

# Systèmes mobiles

## Laboratoire n°1 : Introduction à *Android*

21.09.2021

### Introduction

Ce laboratoire est constitué de plusieurs manipulations destinées à implémenter une application élémentaire sur un émulateur et/ou sur un smartphone *Android* dans le but de vous familiariser avec l'environnement (IDE et SDK) de développement *Android*.

### Installation de l'environnement de développement



Avant de commencer l'installation et la réalisation de ce laboratoire, nous vous demandons de prendre connaissance des règles et des informations générales concernant les laboratoires.

Vous pourrez ensuite télécharger et suivre les instructions d'installation sur la page du site *Android* : <https://developer.android.com/studio/index.html>.

### Manipulations

#### 1. Ouverture de l'application template

La première étape de ce laboratoire, une fois l'environnement de développement installé, sera d'ouvrir le template d'application à utiliser pour la suite de ce laboratoire. Nous vous fournissons deux versions de ce template, la première avec du code en *Java* et la seconde en *Kotlin*. Nous vous conseillons de faire immédiatement l'effort de partir sur le langage *Kotlin*, celui-ci va certainement être, à long terme, le seul langage officiellement supporté sur *Android* ; toutefois si vous vous sentez plus à l'aise avec *Java* vous pouvez sans autre réaliser les laboratoires en *Java*. Le but du cours de systèmes mobiles n'est pas d'apprendre un nouveau langage de programmation. Les concepts importants que nous allons aborder durant ce cours ne sont que très peu (et la plupart du temps pas) liés au langage utilisé.

Une fois le template ouvert dans *Android Studio*, l'IDE va prendre quelques minutes pour configurer le projet et télécharger les dépendances. Vous pourrez ensuite créer un émulateur  et ensuite lancer l'application sur celui-ci . Cette application ne sera pas forcément très pertinente du point de vue de la logique de l'interface utilisateur, elle est uniquement destinée à vous familiariser avec certains aspects élémentaires du développement sur *Android* (spécialement la logique des layouts, des ressources et des activités).

Pour chacune de ces manipulations, vous effectuerez les adaptations nécessaires à partir de l'application fournie et vous documenterez et expliquerez ces adaptations dans votre rapport. Vos explications doivent être suffisamment étoffées pour montrer que vous avez compris le fonctionnement des différents concepts du monde *Android* que nous allons aborder durant ce laboratoire.

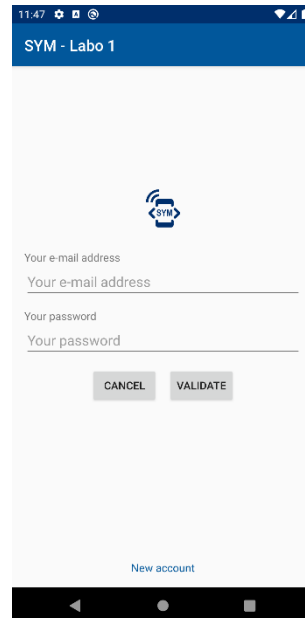


Figure 1 - Première vue de l'application

## 2. Premières constatations

Lorsque l'émulateur aura démarré et que l'application se sera lancée, vous verrez la première page (première activité) de l'application telle que montrée sur la Fig. 1. Celle-ci est composée d'un logo, de deux champs textuels de saisie, de deux boutons ainsi que d'un « lien » cliquable, et elle a pour but de simuler une page de login. En l'état, aucune fonctionnalité n'est implémentée, cela sera votre travail dans les prochaines étapes de ce laboratoire. Avant même de commencer à mettre en œuvre des fonctionnalités, nous pouvons tout de même constater quelques points pouvant être améliorés, ceux-ci sont listés ci-dessous.

### 2.1. Langue de l'interface

Sur *Android*, la langue des applications est automatiquement celle configurée au niveau de l'OS. Notre application reste en anglais quelle que soit la langue du système, votre première tâche sera de mettre place des traductions en français pour les chaînes de caractères utilisées dans cette première activité. De plus vous expliquerez, dans votre rapport, quel est l'intérêt de regrouper les chaînes de caractères dans un fichier XML indépendant à côté des layouts. Vous expliquerez également comment organiser les textes pour obtenir une application multi-langues (français, allemand, italien, langue par défaut : anglais) ? Que se passe-t-il si une traduction est manquante dans la langue par défaut ou dans une langue additionnelle ?

