1. Notation des TP

Chaque semaine vous allez devoir rendre votre travail sur Moodle. Ça sera généralement la partie centrale d'un projet Eclipse. Ces documents seront évalués dans le cadre de la note de TP. Ainsi, il n'y aura pas un TP noté final, mais de multiples travaux à faire dans le cadre d'un contrôle continu.

Cette semaine, rien ne sera pas noté car c'est la prise en main des outils.

2. Avant toute chose

Il est indispensable de vider votre compte de tous les fichiers inutiles. Certains d'entre vous, parfois à leur insu, ont un compte qui frise avec la limite des quotas (1 Go). Les projets Android font environ 30 Mo chacun. Vous allez vite remplir votre compte. Si vous dépassez vos quotas, vous ne pourrez plus travailler.

Par exemple, dans `Téléchargement`, n'y a-t-il pas des choses à supprimer ?

D'autre part, si vous travaillez avec un portable personnel, les prérequis sont les suivants :

- 16Go de RAM. N'essayez même pas si vous avez 4Go et vous aurez énormément de problèmes avec seulement 8Go. Vous ne pourrez pas lancer tous les outils en même temps, Android Studio, un AVD, le navigateur internet...
- 25Go de place libre sur `C:` car Android Studio et le SDK prennent beaucoup de place.
- `C:` est un disque SSD sinon les compilations vont durer nettement plus longtemps.
- écran 17" en HD (1920x1080) parce que la fenêtre principale d'Android Studio est très complexe. Travailler sur un petit écran est devenu impossible.

3. Découverte de Android Studio

On va commencer par le principal outil de travail, l'atelier de création des applications : Android Studio. Il se trouve dans le menu `Développement`.

Au tout premier lancement, il va vous poser quelques questions :

1. D'abord laissez `Do not import settings` coché : vous n'avez encore jamais travaillé avec Android Studio, donc rien à importer.

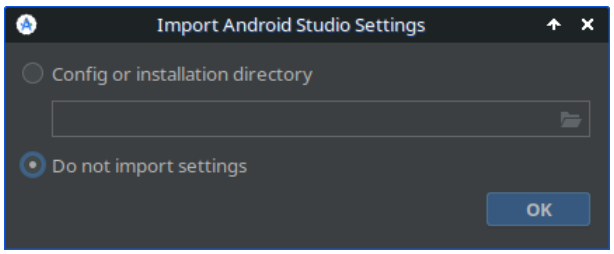

Figure 1: Import settings ?

2. Ensuite, choisissez `Don't send` concernant les statistiques à envoyer à Google.

Il y a ensuite une phase de téléchargement de certains composants du SDK, un peu lente, il faut patienter.

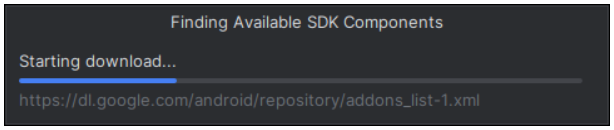

Figure 2: Download SDK Components

3. Dans le dialogue qui apparaît « Android Studio First Run », cliquez sur le bouton grisé `Setup Proxy` puis cochez `Manual proxy configuration`. C'est parce que le réseau de l'IUT est filtré par un *proxy* pare-feu : l'intermédiaire réseau qui filtre les requêtes. Il faut configurer ainsi :

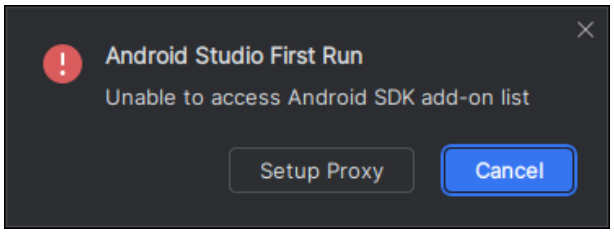

Figure 3: Studio First Run

- `HTTP` doit être coché
- Host name : `129.20.239.11`
- Port number : `3128`
- login et mot de passe : mettez `toto`

Ce login et mot de passe bidons permettent d'éviter beaucoup de boîtes de dialogue surgissant à tout moment pour vous demander une authentification.

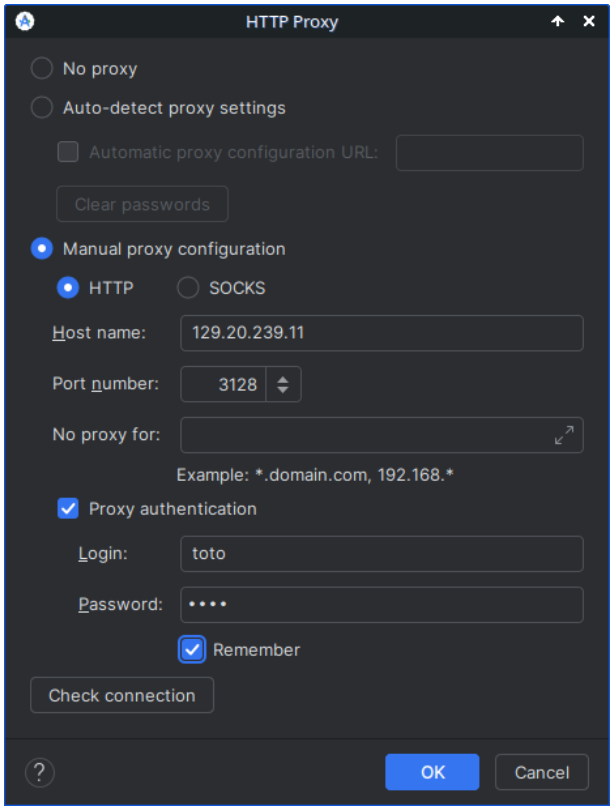

Figure 4: Proxy Settings

4. Dans la fenêtre «Welcome», cliquez sur `Next`.

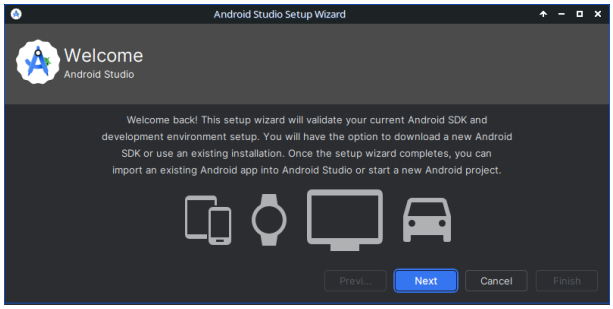

Figure 5: Welcome

5. Ensuite dans «Install Type», cochez installation de type *Custom*, cliquez sur `Next`.

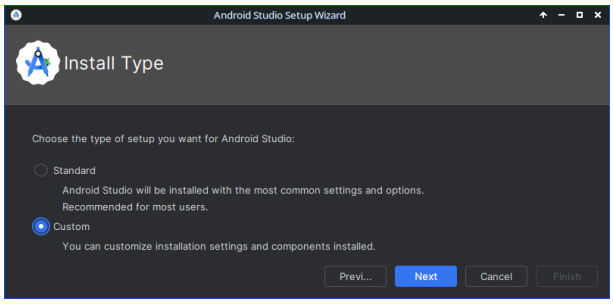

Figure 6: Welcome

6. Choisissez l'esthétique des fenêtres (sombre ou clair), cliquez sur `Next`.
7. Puis, dans la fenêtre `SDK Components Setup`, ne cochez rien car tout est déjà installé. Simplement, vérifiez en bas l'emplacement du SDK Android : `/opt/Android/Sdk`. Normalement, en bas, il vous affiche *An existing Android SDK was detected...*. Tapez sur `Next`.

