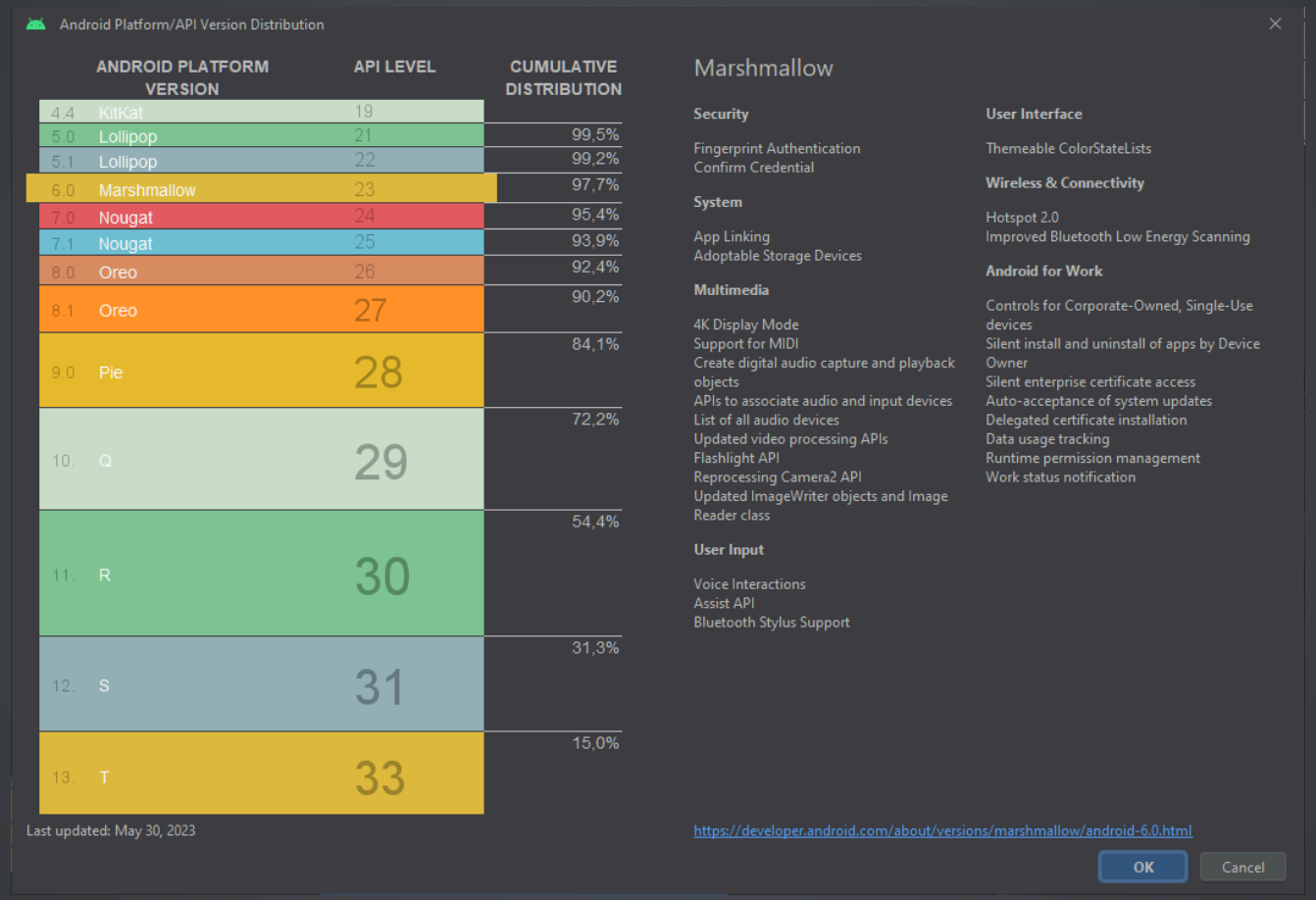
L'histoire d'Android



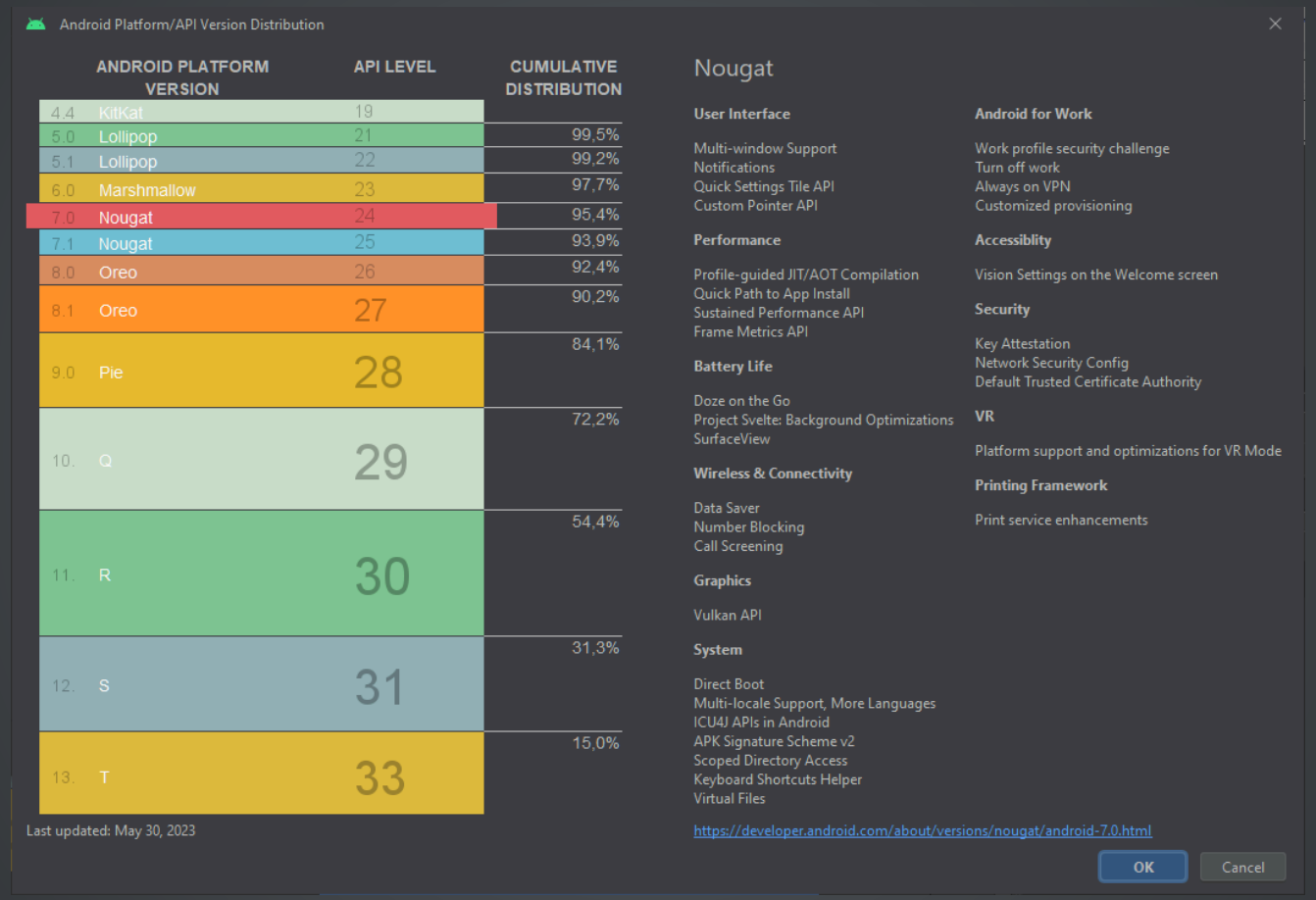
L'histoire d'Android



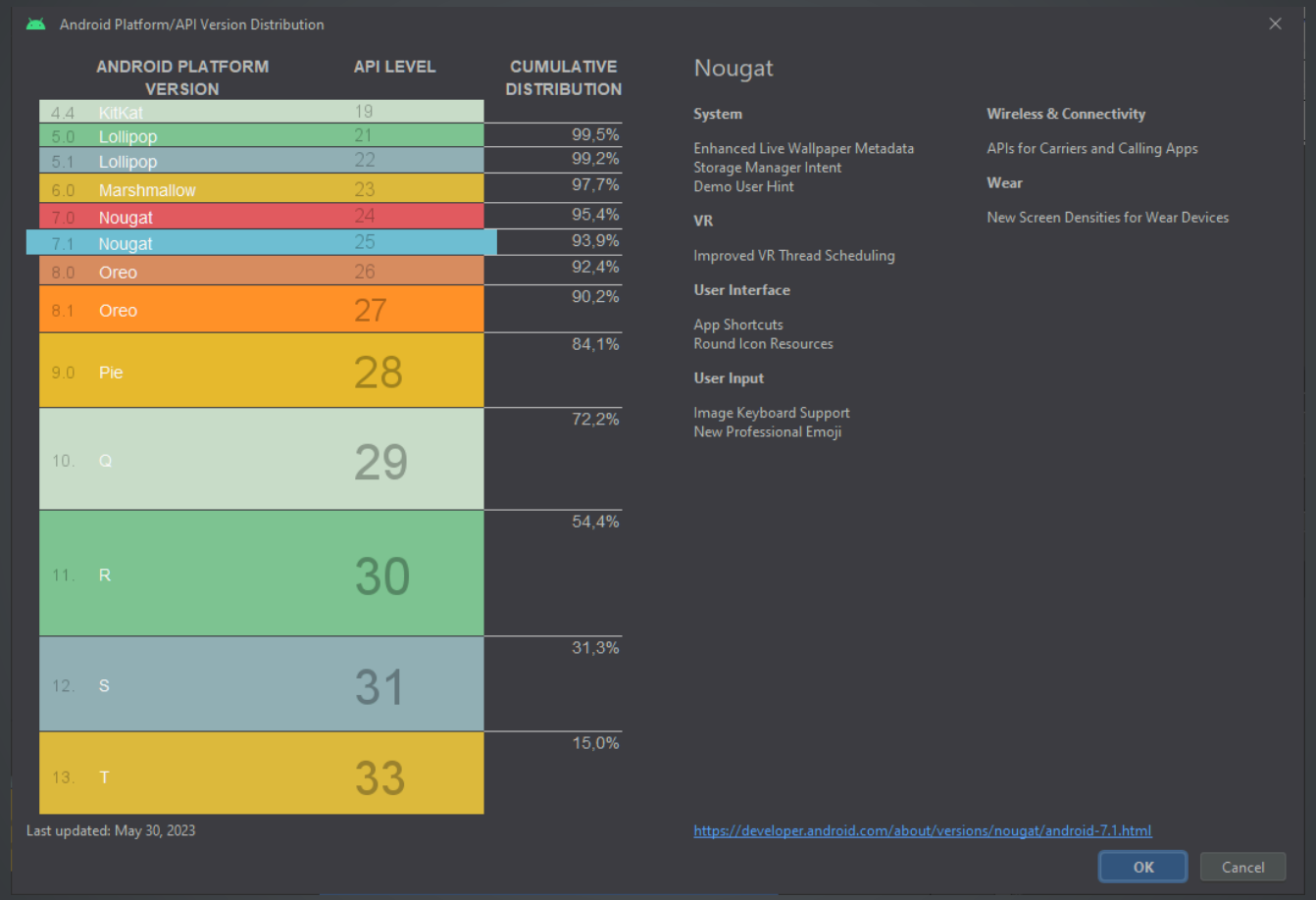
L'histoire d'Android



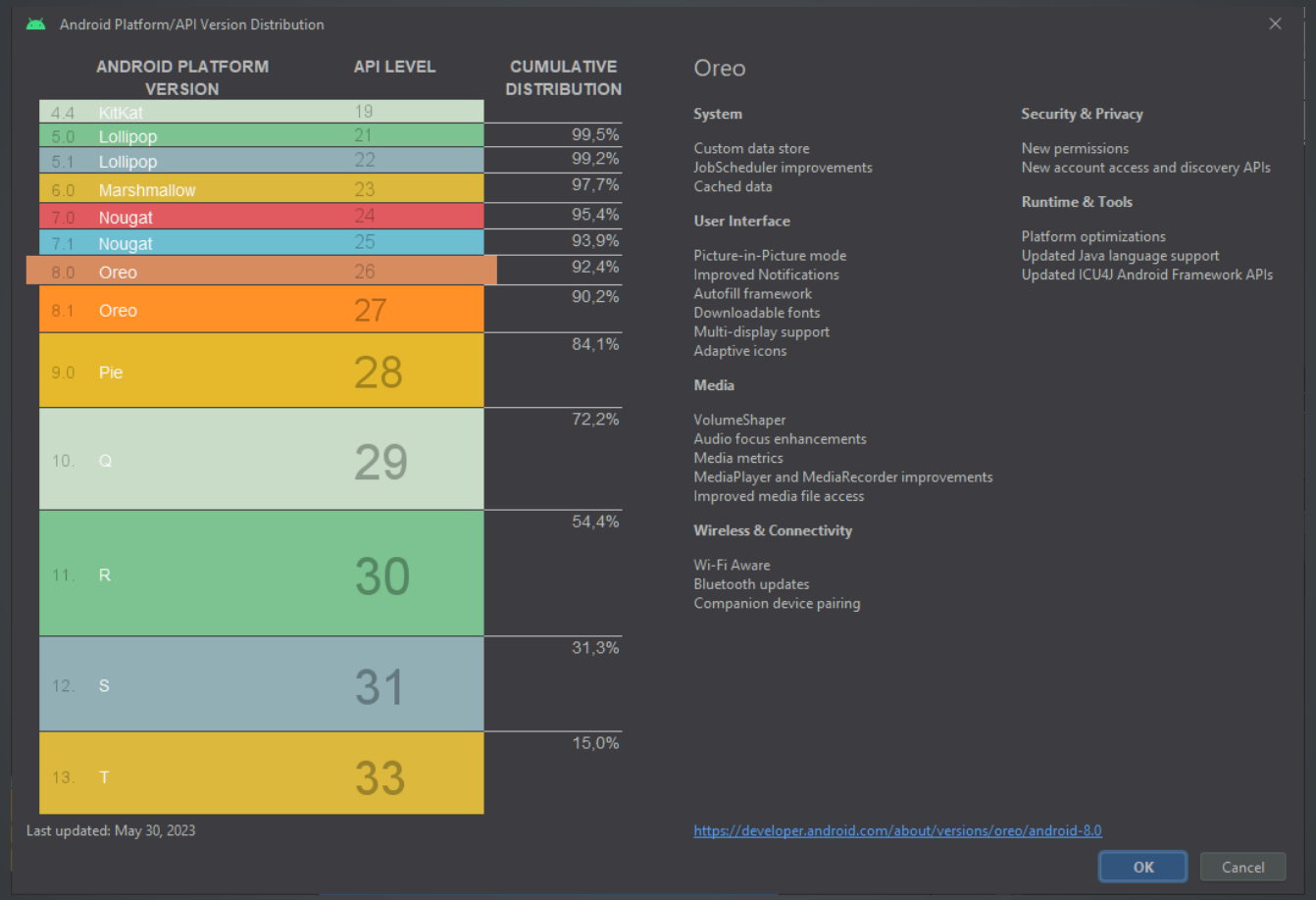
L'histoire d'Android



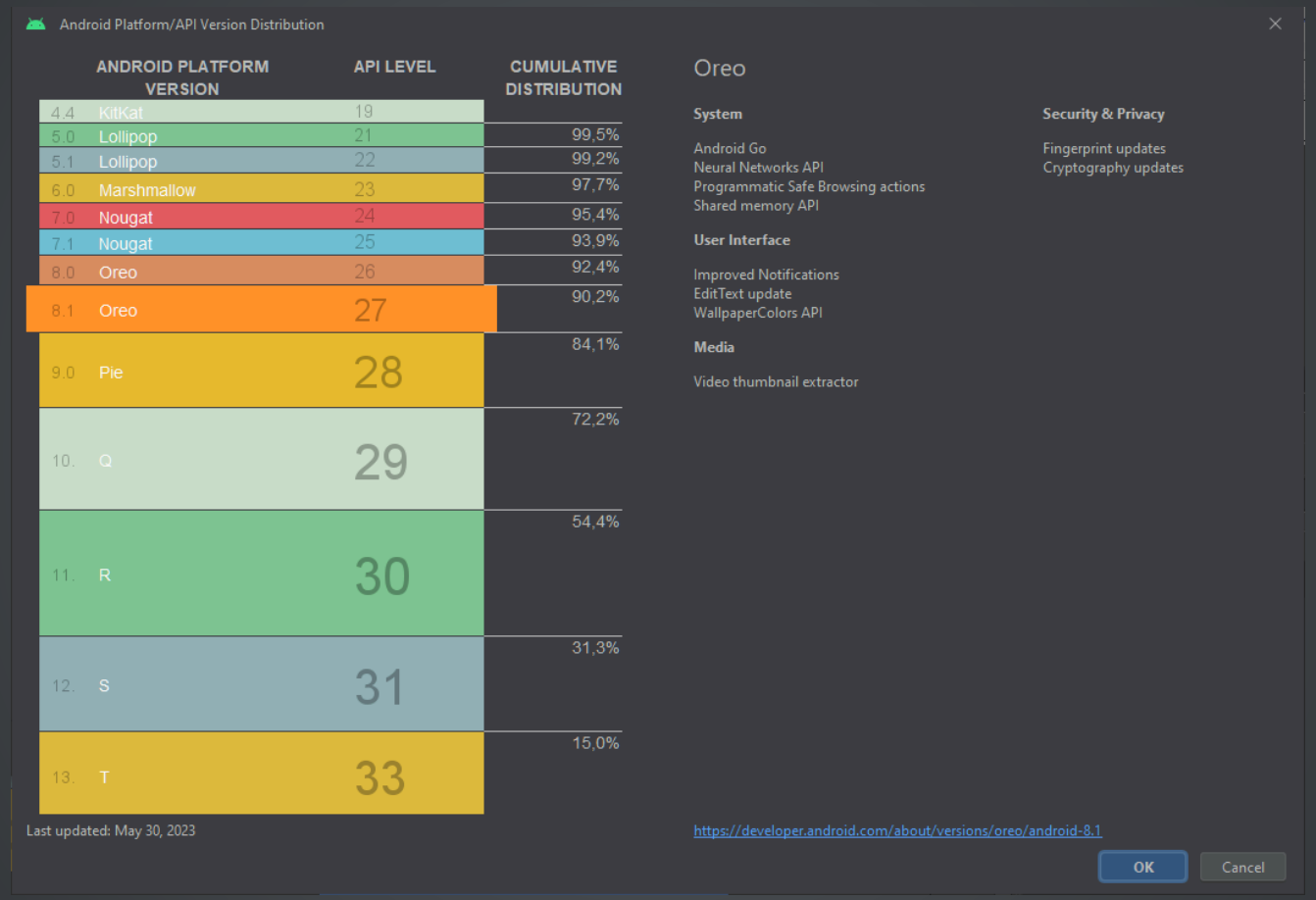
L'histoire d'Android



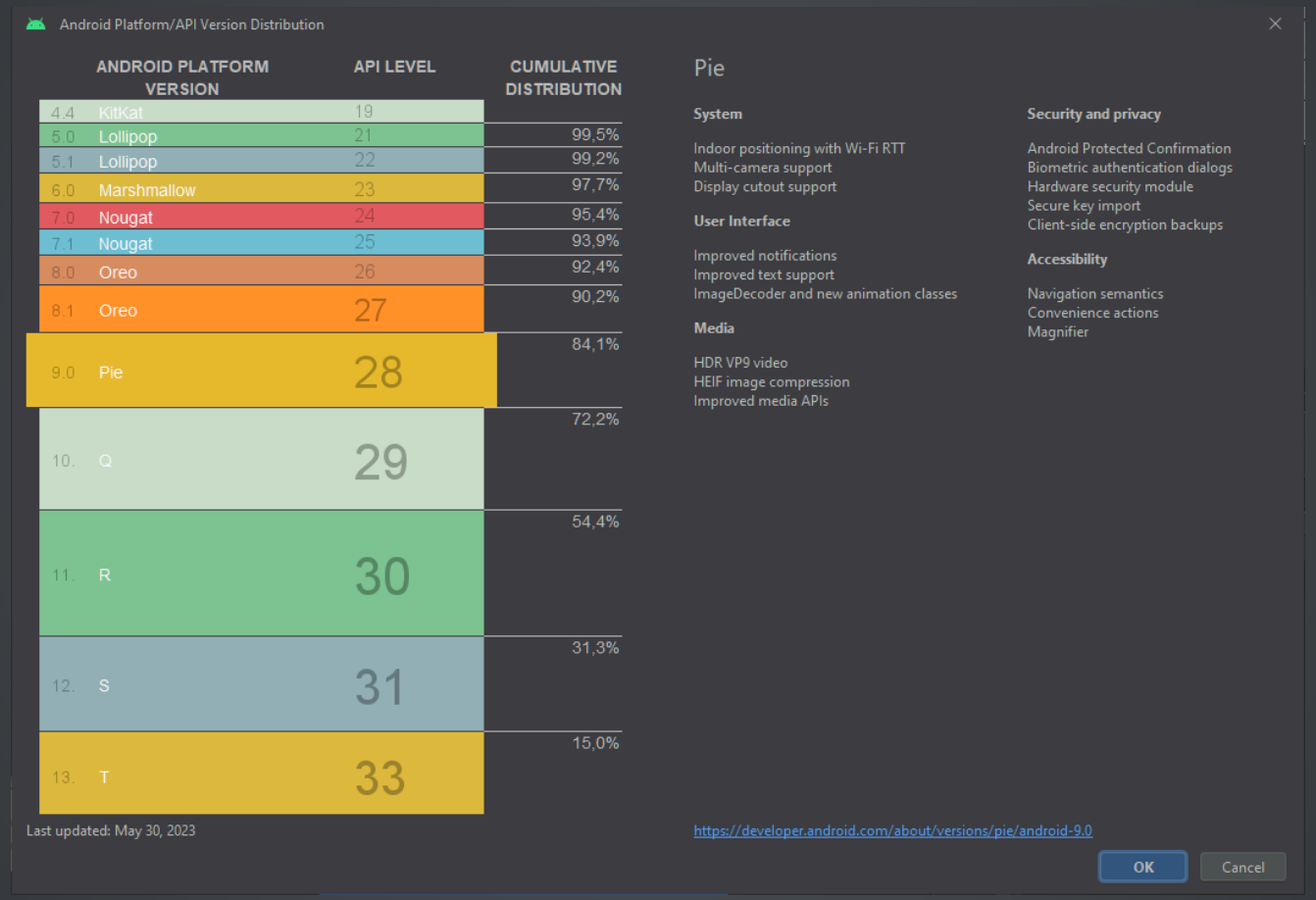
L'histoire d'Android



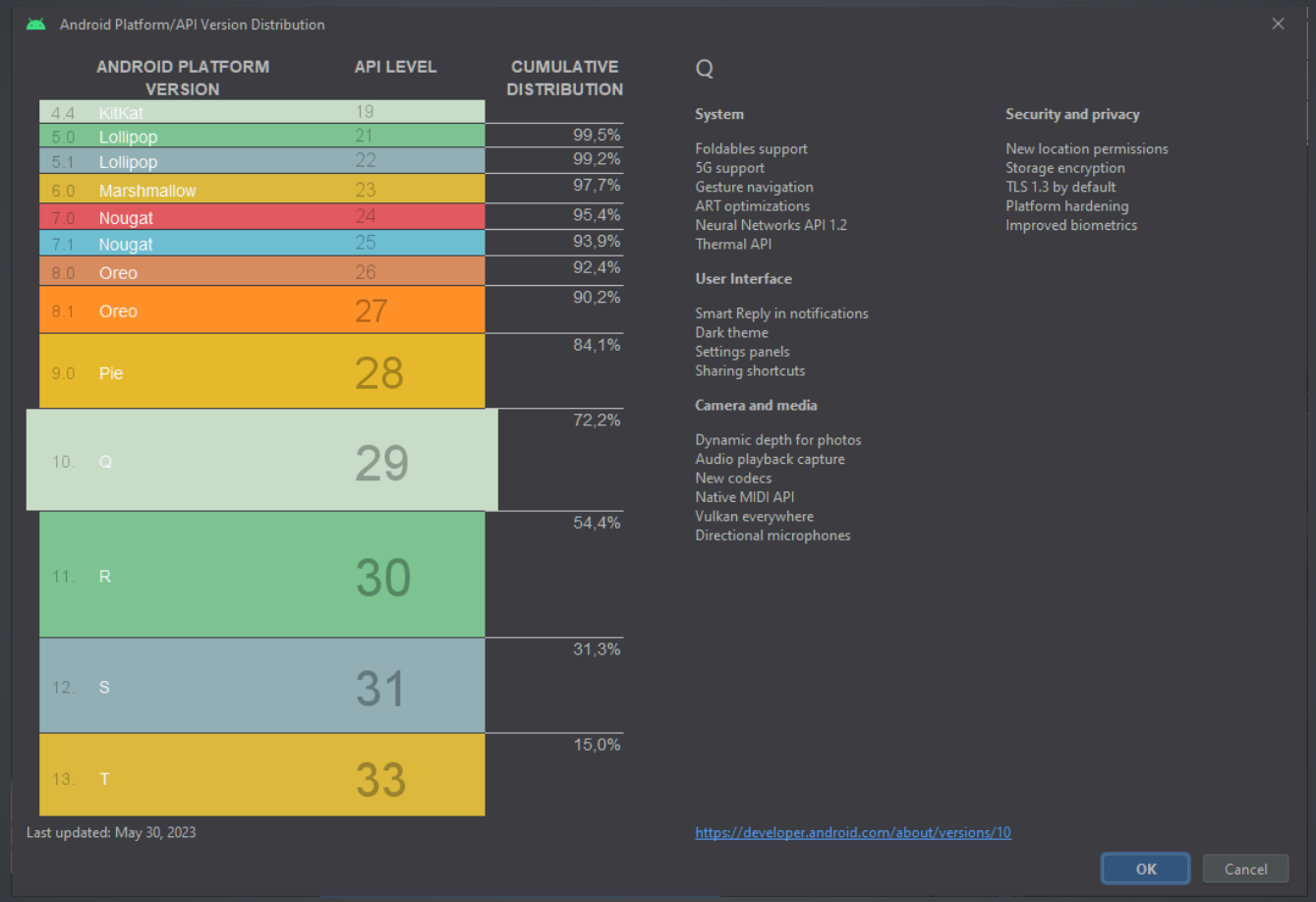
L'histoire d'Android



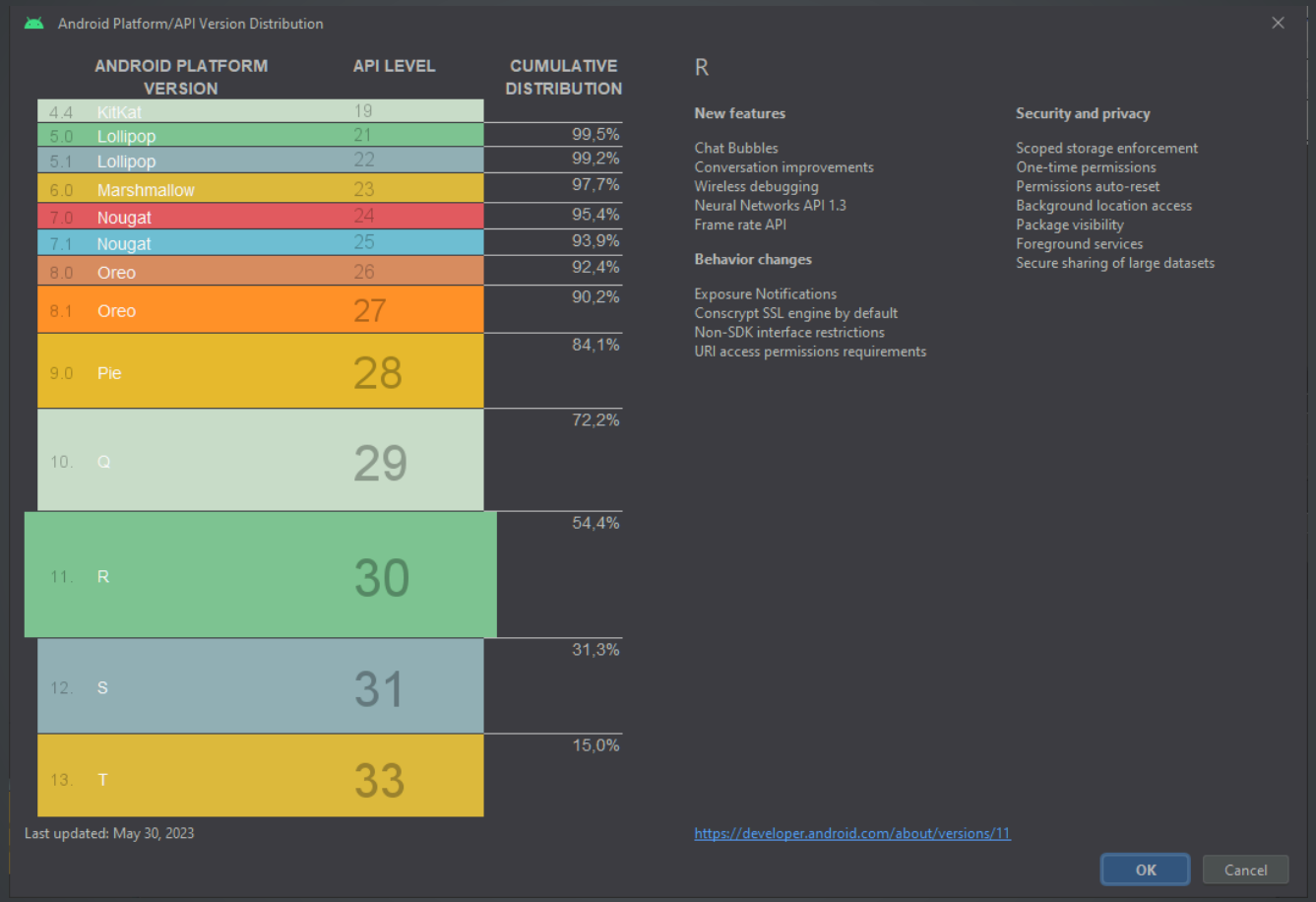
L'histoire d'Android



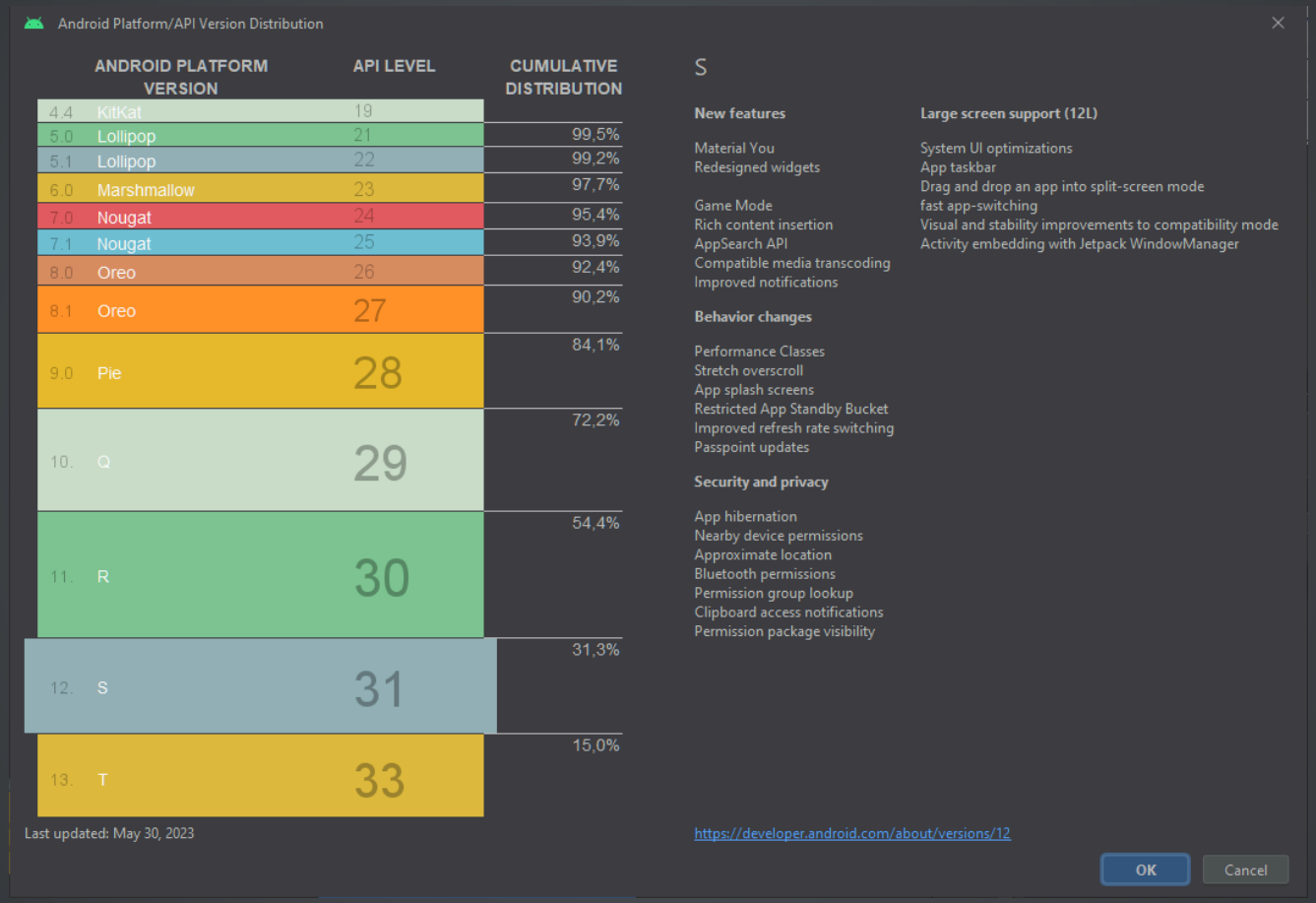
L'histoire d'Android



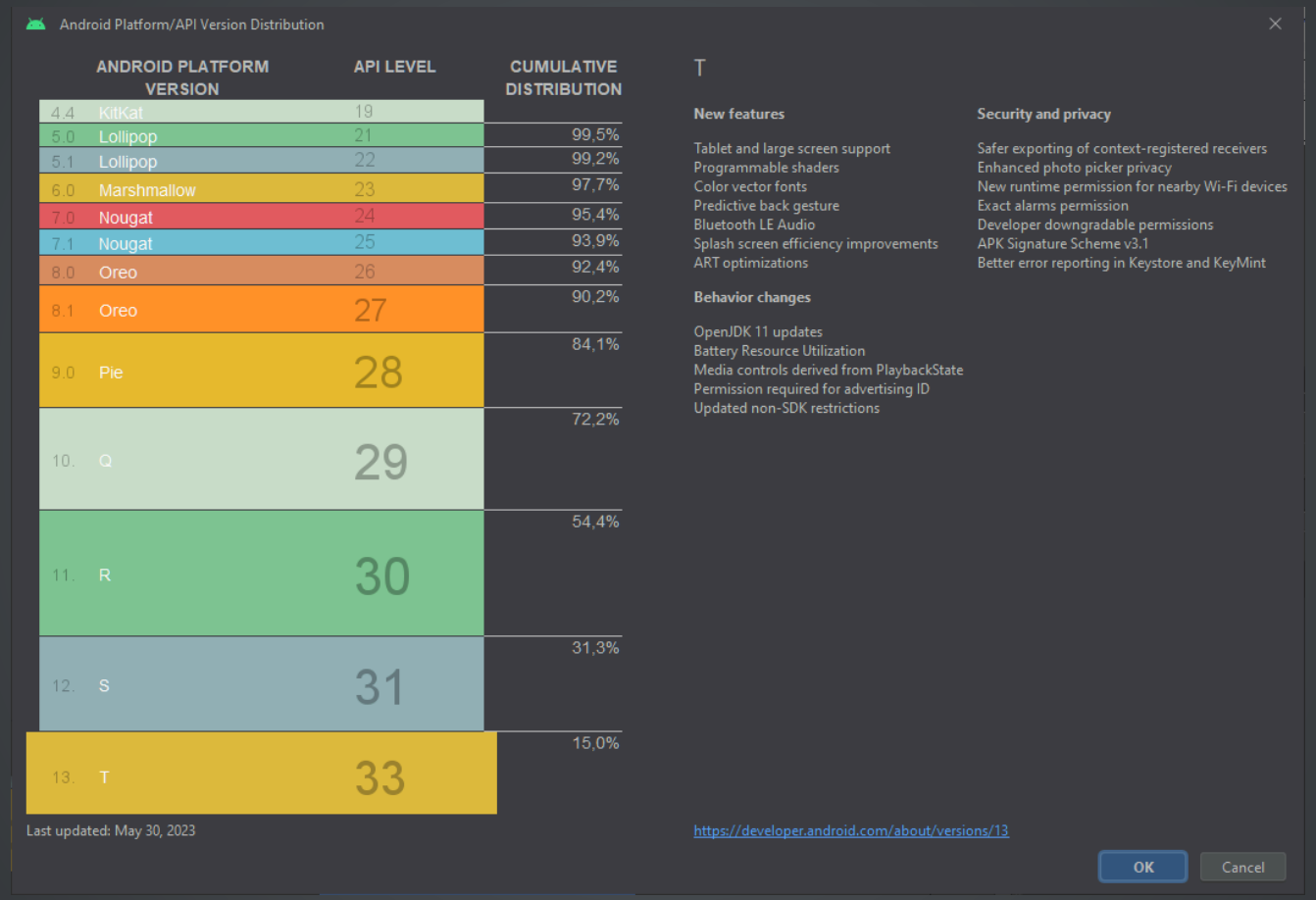
L'histoire d'Android



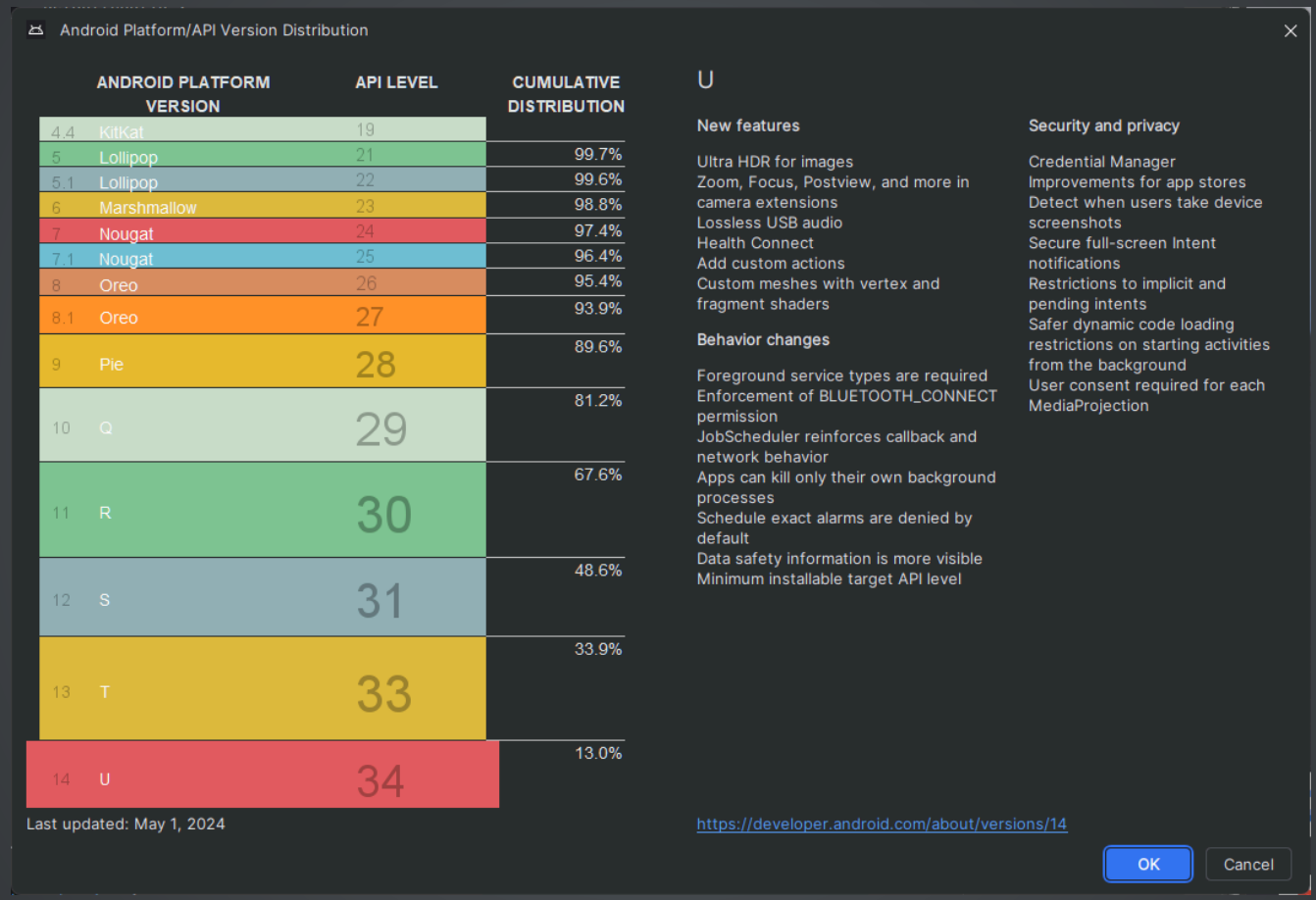
L'histoire d'Android



L'histoire d'Android



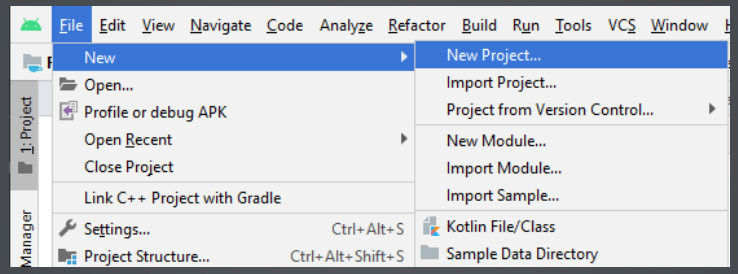
L'histoire d'Android



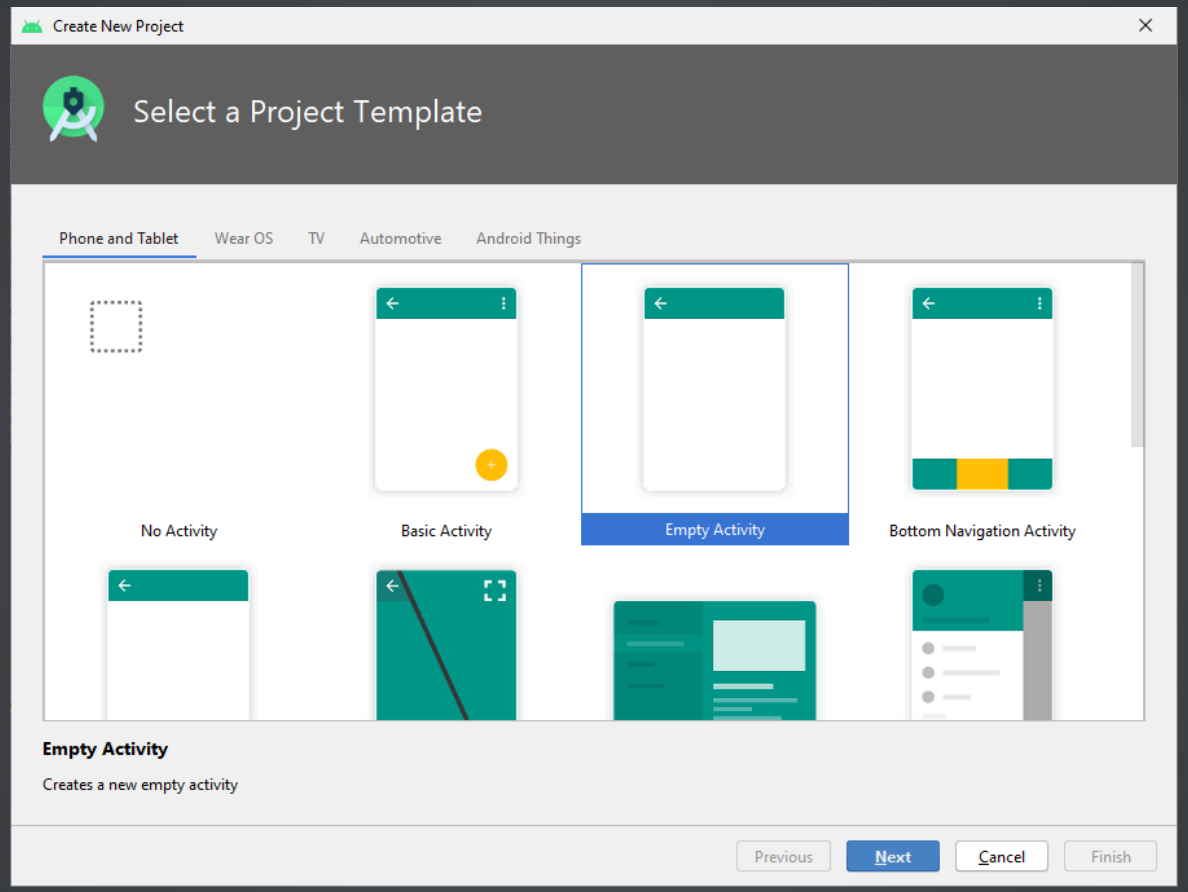
Les terminaux cibles

Pour obtenir la distribution des versions, vous devez créer un nouveau projet Android, puis sur l'écran "Configure Your Project" cliquons sur "Help me choose".

