



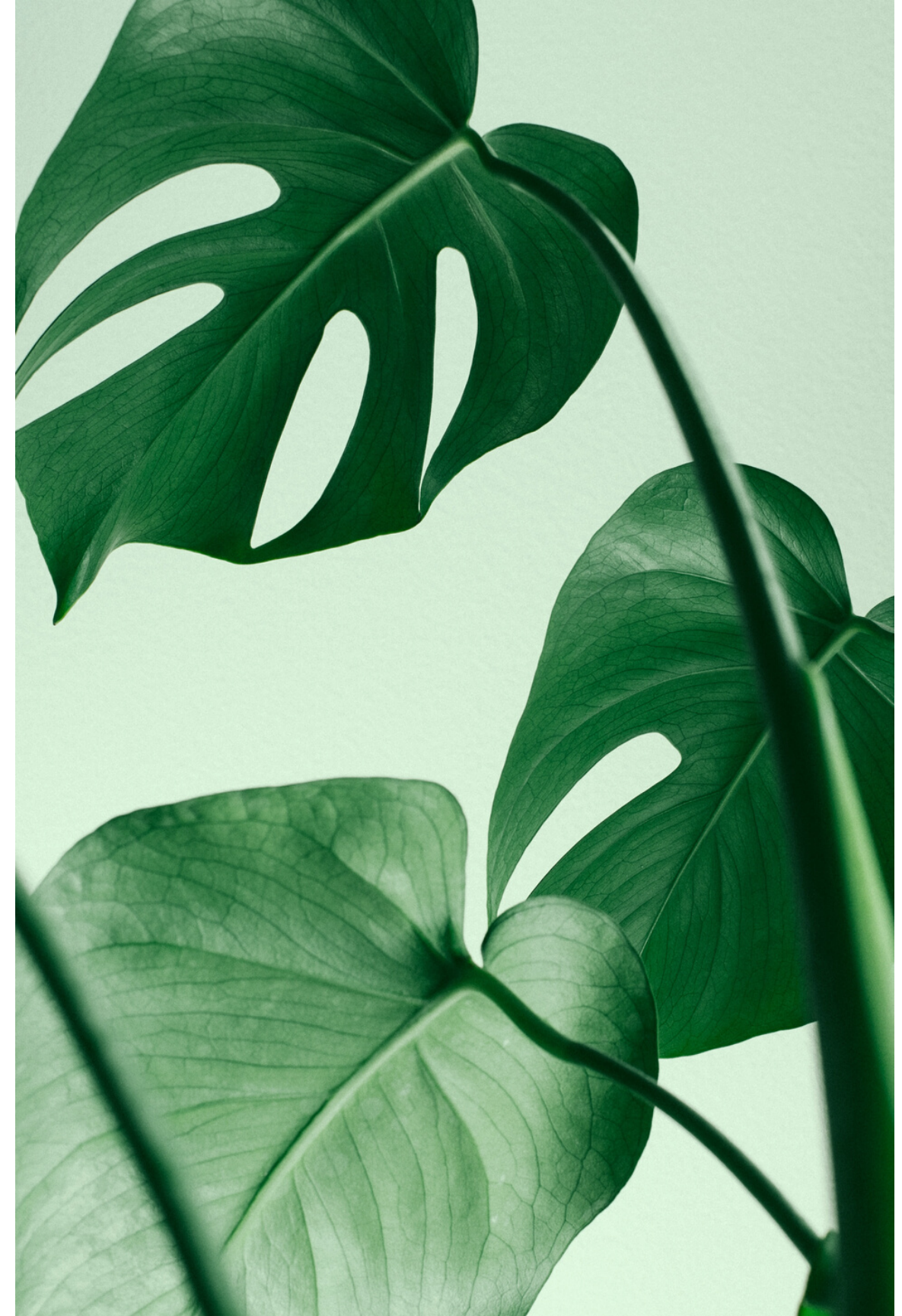
Projet mobile

Green Collection

Dhrif Nour
Henchir Elaa

PLAN

- Présentation de l'application
- Principales fonctionnalités
- Segmentation clients
- Réalisation





Présentation de l'application

Dans le cadre de la réalisation d'un projet mobile développé en Kotlin on a décidé de concevoir une application nommée **GreenCollection** qui permet de collectionner les plantes.