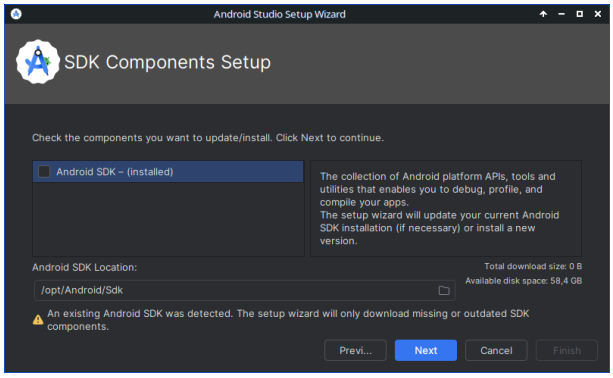

Figure 7: Welcome

8. La fenêtre principale d'Android Studio apparaît enfin, et elle apparaîtra directement les prochaines fois.

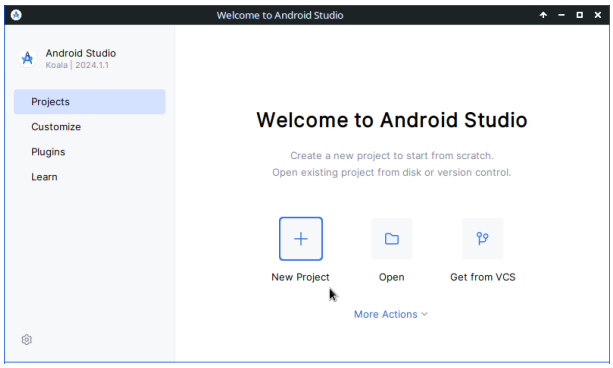

Figure 8: Fenêtre d'accueil

NB: Il y aura des différences entre les copies écran des énoncés de TP et votre installation. Android Studio change très régulièrement la position et l'apparence des vues.

3.1. Création d'un premier projet Android

1. Au tout début, vous n'aurez aucun projet Android. Dans la fenêtre d'accueil, choisissez `New Project` pour commencer à créer votre premier projet.
2. Dans la première boîte de dialogue, choisissez le type d'application souhaitée. Pour le TP1 et parce que c'est le modèle le plus simple, changez la sélection pour *Empty Views Activity*.

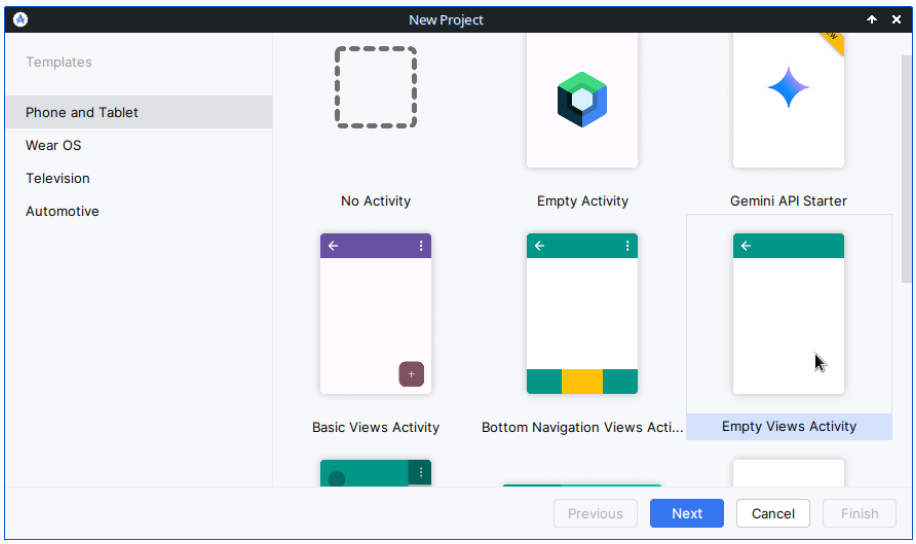

Figure 9: Type de projet

Attention, ce n'est pas *Empty Activity*, mais *Empty Views Activity*. Le type *Empty Activity* sélectionné par défaut, fait créer un projet avec Jetpack Compose qu'on étudiera dans un prochain TP. Elle est beaucoup trop complexe pour un premier TP.

3. Dans la seconde boîte de dialogue, on vous demande plusieurs choses :
   a. Le nom de l'application, mettez `TP1`,
   b. Son paquetage : mettez `fr.iutlan.tp1`,
   c. L'emplacement du projet, laissez l'emplacement proposé par défaut, `/home/etuinfo/vous/AndroidStudioProjects/...`. Si vous êtes chez vous, laissez le dossier proposé par l'assistant.
   d. Le langage proposé est Kotlin. Dans la version d'Android Studio de l'IUT, on peut encore choisir Java, mais il y a de nombreux bugs. Et dans les versions actuelles, le choix du langage a disparu.
   e. Le niveau d'API SDK voulu, par exemple la 16, 22, 24, et.. C'est la version minimale que devront avoir les smartphones pour installer votre application : ils devront être au moins aussi récents que cette API. Cela définit le niveau de fonctionnalités de votre application : s'il est trop élevé, peu de smartphones seront capables de faire tourner votre application ; trop bas, vous n'aurez pas beaucoup de fonctions agréables à votre disposition. Vous pouvez cliquer sur *Help me choose* pour avoir un graphique de la distribution des versions d'API dans le monde.

