Sophie Bousquet
Liu François
L3-A

# Rapport Reminder

## Sujet :

Notre projet consiste en l'implémentation d'un reminder, c'est-à-dire une application qui permet de créer des rappels pour les prochaines 24h.

## Fonctionnement :

L'utilisateur doit appuyer sur le bouton plus afin de créer un nouveau rappel.

L'utilisateur devra par la suite définir un titre pour son rappel ainsi qu'une heure via l'horloge qui est un TimePicker.

Pour valider le rappel il devra appuyer sur le bouton qui ressemble à un réveil, puis l'alarme sera créée. Quand l'horaire indiqué par l'utilisateur sera atteint une notification apparaitra, le téléphone se mettra à vibrer et le rappel sera supprimé sur l'application.

De plus on a un bouton poubelle qui permet de supprimer un rappel.

De même, on a un bouton plus qui permet d'ajouter de nouveau rappel.

On a également un bouton quitter qui permet de fermer l'application.

Cependant, pour que les rappels fonctionnent, il ne faut pas fermer l'application, il faut au moins la laisser en arrière-plan.

## Contenu :

Notre application ne contient qu'une seule interface et fonctionne uniquement en mode portrait.

Elle utilise un seul capteur, les vibrations et les notifications.

Notre application contient 3 fichiers importants, MainActivity.kt, activity_main.xml et item1.xml.

## Activity_main.xml :

Ce fichier contient l'interface graphique de l'application. On peut y retrouver 2 ImageButton qui servent à ajouter un rappel ou à quitter l'application, ainsi qu'un RecyclerView qui contient la liste des rappels. Il y a également un TextView qui correspond au titre de l'application, ainsi qu'une ImageView qui correspond à la bordure de notre liste de rappel.

Sophie Bousquet
Liu François
L3-A

## Item_1.xml :

Ce fichier correspond aux items que chaque rappel aura, c'est à dire un EditText qui correspond au titre du rappel, un TimePicker qui correspond à l'horaire de notre rappel, et 2 ImageButton : le bouton qui programme le rappel et le bouton qui supprime un rappel.

## MainActivity.kt :

Ce fichier correspond au corps de l'application.

Tout d'abord nous avons la class MainActivity. Dans cette class, nous avons initialisé différentes variables privées : adapter qui sert à créer une instance de la class ItemAdapter, itemList qui contient les données pour chaque élément de la liste, et imageList qui contient les ressources d'images qui représentent les images par défaut pour les alarmes.

Nous avons également CHANNEL_ID qui correspond à l'identifiant du canal de notification, REQUEST_NOTIFICATION_PERMISSION qui correspond au code de demande d'autorisation pour les notifications et notificationId qui sert pour les noms par défaut des rappels.

Ensuite, nous avons la fonction onCreate qui est appelée lors de la création de l'application.

Dans cette fonction, on appelle 2 fonctions : createGridList et touch. Puis on demande l'autorisation pour pouvoir utiliser les notifications.

Enfin, on appelle la fonction createNotificationChannel.

Par la suite, nous avons la fonction touch qui vérifie si on clique sur le bouton add ou le bouton appStop. Si le bouton add est pressé, alors la fonction addItem sera lancée. Si le bouton appStop est pressé alors la fonction finish est lancée.

Ensuite, nous avons la fonction createGridList qui permet d'instancier la liste et d'adapter les items.

En premier lieu on crée la variable myList qui appelle la fonction createList qui sert à créer une liste d'alarmes. Puis, on crée et configure l'adaptateur pour le RecyclerView.

On créer la variable DirectionLayout qui permet de configurer le sens de défilement et la forme de la liste. On a par la suite itemTouchHelper qui permet à l'utilisateur d'agencer les items dans la liste.

Puis on attache ItemTouchHelper au RecyclerView ce qui permet d'activer la fonctionnalite de glisser-déposer des éléments dans la liste.

Sophie Bousquet
Liu François
L3-A

Ensuite on a la fonction createList qui remplit la liste itemList avec des objets de type DataItem.

Par la suite, nous avons la fonction addItem qui sert à rajouter des objets daéns la liste de rappel.

Cette fonction crée un nouvel objet DataItem avec l'image newImage.

Puis elle va ajouter ce nouvel objet à la liste itemList et avertir par la suite l'adaptateur du changement.

La fonction onItemClick est un callback qui est appelé quand un item de la liste est appuyé et qui met à jour l'image de l'ImageView avec l'image de l'élément sélectionné.

La fonction onItemDelete est aussi un callback qui est appelé lorsqu'un élément de la liste doit être supprimé. Cette fonction va supprimer l'élément de la liste à la position donnée et prévenir l'adaptateur que l'élément doit être supprimer.

Ensuite, nous avons la fonction showNotification qui permet d'afficher une notification sur l'appareil.

Cette fonction va en premier lieu récupérer le texte de EditText qui correspond au titre de notre notification.

On va ensuite créer l'intent pour la notification qui permet de lancer l'application lorsqu'on clique sur la notification.

