

GUIDE

ÉTAPE 1 : CONFIGURATION DU PROJET ET PREMIÈRE INTERFACE

Contenu

1. Préparation de l'environnement	1
2. Création d'une interface simple.....	13

1. Préparation de l'environnement

Pour travailler avec Android, il est nécessaire d'installer plusieurs outils. Vous n'allez bien sûr pas coder directement sur un téléphone ! D'ailleurs, peut-être que vous ne possédez même pas un téléphone de type Android. Ce n'est pas grave du tout. Les outils que vous allez utiliser vont vous permettre de simuler sur l'ordinateur un affichage de type téléphone Android et donc de tester votre application. Si vous avez un téléphone Android, vous pouvez aussi le brancher via USB sur votre ordinateur et, lorsque vous lancerez une exécution, vous pourrez choisir de voir le résultat directement sur le smartphone, en mode test. Vous allez aussi voir la démarche pour déployer et donc réellement installer votre application sur un téléphone.

1A. Installation de JDK

Le développement sous Android se fait en Java ou plus récemment, aussi en Kotlin. Ce guide a été fait en Java. Vous avez donc besoin que JDK soit installé. Si c'est déjà le cas sur votre ordinateur (si vous avez déjà programmé en Java), deux possibilités : soit vous gardez le jdk existant (si la version est au moins égale à 8) et vous passez à la suite, soit vous désinstallez la version existante pour installer la dernière version en allant ici :

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Attention, prenez bien la version qui correspond à votre système (et 64 bits si votre système est en 64 bits). Ne prenez pas la version compressée (zip) mais l'exécutable. La version qui a été utilisée pour ce guide est la 16.0.1.

1B. Installation de l'IDE Android Studio

Il existe plusieurs configurations possibles pour programmer sous Android. Par exemple, l'IDE Eclipse peut être paramétré avec le SDK d'Android.

Depuis plusieurs années, l'IDE Android Studio est l'IDE officiel pour Android.

Pour le récupérer, allez sur ce lien :

<https://developer.android.com/studio/index.html>

Ce guide a été réalisé avec Android Studio version 4.2. Si vous utilisez une version plus récente, il est possible (certain) qu'il y ait quelques différences.

Une fois le fichier téléchargé, lancez l'installation en mode admin. Lors de l'installation, gardez les options par défaut (sauf à la fin, décochez la case qui permet de lancer directement Android Studio). Vous risquez d'obtenir un message d'avertissement si vous n'avez pas un processeur Intel. Ce n'est pas grave.

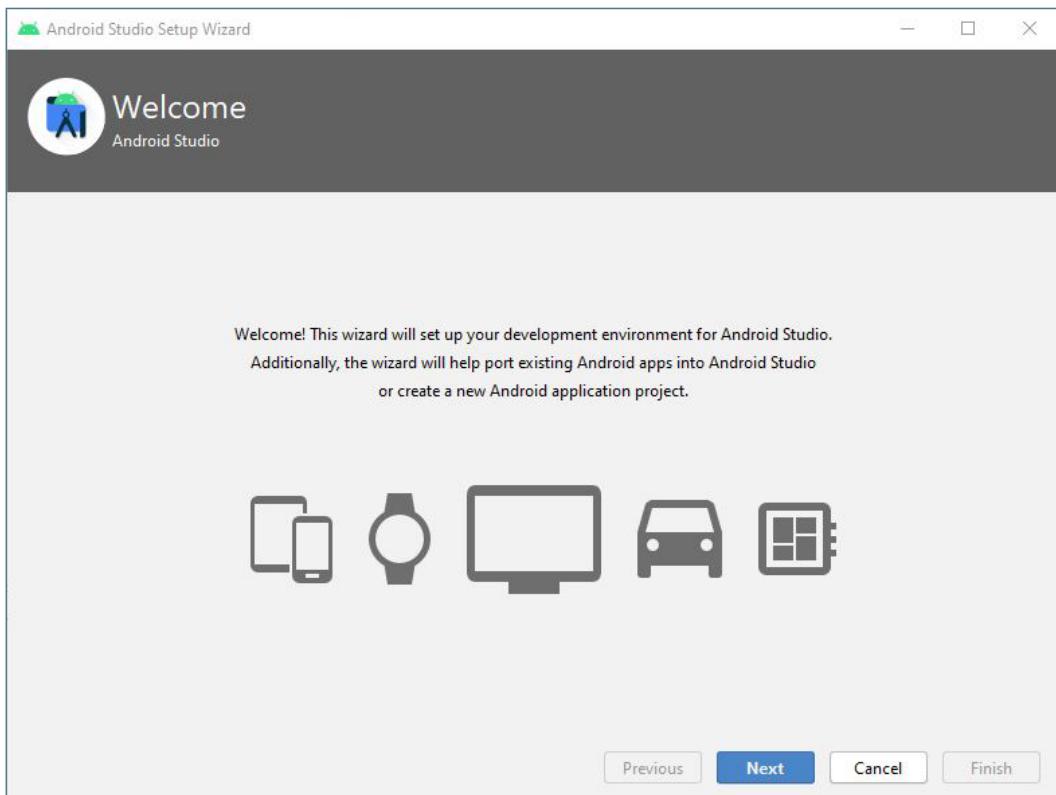


Important

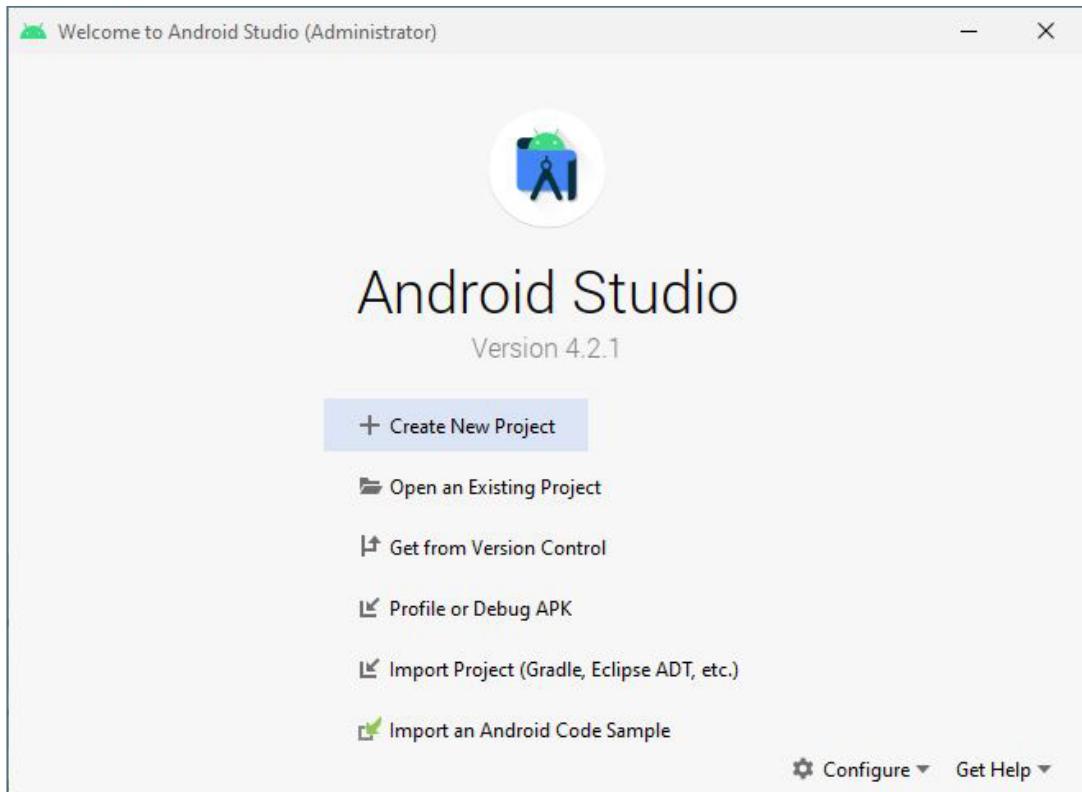
Par la suite, à chaque fois que vous ouvrirez Android Studio, faites-le toujours en mode admin (clic droit, exécuter en tant qu'administrateur)

1C. Configuration avec Android SDK Manager

Lancez Android Studio (mode admin). Lors du premier démarrage, vous allez tomber sur une fenêtre de bienvenue :



Suivez les étapes en gardant les options par défaut : des composants vont s'installer. Cliquez alors sur Finish. Vous allez enfin tomber sur la fenêtre officielle de démarrage.



Sur la fenêtre de démarrage, choisissez, en bas à droite "Configure" puis "SDK Manager".

Il faut maintenant configurer le SDK pour installer les packs nécessaires au type de développement choisi (plus précisément, à la version du matériel pour lequel on veut faire le développement).

Ce choix n'est d'ailleurs pas évident : il y a normalement une compatibilité ascendante (une ancienne version de SDK marche sur tous les appareils qui acceptent cette version et les versions plus récentes), cependant une ancienne version aura moins de possibilités qu'une version plus récente. Il faut donc trouver le compromis entre ce que l'on veut programmer et le nombre d'appareils que l'on cible pour le déploiement. Dans tous les cas, il faut tout de même éviter de prendre une version antérieure à l'API 15.

Pour notre application, l'API 16 est suffisante.

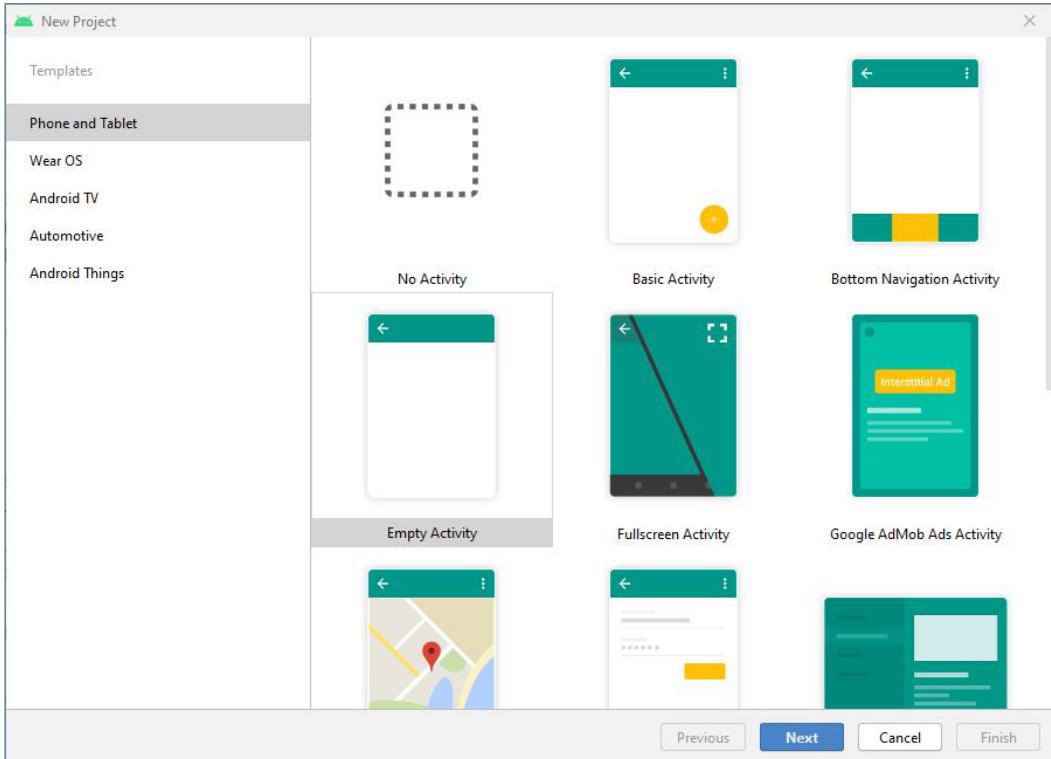
Dans la fenêtre donnant la liste des API, commencez par cocher en bas à droite "Show Package Details". Cocher tout le contenu de l'API 16 (Android 4.1 Jelly Bean) suffisante pour notre application, et laissez ce qui est déjà coché.

Lancez l'installation (pensez à bien cliquer sur "accept" : si dans la colonne de gauche, il y a plusieurs grandes parties : il faut faire accept après avoir sélectionné chacune d'elles. Puis cliquez sur Next. L'installation est assez longue.

1D. Création d'un projet

Une fois la configuration terminée, de retour dans la fenêtre de démarrage, cliquez sur "Create New Project".