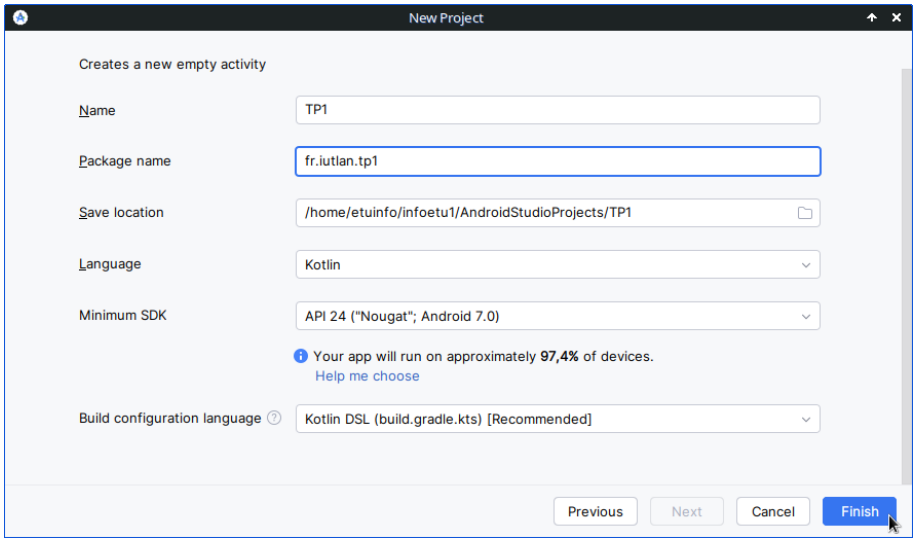

Figure 10: Détails du projet

4. Android Studio crée ensuite le projet.

Il y a parfois une fenêtre demandant de confirmer le proxy. Cochez `ne plus demander` et validez.

Il y a parfois une fenêtre « Service de portefeuilles de KDE ». Choisissez `classique, blowfich`, validez.

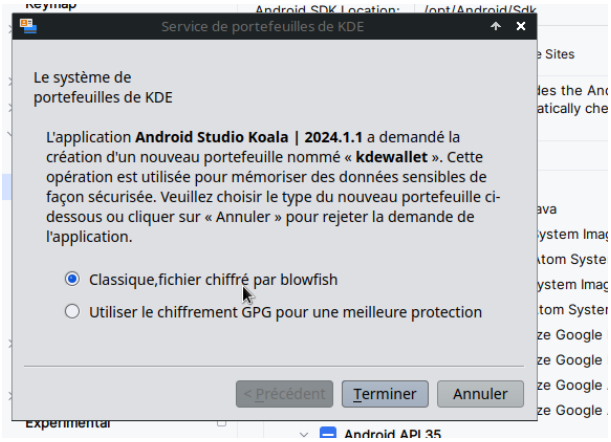

Figure 11: Demande de portefeuille

Dans la fenêtre suivante, mettez votre prénom en tant que mot de passe, ou quelque chose de très simple comme `android`, ou alors ne l'oubliez pas.

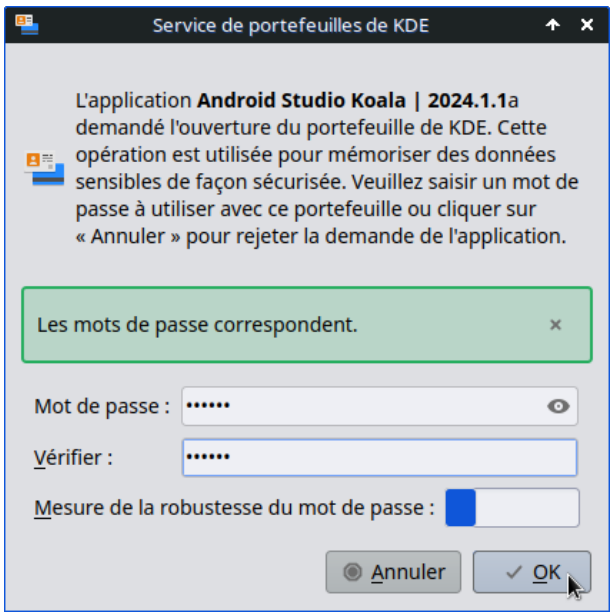

Figure 12: Mot de passe

3.2. Découverte de la fenêtre de travail

L'interface d'Android Studio est un peu déroutante au début. Elle repose sur IntelliJ qui est un concurrent d'Eclipse et de Visual Studio Code. Vous verrez rapidement qu'on ne se sert que d'une toute petite partie, assez vite apprise. Il y a des boutons sous forme d'icônes, tout autour de la fenêtre : `Project`, `Structure`, `Captures`, `Favorites`, `Build Variants`, `Terminal`, `Android Monitor`, `Messages`, `TODO`, etc. Ce sont des boutons à bascule, exclusifs entre eux sur un même côté, qui affichent des onglets ; celui qui est actif est grisé. Par exemple dans l'image ci-dessous, c'est l'onglet `Project` qui est ouvert.

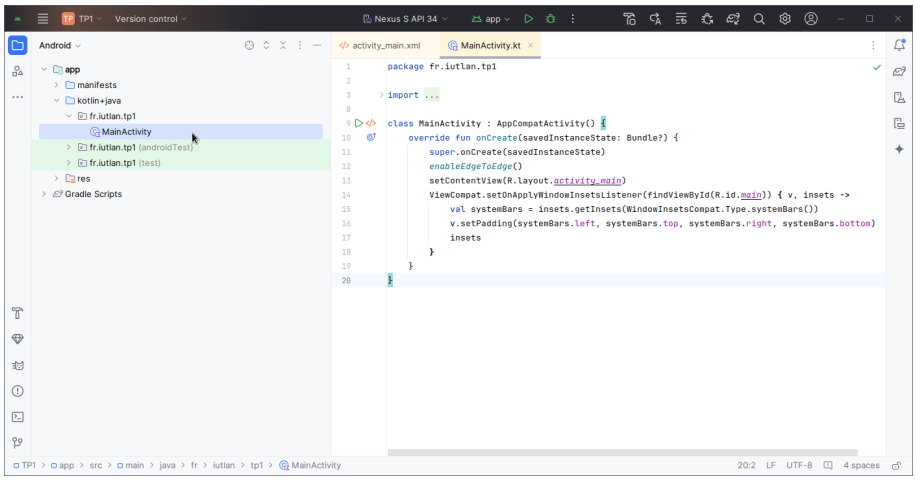

Figure 13: Fenêtre du projet

La structure du projet est visible à gauche, comme dans Eclipse. Les fichiers sont arrangés de manière pratique : manifeste, sources java et ressources, grâce au choix déroulant `Android` juste au dessus.

Prenez quelques instants pour déplier les dossiers `manifests`, `java` et `res` et regarder ce qu'il y a dedans (ça a été décrit en cours).

3.3. Configuration d'Android Studio pour les TP à l'IUT

L'installation sur les PC de l'IUT n'est pas standard parce que les machines manquent de mémoire. Vous allez devoir modifier vous-même certaines choses.

