

5. Gestion des évènements simples

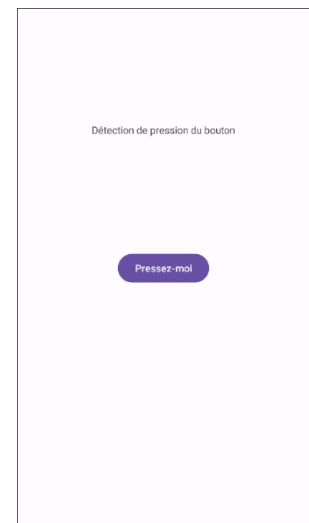
Réagir aux interactions de l'utilisateur avec les widgets, et exploitation des liaisons de vues.

Android Studio Flamingo Essentials, Kotlin Edition : chapitres 18 et 33

5.1. Nouveau projet

1. Créez un nouveau projet basé sur le gabarit « Empty Views Activity » et constitué comme suit

- a. Centrez un `Button` titré « Pressez-moi »
- b. Déposez un `TextView` au-dessus du `Button`, et assignez-lui le texte « Détection de pression du bouton » afin qu'il soit visible dans l'éditeur Design
- c. Attribuez des identificateurs significatifs à chacun (*`messageTextView`* et *`pressezMoiButton`*)
- d. Exécutez l'application pour démontrer son bon fonctionnement



2. Soulignez qu'on désire effacer la chaîne du `TextView` au démarrage afin qu'il n'affiche rien tant que l'utilisateur n'a pas appuyé le bouton.


- a. Demandez aux étudiants de suggérer une façon d'accomplir cette tâche

- b. Solution :

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.TextView

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val messageTV: TextView = findViewById(R.id.messageTextView)
        messageTV.text = null
    }
}
```

- c. Exécutez l'application pour démontrer son bon fonctionnement
3. Profitez de l'occasion pour démontrer l'utilité des attributs dédiés au design de l'interface (ceux de catégorie **tools**, accompagnés du symbole )
- a. Déplacez le texte « Détection de pression du bouton » de l'attribut **android:text** à **tools:text**
 - b. Éliminez le code précédemment inséré dans **onCreate()**
 - c. Exécutez l'application pour démontrer son bon fonctionnement

5.2. Attribut **onClick** des **Button**

- 4. Expliquez qu'il existe un raccourci simple et rapide pour assigner du code *Kotlin* à un toucher court de **Button** : son attribut **onClick**
- 5. Ajoutez la fonction suivante à la classe de l'activité :

```
fun toucherBouton(view: View) {
    val messageTV: TextView = findViewById(R.id.messageTextView)
    messageTV.text = "Bouton pressé"
}
```

- a. Dans le fichier XML de l'activité (ou via l'interface de l'éditeur Design), ajoutez l'attribut suivant

```
<Button
    android:id="@+id/pressezMoiButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Pressez-moi"
    android:onClick="toucherBouton"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

6. Exécutez l'application pour démontrer son bon fonctionnement
7. Expliquez que cette approche est limitée (détection d'un seul type de toucher sur des `Button`)

5.3. Event listeners

8. Expliquez que la façon adéquate de gérer la détection de touchers de widgets dans une application *Android* est d'exploiter des « capteurs d'évènements »
9. Énumérez les principaux évènements capturables par un `View` :
 - a. `onClick()` : détection de toucher puis retrait rapide du doigt de l'écran
 - b. `onLongClick()` : détection de toucher d'écran pendant une période prolongée suivi d'un retrait du doigt
 - c. `onTouch()` : détection de tout contact avec l'écran, incluant les mouvements tels que le *swipe* et le *pinch*
 - d. `onFocusChanged()` : détection de changement du focus d'un `View` à un autre
 - e. `onKey()` : détection d'une pression d'une touche de clavier physique (pas toujours fonctionnel avec le clavier virtuel)
10. Expliquez que la détection d'un évènement se fait en deux étapes
 - a. Installer un capteur pour l'évènement d'intérêt sur le widget
 - b. Définir la fonction à exécuter lorsqu'un évènement est capté
11. Avant de procéder, supprimez le contenu de l'attribut `onClick` du `Button`, ainsi que la fonction créée précédemment, `toucherBouton()`
12. Associez au `Button` de l'activité un capteur de touchers courts