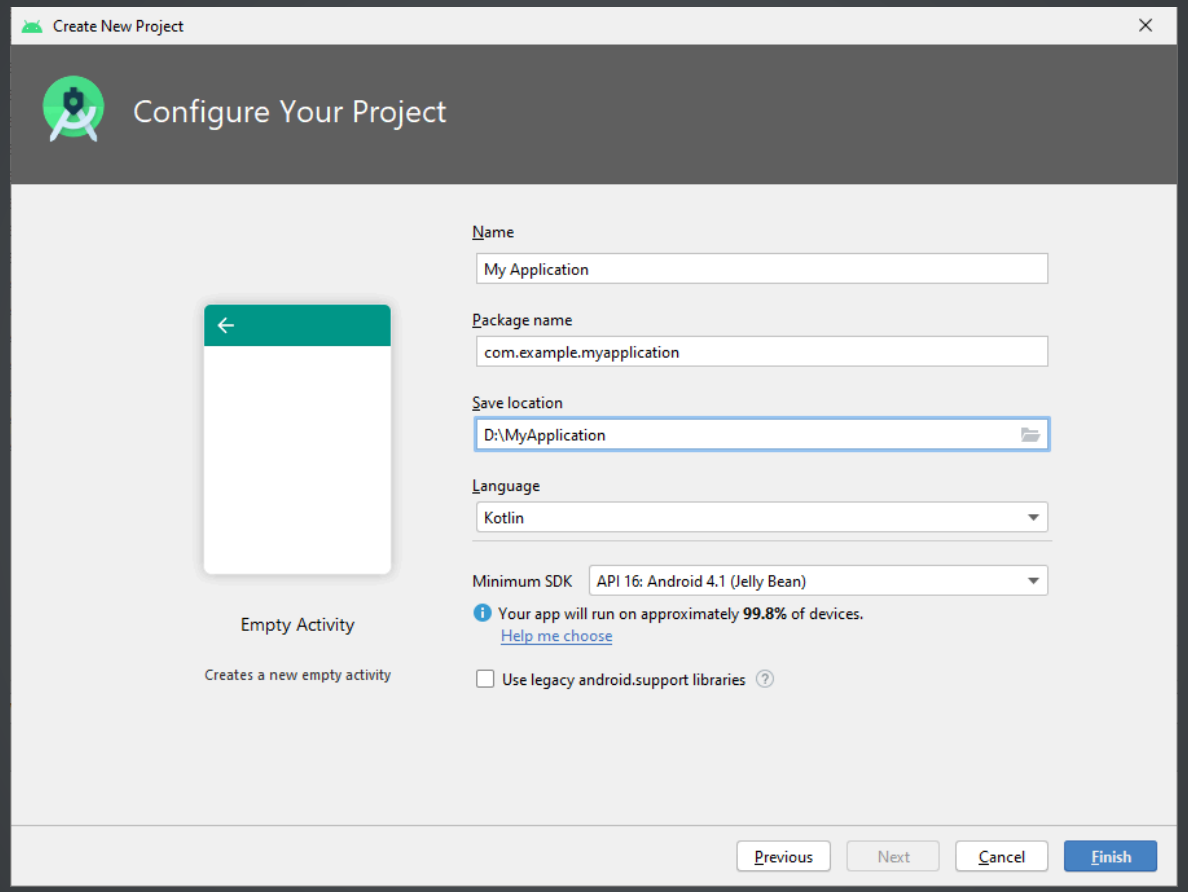
Les terminaux cibles



Les terminaux cibles



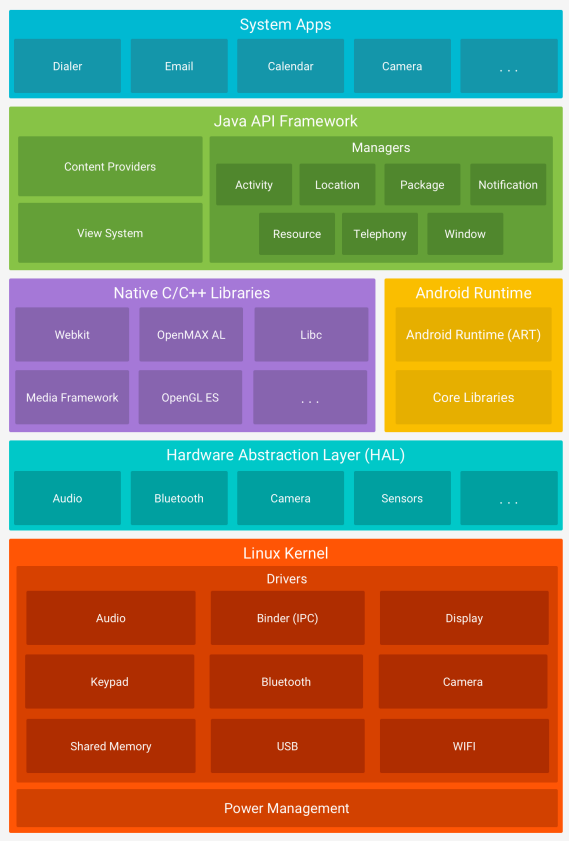
Les terminaux cibles



Développement Android

- Les concepts de base. Le cycle développement.
- Les classes de base du framework.
- Le projet sous Android Studio.
- L'émulateur du SDK. Les outils du SDK, SDK manager, AVD manager.
- L'utilisation des outils sous Android Studio : debugger, profiler, etc.
- Les paramètres du manifest.
- La production de l'application, la publication.

Les concepts de base



Source : https://developer.android.com/guide/platform

Les concepts de base

Optimisation

Les applications sur smartphone doivent veiller à optimiser tous les points, tels quel :

- Mémoire.
- Processeur.
- Radio (WiFi, 3G, BT).
- Graphique.

En effet tous ces points vont avoir une répercussion sur la consommation d'énergie, et donc sur la batterie.

Les concepts de base

Le cycle développement

Le développement d'une application peut suivre plusieurs schémas, nous allons en détailler une possibilité :

1. Lancement : toutes les applications commencent par une idée. Cette idée est généralement affinée jusqu'à constituer une base solide pour une application.
2. Conception : la phase de conception consiste à définir l'expérience utilisateur de l'application, comme la présentation générale, son fonctionnement, etc., ainsi que la conversion de cette expérience utilisateur en une interface utilisateur appropriée, généralement avec l'aide d'un infographiste.
3. Développement : généralement la phase la plus consommatrice de ressources, c'est la phase de création réelle de l'application.
4. Stabilisation : quand le développement est suffisamment avancé, l'assurance qualité commence généralement à tester l'application et les bogues sont corrigés. Le plus souvent, une application passe par une phase bêta limitée, durant laquelle un public plus large a la possibilité de l'utiliser, de fournir des commentaires et d'obtenir des modifications.
5. Déploiement

Les concepts de base

Conception des interfaces utilisateur

Afin de définir notre interface utilisateur, Google fournit des guides pour bien réaliser cette tâche : https://developer.android.com/design/index.html
Il nous appartiendra à bien qualifier notre cible (type de terminaux : smartphone, tablette, TV, montre, ordinateur de voiture, ...).

Les concepts de base

Phases de développement

Nous pouvons passer par plusieurs phases pour réaliser l'application finale :

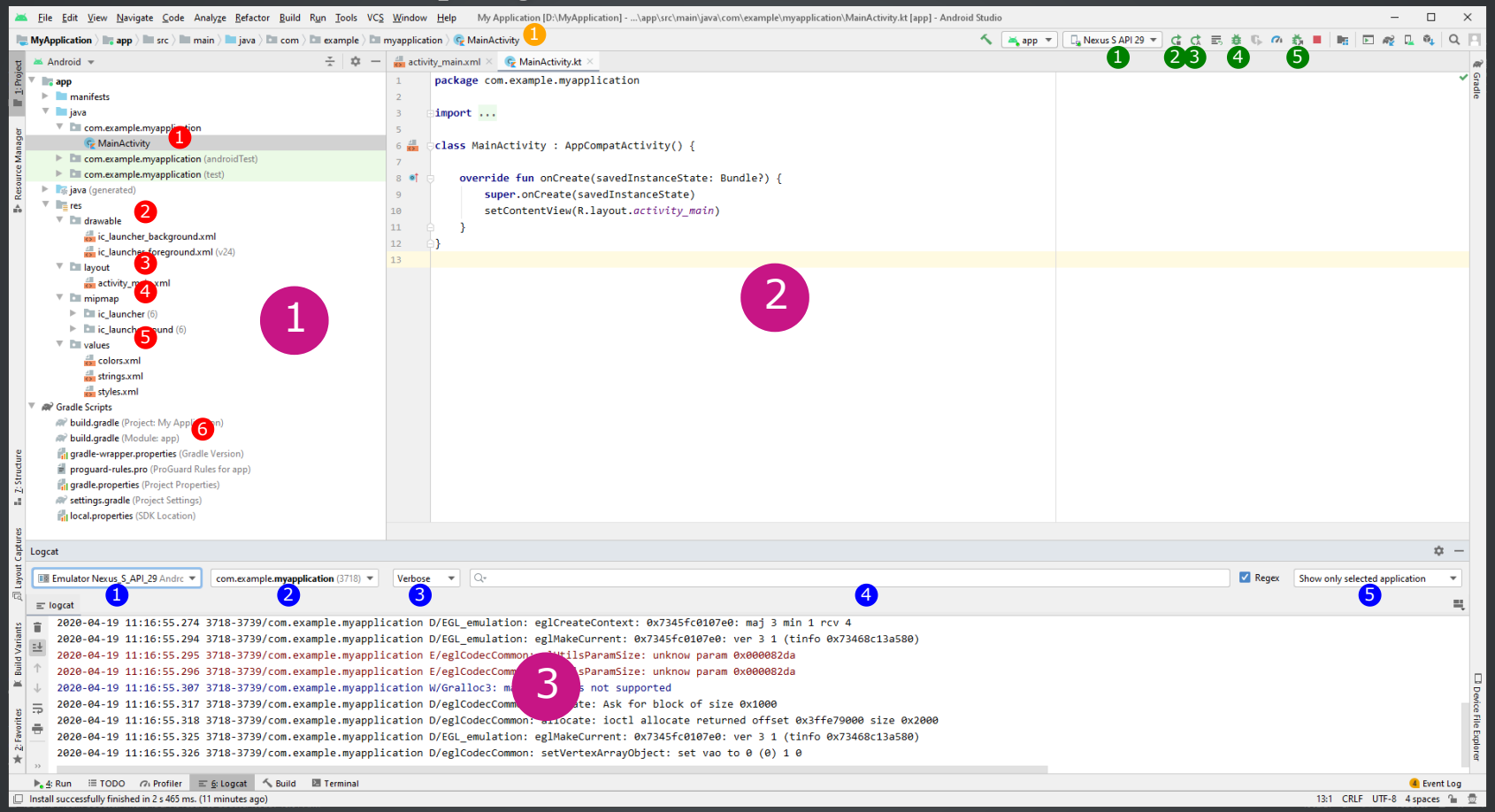
- Prototype/MVP : l'application est toujours en phase de preuve de concept, et seules les fonctionnalités principales ou des parties spécifiques de l'application fonctionnent. Des bogues majeurs y sont présents.
- Alpha : les fonctionnalités principales sont généralement entièrement présentes dans le code (qui est généré, mais pas entièrement testé). Des bogues majeurs sont encore présents, des fonctionnalités périphériques peuvent ne pas encore être présentes.
- Bêta : la plupart des fonctionnalités sont maintenant terminées, une partie des tests et de la correction des bogues a été effectuée. Des problèmes majeurs connus peuvent encore être présents.
- Version Release Candidate : toutes les fonctionnalités sont terminées et testées. Sauf si de nouveaux bogues apparaissent, l'application est candidate à la publication.

Les classes de base du framework

Plusieurs classes de bases sont disponibles dans le Framework, nous les détaillerons par la suite :

- Activity
- Fragment (3.0+)
- Service
- Content providers
- Broadcast receivers
- Intent

Le projet sous Android Studio



Le projet sous Android Studio

Les grandes parties en violet :
1. La liste des fichiers.
2. La zone d'édition.
3. Les logs.

Le projet sous Android Studio

Détail des points en rouge :

1. Le code de notre application.
2. Le répertoire des ressources.
3. Le répertoire contenant les mises en forme des écrans (les layouts).
4. Le répertoire contenant les versions des icônes de l'application.
5. Le répertoire contenant les autres ressources.

Le projet sous Android Studio

Détail des points en orange :

1. Le chemin complet du fichier courant sélectionné.

Le projet sous Android Studio

Détail des points en vert :

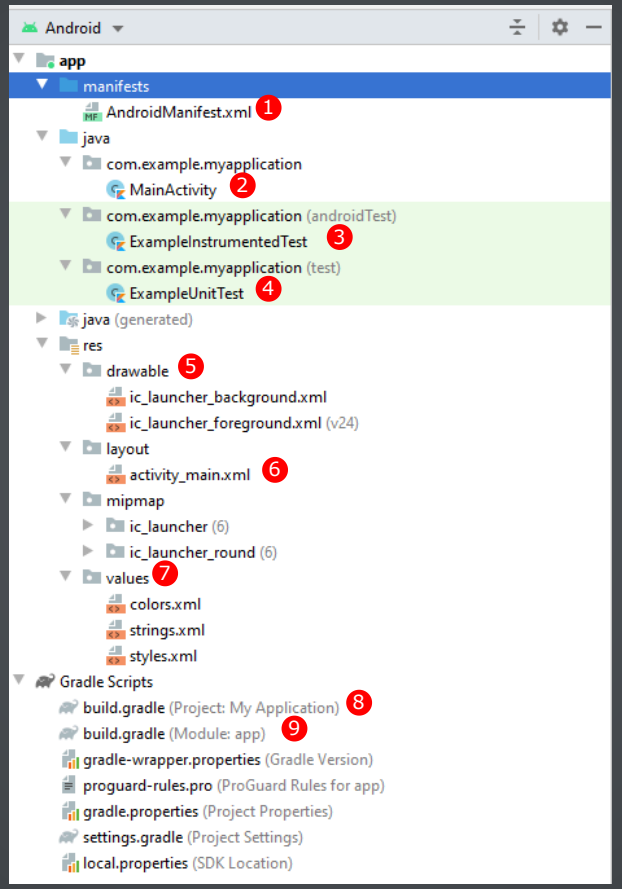
1. Le choix de la cible pour lancer le programme.
2. Relancer l'application.
3. Relancer l'activity.
4. Relancer en mode débug.
5. Attacher à la volée le debugger.

Le projet sous Android Studio

Détail des points en bleu, il s'agit du Logcat :

1. Nous pouvons choisir le device sur lequel se connecter.
2. Sélectionner le process pour filtrer les messages.
3. Sélectionner le type de log des messages.
4. Filtrer les messages.
5. Activer le filtre.

Le projet sous Android Studio



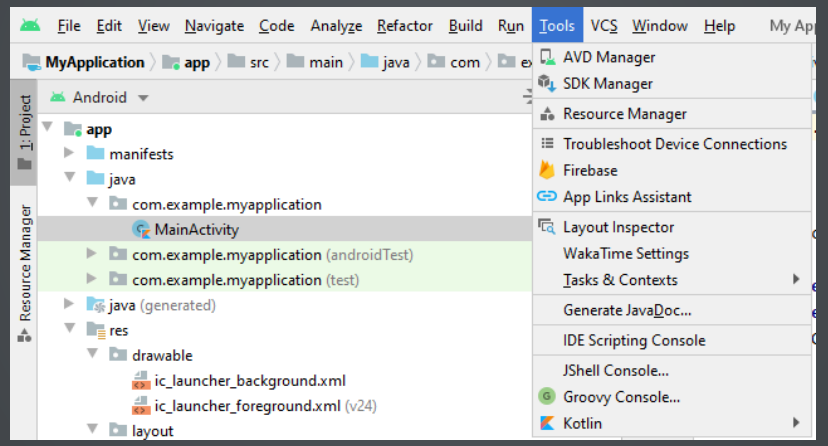
Le projet sous Android Studio

Les fichiers principaux rouge :

1. Le manifest (AndroidManifest.xml), qui contient la carte d'identité de notre application.
2. Le code de l'application.
3. Le code des tests graphiques.
4. Le code des tests unitaires.
5. Les ressources graphiques.
6. Les écrans (layouts) de notre application.
7. Les ressources (texte, couleurs, dimensions, styles, ...) de notre application.
8. Le fichier de configuration global du système de compilation (gradle).
9. Le fichier de configuration de notre projet (celui qui sera le plus souvent modifié).

Émulateur

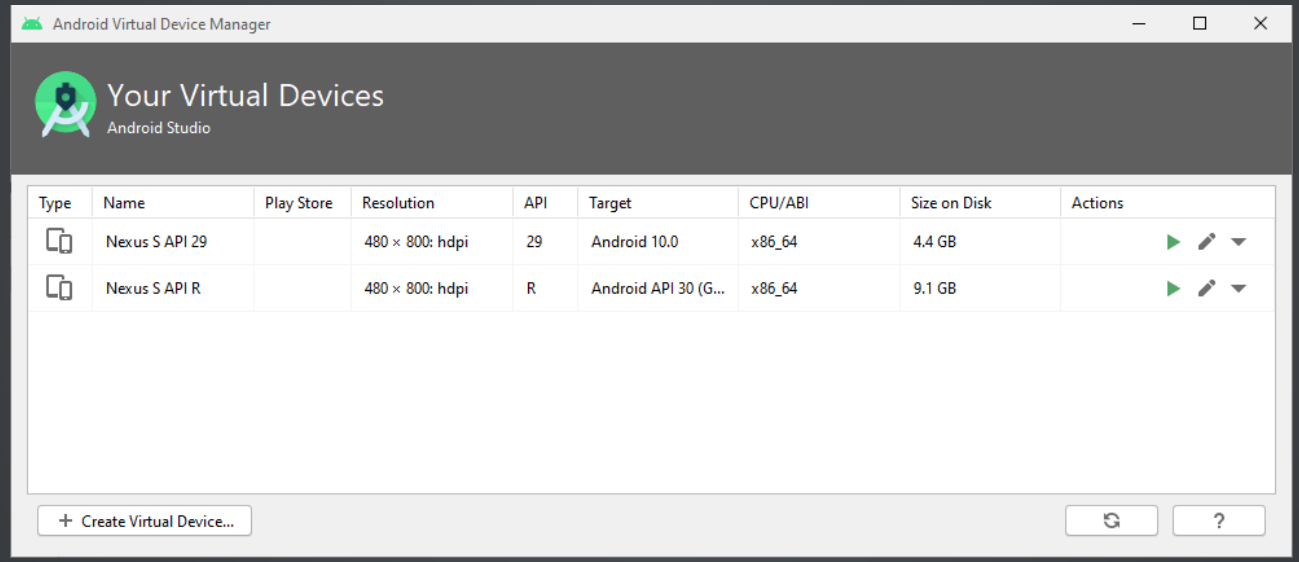
Lançons l'écran de gestion des machines virtuelles, avec le menu : `Tools/AVD Manager`



Émulateur

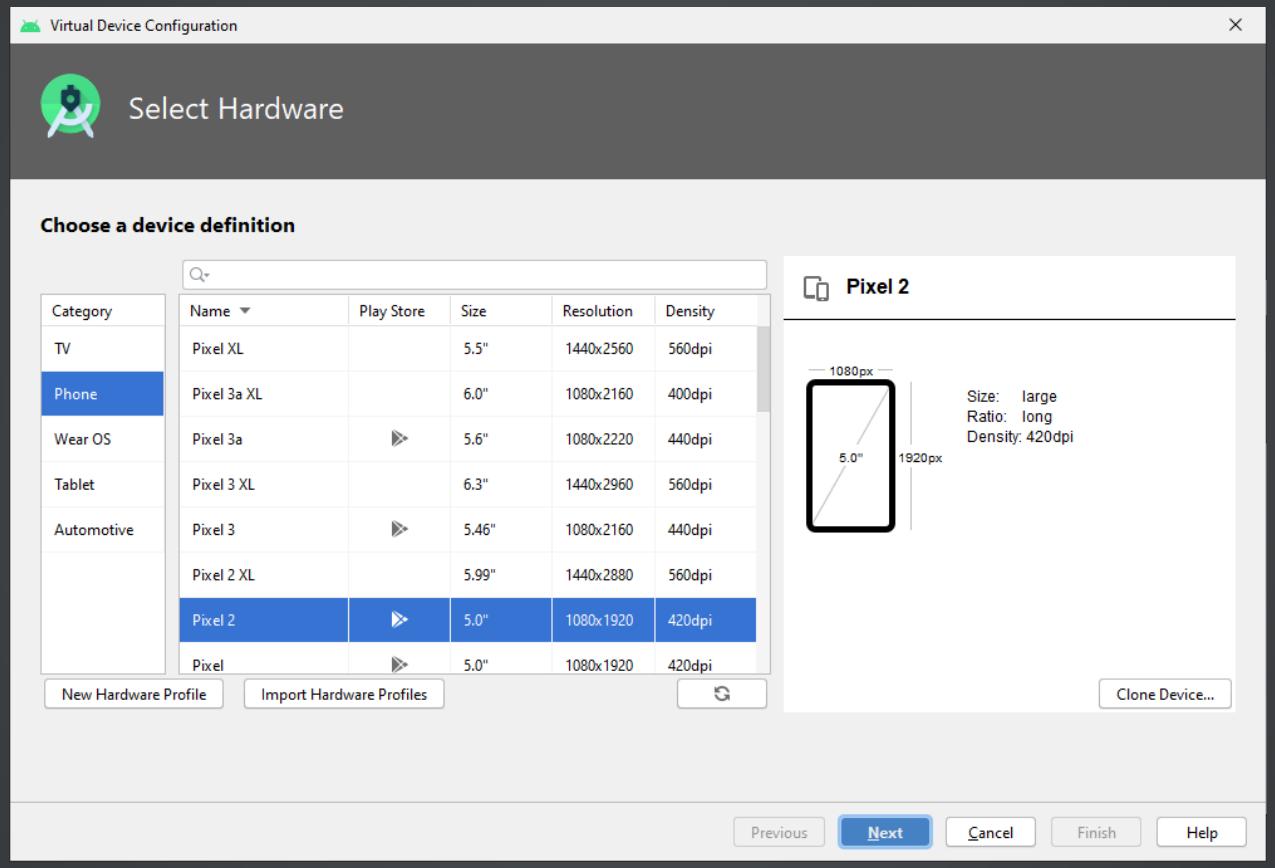
Choix de la machine virtuelle

Sur cet écran, nous pouvons choisir la machine virtuelle à lancer/configurer/supprimer.
Nous pouvons aussi en créer une nouvelle, en cliquant sur le bouton "Create Virtual Device..."



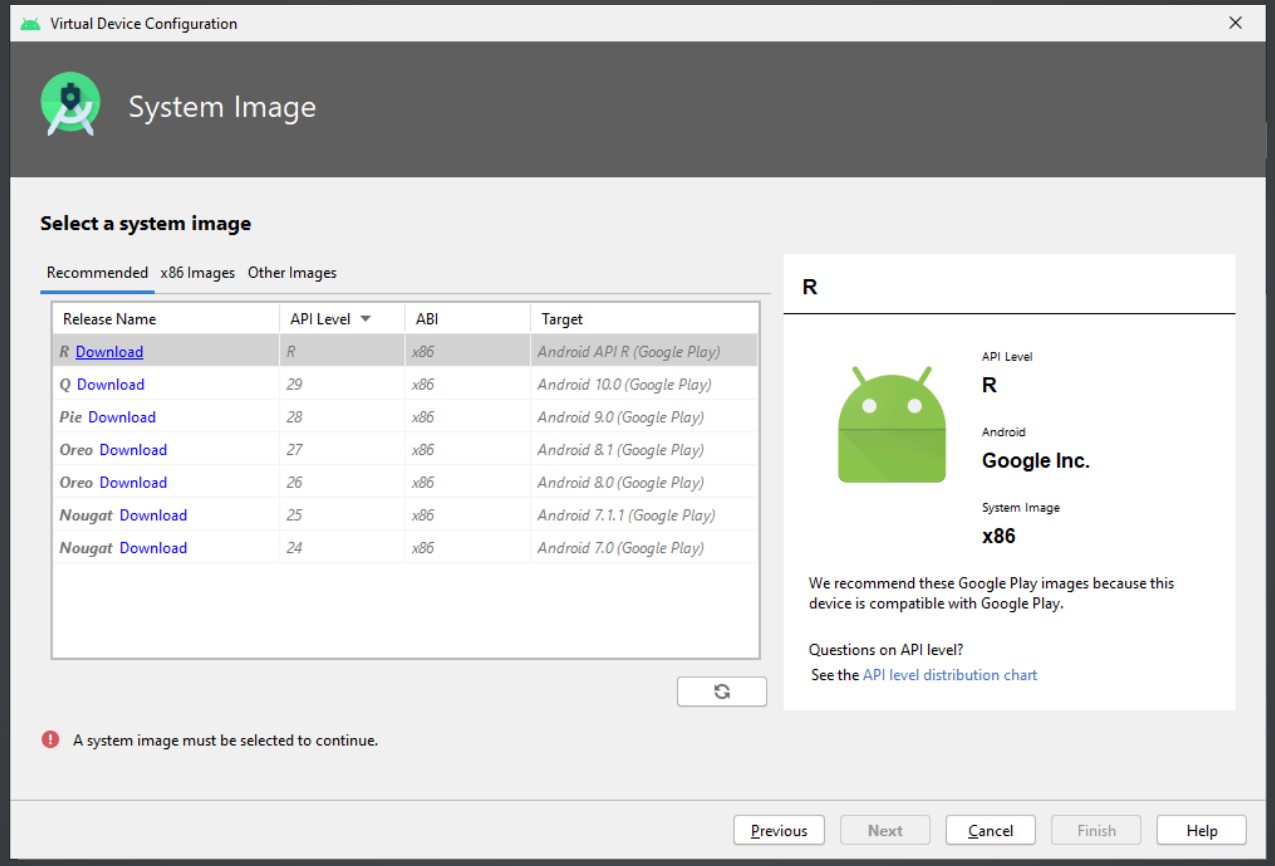
Émulateur

Choisissons le modèle de device que nous voulons émuler :



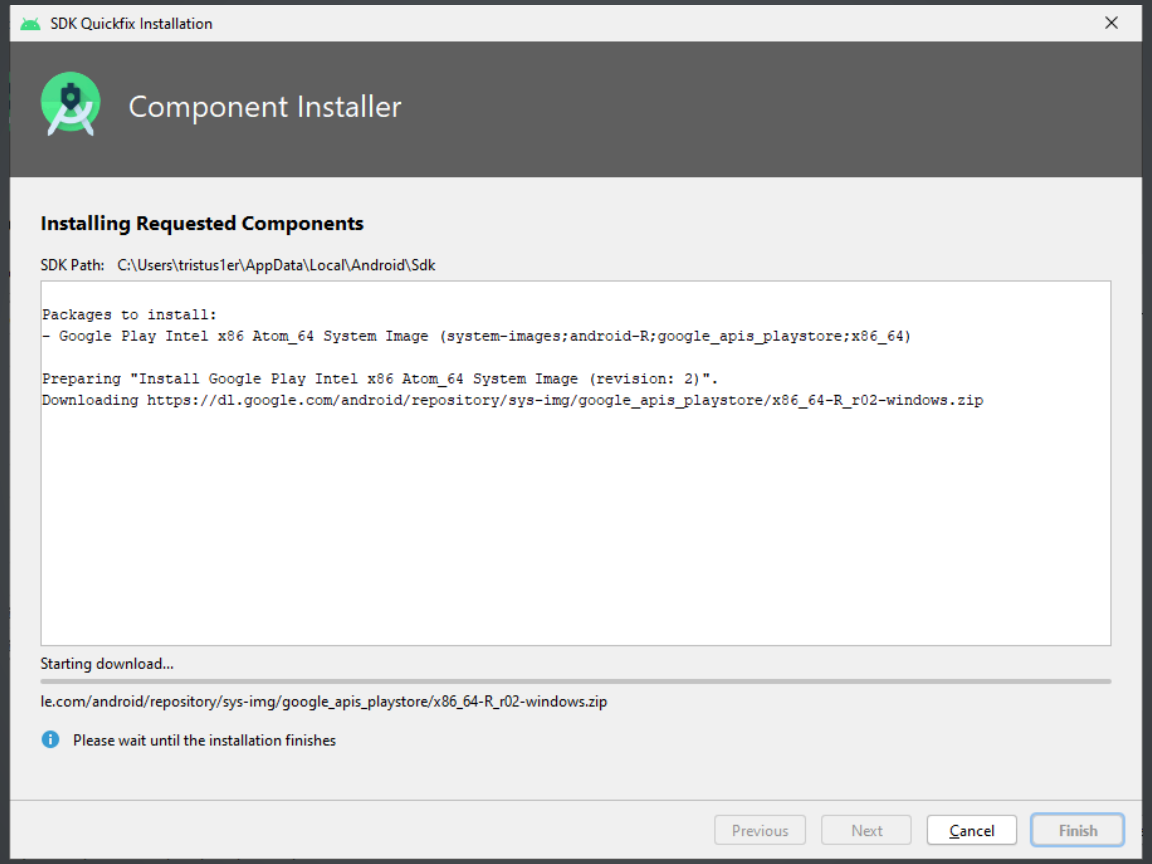
Émulateur

Choisissons la version d'Android :



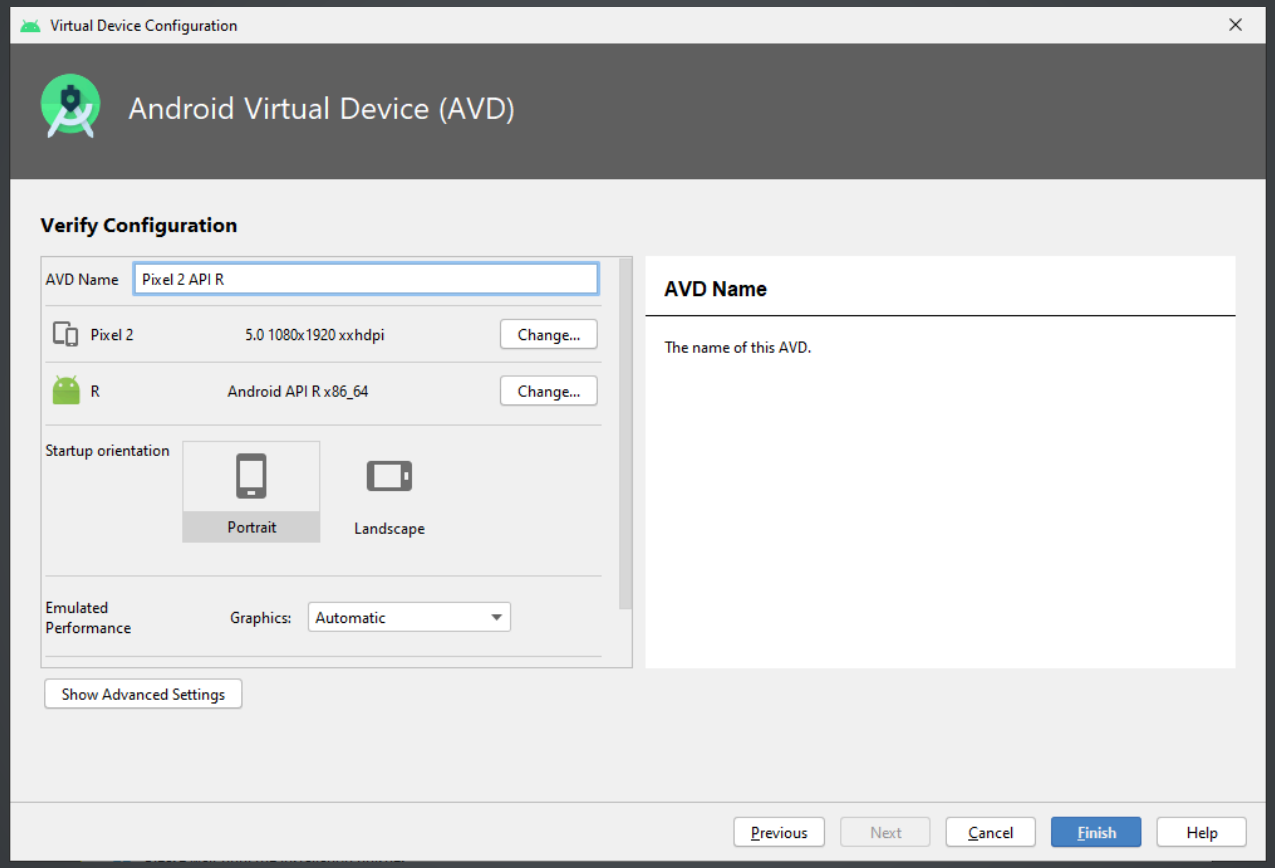
Émulateur

En cas de besoin, nous pouvons directement télécharger une image :



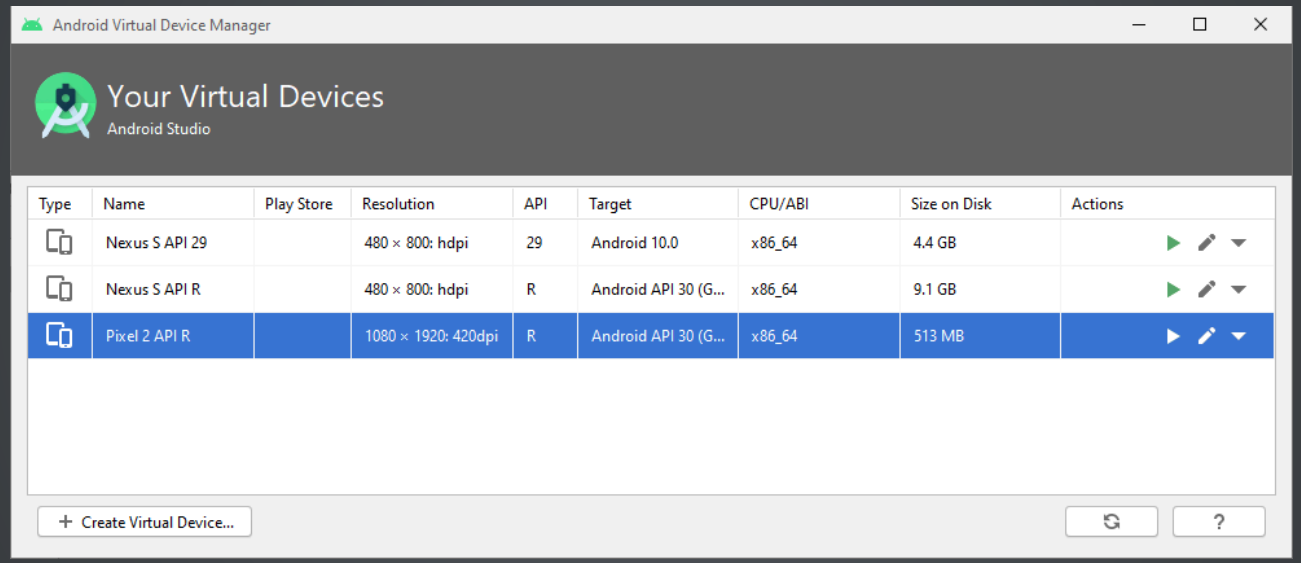
Émulateur

Donnons un nom à notre machine virtuelle, nous laissons les paramètres par défaut :



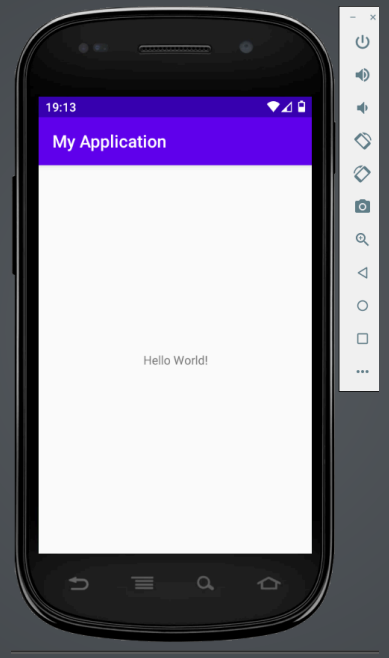
Émulateur

Notre machine virtuelle est maintenant disponible, lançons la :



Émulateur

L'émulateur permet de faire fonctionner nos applications sur des versions d'Android que nous n'avons pas forcément sur notre smartphone :

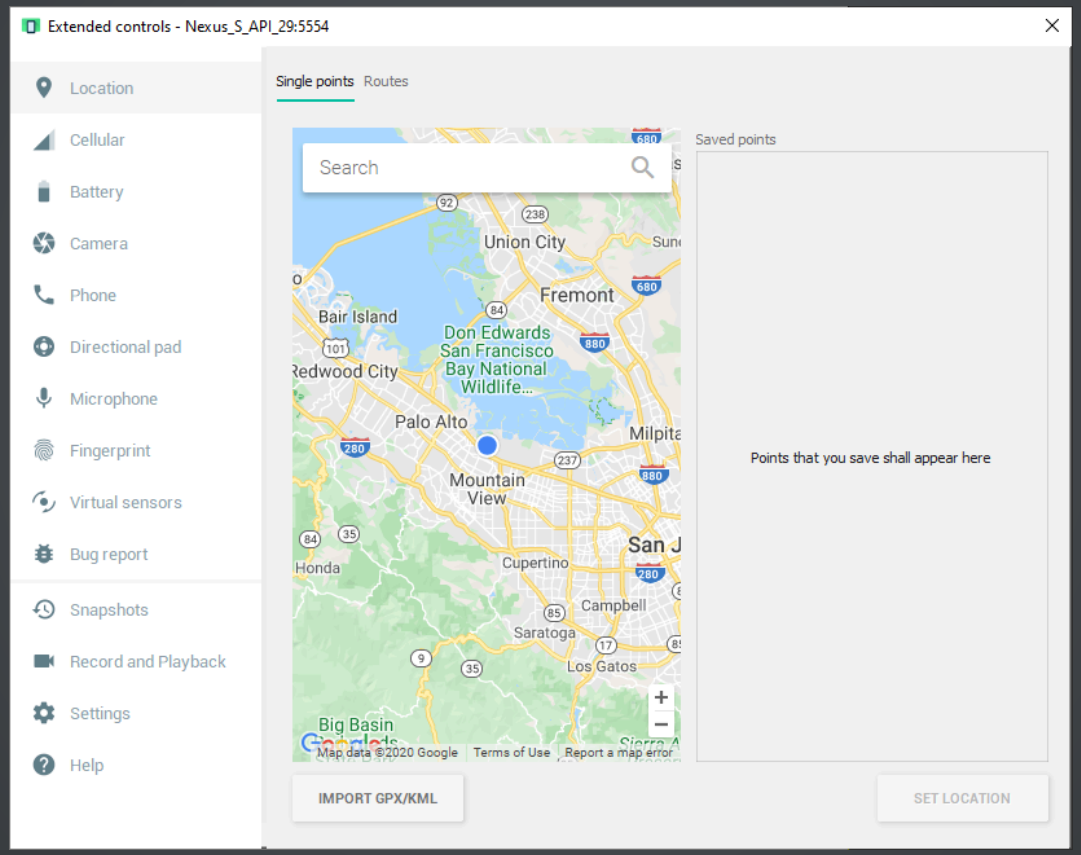


Cliquons sur les "..."

