Le 02/10/2023

Développement mobile

Enseirb-Matmeca,
Filière Info, GL 2023

BORDEAUX INP Enseirb-Matmeca

Marc Peysale

Bordeaux INP ENSEIRB MATMECA

Promotion 2015
Informatique
TM

SCALIAN

Consultant
Développeur mobile

Snapp'

Lead développeur
iOS & Android

niji

Mobile Team Lead
Architecte logiciel

Sommaire

1. Déroulement du module
2. Historique
3. Etat de l’art actuel
4. Eléments
5. Git

But du module

- Découvrir les bases du développement d’applications mobiles
- Savoir utiliser une API HTTP
- Utiliser Git sur un projet concret

Déroulement du module

Agenda

- 1 TD d’introduction de 2h
- 5 séances de TP de 2h

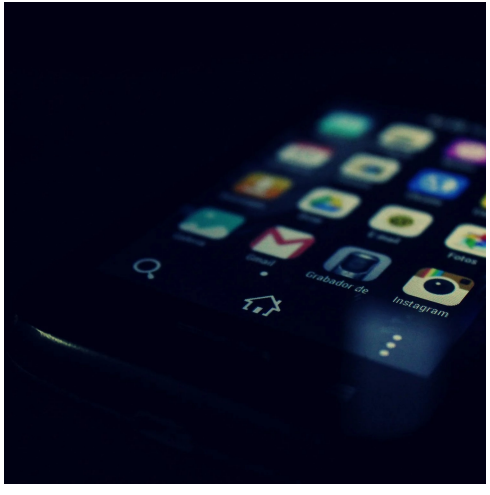
- Evaluation : projet
  - 6 séances de 2h réservées et travail en autonomie
  - Rendu : projet complet sur Github public

Séance 1 – Mise en contexte

- Historique développement applications mobiles
  - iPhone en 2007, Android en 2008, Frameworks cross-platform, web, puis pseudo-natif (React-Native, Flutter)
- Etat de l'art actuel et exemples pour chaque technologie
- Eléments iOS
  - Langage Swift (& Objective-C)
  - IDE Xcode
  - UIKit vs SwiftUI
- Eléments Android (& Java)
  - Langage Kotlin
  - IDE Android Studio
  - XML vs Jetpack Compose
- Eléments Flutter
  - Langage Dart
  - IDE Android Studio ou VS Code
  - Widgets
- Généralités apps mobiles
  - Clients lourds
  - Patterns MVC, MVVM & autres
  - Consommation API Web, architecture 3-tier
  - REST
  - Stockage local
- Git
  - Présentation de Git
  - CLI
  - SourceTree
  - Git Flow

Séance 2 – Bases d'Android

Découverte d'Android Studio, bases de Kotlin, lancement d'une app et premières vues

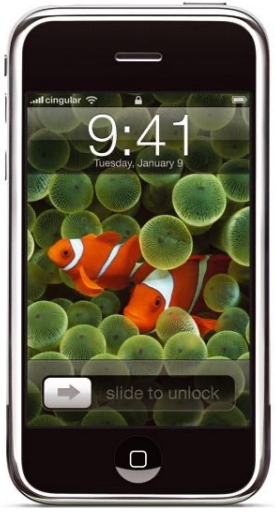


- Installation de l'IDE
- Découverte et création d'un projet
- Layout XML vs Compose
- Présentation du TP
  - Affichage d'une liste de planètes et détail de chaque planète
  - API SWAPI
  - UI, appels réseau, architecture
- Lancement de l'app sur un émulateur

Historique

Un peu d’histoire

2007 : Apple présente l’iPhone
2008 : Ouverture de l’App Store et mise à disposition du SDK iPhone OS
2008 : HTC présente le Dream, premier smartphone sous Android 1.0, compatible avec l’Android Market
2011 : Xamarin crée le premier SDK multi-plateformes
2011 : Adobe crée PhoneGap, puis le rend open-source sous le nom Apache Cordova
2015 : Facebook sort la v1.0 de React Native
2016 : Microsoft rachète Xamarin
2018 : Google sort Flutter
2022 : Microsoft sort .Net 6, et MAUI, la fin du support Xamarin est prévue pour 2023

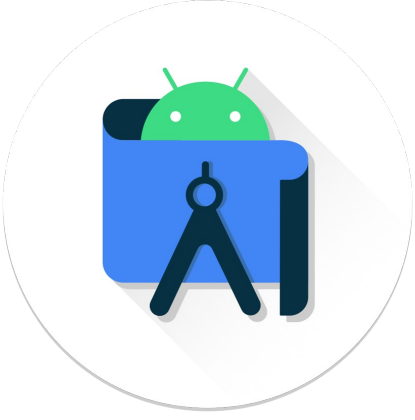


Développement natif iOS

IDE : Xcode
Langages : Objective-C (historique), puis Swift
Frameworks UI : UIKit, puis SwiftUI