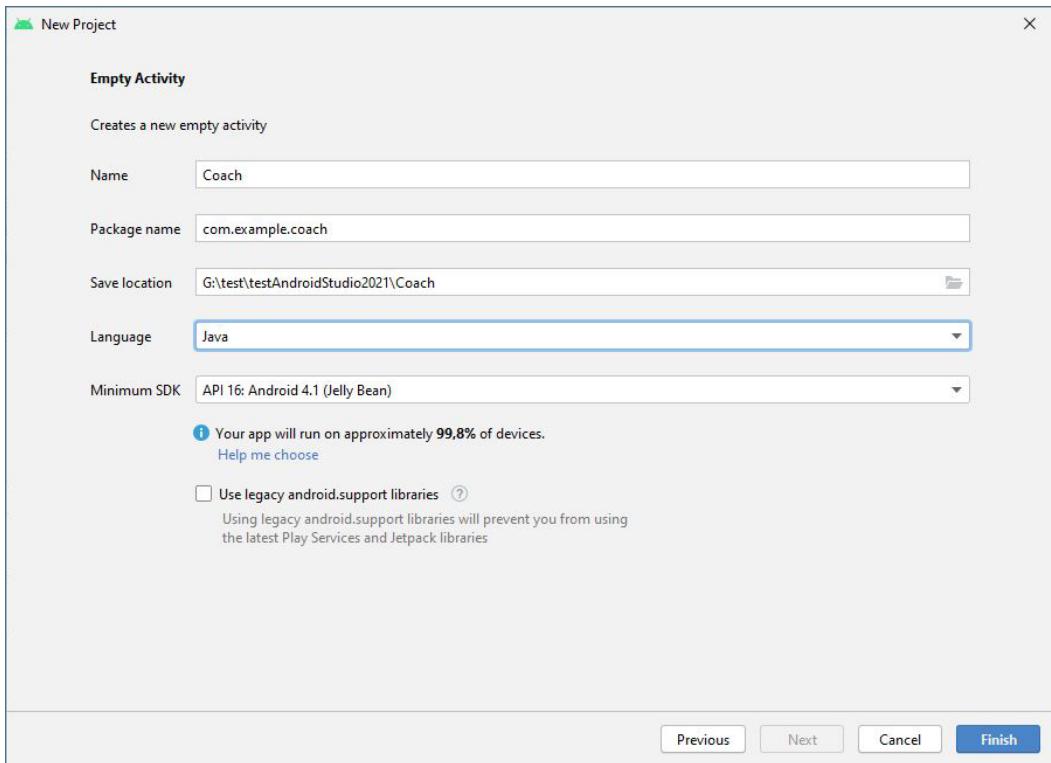
Dans la nouvelle fenêtre, vous allez faire le choix du type de projet et plus précisément de la présentation de la première interface de l'application (on parle "d'activity"). Remarquez, à gauche, les différents templates : ils permettent de choisir le type de support pour l'application. Restez sur le premier choix par défaut "Phone et Tablet" et normalement, "Empty Activity" est sélectionné par défaut :



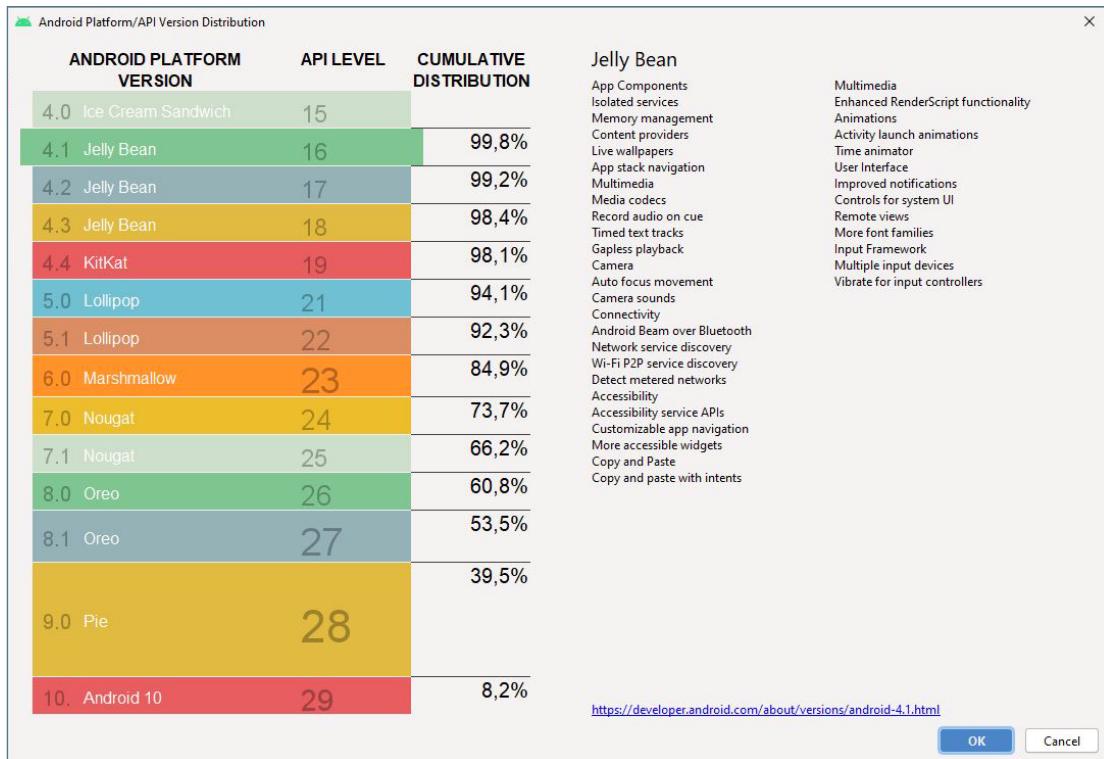
Gardez "Empty Activity" et cliquez sur Next.

Dans la nouvelle fenêtre, vous devez donner le nom de l'application (tapez "Coach"), normalement le "package name" s'est automatiquement mis à jour. En ce qui concerne le dossier de sauvegarde (Save Location), soit vous gardez le dossier par défaut (remarquez que le chemin se termine par "\Coach"), soit vous choisissez un dossier de travail (que vous aurez créé au préalable) et dans ce cas pensez à ajouter à la fin du chemin "\Coach". Au niveau "Language", sélectionnez bien "Java" (surtout ne sélectionnez pas Kotlin : ce guide a été fait en Java et non en Kotlin qui est le nouveau langage Android Studio).

Remarquez le minimum SDK : normalement c'est le "API 16" qui est sélectionné. Si vous regardez la liste, c'est celui qui est de plus bas niveau. Le SDK permet de déterminer sur quelles versions de mobiles l'application va pouvoir tourner. L'idée est de choisir la version la plus ancienne (pour que l'application fonctionne sur le plus grand nombre de mobiles) et en même temps, la version adaptée aux besoins de l'application. En effet, suivant les outils nécessaires, vous ne pourrez pas utiliser une version trop ancienne. Pour l'application que nous allons créer, l'API 16 est suffisante.



Pour mieux comprendre le principe des API, ne cliquez pas tout de suite sur Finish, mais sur "Help me choose".

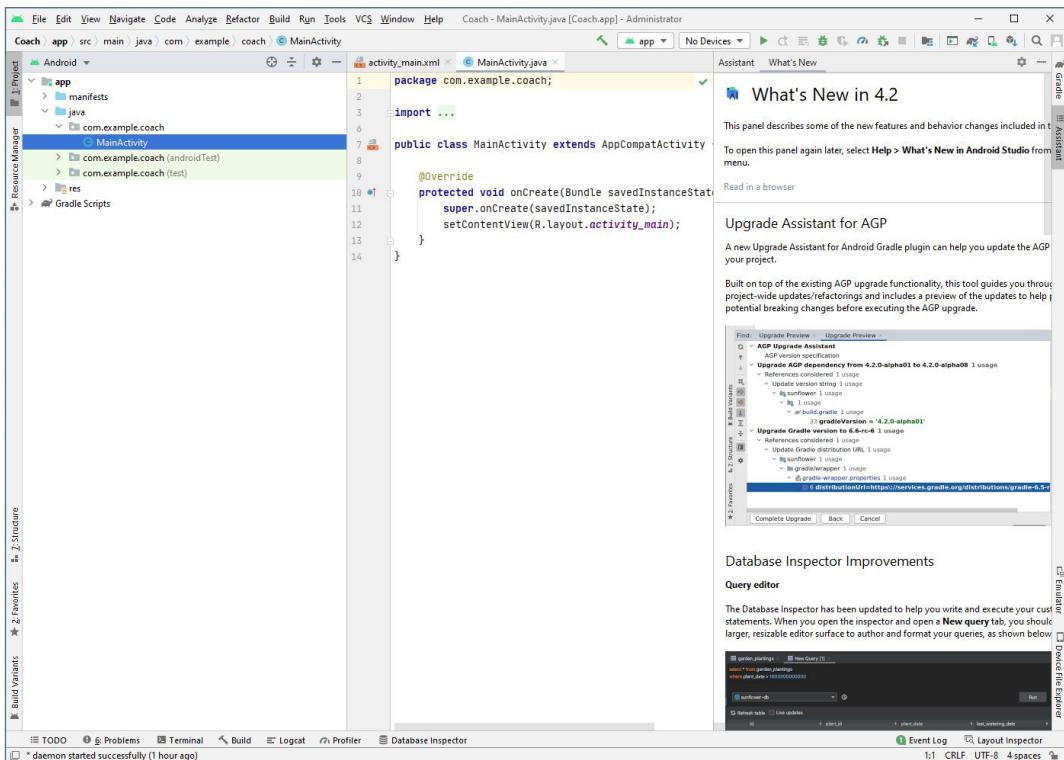


Cette fenêtre présente les différentes versions : en cliquant sur chaque version, vous pourrez voir à droite le détail des fonctionnalités accessibles. Remarquez que l'API 16 couvre 99,8 % des supports.

Cliquez ensuite sur Cancel pour revenir à la fenêtre précédente. Contrôlez que c'est bien l'API 16 qui est sélectionnée et cliquez sur Finish.

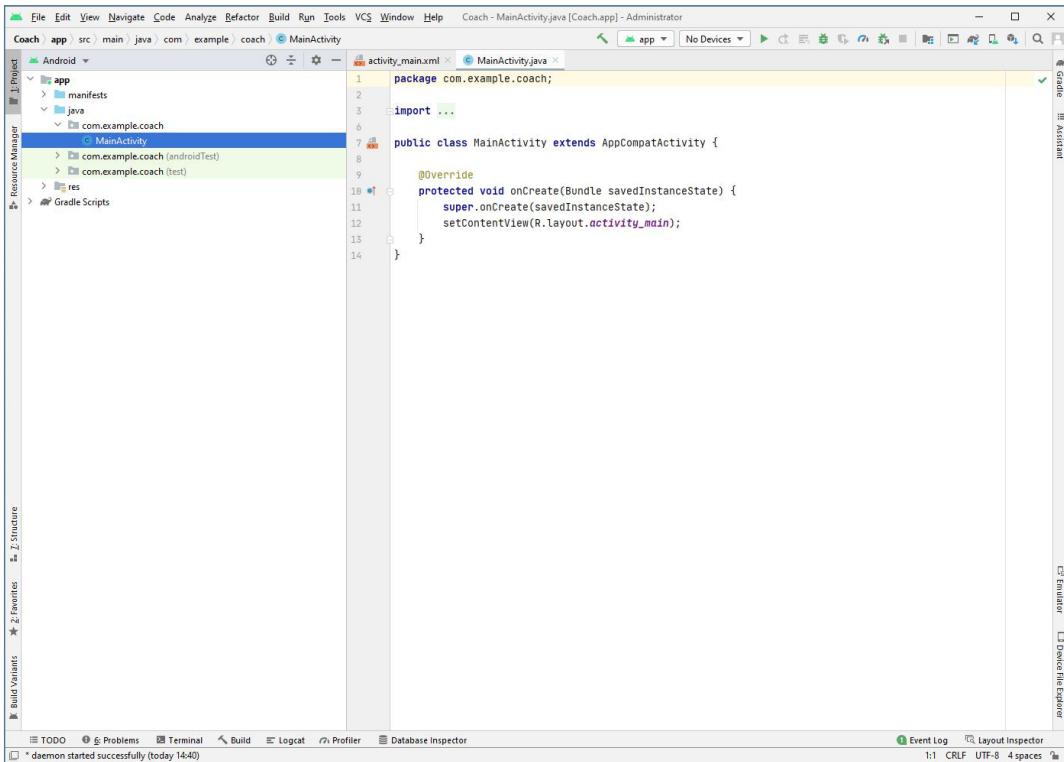
Si le pare-feu s'affole, autorisez l'accès. Le chargement du projet est assez long (observez le petit cercle tournant pour patienter, en bas à droite et ensuite une petite barre de progression). Attendez la fin complète du chargement.

En fin de téléchargement, on obtient une fenêtre théoriquement semblable à ceci :



Suivant la version d'Android Studio utilisée, il est possible que vous obteniez une fenêtre contenant, dans la partie basse, une erreur et une invitation à faire une installation complémentaire en cliquant sur un lien. Dans ce cas, suivez les indications et cliquez sur le lien. Si, après cette installation, une nouvelle erreur apparaît et une nouvelle installation est proposée, suivez encore les indications. Faites-le tant que des propositions sont faites. Au final, vous devriez obtenir la fenêtre précédente.