Il n'y a pas de barre de menus permanente. Il faut aller sur l'icône en haut à gauche, en forme de 4 lignes horizontales superposées. C'est là que sont les menus.

- Cliquez sur l'icône menu en haut à gauche, entouré en rouge ci-dessous.

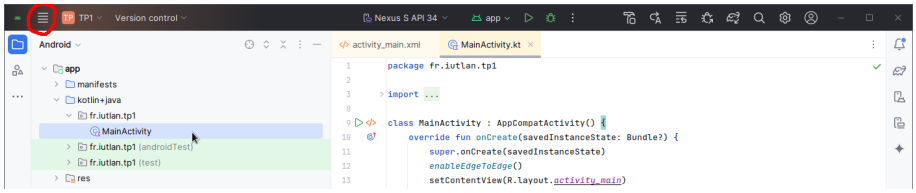

Figure 14: Barre de menus

- Dans le menu `File`, cliquez l'item `Settings`. Ça ouvre une grande boîte de dialogue.
- Dans la fenêtre, catégorie `Appearance & Behavior`, sous-catégorie `System Settings`, item `Memory Settings`, mettez ces valeurs :
  - `IDE max heap size` : 1024 MB
  - `Gradle daemon max heap size` : 1536 MB
  - `Kotlin daemon max heap size` : 1024 MB

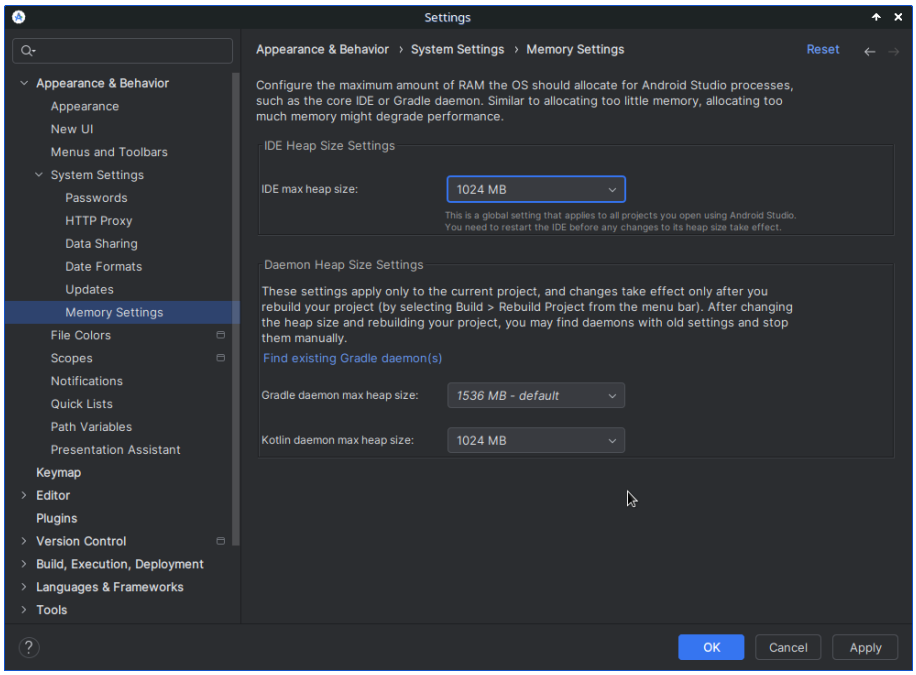

Figure 15: Réglages mémoire

- Validez puis acceptez de redémarrer Android Studio.

Au cas où ce serait nécessaire, pour recommencer la configuration de zéro d'AndroidStudio, il faut supprimer les dossiers `~/.gradle`, `~/.config/Google/AndroidStudio*` et `~/.cache/Google/AndroidStud`

3.4. Structure du projet

Allez voir le dossier correspondant au projet, dans le dossier `~/AndroidStudioProjects/TP1` de vos documents, le nom complet est dans le titre de la fenêtre Android Studio :

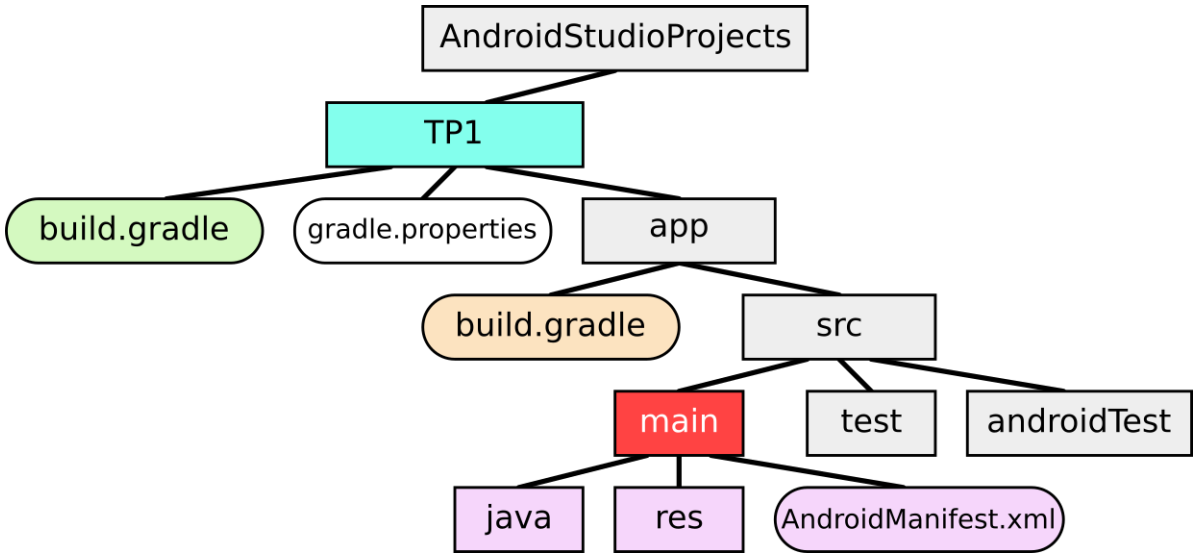

Figure 16: Structure des fichiers d'un projet Android

- Le dossier du projet contient le premier `build.gradle`, celui en vert qui indique les dépôts à utiliser pour la compilation (Google et mavenCentral). Le fichier `gradle.properties` contient la configuration du proxy. On trouve aussi un `local.properties` contenant le chemin du SDK.
- Le dossier `app` contient le second `build.gradle`, celui en orange qui décrit les caractéristiques du projet : API et dépendances.
- Le dossier `app/src/main` en rouge contient le cœur du projet : manifeste, sources et ressources.
- `app/src/main/java` contient les dossiers correspondant au package que vous avez déclaré, avec les sources Kotlin tout au bout du package, ici `MainActivity.kt`. Oui, les sources sont en Kotlin, mais placés dans le dossier Java.
- `app/src/main/res/drawable` contiendra des images vectorielles utilisées dans votre projet.
- `app/src/main/res/layout` contient les dispositions d'écran des activités de votre projet (voir le TP2).
- `app/src/main/res/menu` contient les menus contextuels et d'application du projet (voir le TP5).
- `app/src/main/res/mipmap*` contient les icônes bitmap dans diverses résolutions.
- `app/src/main/res/values` contient les constantes du projet, dont les chaînes de caractères avec toutes leurs traductions.