Développement natif Android

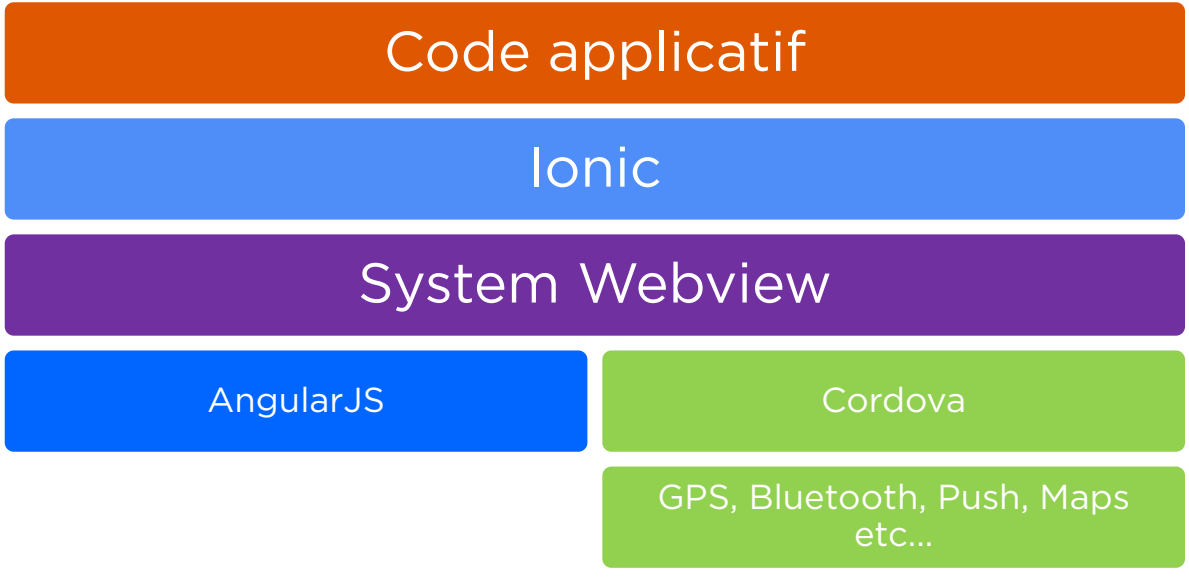
IDE : Eclipse, avec plugin Android (historique), puis Android Studio
Langages : Java (historique), puis Kotlin
Frameworks UI : XML Layout, puis Jetpack Compose



Développement X-platform

Ionic : Sorti en 2011

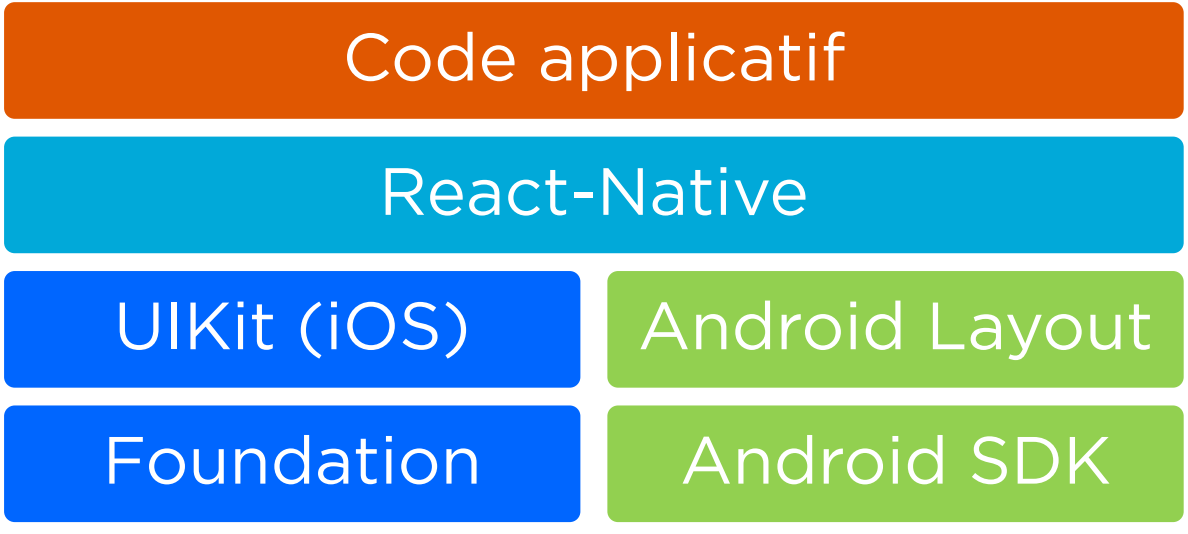
Ionic = Cordova + AngularJS + Thèmes mobiles

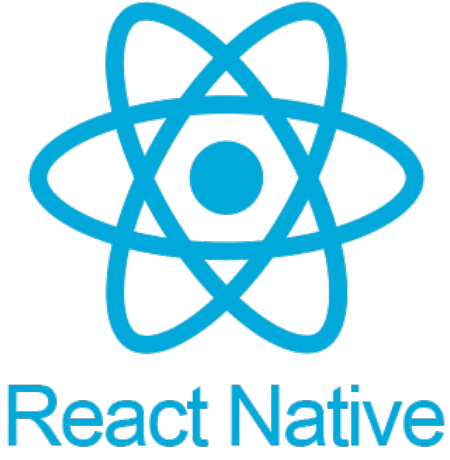


Développement X-platform

React Native : Sorti en 2015

Couche d'abstraction écrite en JS pour piloter des objets natifs.