Puis on va créer le builder pour la notification ce qui correspond aux caractéristiques de la notification c'est-à-dire l'icone de la notification, le titre de la notification (ici elle correspond à la variable valeurEditText), d'une priorité (ici elle est normale), de la variable PendingIntent et de savoir si la notification disparait quand on clique dessus.

On va ensuite créer les vibrations qui seront lancées en même temps que la notification.

Enfin, on va afficher la notification. En premier lieu, on va vérifier si la permission est accordée et la demander si elle n'est pas accordée. Puis, on affiche la notification avec un identifiant que l'on va par la suite incrémenter pour la prochaine notification que l'on va créer.

On a la fonction requestNotificationPermission qui permet de demander l'autorisation des notifications.

Lorsque cette fonction est appelée, elle déclenche l'affichage d'une boîte de dialogue à l'utilisateur, en lui demandant s'il souhaite accorder ou refuser la permission demandée.

Si l'utilisateur accorde la permission, le système appelle la méthode onRequestPermissionsResult de l'activité ou du fragment pour informer de la réponse de l'utilisateur afin de pouvoir afficher ou non les notifications.

Sophie Bousquet
Liu François
L3-A

On a la fonction createNotificationChannel qui permet de créer un canal de notification avec des paramètres spécifiques, puis enregistre ce canal auprès du système.

Ici on a défini l'importance des notifications de ce canal comme des notifications normales.

Un canal, pour être créé, a besoin d'un identifiant unique (ici c'est la variable CHANNEL_ID), d'un nom de canal (ici « Reminder »), d'une importance (comme dit plus tôt l'importance est normale) et d'une description qui est optionnelle, mais ici elle correspond à « Alerte ».

Ensuite, on a une nouvelle class DataItem qui définit des données dans les items.

Par la suite on a la class ViewHolder qui permet l'instanciation des composants des items.

Dans cette class on va tout d'abord créer des variables qui correspondent à des éléments du fichier item_1.xml c'est à dire les différentes ImageButton, l'EditText et le TimePicker.

Ensuite nous avons un init. En premier lieu, on va définir ce qu'il se passe lorsqu'on clique sur le bouton poubelle ou sur le bouton alarme.

Si on clique sur le bouton poubelle, la fonction onItemDelete sera appelée, ce qui va supprimer l'item de la liste.

Si on clique sur le bouton alarme, alors on va récupérer l'heure actuelle.

Puis, on va récupérer l'heure et les minutes initialisées dans le TimePicker dans les variables hourOfDay et minute.

Par la suite, on va définir l'heure sélectionnée par l'utilisateur grâce à Calendar dans la variable selectedTime, on va lui assigner l'heure, les minutes, ainsi que les secondes.

Ensuite, grâce à timeInMillis, on va calculer la difference de temps entre l'heure actuelle et l'heure définie par l'utilisateur.

Enfin, on définit ce que l'on doit faire quand le temps est atteint. Ici on appelle la fonction showNotification grâce à un callback vu que la fonction est en dehors de cette class puis grâce à listener on appelle la fonction onItemDelete.

Par la suite on a ViewHolderCallback qui appelle showNotification, qui permet d'utiliser la fonction showNotification même si on est en dehors de la class où la fonction a été définie.

Ensuite, on a la fonction Onclick qui est appelée à chaque fois qu'un élément est cliqué. Pour cela elle va récupérer la position de l'élément, vérifier si la position est valide et ensuite, si elle est valide, on va appeler la fonction onItemClick grâce au listener.

Sophie Bousquet
Liu François
L3-A

Enfin, nous avons OnItemClickListener qui appelle les fonctions onItemClick et onItemDelete. Cela permet d'utiliser ces 2 fonctions en dehors de leur class.

Ensuite, nous avons la class ItemAdapter qui attribut des valeurs à chaque composants des items.

En premier lieu, nous avons la fonction onCreateViewholder qui permet de définir le visuel de l'item et instancier les variables pour chaque objets composants.

En deuxième lieu, on a la fonction getItemCount qui permet de définir le nombre d'items dans la liste.

Enfin, nous avons la fonction onBindViewholder qui permet d'attribuer les données aux composants.

## Problèmes rencontrés :

Tout d'abord nous avons eux un problème sur le linearlayout car on avait repris un ancien exercice et rien ne fonctionnait quand on faisait des modifications, donc on l'a repris de zéro en faisant pas à pas.

Puis nous avons eu des problèmes pour les notifications. Il n'y avait aucune erreur mais aucune notification n'apparaissait. Par la suite on a découvert qu'il suffisait d'autoriser les notifications dans les paramètres de l'application. Puis on a rajouté la boite de dialogue qui nous demandait l'autorisation.

Ensuite, on a eu des problèmes pour relier les notifications et le linearlayout car les 2 étaient dans 2 projets différents.

De plus, le bouton pour supprimer les items dans la liste ne marchait pas non plus.

Pour résoudre ces 2 problèmes, notre professeur nous a donné une autre manière de faire le linearlayout, ce qui nous a permis de plus facilement supprimer les items dans la liste ainsi que d'implémenter les notifications dans le linearlayout.

## Ce qu'il manque :

Il nous manque uniquement que lorsqu'on supprime un élément de la liste, si on lui a déjà programmé une alarme, que cette alarme ne se lance pas.