Ces fichiers n'apparaissent pas ainsi dans l'interface, ils sont regroupés autrement, à vous de voir comment.

4. Découverte du SDK Android

On va regarder les outils de développement : le *Software Development Toolkit* d'Android.

4.1. Gestionnaire de SDK

- Ouvrez le menu `Settings` puis la catégorie `Languages & Frameworks` et enfin `Android SDK`. Ça affiche la grande fenêtre ci-dessous.

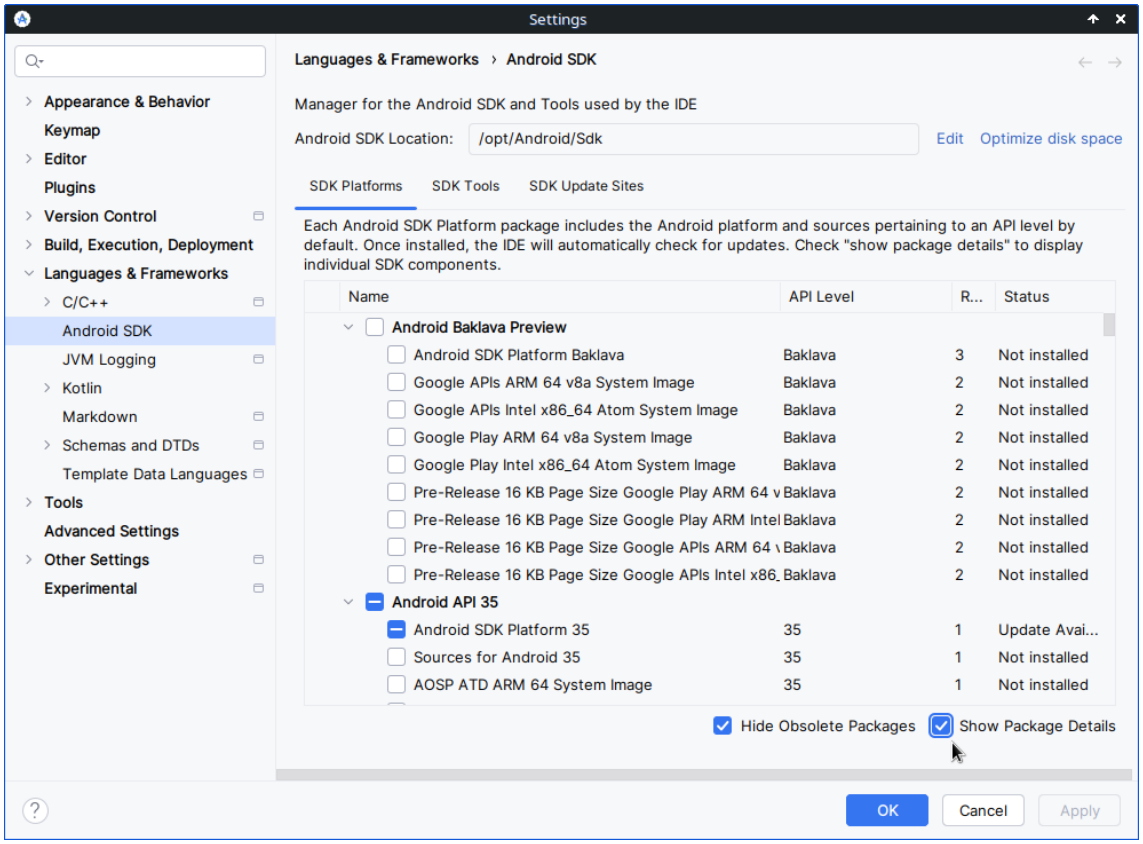

Figure 17: SDK Manager

Il y a une autre manière de l'atteindre. Il faut cliquer le bouton en forme de roue dentée en haut à droite de la fenêtre principale, puis choisir `SDK Manager` dans le menu qui apparaît.

Dans la fenêtre, on voit les éléments du SDK qui sont disponibles, et s'ils sont cochés c'est qu'ils sont installés. On voit que la prochaine version du SDK est proche de la publication. Il n'est pas recommandé d'installer des versions *preview* car elles sont instables. Elles permettent aux développeurs les plus avancés de tester les nouveautés.

- Cochez `Show Package Details` en bas à droite pour voir tous les éléments installables : différentes machines virtuelles, avec ou sans Google Maps, etc.

Voyez que le SDK Platform API 35 est installé. Ce sont les outils de compilation avec toutes les librairies Java et Kotlin pour construire les applications. À l'inverse, il n'est pas recommandé d'installer des versions plus anciennes, parce qu'elles contiennent des bugs ou pas toutes les fonctionnalités.

Normalement, vous ne devez pas modifier le SDK, parce qu'il est mis en commun avec tous les étudiants qui utiliseront ce PC. Si vous cassez le SDK, vous cassez l'installation et plus personne ne pourra travailler avec ce PC tant qu'il ne sera pas réinstallé. Votre nom apparaîtra dans les fichiers modifiés ou cassés... attendez-vous à une entrevue avec les ingénieurs système de l'IUT.

Vous allez télécharger (ou pas) une image pour AVD. On verra plus bas ce que c'est.

1. Descendez jusqu'à `Android 7.0 ("Nougat")`
2. Cochez `Intel x86_64 Atom System Image` s'il n'est pas déjà coché – c'est le cas si d'autres étudiants sont passés avant vous et donc vous n'avez rien à faire.