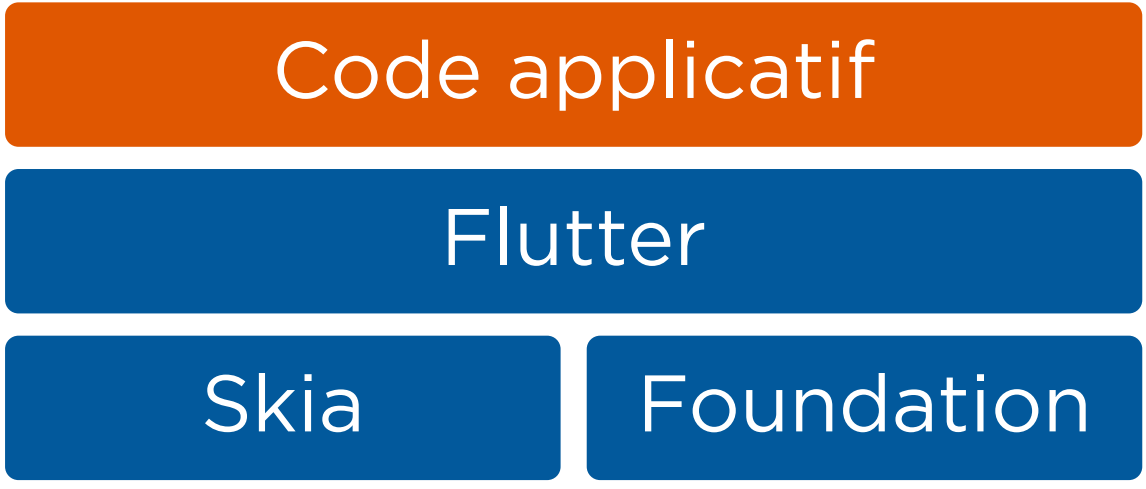




Développement X-platform

Flutter : Sorti en 2018

Réimplémentation de l'ensemble des vues/composants natifs en OpenGL, et couche abstraction



Développement X-platform

KMM: Stable fin 2023 ?

Abstraction Kotlin pour créer la partie Business de l'app sur iOS et Android, les vues restes développées en natif

Code applicatif KMM

UIKit/SwiftUI (iOS)

Android Layout/Jetpack Compose

Synthèse

Pourquoi choisir un développement Web, natif ou cross-platform ?

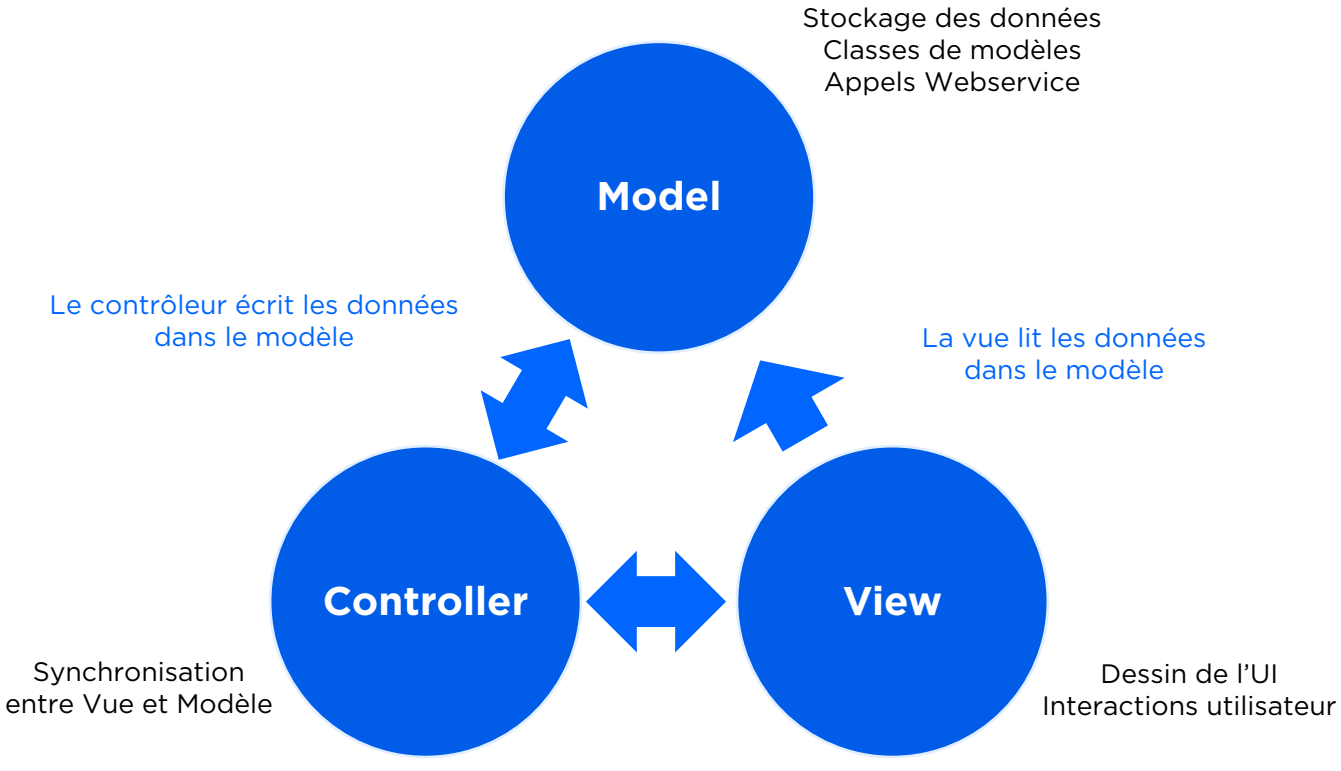
- Développement natif :
  - Implique 2 bases de code différentes
    - Nécessite une synchronisation entre les équipes
    - Coûts de développement importants
  - Permet de développer rapidement les dernières nouveautés des OS (ML, Widgets, Réalité augmentée etc…)
  - Qualité perçue optimale