Principales fonctionnalité



-
- Fournir aux utilisateurs des informations concernant des plantes ,grâce à la base de données.En effet chaque client est capable d'ajouter des éléments en remplissant un formulaire ,de supprimer l'une de ses propres plantes et consulter les plantes enregistrées dans la base.
 - Permettre aux collectionneurs d'exercer leurs passion en personnalisant leurs collections à l'aide de l'option d'ajout aux favoris .





Segmentation client

Qui sont nos clients ?



L'application est destinée aux personnes qui sont passionnées par les plantes et s'intéressent aux leurs caractéristiques.

Elle peut être aussi ciblée aux collectionneurs.



Réalisation

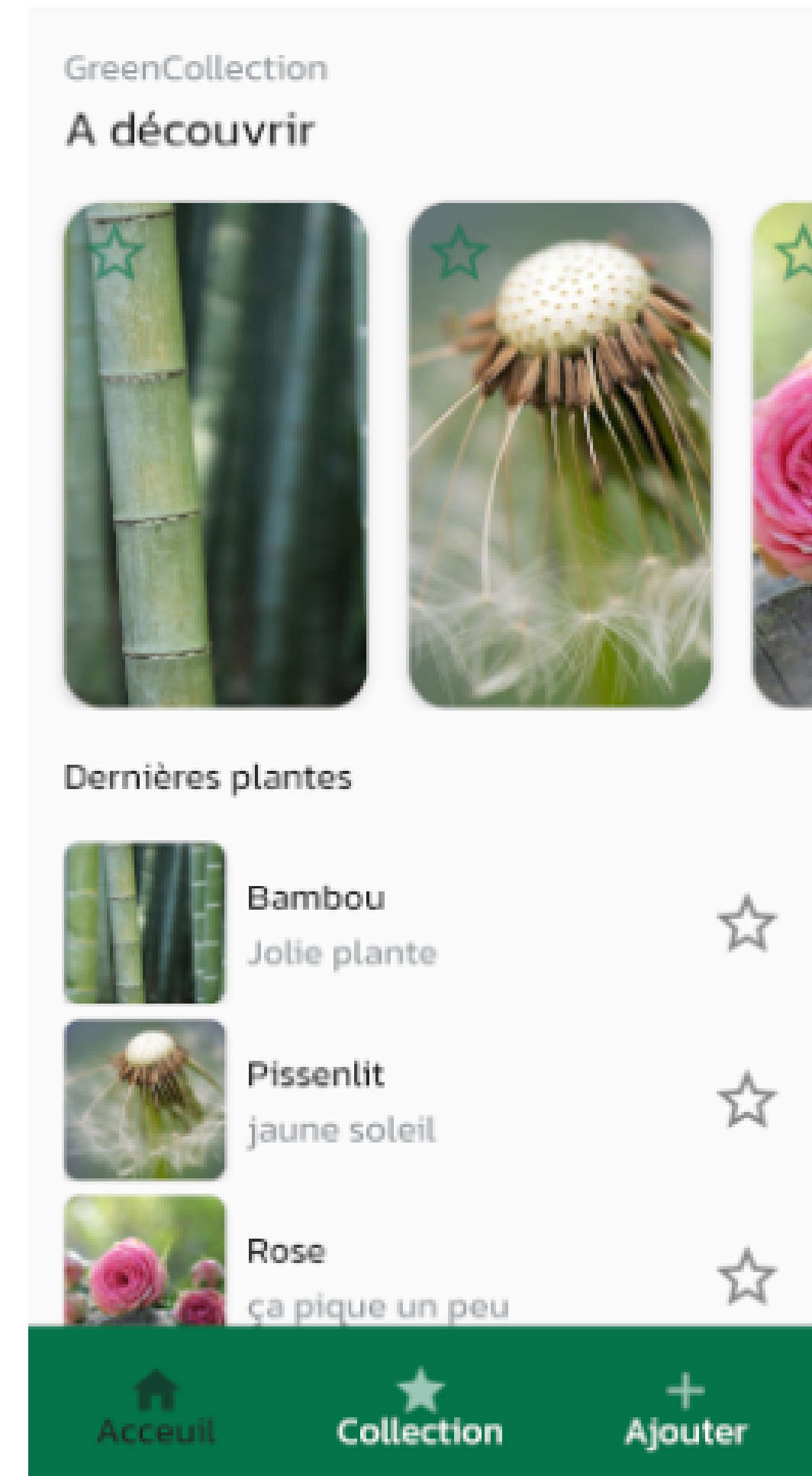


I-La page d'accueil:

