Programmation des composants mobiles
Utiliser une image de la galerie d'image

1 Choisir une image pour une plante

On suppose que nous avons une activité, que nous appelons ici `AddPlantActivity`, qui permet d'ajouter une nouvelle plante. Dans cette activité on spécifie le nom de la plante et d'autres données (comme la fréquence d'arrosage). Mais ce qui nous intéresse c'est comment sélectionner une image de la plante.
En pratique cela veut dire que `AddPlantActivity` va lancer une autre activité qui permet de sélectionner l'image et au retour de cette nouvelle activité `AddPlantActivity` doit recupérer l'Uri de l'image pour le sauvegarder, sous forme d'un String, dans la base de données.
Il s'avère que, bien que l'Uri obtenu de cette façon permet d'afficher l'image dans `AddPlantActivity` , cette Uri n'est plus valable quand on essaie d'afficher l'image dans une autre activité.
En particulier, il ne sert à rien de sauvegarder cette Uri dans la base de données.
La solution consiste à faire copier l'image dans la mémoire interne de l'application.

2 Lancer une nouvelle activité pour un résultat

Le squelette de code qui lance une nouvelle activité qui permet de sélectionner l'image :

```
class AddPlantActivity : AppCompatActivity() {

      /* propriété : Uri de fichier local */
      var localUri : Uri? = null

     /* la propriété getContent pour enregistrer une méthode callback
      * appelée au retour de la nouvelle activité.
      * Ici cette méthode callback est implementée par une lambda qui
      * qui obtient  l'Uri de l'image comme paramètre.   */
     val getContent = registerForActivityResult(ActivityResultContracts
                                                .GetContent())
       { uri: Uri? ->
          if( uri == null ) return@registerForActivityResult

        /*  à faire :
         *  si uri de l'image différent de null alors copier l'image dans
    la
         *  mémoire interne de l'appareil et mémoriser l'emplacement de l
   'image
         *  copiée dans la base de données.
         */
       }

     override fun onCreate(savedInstanceState: Bundle?) {

    /* Le bouton qui lance la sélection de l'image.
```

```
     * Chez vous il doit être plutôt obtenu grâce au view binding,
     * sans l'appel à findViewById() */
    val selectButton = findViewById<Button>(R.id.select_button)

    /* Installer le listener sur le bouton. Quand l'utilisateur clique
   sur
     * le bouton une nouvelle activité sera lancée et on pourra
     * sélectionner l'image. Au retour de cette cette activité la méthode
     * callback définie ci-dessus  sera appelée.   */

    selectButton.setOnClickListener {
        // passer le mime type de l'image à la méthode launch
        // puisque nous voulons sélectionner une image

        getContent.launch("image/*")
    }
}
```

3 Traitement de l'image

Il nous reste de compléter la définition de la méthode callback.
Chaque fois quand on ajoute une image dans la mémoire de l'application il faut fabriquer un nom de fichier local unique qui stocke l'image. Dans le code ci-dessous le nom de ce fichier local est composé de String `"plante"` auquel on concatène un numéro. Ce numéro est stocké dans `SharedPreferences` et est incrémenté à chaque nouvelle image.

```
val getContent =
        registerForActivityResult(ActivityResultContracts.GetContent()){
   uri: Uri? ->
      /* si uri null rien à faire */
      if( uri == null ) return@registerForActivityResult

      /* inputStream avec l'image de la plante */
      val inputStream = getContentResolver().openInputStream(uri)

      /* fabriquer le nom de fichier local pour stocker l'image */
      val fileNamePrefix = "plante";
      val preferences = getSharedPreferences("numImage", Context.
   MODE_PRIVATE)
      val numImage = preferences.getInt("numImage", 1)
      val fileName = "$fileNamePrefix$numImage"


      /* ouvrir outputStream vers un fichier local */
      val file = File(this.filesDir, fileName)
      val outputStream = file.outputStream()
```

```
        /* sauvegarder le nouveau compteur d'image */
        preferences.edit().putInt("numImage",numImage+1).commit()

        /* copier inputStream qui pointe sur l'image de la galerie
         * vers le fichier local */
        inputStream?.copyTo(outputStream)

        /* mémoriser Uri de fichier local dans la propriété localUri */
        localUri = file.toUri()
        outputStream.close()
        inputStream?.close()

        //éventuellement afficher l'image dans ImageView
        binding.imageView.setImageURI(localUri)
    }
```

Dans la même activité `AddPlantActivity`, quand on sauvegarde les informations concernant la plante dans la base de données, il suffit de sauvegarder le String : `localUri.toString()`.

4 Afficher les images de plantes

Quand nous voulons afficher l'image d'une plante dans une autre activité il faut récupérer dans la base de données le String `s` qui code `localUri` et ensuite on procède comme d'habitude :

```
val localUri = Uri.parse( s )
imageView.setImageUri( localUri )
```

5 Copier l'image de la galerie vers la mémoire externe

Dans l'exemple on a copié l'image de la galerie d'images vers la mémoire privée interne de l'application. Bien sûr au lieu de copier vers le mémoire interne on peut aussi copier vers la mémoire externe.
Mais les deux solutions ont un inconvénient : la même image est stockée deux fois, une fois dans la galerie d'images et une copie soit dans mémoire interne soit dans la mémoire externe. Est-ce qu'on peut éviter cette copie ?