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
```

```
import android.widget.Button
import android.widget.TextView

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val bouton: Button = findViewById(R.id.pressezMoiButton)
        bouton.setOnClickListener {
            val messageTV: TextView = findViewById(R.id.messageTextView)
            messageTV.text = "Bouton pressé"
        }
    }
}
```

13. Exécutez l'application pour démontrer que ça fonctionne

14. Associez maintenant au Button de l'activité un capteur de touches prolongés

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    val bouton: Button = findViewById(R.id.pressezMoiButton)
    bouton.setOnClickListener {
        val messageTV: TextView = findViewById(R.id.messageTextView)
        messageTV.text = "Bouton pressé"
    }

    bouton.setOnLongClickListener {
        val messageTV: TextView = findViewById(R.id.messageTextView)
        messageTV.text = "Bouton pressé longuement"

        true
    }
}
```

15. Soulignez que `setOnLongClickListener()` retourne un booléen indiquant si l'évènement a été « consommé » ou non par le capteur

- a. Expliquez que si le gestionnaire retourne `false`, l'évènement sera passé aux autres capteurs du View (s'il y en a d'autres) jusqu'à ce qu'un de ceux-ci « consomme » l'évènement

16. Exécutez l'application pour démontrer qu'elle détecte les touches courts et prolongés distinctement

17. Modifiez `setOnLongClickListener()` afin qu'elle retourne `false`, puis exécutez à nouveau l'application pour démontrer qu'un toucher prolongé du bouton déclenche les deux évènements

5.4. Refactorisation

18. Puisque le contenu des deux routines est presque identique, refactorisez le code ainsi :

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val bouton: Button = findViewById(R.id.pressezMoiButton)
        bouton.setOnClickListener {
            afficherMessage("Bouton pressé")
        }

        bouton.setOnLongClickListener {
            afficherMessage("Bouton pressé longuement")
            true
        }
    }

    private fun afficherMessage(message: String) {
        val messageTV: TextView = findViewById(R.id.messageTextView)
        messageTV.text = message
    }
}
```

19. Démontrez qu'on peut attribuer des *listeners* à n'importe quel widget, incluant les *layouts*

- a. Attribuez un `id` au `ConstraintLayout` de l'activité (p.ex. `mainLayout`)
- b. Ajoutez le code suivant à la fin de `onCreate()` :