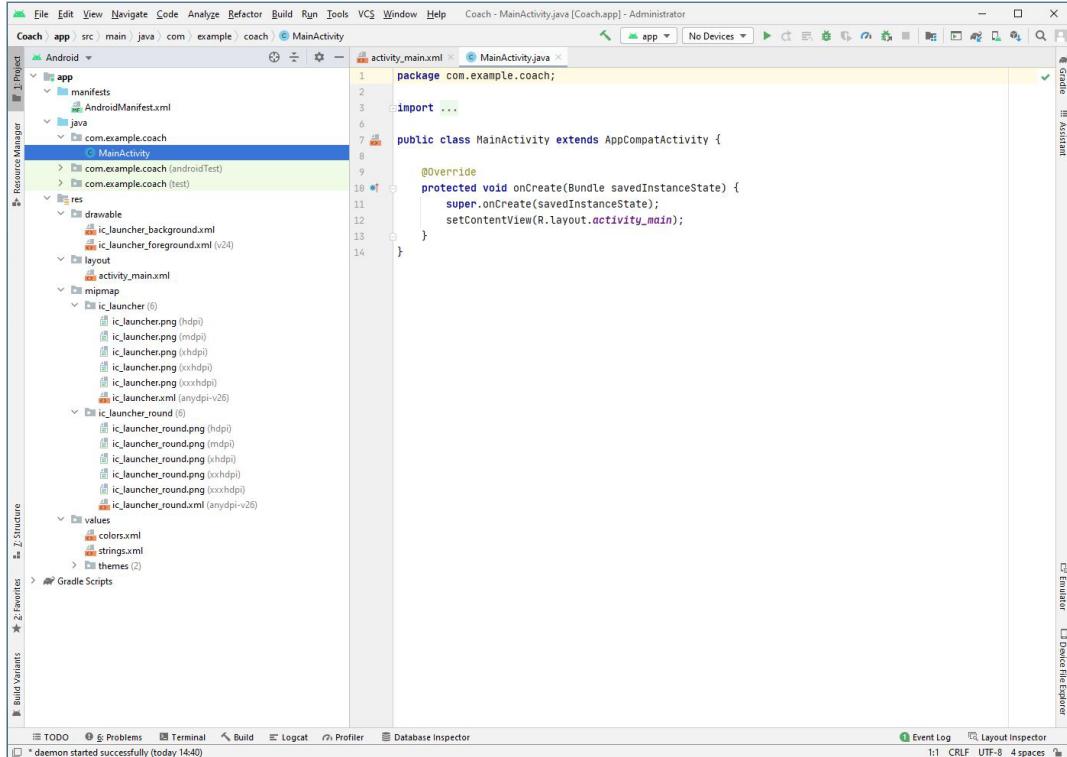
Il est possible que dans la partie droite, comme on le voit dans la capture précédente, il y ait une partie "What's New". Fermez cette partie en cliquant sur le "-" en haut à droite de cette zone (hide) ou sur "Assistant" qui est marqué verticalement tout à droite. Le but est d'obtenir la fenêtre suivante :



Si vous avez une partie basse ouverte (partie qui affiche entre autres, les erreurs), fermez-là en cliquant sur le "-" à droite. Pour le moment, on n'en a pas besoin.

1E. Structure d'un projet Android

Prenons le temps de comprendre la structure d'une application Android. Dans la colonne de gauche, en cliquant sur les flèches horizontales à gauche, déployez les différentes parties (manifests, res et sous parties de res) pour obtenir ceci :



Une application Android contient les parties suivantes :

manifests : contient ici `AndroidManifest.xml` qui représente la description de l'application (vous pouvez double-cliquer dessus pour voir son contenu, puis fermez-le).

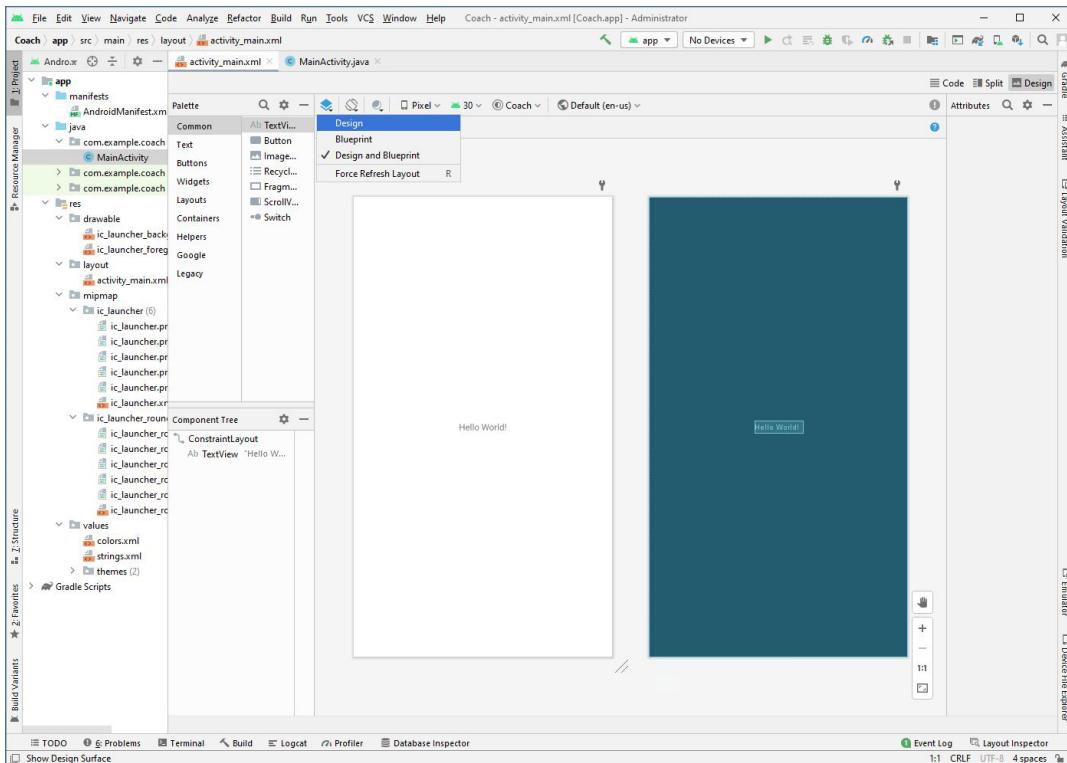
java : contient les sources du projet (avec des parties pour les tests, que vous ne toucherez pas pour le moment). Normalement `MainActivity.java` est déjà ouvert. Vous pouvez jeter un œil : c'est la classe de démarrage de l'application. Elle contient déjà un peu de code.

res : contient les ressources accessibles au projet (images, interfaces, constantes de textes, fichiers...) en particulier layout qui contient les interfaces (que l'on appelle Activity dans une application mobile).

1F. Activity dans un projet Android

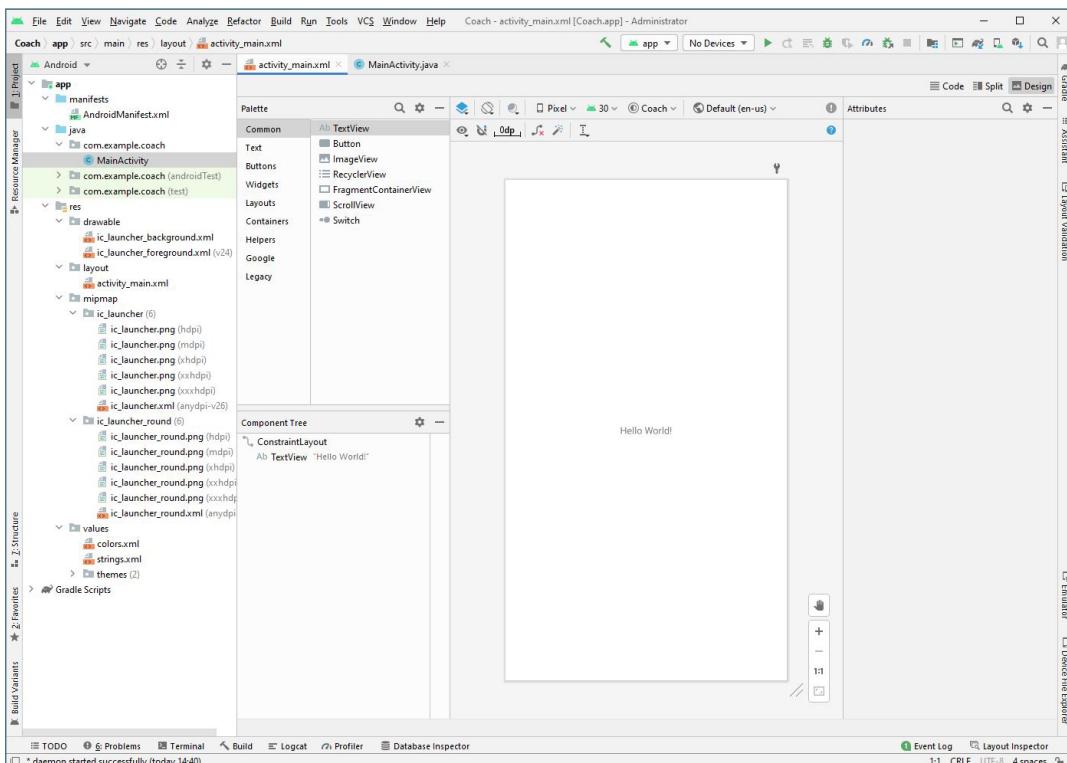
Un projet Android est constitué d'une ou plusieurs "Activity" qui sont en fait les interfaces du projet. Chaque interface possède un fichier XML contenant les balises des objets graphiques, et généralement un fichier java associé qui permet de gérer l'interface et les événements associés.

Dans la partie centrale, vous retrouvez la visualisation des fichiers ouverts. Les activity apparaissent en mode XML ou en mode visuel. Pour voir celui qui correspond à `MainActivity.java`, cliquez sur l'onglet `activity_main.xml` (car normalement il est déjà ouvert) ou doublez à gauche sur "`activity_main.xml`". En réduisant un peu les colonnes de gauche et droite et/ou en agrandissant la fenêtre et en utilisant "+" pour agrandir (en bas à droite), vous devriez obtenir environ ceci :



Au centre se trouve le visuel dans lequel vous pourrez placer et voir les objets graphiques. Deux types de visuels sont proposés. À mon avis un seul est suffisant. Pour n'en afficher qu'un, cliquez sur l'icône "Select Design Surface" en haut à gauche de la zone de visualisation (comme montré ci-dessus), et sélectionnez "Design".

Vous allez obtenir ceci :



Remarquez en haut à droite, les 3 possibilités d'affichage :

- "Design" (l'option actuelle) pour avoir le résultat visuel de l'activity ;
- "Code" pour voir le code XML correspondant ;
- "Split" pour avoir les 2 à la fois.

Testez les 3 pour vous faire une idée puis revenez sur "Design".

1G. AVD Manager

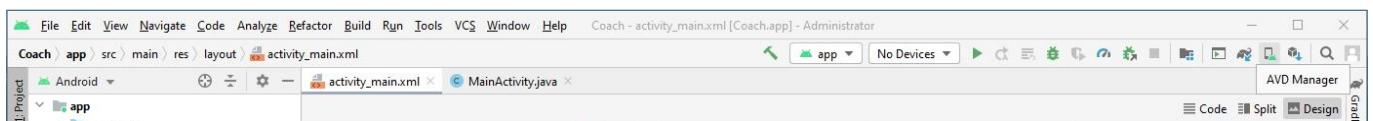
Il faut maintenant s'occuper de configurer l'émulateur qui va permettre de tester l'application sur l'ordinateur, comme si c'était sur un vrai téléphone.



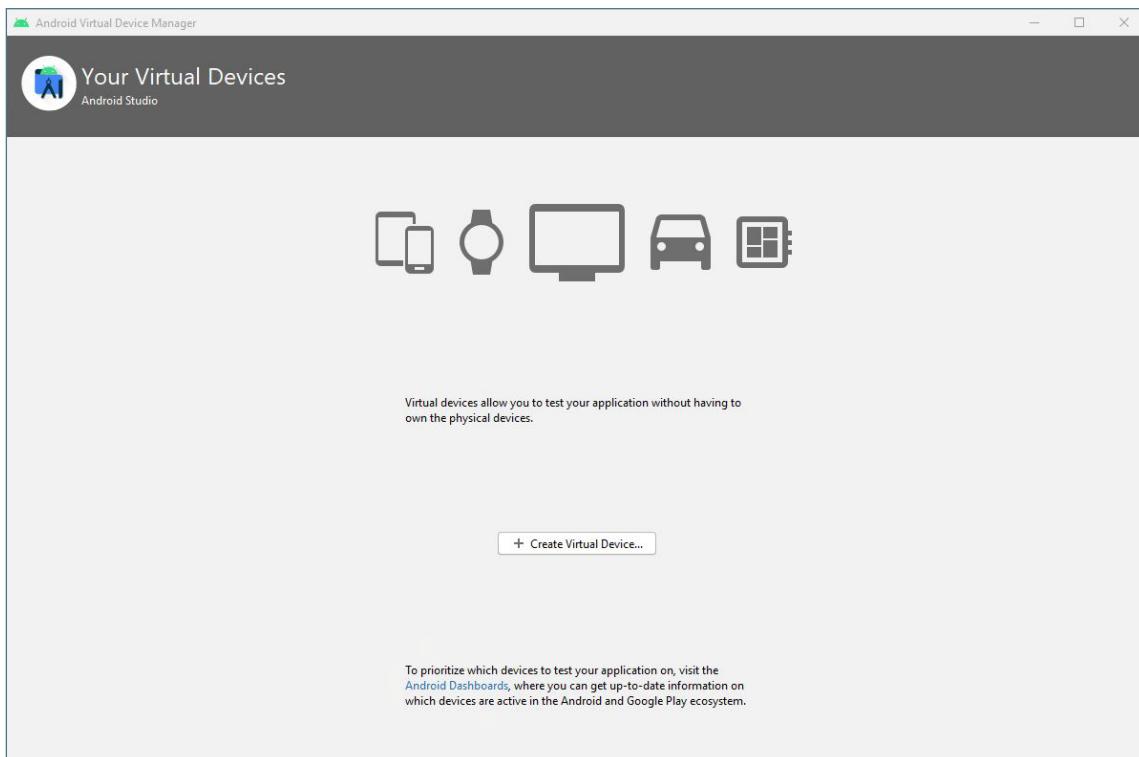
Rappel

Si vous avez un téléphone de type Android, vous pouvez aussi le brancher directement sur votre ordinateur pour faire les tests sur le téléphone en mode débug (après avoir donné l'autorisation sur votre téléphone). Ne passez pas pour autant cette partie qui vous montre comment configurer un émulateur sur votre ordinateur, car lorsque vous arriverez à la partie sur la base distante, les tests sur le téléphone en mode debug ne fonctionneront pas.