- Développement cross-platform
  - Mutualisation des développements
    - Coûts de développement et de recette réduits
  - Compromis sur la qualité des apps
  - TTM (Time-to-Market) raccourci
  - Expérience utilisateur dégradée

- Développement Web
  - Site web responsive
  - Pas besoin de maintenir un client lourd en plus du web
  - Possibilité de créer une PWA

Architectures

Pattern d’architecture logicielle : MVC

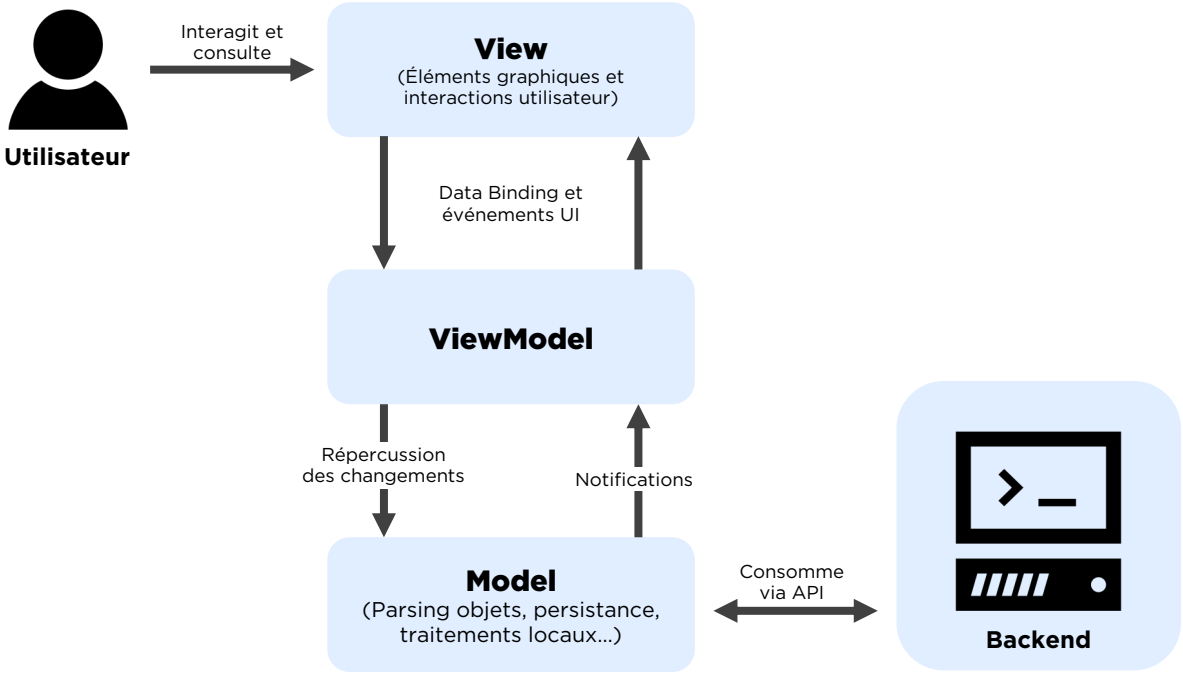


L'architecture MVVM

Apple et Google préconisent l'implémentation du pattern MVVM dans les applications, conformément aux bonnes pratiques actuelles

MVVM (Model-View-ViewModel) est le pattern d'architecture de code conçu pour les apps mobiles modernes.
Les frameworks de développements actuels (SwiftUI, Jetpack Compose) reposent dessus.

Le respect des principes du MVVM permet un découpage clair et évolutif du code, ainsi qu'une modularisation afin d'obtenir un code réutilisable et testable pour une qualité optimale.



API HTTP

Verbes : GET/POST/PATCH/PUT/DELETE
URL
Query Parameters
Body
Headers

POST https://example.com/accounts?authenticate=true
Accept: application/json
Content-Type: application/json

```
{
   "username": "test@test.com",
   "password": "test1234"
}
```

API HTTP

Authentification via headers

Méthodes standard :

- Basic auth :
  Authorization: `Basic␣hash`
  Avec hash = base64("login:mdp")