```
val layout: ConstraintLayout = findViewById(R.id.mainLayout)
layout.setOnClickListener {
    afficherMessage("Fond d'écran pressé")
}
```

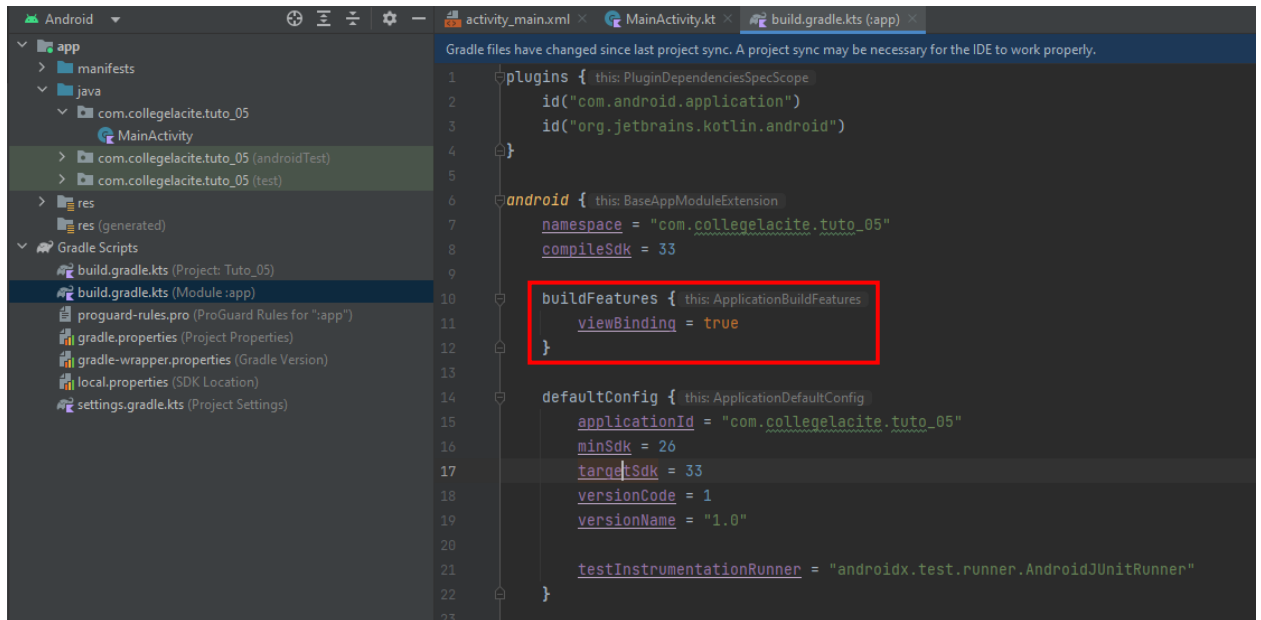
5.5. Liaisons de vue

20. Expliquez que l'utilisation de `findViewById()` pour référer à un `View` dans le code *Kotlin* peut être simplifié en exploitant la *liaison de vue* (*view binding* en anglais)

- a. Consiste à créer une classe dans laquelle à chaque widget de l'activité correspond un attribut de la classe nommée selon le `id` du widget

- b. La plupart des gabarits de projet exploitent la liaison de vue, sauf « Empty Views Activity ». Il faut donc intégrer la liaison de vue au projet créé précédemment

21. Il faut premièrement modifier le fichier de projet *app* » *Gradle Scripts* » *build.gradle.kts* (*Module:apps*) :



- a. Une fois la modification à *Gradle* effectuée, cliquez le lien [Sync Now](#) au haut de l'éditeur, puis recompilez le projet afin de vous assurer que la liaison de vue y est activée

22. Enfin remplacez les invocations à `findViewById()` par des références à la classe issue de la liaison de vue :

```

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.TextView
import androidx.constraintlayout.widget.ConstraintLayout
import com.collegelacite.tuto_05.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

```

```

setContentView(R.layout.activity_main)

binding = ActivityMainBinding.inflate(layoutInflater)
setContentView(binding.root)

val bouton: Button = findViewById(R.id.pressezMoiButton)
binding.pressezMoiButton.setOnClickListener {
    afficherMessage("Bouton pressé")
}

binding.pressezMoiButton.setOnLongClickListener {
    afficherMessage("Bouton pressé longtemps")
    true
}

val layout: ConstraintLayout = findViewById(R.id.main_Layout)
binding.mainLayout.setOnClickListener {
    afficherMessage("Fond d'écran pressé")
}

private fun afficherMessage(message: String) {
    val messageTV: TextView = findViewById(R.id.messageTextView)
    binding.messageTextView.text = message
}
}

```

23. Exécutez l'application pour démontrer qu'elle fonctionne comme avant

24. Soulignez que l'auteur exploite exclusivement la liaison de vue dans son manuel de référence

- a. Conséquemment nous allons aussi l'exploiter à partir de maintenant, et ce jusqu'à la fin du trimestre

25. Donc en résumé, voici les deux étapes à faire pour activer la liaison de vues dans un projet créé à partir du gabarit « Empty Views Activity » :

- a. Dans *app* » *Gradle Scripts* » *build.gradle.kts* (*Module:apps*) ajoutez :

```

buildFeatures {
    viewBinding = true
}

```

b. Modifiez MainActivity.kt :

```
class MainActivity : AppCompatActivity() {  
    private lateinit var binding: ActivityMainBinding  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
  
        binding = ActivityMainBinding.inflate(layoutInflater)  
        setContentView(binding.root)  
  
        ...  
    }  
}
```

et accédez aux widgets de l'activité via **binding**.

Évaluation formative 05 Indiquez aux étudiants qu'ils peuvent récupérer l'énoncé de l'exercice (en format PDF) sur le portail éducatif du collège. Ils devraient pouvoir solutionner l'exercice sans aide en se référant aux chapitres 18 et 33 du livre de référence