Émulateur

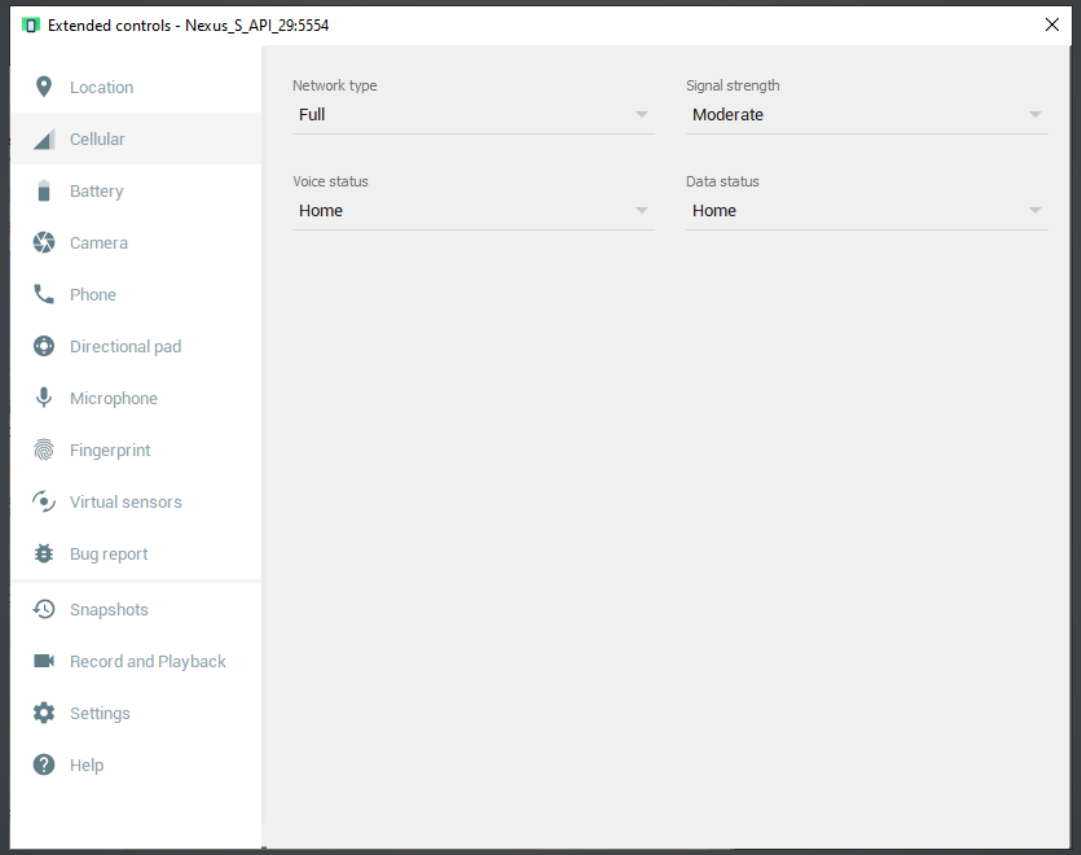
Nous pouvons :

Gérer la position du device directement sur une carte (nouvelle fonctionnalité Android Studio 3.6) :



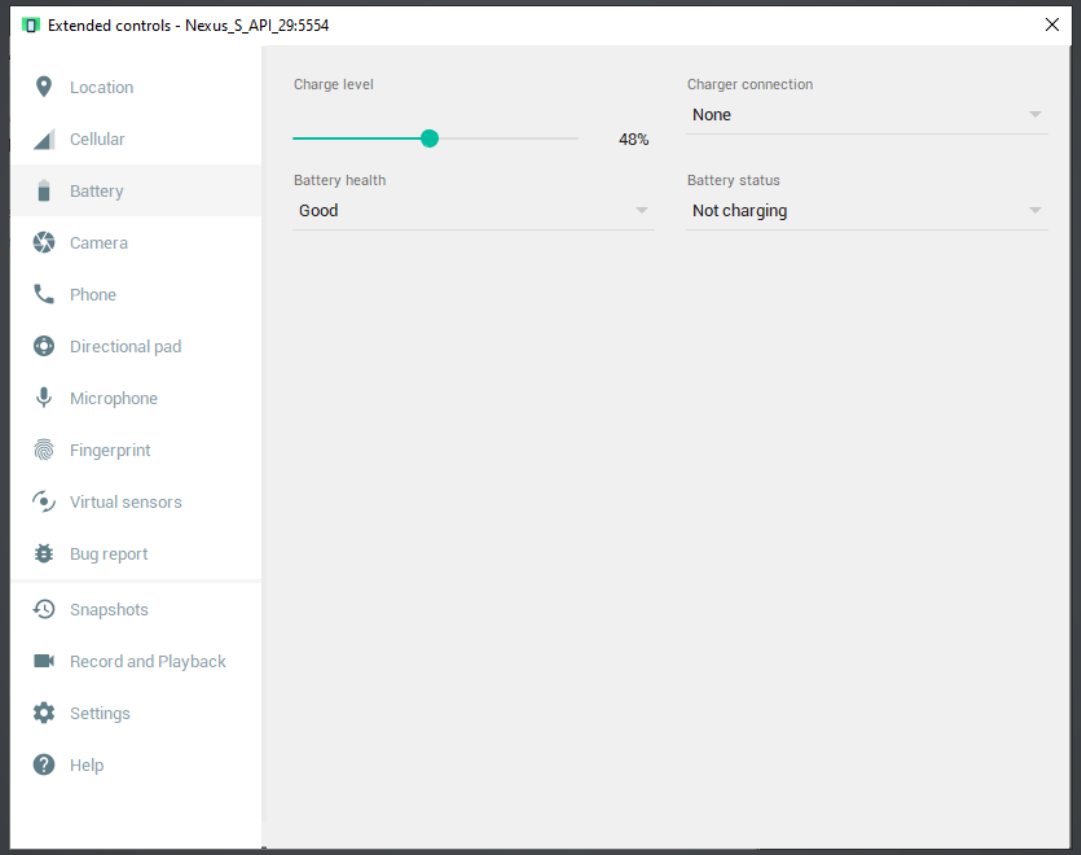
Émulateur

Gérer la réception réseau :



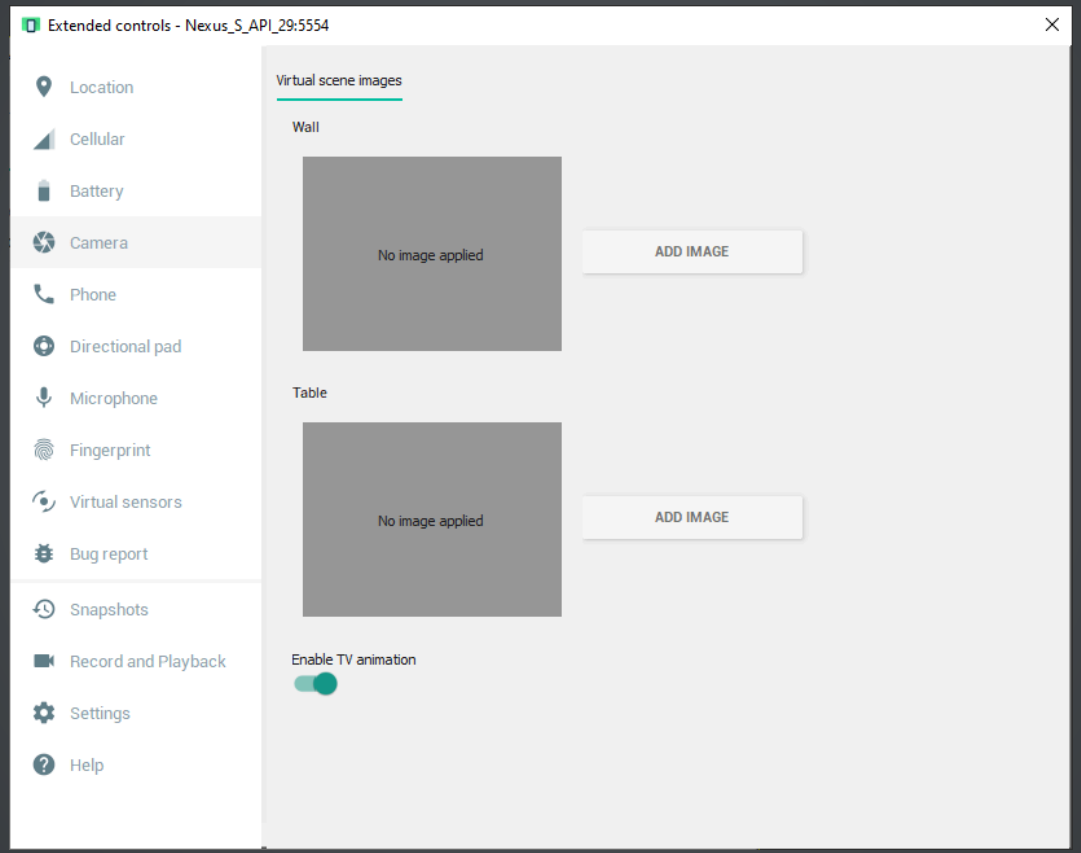
Émulateur

Définir le niveau de batterie :



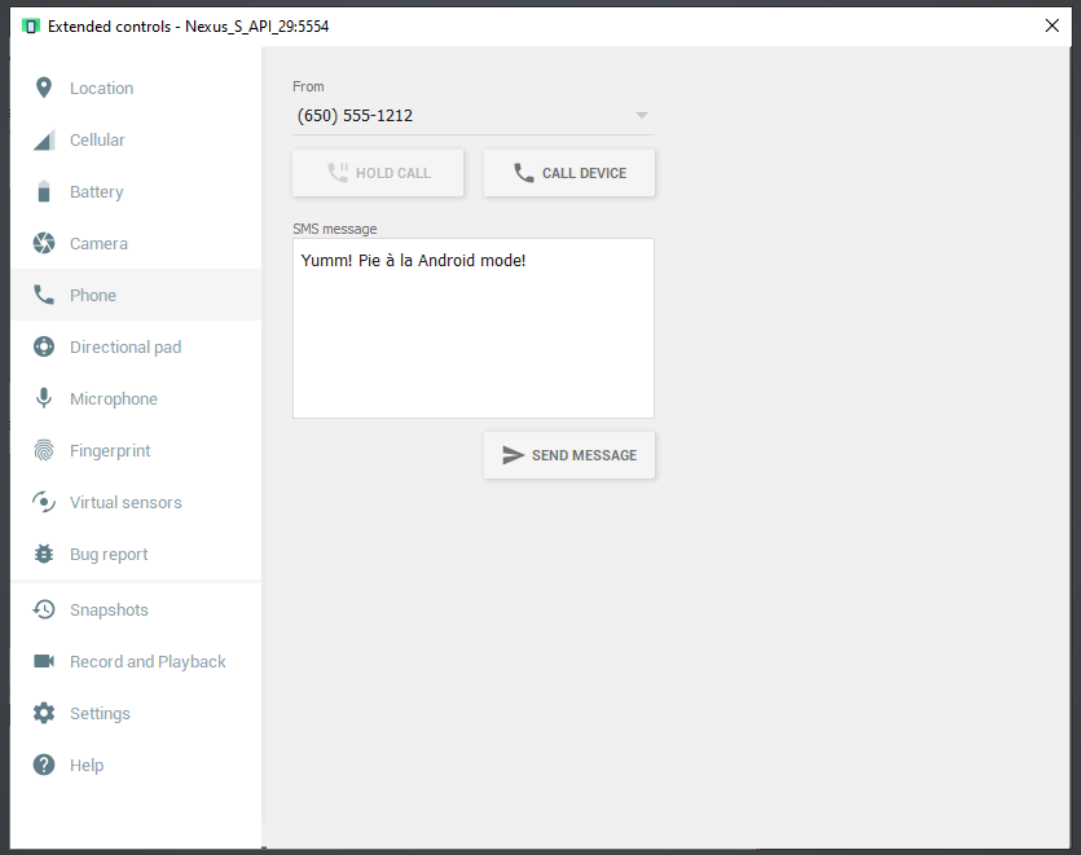
Émulateur

Définir ce que "voit" la/les caméra(s) :



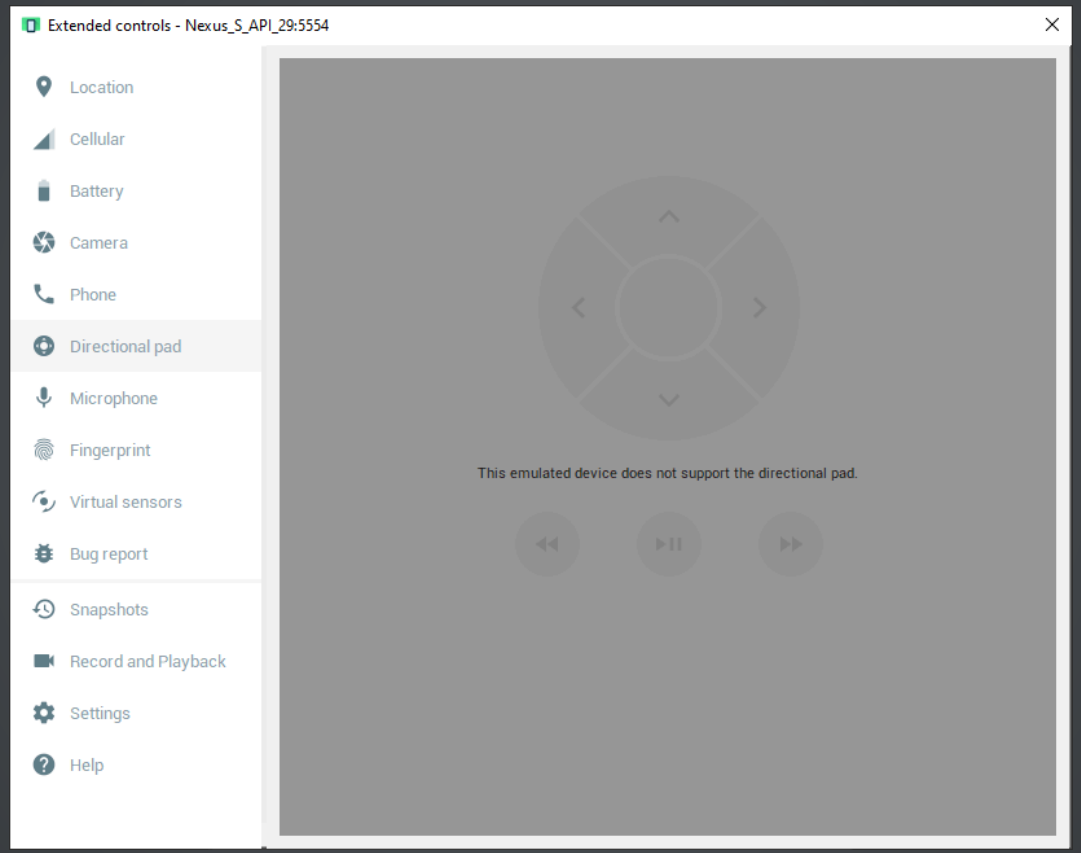
Émulateur

Lancer des appels téléphoniques
et envoyer des SMS :



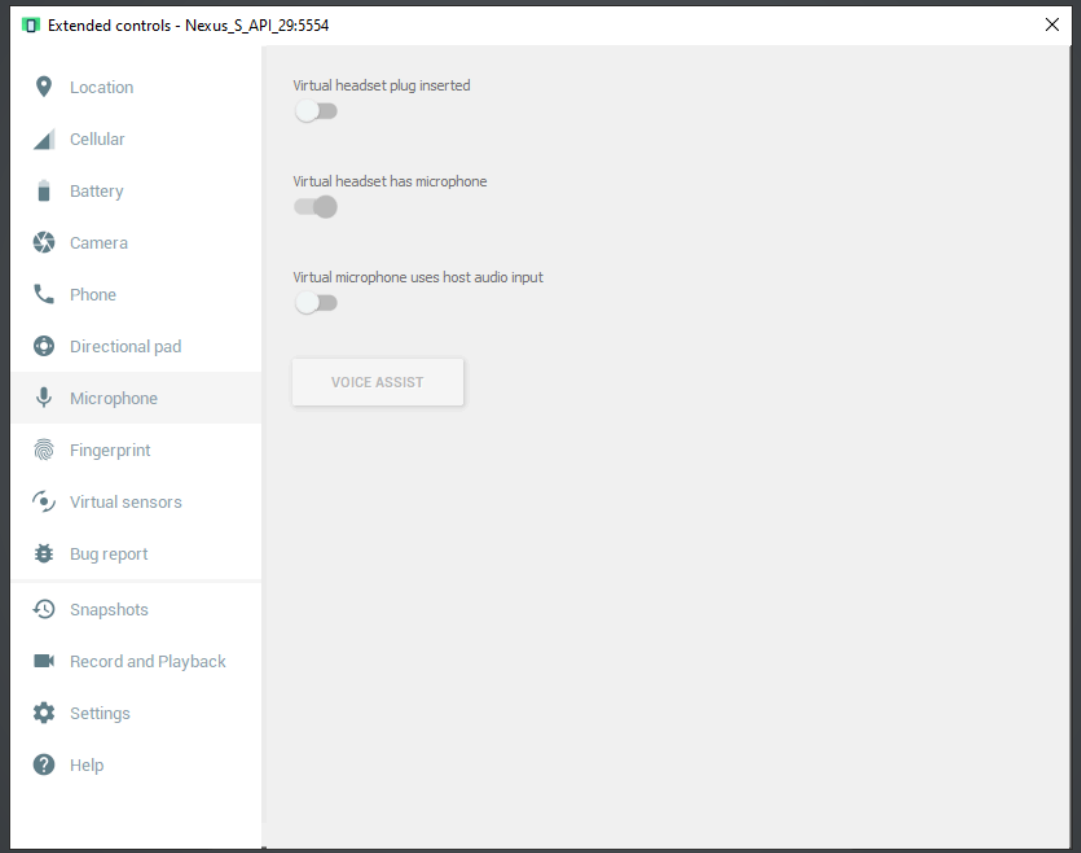
Émulateur

Émuler le pad :



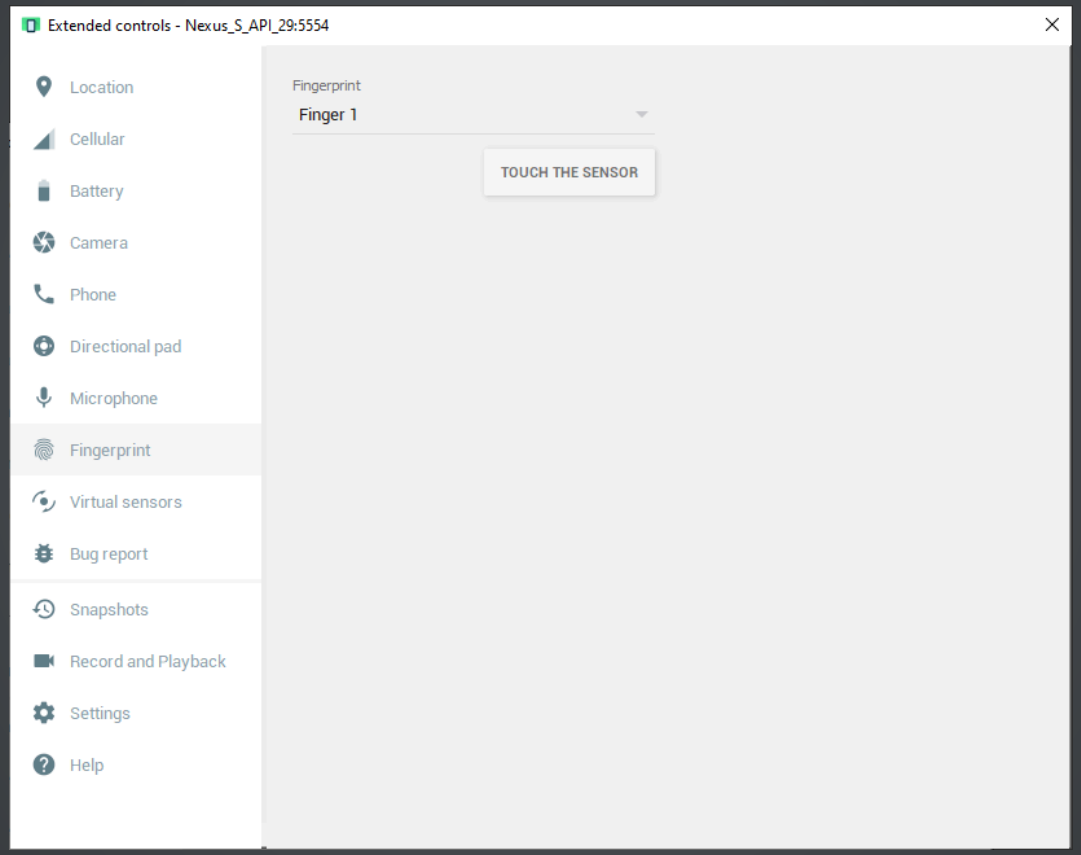
Émulateur

Émuler la gestion du micro :



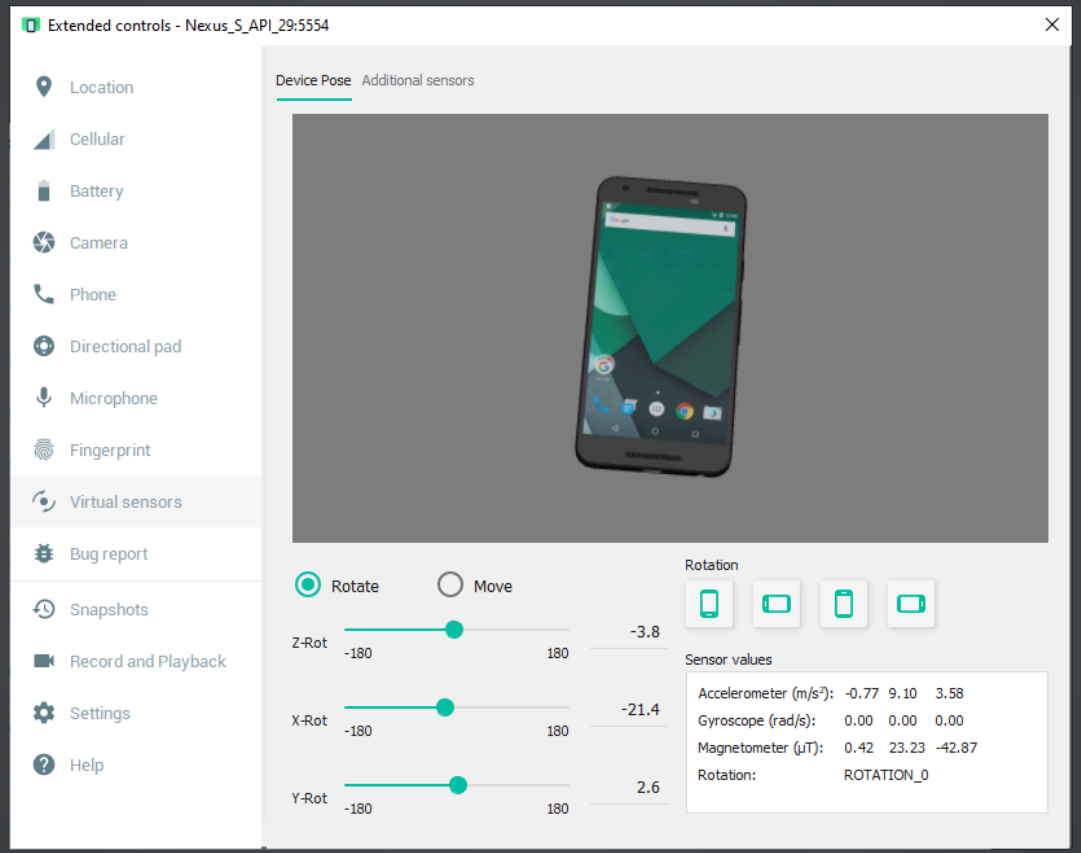
Émulateur

Gérer les empruntes tactiles :



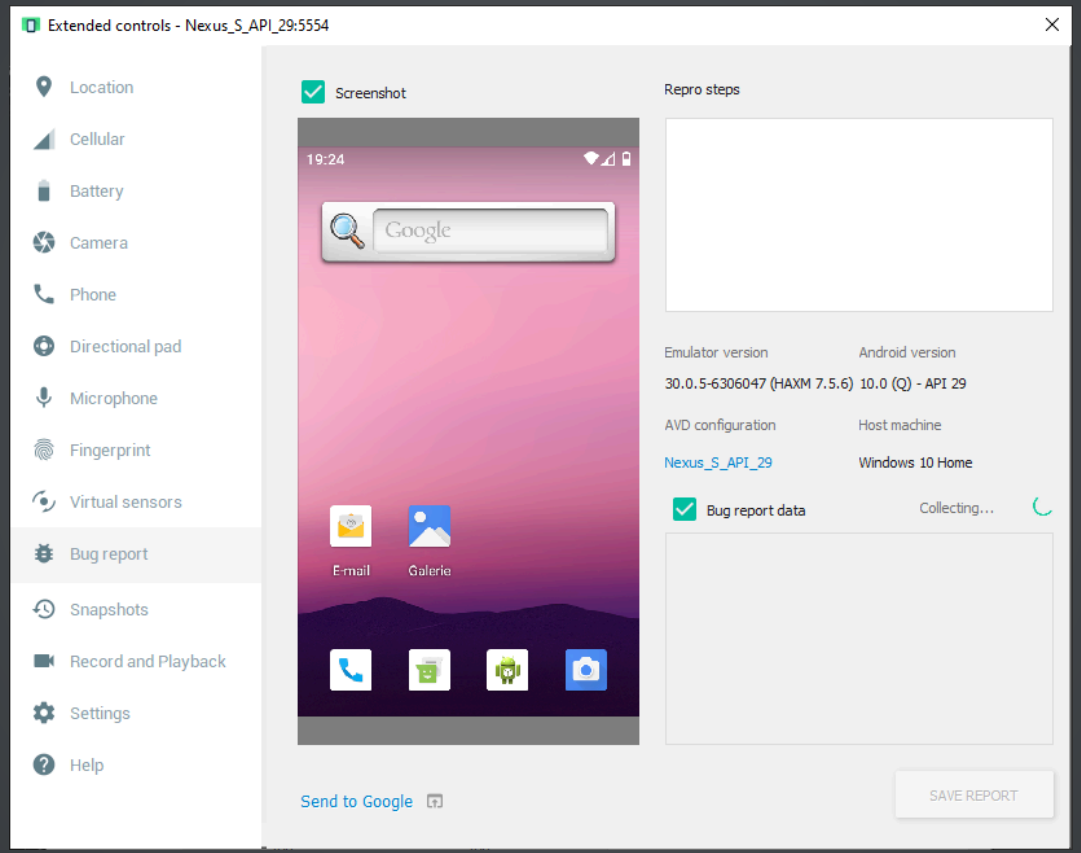
Émulateur

Émuler les capteurs (rotations, et accélération) :



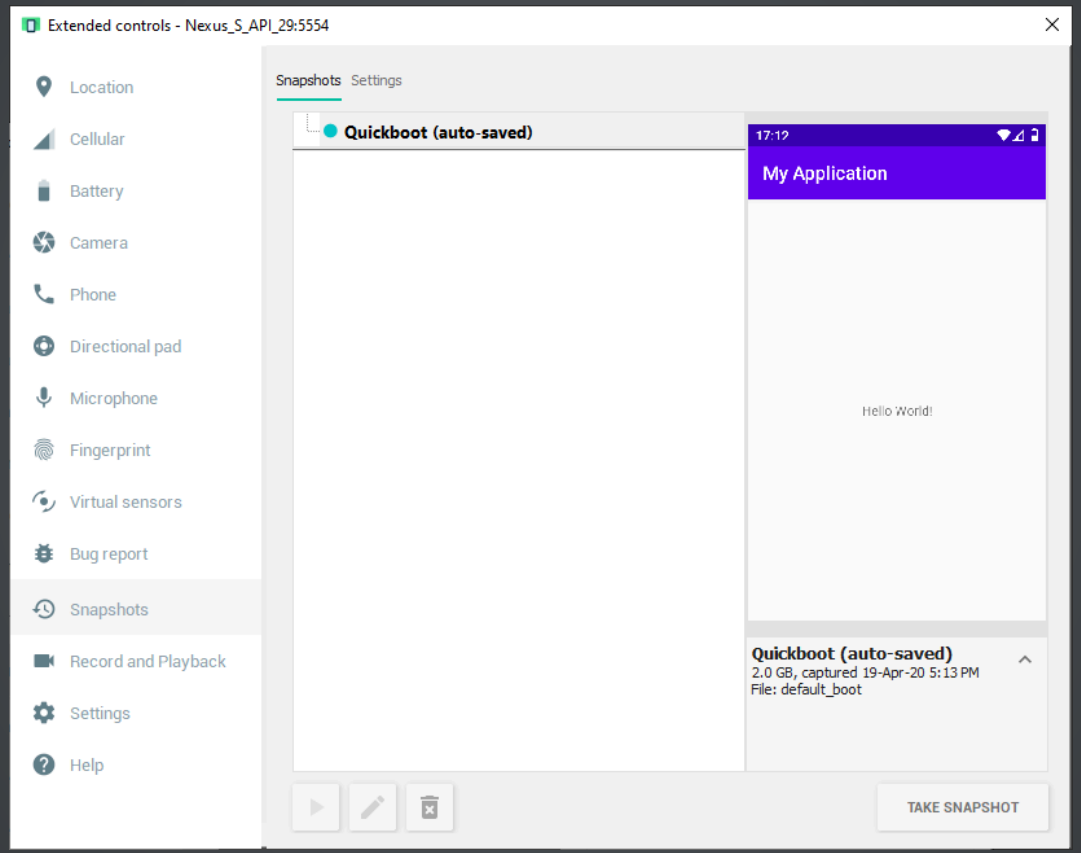
Émulateur

Envoyer un rapport de bug :



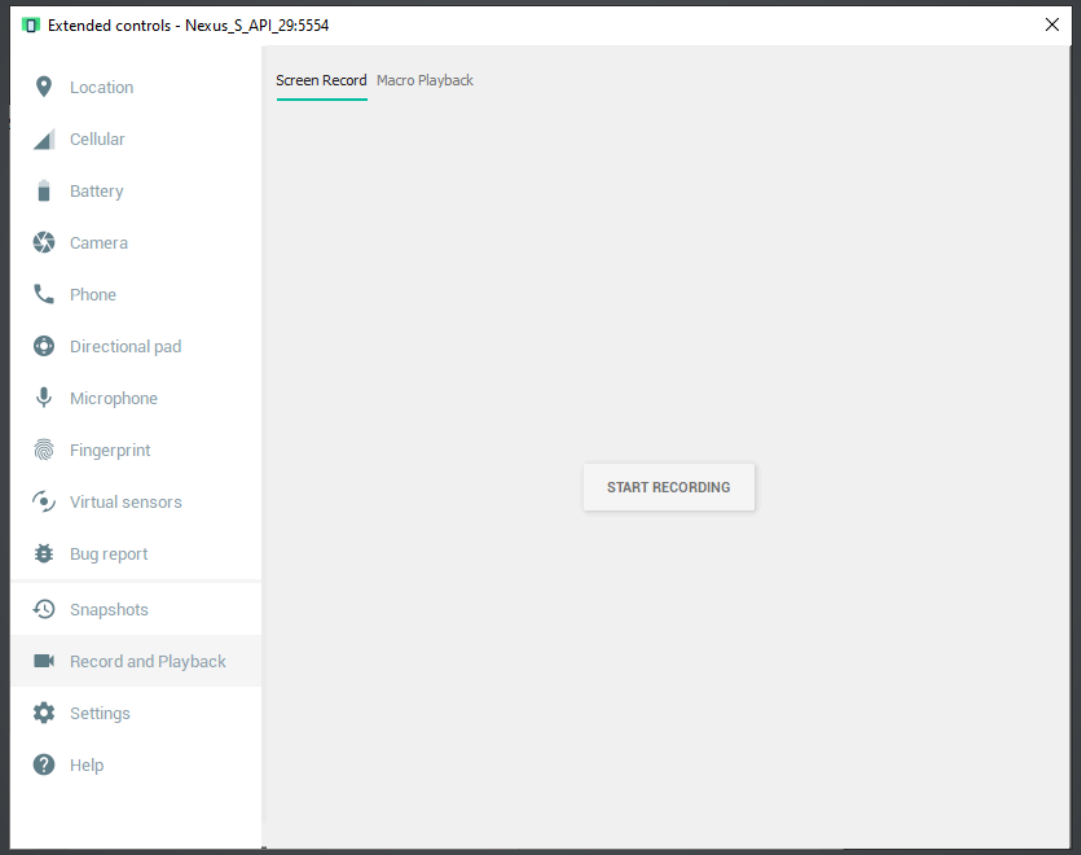
Émulateur

Définir des points de restauration :



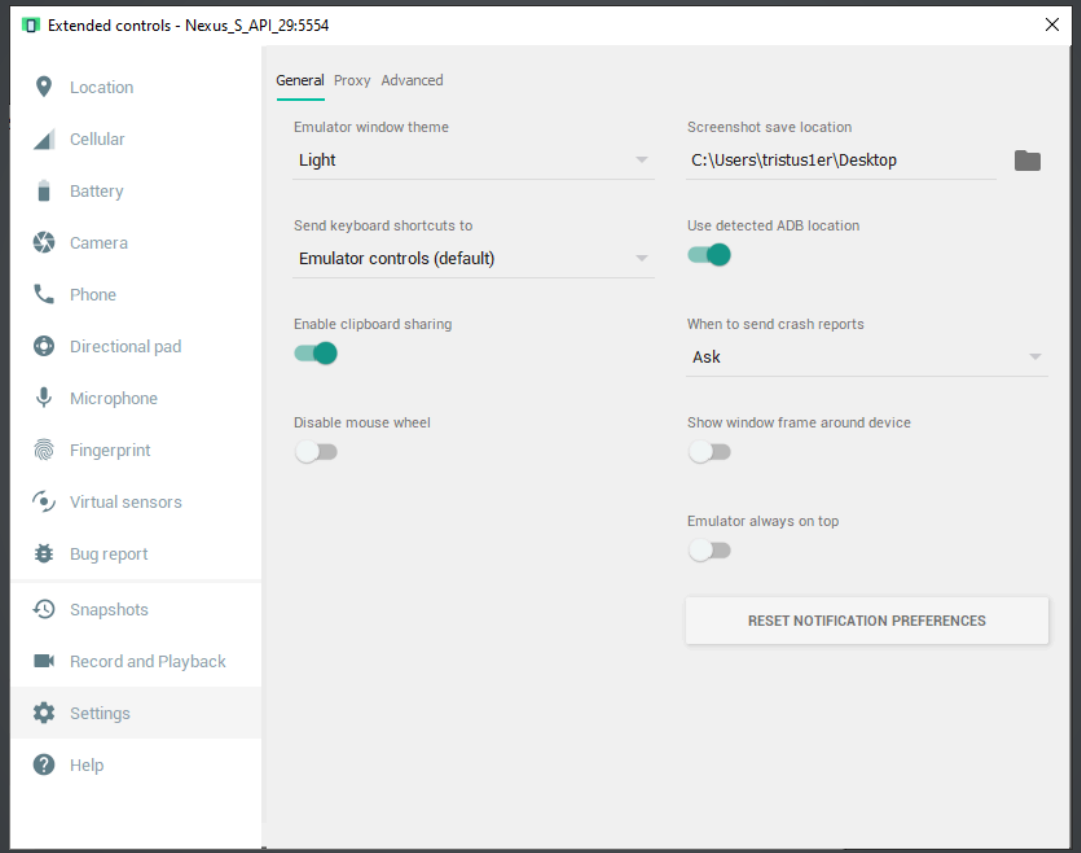
Émulateur

Enregistrer l'écran :



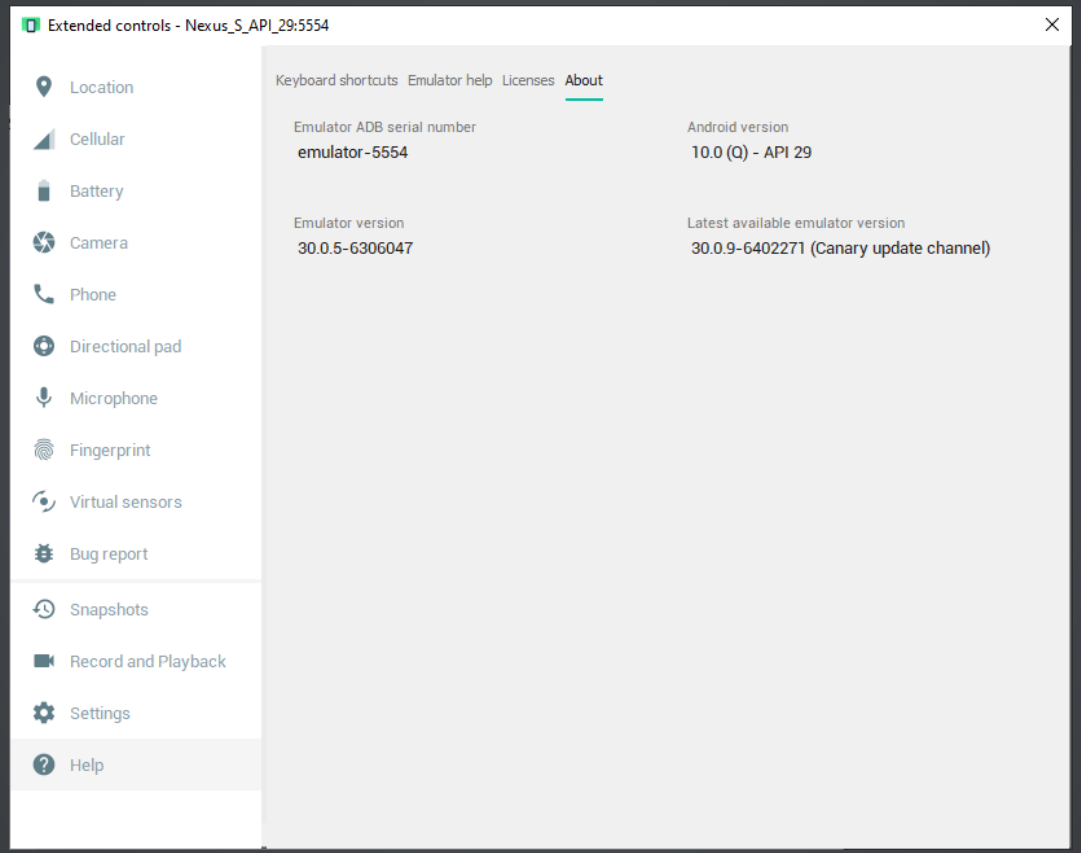
Émulateur

Définir certains paramètres :



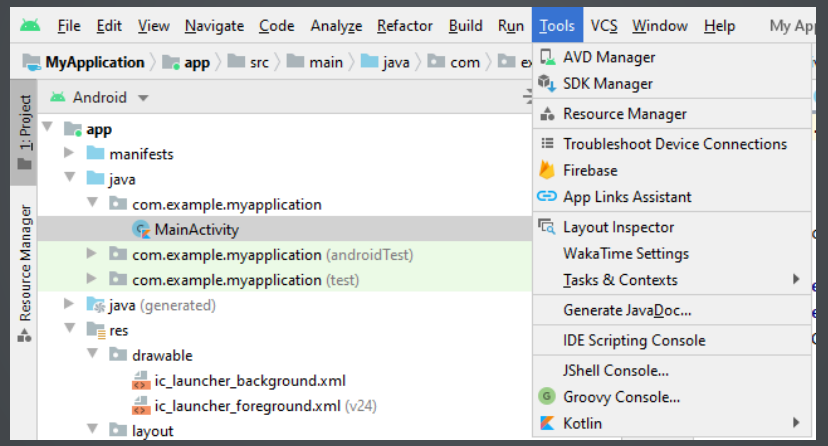
Émulateur

Obtenir des informations sur l'émulateur :



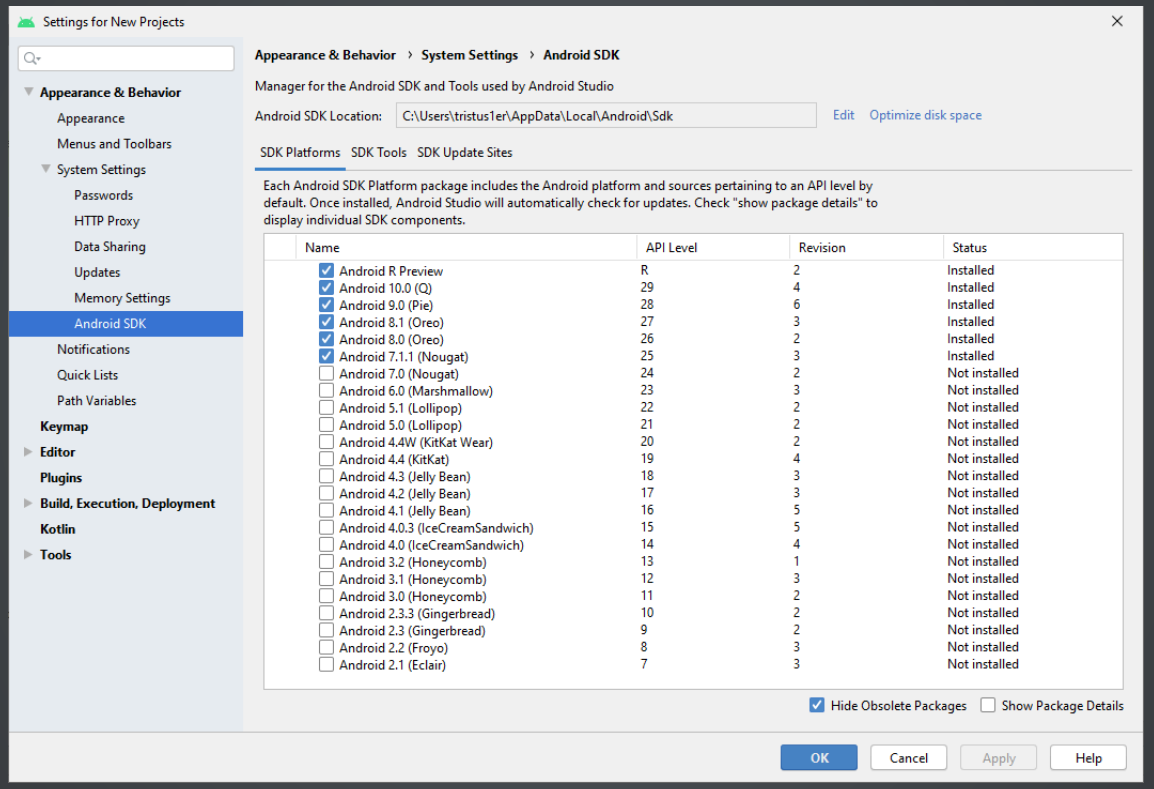
SDK

Lançons l'écran de gestion du SDK :
`Tools/SDK Manager`



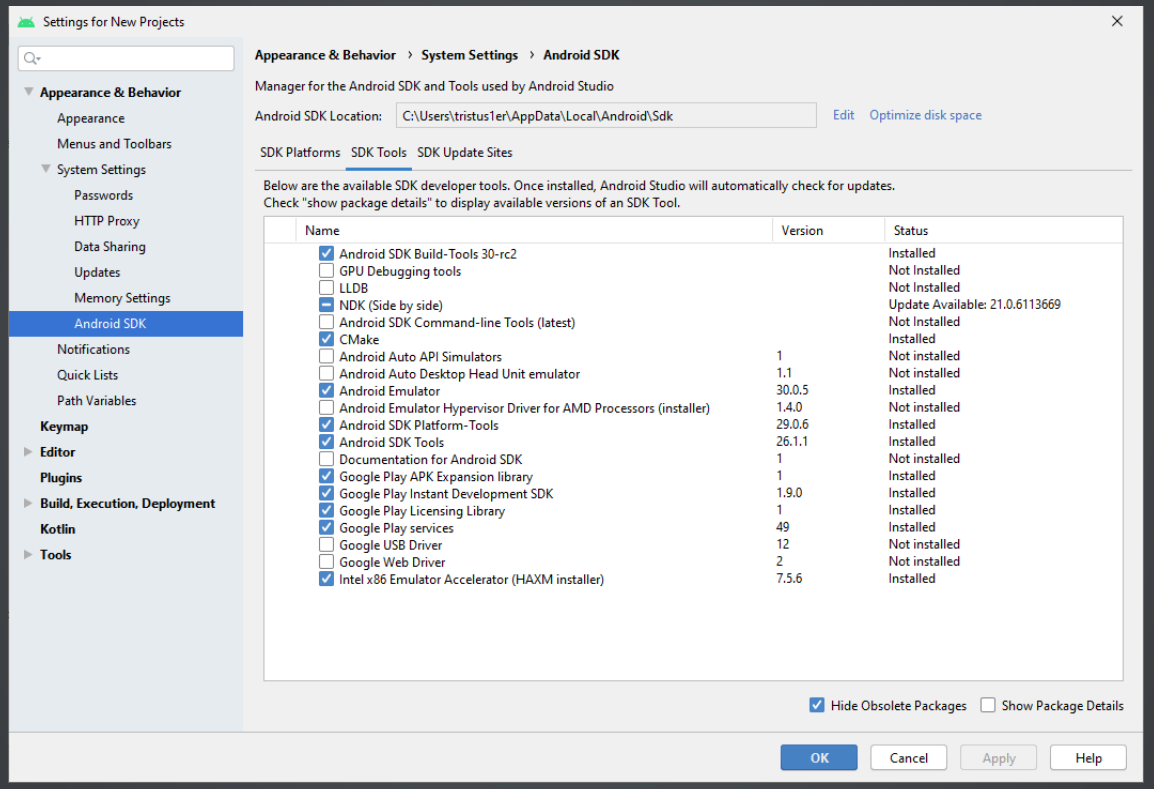
SDK

Nous pouvons choisir les versions de l'OS à télécharger :



SDK

Et les outils à installer :



Le Manifest

Le manifest est le fichier qui décrit l'application, pour les outils de compilation, pour le système d'exploitation et pour Google Play. On y retrouve principalement :

- Les interfaces avec l'extérieur : les activities, les services, ...
- Les permissions que nous allons utiliser.
- Le matériel nécessaire à notre application (NFC, Bluetooth, ...).

Tous les détails ici.

Le Manifest

Exemple

Copier

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.myapplication">
    <application
            android:allowBackup="true"
            android:icon="@mipmap/ic_launcher"
            android:label="@string/app_name"
            android:roundIcon="@mipmap/ic_launcher_round"
            android:supportsRtl="true"
            android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

La production de l'application, la publication

La production de l'application

Une fois développée, principalement via Android Studio (il existe d'autres moyens, par exemple via des outils de développement cross plate-forme), l'application est enregistrée dans un APK/App Bundle, qui contient :

- Le code de l'application.
- Les ressources.
- Les assets.
- Les certificats/signatures (pour l'App Bundle, la signature est effectuée par Google).
- Le Manifest.

Le fichier APK est un fichier ZIP qu'il est possible d'ouvrir avec n'importe quel utilitaire sachant traiter ce type de fichier (7Zip, Winzip, ...).

La production de l'application, la publication

La publication

Une fois packagée, l'application est le plus souvent publiée sur la boutique de Google : Google Play. Il existe d'autres boutiques, des stores alternatifs, tels que :

- F-Droid (applications open source)
- APK Miror
- App brain
- Amazon
- GETJAR
- Uptodown App Store
- Aptoide
- QooApp

La production de l'application, la publication

Il peut aussi s'agir de stores de constructeurs, tels que :

- Huawei AppGallery
- Samsung Galaxy App
- Boutique Mi

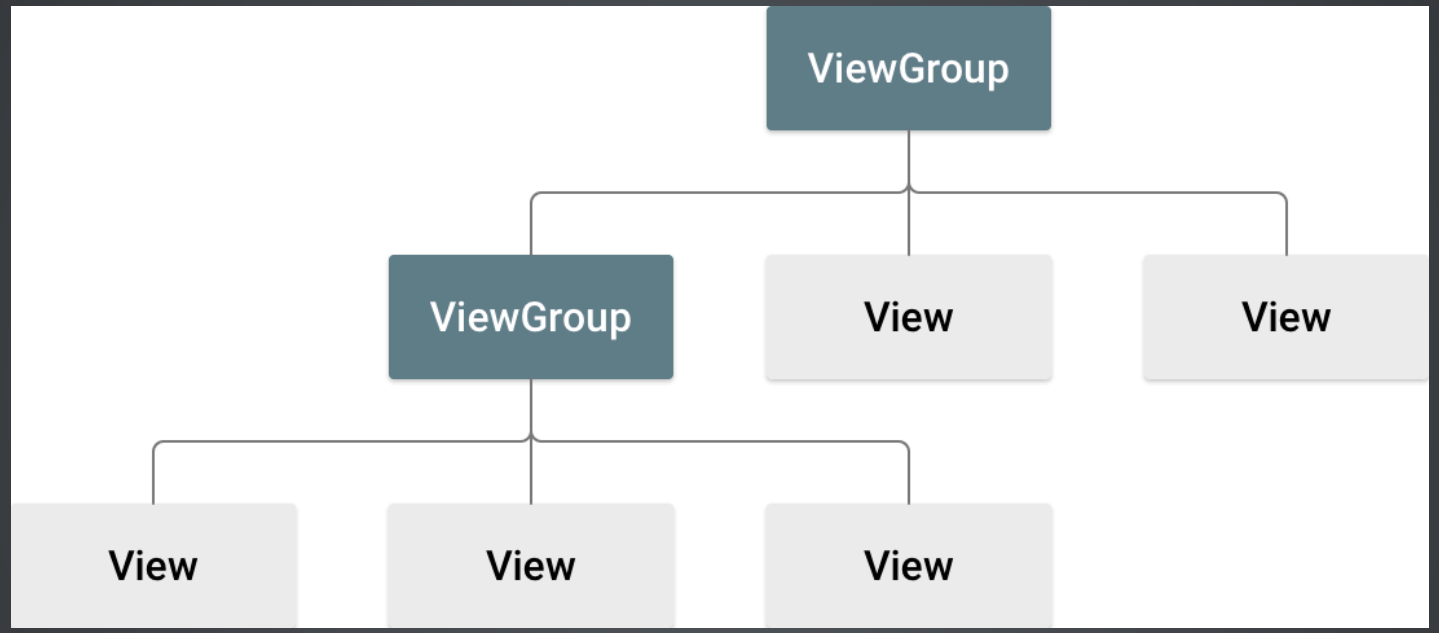
La production de l'application, la publication

La solution classique consiste quand même à publier sur Google play store, via la Google Play Console. Pour cela, il faudra un compte Google, et débourser 25€ pour avoir la possibilité de publier une application sur le store, à vie.

