2 - Créer sa première application Android

Objectifs

- Créer une application Android avec Android Studio
- Utiliser Android Studio et son outil de prévisualisation
- Utiliser Jetpack Compose pour mettre à jour l'interface utilisateur
- Afficher une preview de l'application Compose

Création du projet pas à pas

Démonstration

https://developer.android.com/codelabs/basic-android-kotlin-compose-first-app?hl=fr#0

Lancement sur émulateur

https://developer.android.com/codelabs/basic-android-kotlin-compose-emulator?hl=fr

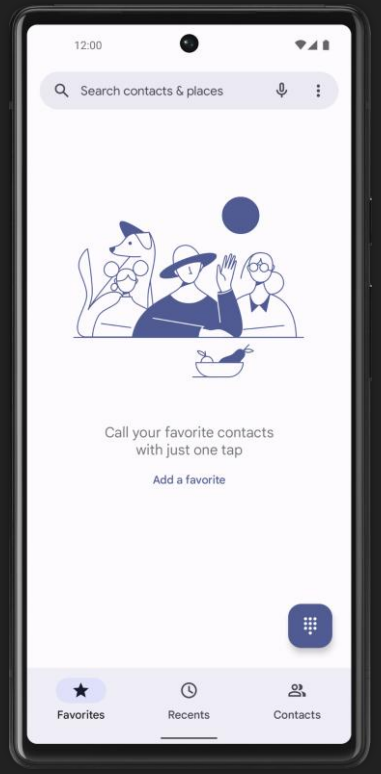
Lancement sur un véritable appareil

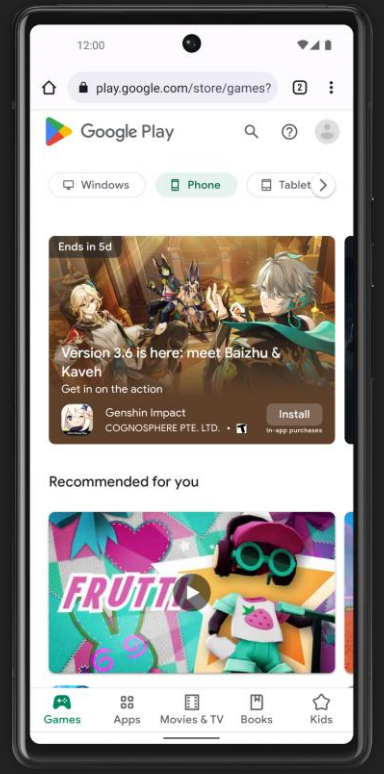
https://developer.android.com/codelabs/basic-android-kotlin-compose-connect-device?hl=fr&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-1-pathway-2%3Fhl=fr%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-connect-device - 0

Introduction à Compose

Comment gérer l'interface utilisateur d'une app

Quels éléments d'interface voyez-vous sur ces deux captures d'écrans ?





Comment gérer l'interface utilisateur d'une app

- Vues XML (ancienne façon)
- Jetpack Compose (nouvelle façon)
- Permet de définir les éléments à afficher à l'écran et les interactions possible

Jetpack Compose

- Open source
- Développé par les équipes d'Android
- Fait partie de la suite de bibliothèque Jetpack
- Déclaratif
- La déclaration de l'interface se fait directement dans le code en Kotlin
- Plus concis et rapide à écrire que les Vues en XML

Compose (Kotlin) vs Vue (XML)

View

```
androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:fontFamily="sans-serif-light"
        android:text="Happy Birthday, Sam!"
        android:textColor="@android:color/black"
        android:textSize="36sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:fontFamily="sans-serif-light"
        android:text="From Emma."
        android:textColor="@android:color/black"
        android:textSize="36sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

VS.

Compose

```
fun BirthdayGreetingWithText(message: String, from: String) {
    Column {
        Text(
            text = message,
            fontSize = 36.sp,
            modifier = Modifier
                .fillMaxWidth()
                .wrapContentWidth(Alignment.CenterHorizontally)
                .padding(start = 16.dp, top = 16.dp)
        )
        Text(
            text = from,
            fontSize = 24.sp,
            modifier = Modifier
                .fillMaxWidth()
                .wrapContentWidth(Alignment.CenterHorizontally)
                .padding(start = 16.dp, end = 16.dp)
        )
    }
}
```

Jetpack Compose

- Immuable
- S'il y a une modification, le composable est recrée
- C'est la **recomposition**
- Les composables qui n'ont pas changé ne sont pas recomposés

Annotations

- Déclaration d'une fonction composable avec l'annotation **@Composable**
- Une annotation noté avec l'**@**, permet de rajouter de l'information au code pour le développeur et le compilateur et permettent d'ajouter des fonctionnalités supplémentaires
- Possibilité d'ajouter des paramètres
- Example : @Preview(showBackground = true)

Composables

- Utilisation de fonctions composables pour définir l'interface
- Une fonction composable peut avoir des paramètres
- Une fonction composable peut en appeler une autre
- Une fonction composable ne renvoit rien
- Le nom doit être un nom écrit en PascalCase

