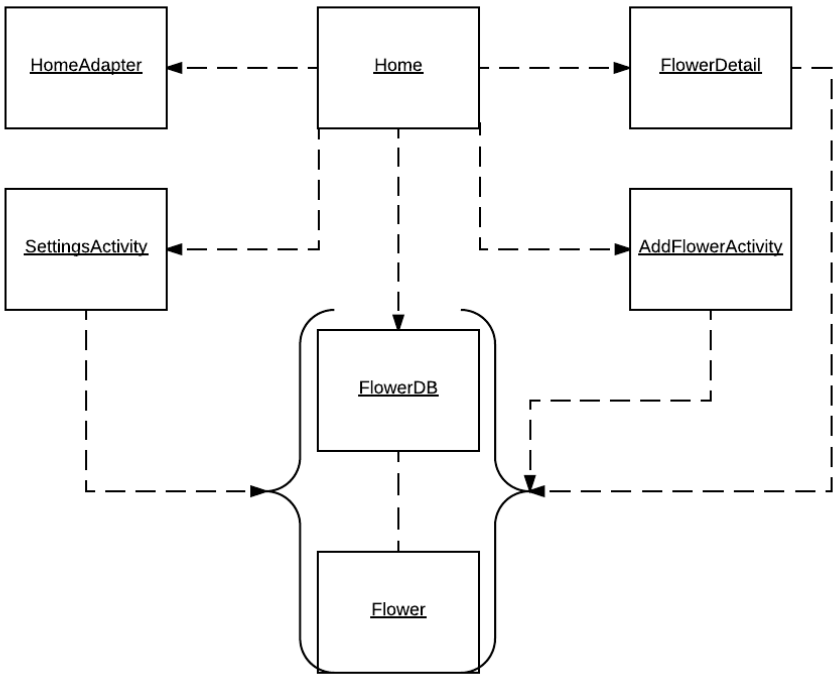
Documentation Technique - Flowering

Jean-Frédéric Durand

Description de l'architecture

- Home : Correspond à l'activité principale de l'application. Elle permet d'afficher la liste des plantes et est le point de départ pour toutes les autres activités :
  - AddFlowerActivity
  - SettingsActivity
  - FlowerDetail
- HomeAdapter : Correspond à la modélisation de l'adapteur de tableau vers liste. C'est ici qu'est programmée la mise en forme de la liste via un tableau de String correspondant au nom des plantes. C'est également ici qu'est conditionné l'affichage du background.
- SettingsActivity : Correspond à l'activité permettant de régler l'heure et d'importer les fixtures (10 plantes).
- AddFlowerActivity : Correspond à l'activité permettant d'ajouter une plante.
- FlowerDetail : Correspond à l'activité permettan d'afficher le détail d'une plante par un clic depuis la liste de l'activité Home. Il est également possible de modifier le nom d'une plante en cliquant sur "Modify" ou de supprimer celle-ci en cliquant sur "Delete"
- Flower : Correspond à la classe modélisant une plante. Le constructeur par défaut permet d'ajouter une plante en lui passant en paramêtre un nom et une fréquence d'arrosage.
- FlowerDB : Correspond à la classe permettant de créer et de communiquer avec la base SQLite. Une base de donnée est initiée



La figure représente les différents appels entre les classes et activités effectuées au sein du projet.

PS : Les classes Flower et FlowerDB on été regroupées sur le schéma pour signaler qu'on ne peut effectuer des opérations sur la base de données sans appeler au préalable l'objet Flower.