L'application s'ouvre sur une page d'accueil qui se débute par des images des plantes à découvrir positionnées horizontalement.

Dans le corps de cette page on trouve une liste des dernières plantes accompagnées par une petite description. En cliquant sur l'une de ces plantes une pop up s'affiche indiquant les informations de chaque plante.

L'accueil se termine par une barre de navigation.



Pour l'interface de la première page on a utilisé **Recycler View**

.Recycler View:

Le composant "RecyclerView" remplace les anciens composants "ListView" et "GridView". Il est notamment adapté pour des éléments basés sur de grands ensembles de données ou des données qui changent fréquemment.

Le premier fichier à implémenter est le layout principal de l'activité. C'est dans ce fichier XML que l'on déclare l'utilisation de la "RecyclerView". Pour ce faire nous avons ajouté un nouveau layout ressource file nommé fragment_home.xml



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/horizontal_recycler_view"
        android:layout_width="match_parent"
        android:layout_height="250dp"
        android:orientation="horizontal"
        app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dernières plantes"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="20dp"
        style="@style/SubtitleTextStyle"/>
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/vertical_recycler_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"/>
</LinearLayout>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Dernières plantes"
    android:layout_marginTop="20dp"
    android:layout_marginLeft="20dp"
    style="@style/SubtitleTextStyle"/>
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/vertical_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="20dp"
    android:layout_marginLeft="20dp"
    android:orientation="vertical"
    app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
    nearLayout>
```

On crée également un autre fichier XML. Celui-ci va correspondre au contenu d'une cellule.

Pour la partie horizontale on a créé le fichier **item_horizontal_plant.xml** et pour la verticale le fichier **item_vertical_plant.xml**

L'étape suivante est la création de l'adaptateur **PlantAdapter.Kt** lié à notre vue. C'est une classe Kotlin qu'on va la donner a recycler view pour pouvoir adapter à chaque plante son équivalent en image. Cette dernière contient une autre classe *ViewHolder* qui va jouer le rôle d'une boîte pour ranger tous les composants à contrôler .



```
class PlantAdapter(  
    private val context: MainActivity,  
    private val plantList: List<PlantModel>,  
    private val layoutId: Int  
) : RecyclerView.Adapter<PlantAdapter.ViewHolder>()  
{  
  
    // boite pour ranger tout les composants à contrôler  
    class ViewHolder(view: View) : RecyclerView.ViewHolder(view)  
    {  
        val plantImage = view.findViewById<ImageView>(R.id.image_item)  
        val plantName: TextView? = view.findViewById(R.id.name_item)  
        val plantDescription: TextView? = view.findViewById(R.id.description_item)  
        val starIcon = view.findViewById<ImageView>(R.id.star_icon)  
    }  
  
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder  
    {  
        val view = LayoutInflater  
            .from(parent.context)  
            .inflate(layoutId, parent, attachToRoot: false)  
        return ViewHolder(view)  
    }  
}
```

```

override fun onBindViewHolder(holder: ViewHolder, position: Int)
{
    //recuperer les informations de la plante
    val currentPlant = plantList[position]

    //recuperer le repository
    val repo = PlantRepository()

    //utiliser glide pour recuperer l'image à partir de son lien
    Glide.with(context).load(Uri.parse(currentPlant.imageUrl)).into(holder.plantImage)

    //mettre à jour le nom de la plante
    holder.plantName?.text = currentPlant.name

    //mettre à jour la description de la plante
    holder.plantDescription?.text = currentPlant.description