*Remarques : Dans le cas où votre émulateur ou votre téléphone n'est pas déjà configuré en français, veuillez préalablement changer sa langue pour le français (Settings > System > Languages & input > Languages) vous ajouterez le français (Suisse) et le déplacerez en première position, l'interface devrait passer au français, vous ouvrirez ensuite à nouveau notre application.*

## 2.2. Champs textuels de saisie

L'activité comporte deux champs textuels de saisie, le premier va être utilisé pour accueillir une adresse e-mail, le second pour un mot de passe. Nous pouvons constater deux petits problèmes : dans les deux cas le clavier virtuel va activer l'aide à la saisie (correcteur orthographique) ce qui n'est pas réellement adapté à la saisie d'un e-mail et encore moins pour un mot de passe. Et le champ pour le mot de passe ne cache pas la saisie effectuée, nous souhaiterions que seuls des · ou des \* soient affichés. Veuillez adapter la configuration de ces deux éléments d'interface utilisateur (UI) afin qu'ils proposent le comportement décrit ci-dessus. (indice : *inputType*)

## 2.3. Mode paysage

Si vous basculez votre téléphone en mode paysage, l'UI ne sera pas totalement adaptée et certains éléments, comme le logo, ne seront pas affichés correctement (voir Fig. 2a). Votre tâche est ici de spécialiser le layout de cette première activité pour assurer un affichage adapté en mode paysage (voir un exemple sur la Fig.2b).

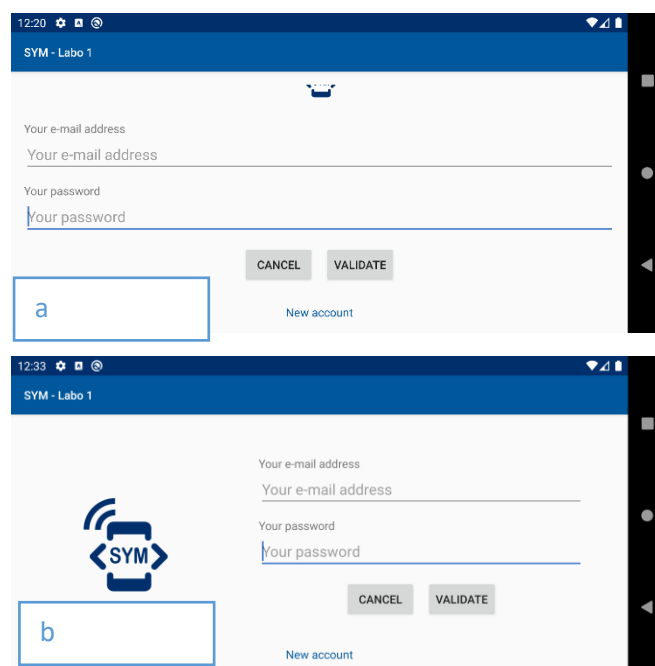


Figure 2 - Première vue de l'application en mode paysage, avec : a) le layout original, b) un layout adapté

## 3. Gestion des événements et mise à jour de l'interface utilisateur

Nous souhaitons avec cette première manipulation illustrer les interactions entre l'utilisateur et l'application en mettant en place la vérification des saisies et les retours à donner à l'utilisateur. Dans le template fourni, un appui le bouton « Cancel » par l'utilisateur aura pour effet de vider les deux champs textuels et d'annuler les éventuels messages d'erreurs ; tandis qu'un appui sur le bouton « Validate » affichera un message sur les champs textuels qui n'ont pas été renseignés.

### 3.1. Vérification du format de l'e-mail

Lors de l'appui sur le bouton « Validate », dans le cas où les deux champs ont été complétés, nous souhaitons dans un premier temps vérifier que l'e-mail saisi soit valide (dans le cadre de ce laboratoire

nous demandons au minimum à vérifier la présence du caractère @), dans le cas contraire vous afficherez un message d'erreur à l'utilisateur sous la forme d'un *Toast*<sup>1</sup>.