Description détaillée

Home (Activity)

Le `onCreate()` de home initialise tous les composants à savoir :

- La bouton d'option associé à la barre supérieure qui permet d'accéder au menu contextuel et donc au setting.
- Le bouton "+" permettant de lancer l'activité plante
- La listView contenant la liste des plantes

La fonction appelle également une autre fonction : `refreshList()` qui permet d'initialiser et de rafraichir la liste des plantes. En effet cette fonction appelle la base de donnée pour recevoir les plantes et insère via la classe HomeAdapter (dont nous parlerons plus bas) les plantes dans la liste. Cette fonction fait également appel à deux sous fonctions des listViews :

- `onItemClick()` dont le rôle est d'appeler l'activité FlowerDetail avec en paramètre (Bundle) le nom de la plante.
- `onItemLongClick()` dont le rôle est de définir le fond d'écran de l'item à vert puis de mettre à jour la date de dernière arrosage de la plante dans la base de donnée définie à aujourd'hui via la fonction `getToday()` .

HomeAdapter

Le rôle de HomeAdapter est d'une part de mettre en forme le tableau de plante récupéré via la requête sur la base de donnée. Et d'autre part de s'assurer la mise en forme conditionnelle de la couleur de fond d'écran de chaque item.

Pour cela j'ai Override la méthode `getView()` pour qu'elle compare la date d'aujourd'hui avec la date de prochain arrosage. La date de prochain arrosage est calculée de la façon suivante : date de dernier arrosage * frequence d'arrosage.

> Attention j'ai travaillé avec des timestamp en miliseconds. Ainsi 86400000 correspond à un jour en timestamp (24*60*60*1000)

```
Long today = SettingsActivity.getToday();
Long nextWatering = flower.getLastWaterDay()+(flower.getFrequency()*86400000);
if (nextWatering < today)
```

```
     v.setBackgroundColor(Color.RED);
else if (nextWatering > today +86400000)
    v.setBackgroundColor(Color.GREEN);
else
    // correspond à ((nextWatering <= today) && (nextWatering >= today+86400000))
    v.setBackgroundColor(Color.YELLOW);
return v;
```

SettingsActivity

Cette Activité permet deux actions :

- Modifier la date d'aujourd'hui provisoirement (la date n'est pas enregistrée en base de donnée).
- Importer les fixtures.

Elle implémente également une fonction `getToday()` qui renvoie la date d'aujourd'hui si la date n'a pas été changé. Dans le cas contraire elle renvoie la date sélectionnée par le composant DatePicker.

AddFlowerActivity

Cette activité est constitué de deux entrée textuelles et d'un bouton de validation :

- Entrée texte 1 : nom de la plante. Il est possible d'entrer n'importe quelle chaine de caractère en nom de plante.
- Entrée texte 2 : fréquence d'arrosage. Calibrée pour entrer un nombre.
- Bouton de validation : Le bouton de validation vérifie qu'aucun de ces deux champs n'est vide. De plus il vérifie que la fréquence d'arrosage est différent de 0 et de 1 pour respecter le cahier des charges.

FlowerDetail

Cette Activité renvoie les informations enregistrées en base de donnée (évoquée plus bas). Elle implémente également la possibilité de modifier le nom de la plante en cliquant sur le bouton `modify` et de supprimer la plante en cliquant sur le bouton `delete`.

Dans les deux cas une boite modale s'ouvre offrant la possibilité d'annuler l'action.

Flower

Cette classe correspond à la modélisation de la plante. Une plante est caractérisée selon ces critères :

- int id : un identifiant auto-généré par la base de donnée.
- String name : le nom de la plante.
- int freq : la fréquence d'arrosage (en nombre de jours).
- Long lastWaterDay : La dernière date d'arrosage (sous forme de Timestamp).

Les getters et les setters ont été également définis.

FlowerDB

La classe FlowerDB défini toutes les fonctions de communication avec la base de donnée SQLite.

- onCreate()
- on Upgrade()
- close()

Mais également les fonctions nécéssaire à l'ajout, la modification et la suppression de plante :

- long addFlower(Flower)
- int updateFlower(Flower)
- int deleteFlower(int idFlower)

Des fonctions de lecture de la base pour récupérer les plantes ont également été définies :

- Flower getFlower(int id) : Renvoie une plante en fonction de son id
- Flower getFlowerByName(String name) : Renvoie une plante en fonction de son nom
- ArrayListgetFlower() : Renvoyant la liste des plantes