Développement Android
Laboratoire n°3

Interactions avec l'utilisateur
Approche MVC

22.03.2022

Introduction

Ce laboratoire est constitué de plusieurs manipulations et questions théoriques destinées à appréhender la mise en place des différents composants *Android* de base pour la réalisation d'une *Activité* proposant un formulaire permettant d'éditer les informations d'une personne. Certaines parties de la manipulation 4 nécessite l'utilisation d'un smartphone physique.

Manipulations

1. Mise en place

Pour ce laboratoire vous aller partir d'une application par défaut « empty activity » à laquelle vous intégrerez les ressources que nous vous fournissons, à savoir :

- Le fichier *Person.kt* qui contient la classe abstraite Person ainsi que ses deux sous-classes *Worker* et *Sudent*. Nous vous avons également mis à disposition deux instances statiques exemples : *exampleWorker* et *exampleStudent* contenant des données illustratives. Le diagramme *UML* correspondant est disponible dans les slides de présentation ;
- Le fichier *String.xml* contenant la définition de textes et de tableaux de valeurs utilisables dans ce laboratoire. Il s'agit d'une proposition, sentez-vous libre de l'adapter ;
- L'image vectorielle *cake.xml* représentant un gâteau d'anniversaire ;
- L'image matricielle *placeholder_selfie.jpg* contenant la silhouette d'un selfie.

2. Développement de la vue

Nous vous demandons à présent de réaliser l'interface graphique de notre application. Il s'agit de concevoir le fichier *layout* intégrant et mettant en page tous les textes, les widgets et les différents éléments de notre interface graphique. Vous trouverez sur la Fig. 1 une proposition de mise en page,

vous avez le choix entre l'utilisation de *ConstraintLayout* ou de *RelativeLayout* et *LinearLayout*. Vous veillerez à ne pas imbriquer un trop grand nombre de layout pour ne pas péjorer les performances.