### 3.2. Vérification du couple e-mail / mot de passe

Si le format de l'e-mail saisi est valide, nous vérifierons ensuite si le couple (e-mail et mot de passe) saisi appartient à un utilisateur valide. Pour cela nous mettons à votre disposition, dans le code, la liste `credentials` contenant des couples valides. Si le couple saisi n'est pas valide, nous afficherons à l'utilisateur un message d'erreur sous la forme d'une fenêtre de dialogue<sup>2</sup>, si le couple est valide nous ouvrirons une nouvelle activité (manipulation suivante).

*Remarques : Dans le cadre d'une vraie application, jamais nous ne stockerions des mots de passe de cette manière et nous éviterions d'utiliser plusieurs manières différentes d'afficher des messages à l'utilisateur. Le but ici est uniquement de vous faire découvrir et prendre en mains différents outils et widgets du monde Android.*

## 4. Passage à une autre activité

En cas de saisie correcte à l'étape précédente, nous souhaitons qu'une nouvelle activité soit lancée et présentée à l'utilisateur. Nous allons profiter de cette nouvelle activité pour illustrer certains concepts plus avancés.

*Remarques : vous serez amené à définir un layout pour permettre l'affichage des différentes manipulations dans cette nouvelle activité, celui-ci peut rester très simple (alignement vertical) et il n'est pas nécessaire de l'adapter au mode paysage. Nous ne vous demandons pas de forcément produire un template « joli », nous restons bien entendu à disposition si vous souhaitez approfondir le fonctionnement des layouts.*

### 4.1. Création et lancement de la nouvelle activité

La première manipulation va être de définir cette nouvelle activité dans *Android Studio*. Une activité est principalement composée de deux éléments : son code (une classe héritant d'*Activity* ou d'une de ses sous-classes, comme par exemple *AppCompatActivity*) et d'un fichier *XML* définissant son layout, l'activité devra être ensuite déclarée dans le fichier *Manifest* de l'application, sous peine de ne pas pouvoir être lancée. *Android Studio* permet d'automatiser la création d'une nouvelle activité, toutefois selon le type choisi, le layout automatiquement généré sera très complexe. Une fois l'activité ajoutée dans l'IDE, vous mettrez en place le code permettant de la lancer lors du succès du login.

### 4.2. Passage de paramètres à la nouvelle activité

Dans cette nouvelle activité, nous souhaitons afficher l'adresse e-mail utilisée pour le login. Vous allez donc devoir mettre en place le passage d'un paramètre à la nouvelle activité lors de son lancement et, dans celle-ci, la récupération de ce paramètre et son affichage.

### 4.3. Permissions simples

Nous souhaitons à présent afficher une photo censée représenter l'utilisateur qui vient de se connecter, pour cela nous allons télécharger une image depuis Internet. L'accès à Internet n'est pas possible par défaut pour une application *Android*, celle-ci doit avoir la permission de le faire. Vous allez faire les modifications nécessaires dans le fichier *Manifest* pour autoriser votre application à accéder

---

<sup>1</sup> Documentation Android - les Toasts : <https://developer.android.com/guide/topics/ui/notifiers/toasts>

<sup>2</sup> Documentation Android - les Dialogues : <https://developer.android.com/guide/topics/ui/dialogs>

à Internet, ensuite vous utiliserez le code fourni permettant le téléchargement et l’affichage d’une image dans une *ImageView*:

```
ImageDownloader(connected_image, "https://thispersondoesnotexist.com/image").show()
```

Dans cet extrait de code *connected\_image* est une variable de type *ImageView* représentant l’élément de l’UI dans lequel l’image sera affichée, et le second paramètre est l’url vers l’image à télécharger.

*Remarques : Les opérations IO feront l’objet du prochain laboratoire et d’une série d’exercice, cette fois-ci nous vous fournissons le code nécessaire. Nous verrons également dans les prochains laboratoires que l’accès à certaines fonctionnalités nécessite une acceptation active des permissions par l’utilisateur (Accès à la camera, à la géolocalisation, etc.) durant l’exécution.*

## 5. Navigation entre les activités

Nous souhaitons à présent ajouter une fonctionnalité permettant de créer de nouveaux comptes utilisateurs. Cette fonctionnalité prendra la forme d’une nouvelle activité, permettant de saisir et de valider une adresse e-mail et un mot de passe, ceux-ci seront transmis à l’activité principale lors du retour sur celle-ci.