- Bearer auth :
  Authorization: `Bearer␣token`

Git

Outil de versioning et travail collaboratif

Créé par Linus Torvalds en 2005

Git

Gestion de branches

[master] 6c6faa5 My first commit - John Doe

[develop] 3e89ec8 Develop a feature - part 1 - John Doe

[develop] e188fa9 Develop a feature - part 2 - John Doe

[master] 665003d Fast bugfix - John Fixer

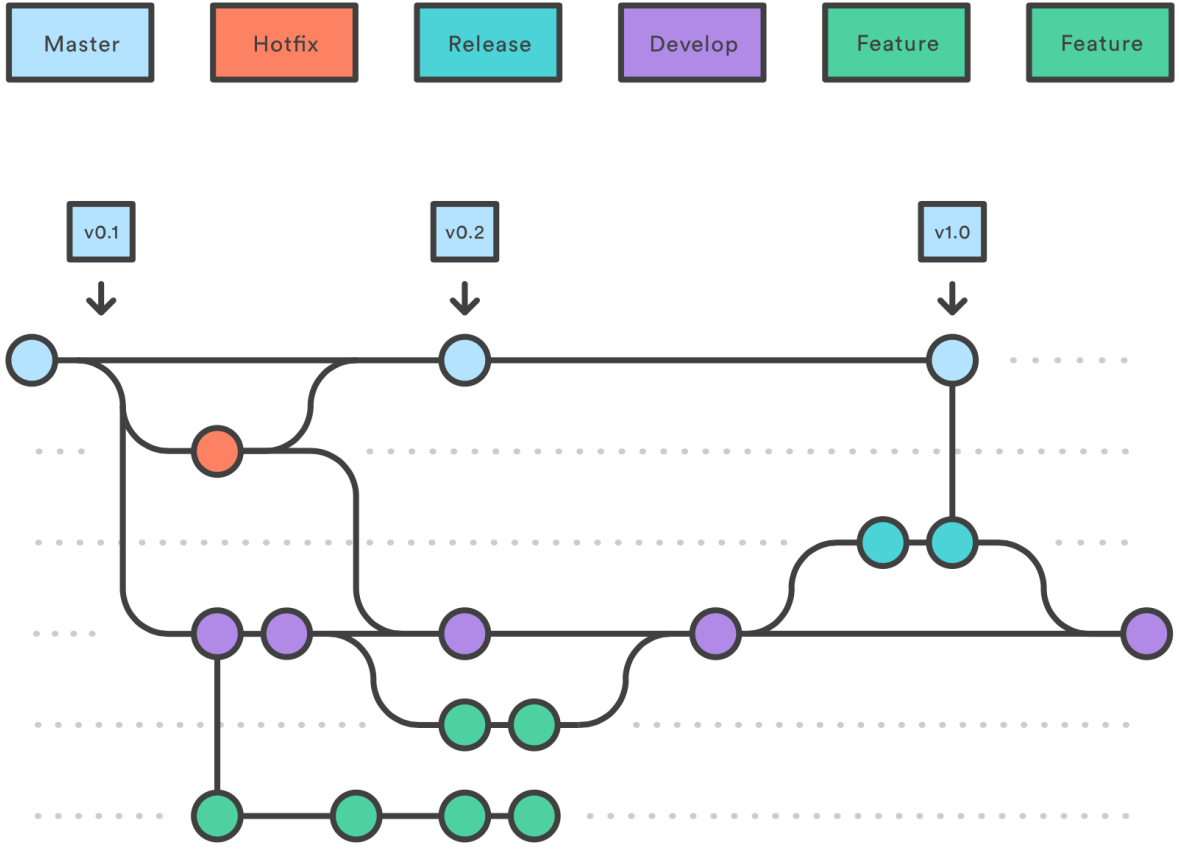
[myfeature] eaf618c New cool feature - John Feature