```
@Composable
fun Greeting(name: String) {
    Text(text = "Hello $name!")
}
```

Atelier – Message d'anniversaire

https://developer.android.com/codelabs/basic-android-kotlin-compose-text-composables?hl=fr

Atelier (20 min)

- Réalisation d'une app avec un message de « Joyeux anniversaire » personnalisé
- Vous pouvez sauter les sections 3 et 4 si vous le souhaitez
- Bien respecter le nom de l'app « Happy Birthday », le nom du theme en dépend « HappyBirthdayTheme »

Layouts standards

- Box -> empiler des éléments en précisiant un alignement (en haut à gauche, au centre…)
- Column -> affichage en colonne verticale
- Row -> affichage en ligne horizontale

Ajouter une image

- 2 solutions :
  - Ajouter l'image au projet
  - Afficher une image depuis Internet (besoin d'une bibliothèque supplémentaire)

Ajouter une image au projet

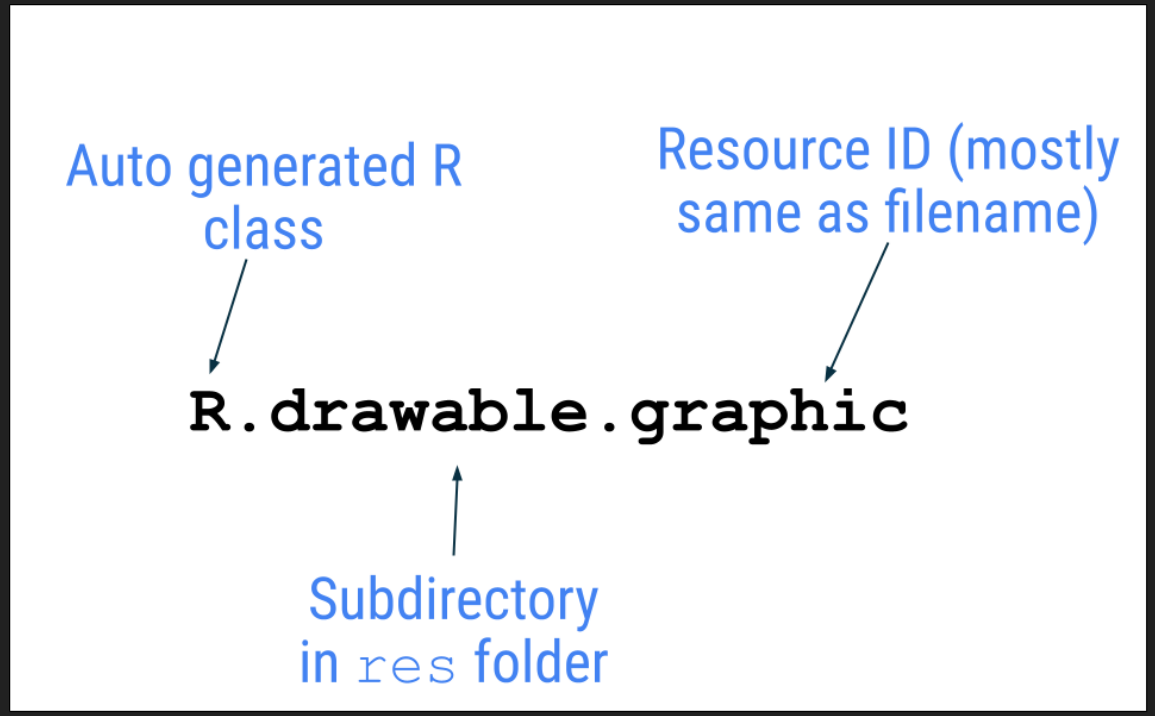
- Utilisation du **Ressource Manager**
- Accessible sur le volet de droite ou **View** > **Tool Windows** > **Ressource Manager**
- Permet d'importer, créer et gérer des ressources dans l'applications (images, texte, couleurs, boutons, assets (autres type de fichiers)…)
- Appuyer sur le bouton **+** > **Import Drawables**

Ajouter une image au projet

- Possibilité d'associer un drawable à qualificatif (taille d'écran, langue, orientation du téléphone…)
- Possibilité d'associer à une taille d'écran avec une densité spécifique
- -> Permet de gérer les tailles et redimensionnent des images suivant la taille de l'écran
- Ici choisir comme **qualifier > Density** et comme **value > No density** pour que l'image ne soit pas redimmensionné
- L'image est ajouté dans **res > drawable** et dans les fichiers du système **main > res > drawable-nodpi**

Accéder aux ressources

- Classe **R** autogénérée par Android
- Permet d'accéder à toutes les ressources du projet
- À importer dans le fichier avec import R



Atelier – Message d'anniversaire avec Image

https://developer.android.com/codelabs/basic-android-kotlin-compose-add-images?hl=fr#0

Atelier (20 min)

- Rajouter une image de fond au message d'anniversaire
- Reprenez le code du projet précédant « Happy Birthday »

Bonne pratique – Ressource string

- Créer des ressources de strings pour toutes les chaînes de caractères de l'app
- Permet de les réutiliser plus facilement en utilisant leurs noms
- Permet la traduction
- Se retrouve dans présente dans **res** > **values** > **strings.xml**
- **Utiliser l'outil d'extraction de string directement dans le code où une string est présente**
- **Ampoule > Extract string resource puis définir son nom et sa valeur**

Bonne pratique – Ressource string

- Le nom est en snake_case (minuscule, mots séparés par des underscore)
- Nommer la ressource suivante sont contexte, où elle est utilisée et son utilité, en anglais de préférence
- Exemple :
  - user_add_user_button_text : le titre de la fonction ajout d'utilisateur dans le contexte de la gestion d'utilisateur
  - home_title : le titre de la page d'accueil de l'application
- Récupérer la chaîne grâce à son nom et la méthode **getString()** ou **stringResource()**
- Ex : **getString(R.string.happy_birthday_text)**

Quiz

Exercices supplémentaires

- S'entraîner aux layout Compose : https://developer.android.com/codelabs/basic-android-kotlin-compose-composables-practice-problems?hl=fr
- Projet : créer une carte de visite : https://developer.android.com/codelabs/basic-android-kotlin-compose-business-card?hl=fr