Cliquez sur "AVD Manager" (bouton en haut à droite, la position est montrée sur la capture ci-dessous) ou menu "Tools > AVD Manager".

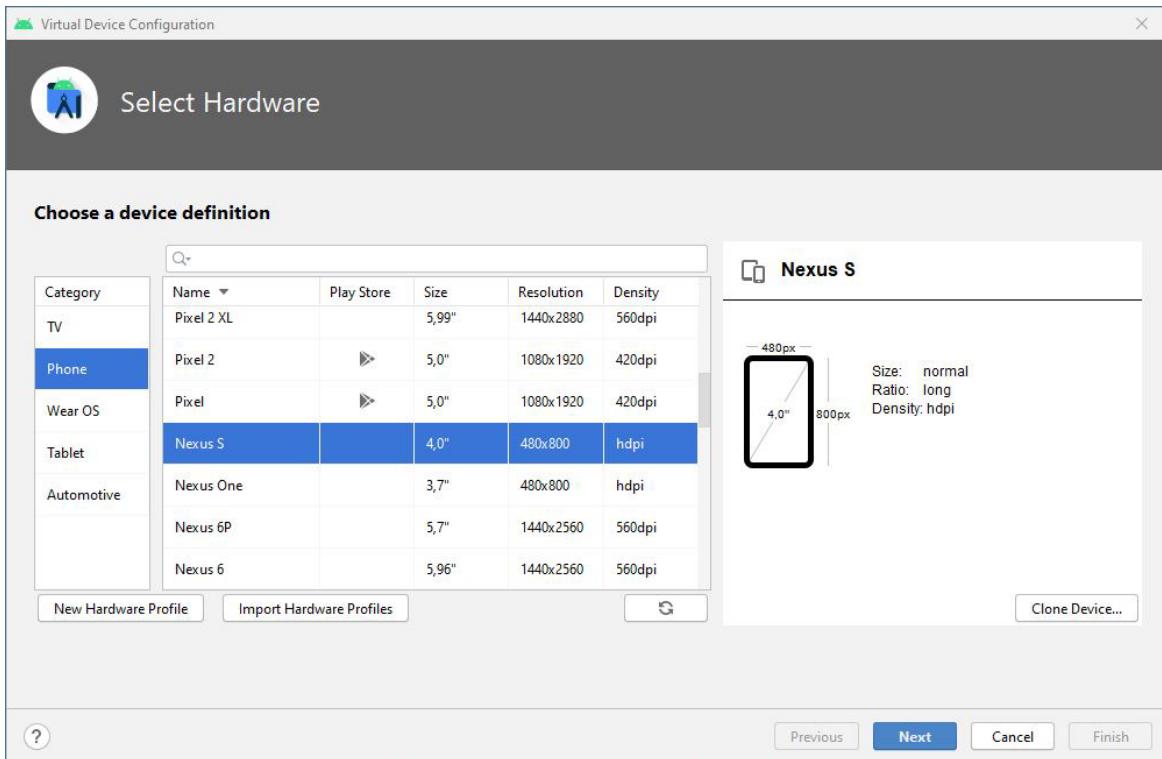


Vous obtenez ceci :

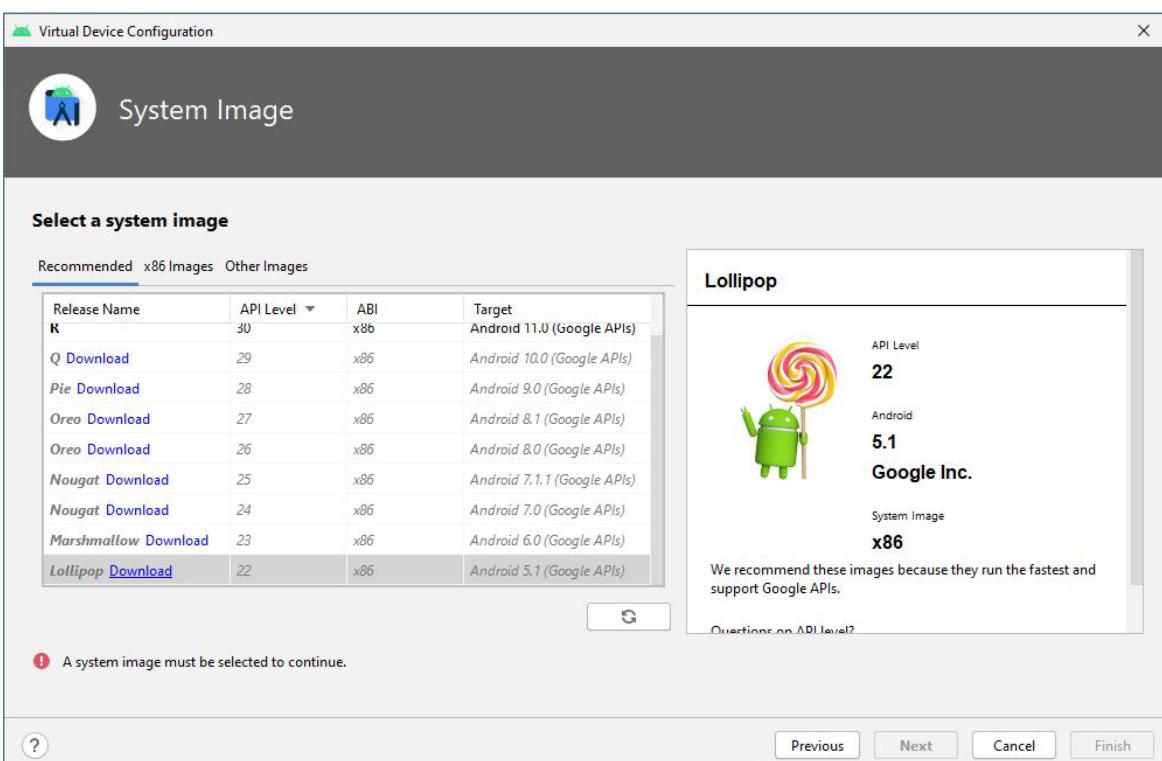


Cet affichage apparaît seulement quand aucun émulateur n'est créé.

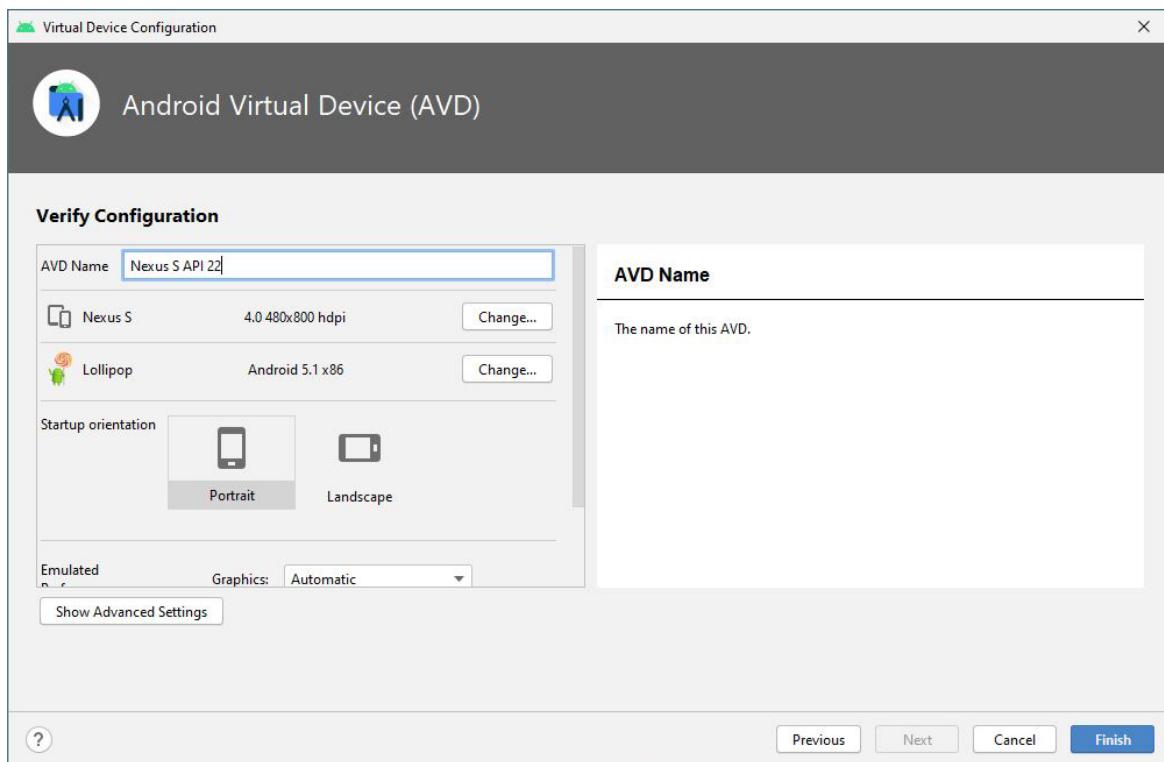
Cliquez sur "Create Virtual Device", sélectionnez "Nexus S" comme ci-dessous (et pas "Nexus 5") :



En réalité, vous pourriez sélectionner à peu près n'importe quel appareil pour faire les tests. D'un appareil à l'autre, il y a quelques nuances (taille, rapidité, options d'affichage, fonctionnalités...). Les différents appareils vont se comporter quasiment comme des vrais. À titre informatif, si en plus vous sélectionnez un appareil qui à l'option "Play Store", vous pouvez même accéder au "play store" et installer des applications. Ici, on fait le choix de "Nexus S" car il est assez léger. Faites Next. Dans la nouvelle fenêtre, onglet "Recommended" (normalement sélectionné par défaut), cliquez sur "download" de Lollipop (ou, suivant la version d'Android Studio que vous avez, choisissez l'API level le plus bas) et laissez l'installation se faire :



En fin d'installation, cliquez sur Finish. Maintenant, Lollipop ne comporte plus le lien download, et vous pouvez le sélectionner et cliquer sur Next.



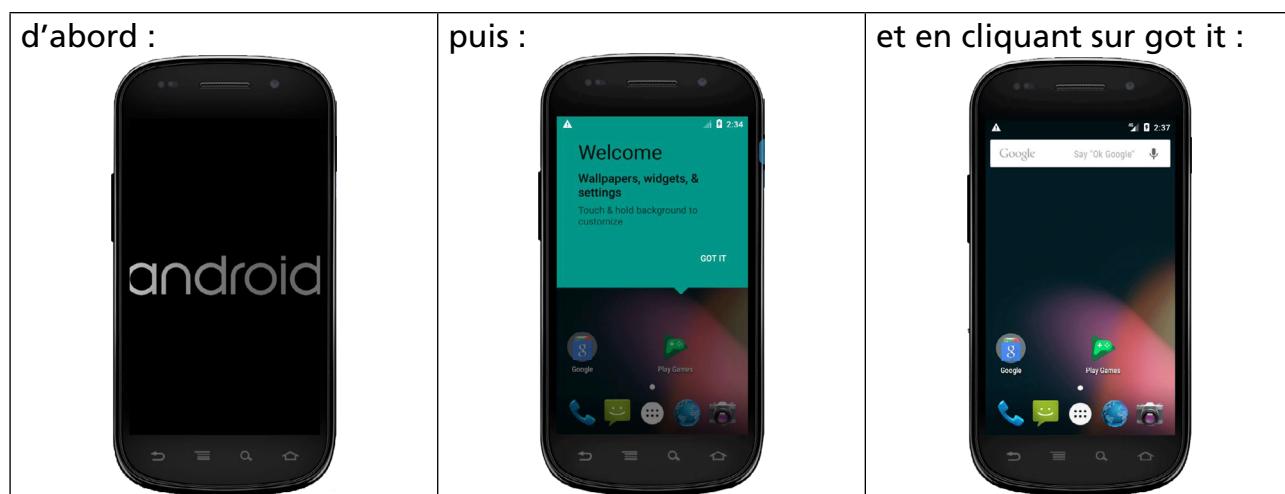
Sur cette dernière fenêtre, vous pouvez éventuellement donner un nom à votre AVD ou garder le nom proposé. Il est préférable de garder le nom proposé car il représente bien les caractéristiques de l'AVD. Cliquez sur Finish. Cette fois, dans l'AVD Manager, vous devriez avoir un émulateur créé :



C'est avec cet émulateur que vous allez pouvoir tester l'application directement sur l'ordinateur. Vous pouvez bien sûr créer plusieurs émulateurs pour voir le rendu sur des supports différents. Si vous avez à votre disposition un support mobile Android (smartphone ou tablette), vous pourrez aussi faire des tests directement sur le support.