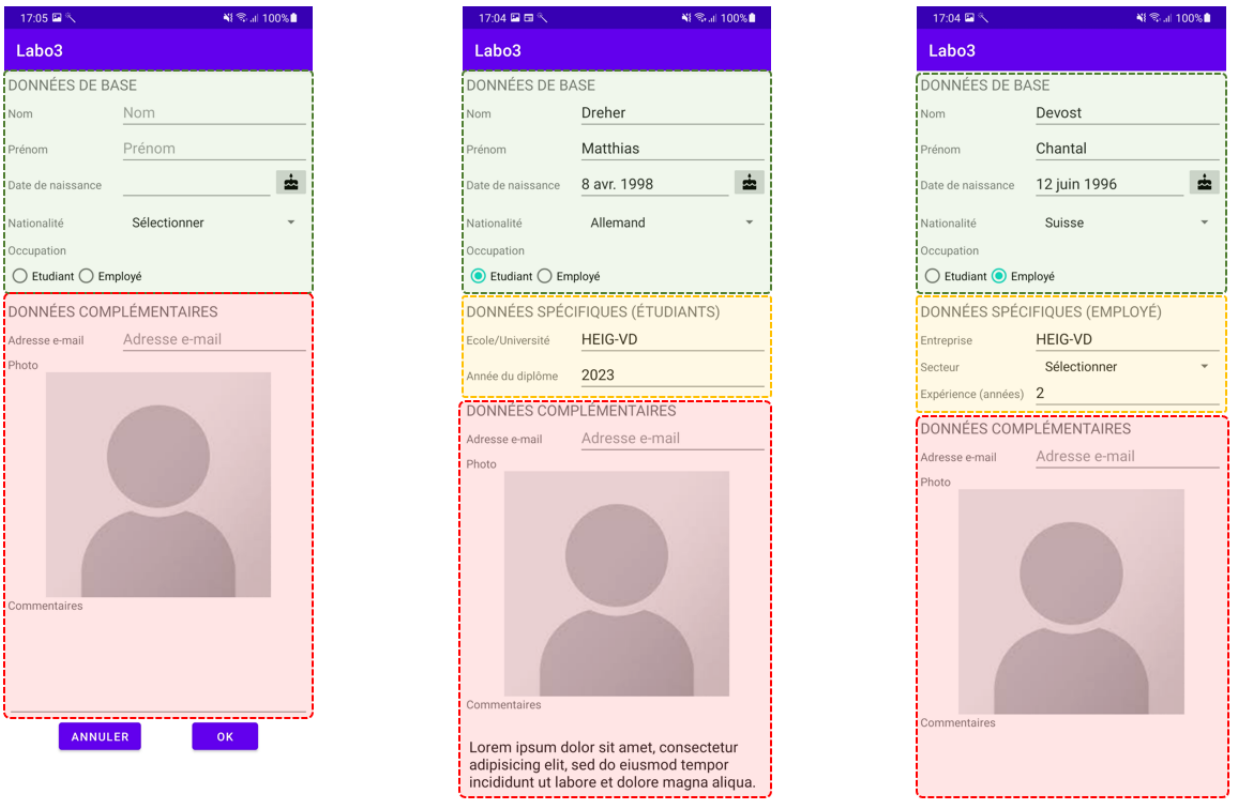

Figure 1 - Exemple de l'interface graphique permettant d'éditer une Personne. Les zones en couleur servent à mettre en évidence les différentes parties du formulaire (en vert : données de base d'une personne, en orange : particularités d'un employé ou d'un étudiant, en rouge les champs complémentaires en commun).

Votre interface devra permettre d'éditer des *Person*, soit des *Worker* ou des *Student*, en mettant à disposition les widgets adaptés à la saisie des différents champs de ces classes, en particulier :

- Des *EditText*, ou similaire, pour la saisie des informations textuelles ou numériques ;
- Des *Spinner* pour le choix d'une valeur parmi une liste (nationalité et secteur d'activité d'un employé) ;
- Des *RadioButton* pour le choix du type de la Person, un *Worker* ou un *Student* ;
- La saisie de la date d'anniversaire se fera à l'aide d'un *DatePicker*, un dialogue qui s'ouvrira lors de l'appui sur un bouton à côté du champ de saisie ou sur le champ de saisie (qui ne doit pas pouvoir être directement édité par l'utilisateur avec le clavier) ;
- La prise du selfie qui s'effectuera en ouvrant l'application « appareil photo » du smartphone (cf. Manipulation 4).

Votre interface devra s'adapter à la saisie d'un type ou de l'autre en fonction du choix de l'utilisateur, les zones de saisie dédiées à des informations concernant uniquement un étudiant ne devront pas être affichées lors de l'édition d'un employé, et vice-versa. Également vous mettrez deux boutons à disposition permettant de : 1) vider le formulaire, 2) valider le formulaire.

3. Le contrôleur

Il s'agit de l'activité qui accueillera la vue et implémentera les différentes fonctionnalités devant être offertes :

- L'initialisation de la vue en fonction d'un objet existant du modèle, par exemple *exampleWorker* ou *exampleStudent* ;
- De créer une instance d'un *Student* ou d'un *Worker* lors de l'appui sur le bouton de validation du formulaire avec les données saisies dans celui-ci et l'affichage de l'instance créée dans les logs (méthode *toString()*) ;
- Supprimer le contenu de tous les éléments de la GUI lors de l'appui sur le bouton correspondant.

4. Fonctionnalité avancée – selfies

Nous souhaitons offrir la possibilité d'associer un selfie aux données de la personne. Pour cela nous vous demandons de mettre en œuvre la prise d'une photo au travers d'un appel à l'application « appareil photo » du smartphone avec l'API basée sur un *Intent* implicite. Dans l'*Intent* reçu en retour, lors du succès de la prise de vue, se trouve uniquement une miniature de la photo. L'accès à l'image en haute qualité nécessite la mise en place d'un *FileProvider* permettant à notre application de fournir dans l'*Intent* l'*Uri* vers le fichier dans lequel l'appareil photo écrira l'image.

Dans cette application, nous ne souhaitons pas que les selfies soient présents dans la galerie du téléphone, nous allons donc créer le fichier qui accueillera l'image dans l'espace privé appartenant à notre application. Une fois la prise d'image effectuée, vous l'afficherez dans le formulaire, à la place du placeholder.

Suivant la position du téléphone lors de la prise de vue, l'image peut ne pas être correctement orientée, veuillez effectuer la rotation nécessaire avant de l'afficher. Attention à ne pas effectuer trop de travail sur le thread principal !

Hint : *ExifInterface*

5. Questions complémentaires

Veuillez répondre aux 5 questions suivantes. Pour chacune d'entre elles vous développerez votre réponse et l'illustrerez par des extraits de code.

5.1 Pour le champ *remark*, destiné à accueillir un texte pouvant être plus long qu'une seule ligne, quelle configuration particulière faut-il faire dans le fichier XML pour que son comportement soit correct ? Nous pensons notamment à la possibilité de faire des retours à la ligne, d'activer le correcteur orthographique et de permettre au champ de prendre la taille nécessaire.