    //verifier si la plante est liké ou non
    if (currentPlant.liked)
    {
        holder.starIcon.setImageResource(R.drawable.ic_like)
    }
    else
    {
        holder.starIcon.setImageResource(R.drawable.ic_unlike)
    }
}

```

```

        // ajouter une interaction sur cette etoiles
        holder.starIcon.setOnClickListener{ it: View!
            //inverser si le bouton est like ou non
            currentPlant.liked= ! currentPlant.liked
            //mettre a jour l'objet plant
            repo.updatePlant(currentPlant)
        }
    }

    override fun getItemCount(): Int
    {
        return plantList.size
    }
}

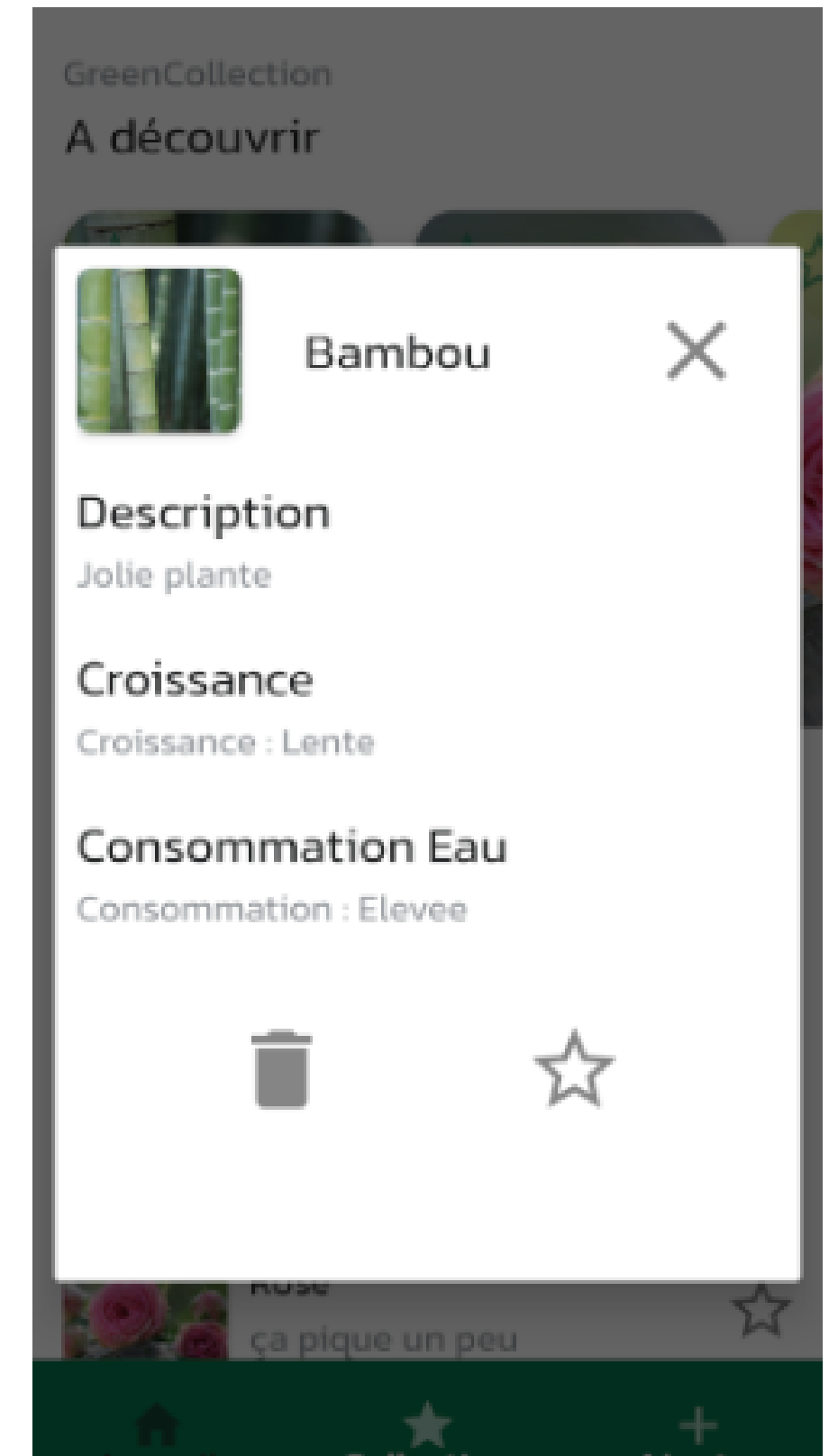
```

La dernière étape consiste à ajouter une classe kotlin **HomeFragment** pour récupérer le premier et le deuxième Recycler View