Les interfaces utilisateurs

- Organisation générale du layout.
- Exemple de layouts : LinearLayout, RelativeLayout, ConstraintLayout.
- Les ressources : drawables, string, styles, ...
- La gestion événementielle.

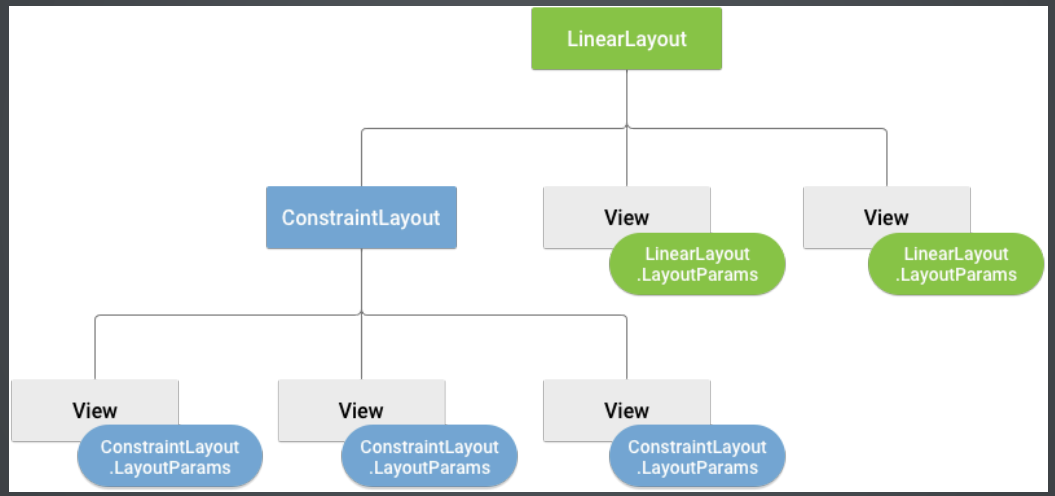
Organisation générale du layout



Source : https://developer.android.com/guide/topics/ui/declaring-layout

Organisation générale du layout

A chaque élément déclaré dans le layout, nous pouvons associer un id qui nous permettra de le retrouver/référencer dans le code (Java/Kotlin/XML).
Pour chaque layout, des contraintes particulières peuvent être appliquées (nous allons voir plusieurs layouts dans les slides suivantes).



Source : https://developer.android.com/guide/topics/ui/declaring-layout

Exemple de layouts

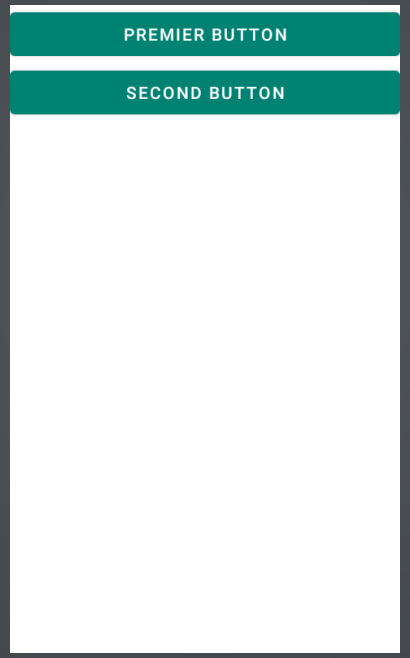
Exercices sur les Layouts

Dans tous les prochains exercices, vous n'utiliserez qu'un seul layout par écran.

Exemple de layouts

LinearLayout Exercice 1

Les 2 boutons sont positionnés l'un au-dessous de l'autre, et prennent toute la largeur :



Exemple de layouts

Solution

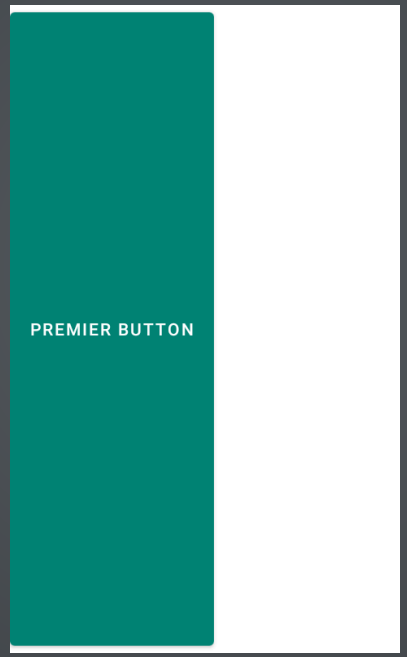
Copier

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical"
              tools:context=".MainActivity">

    <Button
            android:id="@+id/button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Premier button" />

    <Button
            android:id="@+id/button2"
            android:layout_width="match_parent"
```

Exemple de layouts

LinearLayout Exercice 2

Le bouton prend toute la hauteur de l'écran :



Exemple de layouts

Solution

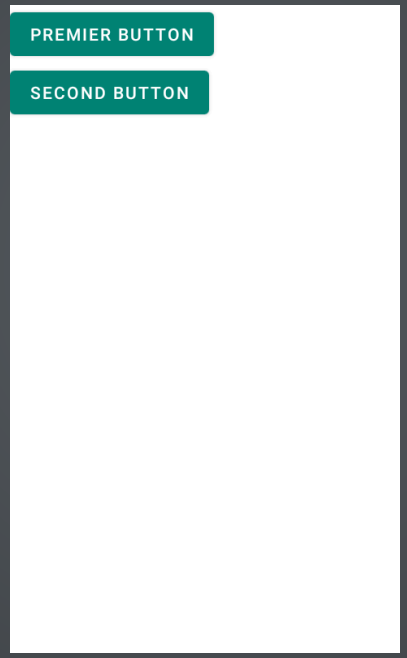
Copier

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical"
              tools:context=".MainActivity">

    <Button
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:text="Premier button" />

</LinearLayout>
```

Exemple de layouts

LinearLayout Exercice 3

Les 2 boutons sont positionnés l'un au-dessous de l'autre, et prennent la largeur nécessaire à leur affichage :



Exemple de layouts

Solution

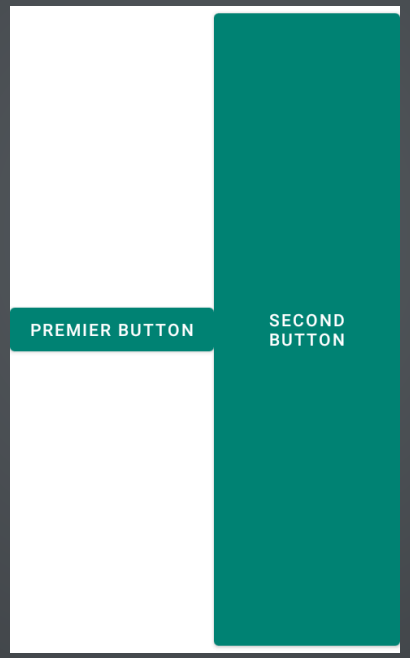
Copier

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical"
              tools:context=".MainActivity">

    <Button
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Premier button" />

    <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
```

Exemple de layouts

LinearLayout Exercice 4

Les 2 boutons sont positionnés l'un à côté de l'autre, centré verticalement, le premier prend la hauteur nécessaire à son affichage, et le second prend toute la hauteur :



Exemple de layouts

Solution

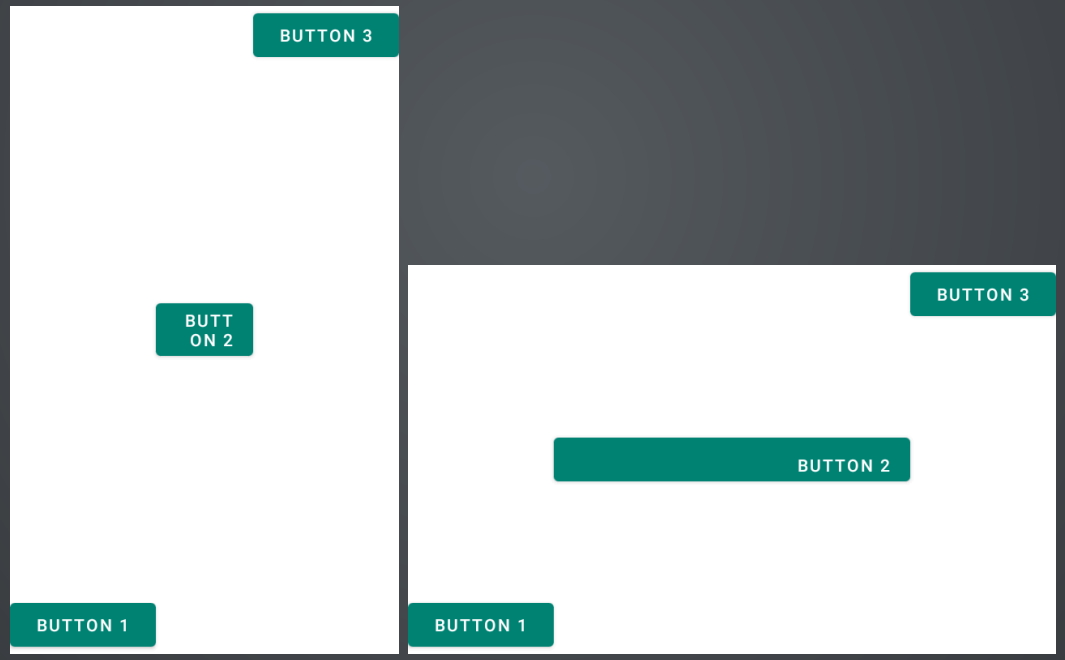
Copier

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="horizontal"
              tools:context=".MainActivity">

    <Button
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:text="Premier button" />

    <Button
            android:id="@+id/button2"
```

Exemple de layouts

LinearLayout Exercice 5

Les 3 boutons sont affichés les uns à côté des autres. Le bouton 1 est en bas, de largeur fixe : 120dp, celui du milieu prend la place disponible, et son texte est aligné en bas à droite. Le 3ième bouton est de largeur fixe : 120dp :



Exemple de layouts

Solution

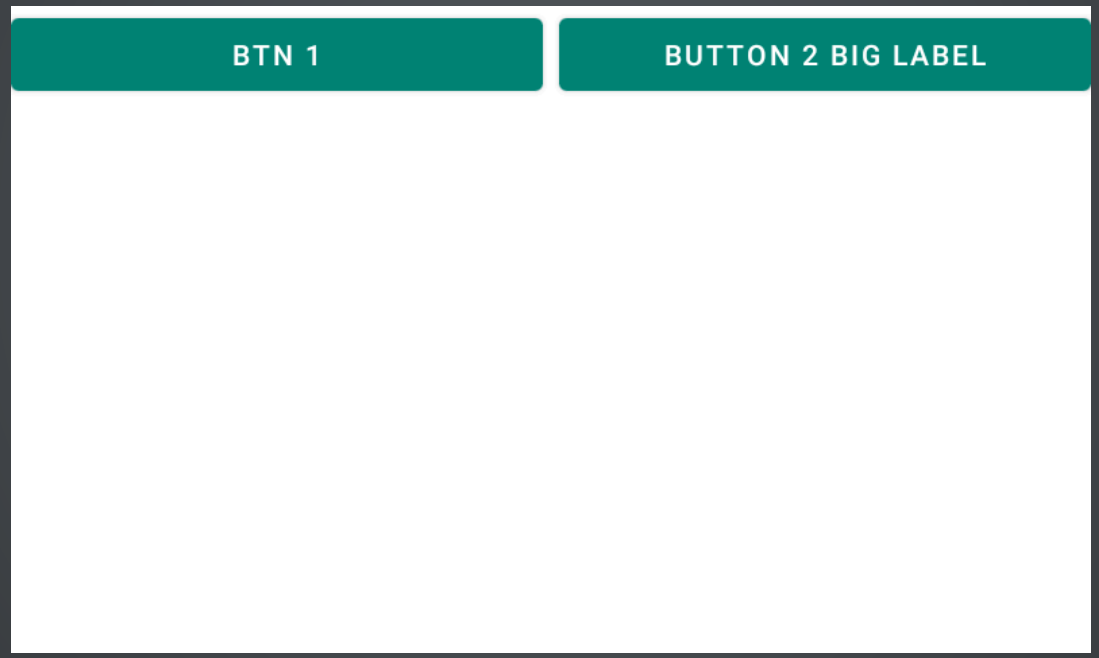
Copier

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="horizontal"
              tools:context=".MainActivity">

    <Button
            android:id="@+id/button"
            android:layout_width="120dp"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:text="button 1" />

    <Button
            android:id="@+id/button2"
```

Exemple de layouts

LinearLayout Exercice 6

Les 2 boutons sont affichés l'un à côté de l'autre, et prennent exactement la moitié de la largeur de l'écran, avec 2 labels de taille très différente :



Exemple de layouts

Solution

Copier

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="horizontal"
              tools:context=".MainActivity">

    <Button
            android:id="@+id/button"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginRight="4dp"
            android:layout_weight="1"
            android:text="btn 1" />

    <Button
```

Exemple de layouts

Afficher un contenu très grand

Dans le cas de l'utilisation d'une listview, verticale, permettant d'ajouter beaucoup d'éléments sur notre écran, il est préférable d'englober notre LinearLayout dans une ScrollView, par exemple :

Copier

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent">

    <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

        <Button
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="BTN" />

        <Button
                android:layout_width="match_parent"
```

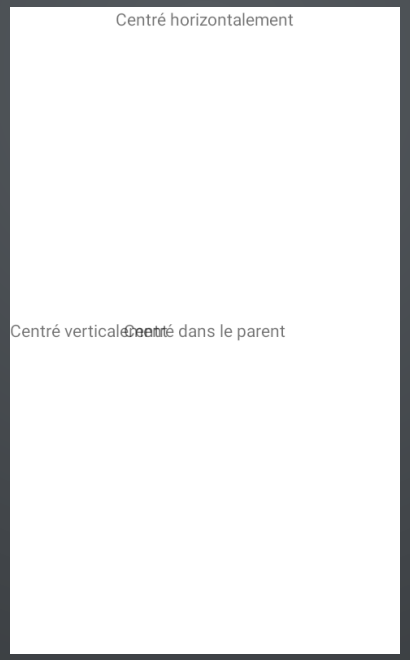
RelativeLayout Exercice 1

Exemple de layouts

Les 3 TextView sont alignés de la manière suivante :

- Centré horizontalement
- Centré verticalement
- Centré dans le parent

Que remarquez-vous d'inapproprié ?



Exemple de layouts

Solution

Copier

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:tools="http://schemas.android.com/tools"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                tools:context=".MainActivity">

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:text="Centré horizontalement" />

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
```

RelativeLayout Exercice 2

Exemple de layouts

Nous allons positionner les TextView suivant les paroles de la chanson (dans le même ordre dans le XML) :

- En haut
- En bas
- A gauche
- A droite
- Ces soirées-là ! (centrée dans le parent)

Exemple de layouts

Solution

Copier

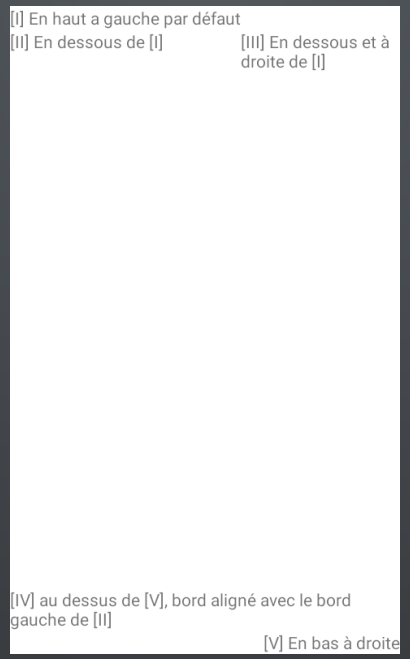
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:tools="http://schemas.android.com/tools"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                tools:context=".MainActivity">

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:text="En haut" />

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
```

RelativeLayout Exercice 3

Exemple de layouts

Nous allons maintenant positionner les TextView suivant les consignes suivantes :

- [I] En haut à gauche par défaut
- [II] En dessous de [I]
- [III] En dessous et à droite de [I]
- [IV] au-dessus de [V], bord aligné sur le bord gauche de [II]
- [V] En bas à droite



Exemple de layouts

Solution

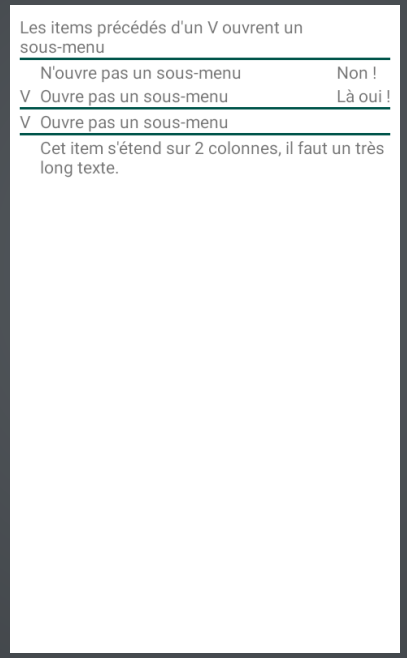
Copier

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:tools="http://schemas.android.com/tools"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                tools:context=".MainActivity">

    <TextView
            android:id="@+id/ex_rel_3_textView_1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:text="[I] En haut à gauche par défaut" />

    <TextView android:layout_marginLeft="30dp"
              android:id="@+id/ex_rel_3_textView_2"
              android:layout_width="wrap_content"
```

Exemple de layouts

TableLayout Exercice 1

Le table layout nous permet de faire des alignements de type tableaux :



Exemple de layouts

Solution

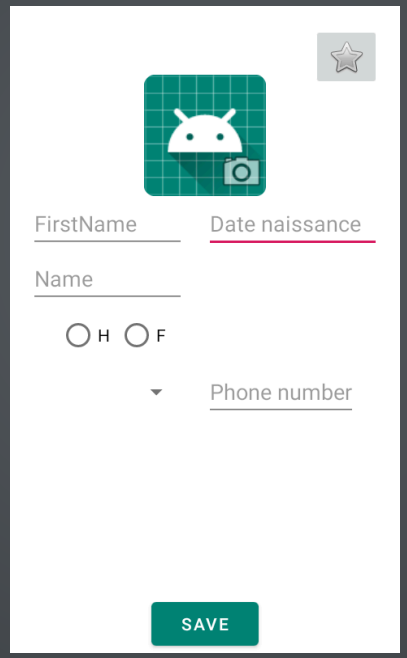
Copier

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
             xmlns:tools="http://schemas.android.com/tools"
             android:layout_width="match_parent"
             android:layout_height="match_parent"
             android:padding="8dp"
             android:stretchColumns="1"
             tools:context=".MainActivity">

    <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_span="3"
            android:text="Les items précédés d'un V ouvrent un sous-menu" />

    <TableRow>

```

Exemple de layouts

ConstraintLayout Exercice 1

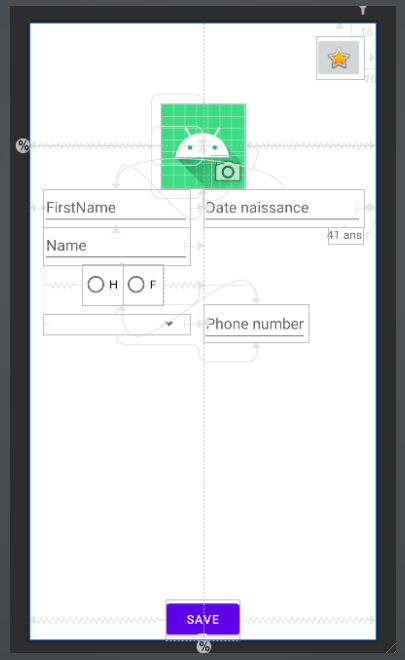
Le constraint layout est le dernier disponible. Il permet de réaliser des mises en forme complexes, tout en gardant une structure de layout plate (pas de layout imbriqué dans un autre layout) :



Exemple de layouts

ConstraintLayout Exercice 1 (détails)

Le détail des contraintes mises en places pour réaliser cette mise en forme de l'écran :



Exemple de layouts

Solution

Copier

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
                                                   xmlns:app="http://schemas.android.com/apk/res-auto"
                                                   xmlns:tools="http://schemas.android.com/tools"
                                                   android:layout_width="match_parent"
                                                   android:layout_height="match_parent"
                                                   tools:context=".MainActivity">

    <androidx.constraintlayout.widget.Guideline
            android:id="@+id/activity_edit_guideline_picture"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            app:layout_constraintGuide_percent="0.2" />

    <androidx.constraintlayout.widget.Guideline
            android:id="@+id/activity_edit_guideline_col1"
```

Les ressources

Les différentes ressources

Les ressources peuvent être de plusieurs types :

- anim : des animations que l'on peut utiliser pour animer des éléments de notre interface graphique.
- drawable : des images, bitmap ou vectorielles.
- layout : l'organisation de nos écrans.
- menu : la description des menus (des activités, des fragments, ...)
- mipmap : les icônes de notre application.
- raw : des données brutes (mp3 par exemple).
- xml : des données en XML.

Les ressources

Les différentes ressources

- values
  - arrays : des tableaux (de chaînes de caractères ou d'entiers)
  - colors : la définition de nos couleurs.
  - bools : des booléens.
  - dimens : des dimensions (tailles d'images, de texte, de marges, ...).
  - integers : des valeurs numériques.
  - plurials : des chaînes de caractères prenant en compte le pluriel.
  - strings : nos chaînes de caractères.
  - styles : pour afficher nos éléments.

Les ressources

Accéder aux valeurs en Java/Kotlin

Nous allons voir plus en détail l'utilisation d'une ressource. Dans un premier temps, pour les récupérer, en Java/Kotlin, nous utiliserons la syntaxe suivante :

```
[package.]R.type.nom

// Par exemple
R.string.app_name
```

Copier

Les ressources

Accéder aux valeurs en XML

En XML la syntaxe est similaire :

```
@[package:]type/nom

// Par exemple :
@string/app_name
```

Copier

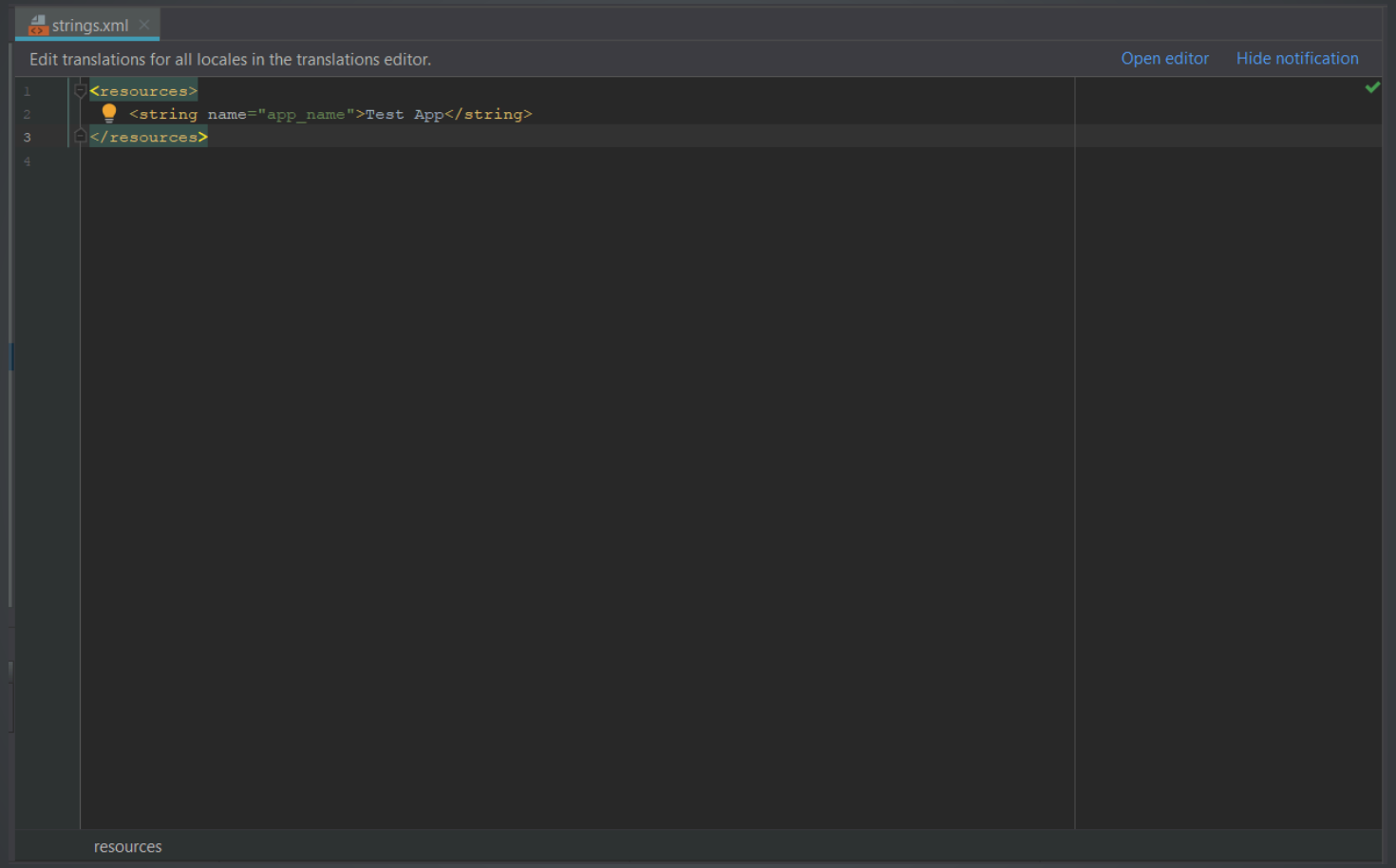
Les ressources

Internationalisation

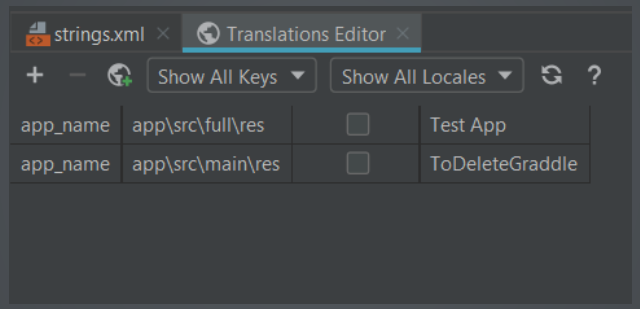
Voyons le fichier de ressource `strings.xml` et son utilisation pour internationaliser notre application.

- Ajouter une langue.
- Ajouter des clés.
- Tester dans l'éditeur.
- Tester sur l'émulateur.
- Tester sur le téléphone.

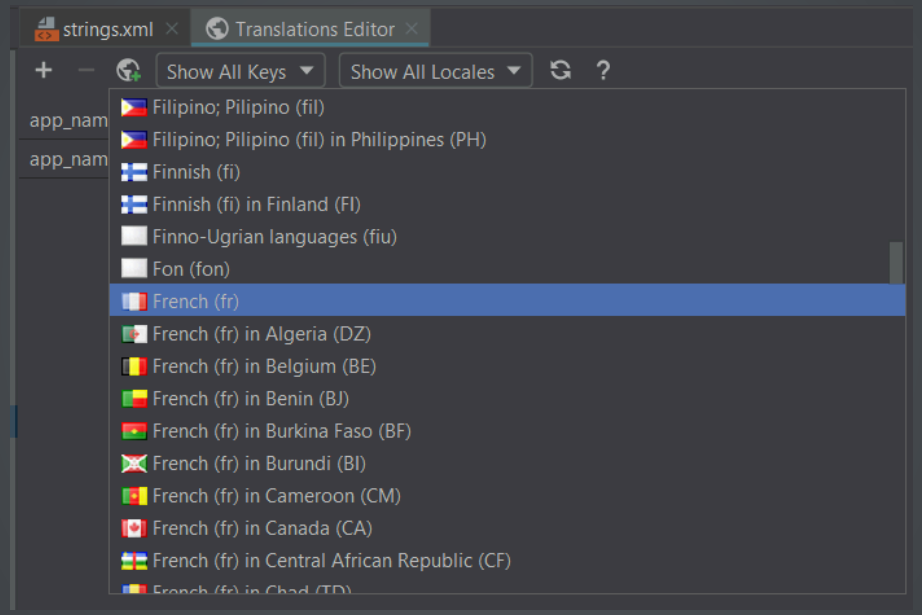
Les ressources



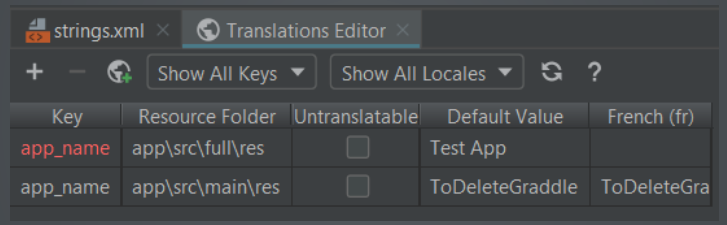
Les ressources



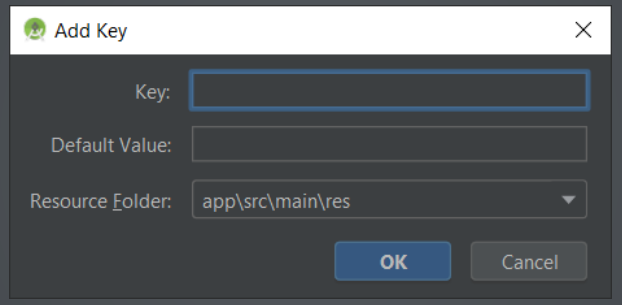
Les ressources



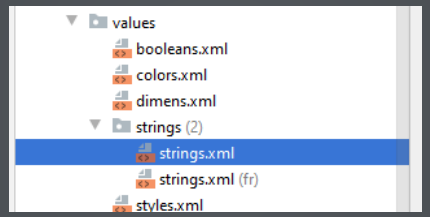
Les ressources



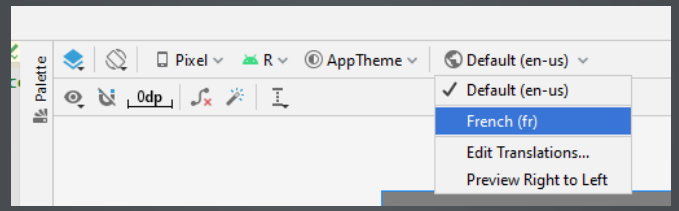
Les ressources



Les ressources



Les ressources



Les ressources

Apostrophes et guillemets

Le format XML nous impose des contraintes par rapport aux caractères ' et ", regardons comment nous pouvons gérer ces cas :

```
<string name="good_example">"This'll work"</string>
<string name="good_example_2">This\'ll also work</string>
<string name="bad_example">This doesn't work</string>
<string name="bad_example_2">XML encodings don't work</string>
```

Copier

Les ressources

Formatage valeurs

Nous pouvons ajouter des variables dans notre texte, qui seront remplacées par les valeurs passées en paramètre, par exemple :

```
1 <string name="welcome_messages">Hello, %1$s! You have %2$d new messages.</string>
```

Copier

```
1 Resources res = getResources();
2 String text = String.format(res.getString(R.string.welcome_messages), username, mailCount);
```

Copier

Les ressources

Le pluriel

Les ressources permettent aussi de gérer le pluriel :

Copier

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals name="numberOfSongsAvailable">
        <!- zero, one, two, few, many, other
        As a developer, you should always supply "one" and "other"
        strings. Your translators will know which strings are actually
        needed for their language. Always include %d in "one" because
        translators will need to use %d for languages where "one"
        doesn't mean 1 (as explained above).
        <item quantity="one">%d song found.</item>
        <item quantity="other">%d songs found.</item>
    </plurals>
</resources>
```

Les ressources

Exemple d'utilisation du pluriel

Il s'agit d'une valeur dynamique, nous utilisons donc du code pour définir la valeur de la chaîne de caractère, par exemple :

```
1 int count = getNumberOfsongsAvailable();
2 Resources res = getResources();
3 String songsFound = res.getQuantityString(R.plurals.numberOfSongsAvailable, count, count);
```

Copier

Les ressources

Mise en forme

Nous avons à disposition quelques outils de mise en forme de notre texte :

```
<b> for bold text.
<i> for italic text.
<u> for underline text.
```

Copier

Par exemple :

```
<string name="welcome">Welcome to <b>Android</b>!</string>
```

Copier

Les ressources

Tableaux

Nous pouvons aussi définir des tableaux, par exemple :

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
    </string-array>
</resources>
```

Copier

Que l'on utilisera :

```
val planets = resources.getStringArray(R.array.planets_array)
```

Copier

Les ressources

Dimens

Le fichier dimens.xml contiendra des définitions de taille, par exemple la taille d'une image, d'une marge, d'un texte, ...
Nous avons plusieurs types de dimensions disponibles :

- px : mesure en pixel, à bannir
- dp : mesure qui s'adapte à la densité de pixel de l'écran.
- sp : mesure utilisée pour les textes, car elle prend en compte le paramétrage de l'utilisateur relatif à la taille de la police de caractère.

Les ressources

Colors

Nous pouvons définir toutes nos couleurs dans ce fichier. Par exemple :

Copier

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>
</resources>
```

Les ressources

Styles

Les styles nous permettent de regrouper des caractéristiques d'affichage d'éléments (des titres, des entêtes, ...). Par exemple :

Copier

```
<style name="Counter" parent="AppTheme">
    <item name="android:textColor">@color/l4e_counter_text</item>
    <item name="android:background">@drawable/rounded_corner</item>
    <item name="android:padding">@dimen/l4e_counter_padding</item>
</style>

<style name="DialogTitle" parent="AppTheme">
    <item name="android:padding">@dimen/l4e_dialog_title_text_padding</item>
    <item name="android:textSize">@dimen/l4e_dialog_title_text_size</item>
    <item name="android:textStyle">bold</item>
</style>
```

La gestion événementielle

Ajout d'un bouton

Nous allons ajouter un bouton à notre interface, via l'éditeur graphique, ou via l'éditeur texte.

La gestion événementielle

Afin de réagir aux click de n'utilisateur, nous allons brancher un View.OnClickListener, ou un View.OnLongClickListener.
Pour cela, nous avons 4 options que nous allons détailler :

- Par héritage
- Classe anonyme
- Attribut
- XML

Voyons tout cela en détail.

La gestion événementielle

Par héritage

Suivons les étapes suivantes :

- Faisons étendre notre MainActivity de View.OnClickListener.
- Implémentons la méthode en utilisant l'ampoule rouge proposée par l'éditeur.
- Remplissons la méthode onClick générée.
- Associons notre listener à un bouton

Contenu de la méthode onClick :

```
when(v?.id){
    R.id.activity_main_button_test -> Log.d(TAG, "onCreate click")
}
```

Copier

Association du listener avec notre classe.

```
findViewById<Button>(R.id.activity_main_button_test).setOnClickListener(this@MainActivity)
```

Copier

La gestion événementielle

Par attribut

Déclarons un attribut de classe :

```
1 private val btnListener: View.OnClickListener = View.OnClickListener {
2     when (it?.id) {
3         R.id.activity_main_button_test -> Log.d(TAG, "onCreate click")
4     }
5 }
```

Copier

Attribuons notre listener à notre bouton :

```
1 findViewById<Button>(R.id.activity_main_button_test).setOnClickListener(btnListener)
```

Copier

La gestion événementielle

Par lambda

C'est la méthode la plus courante :

```
1 findViewById<Button>(R.id.activity_main_button_test).setOnClickListener(){ Log.d(TAG, "onCreate click") }
```

Copier

La gestion événementielle

Par XML

Dans le layout de notre activity, sur le bouton, ajoutons l'attribut onClick :

```
1 android:onClick="handleClick"
```

Copier

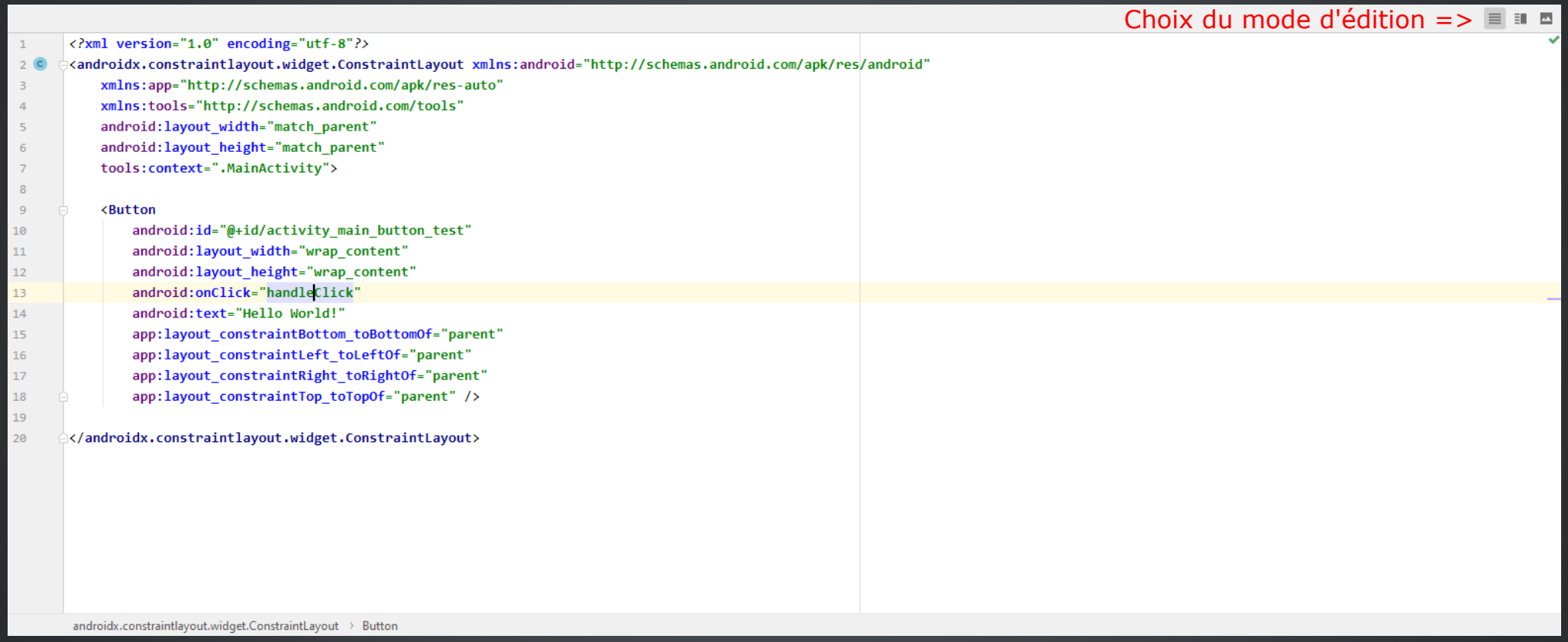
Créons la méthode en utilisant l'aide de l'éditeur, il nous reste plus qu'à implémenter la méthode.

Les éléments graphiques

Plusieurs éléments graphiques de base sont à notre disposition, nous allons en voir plusieurs, et la correspondance entre la partie graphique et la partie texte.

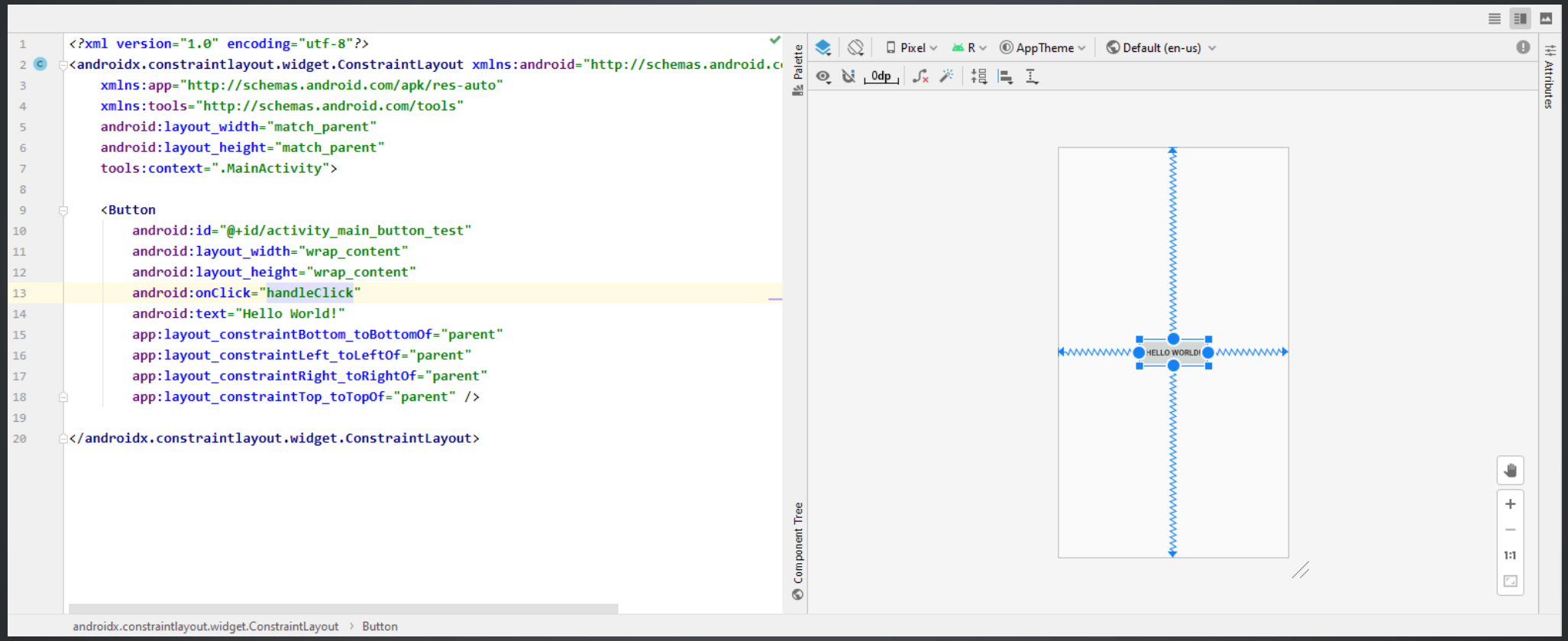
Les éléments graphiques

Éditeur : Code



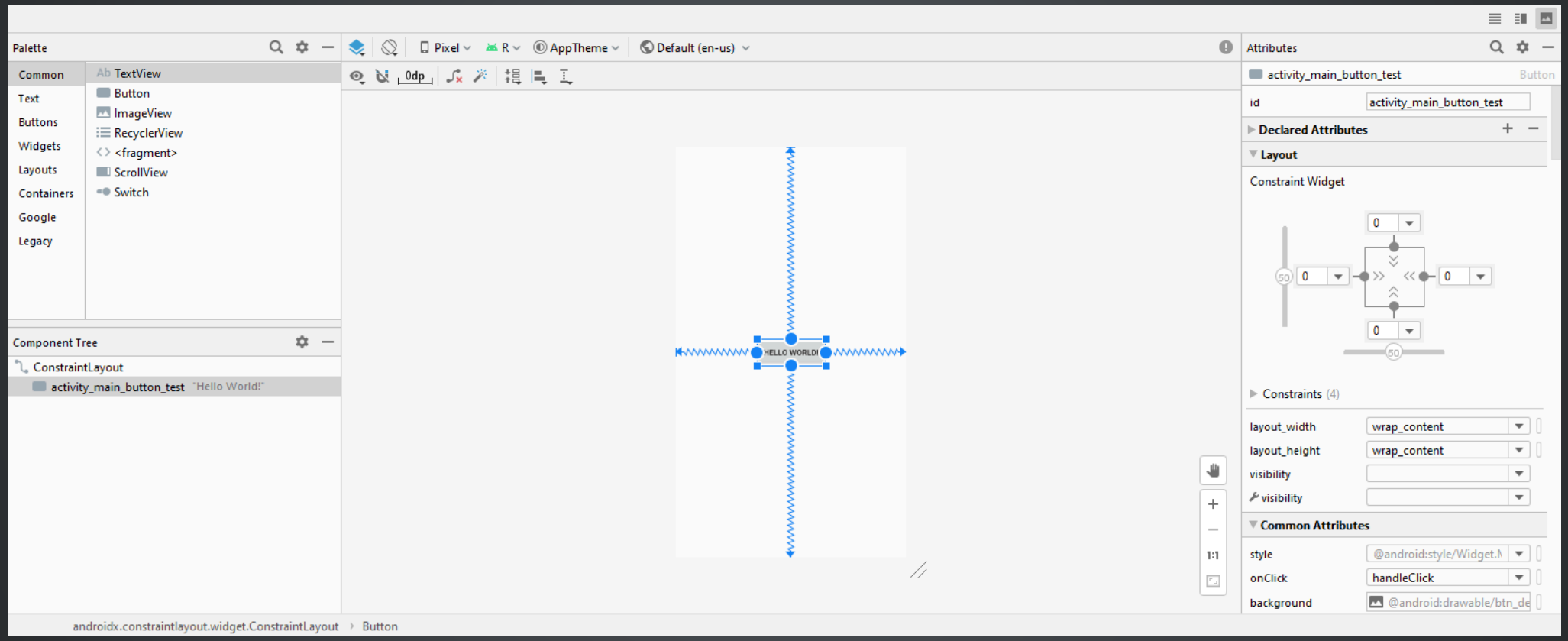
Les éléments graphiques

Éditeur : Split



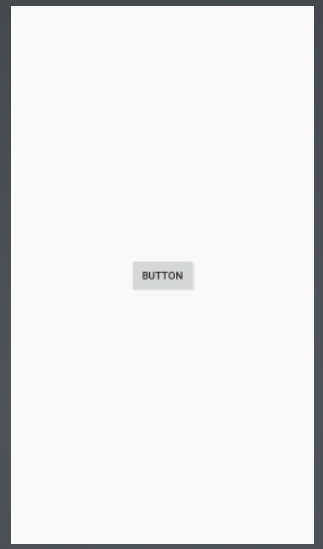
Les éléments graphiques

Éditeur : Design



Les éléments graphiques

Button

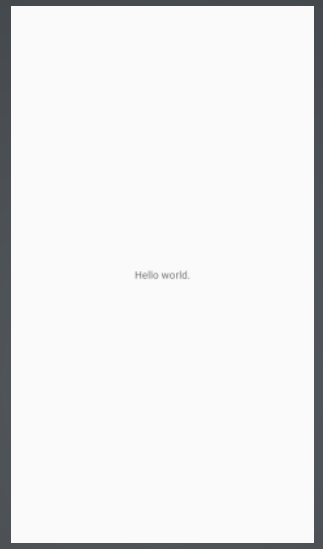


Copier

```
<Button
    android:id="@+id/activity_main_button_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="handleClick"
    android:text="Button"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Les éléments graphiques

Texte affiché

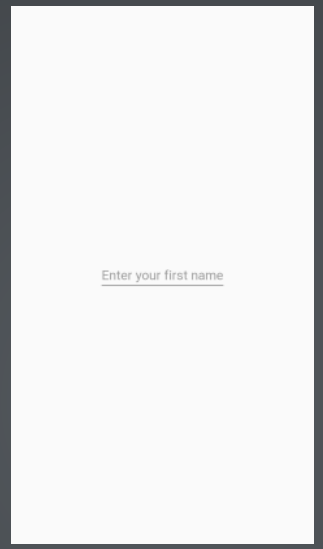


Copier

```
<TextView
    android:id="@+id/activity_main_textView_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello world."
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Les éléments graphiques

Champ texte



Copier

```
<EditText
    android:id="@+id/activity_main_editText_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="Enter your first name"
    android:inputType="textPersonName"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

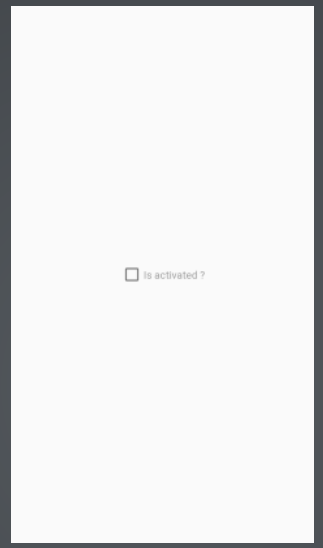
Les éléments graphiques

Champ texte (Material design)

Nous pouvons aussi utiliser la version moderne du champ texte, avec le LiveTemplate `and_material_design_edittext`.
Quelles différences notez vous ?