Cliquez sur la petite flèche à droite (partie Actions) pour lancer l'AVD.

Sans rien toucher, il va passer par plusieurs étapes :



Si vous utilisez un émulateur qui fait apparaître un verrou, il est possible de faire glisser le verrou avec la souris (comme avec le doigt sur un téléphone).

Il est conseillé de laisser tout le temps ouvert l'émulateur pour éviter le temps d'ouverture. En revanche, la fenêtre AVD Manager peut maintenant être fermée.

1H. Premier test

Dans Android Studio, on va tenter maintenant d'exécuter l'application qui doit juste afficher "Hello World!". Cliquez sur la flèche verte en haut à droite (Run 'app').

Si plusieurs émulateurs sont installés (ou votre mobile branché pour servir d'émulateur), il est demandé d'en choisir un. Pour ce premier test, si vous avez un téléphone branché, ça vaut peut-être le coup de tester sur les deux supports. Par la suite, faites comme vous voulez. Si un seul émulateur est configuré (cas actuel si vous n'avez pas branché votre mobile), alors l'application va normalement se lancer directement.

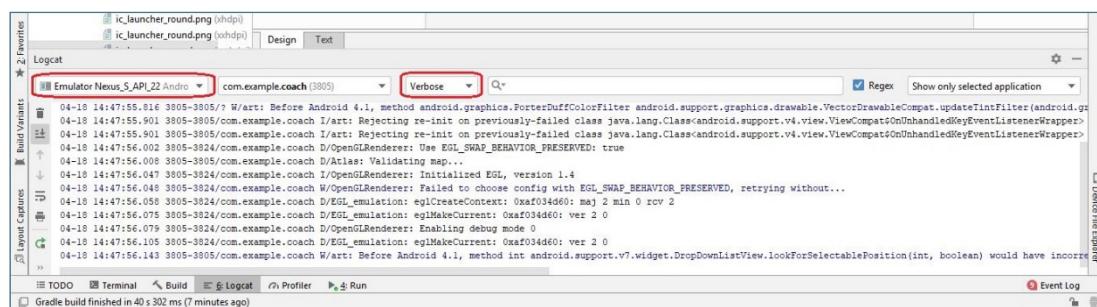
Il est possible qu'une installation complémentaire soit demandée : acceptez en cliquant sur "Install and Continue" puis "Finish" en fin d'installation. Ne vous inquiétez pas : la configuration de départ est un peu longue. Ensuite vous n'aurez plus à passer par ces étapes.

Ensuite, l'application prépare l'APK (le fichier exécutable) pour l'affichage sur l'émulateur. Attention, cela risque de mettre un peu de temps (vous voyez tout en bas d'Android Studio encore une fois le cercle tourner et une barre de progression).

Normalement, il se passe des choses dans la fenêtre du bas. Cliquez sur l'onglet "Logcat" qui permet de voir tous les messages et les erreurs. Cet onglet vous servira souvent pour contrôler votre code en cours d'exécution. Deux points importants au niveau de la fenêtre Logcat :

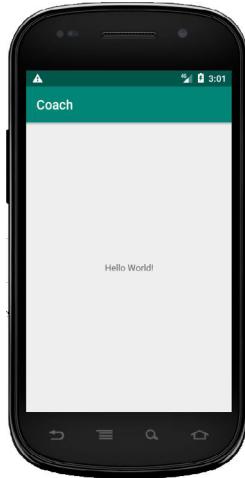
Si vous avez plusieurs émulateurs, vérifiez que vous êtes positionné sur le bon (celui que vous voulez contrôler) : il s'affiche dans le combo en haut à gauche (que j'ai entouré en rouge à gauche, dans la capture ci-dessous).

Soyez toujours en "Verbose" (contrôlez le combo du milieu, que j'ai aussi entouré en rouge) car vous pourrez ainsi voir tous les messages (erreurs, warnings, messages simples...).



Si le logcat est vide (cela peut arriver lorsqu'on l'ouvre après que l'application soit lancée), alors arrêtez l'exécution (avec le carré rouge) et relancez (avec la flèche verte). Cliquez à nouveau sur Logcat pour fermer l'onglet.

Au bout d'un moment, l'émulateur doit afficher le résultat du programme :



Petite remarque : si vous obtenez une couleur de bandeau différente, ce n'est pas grave. Si tout a fonctionné jusque-là, vous êtes prêt à travailler. Dans Android Studio, cliquez sur le carré rouge (à droite de la flèche verte) ou menu "Run > Stop 'app'" pour arrêter l'exécution. Pensez à le faire à chaque fois que vous avez terminé un test. Normalement, l'application a disparu de l'écran de l'AVD.

2. Création d'une interface simple

Étape par étape, vous allez créer une application simple qui va ensuite se complexifier pour exploiter différentes fonctionnalités.

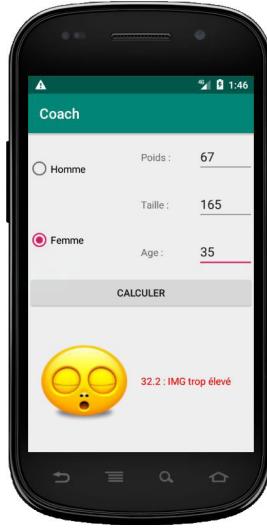
Le but de cette application est de calculer l'IMG (Indice Masse Grasse) d'une personne, en fonction de son poids, sa taille, son âge et son sexe.

L'intérêt de réaliser ce calcul est qu'il va permettre d'aborder plusieurs notions. D'abord, au niveau des objets graphiques, on a besoin de zones de saisie (taille, poids, âge), de boutons radio (sexe), de bouton (pour lancer le calcul), d'image (pour une visualisation graphique du résultat), de zones de texte (pour l'affichage du résultat), de liste interactive (pour l'historique des résultats enregistrés et les interactions sur cette liste), de méthodes de sauvegardes (sérialisations, base locale, base distante) et de manipulation de multi-interfaces (pour passer de l'interface de calcul à l'interface de l'historique).

2A. But de l'application à l'étape 1

Dans cette première étape, vous allez créer la première interface qui permet de saisir les informations et de calculer l'IMG correspondant.

Dans cette version, le résultat sera donné en direct et perdu dès qu'on arrête l'application. Voici un exemple de résultat final attendu :



2A1. À quoi correspond L'IMG ?

C'est un indice, exprimé en pourcentage, permettant de juger la proportion de tissus graisseux chez un adulte, par rapport à sa masse musculaire.

Voici comment se calcule l'IMG (information trouvée sur Wikipédia).

$$\text{IMG} = (1,2 * \text{Poids}/\text{Taille}^2) + [0,23 * \text{age}] - (10,83 * S) - 5,4$$

- avec $S = 0$ pour une femme, $= 1$ pour un homme
- taille en mètres mais que l'on saisira en cm pour éviter la saisie de la virgule.

2A2. Interprétation des résultats

Femmes :

- < 15 % : trop maigre ;
- 15-30 % : normal ;
- > 30 % : trop de graisse.

Hommes :

- < 10 % : trop maigre ;
- 10-25 % : normal ;
- > 25 % : trop de graisse.

Ces résultats peuvent ne pas être cohérents si le corps est musclé.

Maintenant que vous avez vu le but de l'application et à quoi va ressembler l'interface, on va pouvoir commencer la construction du projet.

2B. Construction de l'interface

Au niveau du fichier `activity_main.xml` (et de tous les fichiers XML), toute modification dans la partie XML (onglet Text) aura une incidence dans la partie visuelle (onglet Design), et vice-versa.

2B1. Présentation de l'interface

Rappel de l'interface qui va être construite :

The diagram illustrates the decomposition of a user interface into three nested containers (layouts). The main container (top) contains two smaller containers (radio buttons and input fields) and a 'CALCULER' button. The middle container contains the 'CALCULER' button. The bottom container contains an emoji face and an error message.

À droite, vous avez un visuel du découpage en "gabarits". En effet, pour positionner correctement les éléments sur l'interface, vous allez gérer des layouts, appelés aussi "gabarits" ou "conteneurs". Chaque layout pourra contenir un ou plusieurs objets graphiques et/ou un ou plusieurs autres layouts.

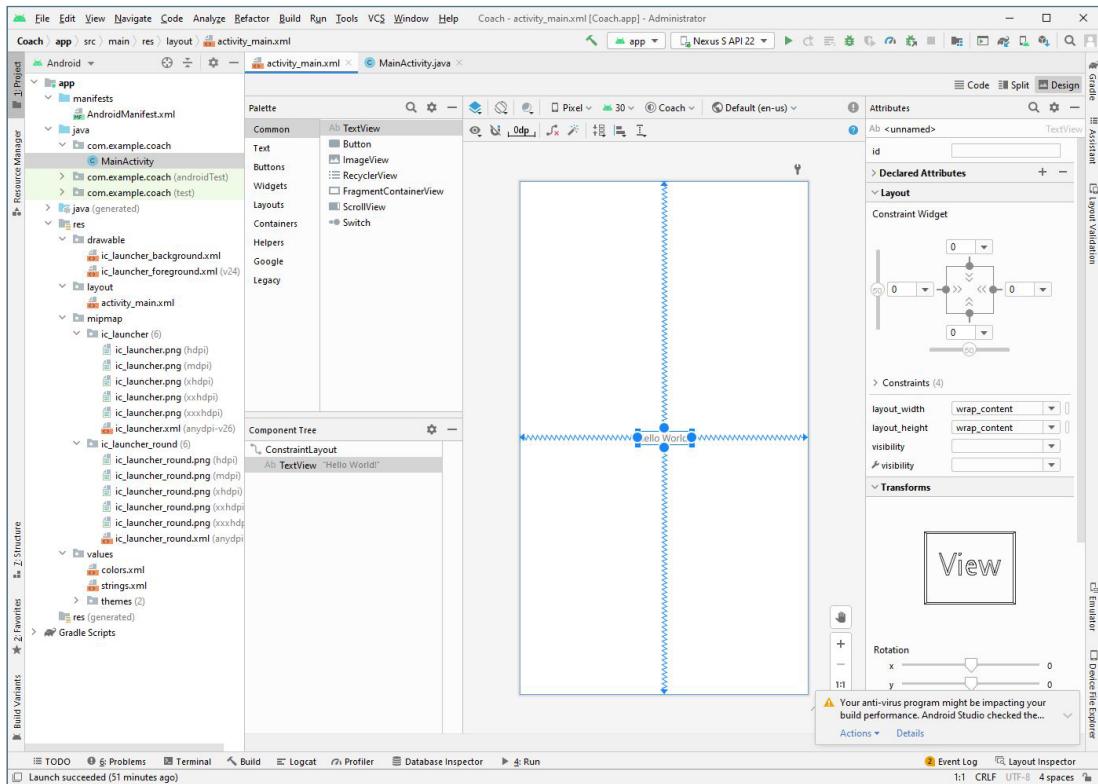
Voici comment va être construite cette interface :

Le gabarit principal va contenir 3 gabarits positionnés les uns en dessous des autres. Le gabarit du haut va contenir 2 gabarits l'un à côté de l'autre, pour placer, à gauche les boutons radios, et à droite le poids, la taille et l'âge (eux-mêmes placés dans 3 gabarits superposés). Le gabarit du milieu va contenir le bouton pour calculer. Le gabarit du bas va contenir l'image et le résultat.

Gardez en mémoire cette structure de construction car c'est elle que nous allons suivre.

Pourquoi passer par des conteneurs et ne pas positionner directement les informations où l'on veut ? Parce que le positionnement direct en fonction des coordonnées n'est pas possible si l'on désire que l'application s'adapte à tout type de support. Il faut gérer des conteneurs qui permettent de positionner horizontalement, verticalement, avec des possibilités de centrage, d'alignement... Le gros avantage de ce principe est de s'adapter au format d'affichage, quel que soit le type d'écran. C'est le principe du "responsive".

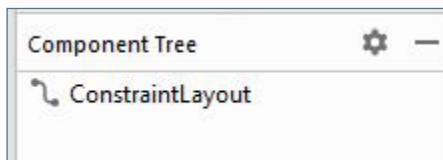
Vous allez donc apprendre à utiliser la partie graphique, mais à chaque étape, n'hésitez pas à aller voir le code XML généré : c'est important de comprendre le code correspondant. Vous êtes donc sous Android Studio, fichier activity_main.xml, onglet Design. Commencez par sélectionner le texte qui est au centre ("Hello World!") :



Appuyez sur la touche suppr pour le supprimer. La fenêtre est maintenant vide.

2B2. Le gabarit principal

Vous remarquez dans une zone à gauche, la structure des conteneurs (Component Tree) :



Cette structure va évoluer au fur et à mesure que l'application va se construire.

Le gabarit (ou layout) principal est le conteneur général de l'application. Il va contenir tous les objets graphiques et éventuellement d'autres conteneurs.

Dans le fichier XML, observez le code suivant :

```
<?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"/>
```

Pour le moment, excepté la balise spécifique de départ qui donne la version du xml et l'encodage, il n'y a qu'une balise qui est un conteneur de type ConstraintLayout. Chaque balise xml comporte un type, des

paramètres éventuels (avec nom="valeur") et, quand la balise est un conteneur, elle peut contenir d'autres balises. Pour le moment, le conteneur ne contient rien, voilà pourquoi cela se termine par "/>". Vous verrez plus loin comment le code évolue quand une balise contient une information ou une autre balise.