```
class HomeFragment (  
    private val context: MainActivity  
) : Fragment() {  
    override fun onCreateView(  
        inflater: LayoutInflater,  
        container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View? {  
        val view = inflater?.inflate(R.layout.fragment_home, container, attachToRoot: false)  
  
        // récupérer le recyclerview  
        val horizontalRecyclerView = view.findViewById<RecyclerView>(R.id.horizontal_recycler_view)  
        horizontalRecyclerView.adapter = PlantAdapter(context, plantList, R.layout.item_horizontal_plant)  
  
        // récupérer le second recycler view  
        val verticalRecyclerView = view.findViewById<RecyclerView>(R.id.vertical_recycler_view)  
        verticalRecyclerView.adapter = PlantAdapter(context, plantList, R.layout.item_vertical_plant)  
        verticalRecyclerView.addItemDecoration(PlantItemDecoration())  
        return view  
    }  
}
```

Concernant le popup on a défini son interface graphique dans un fichier **popup_plants_details** .

Puis pour afficher cet élément on a créé une nouvelle classe file **PlantPopup** dont on a injecté le popup qu'on a créé de manière qu'il soit spécifique à chaque plante



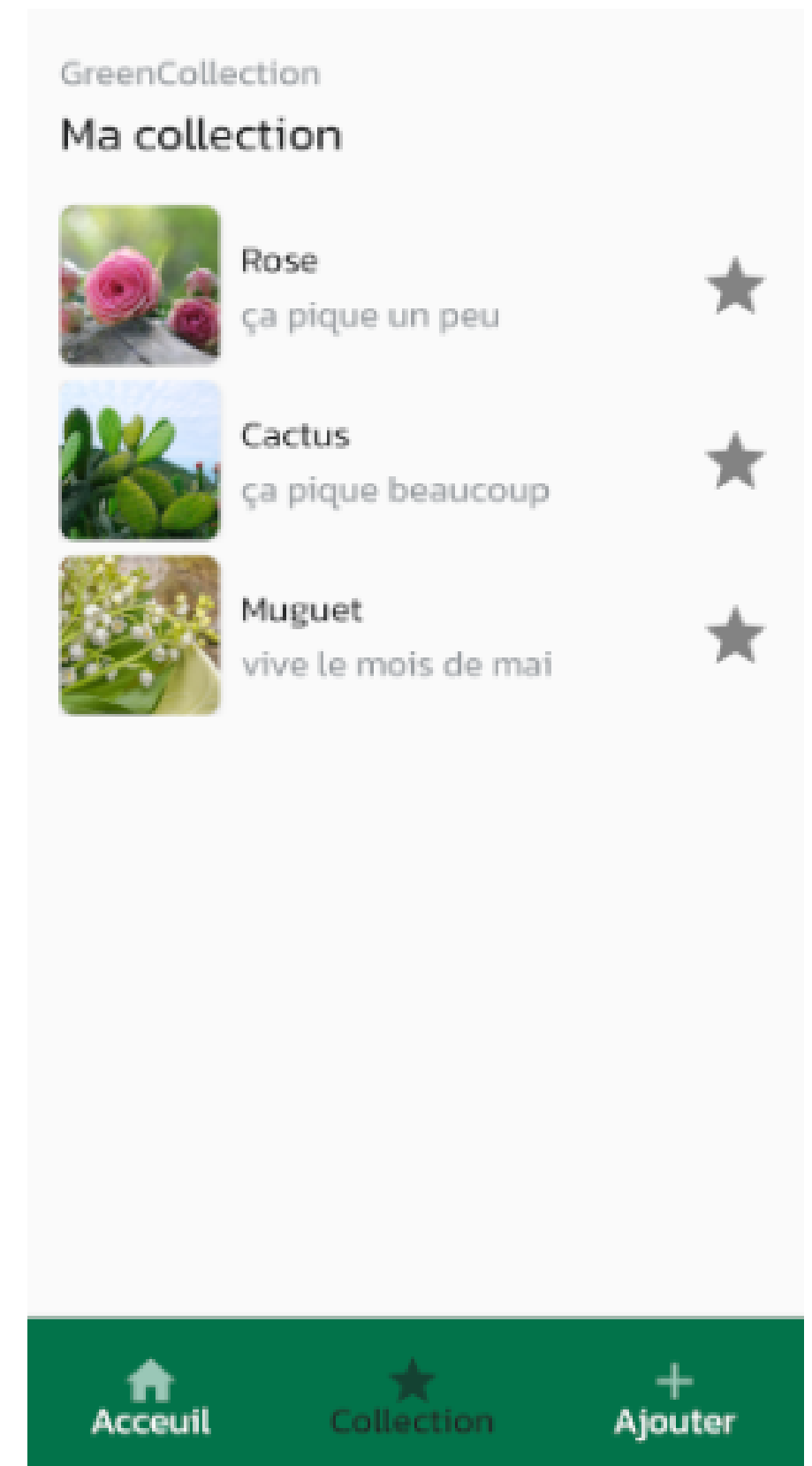
L'interaction avec les icones de fermeture ,de suppression et de like on étaient générés par des fonctions écrites dans cette classe

Pour l'affichage de ce popup sur notre page on a ajouté ce code dans **PlantAdapter.kt**

II. Page de collection:

En cliquant sur l'icone "star" d'une plante elle s'ajoute automatiquement à cette page elle nous sert donc d'afficher nos plantes favorises .

Si on dislike une plante elle n'apparaît plus dans cette activité .



-L'interface graphique a été définie dans le fichier **fragment_collection.xml** où on a utilisé le recycler view