Les éléments graphiques

Case à cocher

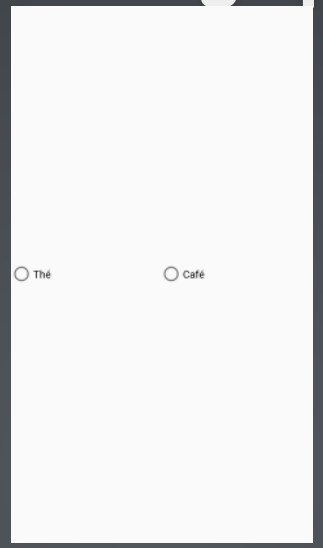


Copier

```
<CheckBox
    android:id="@+id/activity_main_checkBox_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="Is activated ?"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Les éléments graphiques

Boutons de choix




```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Thé" />

    <RadioButton
        android:layout_width="wrap_content"
```

Copier

Les éléments graphiques

Liste de choix

Dans la méthode onCreate de notre Activity utilisons le LiveTemplate : android_spinner_string_array.

Les éléments graphiques

Liste de choix encore plus rapide.

Gardons le `string-array` contenant la liste des planètes, retirons tout le code pour gérer le `Spinner` et ajoutons l'attribut `android:entries="@array/planets_array"`.

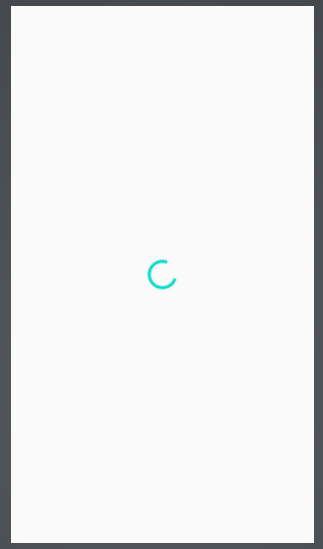
```
<Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:entries="@array/planets_array" />
```

Copier

Régulièrement des facilités de développement sont disponibles dans le SDK ou l'IDE.

Les éléments graphiques

Barre de progression

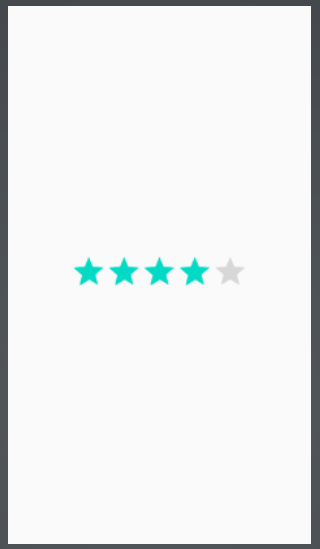


```
<ProgressBar
    android:id="@+id/activity_main_progressBar_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="true"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Copier

Les éléments graphiques

Notation (étoiles)

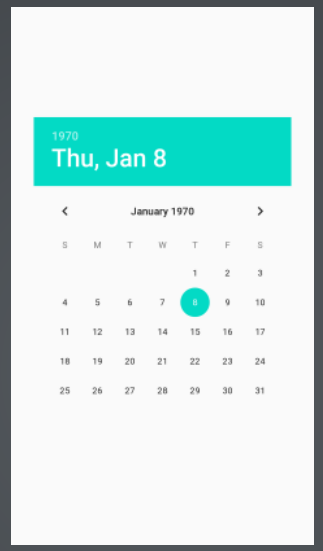


```
<RatingBar
    android:id="@+id/activity_main_ratingBar_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numStars="5"
    android:rating="4"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Copier

Les éléments graphiques

Date

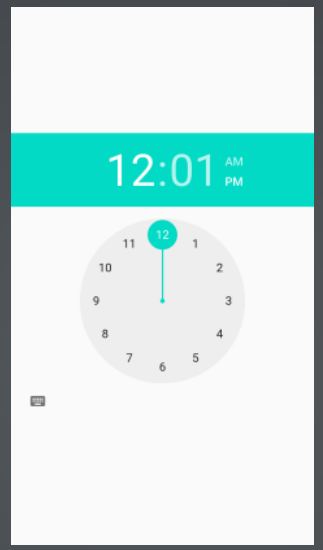



```
<DatePicker
    android:id="@+id/activity_main_datePicker_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Copier

Les éléments graphiques

Time

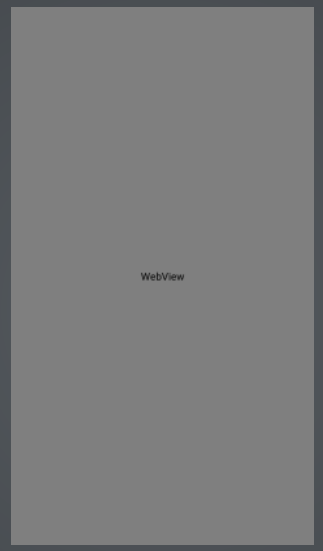



```
<TimePicker
    android:id="@+id/activity_main_timePicker_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```

Copier

Les éléments graphiques

Vue Web




```
<WebView
        android:id="@+id/activity_main_webView_test"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

Copier

Les éléments graphiques

Vue Web (suite)

Pour fonctionner, nous allons préciser l'URL que nous souhaitons afficher dans la méthode onCreate de notre Activity :

```
1 var webView = findViewById<WebView>(R.id.activity_main_webView_test)
2 webView.loadUrl("http://light4events.fr")
```

Copier

Quel est le problème qui empêche la page de s'afficher ? Comment résoudre le problème ?
Que remarque-t-on quand on clique sur un lien ? Comment résoudre le problème ?

Les éléments graphiques

Solution

Pour accéder à internet, nous devons demander la permission dans le manifest en ajoutant (en dessous de la balise manifest, mais avant la balise application) :

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

Copier

Pour empêcher l'ouverture d'un navigateur externe, nous allons ajouter la ligne suivante (avant de charger l'URL) :

```
1 webview.setWebViewClient(new WebViewClient());
```

Copier

Pour aller plus loin, nous pouvons utiliser le LiveTemplate : and_webview dans notre méthode onCreate.

Les éléments graphiques

Material design

Nous avons utilisé une forme de l'`EditText` en Material Design, nous avons d'autres LiveTemplate à disposition, en voici la liste exaustive :

- and_material_design_divider
- and_material_design_edittext
- and_material_design_progress
- and_material_design_slider
- and_material_design_switch

Le modèle de composants

- Les Intents et leur gestion par l'activité.
- La relation activité mère-fille.
- Les fragments, les services, les IntentServices.

Les Intents et leur gestion par l'activité

Les différents modules applicatifs ne sont pas directement instanciés par le développeur, un bus de messages permet au système de choisir le composant à monter en mémoire.
Les messages sont de type Intent.
Les valeurs envoyées dans l'Intent sont contenues dans un Bundle (optionnel).
Les intentions décrivent quelle est l'opération qui devra être effectuée.
Plusieurs composants applicatifs peuvent répondre à un même Intent, dans ce cas, l'utilisateur aura alors le choix.
Nous allons tester plusieurs Intent simple pour mettre en œuvre ces principes.

Les Intents et leur gestion par l'activité

LiveTemplates Intent

Nous avons à notre disposition plusieurs intents, entre autres :

- and_intent_mail : pour envoyer un email avec les champs préremplis.
- and_intent_map : pour ouvrir l'application Maps affichant une coordonnée.
- and_intent_phone_call : pour lancer le numéroteur téléphonique.
- and_intent_sms : pour envoyer un SMS (pour lancer l'application de SMS, prête à envoyer le SMS).
- and_intent_web : pour lancer un navigateur web pour afficher une URL.

Les Intents et leur gestion par l'activité

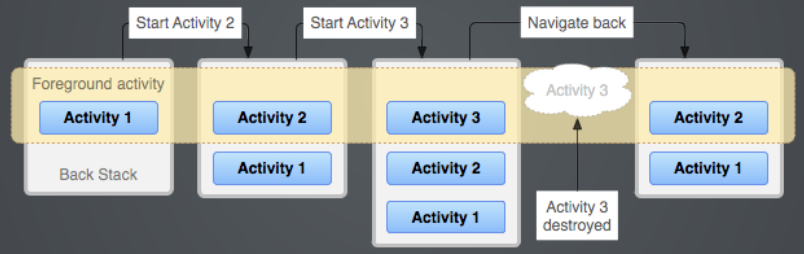
De la même manière, il est possible de déclarer dans le manifest des capacités de nos Activity à gérer des Intents. Par défaut l'Activity principale (MainActivity en général) gère l'appel par le Launcher. Dans le AndroidManifest.xml, nous voyons les lignes suivantes :

Copier

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

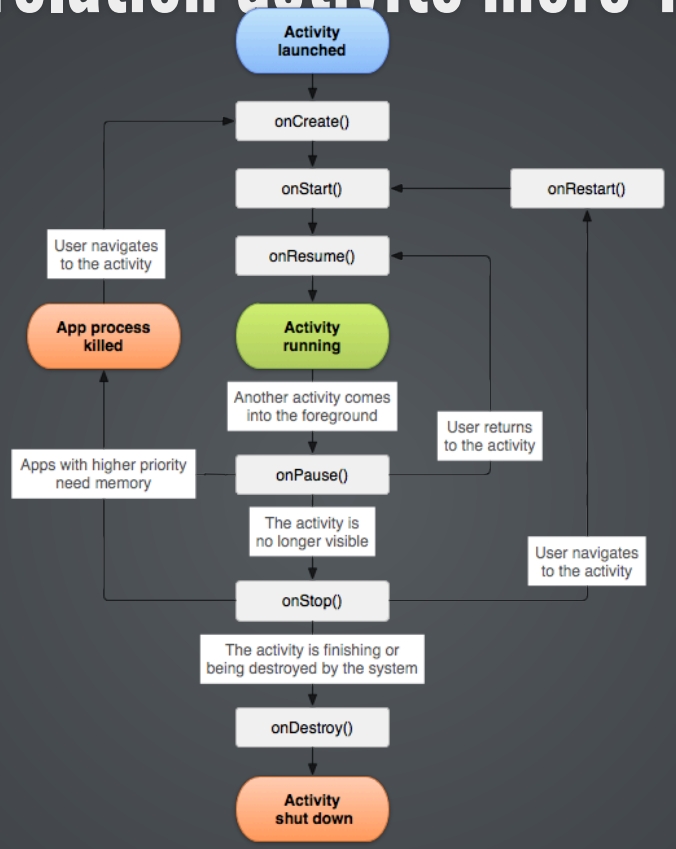
La relation activité mère-fille

Une application contient souvent de multiples activités.
Lorsqu'on en lance une, celle précédemment ouverte se stoppe (pause). La nouvelle activité se retrouve au-dessus de la pile de type « last in, first out » c'est-à-dire « dernier arrivé, premier sorti ».



Référence.

La relation activité mère-fille



Référence.

La relation activité mère-fille

Mise en oeuvre

Pour visualiser les principales étapes du cycle de vie de notre activity, utilisons le LiveTemplate : and_lifecycle_activity, dans la classe, mais en dehors d'une méthode.
Au besoin, nous utilisons le LiveTemplate : and_tag pour définir la constante TAG au début de notre classe.
Démarrons notre application, regardons les logs et tournons l'écran.
Que remarquons-nous ?

La relation activité mère-fille

Communication inter Activity

Nous allons maintenant voir comment lancer une nouvelle Activity, et lui envoyer des informations. Pour cela, nous allons passer par plusieurs étapes :

- Ajoutons un champ texte à notre interface.
- Récupérons la valeur du texte saisie.
- Envoyons-la à notre Activity2.
- Affichons la dans l'Activity2.

La relation activité mère-fille

Nous déclarons le champ texte et le bouton dans notre layout :

Copier

```
<EditText
        android:id="@+id/activity_main_editText_value"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

<Button
        android:id="@+id/activity_main_button_send"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Envoyer"
        app:layout_constraintTop_toBottomOf="@id/activity_main_editText_value" />
```

La relation activité mère-fille

Créons notre Activity2 via le click droit/New/Activity/Empty Activity.
Lançons cette activité lors du click bouton, avec en paramètre le texte de l'EditText : Nous déclarons une variable d'instance :

```
1 lateinit var etValue: EditText
```

Copier

Puis ajoutons le code suivant à notre méthode onCreate :

```
1 etValue = findViewById<EditText>(R.id.activity_main_editText_value)
2 findViewById<Button>(R.id.activity_main_button_send).setOnClickListener() {
3     var intent = Intent(this@MainActivity, Main2Activity::class.java)
4     intent.putExtra(INTENT_PARAMS_EXTRA_VALUE, etValue.text.toString())
5     startActivity(intent)
6 }
```

Copier

Je m'aide du LiveTemplate and_intent_activity pour écrire le code.

La relation activité mère-fille

Du côté de Main2Activity, je récupère la valeur envoyée, dans la méthode onCreate, avec le code :

```
1 var value = savedInstanceState?.getString(INTENT_PARAMS_EXTRA_VALUE) ?: intent.getStringExtra(INTENT_PARAMS_EXTRA_VALUE)
```

Copier

Je m'aide du LiveTemplate and_intent_onCreate pour écrire le code.
J'ajoute le même champ et bouton que la première activité, et j'affecte la valeur reçue à l'EditText :

```
1 findViewById<EditText>(R.id.activity_main2_editText_value).setText(value)
```

Copier

La relation activité mère-fille

Une activity peut aussi retourner des valeurs, par exemple demandons au système de sélectionner un contact de notre répertoire téléphonique.
Pour cela, sur un click de bouton, par exemple, je vais lancer le code généré par le LiveTemplate and_intent_select_contact et suivre les instructions.

La relation activité mère-fille

Retour de valeur

Voyons ensemble comment ce mécanisme fonctionne, je souhaite demander une information à une Activity, qui se lancera et me renverra une valeur (dans notre cas, une chaîne de caractère, mais cela peut être n'importe quel type d'information : un contact de notre répertoire, une image, ...).
Pour cela 3 étapes sont nécessaires :

- Activity1 : lancer l'Activity2 avec la méthode : startActivityForResult
- Activity2 : effectuer notre traitement (dans notre cas, laisser l'utilisateur saisir un texte et valider avec un bouton).
- Activity1 : récupérer la valeur envoyée par l'Activity2.

La relation activité mère-fille

Dans l'Activity, déclarons la constante :

```
1 const val REQUEST_CODE_GET_TEXT = 4567 // Arbitrary value
```

Copier

Dans l'Activity, toujours, lançons l'appel à l'Activity2, en attendant une réponse :

```
1 var intent = Intent(this@MainActivity, Main2Activity::class.java)
2 startActivityForResult(intent, REQUEST_CODE_GET_TEXT)
```

Copier

La relation activité mère-fille

Dans l'Activity2, je récupère la valeur saisie par l'utilisateur et je la renvoie. Pour cela, je m'aide du LiveTemplate : and_intent_returnValue. Nous pouvons maintenant récupérer la valeur renvoyée par l'Activity2, dans l'Activity1.
Pour cela, nous utilisons le LiveTemplate and_onActivityResult en dehors d'une méthode dans la classe MainActivity.

La relation activité mère-fille

Pour améliorer l'expérience utilisateur, nous pouvons préciser quel type d'informations l'utilisateur doit saisir afin de présenter un clavier adapté.
Les détails : Specify the input method type.
Par exemple pour un numéro de téléphone :

Copier

```
<EditText
    android:id="@+id/phone"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/phone_hint"
    android:inputType="phone" />
```

La relation activité mère-fille

Gestion du bouton OK

Nous pouvons aussi modifier le bouton OK/Valider du clavier pour refléter l'action à réaliser et déclencher le code directement depuis l'action sur le clavier, au lieu de passer par un bouton. Par exemple pour notre champ téléphone :

```
android:hint="@string/phone_hint"
android:inputType="phone"
android:imeOptions="actionSend"
```

Copier

Dans notre Activity, nous gérons le click sur le bouton "Envoyer" du clavier avec :

```
etValue.setOnEditorActionListener(){v, actionId, event ->
    if(actionId == EditorInfo.IME_ACTION_SEND){
            Log.d(TAG, "Action send to be handled.")
        true
    } else {
        false
    }
}
```

Copier

Activités aller plus loin

L'activité peut avoir des spécificités :

- Être appelée depuis une autre application.
- Pourquoi pas en plein écran (live template : `and_fullscreen_immersive`, à l'extérieur d'une méthode).
- Ou encore en mote PIP (Picture In Picture) (live template : `and_pip`, à l'extérieur d'une méthode).

Activités aller plus loin

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 aller plus loin

Création du bouton

- On déclare le bouton dans le `Layout`.
- On ajoute un `OnClickListener`.
- On implémente le partage du texte.

Activités aller plus loin

Pour partager le texte, on va utiliser (Java) :

Copier

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

Activités aller plus loin

Pour partager le texte, on va utiliser (Kotlin) :

Copier

```kotlin
// Create the text message with a string
val sendIntent = Intent()
with(sendIntent){
    action = Intent.ACTION_SEND
    type = "text/plain"
    putExtra(Intent.EXTRA_TEXT, "Un texte à partager")
}
// Start the activity
startActivity(sendIntent)
```

Activités aller plus loin

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>
```

Copier

Activités aller plus loin

Récupération du texte partagé (Java) :

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

Copier

Activités aller plus loin

Récupération du texte partagé (Kotlin) :

```kotlin
val receivedText = intent.extras?.getString(Intent.EXTRA_TEXT) ?: ""
Log.d(TAG, "onCreate received text = $receivedText")
```

Copier

Activités aller plus loin

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");
```

Copier

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 }
```

Copier

Activités aller plus loin

Chooser (Java).

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 :

Copier

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

Activités aller plus loin

Chooser (Kotlin).

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 :

Copier

```kotlin
// Always use string resources for UI text.
// This says something like "Share this photo with"
val title = resources.getString(R.string.chooser_title)
// Create intent to show the chooser dialog
val chooser = Intent.createChooser(sendIntent, title)
// Verify the original intent will resolve to at least one activity
if (sendIntent.resolveActivity(packageManager) != null) {
    startActivity(chooser)
}
```

Activités aller plus loin

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>
```

Copier

Essayons de cliquer sur le lien suivant : Ouvrir application avec "http://test.link.com".

Les fragments, les services, les IntentServices

Fragments

Étudions les templates :

- click droit / New / Activity / Primary/Detail Flow
- click droit / New / Activity / Tabbed Activity

Les fragments, les services, les IntentServices

Intent service

Créons un IntentService en utilisant le template :
File / New / Service / Service (IntentService)

Les fragments, les services, les IntentServices

Service

Créons un Service en utilisant le template :
File / New / Service / Service

Les fragments, les services, les IntentServices

Création d'une implémentation d'exemple (en Kotlin) : un service de génération de random.
Pour cela dans notre service, utilisons dans la classe de service un LiveTemplate pour écrire l'implémentation du service.

```
1 and_service_implementation
```

Copier

Pour appeler notre service (en Kotlin), nous utiliserons dans notre activity, un LiveTemplate, et nous suivrons à chaque fois les indications en TODO.

```
1 and_service_calling
```

Copier

NDK

Présentation du développement natif avec NDK. JNI.

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

NDK

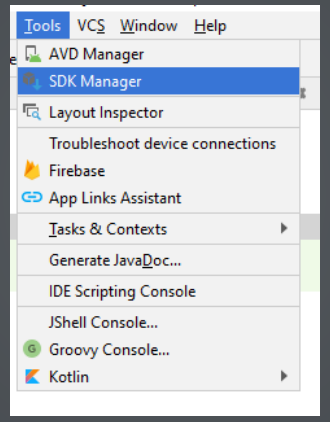
Création d'un projet.

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.

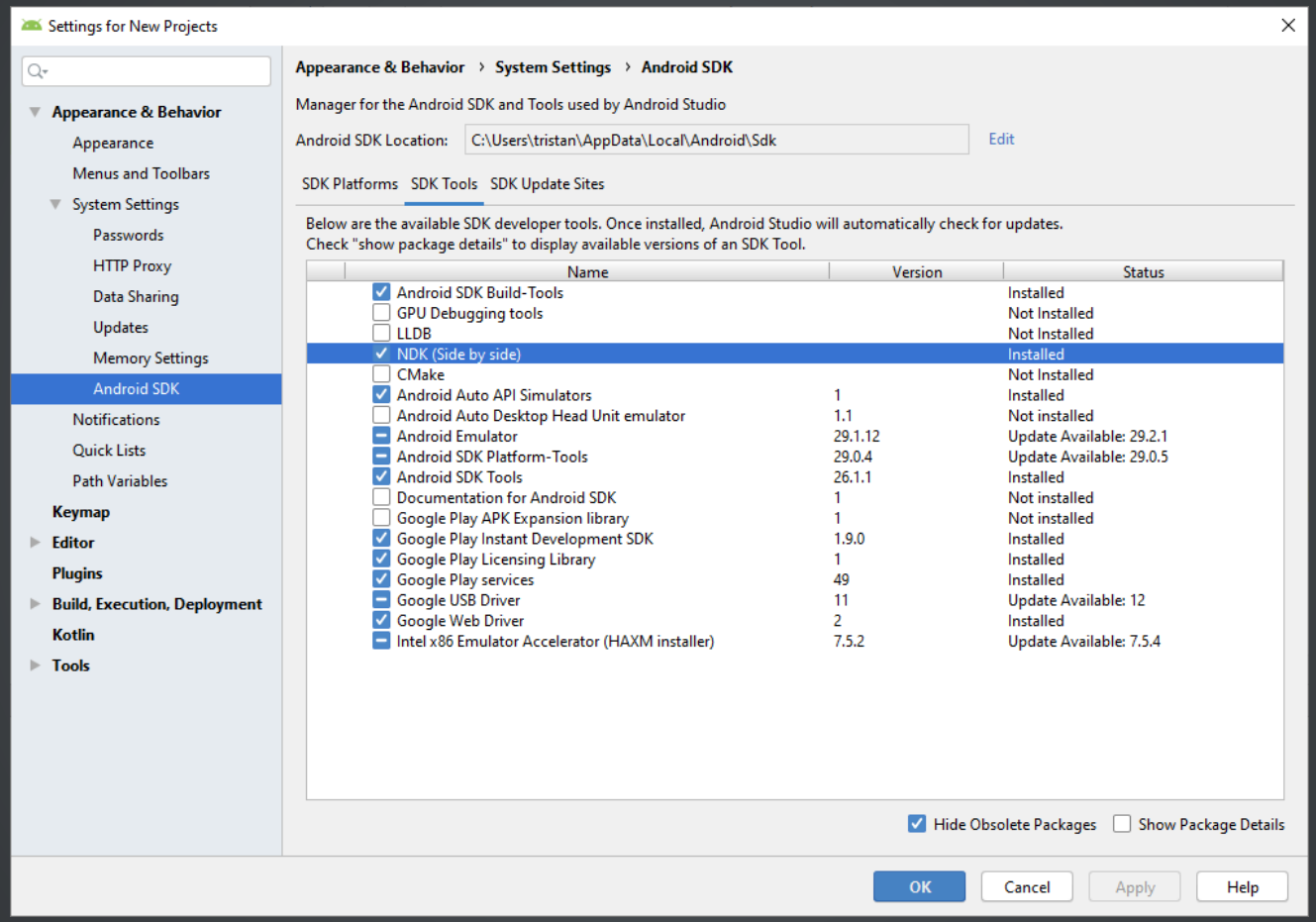
NDK

Installation du NDK.



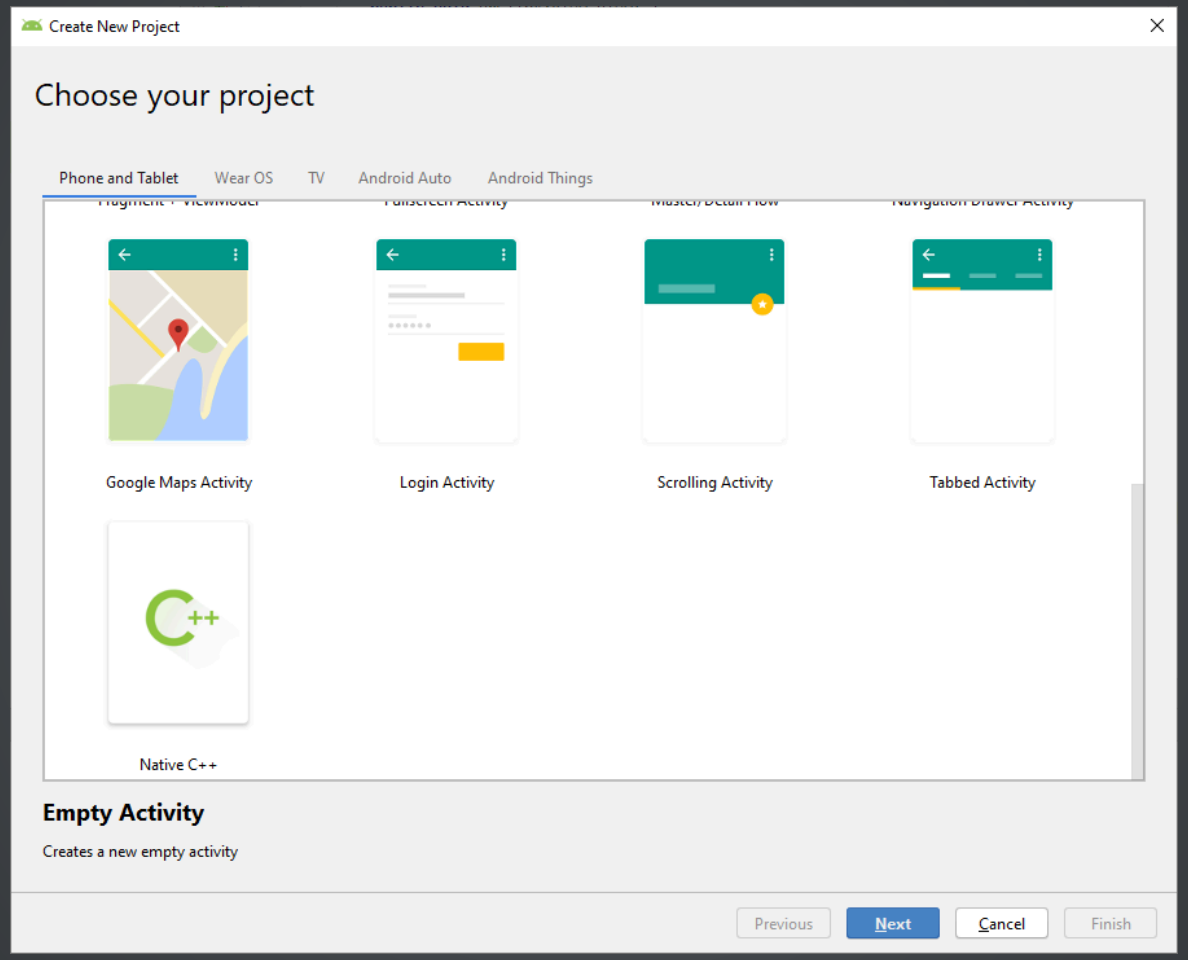
Installation du NDK.

NDK



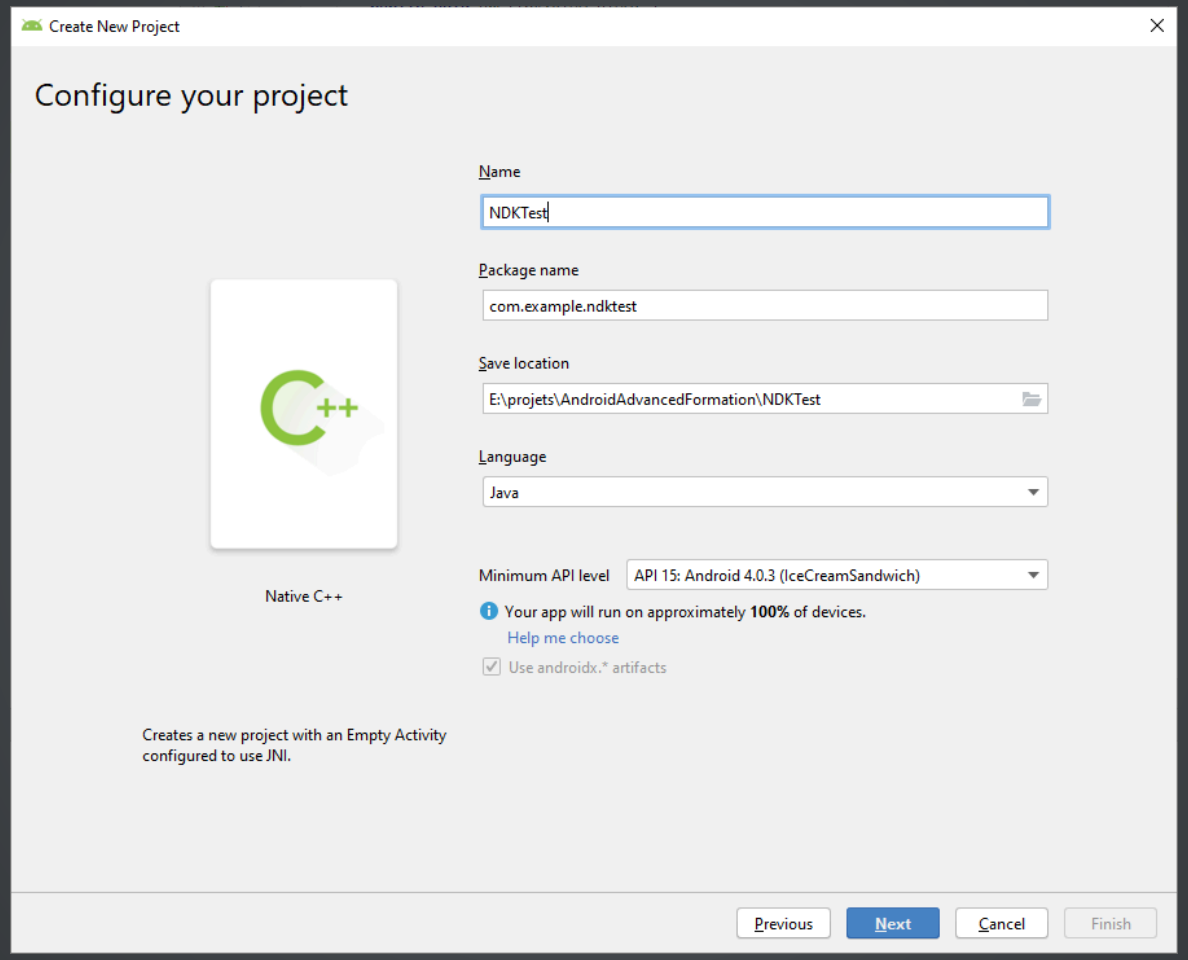
Création d'un projet.

NDK



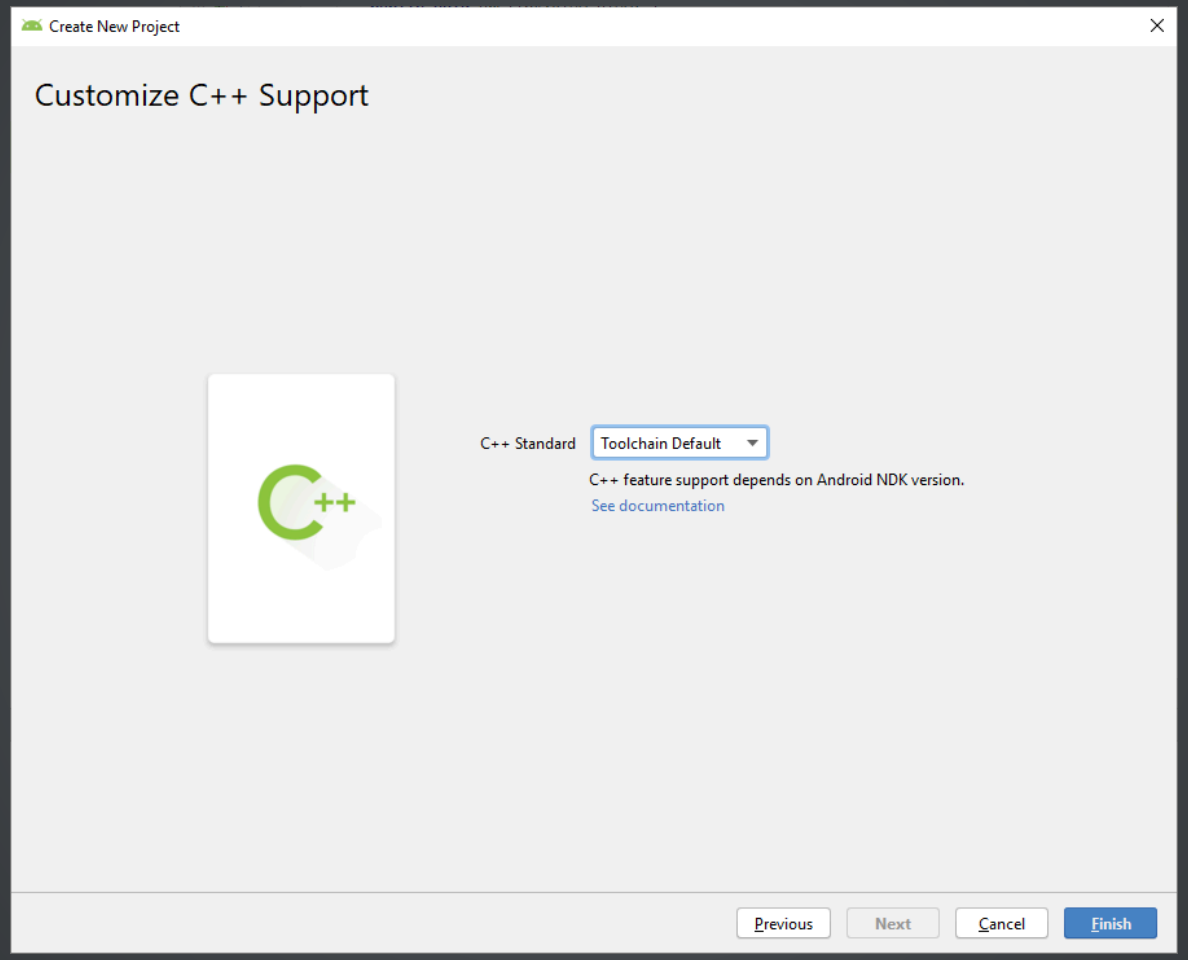
Nommage du projet.

NDK



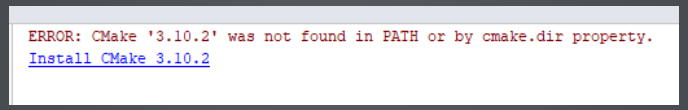
Sélection de la toolchain.

NDK



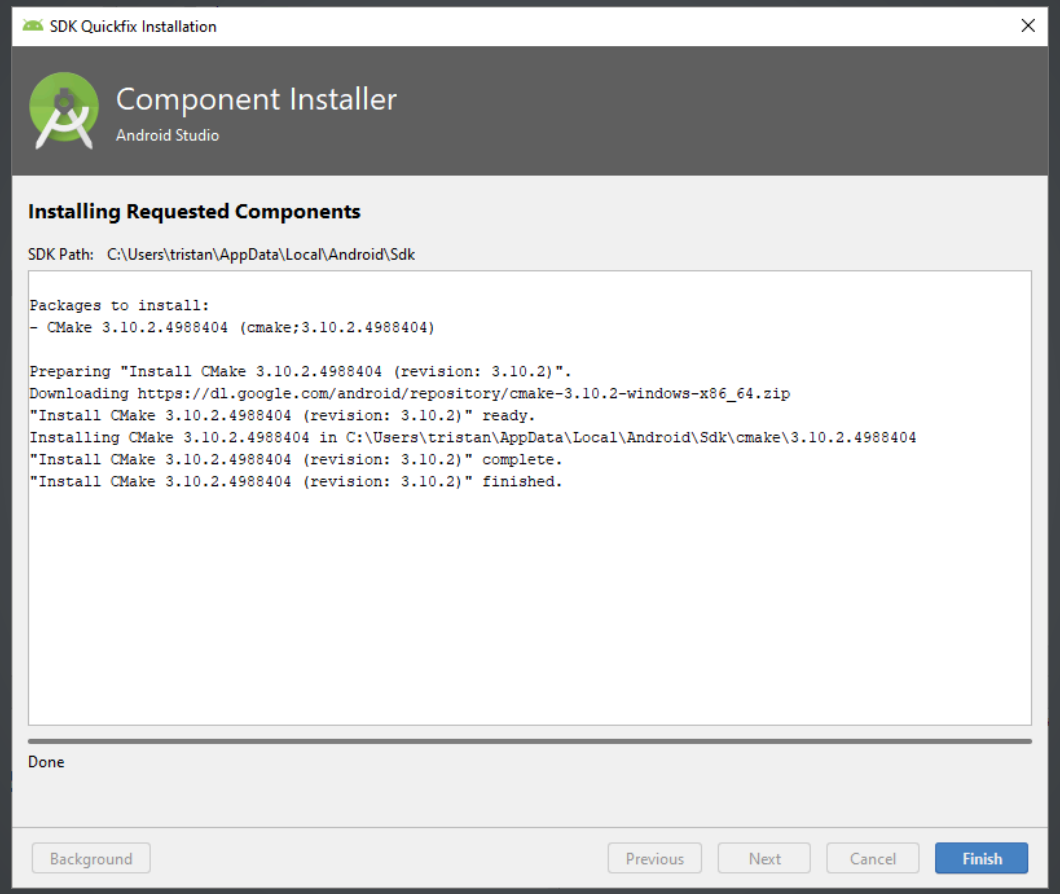
NDK

Gestion erreur CMAKE.



CMAKE Installé.