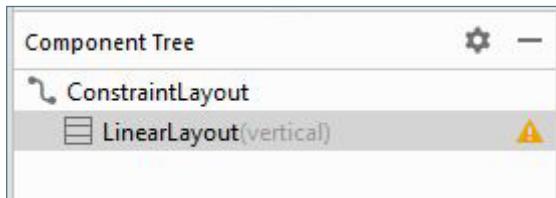
Vous remarquez, entre autres, les attributs :

- layout_width : permet de spécifier la largeur réservée pour ce gabarit
 - "match_parent" : indique que le gabarit doit prendre toute la largeur disponible sur l'écran (c'est le cas ici) ;
 - "wrap_content" : indique que le gabarit ne doit occuper que la place qui lui est nécessaire (par rapport à son contenu) ;
 - "Xdp" : fixe la largeur à X pixels à densité indépendante (1dp = 1px sur un écran de 160px) quand on veut une taille fixe ; (il existe d'autres mesures). Bien sûr cette possibilité est à éviter.
- layout_height : permet de spécifier la hauteur réservée pour ce gabarit (mêmes valeurs possibles que pour layout_width).

Lors d'ajout d'autres types de layouts, vous aurez aussi l'occasion de voir les attributs suivants :

- padding : permet de définir les marges.
- layout_weigh : permet de donner la proportionnalité de la place par rapport aux autres objets (par exemple si vous mettez tout à 1, tout sera équilibré au niveau place, alors que si un layout est à 2, il prendra 2 fois plus de place).
- orientation : permet de choisir le sens d'imbrication des objets et/ou des gabarits contenus dans le gabarit actuel. Par exemple le layout vertical permet de contenir des éléments positionnés les uns en dessous des autres.

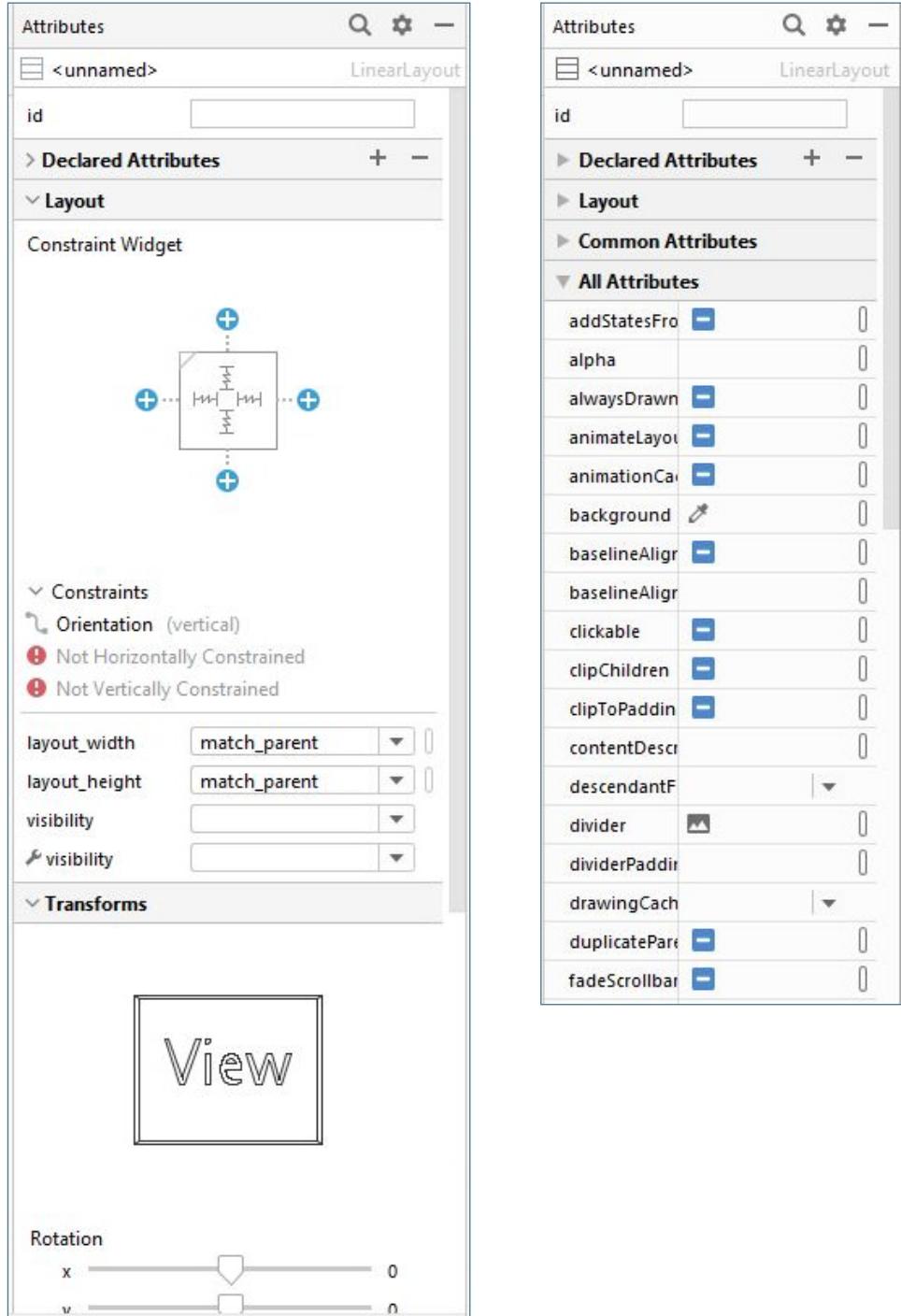
Le but est d'avoir un premier gabarit, de type vertical, de la taille totale de l'interface et qui permettra de positionner 3 gabarits l'un en dessous de l'autre. Dans l'onglet Design, partie Palette (à gauche), sélectionnez layouts, puis "LinearLayout(Vertical)" et faites-le glisser en dessous, sur "ConstraintLayout" dans la partie Component Tree. Vous devriez maintenant voir l'arborescence suivante :



Ne vous inquiétez pas du triangle : c'est juste un warning qui vous dit que le layout est vide, donc il ne sert à rien. Il va bien sûr être rempli par la suite.

Dans la partie centrale de visualisation, vous avez l'impression qu'il ne s'est rien passé. En réalité, le nouveau layout prend toute la place, voilà pourquoi on ne le voit pas. Il devrait tout de même y avoir un trait bleu tout autour.

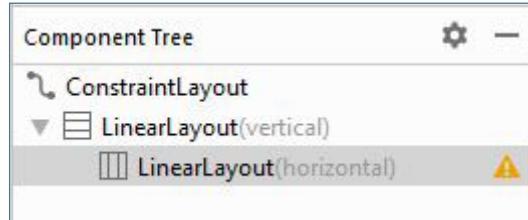
Lorsqu'un objet est sélectionné, dans "Component Tree" (ce qui est normalement le cas pour le nouveau layout que vous venez de positionner), vous pouvez voir ses attributs dans la partie droite. Si ce n'est pas déjà fait, sélectionnez le nouveau LinearLayout et vous devriez voir ceci (capture de gauche) :



Ce sont les attributs principaux. Vous pouvez voir, en tout premier, l'attribut id. Il sera renseigné quand on aura besoin d'accéder à un composant graphique, dans le code. Contrôlez que layout_width et layout_height sont bien à "match_parent". Sinon, faites la correction. Ici vous ne voyez que les attributs considérés comme "importants". Vous pouvez voir la liste complète des attributs en déployant "All Attributs" tout en bas (en cliquant sur la flèche). Vous pouvez du coup aussi fermer la partie "Layout" qui montre les attributs principaux, là encore en cliquant sur la flèche aussi fermer certaines zones (en cliquant sur les flèches, pour ne laisser ouvert que ce que vous voulez voir. Essayez, vous obtenez la capture de droite ci-dessus. Personnellement, je préfère rester en "all attributes". Vous pouvez aussi voir juste les attributs qui contiennent une valeur en déployant "Declared Attributes" : c'est aussi bien pratique.

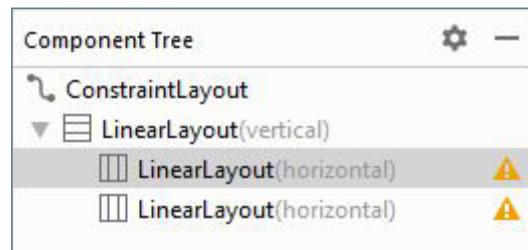
2B3. Les autres gabarits

Il faut maintenant placer les trois gabarits les uns en dessous des autres. Le premier doit contenir deux gabarits l'un à côté de l'autre, donc il doit être "Horizontal", et pour les 2 autres, les éléments sont aussi globalement les uns à côté des autres. Donc pour ces trois gabarats qui doivent être ajoutés dans le gabarit précédent, il faut choisir "LinearLayout (Horizontal)". Sélectionnez une première fois ce type de gabarit et faites glisser sur le LinearLayout précédent que vous avez placé dans Component Tree. On pourrait faire glisser les objets directement dans la fenêtre de visualisation, mais c'est plus facile de respecter la hiérarchie en les mettant directement dans le Component Tree. Vous devriez obtenir ceci (après éventuellement avoir ouvert le LinearLayout avec la flèche) :

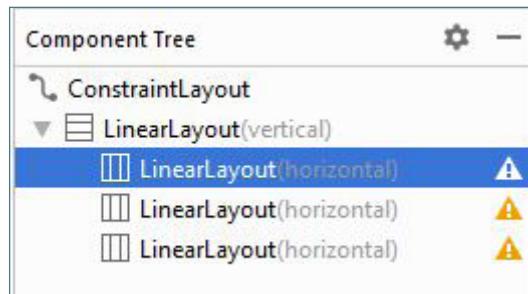


Si l'arborescence est correcte, vous pouvez continuer, sinon déplacez le LinearLayout pour obtenir cette arborescence, ou supprimez-le et recommencez. Vous remarquez que vous ne voyez rien dans la partie centrale de visualisation : pour le moment c'est normal : le layout qui vient d'être ajouté prend toute la place dans le layout du dessus.

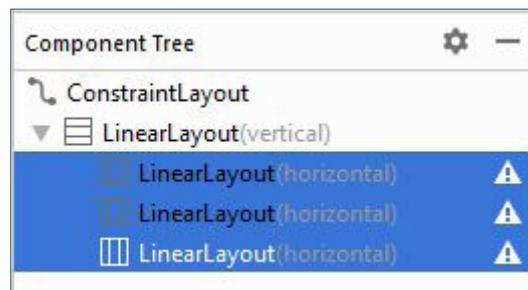
Placez un autre "LinearLayout (Horizontal)" toujours en le faisant glisser sur le LinearLayout vertical, car il doit être au même niveau que le précédent, au niveau de l'arborescence. Si vous n'obtenez pas cette arborescence, faites en sorte de l'obtenir en déplaçant le LinearLayout sur le layout parent.



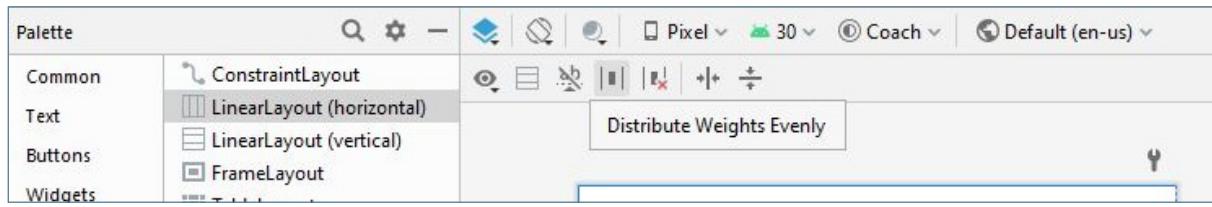
Et enfin, placez un troisième LinearLayout horizontal, toujours au même niveau, en le faisant glisser sur le LinearLayout vertical. Contrôlez l'arborescence :