```
package com.example.myapplication.fragments

import ...

class CollectionFragment(
    private val context: MainActivity
) : Fragment() {
    //injecter element
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater?.inflate(R.layout.fragment_collection, container, attachToRoot: false)

        //recupérer ma recyclerview
        val collectionRecyclerView = view.findViewById<RecyclerView>(R.id.collection_recycler_list)
        collectionRecyclerView.adapter = PlantAdapter(context, plantList.filter { it.liked }, R.layout.item_vertical_plant)
        collectionRecyclerView.layoutManager = LinearLayoutManager(context)
        collectionRecyclerView.addItemDecoration(PlantItemDecoration())

        return view
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/collection_recycler_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginLeft="20dp"/>

</LinearLayout>
```

Pour gérer ce fragment on a ajouté la classe **CollectionFragment** .


III. Page d'ajout :

Cette page nous permet d'ajouter une nouvelle plante en remplissant un formulaire contenant un bouton pour charger une image à partir du téléphone, des édit text et des spinner pour remplir les informations de la plante ajoutée.

GreenCollection

Ajout d'une plante

CHARGER IMAGE




Nom de la plante


Description de la plante


Croissance : Lente

Consommation : Faible

CONFIRMER

Acceuil

Collection

Ajouter

-L'interface graphique a été définie dans le **fragment_add_plant.xml** où on a utilisé le recycler view.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="5dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <Button
            android:id="@+id/upload_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Charger Image"
            android:background="@color/xxx"
            style="@style/TitleTextStyle"
            android:textColor="@color/white"
            android:padding="20dp"/>

        <ImageView
            android:id="@+id/preview_image"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:src="@drawable/trending2"
            android:scaleType="centerCrop"/>

    </LinearLayout>
```

```
<EditText
    android:id="@+id/name_input"
    android:layout_width="match_parent"
    android:layout_height="80dp"
    android:hint="@string/item_plant_name"
    android:paddingLeft="10dp"
    android:layout_marginTop="5dp"
    style="@style/DefaultTextStyle"/>

<EditText
    android:id="@+id/description_input"
    android:layout_width="match_parent"
    android:layout_height="80dp"
    android:hint="@string/item_plant_description"
    android:paddingLeft="10dp"
    android:layout_marginTop="5dp"
    style="@style/DefaultTextStyle"/>

<Spinner
    android:id="@+id/grow_spinner"
    android:layout_width="match_parent"
    android:layout_height="80dp"
    android:entries="@array/add_plant_page_grow_spinner_input"
    android:paddingLeft="10dp"
    android:layout_marginTop="5dp" />
```