[master] 8f1e0e7 Merge branch `develop` into `master` - John Doe

[master] 6a3dacc Merge branch `myfeature` into `master` - John Doe

[master] abcdef0 Release of version 0.1 - John Releaser

Git – Git Flow



https://marc.peysale.com/Seance_1.pdf

Marc PEYSALE
Mobile Team Lead
M : +33 6 31 77 86 65
@ : marc.peysale@niji.fr

niji.fr

Le 10/10/2023

Développement mobile

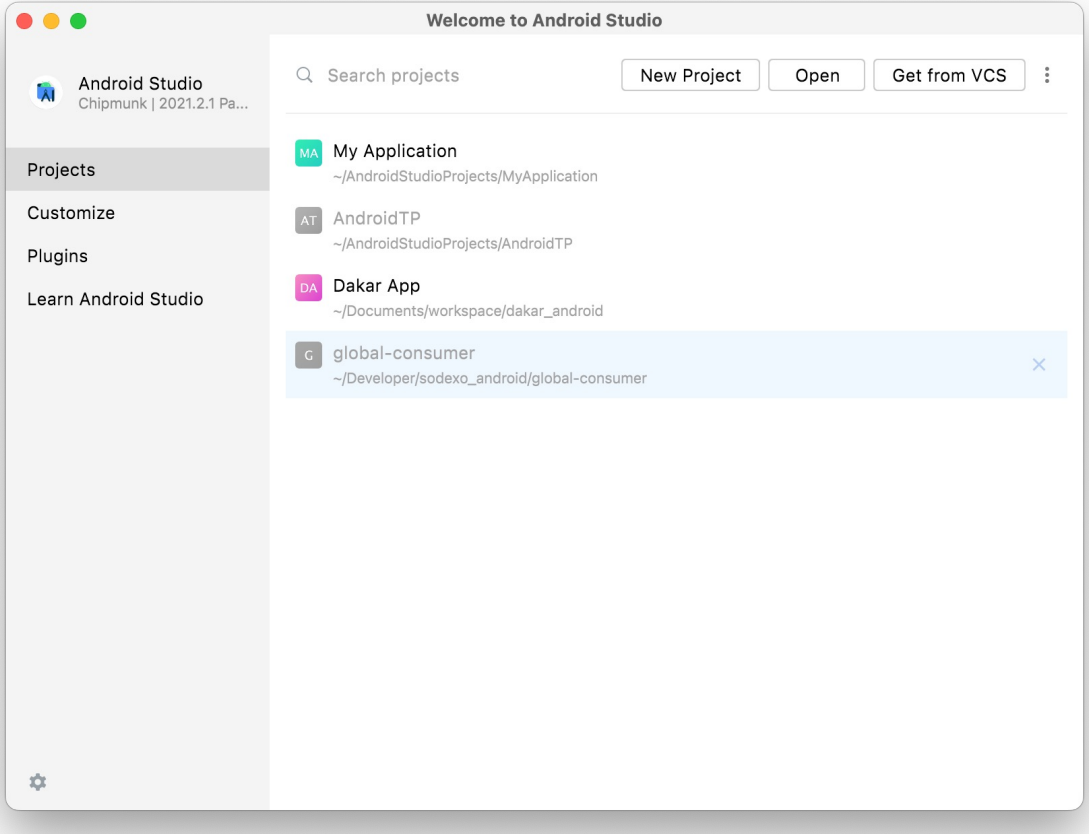
Enseirb-Matmeca,
Filière Info, GL 2023

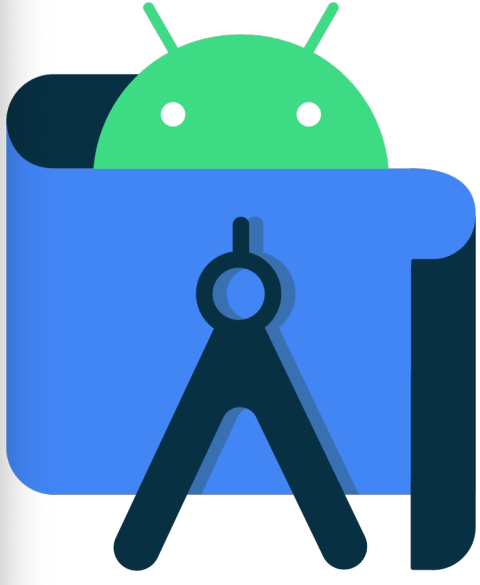
BORDEAUX INP Enseirb-Matmeca

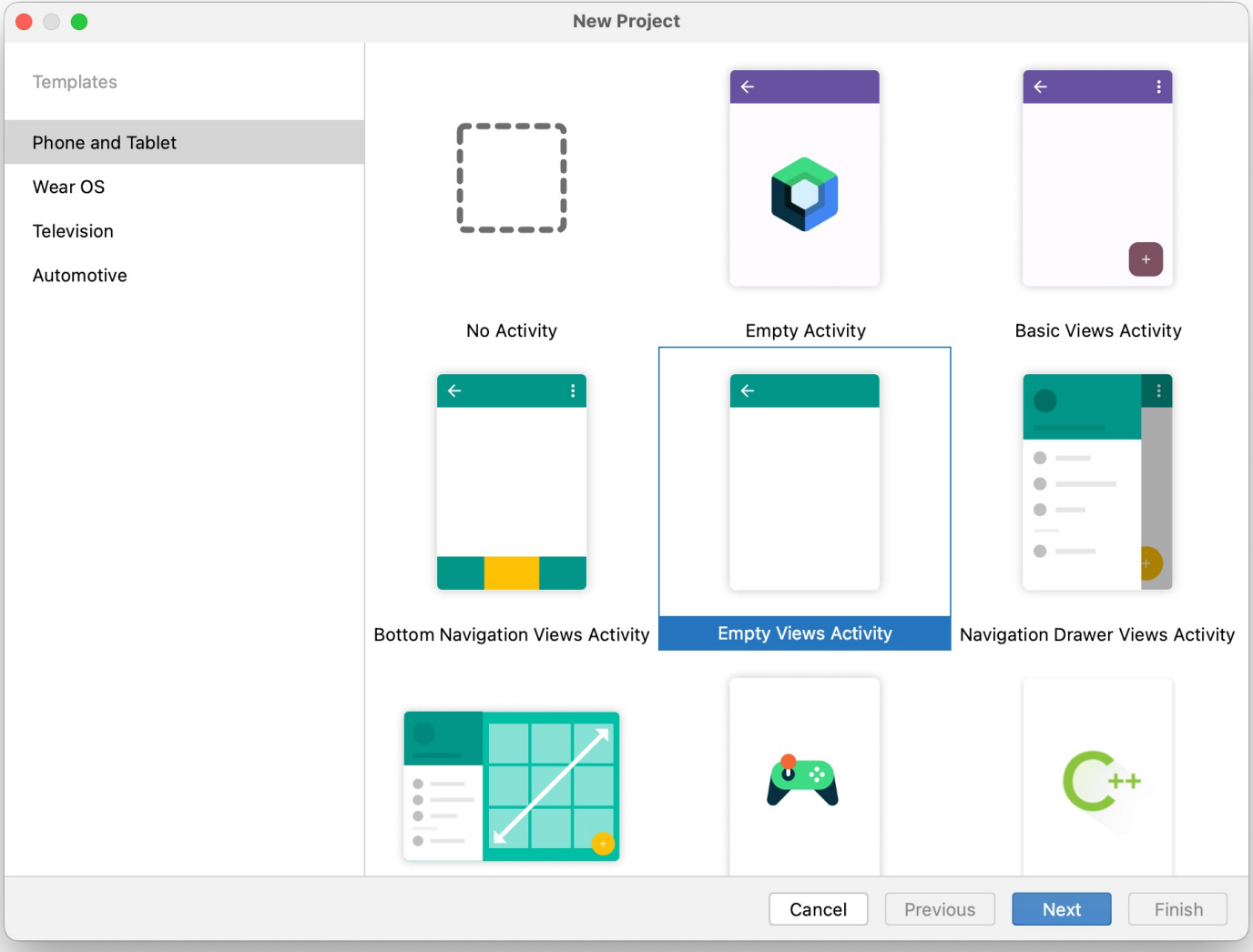
Sommaire

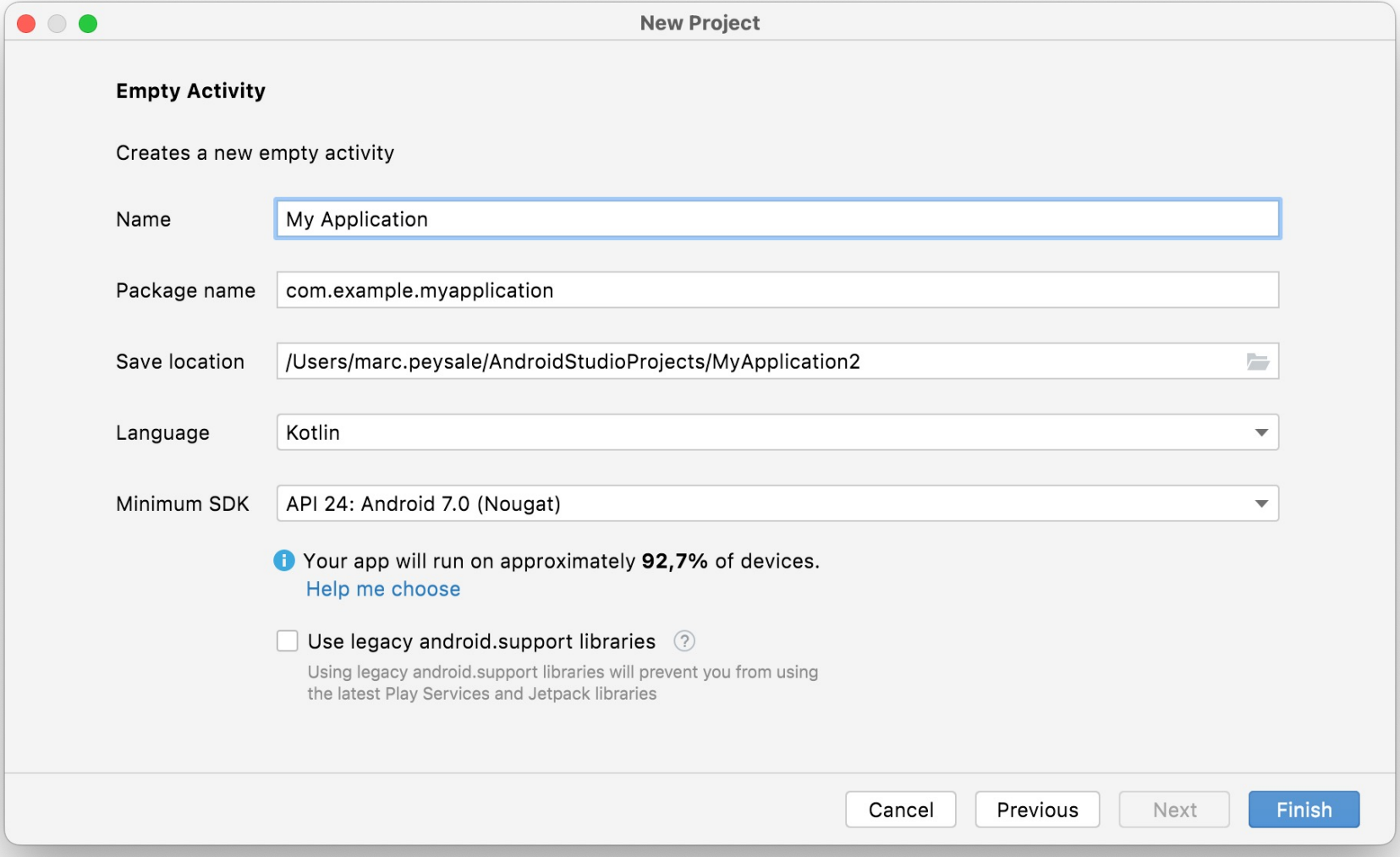
1. Android Studio
2. Gradle
3. Activity
4. Layouts + Binding
5. Intents