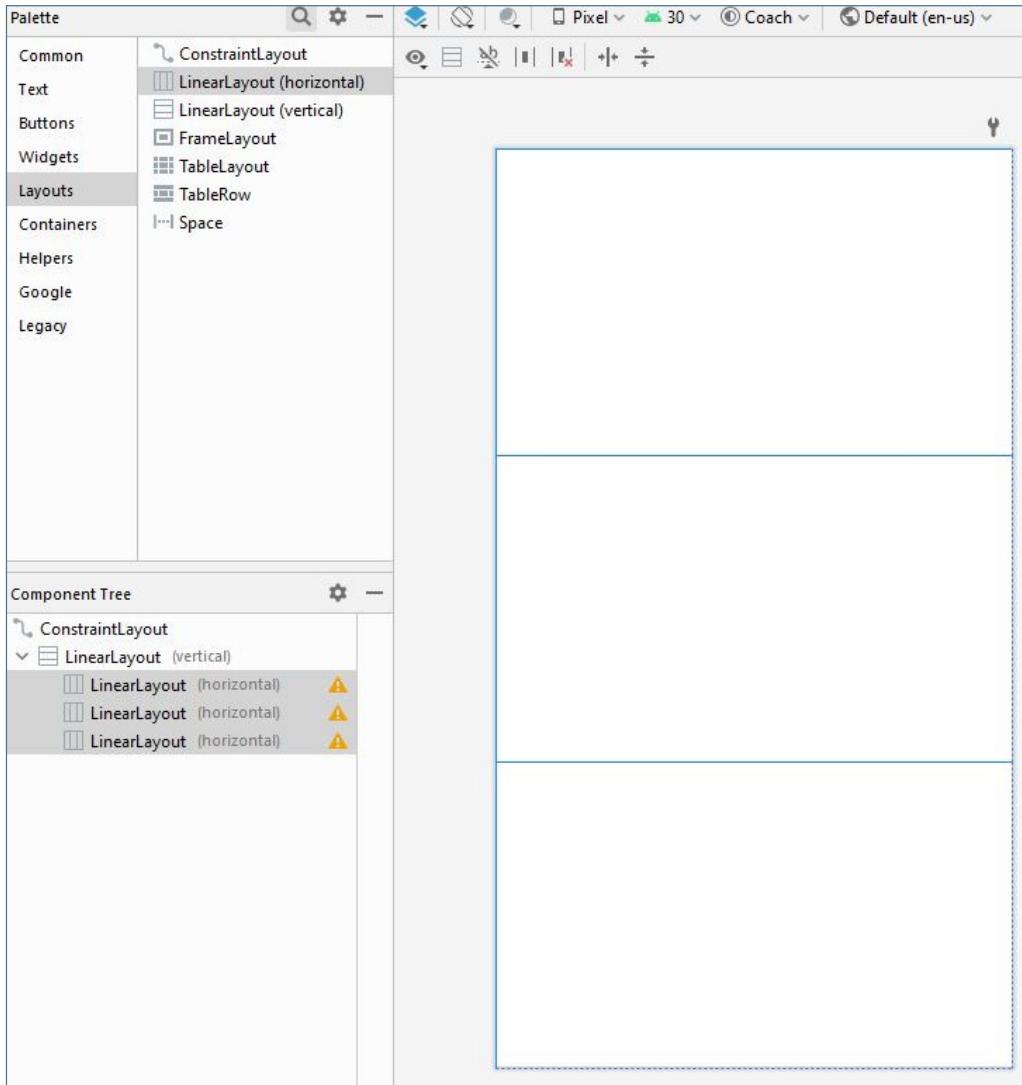
Maintenant, dans l'arborescence, sélectionnez les trois gabarits du bas, en utilisant la touche shift et la souris :



Une fois les 3 éléments sélectionnés, cliquez sur le bouton "Distribute Weights Evenly" qui se trouve au-dessus de l'interface.



Cette fois vous pouvez voir les gabarits qui ont été redimensionnés pour être tous les trois de même hauteur.

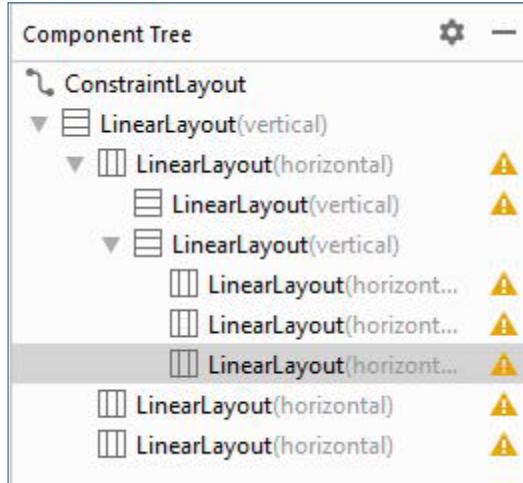


Si vous cliquez sur chaque gabarit (dans Component Tree ou dans l'interface), vous pouvez les sélectionner un par un. Sélectionnez chaque gabarit et allez dans les attributs. Vous verrez que le layout_weight a été mis à 1 pour qu'ils soient tous de taille équivalente. Pour chacun des 3 gabarits, déroulez la propriété layout_gravity pour cocher "center_horizontal".

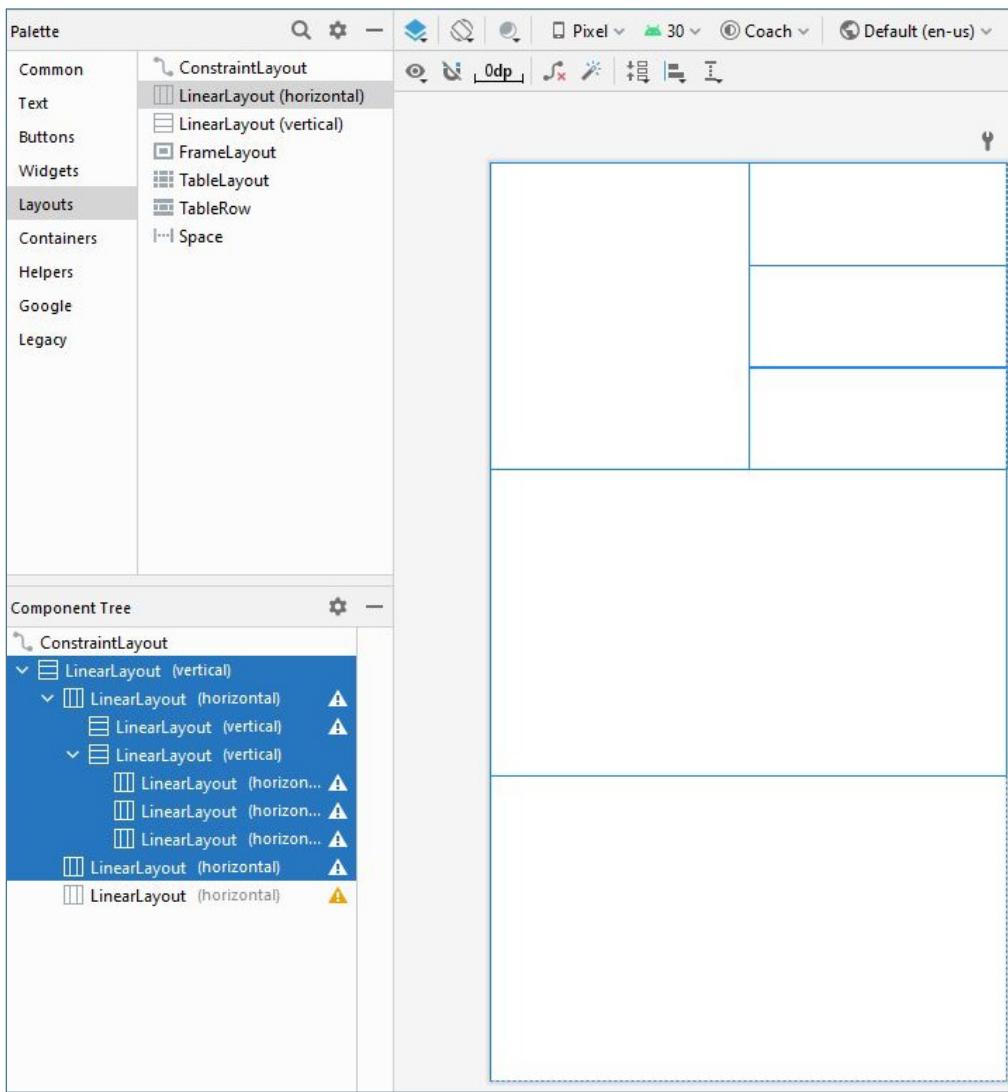
Maintenant que vous avez compris le principe, ajoutez deux gabarits verticaux dans le gabarit du haut (le premier LinearLayout horizontal), et faites aussi en sorte qu'ils soient de même largeur (toujours avec distribute weights evenly) et avec un layout_gravity centré horizontalement.

Enfin, dans le dernier gabarit que vous venez d'ajouter (celui qui est en haut à droite), insérez encore trois gabarits horizontaux et appliquez-leur les mêmes caractéristiques que les précédents (même hauteur et layout_gravity centré horizontalement).

Vous devriez obtenir l'arborescence suivante :



Si vous sélectionnez tous les gabarits, vous devriez obtenir ceci :



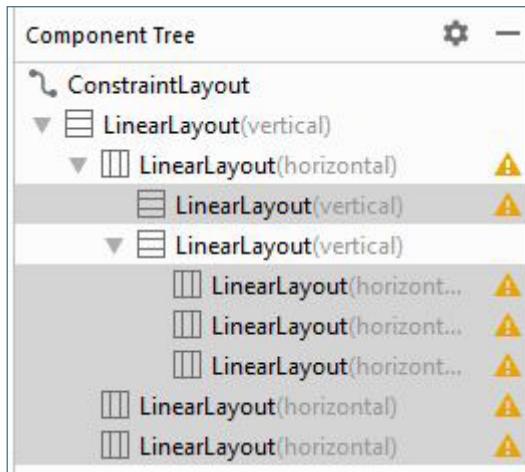
Sélectionnez juste le gabarit du milieu et amusez-vous à un peu changer sa hauteur (en tirant vers le bas le point bas central du gabarit) : vous allez voir que tout est recalculé et que le gabarit du milieu reste centré. Rappelez-vous que le gabarit du milieu ne doit contenir que le bouton, donc il est normal qu'il soit un peu moins haut que les autres. Gardez en tête l'interface que vous devez construire (qui a été présentée il y a quelques pages, au début de la "construction de l'interface"). Vous verrez cependant plus loin que la hauteur du gabarit va être relative à son contenu. Car pour le moment, le redimensionnement

que vous venez de faire "à la main" a fixé une hauteur "en dur" (observez la propriété `layout_height`), ce qu'il faut bien sûr éviter. Ce sera modifié plus loin mais pour le moment, cela va fonctionner ainsi, et c'était l'occasion de vous montrer cette possibilité.

Vous pouvez aussi jeter un œil dans le code XML qui a été généré : vous verrez clairement les imbriques des différents layouts et certains attributs choisis.

Revenez à l'onglet "Design". On désire que dans tous les gabarits, les éléments soient centrés horizontalement et verticalement. Sélectionnez par exemple le premier gabarit en haut à gauche. Dans la liste des propriétés (attributes), ouvrez `gravity` et cochez "center_vertical" et "center_horizontal".

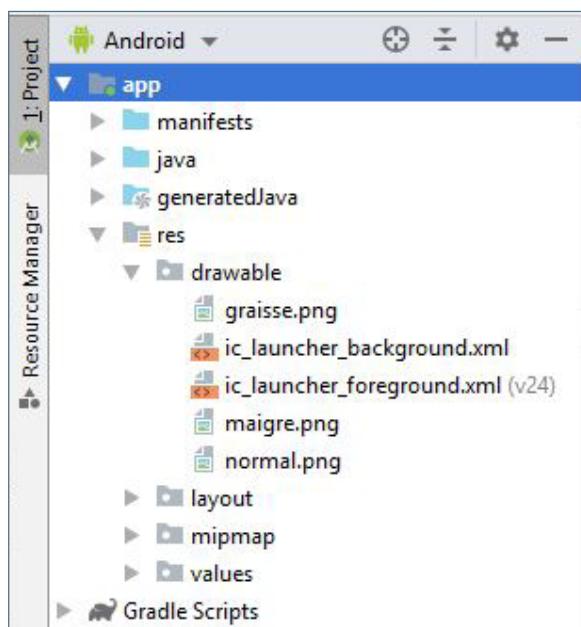
Le but est de fixer ce centrage pour tous les gabarits (attention, uniquement les gabarits de bas niveau, c'est-à-dire ceux qui ne contiennent plus de gabarits). Vous pouvez le faire en une fois en sélectionnant plusieurs gabarits avec la touche Ctrl :



Vous pouvez alors aller dans la propriété `gravity` et faire les modifications demandées : elles seront faites pour tous les objets sélectionnés.