Pour récupérer les informations de ce formulaire on a ajouté la classe **AddPlantFragment** qui va envoyer ces données à notre base.

```
private var file: Uri?=null
private var uploadedImage : ImageView? = null

override fun onCreateView(
    inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    val view = inflater?.inflate(R.layout.fragment_add_plant, container, attachToRoot: false)

    //recuperer uploadedImage pour lui associer son composant
    uploadedImage = view.findViewById(R.id.preview_image)

    //recuperer le bouton
    val pickupImageButton = view.findViewById<Button>(R.id.upload_button)

    //click ça ouvre image du tel

    pickupImageButton.setOnClickListener { pickupImage() }

    //recuperer le bouton confirmer
    val confirmButton = view.findViewById<Button>(R.id.confirm_button)
    confirmButton.setOnClickListener { sendForm(view) }

    return view
}
```

```
private fun sendForm(view: View) {
    val repo = PlantRepository()
    repo.uploadImage(file!!) {
        val plantName = view.findViewById<EditText>(R.id.name_input).text.toString()
        val plantDescription = view.findViewById<EditText>(R.id.description_input).text.toString()
        val grow = view.findViewById<Spinner>(R.id.grow_spinner).selectedItem.toString()
        val water = view.findViewById<Spinner>(R.id.water_spinner).selectedItem.toString()
        val downloadImageUrl = downloadUri

        //créer un nouvel objet PlantModel
        val plant = PlantModel (
            UUID.randomUUID().toString(),
            plantName,
            plantDescription,
            downloadImageUrl.toString(),
            grow,
            water
        )

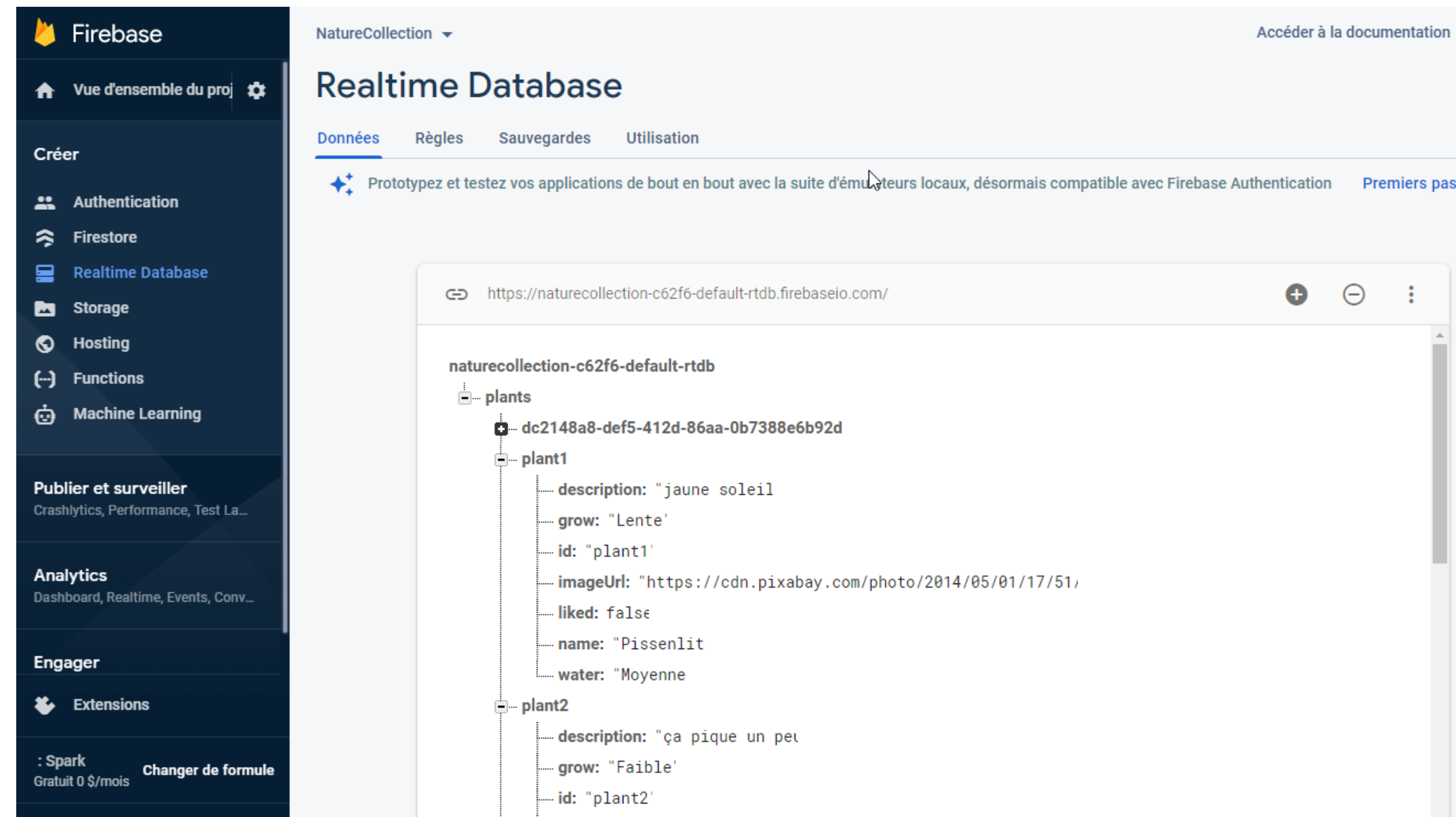
        //envoyer en bd
        repo.insertPlant(plant)
    }
}
```