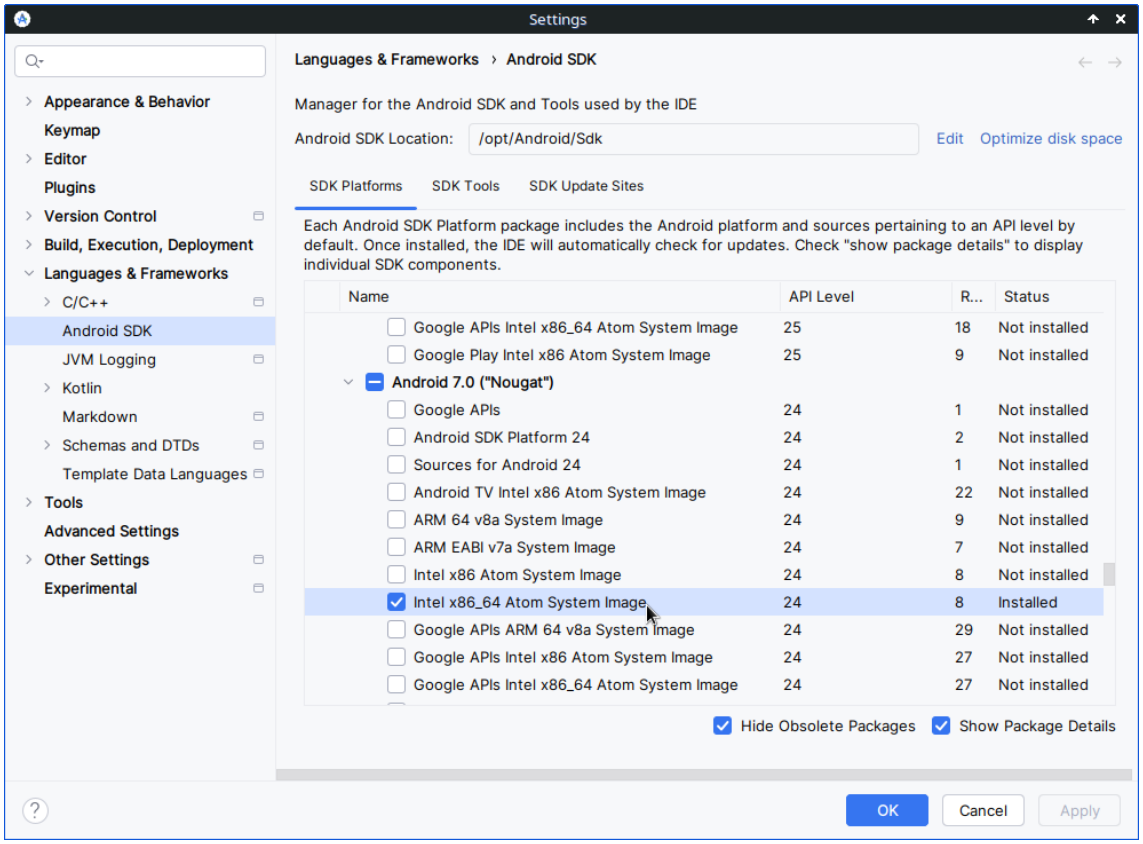

Figure 18: Choisir une image API24

Attention, ce n'est pas `Intel x86 Atom System Image` ou pire `ARM 64 v8a System Image`. Il faut impérativement choisir l'image qui correspond le mieux à notre ordinateur. Le PC devant vous contient un Intel i5 64 bits, compatible avec les ATOM?. Ce sont des processeurs qui ont un jeu d'instruction réduit par rapport à la plateforme intel64 et amd64, pour nécessiter moins de transistors et consommer moins d'énergie. Voir Architecture ATOM .

C'est l'API level 24. C'est une image système, c'est à dire une copie de toute la mémoire flash d'un vrai téléphone. On la choisit exprès ancienne (l'API 24 date de 2016) parce qu'elle est petite (3Go), et qu'elle pourra quand même faire fonctionner toutes nos applications.

Il ne faut pas confondre le niveau d'API des images système (`Intel x86_64 System Image`) avec le niveau d'API servant à créer les applications (`Android SDK Platform` *nn*). Le premier définit quels smartphones virtuels on pourra créer. Le second définit les fonctionnalités qui seront

utilisables en programmation. Il est conseillé d'avoir toujours le SDK le plus récent (mais pas en *preview*) et on peut choisir n'importe quelle image système.

3. Cliquez sur le bouton `Apply`. Il va télécharger l'image en question.

L'onglet `SDK Tools` montre les outils installés, par exemple `Android Emulator`, `Android SDK Tools`. Ne vous inquiétez pas de ne rien y comprendre au début.

Mémorisez l'emplacement du SDK pour l'exercice suivant : `/opt/Android/Sdk`.

4. Fermez la fenêtre avec le bouton `Ok`.

4.2. Contenu du `SDK`

Avec le navigateur de fichiers du système, ouvrez le dossier du SDK, `/opt/Android/Sdk` et regardez ce qu'il y a dedans :

- `platform-tools` qui contient quelques programmes utiles comme `adb`. Ces programmes sont des commandes qui sont normalement disponibles dans une fenêtre appelée *Terminal* en bas d'Android Studio. En principe, vous devriez pouvoir taper `adb` dans ce shell et ça vous affiche l'aide de cette commande. On va s'en servir un peu plus bas.
- `system-images` qui contient les images de la mémoire de tablettes virtuelles. Juste pour comprendre pourquoi il n'y a que ça, regardez la taille des fichiers `.img`. Ce sont des images de disques virtuels à lancer avec qemu. qemu est un simulateur de machines virtuelles comme VirtualBox ou VMware, mais en ligne de commande.

5. Android Virtual Device : `AVD`

On revient encore dans Android Studio, pour lancer le gestionnaire d'AVD en cliquant sur le bouton entouré en rouge.

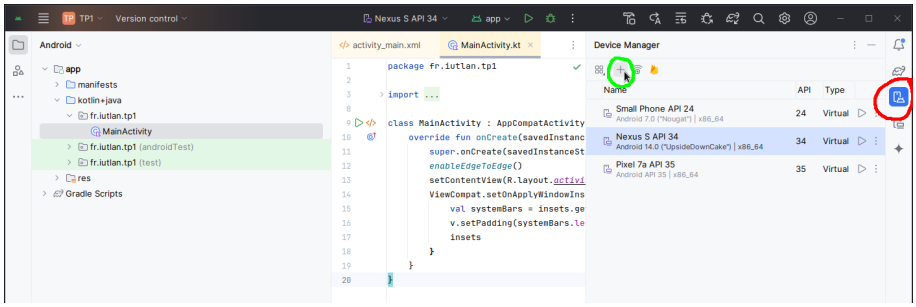

Figure 19: Gestionnaire d'AVD

Normalement, vous devez déjà trouver deux AVD existants. Malheureusement, ils sont trop volumineux pour fonctionner à l'IUT. Il n'est pas du tout recommandé de les lancer, ils vont manger toute la mémoire et ralentir terriblement votre PC.

5.1. Création d'un AVD en API 24

Commencez à créer un smartphone virtuel, avec le bouton `+` de la barre d'outils du Manager, entouré en vert dans l'image précédente. Ensuite, vous avez beaucoup de choix possibles. Le

principe est de ne pas créer un smartphone virtuel trop riche. Si le smartphone virtuel est trop puissant, trop gros (RAM et taille d'écran), il va occuper énormément de place en mémoire et sur le disque. Vous allez vous retrouver à court.