NDK



NDK

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
```

NDK

Test.

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

Notifications

Les Toasts

Avons-nous besoin de revoir les Toasts ?
Note, en Android 13+ il est nécessaire de demander une permission d'affichage des notifications (avec un channel vide par exemple) pour afficher des Toasts en arrière-plan.

Notifications

Notification

Les notifications, sont bien plus puissantes qu'il n'y parait, voyons un peu plus en détail leur fonctionnement.
Le template n'étant plus disponible, nous allons utiliser directement la documentation officielle.
Create a Notification

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.

Copier

```
<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 }
```

Copier

Permissions 6.0

Demande de la permission.

Copier

```
// 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
        ActivityCompat.requestPermissions(thisActivity,
                new String[]{Manifest.permission.CALL_PHONE},
                MY_PERMISSIONS_REQUEST_CALL_PHONE);
```

Permissions 6.0

Gestion de la réponse

Copier

```
@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 this app might request.
    }
}
```

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 : `and_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 allons cette fois-ci utiliser une méthode plus moderne de la classe Activity (Version 1.2.0 du 10 février 2021) :
Utiliser le contrat `RequestPermission`.
Pour cela utilisons au choix le Live Template `and_request_permission_single` ou `and_request_permission_multiple` en fonction des besoins.

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);
```

Copier

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 contacts (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 gradle de base :

Copier

```
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

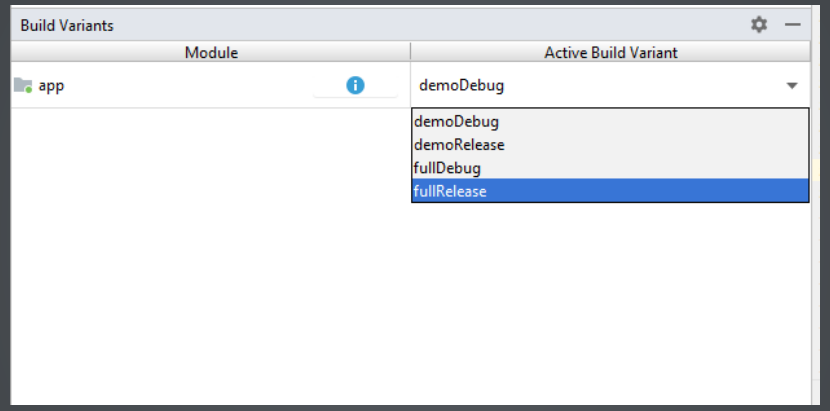
Nous pouvons définir plusieurs dérivés de notre application principale, via les `flavors`.
Tous les détails ici :
https://developer.android.com/studio/build/build-variants#product-flavors.

Ajoutons le code suivant :

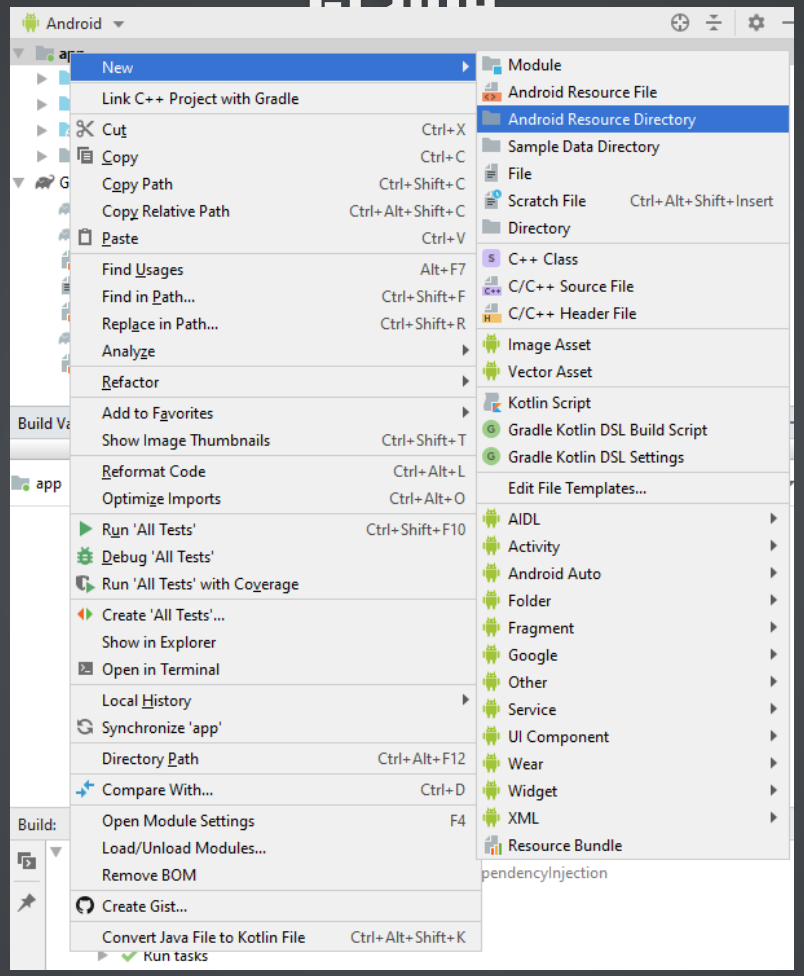
Copier

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

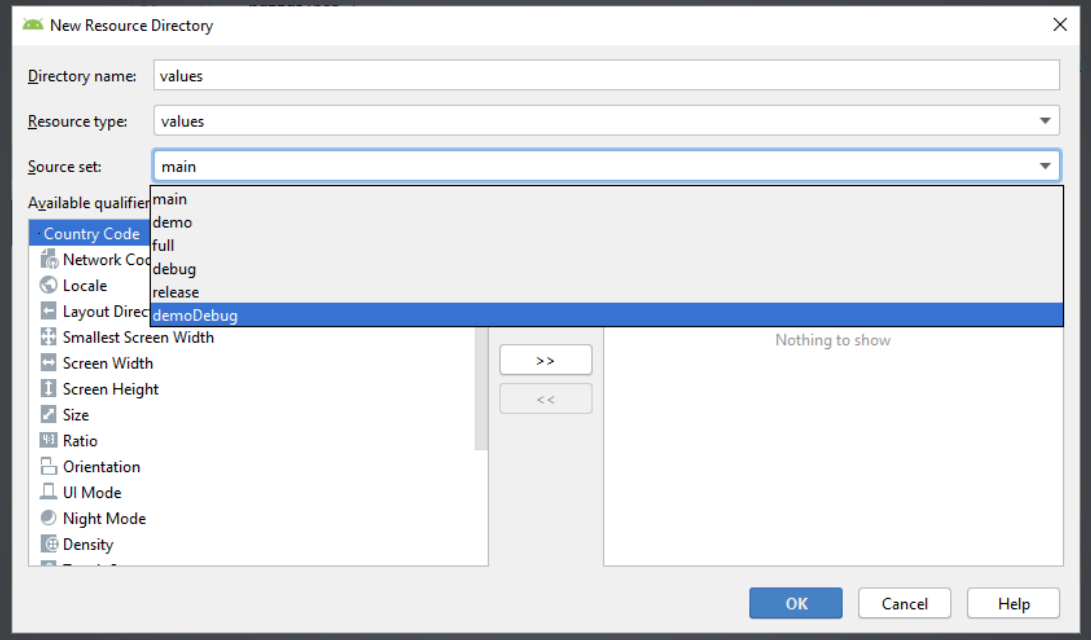
Gradle


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

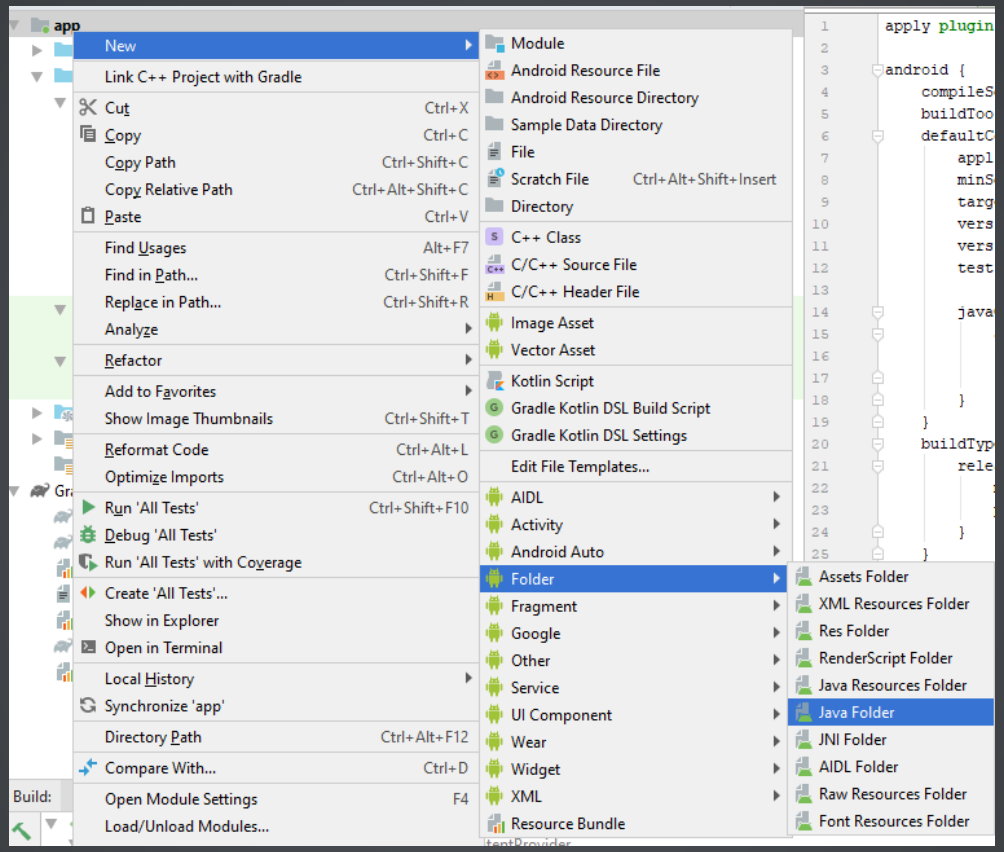
Gradle


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

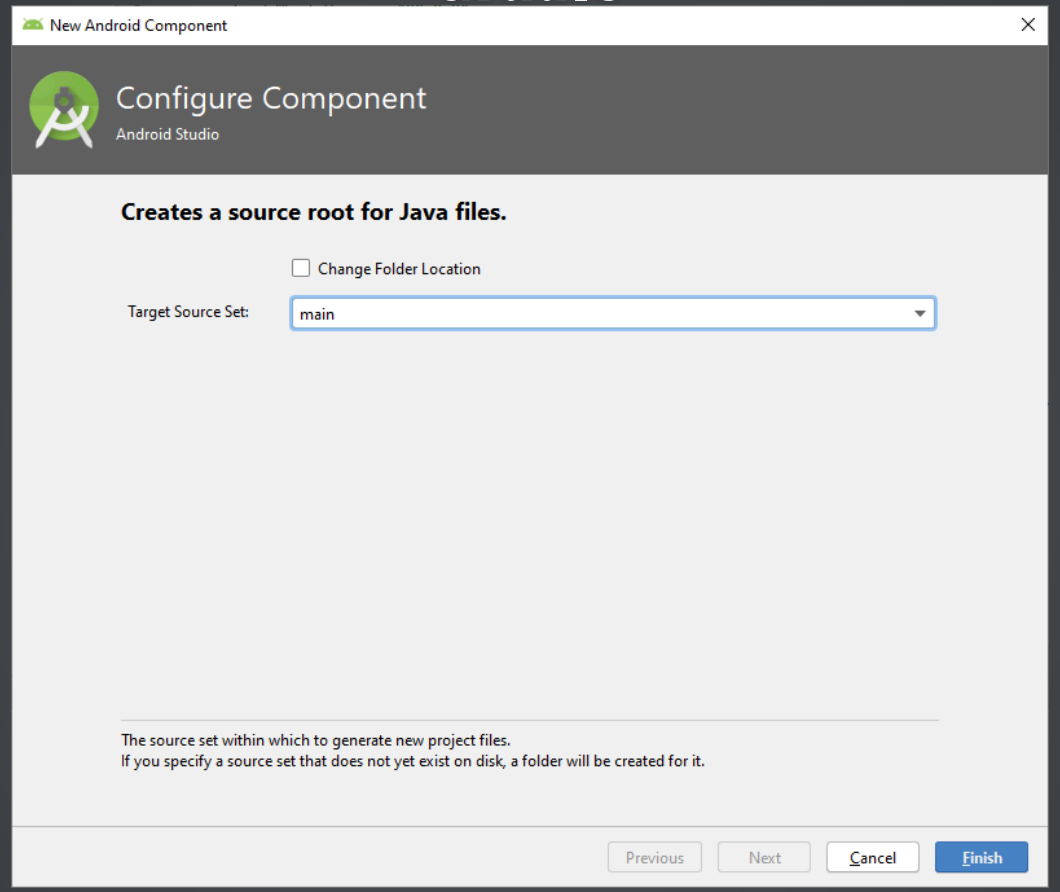
Gradle


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

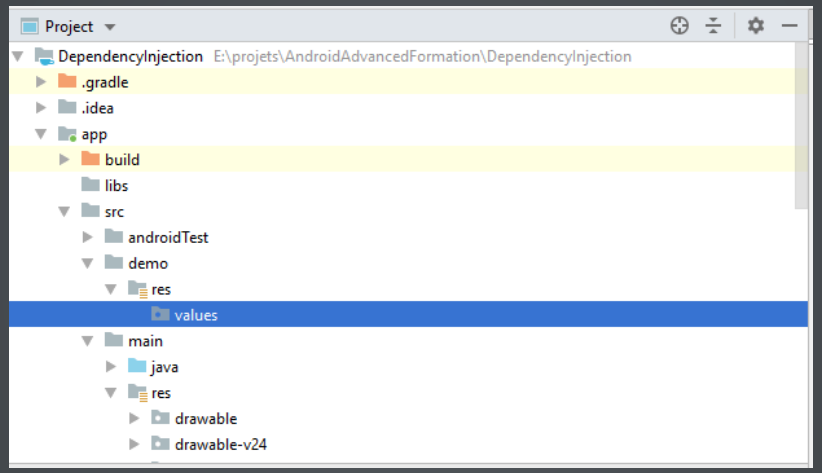
Gradle


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
Load/Unload Modules...
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
Assets Folder
XML Resources Folder
Res Folder
RenderScript Folder
Java Resources Folder
Java Folder
JNI Folder
AIDL Folder
Raw Resources Folder
Font Resources Folder
Build:
apply plugin
android {

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.

Gradle

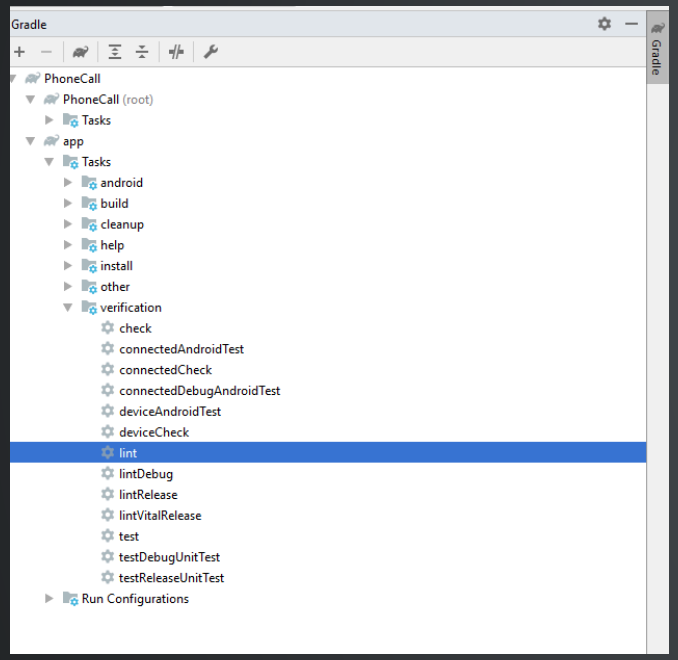
Mise en variable des versions

Copier

```
ext {
    appcompatVersion = '1.1.0'
    constraintLayoutVersion = '1.1.3'
    junitVersion = '4.12'
    runnerVersion = '1.2.0'
    espressoVersion = '3.2.0'
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "androidx.appcompat:appcompat:$appcompatVersion"
    implementation "androidx.constraintlayout:constraintlayout:$constraintLayoutVersion"
    testImplementation "junit:junit:$junitVersion"
    androidTestImplementation "androidx.test:runner:$runnerVersion"
    androidTestImplementation "androidx.test.espresso:espresso-core:$espressoVersion"
}
```

Améliorer son code source avec Lint

Lancer lint manuellement :

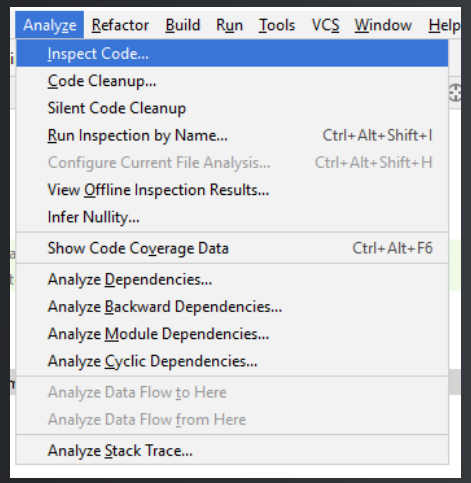


Ouvrez le rapport.

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

Copier

```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" />
    </issue>

    <!-- Change the severity of hardcoded strings to "error" -->
```

Optimisation

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

Optimisation de l'APK avec ProGuard.

Nous allons suivre le tutoriel suivant : Getting Started with ProGuard.
Certains ajustements 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.

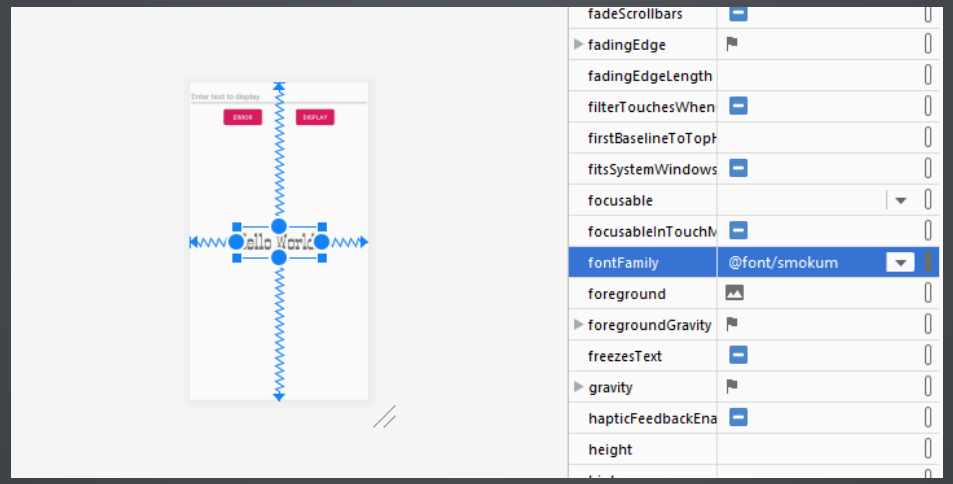
Aller un peu plus loin avec le Material Design.

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

- Les polices de caractères.
- TextView ajustable.
- Résumé des bonnes pratiques.

Fonts externes

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



SeekBar

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

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.

Bouton flottant

Ajoutons un bouton flottant : Référence

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`

Déclaration des couleurs

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

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, ...

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.

Copier

```
<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>
```

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)

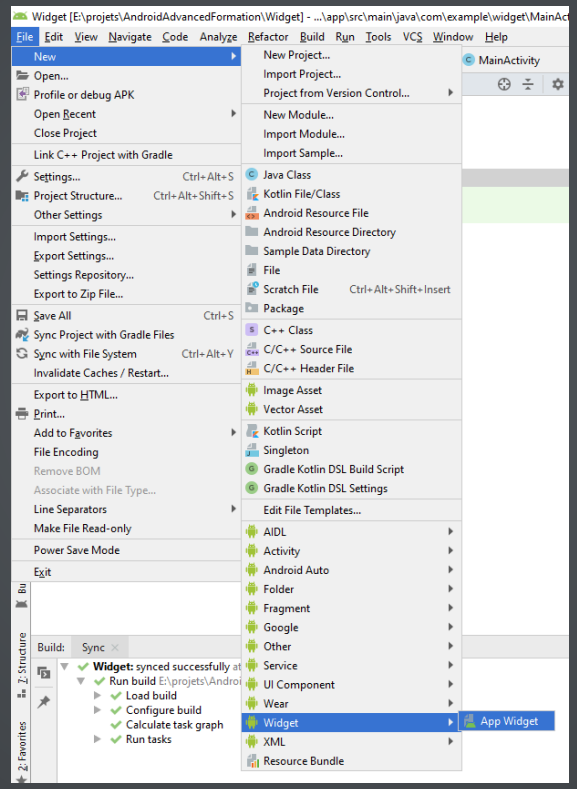
Le matérial design

Pour tous les détails sur le Material Design 3.
Il s'agit d'une évolution du Material Design 2.

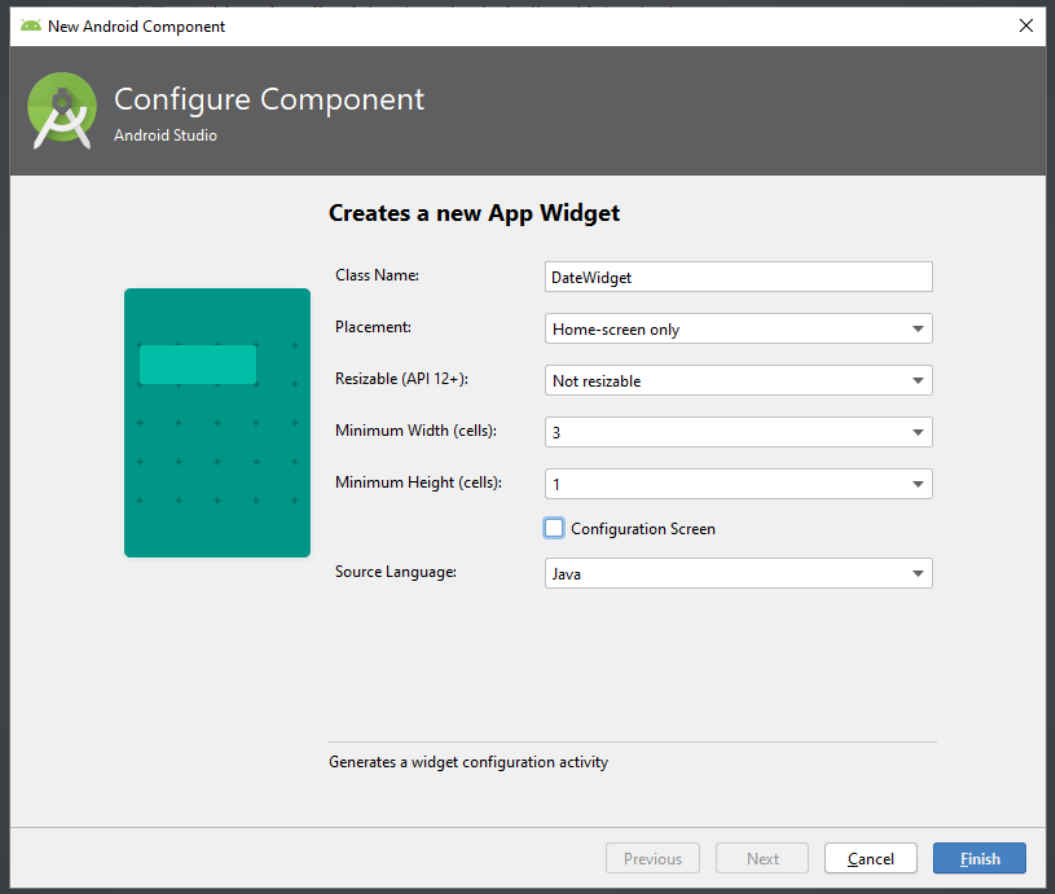
Mécanismes des widgets

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

Ajoutons un widget :



Paramétrons le template :



Remplaçons :

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

Copier

Par :

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

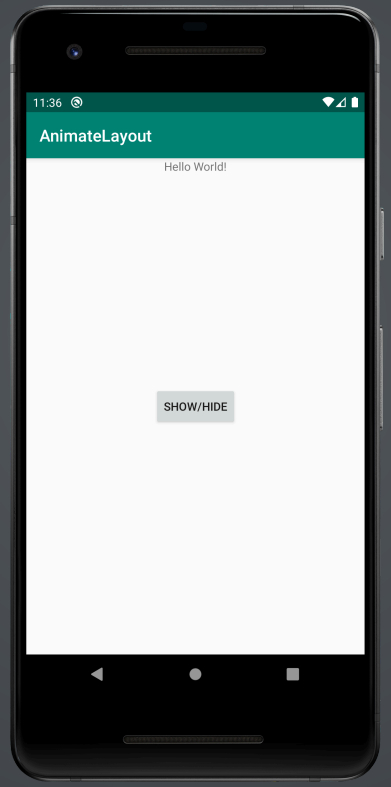
Copier

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.

Ajout d'animations

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



Définissons l'attribut :

```
TextView tvText;
```

Copier

Définissons le code du bouton :

```
tvText = findViewById(R.id.text);

findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        tvText.setVisibility(tvText.getVisibility() == View.VISIBLE ? View.GONE : View.VISIBLE );
    }
});
```

Copier

Testons.

Ajoutons un attribut dans le layout parent :

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

Copier

Testons de nouveau.

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>
```

Copier

Ajoutons le en background de notre layout principal.

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.

Haut niveau d'animation

Un exemple d'interface animée : Application de pizza.

`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"
```

Copier

`region`

Dans notre code, il est possible d'utiliser le commentaire `// region nomDeNotreRegion` puis `// endregion` pour regrouper des blocs de code, afin de mieux les organiser.
Pour visualiser l'effet, utilisons l'onglet `Structure` présent à gauche de notre interface.

Dernière version d'une librairie

Il arrive régulièrement de trouver un tutoriel utilisant une librairie, mais la version n'est plus d'actualité. Pour déterminer la dernière version utilisable dans notre projet, le plus facile est de consulter le serveur de ressources MVN repository.

Librairies et services

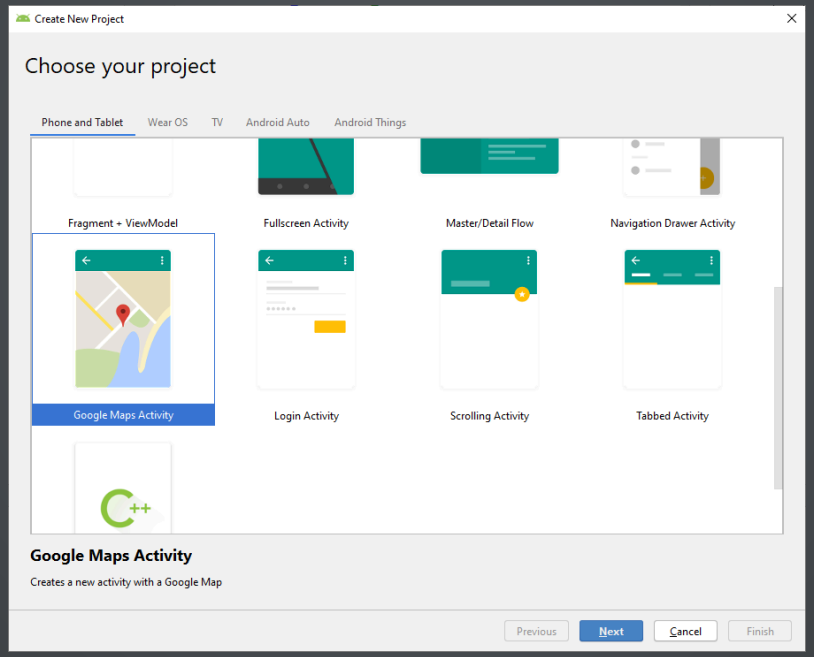
Nous allons voir plusieurs services proposés par la librairie Google Play Services :

- Google Maps
- Géolocalisation
- LeaderBoard

Librairies et services

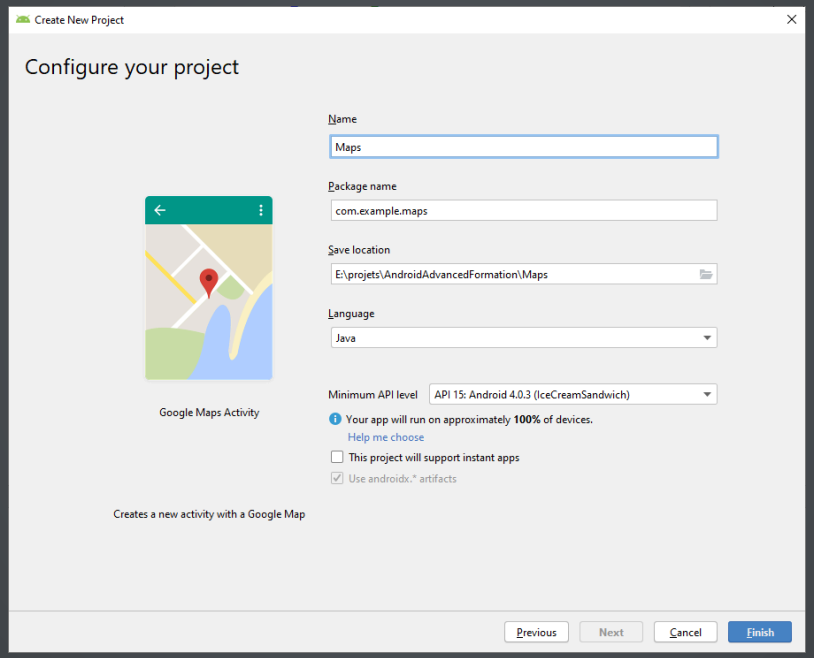
Utiliser les Google Play Services.

Nous allons par exemple mettre en place une Google Maps.



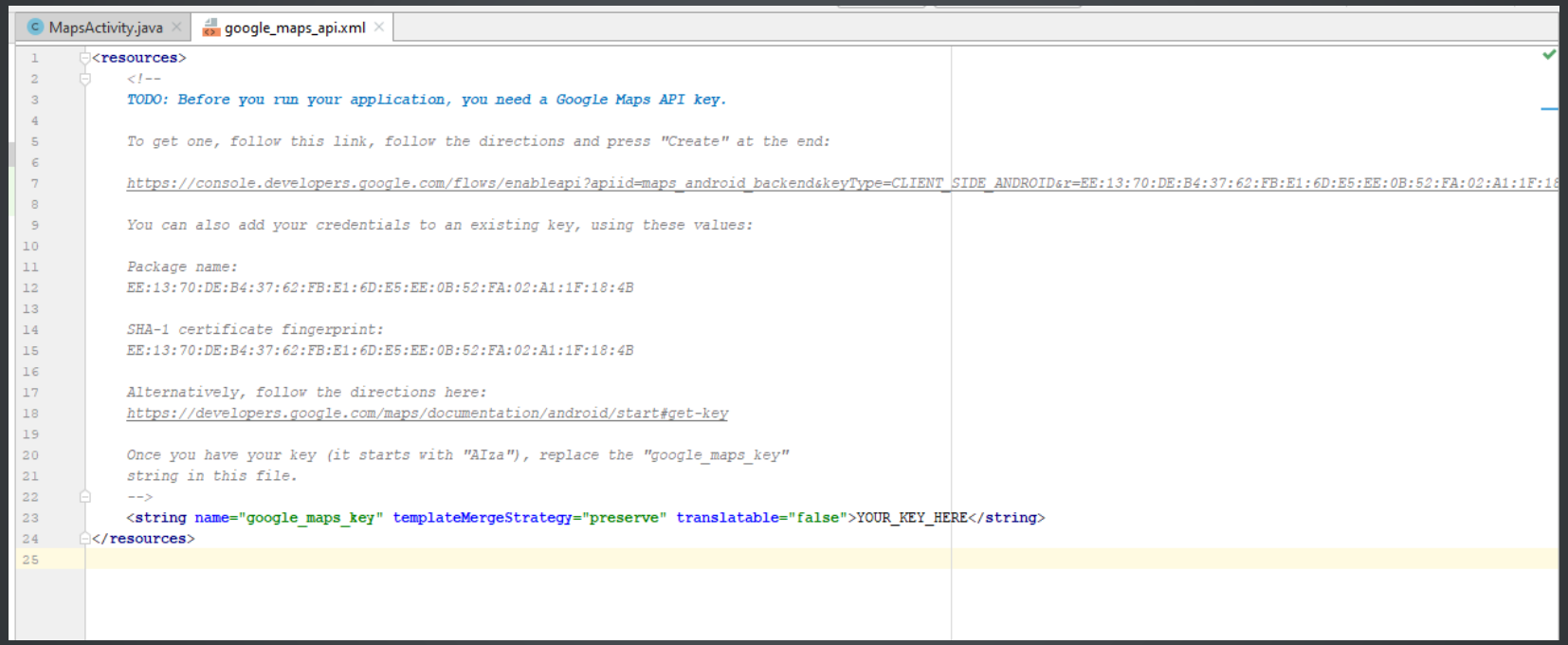
Librairies et services

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 :

Copier

```
@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 at 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 :

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

Copier

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

```
private Marker currentPosition;
```

Copier

Librairies et services

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

Copier

```
FusedLocationProviderClient client = LocationServices.getFusedLocationProviderClient(this);

// Get the last known location
client.getLastLocation().addOnCompleteListener(this, new OnCompleteListener<Location>() {
    @Override
    public void onComplete(@NonNull Task<Location> task) {
        // TODO
    }
});
```

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(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN)));
```

Copier

Librairies et services

Gestion des autorisations

Nous avons l'erreur suivante :