IV. Base de données :

Dans notre application on utilise la real time data base de la plateforme mobile de google **Fire Base**.

Firestore Realtime Database

n'est autre qu'une base de données NoSQL, bénéficiant d'un hébergement « Cloud » et permettant le stockage et la synchronisation de données de nos utilisateurs. Les développeurs peuvent gérer cette base de données en temps réel.



L'interaction avec cette base de données était garantie par la classe **PlantRepository** qui nous permet la récupération des données et leurs mise à jour dans la base

```
class PlantRepository {  
    object Singleton {  
        //donner le lien pour accéder au bucket  
        private val BUCKET_URL: String = "gs://naturecollection-c62f6.appspot.com"  
  
        //se connecter à notre espace de stockage  
        val storageReference = FirebaseStorage.getInstance().getReferenceFromUrl(BUCKET_URL)  
  
        // se connecter à la ref plante  
        val databaseRef = FirebaseDatabase.getInstance().getReference(path: "plants")  
  
        // créer une liste qui contient les plantes  
        val plantList = arrayListOf<PlantModel>()  
  
        //contenir le lien de l'image courante  
        var downloadUri : Uri? = null  
    }  
}
```

```
fun updateData(callback: () -> Unit) {  
    //absorber les données dans la database ref -> liste de plant  
    databaseRef.addValueEventListener(object : ValueEventListener {  
        override fun onCancelled(p0: DatabaseError) {  
        }  
  
        override fun onDataChange(p0: DataSnapshot) {  
            // retirer les anciennes  
            plantList.clear()  
            //recueillir la liste  
            for (ds in p0.children) {  
                //construire un objet plante  
                val plant = ds.getValue(PlantModel::class.java)  
                // verifier plant pas nul  
                if(plant != null ) {  
                    //ajout plante à autre liste  
                    plantList.add(plant)  
                }  
            }  
            // actionner callback  
            callback()  
        }  
    } )  
}
```

```
//mettre à jour un objet plant en bdd  
fun updatePlant(plant: PlantModel) = databaseRef.child(plant.id).setValue(plant)  
  
//insérer une nouvelle plante en bd  
fun insertPlant(plant: PlantModel) = databaseRef.child(plant.id).setValue(plant)  
  
//supprimer plante de bd  
fun deletePlant(plant: PlantModel) = databaseRef.child(plant.id).removeValue()
```

**Merci pour votre
attention**