Un autre point important est le CPU du smartphone virtuel. Les vrais smartphones fonctionnent en général avec un processeur ARM. Ça désigne un modèle de programmation (registres et instructions) spécifiques qui doit être simulé sur votre ordinateur. Le mieux est de choisir un AVD ayant un processeur de type amd64 ou atom64 qui n'aura pas besoin d'être simulé, comme avec Docker.

Voici ce qu'il faut faire à l'IUT.

Une solution prudente consiste donc à choisir l'AVD le moins rutilant. Par exemple, choisissez le type `Legacy`, modèle `Nexus S 4,0" 480x800` en *hdpi*.

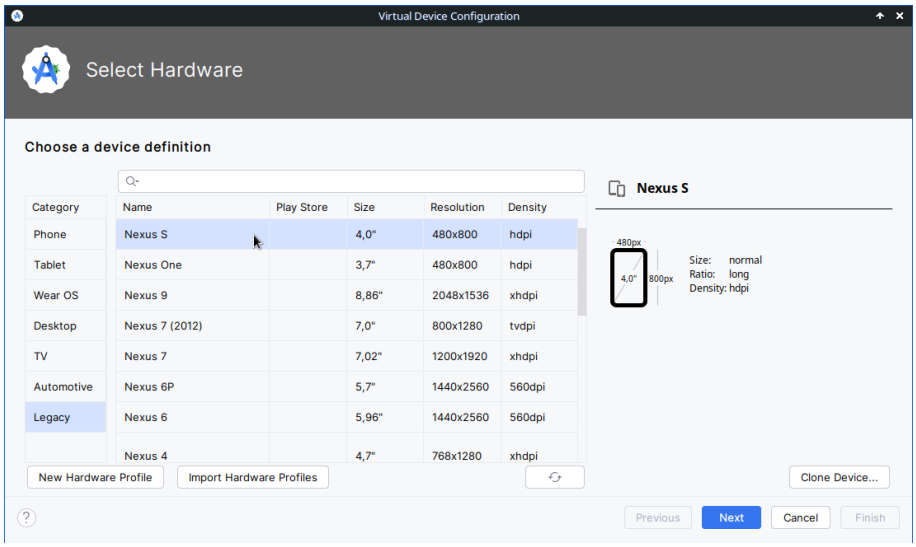

Figure 20: Choix du modèle

Dans l'écran suivant, changez d'onglet pour `x86 images` et choisissez l'`API 24 Nougat`, celle qu'on a installé précédemment.

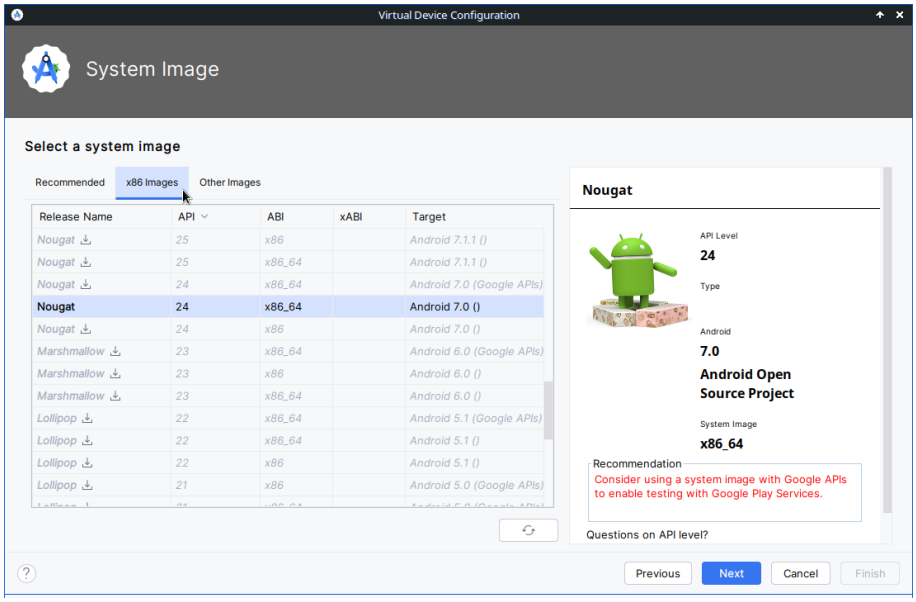

Figure 21: Choix de l'image

Dans le dernier écran, activez `Show Advanced Settings` en bas puis changez le nombre de cœurs, seulement 2 ou 3, la taille de la RAM : au lieu de 343 Mo, mettez 512 Mo, 64Mo pour le `VM Heap`, cochez `No SDCard` et décochez `Enable Device Frame`. Validez la création de ce smartphone virtuel.

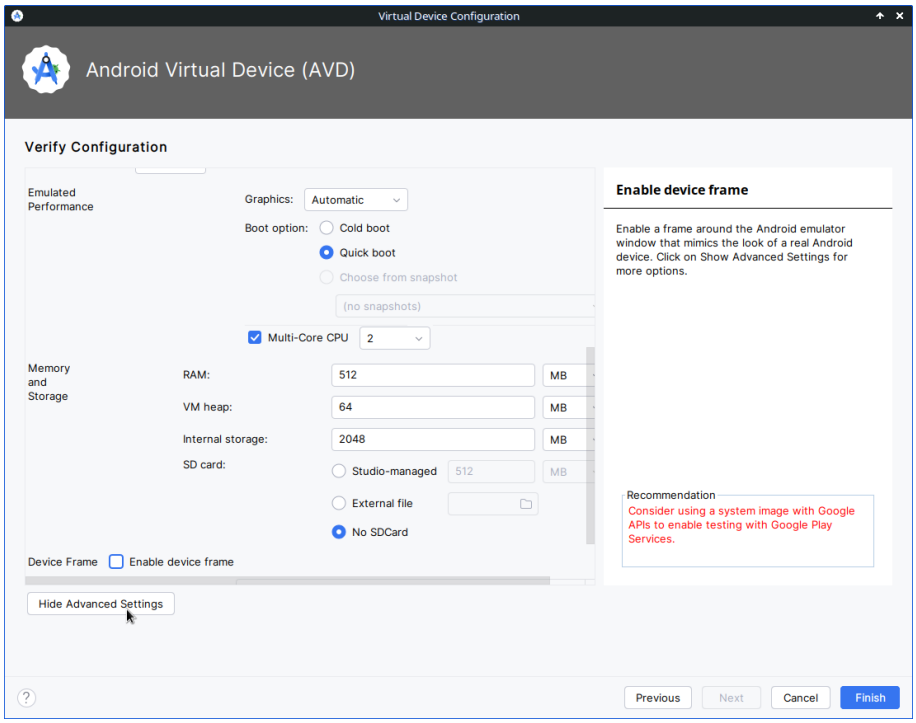

Figure 22: Configuration avancée

La création d'un AVD prend un peu de temps. Il faut générer l'image de la mémoire virtuelle, quelques Go. Une fois que c'est fini, l'AVD apparaît dans la liste et vous pouvez le lancer en cliquant sur le triangle à sa droite.