```
com.google.android.gms.tasks.RuntimeExecutionException: java.lang.SecurityException: Client must have ACCESS_COARSE_LOCATION or ACCESS_FINE_LOCATION permission to perform any location operations.
```

Copier

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.

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 du sélecteur doit être la couleur du fond de l'application.

Récupération du code couleur du fond de l'application :

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

Copier

Librairies et services

Intégration d'un scanner de QRCode

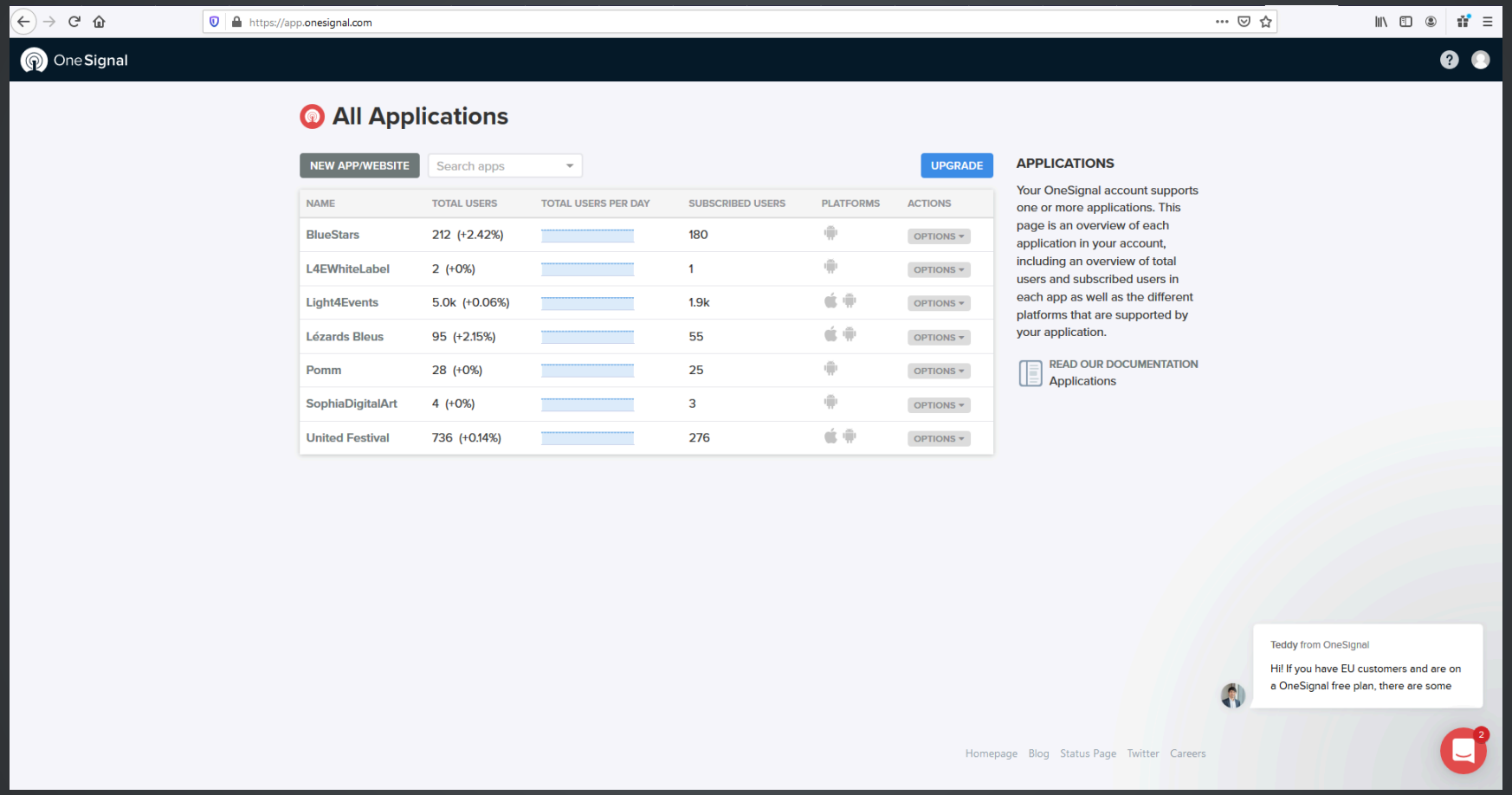
Voir le PDF ici.
Une façon plus modern de réaliser cette tâche est d'utiliser la librairie de Google qui permet de faire de la reconnaissance d'image. Par exemple un scanner de QRCode.
Tous les détails, ici.
L'application de démo, ici.

Librairies et services

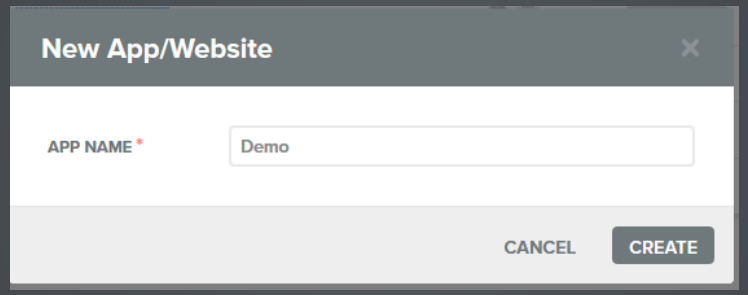
Intégration d'une push notification

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

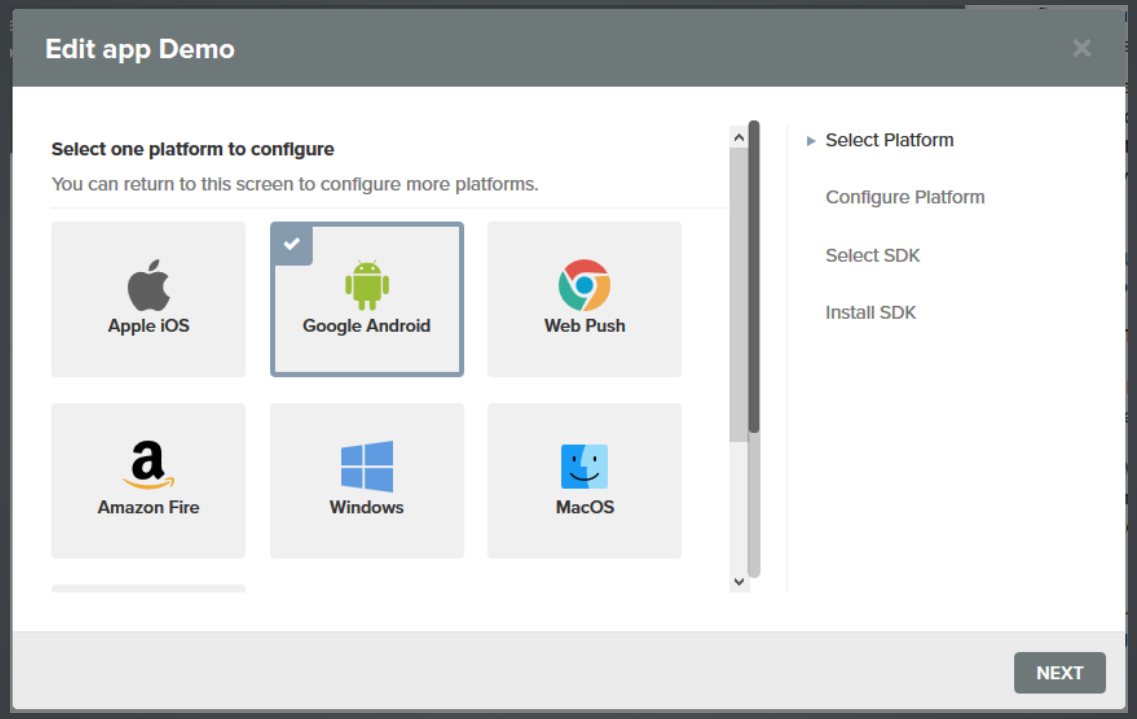
Librairies et services



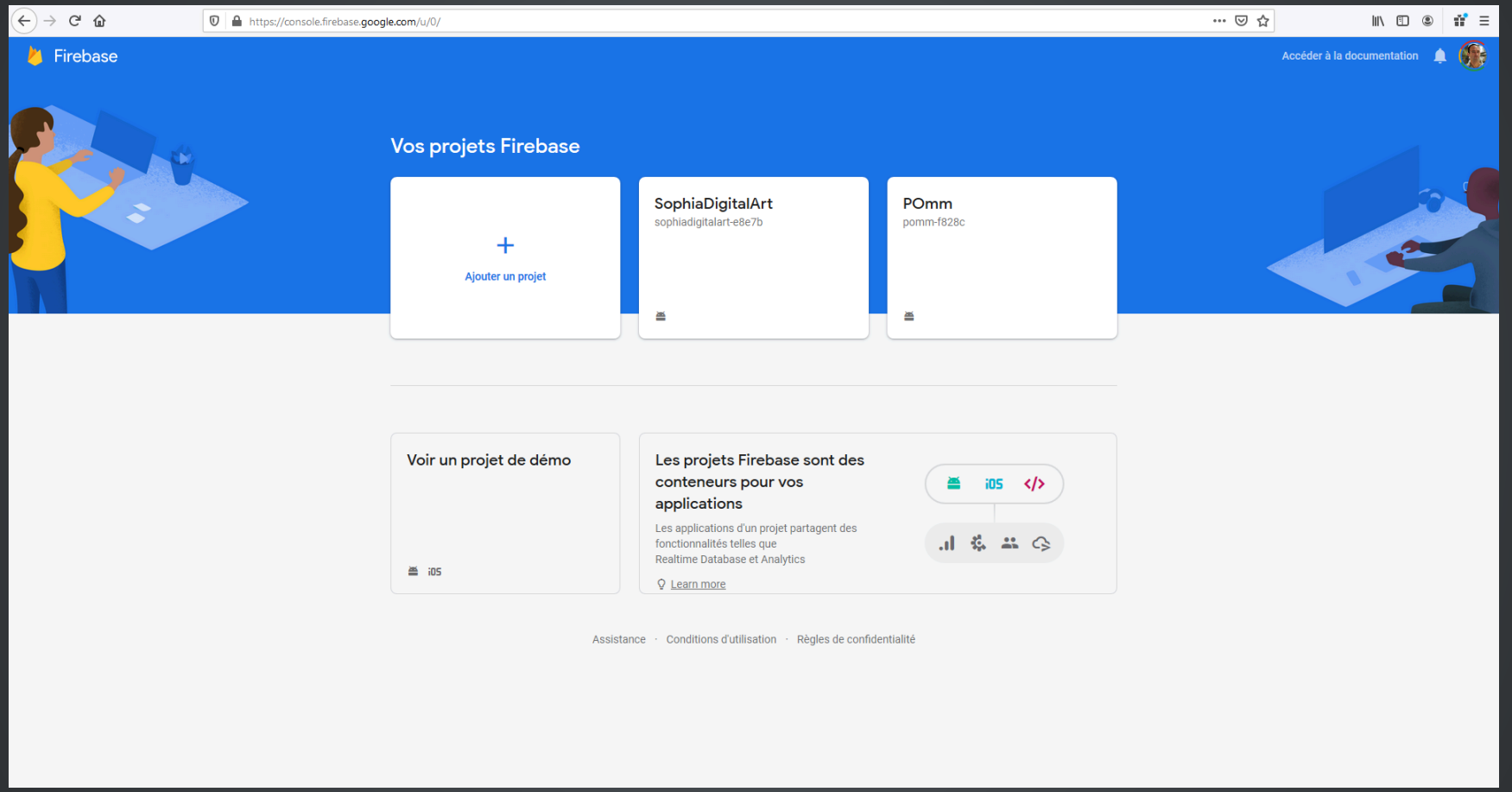
Librairies et services



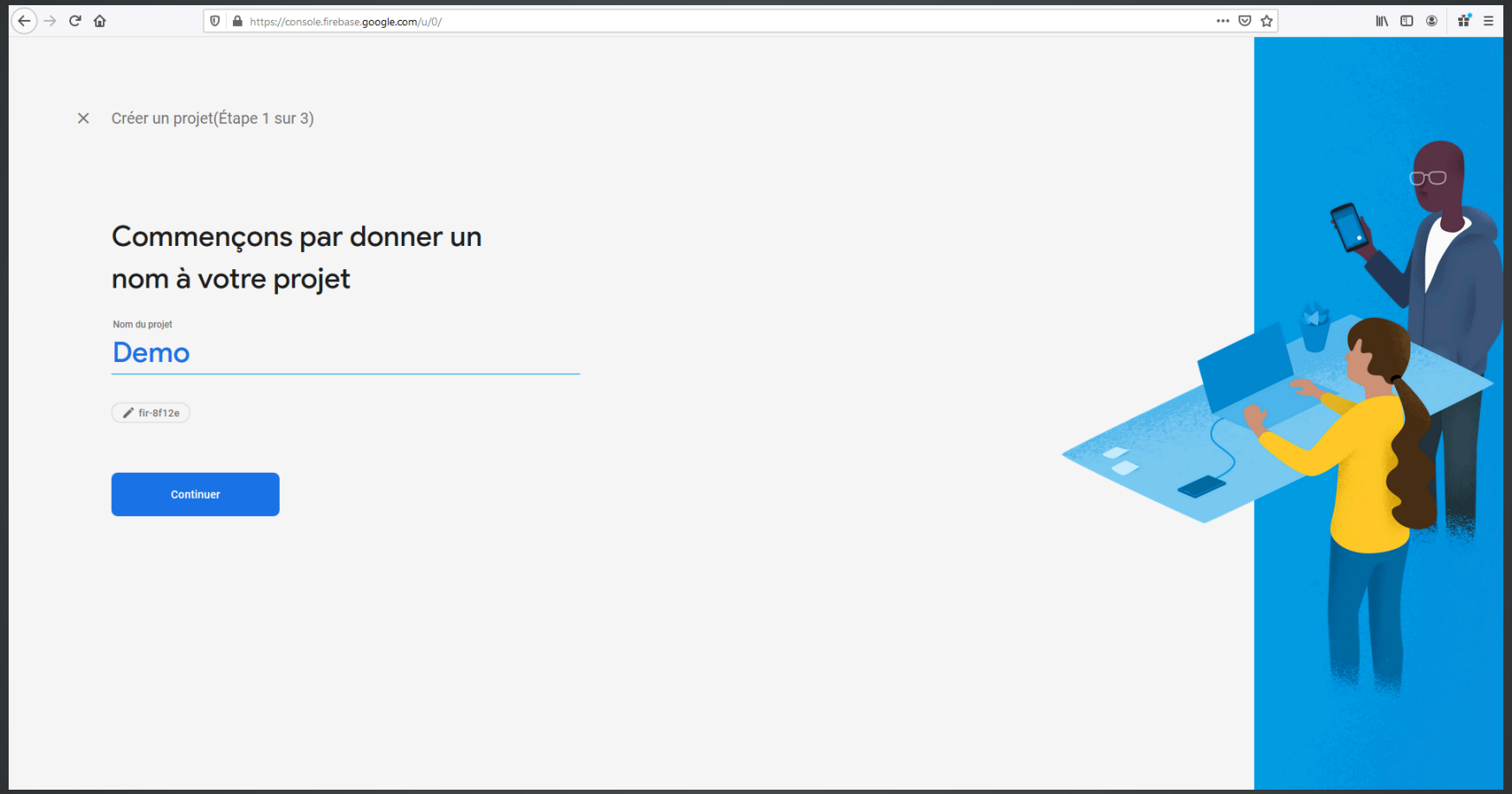
Librairies et services



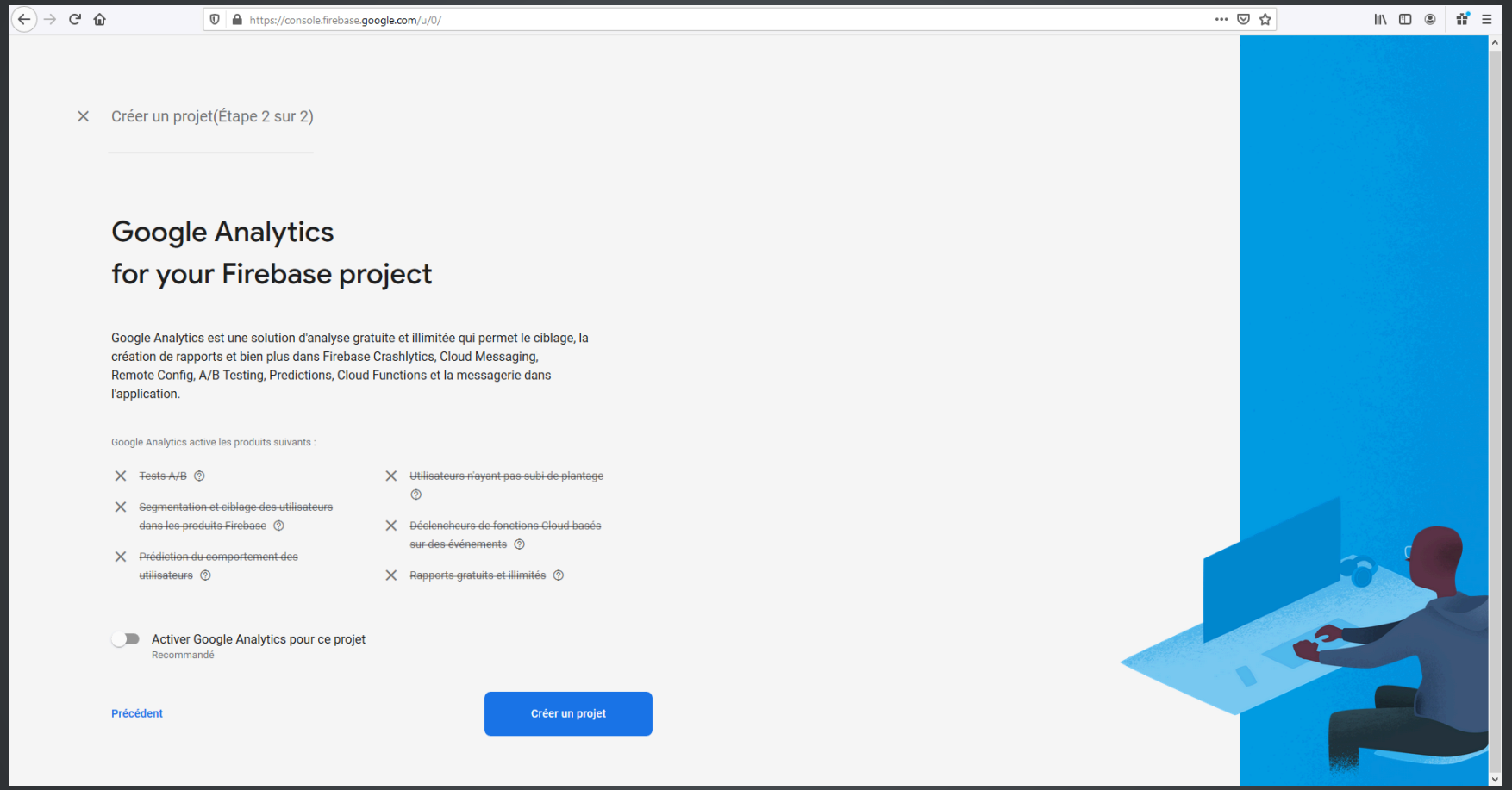
Librairies et services



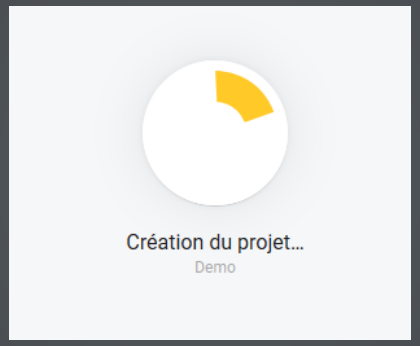
Librairies et services



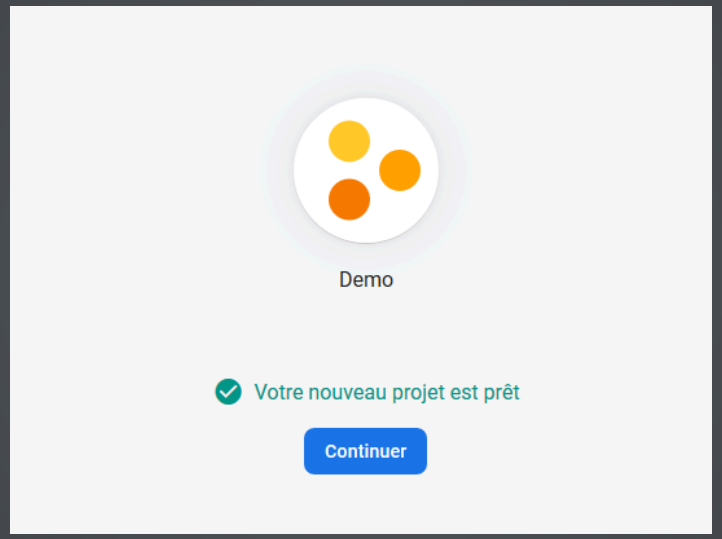
Librairies et services



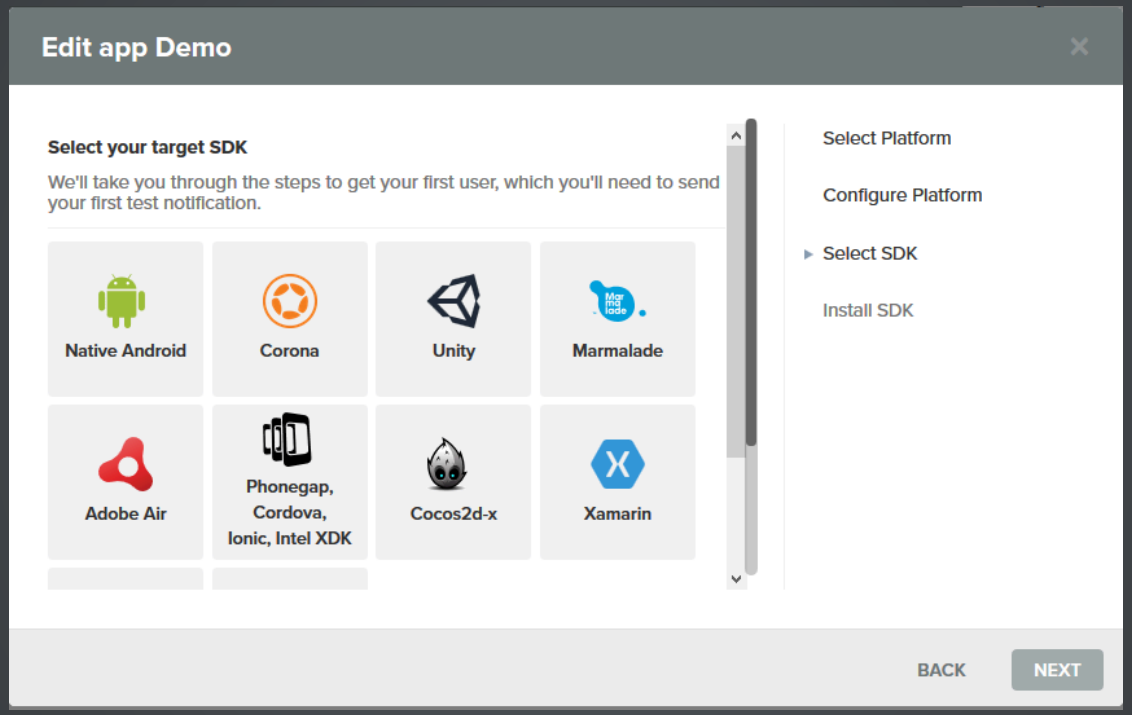
Librairies et services



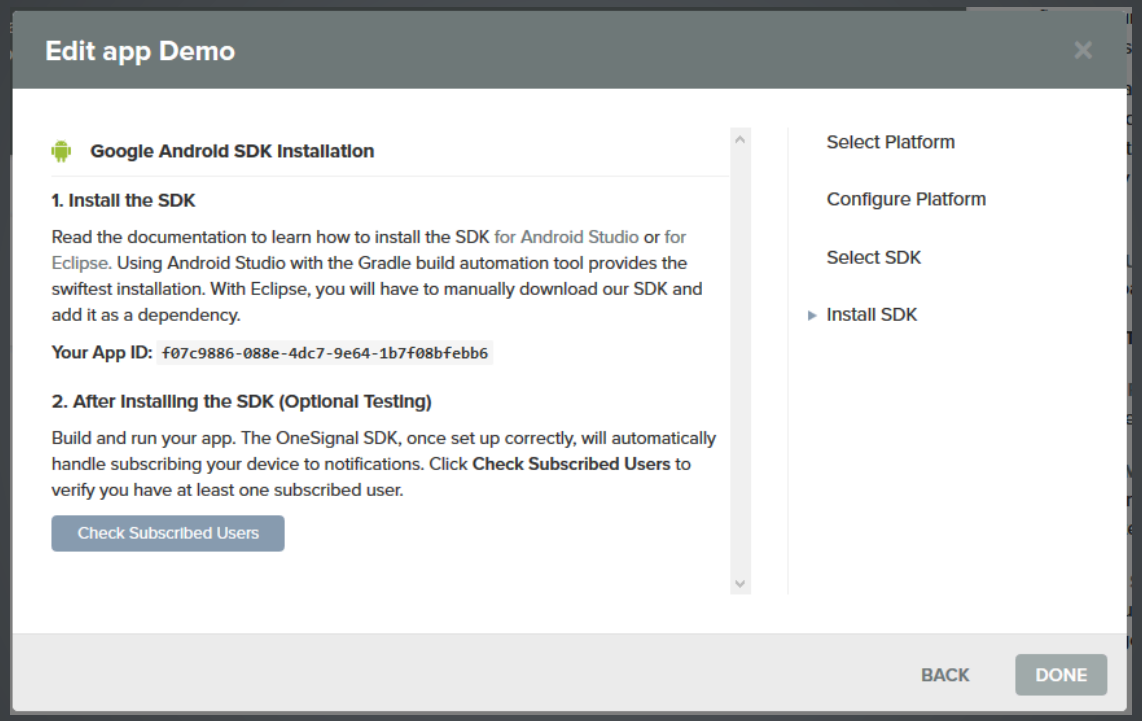
Librairies et services



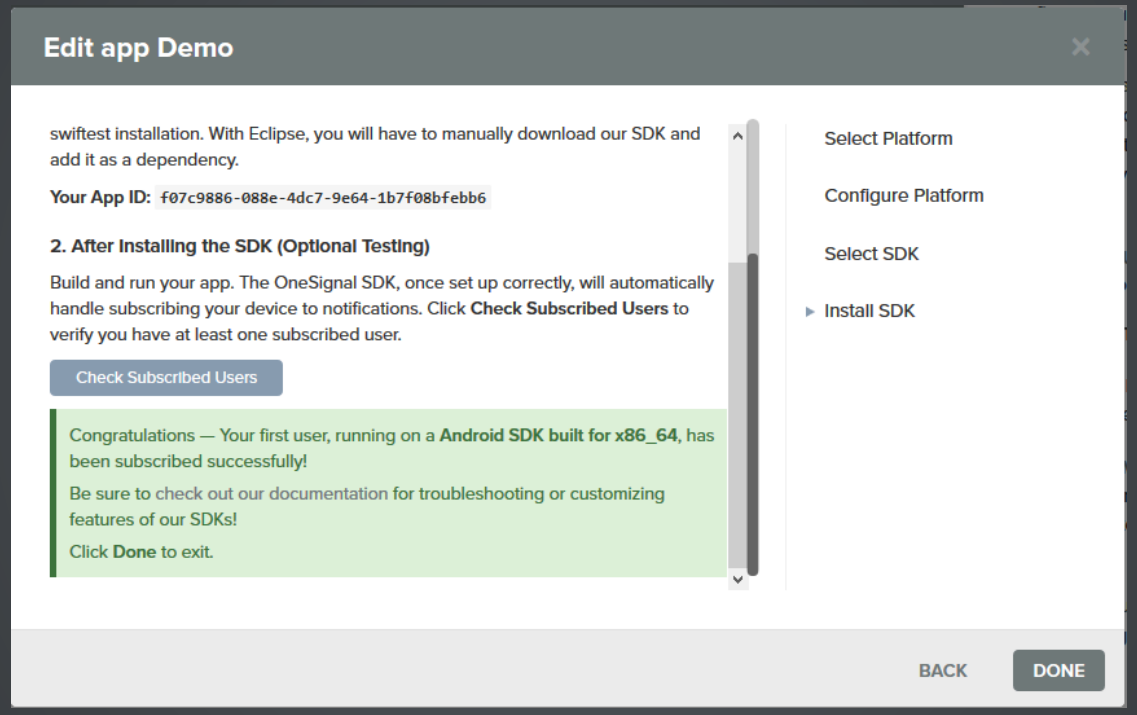
Librairies et services



Librairies et services



Librairies et services



Librairies et services

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);
```

Copier

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);
```

Copier

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 allons suivre le tutoriel suivant :
https://developer.android.com/training/dependency-injection.

Dans un second temps, nous allons suivre le tutoriel suivant :
Using Dagger 2 for dependency injection in Android.
D'autres tutoriels sont disponibles : ici, ou ici.
Ma préférence va à Koin.

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 créer le modèle, nous allons utiliser le menu :
`File / New / Ui Component / Custom View`

Custom View

Utilisation du modèle

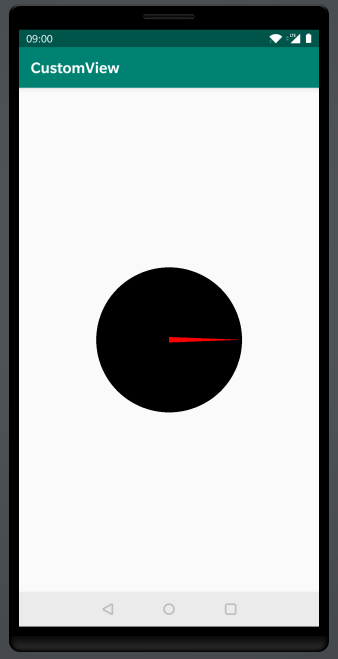
Nous allons suivre les étapes suivantes :

- Utilisation de la vue custom.
- Détail des attributs.
- Détail du code.

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`.

Custom View

Solution (Java)

On définit l'attribut suivant :

```
1 private int mAngle = 0;
```

Copier

Et le setter associé :

```
1 public void setmAngle(int mAngle) {
2         this.mAngle = mAngle;
3         invalidate();
4 }
```

Copier

Custom View

Solution (Java)

Et enfin le code pour dessiner la vue :

Copier

```
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);

    int cx = getWidth() / 2;
    int cy = getHeight() / 2;

    Paint paint = new Paint();

    // Définir la flèche
    Path northPath = new Path();
    northPath.moveTo(0, -cy);
    northPath.lineTo(-10, 0);
    northPath.lineTo(10, 0);
    northPath.close();

    // Dessiner le cercle de fond
    // Définir le rectangle contenant le cercle que l'on va dessiner
```

Custom View

Solution complète (Java)

Copier

```java
public class MyView extends View {
    private int mAngle = 0;

    public MyView(Context context) {
        super(context);
    }

    public MyView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public MyView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
```

La gestion des données

- Les préférences.
- Les fichiers, le stockage interne et externe.
- SQLite / Room.
- Les Content Provider.

Les préférences

Stocker une préférence

Le framework nous met à disposition une possibilité de stocker des informations dans l'application : les SharedPreferences. Sous forme de clé/valeur, nous pouvons stocker de l'information, dans un fichier, lié à l'application. Pour cela, nous utilisons le code suivant :

```
1 val sharedPref = PreferenceManager.getDefaultSharedPreferences(applicationContext)
2 val editor = sharedPref.edit()
3 editor.putString(SHARED_PREFS_VALUE, "Value")
4 editor.commit()
```

Copier

Ou plus rapidement le LiveTemplate : and_SharedPreferences_save.
Nous n'oublions pas d'ajouter au gradle, la dépendance :
`implementation 'androidx.preference:preference-ktx:1.2.1'`.

Les préférences

Charger une préférence

Pour récupérer la valeur stockée, nous utilisons :

```
1 val sharedPref = PreferenceManager.getDefaultSharedPreferences(applicationContext)
2 val defaultValue = resources.getInteger(R.string.value_default)
3 sharedPref.getInt(SHARED_PREFS_VALUE, defaultValue)
```

Copier

Ou plus rapidement le LiveTemplate : and_SharedPreferences_load.
Nous pouvons bien entendu mutualiser l'objet `sharedPref`.

La version plus moderne consiste à utiliser un datastore.

Les fichiers

Copier

```
@file:JvmName("FileTools")

package com.example.myapplication

import android.content.Context
import java.io.File

fun readFile(context: Context, isExternal: Boolean, path: String, fileName: String): String {
    val currentFile = getFile(context, isExternal, path, fileName)

    // READ
    // val inputAsString = FileInputStream(currentFile).bufferedReader().use { it.readText() }

    var sb = StringBuilder()
    currentFile?.forEachLine { sb.append( "$it\n") }

    return sb.toString()
```

Les fichiers

Utilisation

Utilisons les méthodes que nous venons d'écrire :

Copier

```
var isExternal = false
var path = "trs"
var fileName = "test.txt"
var append = true

writeFile(this@MainActivity, isExternal, path, fileName, "AAAAA\n", append)

val stringContent = readFile(this@MainActivity, isExternal, path, fileName)
if (BuildConfig.DEBUG) {
    Log.d(TAG, "onCreate $stringContent")
}
```

SQLite / Room

Android propose un système de base de donnée : SQLite, celui-ci étant un peu complexe à utiliser, plusieurs outils ont été mis en place pour gérer une base de donnée plus facilement (des ORM : Mapping Objet-Relationnel). L'ORM officiel sur Android est maintenant ROOM.
Voyons un peu plus en détail la mise en œuvre de cette base de donnée. Nous allons étudier 2 méthodologies différentes (utilisant 2 outils différents) :
1. En utilisant la documentation officielle : ROOM.
2. En utilisant des LivesTemplates (and_db_room_00_documentation) (En JAVA uniquement).

SQLite / Room

Aller plus loin

Les bases de données ne pouvant pas être appelées dans le thread graphique et pouvant évoluer dans le temps, il est recommandé d'utiliser les LiveData pour gérer l'affichage des données. Nous pouvons suivre le tutoriel suivant : Android Room with a View - Kotlin.

Les Content Provider

Définition

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

Les Content Provider

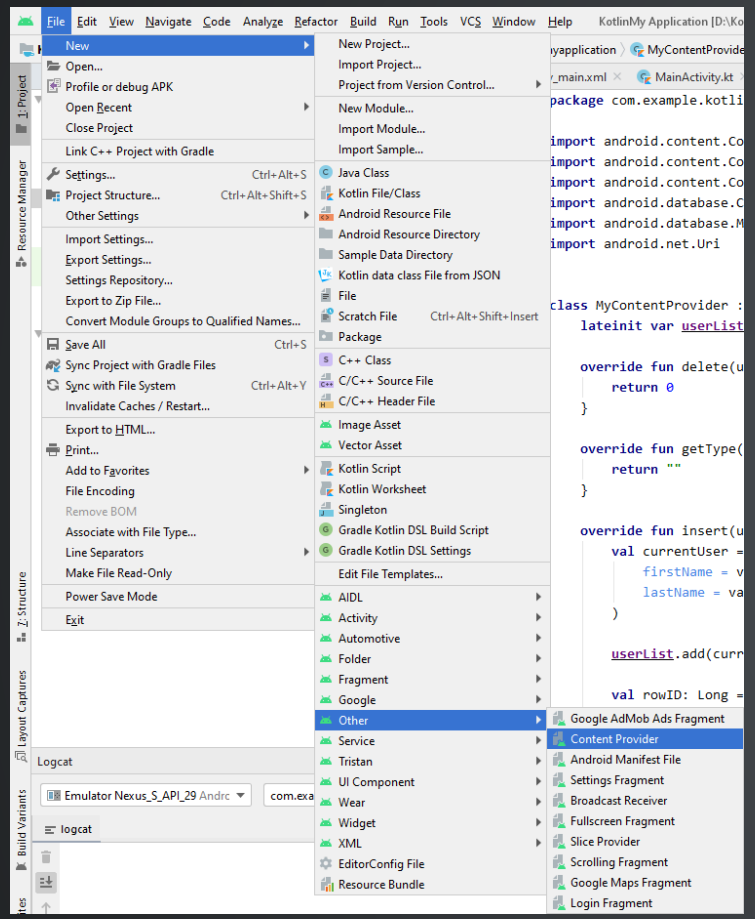
Utilisation d'un content provider

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

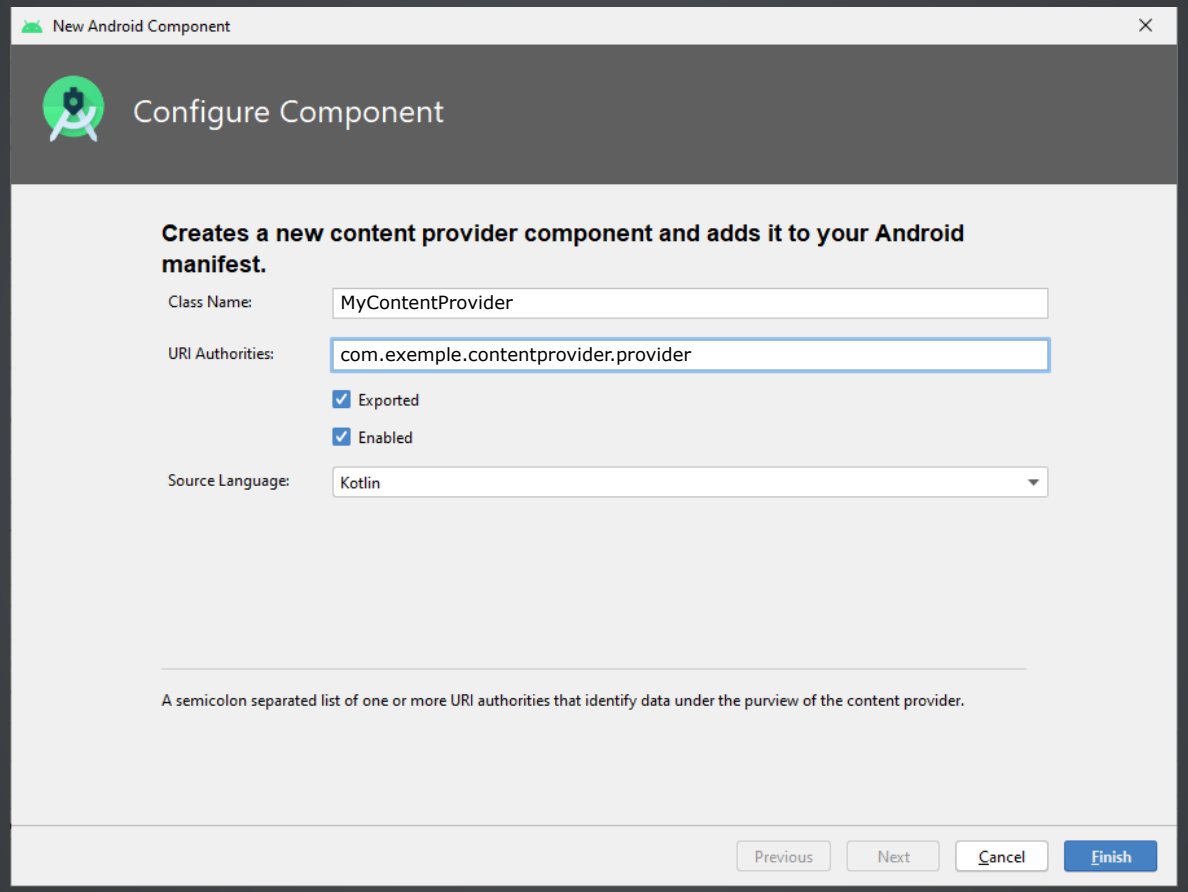
Copier

```
Uri media = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
String[] projection = { MediaStore.Audio.Media._ID,           // 0
        MediaStore.Audio.Media.ARTIST,        // 1
        MediaStore.Audio.Media.TITLE,         // 2
        MediaStore.Audio.Media.ALBUM_ID,      // 3
        MediaStore.Audio.Media.ALBUM,         // 4
        MediaStore.Audio.Media.DATA,          // 5
        MediaStore.Audio.Media.DISPLAY_NAME,  // 6
        MediaStore.Audio.Media.DURATION };    // 7

String selection = MediaStore.Audio.Media.IS_MUSIC + " != 0";

Cursor cursor = getContentResolver().query(media,projection,selection,null,null);
while(cursor.moveToNext()){
    if(BuildConfig.DEBUG){
        Log.d(TAG, "onCreate " +
                        cursor.getString(0) + " " +
```

Les Content Provider



Les Content Provider



Les Content Provider

POJO : La classe User

Déclarons une nouvelle classe qui va contenir nos données, la classe `User` :

```
1 data class User(var firstName: String, var lastName: String)
```

Copier

Les Content Provider

Query

Pour commencer, nous allons simplement retourner une liste en dur via le code :

Copier

```
override fun query(
    uri: Uri, projection: Array<String>?, selection: String?,
    selectionArgs: Array<String>?, sortOrder: String?
): Cursor? {

    val cursor = MatrixCursor(arrayOf("name", "firstname"))

        cursor.newRow()
            .add("name", "SALAUN")
            .add("firstname", "Tristan")

        cursor.newRow()
            .add("name", "SNOW")
            .add("firstname", "John")

    return cursor
}
```

Les Content Provider

ArrayList

Afin d'avoir une certaine persistance et aller plus rapidement, dans l'implémentation du ContentProvider, nous allons utiliser une ArrayList.
Déclarons une variable de classe :

```
lateinit var userList: ArrayList<User>
```

Copier

Nous allons initialiser la liste dans la méthode onCreate :

```
override fun onCreate(): Boolean {
    // Init some values at the create
    userList = arrayListOf(User("Tristan", "SALAUN"))
    return true
}
```

Copier

Notre méthode query va donc prendre la forme :

```
override fun query(
    uri: Uri, projection: Array<String>?, selection: String?,
    selectionArgs: Array<String>?, sortOrder: String?
    ): Cursor? {

    val cursor = MatrixCursor(arrayOf("name", "firstname"))

    for (curentUser in this.userList) {
        cursor.newRow()
        .add("name", curentUser.lastName)
        .add("firstname", curentUser.firstName)
    }

    // cursor.newRow()
    // .add("name", "SALAUN")
    // .add("firstname", "Tristan")
```

Copier

Les Content Provider

La méthode insert prendra quant à elle la forme :

Copier

```
override fun insert(uri: Uri, values: ContentValues?): Uri? {
    val currentUser = User(
        firstName = values?.getAsString(UserContract.MyDatas.KEY_COL_FIRSTNAME),
        lastName = values?.getAsString(UserContract.MyDatas.KEY_COL_NAME)
    )

    userList.add(currentUser)

    val rowID: Long = userList.size.toLong()

    val _uri =
    ContentUris.withAppendedId(UserContract.MyDatas.CONTENT_URI, rowID)
    context!!.contentResolver.notifyChange(_uri, null)
    return _uri
}
```

Les Content Provider

Partager notre Content Provider

L'objectif étant que notre Content provider soit utilisable par d'autres applications, il est d'usage de définir une classe Contrat, dans notre cas la classe UserContract :

Copier

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

class UserContract {

    companion object {
        // The Authority
        private const val AUTHORITY = "com.exemple.contentprovider.provider"

        // The path to the data… and explain
        private const val PATH_TO_DATA = "users" //Vous pouvez déclarer plusieurs paths (les paths utilisent les /)
    }

    interface MyDatas : BaseColumns {
        companion object {

            // The URI and explain, with example if you want
```

Les Content Provider

Création de ContentProviderClient

C'est la classe UserContract que nous utilisons aussi dans le projet client de notre ContentProvider : ContentProviderClient.
Créons le projet `ContentProviderClient`.

Les Content Provider

Lecture dans ContentProviderClient

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

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

Copier

Les Content Provider

Insertion dans ContentProviderClient

Ajoutons maintenant un appel pour insérer une donnée dans le `ContentProvider` :

Copier

```
// Defines an object to contain the new values to insert
val newValues = ContentValues().apply {
    /*
     * Sets the values of each column and inserts the data. The arguments to the "put"
     * method are "column name" and "value".
     */
    put(UserContract.MyDatas.KEY_COL_FIRSTNAME, "John")
    put(UserContract.MyDatas.KEY_COL_NAME, "Doe")
}

val newUri = contentResolver.insert(
    UserContract.MyDatas.CONTENT_URI,   // The UserDictionary content URI
    newValues                           // The values to insert
)
```

Les Content Provider

Pour aller un peu plus loin : Content provider basics.

La gestion réseau

- Les infos de connectivité. Utiliser HTTP.
- Parser du JSON.
- Les accès aux Web Services : Volley, Retrofit.

Connectivité et HTTP

Commençons par mettre en place une classe utilitaire : `NetworkTool` :

Copier

```
import android.content.Context
import android.net.ConnectivityManager
import android.net.NetworkCapabilities

import android.net.NetworkInfo
import android.os.Build

class NetworkTool {

    companion object {

        // NEED : <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
        private fun isNetworkAvailable(context: Context): Boolean {
            val connectivityManager = context.getSystemService(Context.CONNECTIVITY_SERVICE) as ConnectivityManager
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                val nw = connectivityManager.activeNetwork ?: return false
                val actNw = connectivityManager.getNetworkCapabilities(nw) ?: return false
```

Connectivité et HTTP

Puis notre receiver : `NetworkStateReceiver`

Copier

```
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.net.ConnectivityManager
import android.net.NetworkInfo
import android.util.Log
import java.lang.Boolean

class NetworkStateReceiver: BroadcastReceiver() {

    companion object {
        private const val TAG = "NetworkStateReceiver"
    }

    // post event if there is no Internet connection
    override fun onReceive(context: Context?, intent: Intent) {
        Log.d(TAG, "onReceive $intent")
```

Connectivité et HTTP

Il ne nous reste plus qu'à enregistrer le listenner : nous commençons par déclarer l'attribut de classe :

```
1 private val networkStateReceiver = NetworkStateReceiver()
```

Copier

Puis dans la méthode onCreate on enregistre le listenner :

```
1 registerReceiver(networkStateReceiver, IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION))
```

Copier

Et on le désenregistre dans le onStop :

```
1 unregisterReceiver(networkStateReceiver)
```

Copier

Pour tester, on peut couper la connexion WiFi par exemple, et la réactiver.

Parser du JSON

Nous allons utiliser la librairie GSON.
Nous allons suivre les étapes suivantes :

- Ajouter une référence à la librairie dans nos dépendances.
- Générer du JSON à partir d'objets.
- Parser du JSON pour instancier des objets.

Cet exemple est largement inspiré de : Android : Utiliser Gson pour faciliter l'utilisation du Json

Parser du JSON

Ajout de la dépendance.

Ajoutons une dépendance dans notre fichier build.gradle (Module: app) :

```
dependencies {
    ...
    // JSon parser
    implementation 'com.google.code.gson:gson:2.10.1'
}
```

Copier

Parser du JSON

Générer du JSON à partir d'un objet.

Créons l'objet Book :

```
data class Book(var id: String, var name: String, var author: String, var genre: String, var numpages: Int, var releaseDate: String, var cover: String)
```

Copier

Et remplissons une liste de Book :

```
var list = arrayListOf(
    Book(
        id = "01fsEF", name = "1984",
        author = "George Orwell",
        genre = "Fiction dystopique",
        numpages = 376,
        releaseDate = "1949",
        cover = "1984.png"
    ),
    Book(
        id = "3H1J0n",
        name = "Le Meilleur des mondes",
        author = "Aldous Huxley",
        genre = "Science-fiction",
        numpages = 285,
        releaseDate = "1932",
        cover = "meilleur-des-mondes.jpg"
```

Copier

Parser du JSON

Nous pouvons maintenant transformer notre liste d'objets en JSON.

Copier

```
1 val gson = Gson()
2 val listType = object : TypeToken<ArrayList<Book?>?>() {}.type
3 val jsonResult: String = gson.toJson(list, listType)
4 Log.d(TAG, "onCreate jsonResult = $jsonResult")
```

Parser du JSON

Bonus

Une fonction pour faire un formatage "pretty print" :

```
1 private fun jsonToPrettyFormat(jsonString: String?): String? {
2     val json = Gson().fromJson(jsonString, JsonElement::class.java)
3     val gson = GsonBuilder()
4         .serializeNulls()
5         .disableHtmlEscaping()
6         .setPrettyPrinting()
7         .create()
8     return gson.toJson(json)
9 }
```

Copier

Que l'on appellera par exemple :

```
1 Log.d(TAG, "onCreate jsonResult = ${jsonToPrettyFormat(jsonResult)}")
```

Copier

Parser du JSON

JSON vers objet

Nous allons maintenant réaliser l'opération inverse, consistant à passer du JSON (format texte) vers des objets :

```
1 // Deserialization
2 val bookList2: ArrayList<Book> = Gson().fromJson(jsonResult, listType)
3 Log.d(TAG, "onCreate bookList2 = $bookList2")
```

Copier

Parser du JSON

Autres librairies

Nous avons vu l'usage de GSON, mais il existe d'autres librairies (liste non exhaustive) :

- Jackson
- klaxon
- Moshi
- kotlinx.serialization

Un article comparant les solutions (2019)
Un article comparant les solutions (2020)
Un article comparant les solutions (2022)

Les accès aux Web Services

Il existe plusieurs façons de faire des appels à des web services, par exemple :

- HttpURLConnection (sans librairie).
- Volley (sur GitHub).
- Android Asynchronous Http Client.
- Retrofit.

La solution la plus populaire, semble être Retrofit, c'est celle-là que nous allons utiliser. Il est à noter que Ktor gagne en popularité.

Les accès aux Web Services

Application de news

Nous allons écrire une application nous permettant d'afficher des nouvelles, en provenance d'un site proposant une API gratuitement (pour les développeurs). Les étapes que nous allons suivre sont :

- Création du compte sur le site.
- Ajout de la librairie à notre projet.
- Écriture des fichiers POJO.
- Écriture de l'interface avec l'API.
- Instantiation du client.
- Lancement d'une requête.

La source du sujet est inspiré de : Live data, ViewModel, Retrofit Android Architecture Component.

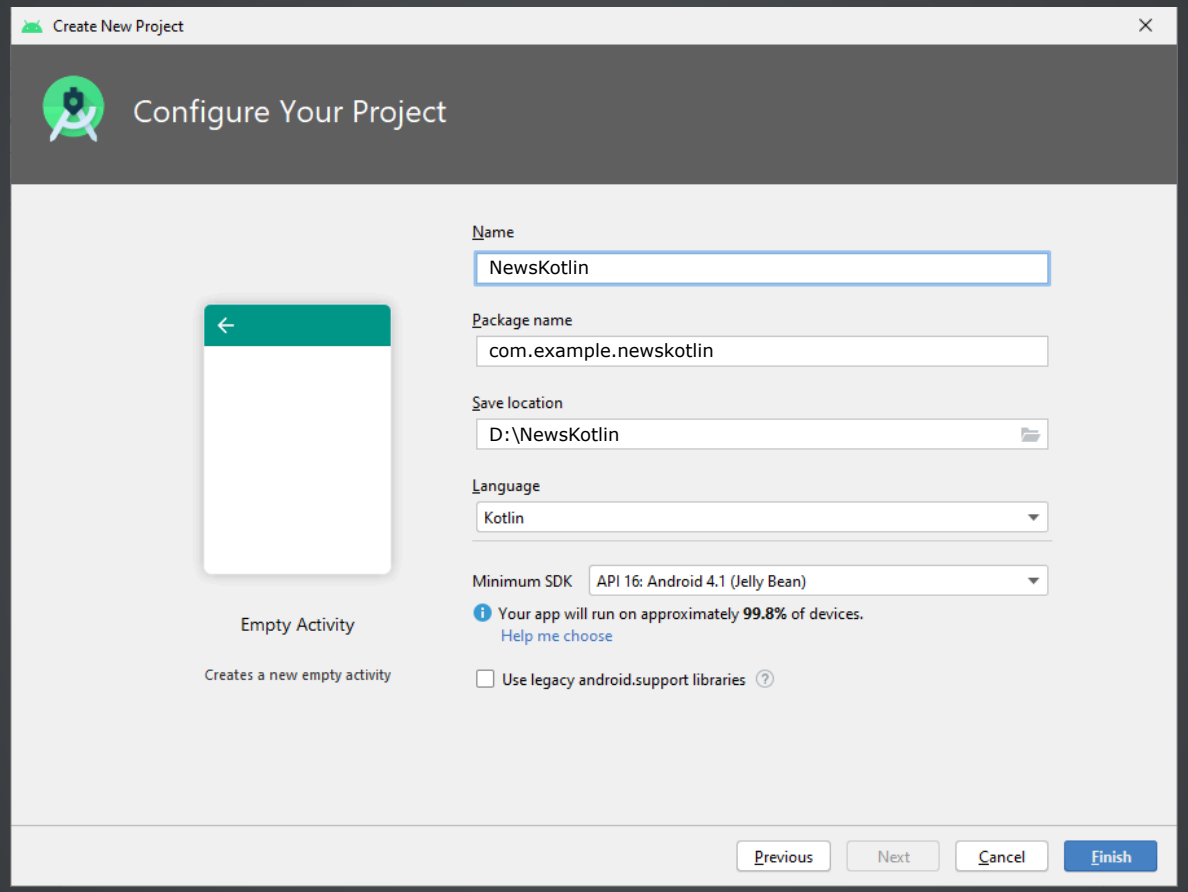
Les accès aux Web Services

Points techniques

Dans cette application, nous verrons les points suivants :

- LiveData.
- La librairie GSON.
- La librairie Retrofit.
- Le service en ligne JSON => POJO.
- Le service de news par API : News API.

Les accès aux Web Services



Les accès aux Web Services

Ajout des librairies à notre projet

Copier

```
// JSon parsing
implementation 'com.google.code.gson:gson:2.10.1'

// Network calls
implementation 'com.squareup.retrofit2:retrofit:2.8.1'
implementation 'com.squareup.retrofit2:converter-gson:2.8.1'

// handle recyclerview
implementation "androidx.recyclerview:recyclerview:1.3.2"

// handle livedata life cycle in fragments
implementation "androidx.fragment:fragment-ktx:1.8.0"

// Handle images loading (choose one)
// implementation 'com.squareup.picasso:picasso:2.71828'
implementation 'com.github.bumptech.glide:glide:4.15.1'
```

Les accès aux Web Services

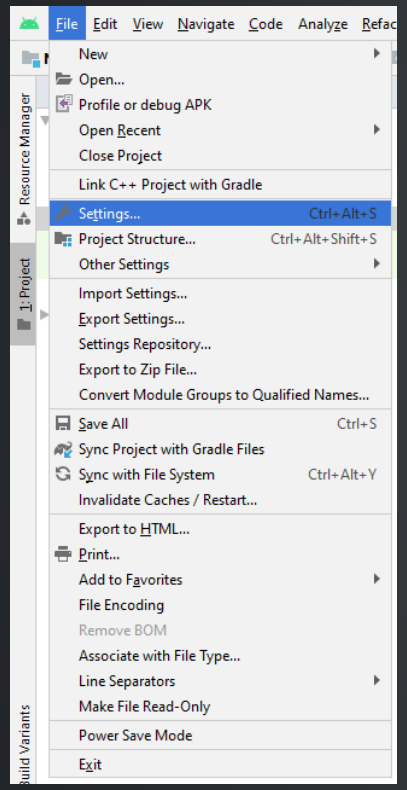
Écriture des fichiers POJO

Pour écrire les fichiers qui correspondent à notre API de news, nous allons suivre les étapes suivantes :

- Nous allons installer le plugin JSON To Kotlin.
- Nous allons utiliser le JSON renvoyé par le site web.
- Nous allons le copier/coller dans le plugin JSON To Kotlin.
- Renseigner le champ Class Name: avec : NewsResponse.
- Il ne reste qu'à cliquer sur Generate.