5.2 Pour afficher la date sélectionnée via le *DatePicker* nous pouvons utiliser un *DateFormat* permettant par exemple d'afficher *12 juin 1996* à partir d'une instance de *Date*. Le formatage des dates peut être relativement différent en fonction des langues, outre la traduction des mois, par exemple la même date en anglais britannique serait *12th June 1996* et en anglais américain *June 12, 1996*. Comment peut-on gérer cela au mieux ?

a. Si vous avez utilisé le *DatePickerDialog*[1] du SDK. En cas de rotation de l'écran du smartphone lorsque le dialogue est ouvert, une exception *android.view.WindowLeaked* sera présente dans les logs, à quoi est-elle dûe ?
b. Si vous avez utilisé le *MaterialDatePicker*[2] de la librairie *Material*. Est-il possible de limiter les dates sélectionnables dans le dialogue, en particulier pour une date de naissance il est peu probable d'avoir une personne née il y a plus de 110 ans ou à une date dans le futur. Comment pouvons-nous mettre cela en place ?

5.3 Lors du remplissage des champs textuels, vous pouvez constater que le bouton « suivant » présent sur le clavier virtuel permet de sauter automatiquement au prochain champ à saisir, cf. Fig. 2. Est-ce possible de spécifier son propre ordre de remplissage du questionnaire ?
Arrivé sur le dernier champ, est-il possible de faire en sorte que ce bouton soit lié au bouton de validation du questionnaire ?
Hint : Le champ *remark*, multilignes, peut provoquer des effets de bords en fonction du clavier virtuel utilisé. Vous pouvez l'échanger avec le champ *e-mail* pour faciliter vos recherches concernant la réponse à cette question.

5.4 Pour la prise de vue du selfie, nous souhaiterions que l'application « appareil photo » démarre directement sur la caméra frontale. L'API supporte-t'elle officiellement un tel paramètre ? Est-il tout de même possible de le réaliser, si oui comment ?

5.5 Pour les deux *Spinners* (nationalité et secteur d'activité), comment peut-on faire en sorte que le premier choix corresponde au choix *null*, affichant par exemple « Sélectionner » ? Comment peut-on gérer cette valeur pour ne pas qu'elle soit confondue avec une réponse ?

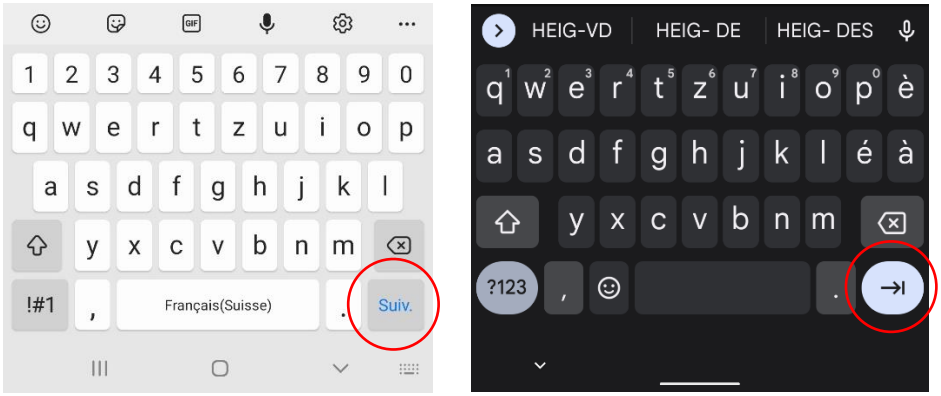

Figure 2 - Claviers virtuels (Samsung à gauche, et Google à droite) lors du remplissage de champs textuels. Les cercles rouges identifient un bouton permettant de passer automatiquement au prochain champ sans devoir fermer le clavier.

[1] https://developer.android.com/reference/android/app/DatePickerDialog
[2] https://github.com/material-components/material-components-android/blob/master/docs/components/DatePicker.md

Durée / Evaluation

- 4 périodes
- A rendre le dimanche **03.04.2022** à **23h55** au plus tard.
- Pour rendre votre code, nous vous demandons de bien vouloir zipper votre projet Android Studio en veillant à bien supprimer les dossiers build (à la racine et dans app/) pour limiter la taille du rendu. Vous remettrez également 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, sur vos réponses et de commenter vos solutions.