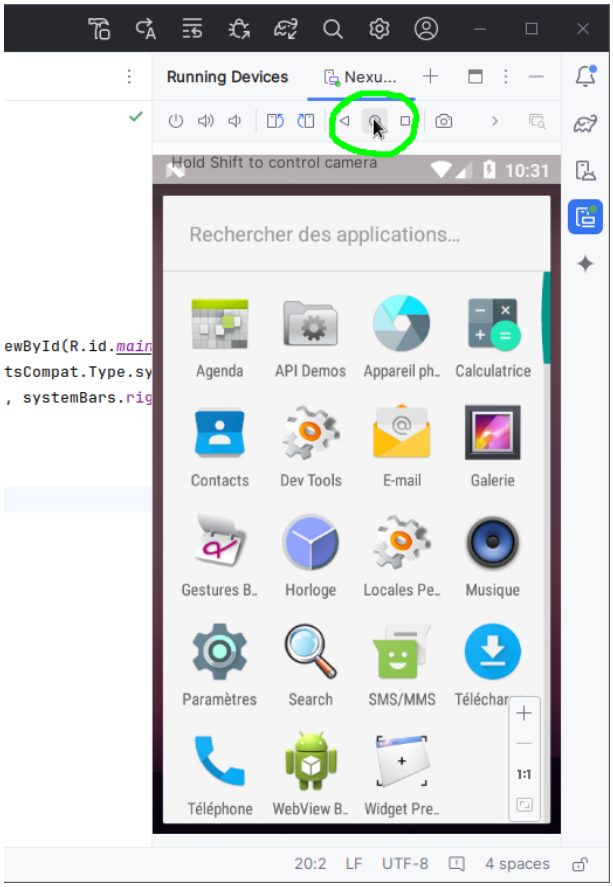

Figure 23: AVD en fonctionnement

Les trois boutons triangle, cercle et carré se trouvent, entourés en vert, dans la barre d'outils au dessus de l'AVD. Il y a un bouton pour mettre en horizontal.

Modifiez les préférences de cette tablette : ouvrez l'application `Settings` (c'est un vieux téléphone à cause de son API).

- Mettez-la en langue française : `Language & Input`, `Language`, ajoutez `Français (France)` et mettez-le en premier dans la liste (c'est selon la variante d'Android que vous avez choisie),
- Enlevez le verrouillage de l'écran en cas d'inaction : `Sécurité`, `Verrouillage de l'écran`, `Aucun`.

5.2. Exécution de l'application

Le lancement de l'application est simple. Il suffit de cliquer sur le bouton triangulaire vert dans la barre d'outils (celui qui est entouré en rouge ci-dessous). L'application `app` sera lancée sur l'AVD qui est sélectionné dans la barre d'outils.

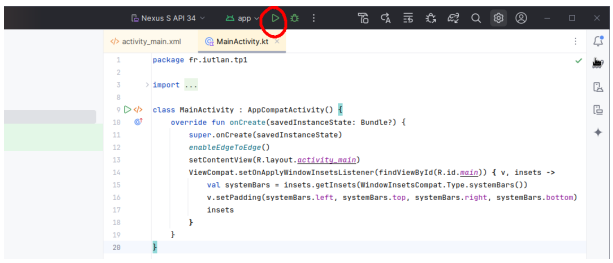

Figure 24: Bouton de lancement

La première compilation prend énormément de temps, car il y a de nombreuses dépendances à télécharger. Vous pouvez voir l'avancement dans l'onglet `Build`, l'espèce de marteau à gauche en bas.

5.3. Android Debug Bridge : `ADB`

Pour finir, on va communiquer avec cette tablette virtuelle.

2. Ensuite des expérimentations :
   b. Démarrez un AVD
   c. Ouvrez la fenêtre `Terminal` à gauche en bas d'Android Studio, puis essayez les commandes suivantes :

- `adb kill-server` : arrête la connexion avec l'AVD. Bizarrement, il peut être nécessaire de faire ça suivi de la commande suivante.
- `adb start-server` : relance le serveur de connexion.
- `adb devices` : affiche la liste des smartphones connectés : les smartphones réels connnectés avec un câble USB et les AVD virtuels actifs.
- `adb shell` : ouvre un shell dans l'AVD. Dans ce shell, tapez :
  - `ls` : c'est une arborescence Unix, mais on n'a pas beaucoup de droits, on n'est pas *root* alors il y a des erreurs dues aux protections.
  - `cd /sdcard` : c'est l'un des rares dossiers où on peut aller.
  - `mkdir essais` : crée un dossier sur la carte sd virtuelle (vérifiez la création avec `ls`).
  - `exit` pour sortir du shell et revenir dans Linux.
- Le dossier du PC `/usr/share/wallpapers` contient des images d'arrière-plan. Choisissez-en une, par exemple `Lake_Sunset-*.jpg`. Ensuite faites par exemple `adb push /usr/share/wallpapers/Lake_Sunset-*.jpg /sdcard/Pictures` : ça envoie l'image sur l'AVD, dans le dossier correspondant au stockage interne, pas celui nommé SD Card. L'image sera visible dans l'application `Galerie`, mais seulement après avoir redémarré l'AVD, une autre bizarrerie.

Vous voyez la ressemblance de `adb` avec des outils comme `ssh` et `ftp`.

6. Travail à rendre

Avec le navigateur de fichiers, allez dans le dossier de votre projet `~/AndroidStudioProjects/TP1`. Ensuite descendez successivement dans `app` puis `src`. Vous devez y voir le dossier `main` ainsi que `test` et `AndroidTest`. Cliquez droit sur le dossier `main`, celui qui est en rouge dans la figure 16

et choisissez `7z` puis `Compresser...`. Ça doit créer `main.7z` ou `main.zip` contenant tout votre travail.

Vous devrez déposer le fichier `main.7z` ou `main.zip` dans le dépôt Moodle Android à la fin du TP, voir sur la page Moodle R4.A11 Programmation Mobile. Cette semaine, il ne sera pas noté, puisque c'est un TP de mise en route avec sûrement des problèmes en séance, mais c'est pour s'habituer à rendre le travail à chaque TP.