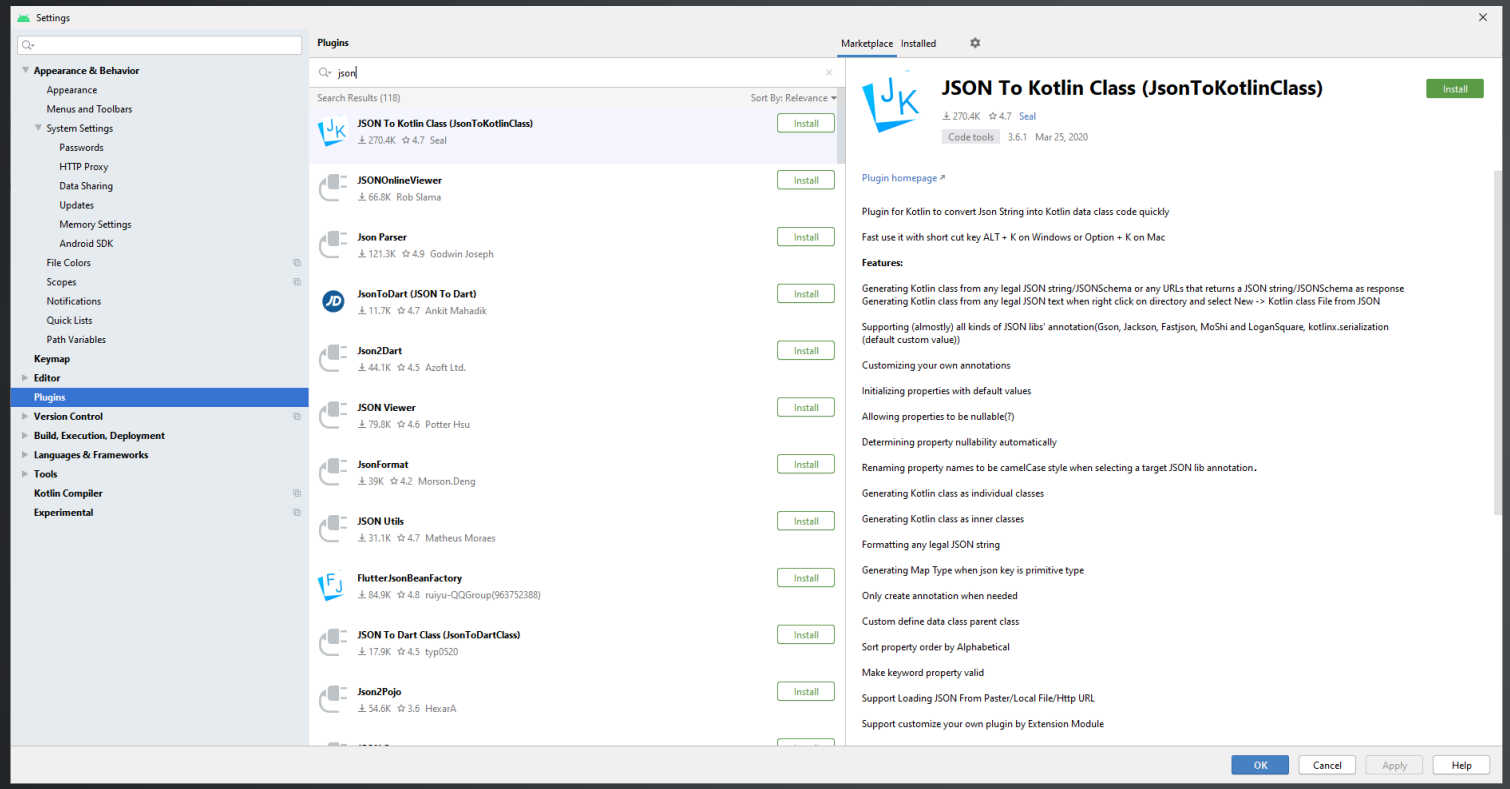
Les accès aux Web Services

Installons le plugin JSON to Kotlin



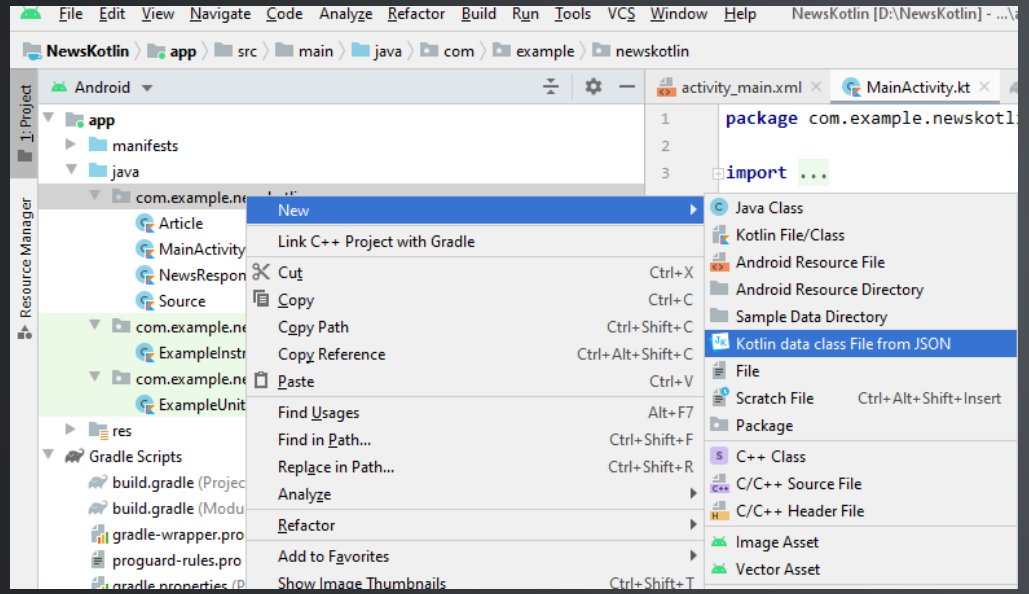
Les accès aux Web Services

Installons le plugin JSON to Kotlin



Les accès aux Web Services

Utilisons JSON to Kotlin



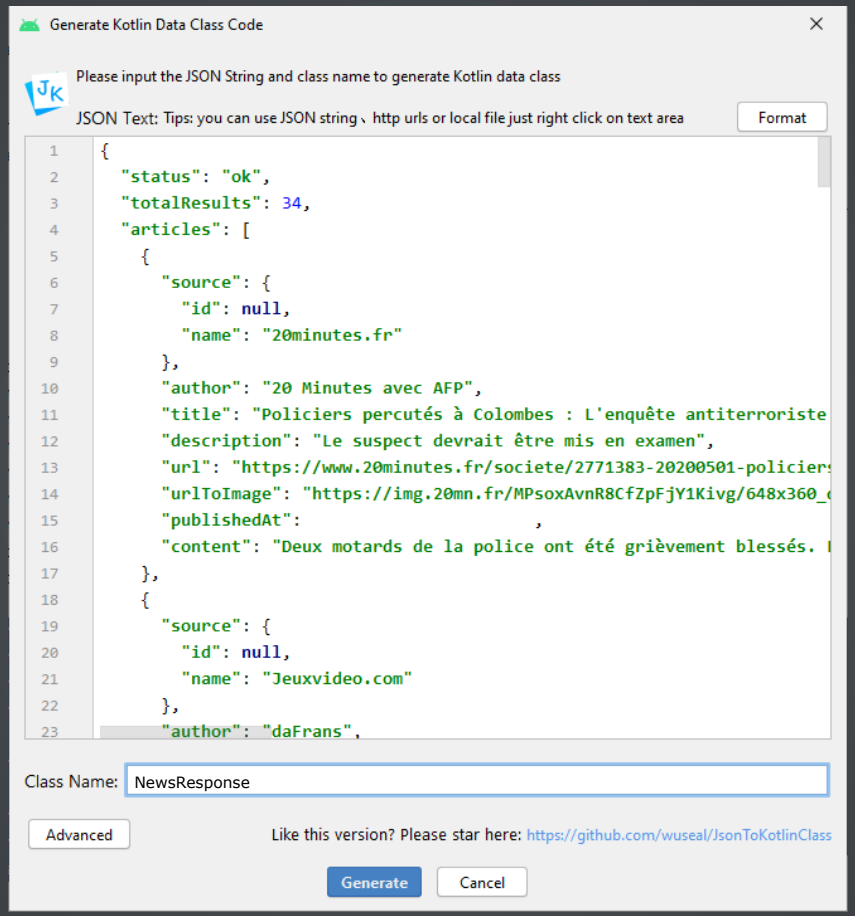
Les accès aux Web Services

Création du compte sur le site de news

Afin d'accéder à l'API de News, nous allons suivre les étapes suivantes :

- Créer un compte (gratuit) sur le site News API : https://newsapi.org/.
- Aller dans la section : "News sources" (tout en bas), puis France.
- Copier l'URL de l'API (par exemple : http://newsapi.org/v2/top-headlines?country=fr&apiKey=API_KEY)
- Récupérer le contenu du message (click droit, code source de la page => view-source:http://newsapi.org/v2/top-headlines?country=fr&apiKey=API_KEY (sur chrome).
- Générer les POJO (détail page suivante).

Les accès aux Web Services



Les accès aux Web Services

Paramétrage avancé

Il est possible de configurer le générateur avec le bouton "Advanced". Nous allons par exemple activer l'Annotation Gson.
Nous avons donc maintenant 3 nouvelles classes :

- NewsResponse.
- Article.
- Source.

Vérifions que les types des attributs sont bien des `String`.

Les accès aux Web Services

Écriture de l'interface avec l'API

Créons une nouvelle interface : NewsApi contenant le code suivant :

Copier

```
import retrofit2.Call
import retrofit2.http.GET
import retrofit2.http.Query

interface NewsApi {
    @GET("top-headlines")
    fun getNewsList(
        @Query("country") newsSource: String?,
        @Query("apiKey") apiKey: String?
    ): Call<NewsResponse?>?
}
```

Les accès aux Web Services

Creation de la classe de service

Copier

```
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class RetrofitService {
    private val retrofit = Retrofit.Builder()
        .baseUrl("https://newsapi.org/v2/")
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    fun <S> createService(serviceClass: Class<S>): S {
        return retrofit.create(serviceClass)
    }
}
```

Les accès aux Web Services

NewsRepository

Copier

```
1  import android.util.Log
2  import androidx.lifecycle.MutableLiveData
3  import retrofit2.Call
4  import retrofit2.Callback
5  import retrofit2.Response
6
7  object NewsRepository {
8      private const val TAG = "NewsRepository"
9
10     private var newsApi: NewsApi? = null
11
12     init {
13         newsApi = RetrofitService().createService(NewsApi::class.java)
14     }
15
16     fun getNews(country: String?, key: String?): MutableLiveData<NewsResponse?> {
17         val newsData = MutableLiveData<NewsResponse?>()
```

Les accès aux Web Services

NewsViewModel

Copier

```
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel

class NewsViewModel : ViewModel() {
    private var mutableLiveData: MutableLiveData<NewsResponse?>? = null
    fun init() {
        if (mutableLiveData != null) {
            return
        }
        mutableLiveData = NewsRepository.getNews("fr", "96c6792fdad6434fbc3a893daba40e0f")
    }

    fun getNewsRepository(): LiveData<NewsResponse?>? {
        return mutableLiveData
    }
}
```

Les accès aux Web Services

Ajout d'un fragment

Nous allons ajouter un fragment `NewsFragment` à notre projet, et dans le layout du fragment, ajouter une `RecyclerView` :

Copier

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             xmlns:tools="http://schemas.android.com/tools"
             android:layout_width="match_parent"
             android:layout_height="match_parent"
             tools:context=".NewsFragment">
    <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rvNews"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
</FrameLayout>
```

Les accès aux Web Services

Modification du layout

Nous allons ajouter ce fragment au layout de notre `activity_main.xml` :

Copier

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             xmlns:tools="http://schemas.android.com/tools"
             android:layout_width="match_parent"
             android:layout_height="match_parent"
             tools:context=".MainActivity">
    <fragment android:id="@+id/news_fragment"
              android:name="com.example.todeletenetwork.NewsFragment"
              android:layout_width="match_parent"
              android:layout_height="match_parent" />
</FrameLayout>
```

Les accès aux Web Services

Création du layout pour afficher une news

Nous allons par exemple écrire un layout : item_news.xml :

Copier

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
                                xmlns:app="http://schemas.android.com/apk/res-auto"
                                xmlns:tools="http://schemas.android.com/tools"
                                android:layout_width="match_parent"
                                android:layout_height="wrap_content"
                                android:layout_margin="5dp">

    <TextView
            android:id="@+id/item_news_textView_title"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:transitionName="title"
            app:layout_constraintEnd_toStartOf="@id/item_news_imageView_image"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            tools:text="Title title title title title title title title title title title title title title title title title title title title title title title title
```

Les accès aux Web Services

Création de l'adapter : NewsAdapter

Copier

```
import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import androidx.recyclerview.widget.RecyclerView.ViewHolder
import com.bumptech.glide.Glide

class NewsAdapter(val context: Context, val myDataset: List<Article>):
    RecyclerView.Adapter<NewsAdapter.MyViewHolder>() {
    private var mDataset: List<Article>? = null
    private var mInflater: LayoutInflater? = null

    init {
        mInflater = LayoutInflater.from(context)
```

Les accès aux Web Services

Appel dans notre fragment

Déclarons les variables de classes suivantes :

```
1 private var newsAdapter: NewsAdapter? = null
2 private val newsViewModel: NewsViewModel by viewModels()
3 private var rvHeadline: RecyclerView? = null
4 private var articleArrayList = arrayListOf<Article>()
```

Copier

Les accès aux Web Services

Appel dans notre fragment

Déclarons la fonction suivante :

Copier

```
1  private fun setupRecyclerView(context: Context) {
2      if (newsAdapter == null) {
3          newsAdapter = NewsAdapter(context, articleArrayList)
4          rvHeadline?.layoutManager = LinearLayoutManager(context)
5          rvHeadline?.adapter = newsAdapter
6          //rvHeadline?.setItemAnimator(DefaultItemAnimator())
7          //rvHeadline?.setNestedScrollingEnabled(true)
8      } else {
9          newsAdapter?.notifyDataSetChanged()
10     }
11 }
```

Les accès aux Web Services

Appel dans notre fragment

Remplaçons le corps de notre méthode onCreateView :

Copier

```
// Inflate the layout for this fragment
val rootLayout = inflater.inflate(R.layout.fragment_news, container, false)

context?.let { safeContext ->
    rvHeadline = rootLayout.findViewById(R.id.rvNews)
    newsViewModel.init()
    newsViewModel.getNewsRepository()?.observe(viewLifecycleOwner) {
        val newsArticles = it?.articles
        if (newsArticles != null) {
            articleArrayList.addAll(newsArticles)
            newsAdapter?.notifyDataSetChanged()
        }
    }
    setupRecyclerView(safeContext)
}
return rootLayout
```

Les accès aux Web Services

Erreurs possibles

Si vous avez l'erreur :

```
java.lang.NoSuchMethodError: No static method metafactory(Ljava/lang/invoke/MethodHandles
```

Copier

Il faut activer la compatibilité JVM 1.8 en ajoutant dans le build.gradle, le code suivant :

```
 android {
...
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = "1.8"
    }
}
```

Copier

Les accès aux Web Services

Erreurs possibles

Si vous avez l'erreur, sur l'émulateur :

```
1 java.net.SocketException: socket failed: EPERM (Operation not permitted)
```

Copier

Il suffit que nous désinstallions l'application, pour la réinstaller.

Les accès aux Web Services

Erreurs possibles

Si nous avons une SocketTimeoutException, il suffit d'allonger la durée du Timeout.
Dans la classe : RetrofitService remplaçons

```
private val retrofit = Retrofit.Builder()
    .baseUrl("https://newsapi.org/v2/")
    .addConverterFactory(GsonConverterFactory.create())
    .build()
```

Copier

Par :

```
var client = OkHttpClient.Builder()
    .connectTimeout(100, TimeUnit.SECONDS)
    .readTimeout(100, TimeUnit.SECONDS).build()
private val retrofit = Retrofit.Builder()
    .baseUrl("https://newsapi.org/v2/").client(client)
    .addConverterFactory(GsonConverterFactory.create(Gson()))
    .build()
```

Copier

Cela nous permet d'attendre un peu plus de temps avant d'avoir une erreur. Il faut dans ce cas-là, afficher une indication à l'utilisateur pour qu'il comprenne qu'il doit attendre.

Les accès aux Web Services

Erreurs possibles

Si nous avons l'erreur suivante :

```
1 Caused by: java.lang.IllegalArgumentException: Binary XML file line #11: Must specify unique android:id, android:tag, or have a parent with an id for
  com.example.todeletenetwork.NewsFragment
```

Copier

Il suffit de préciser un id au tag `fragment` dans le layout de notre activity.

Compléments

- Les capteurs. Les API Google de localisation.

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 rallumer 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 (Java).

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

Copier

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

Utilisation des capteurs

Liste des capteurs (Kotlin).

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

```kotlin
val sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
var liste = sensorManager.getSensorList(Sensor.TYPE_ALL)
if (BuildConfig.DEBUG) {
    for (currentSensor in liste) {
        Log.d(TAG, "onCreate sensor $currentSensor")
    }
}
```

Copier

Utilisation des capteurs

Récupérons le capteur de proximité (Java).

```
1 private Sensor mProximitySensor = null; // En attribut de classe.
2 mProximitySensor = sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
```

Copier

Utilisation des capteurs

Récupérons le capteur de proximité (Kotlin)

```kotlin
var mProximitySensor:Sensor? = null // En attribut de classe.
mProximitySensor = sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY)
```

Copier

Utilisation des capteurs

Enregistrons-nous, sur les changements (Java).

Dans un premier temps, nous récupérons le manager des capteurs :

```
1 private SensorManager mSensorManager = null; // En attribut de classe.
2 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

Copier

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 }
```

Copier

Utilisation des capteurs

Le listener (Java).

Le listener qui va réagir aux changements est définis par :

```
final SensorEventListener mSensorEventListener = new SensorEventListener() {
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // Que faire en cas de changement de précision ?
    }

    public void onSensorChanged(SensorEvent sensorEvent) {
        if(BuildConfig.DEBUG){
            Log.d(TAG, "onSensorChanged " + sensorEvent.values.length);
            Log.d(TAG, "onSensorChanged " + sensorEvent.values[0]);
        }
    }
};
```

Copier

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

Enregistrons-nous, sur les changements (Kotlin)

Dans un premier temps, nous récupérons le manager des capteurs :

```
var mSensorManager: SensorManager? = null // En attribut de classe.
mSensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
```

Copier

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

```
override fun onResume() {
    super.onResume()
    mSensorManager?.registerListener(
        mSensorEventListener,
        mProximitySensor,
        SensorManager.SENSOR_DELAY_NORMAL
    )
}

override fun onPause() {
    super.onPause()
    mSensorManager?.unregisterListener(mSensorEventListener, mProximitySensor)
}
```

Copier

Utilisation des capteurs

Le listener (Kotlin)

Le listener qui va réagir aux changements est définis par :

Copier

```kotlin
val mSensorEventListener: SensorEventListener = object : SensorEventListener {
    override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) {
        // Que faire en cas de changement de précision ?
    }

    override fun onSensorChanged(sensorEvent: SensorEvent) {
        if (BuildConfig.DEBUG) {
            Log.d(TAG, "onSensorChanged ${sensorEvent.values.size}")
            Log.d(TAG, "onSensorChanged ${sensorEvent.values[0]}")
        }
    }
}
```

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

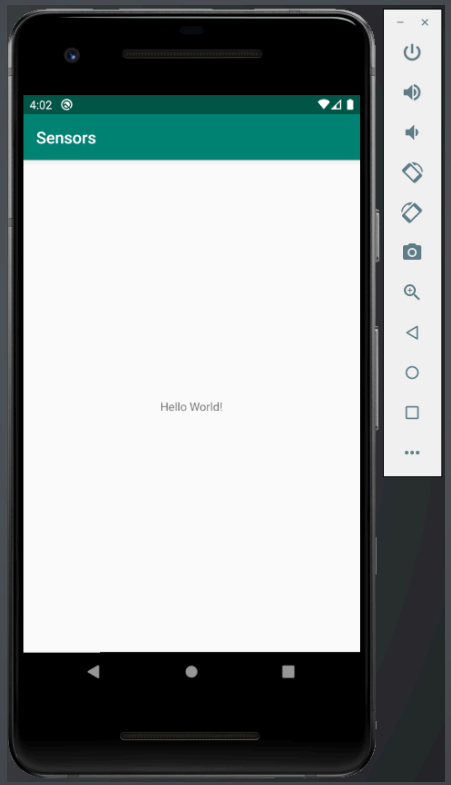
Utilisation des capteurs

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 a vu ici.
Un exemple simple pour afficher les valeurs de l'accéléromètre ici.

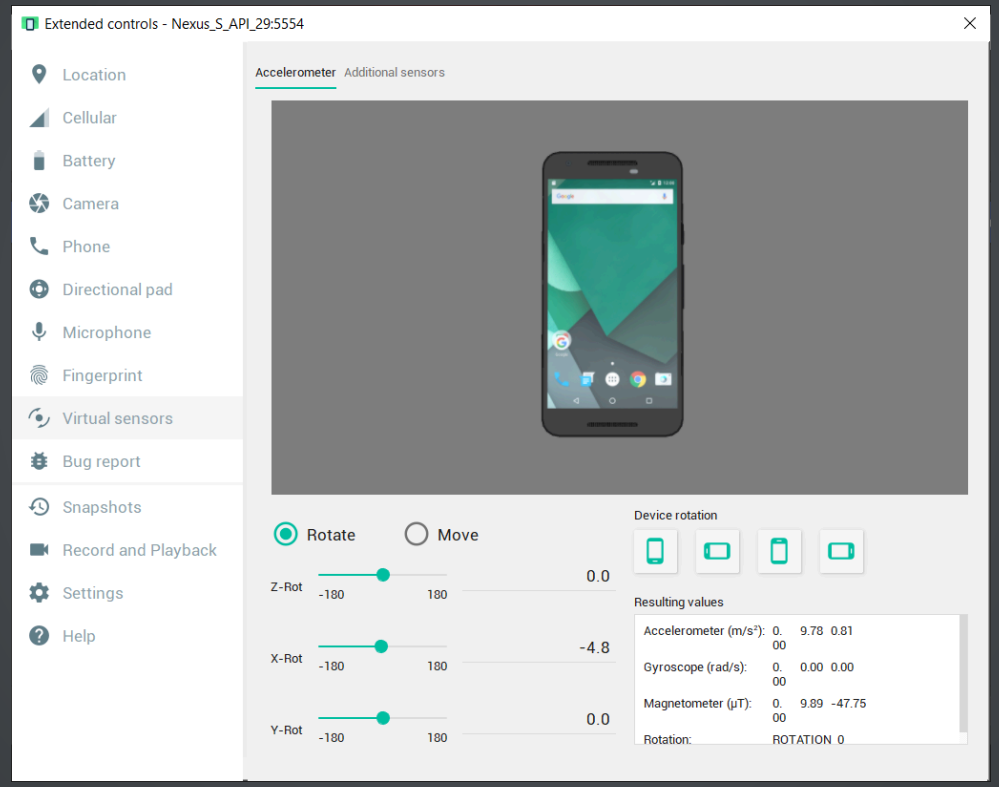
Utilisation des capteurs

Paramétrage dans le simulateur des capteurs.



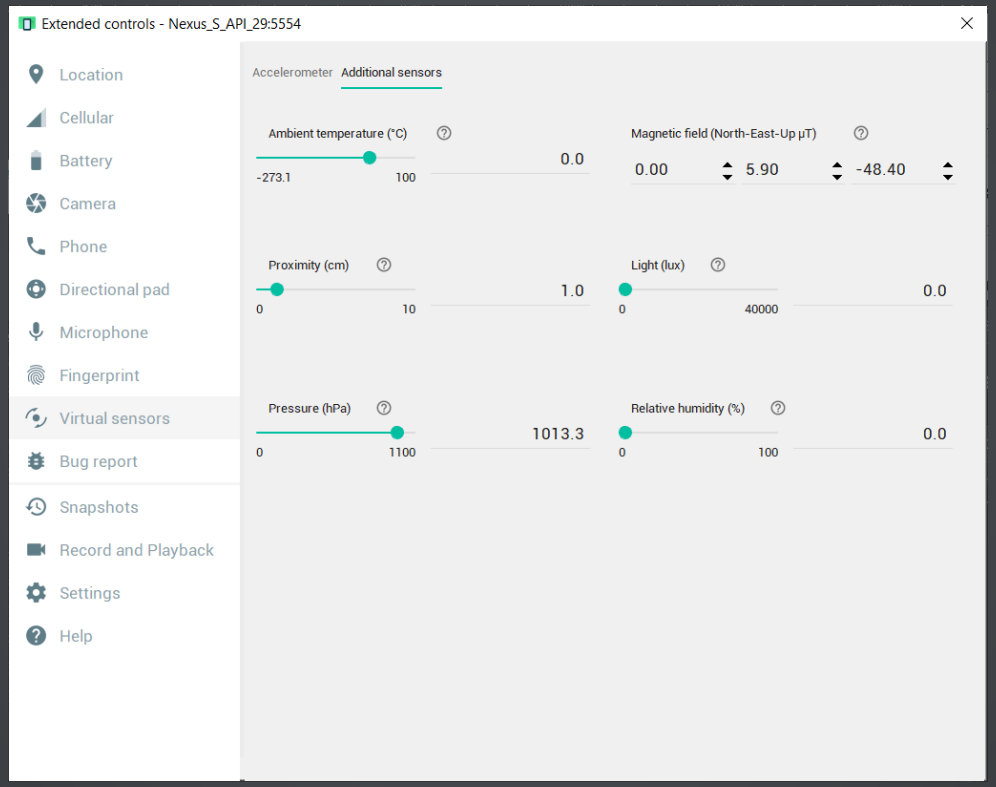
Utilisation des capteurs

Paramétrage dans le simulateur des capteurs.



Utilisation des capteurs

Paramétrage dans le simulateur des capteurs.



Send SMS

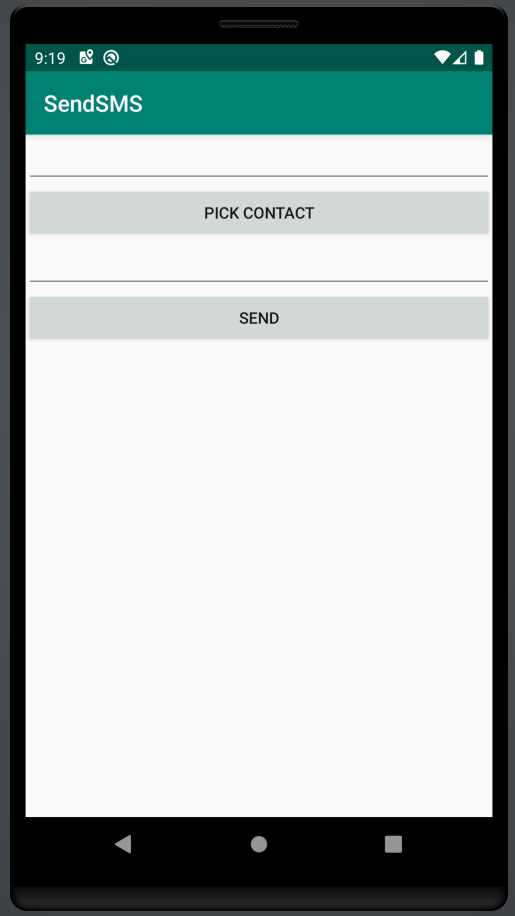
Présentation

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.

GUI

Send SMS

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 : `and_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;
```

Copier

Récupérons leur valeur :

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

Copier

Send SMS

Implémentation du code

Implémentons le code pour envoyer les SMS : utilisons, hors d'une méthode, le Live Template : `and_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());
```

Copier

Send SMS

Permissions

Utilisons le Live Template `and_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.
```

Copier

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

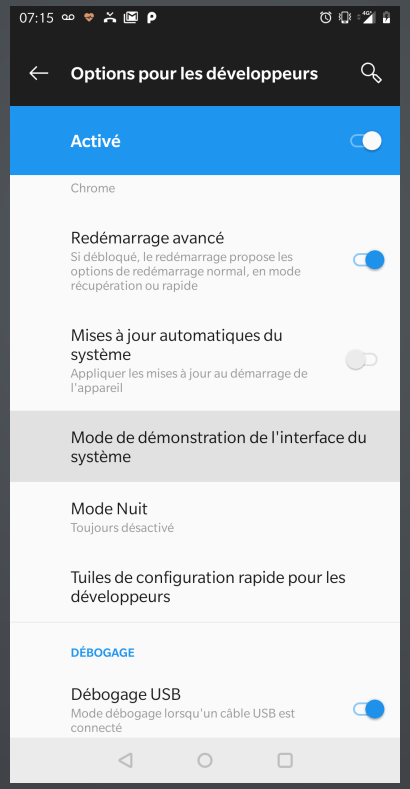
Raccourcis claviers utiles

Touches	Action
Shift deux fois rapidement.	Ouvrir la recherche.
F2	Aller jusqu'à la prochaine erreur.
Ctrl + Y	Effacer une ligne.
Shift + F6	Renommer.
Shift + Ctrl + Flèche haut/bas	Déplacer une ligne de code.
Ctrl + Alt + L	Indenter le code.
Alt + F7	Find usage.
Ctrl + Alt + O	Organiser les imports.
Ctrl (maintenu) + click souris	Ouvrir la source cliquée.
Ctrl + D	Dupliquer la ligne.

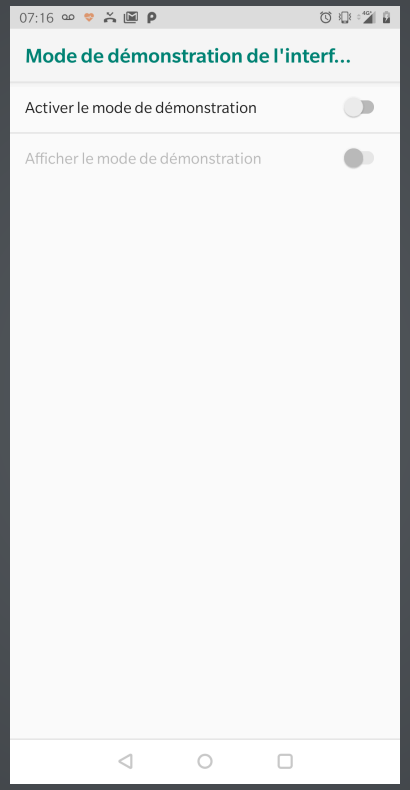
Annexes

Le mode démo

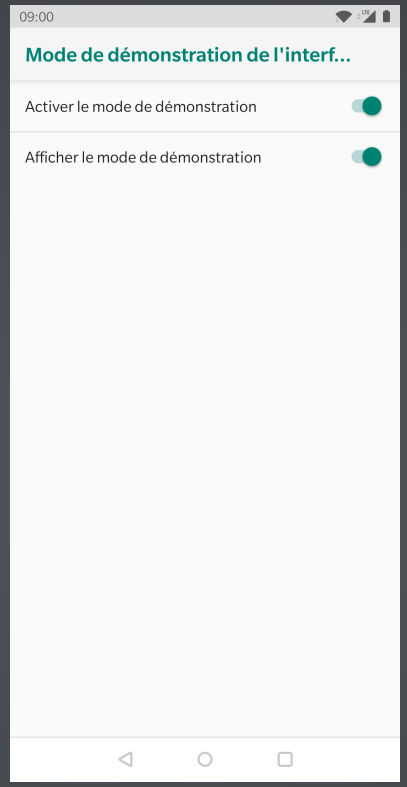
Sur le téléphone, allons dans le menu développeur, Mode de démonstration de l'interface du système et activons le mode démonstration et affichons-le :



Annexes


07:16
Mode de démonstration de l'interf...
Activer le mode de démonstration
Afficher le mode de démonstration

Annexes



Quels changements observons nous ?

Annexes

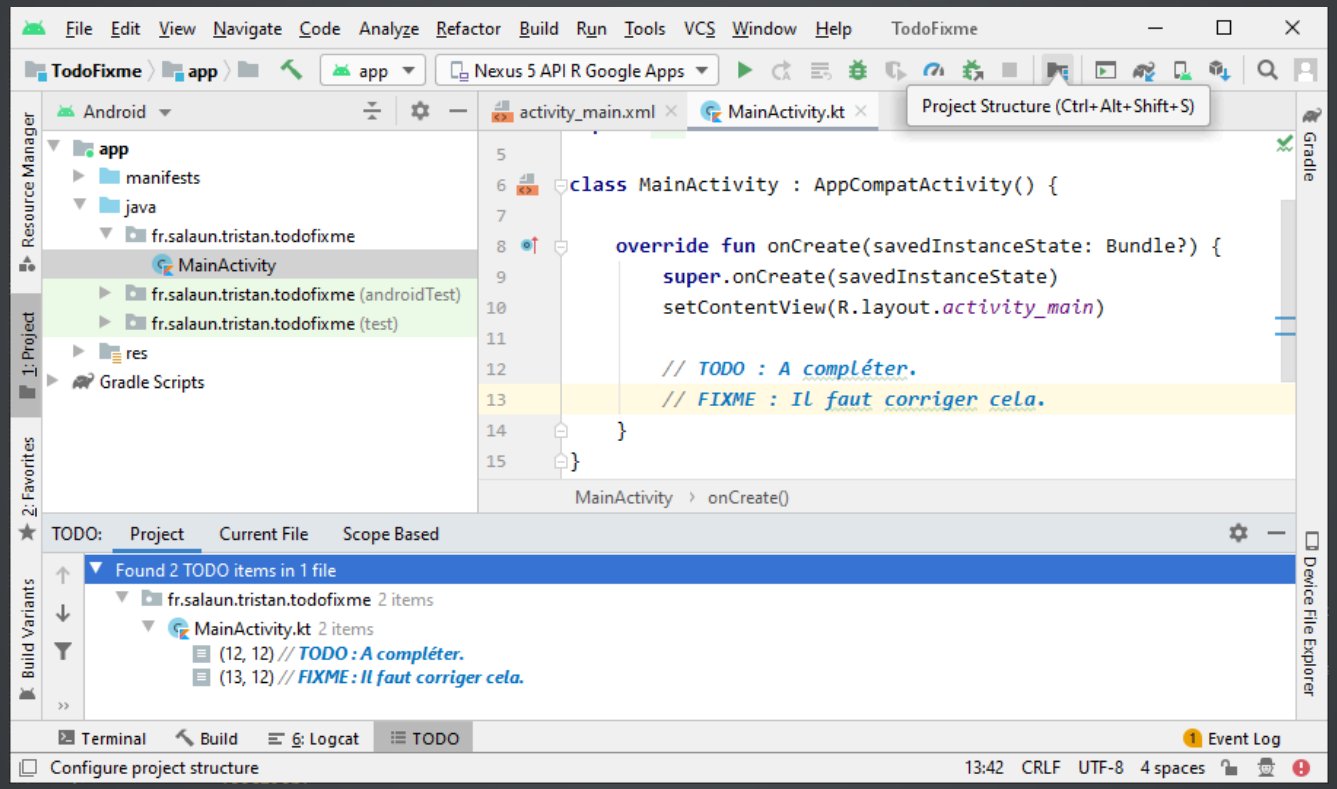
LivesTemplates

Nous avons à notre disposition plusieurs LiveTemplates, tels que :

- `and_tts` : pour activer une synthèse vocale.
- `and_intent_voice_recognition` : pour activer une reconnaissance vocale.

TODO / FIXME

En commentaires, les mots clés TODO et FIXME sont interprétés par l'éditeur :



Annexes

Universal ADB Drivers

Un outil indispensable sous Windows pour installer les drivers manquants pour que le système reconnaisse son téléphone/tablette :
Universal ADB Drivers

Annexes

Les gestures

Nous pouvons mettre en place la reconnaissance de gesture (motifs) définies par le développeur.

- Projet gesture.

Annexes

Le swipe

Nous pouvons aussi réagir à des mouvements simples : le swipe (haut, bas, droite et gauche).

- OnSwipe.

Annexes

Références utiles

- Liste de nombreuses librairies.
- JetPack.
- Liste de cours en Kotlin intéressants.
- GitLab.

Annexes

Jetpack Compose

- Jetpack Compose
- Jetpack Compose tutorial

Annexes

La passerelle Web et JS

Nous pouvons mettre en place une passerelle JS pour appeler du code natif Android depuis une page Web affichée dans une webview :

- Tutoriel ici.

Annexes

Contact Picker

Exemple de sélection d'un contact dans notre répertoire téléphonique.
Dans le clickListener d'un bouton, nous allons utiliser le LiveTemplate suivant :
`and_intent_select_contact`, et suivre les instructions.

Annexes

Scripts batch/bash

Voyons ensemble sur les prochaines diapositives des scripts qui vont nous aider à développer/tester/installer plus rapidement nos applications.

`clear.bat`, pour effacer les données d'une application :

```
1 echo Waiting for device on USB
2 adb wait-for-usb-device
3 echo Device found
4
5 echo Clear APP Tristan
6 adb shell pm clear fr.salaun.tristan.android.myapplication
```

Copier

`delete.bat`, pour effacer une application :

```
1 echo Waiting for device on USB
2 adb wait-for-usb-device
3 echo Device found
4
5 echo "Delete APP Tristan"
6 adb uninstall fr.salaun.tristan.android.myapplication
```

Copier

Annexes

Scripts batch/bash

`install.sh`, pour installer plusieurs applications d'un répertoire :

Copier

```
#!/bin/bash

# example : ./install_all_apk.sh /c/apk/

echo Waiting for device on USB
adb wait-for-usb-device
echo Device found

echo "The path to use : $1"

for filename in $1/*.apk; do
   echo "installing $filename"
   adb install -r -d -g $filename
done
```

Annexes

Scripts batch/bash

`install_apk_all_devices.bat`, pour installer une application sur plusieurs devices :

Copier

```
@echo off

SETLOCAL ENABLEDELAYEDEXPANSION
:: INSTALL ON ALL ATTACHED DEVICES ::
FOR /F "tokens=1,2 skip=1" %%A IN ('adb devices') DO (
  start "Sub" install_apk_all_devices_installer.bat %%A
)
ENDLOCAL

:EOF
```

Annexes

Scripts batch/bash

`install_apk_all_devices_installer.bat`, qui est appelé par le script précédent :

Copier

```
@echo off
SET ARGUMENTS=%~1

if "%ARGUMENTS%" == "" (
    GOTO EOF
)

adb -s %ARGUMENTS% uninstall fr.salaun.tristan.android.myapplication

echo Waking up the device %ARGUMENTS%.
adb -s %ARGUMENTS% shell input keyevent KEYCODE_WAKEUP

REM Sleep for 2 seconds
echo Wait a bit
ping -n 5 127.0.0.1>nul

echo Unlock the screen
```

Utilisation du JSON avec GSON.

Annexes

Créons le projet suivant : `com.example.jsonmyapplication`.
Ajoutons la dépendance dans le `build.graddle` (app) :

```
implementation 'com.google.code.gson:gson:2.8.6'
```

Copier

Puis créons les classes suivantes :

```
data class Book(var id: String, var name: String, var author: String, var genre: String, var numpages: Int, var releaseDate: String, var cover: String)
```

Copier

```
data class BookList (val bookList: List<Book>)
```

Copier

Déclarons une liste de livres :

```
var list = arrayListOf(
Book(
    id = "01fsEF", name = "1984",
    author = "George Orwell",
    genre = "Fiction dystopique",
    numpages = 376,
    releaseDate = "1949",
    cover = "1984.png"
),
Book(
    id = "3H1J0n",
    name = "Le Meilleur des mondes",
    author = "Aldous Huxley",
    genre = "Science-fiction",
    numpages = 285,
    releaseDate = "1932",
    cover = "meilleur-des-mondes.jpg"
```

Copier

Annexes

Utilisation du JSON avec GSON (suite).

Définissons une méthode utilitaire pour formater le JSON :

Copier

```
fun jsonToPrettyFormat(jsonString: String?): String? {
    val json = JsonParser.parseString(jsonString).asJsonObject
    val gson = GsonBuilder()
        .serializeNulls()
        .disableHtmlEscaping()
        .setPrettyPrinting()
        .create()
    return gson.toJson(json)
}
```

Annexes

Utilisation du JSON avec GSON (suite et fin).

Mettons en place la "Serialization" :

```
// Serialization
val gson = Gson()
val listType: Type = object : TypeToken<BookList?>() {}.type
val jsonResult: String = gson.toJson(bookList, listType)
Log.d(TAG, "onCreate jsonResult = $jsonResult")

Log.d(TAG, "onCreate jsonResult = ${jsonToPrettyFormat(jsonResult)}")
```

Copier

Et enfin testons la "Déserialization" :

```
// Deserialization
val bookList2: BookList = Gson().fromJson(jsonResult, listType)
Log.d(TAG, "onCreate bookList2 = ${bookList2}")
```

Copier

Annexes

Utiliser le presse-papier

Créons un projet simple (en Java ou Kotlin), donnons un id au TextView déjà présent, et dans le code de l'Activity principale :
En Java :

```java
TextView tvDemo = (TextView) findViewById(R.id.textview);
tvDemo.setOnLongClickListener(new View.OnLongClickListener() {
  @Override
  public boolean onLongClick(View v) {
    copyContentToClipboard("content", ((TextView)v).getText().toString());
    return true;
  }
});
```

Copier

En Kotlin :

```kotlin
findViewById<TextView>(R.id.textview).setOnLongClickListener(View.OnLongClickListener {
    copyContentToClipboard("label", (it as TextView).text.toString())
    return@OnLongClickListener true
})
```

Copier

Utilisons le Live Template and_copy_to_clipboard pour écrire la méthode : `copyContentToClipboard`.

Annexes

MVVM

Live data, ViewModel, Retrofit Android Architecture Component

- Live data, ViewModel, Retrofit Android Architecture Component.
- MVVM (Model View ViewModel) + Kotlin + Google Jetpack.

Annexes

RecyclerView

Gestion du click en Kotlin.

Annexes

Sources

- Sources

Annexes

handleUncaughtException

- Sources

Annexes

Timber

- Timber

Annexes

Décompiler un APK

Récupération de l'APK en utilisant par exemple : ES Explorateur de Fichiers ou le backup android.

- JADX
- Online
- Online
- WiKi
- APK Studio

Annexes

Koin

Injection de dépendances faciles :

- Introduction à l'injection.
- Kotlin Koin - Android Tutorial for Beginners - Step By Step Guide.
- Dependency Injection With Koin.
- Projet d'exemple.
- Aller plus loin.

Annexes

run-as

Récupération de fichiers dans une application débuggable :
Le plus simple étant d'utiliser le `View / Tool Windows / Device File Explorer`.
Un seul fichier

```
adb exec-out run-as debuggable.app.package.name cat databases/file > file
```

Copier

Plusieurs fichiers

```
adb exec-out run-as debuggable.app.package.name tar c databases/ > databases.tar
adb exec-out run-as debuggable.app.package.name tar c shared_prefs/ > shared_prefs.tar
```

Copier

A tester :

```
> adb shell
shell $ run-as com.example.package
shell $ chmod 666 databases/file
shell $ exit                                          ## exit out of 'run-as'
shell $ cp /data/data/package.name/databases/file /sdcard/
shell $ run-as com.example.package
shell $ chmod 600 databases/file
> adb pull /sdcard/file .
```

Copier

Annexes

Backup

Backup complet :

```
adb backup -apk -shared -all -f <filepath>/backup.ab
```

Copier

Restauration :

```
adb restore <filepath>/backup.ab
```

Copier

Backup d'une seule application (avec APK -apk) :

```
adb backup -f ""“D:\myfolder\myapp.ab" -apk <package name>
```

Copier

Multiples exemple.

Pour extraire facilement les fichiers d'un backup, utiliser : adbextractor.

```
java -jar "C:\ADB\android-backup-tookit\android-backup-extractor\android-backup-extractor-20180203-bin\abe.jar" unpack c:\adb\backup2.ab backup-extracted.tar  (You'll be asked to enter the password)
```

Copier

Annexe

Mode développeur

Pour mettre son téléphone Android en mode développeur :

- Allons dans les paramètres.
- À propos du téléphone.
- Cliquons plusieurs fois sur le numéro de build.
- Une fois débloqué, nous pouvons activer le mode débugger, et la connexion USB.

Si au lieu de cliquer sur la version du build, nous cliquons sur la version d'Android, une surprise nous attend :-)

Annexe

Erreur classique

Une erreur : `Unsupported major.minor version 52.0` nous indique, que nous avons compilé le code avec une version de JDK 1.8 et que nous essayons de le lancer avec une version antérieure :

Références

Les références

- La référence : https://kotlinlang.org/docs/reference
- La source : https://github.com/JetBrains/kotlin
- Autre ressource très complète en FR : https://riptutorial.com/Download/kotlin-fr.pdf
- Rappels des bases : https://kotlinlang.org/docs/reference/basic-syntax.html

Références

Cours en ligne

- Sur Openclassrooms : https://openclassrooms.com/fr/courses/5353106-initiez-vous-a-kotlin
- Sur CodinGame : https://www.codingame.com/playgrounds/28826/formation-kotlin/les-bases-de-kotlin
- Kotlin Koans : https://play.kotlinlang.org/koans/overview

Références

Tests/Challenges

- https://github.com/Kotlin/kotlin-koans-edu
- https://tech.io/explore
- https://github.com/SK1dev/KotlinChallenges/blob/master/README.md
- https://github.com/SK1dev/KotlinChallengesPart2
- https://github.com/igorwojda/kotlin-coding-puzzle

Références

Articles sur des points précis

- https://typealias.com/
- http://zetcode.com/all/
- Pourquoi adopter Kotlin : https://medium.com/videdressing-engineering/pourquoi-je-suis-pass%C3%A9-%C3%A0-kotlin-7d40d79054a4

Références

Livres

- https://kotlinlang.org/docs/books.html

Références

Liste des mots clés

- https://kotlinlang.org/docs/reference/keyword-reference.html

Références

Kotlin pour les applications Backend

- https://soat.developpez.com/tutoriels/kotlin-application-backend/

Références

Kotlin pour Android

- https://www.udacity.com/course/ud9012
- https://blog.ippon.fr/2017/12/11/introduction-a-kotlin-pour-android/
- https://kotlinlang.org/docs/reference/android-overview.html
- https://developer.android.com/kotlin
- https://codelabs.developers.google.com/codelabs/android-room-with-a-view-kotlin/#0
- https://codelabs.developers.google.com/codelabs/kotlin-coroutines/#0