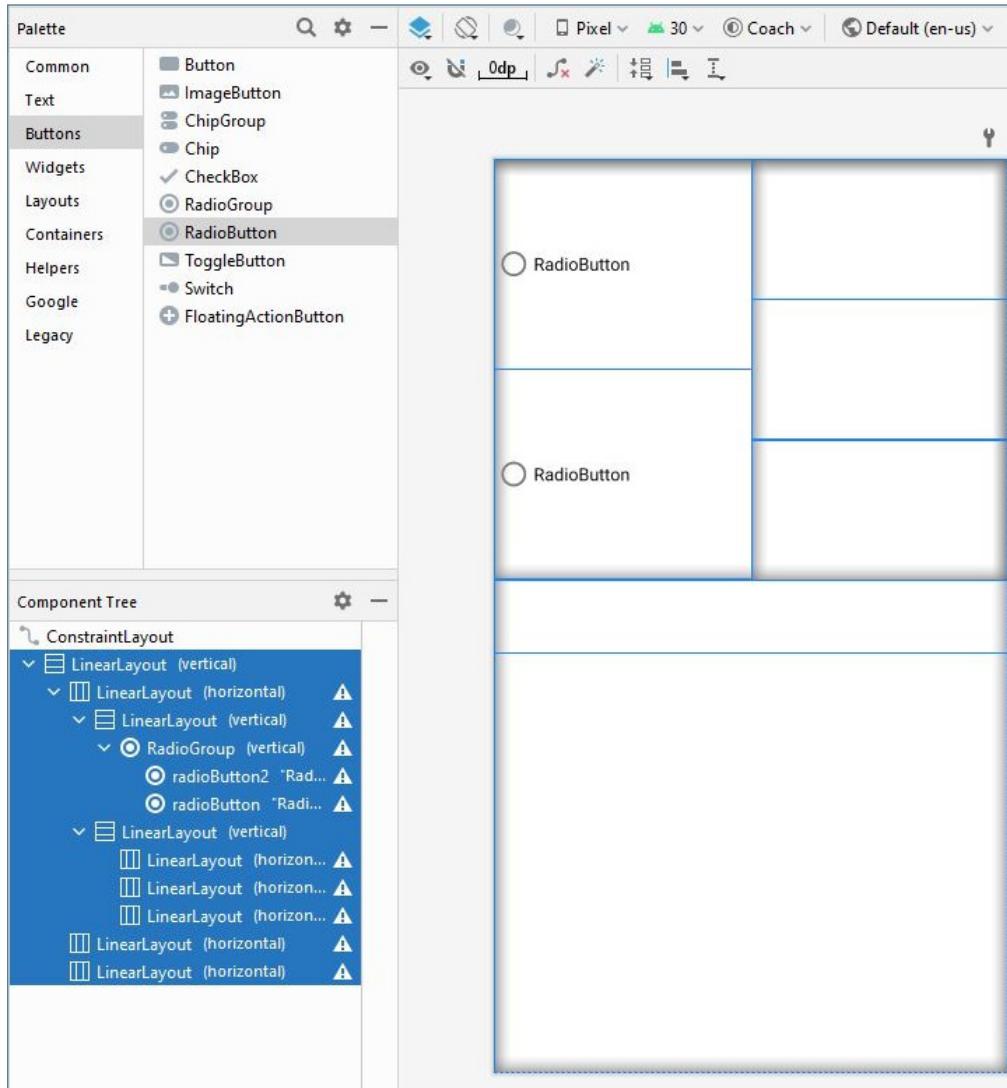
2B4. Ajout des éléments simples

Pour la suite, vous avez besoin de certaines images. Dans le zip des sources que vous avez téléchargé, allez dans le dossier "sources/fichiers images" et récupérez les images "graisse.png", "maigre.png" et "normal.png" pour les copier directement sur le disque, dans le dossier "Coach\app\src\main\res\drawable" de votre projet. Sous Android Studio, remarquez à gauche, dans l'arborescence, que les images se sont ajoutées dans "app/res/drawable" :



Dans la partie visuelle, il faut maintenant ajouter les éléments simples (boutons, zones de texte, images...).

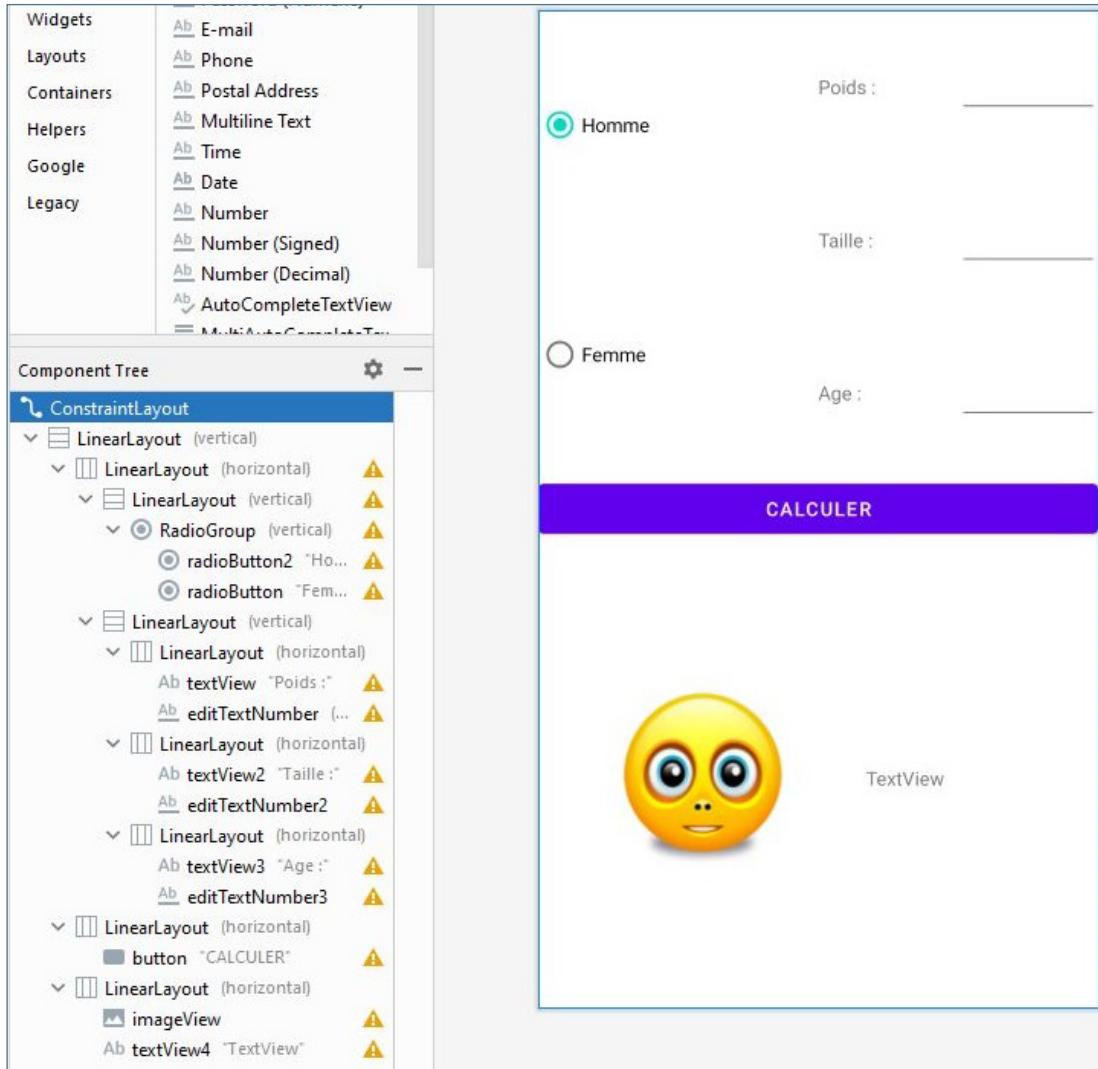
Dans les outils ("Palette", à gauche du visuel), catégorie "Buttons", faites glisser un "RadioGroup" dans le gabarit du haut, à gauche : privilégiez toujours le glisser/déposer dans l'arborescence (Component Tree) plutôt que directement dans le visuel pour être sûr que les objets soient bien positionnés. De la même manière, catégorie "Buttons", insérez 2 "RadioButton" dans le "RadioGroup". Il faut que les 2 RadioButton soient bien de même niveau et dans le RadioGroup. Comme d'habitude, pensez à équilibrer la taille des 2 RadioButton pour qu'ils prennent toute la place. Vous devez obtenir cette arborescence et visualisation :



Sélectionnez le premier RadioButton. Dans la liste des propriétés, modifiez la propriété "Text" pour mettre "Homme". Vous devriez directement le voir apparaître dans la partie visualisation. De même, changez la propriété "Text" du second RadioButton pour mettre "Femme". Pour le premier RadioButton, cochez la propriété Checked. Ce RadioButton sera sélectionné par défaut.

En observant la capture ci-dessous, vous devriez arriver à remplir par vous-même les layouts d'en haut à droite : prenez bien, dans la catégorie "Text", les "TextView" pour les labels, comme par exemple "Poids", et les "Number" pour les zones de saisie. Pensez à équilibrer les tailles, comme vous savez déjà faire, en sélectionnant les 2 objets et en cliquant sur "Distribute Weights Evenly". Modifiez correctement le contenu des labels (propriété Text). De même, il ne devrait pas y avoir de difficultés à placer le bouton (Button) dans le layout du milieu, et à changer son contenu. En ce qui concerne le layout du bas, il doit contenir un objet ImageView (dans Widgets) et un TextView, mais leur contenu n'a aucune importance pour le moment car ils seront remplis par le programme. Cependant vous êtes obligé de sélectionner

une image donc, dans la fenêtre qui s'ouvre, sélectionnez à droite "Drawable", puis au centre ouvrez "Project" et sélectionnez "normal" puis Ok. Encore une fois équilibrer avec le TextView. Au final vous devriez avoir un visuel qui ressemble à ceci (contrôlez aussi et surtout l'arborescence) :



Enfin, pour éviter qu'au lancement de l'application, le message "TextView" soit visible en bas à droite, videz la propriété Text de cet élément.

Il reste à modifier les id des composants qui vont être manipulés par programme. La propriété id est en tête de liste. Faites les modifications en passant par les propriétés ou dans le XML. Voici les id que vous devez donner (attention @+id/ n'apparaît que dans le XML ; si vous changez l'id directement dans les propriétés, ne prenez que le nom qui est marqué après /) :

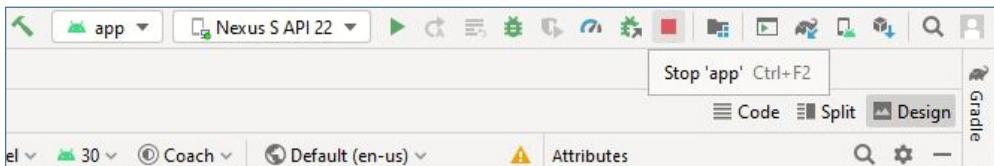
Élément	id
groupe de boutons radio	@+id/grpRadioSexe
bouton radio "homme"	@+id/rdHomme
bouton radio "femme"	@+id/rdFemme
zone de saisie du poids	@+id/txtPoids
zone de saisie de la taille	@+id/txtTaille
zone de saisie de l'âge	@+id/txtAge
bouton Calculer	@+id/btnCalc
Image	@+id/imgSmiley
Texte du message résultat	@+id/lblIMG

Il est inutile de changer les id des objets graphiques que l'on ne va pas manipuler dans le code (comme par exemple les labels qui contiennent "Poids :", "Taille :...").

Encore une petite modification en ce qui concerne la taille du gabarit du milieu. Rappelez-vous, vous aviez modifié sa taille "à la main". Sélectionnez le gabarit du milieu (celui qui contient le bouton "calculer") et regardez le contenu de layout_height : il contient pour le moment une valeur précise en dp. Changez en mettant wrap_content et mettez la valeur 0 à la propriété layout_weight. Après ces 2 modifications, normalement le layout est juste de la taille de son contenu (le bouton) et les 2 autres layouts se partagent tout le reste de la place.

La construction de l'interface est terminée. Pensez à enregistrer régulièrement votre projet (Ctrl-S).

Avant de lancer un test, pensez toujours à arrêter le précédent en cliquant sur le bouton stop (carré) si le carré est rouge.



Lancez l'application (avec la flèche verte) pour voir le visuel. Attendez l'affichage dans l'émulateur. Évidemment pour le moment, il n'y a que du visuel. Vous remarquerez tout de même qu'en sélectionnant Femme, Homme est automatiquement désélectionné, et vice-versa : cela vient du fait que les 2 boutons radio sont dans un même groupe de boutons radio, donc un seul peut être sélectionné à la fois. Vous pouvez même saisir des valeurs dans les zones de saisies (remarquez au passage que vous ne pouvez saisir que des valeurs numériques). Stoppez l'exécution (carré rouge) mais laissez l'émulateur ouvert.

2C. Association de l'interface au code Java

Passons maintenant au code qui doit s'exécuter.

À gauche, partie projet Android, développez le dossier java simple. Ouvrez le premier package (com.example.coach) et non pas l'un des autres packages réservés pour les tests. Double-cliquez sur le fichier MainActivity. Dans la partie centrale où s'affiche le code, si vous cliquez sur le + à gauche de "import ...", vous obtenez le code :

```
package com.example.coach;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Actuellement le code ne contient pas grand-chose. La classe principale (MainActivity) va être la classe de démarrage de l'application. Elle hérite de AppCompatActivity qui permet de gérer un certain type d'Activity (d'interface). Cette classe mère contient entre autres la méthode onCreate. Cette méthode est événementielle : elle se déclenche automatiquement lors de la création de l'Activity. Elle est ici redéfinie dans la classe MainActivity.

Pour le moment, cette méthode redéfinie contient 2 instructions :

- l'appel de la méthode `onCreate` de la classe mère (qui contient à priori des instructions permettant de préparer l'Activity) ;
- l'appel de la méthode `setContentView` de la classe mère, en lui envoyant le fichier XML de présentation de l'Activity, ce qui permet de paramétriser l'Activity pour qu'elle soit en mesure d'afficher les objets graphiques.

Un point intéressant à savoir : R représente le dossier res. D'où le `R.layout.activity_main` pour accéder à l'interface `activity_main` qui se trouve dans le dossier `res/layout`. On verra plus loin que R permet aussi d'accéder à un objet graphique par son id.



Les cours du CNED sont strictement réservés à l'usage privé de leurs destinataires et ne sont pas destinés à une utilisation collective. Les personnes qui s'en serviraient pour d'autres usages, qui en feraient une reproduction intégrale ou partielle, une traduction sans le consentement du CNED, s'exposeraient à des poursuites judiciaires et aux sanctions pénales prévues par le Code de la propriété intellectuelle. Les reproductions par reprographie de livres et de périodiques protégés contenues dans cet ouvrage sont effectuées par le CNED avec l'autorisation du Centre français d'exploitation du droit de copie (20, rue des Grands-Augustins, 75006 Paris).

CNED, BP 60200, 86980 Futuroscope Chasseneuil Cedex, France

© CNED 2021

87D22TDWB1921