### 5.1. Création et lancement de la nouvelle activité

Comme pour le point 4.1, vous allez commencer par définir une nouvelle activité et un layout associé. Cette activité, et son layout, sont très semblables à l’activité principale il est donc certainement possible de s’en inspirer ou même de réutiliser certains éléments. Cette activité permettra de saisir une adresse e-mail et un mot de passe, elle effectuera une validation de ceux-ci (sur la forme uniquement, comme pour le point 3.1, nous n’allons pas faire de vérifications additionnelles), et retournera le compte (couple e-mail / password) à l’activité principale qui l’ajoutera alors à la liste *credentials*.

Il existe deux façon de réaliser cette étape, soit en se basant sur le mécanisme d’origine d’Android<sup>3</sup>, soit en se basant sur la nouvelle méthode proposée par la librairie AndroidX<sup>4</sup>. Nous nous trouvons ici dans un cas assez particulier, avec une façon de faire existant depuis les débuts d’Android qui se retrouve remplacée sans pour autant être officiellement dépréciée. Vous êtes libre d’utiliser la méthode de votre choix dans ce laboratoire, toutefois vous discuterez dans votre rapport des avantages / inconvénients des deux méthodes.

### 5.2. Affichage d’une image

Sur l’activité principale, une image vectorielle est affichée sur le haut de l’activité. Nous souhaitons mettre une autre image sur l’activité d’inscription. Pour cela vous pouvez choisir une image sur Internet (*PNG* ou *JPEG*), la télécharger et l’inclure dans les ressources de l’application afin de pouvoir l’ajouter au layout. Dans quel(s) dossier(s) devons-nous ajouter cette image ? Veuillez décrire brièvement la logique derrière la gestion des ressources de type « image matricielle » sur *Android*. Quel intérêt voyez-vous donc à utiliser une image vectorielle ? Est-ce possible pour tout type d’images (logos, icônes, photos, etc.) ?

### 5.3. Factorisation du code

Vous remarquerez les très nombreuses similarités entre ces deux activités : le code de vérification et certains éléments de l’interface graphique. Nous souhaitons factoriser ce code pour en faciliter sa maintenance. Veuillez mettre en place une factorisation « simple » du code de ces deux activités (en

<sup>3</sup> Résultat d’une activité 1 : <https://developer.android.com/reference/android/app/Activity#StartingActivities>

<sup>4</sup> Résultat d’une activité 2 : <https://developer.android.com/training/basics/intents/result>

*Kotlin* ou *Java*), vous commenterez votre approche dans votre rapport. Vous discuterez également des possibilités de factoriser des layouts, sans forcément le réaliser.

#### 5.4. Cycle de vie

Dans vos trois activités, veuillez implémenter des sorties dans les logs pour les méthodes `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, etc... qui marquent le cycle de vie d'une application *Android*. Décrivez brièvement à quelles occasions ces méthodes sont invoquées. Vous expliquerez aussi l'enchaînement de ces appels lorsque l'on passe d'une activité à une autre. Comme pour le point 5.3, il est certainement possible de factoriser votre code pour réaliser cette étape.

## Durée

- 6 périodes
- A rendre le dimanche **10.10.2021 à 23h55**, au plus tard.

## Donnée

Un template d'application est disponible sur la page *Cyberlearn* du cours :

<https://cyberlearn.hes-so.ch/course/view.php?id=19738>

## Rendu/Evaluation

Pour rendre votre code, nous vous demandons de bien vouloir zipper votre projet *Android Studio*, vous veillerez à bien supprimer les dossiers `build` (cf. `git clean -fdX`) pour limiter la taille du rendu. En plus, vous remettrez un document **pdf** comportant au minimum les réponses aux questions posées. Merci de rendre votre travail sur *CyberLearn* dans un zip unique. N'oubliez pas d'indiquer vos noms dans le code, avec vos réponses et de commenter vos solutions.

**Bonne chance !**