Android Studio






New Project
Templates
Phone and Tablet
Wear OS
Television
Automotive
No Activity
Empty Activity
Basic Views Activity
Bottom Navigation Views Activity
Empty Views Activity
Navigation Drawer Views Activity
Cancel
Previous
Next
Finish


New Project
Empty Activity
Creates a new empty activity
Name
My Application
Package name
com.example.myapplication
Save location
/Users/marc.peysale/AndroidStudioProjects/MyApplication2
Language
Kotlin
Minimum SDK
API 24: Android 7.0 (Nougat)
Your app will run on approximately 92,7% of devices.
Help me choose
Use legacy android.support libraries
Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries
Cancel
Previous
Next
Finish

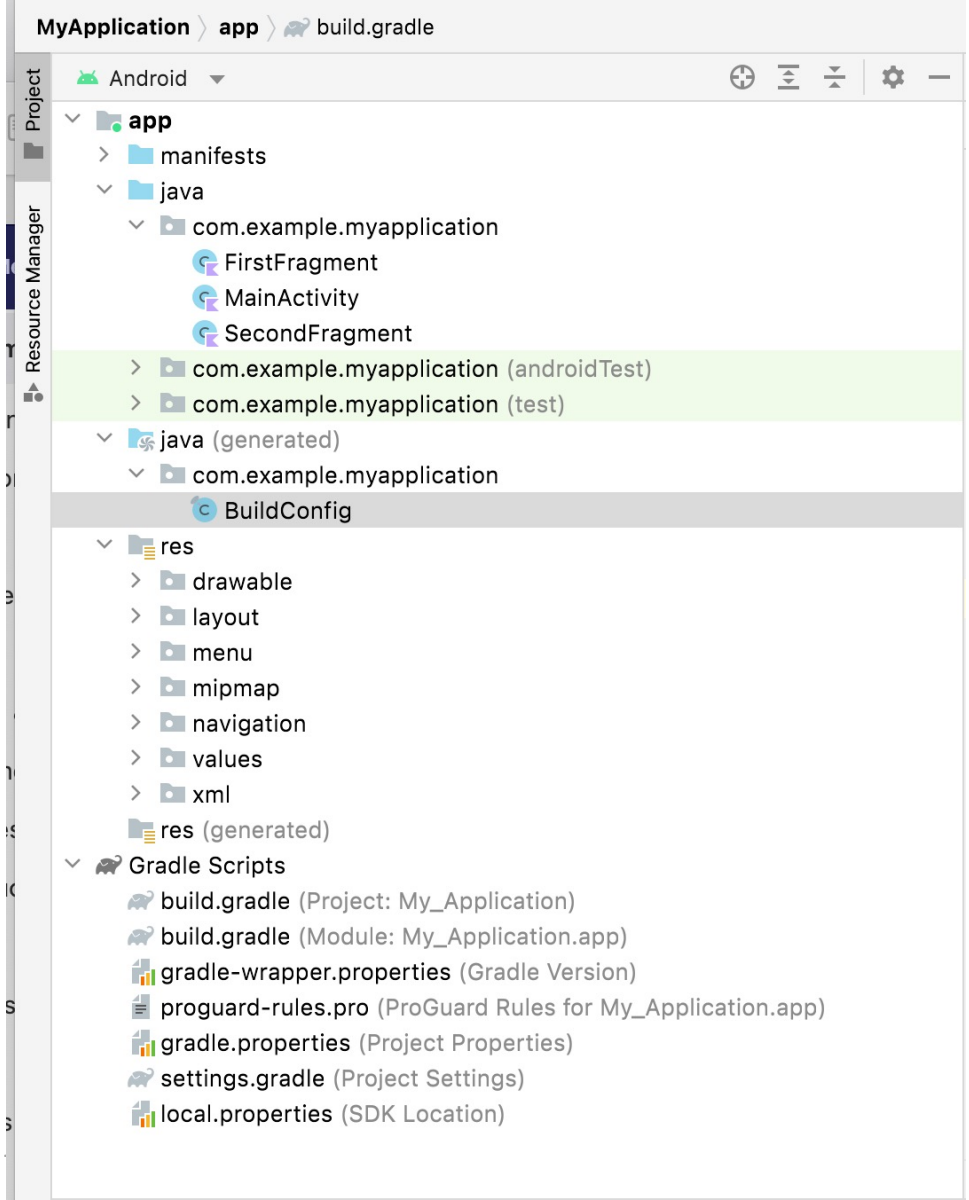
Gradle

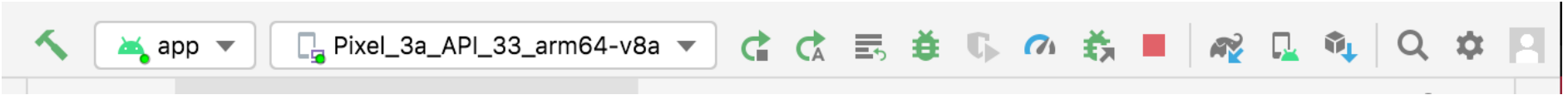
- Moteur de production Java/Kotlin
- Code en Groovy ou Kotlin

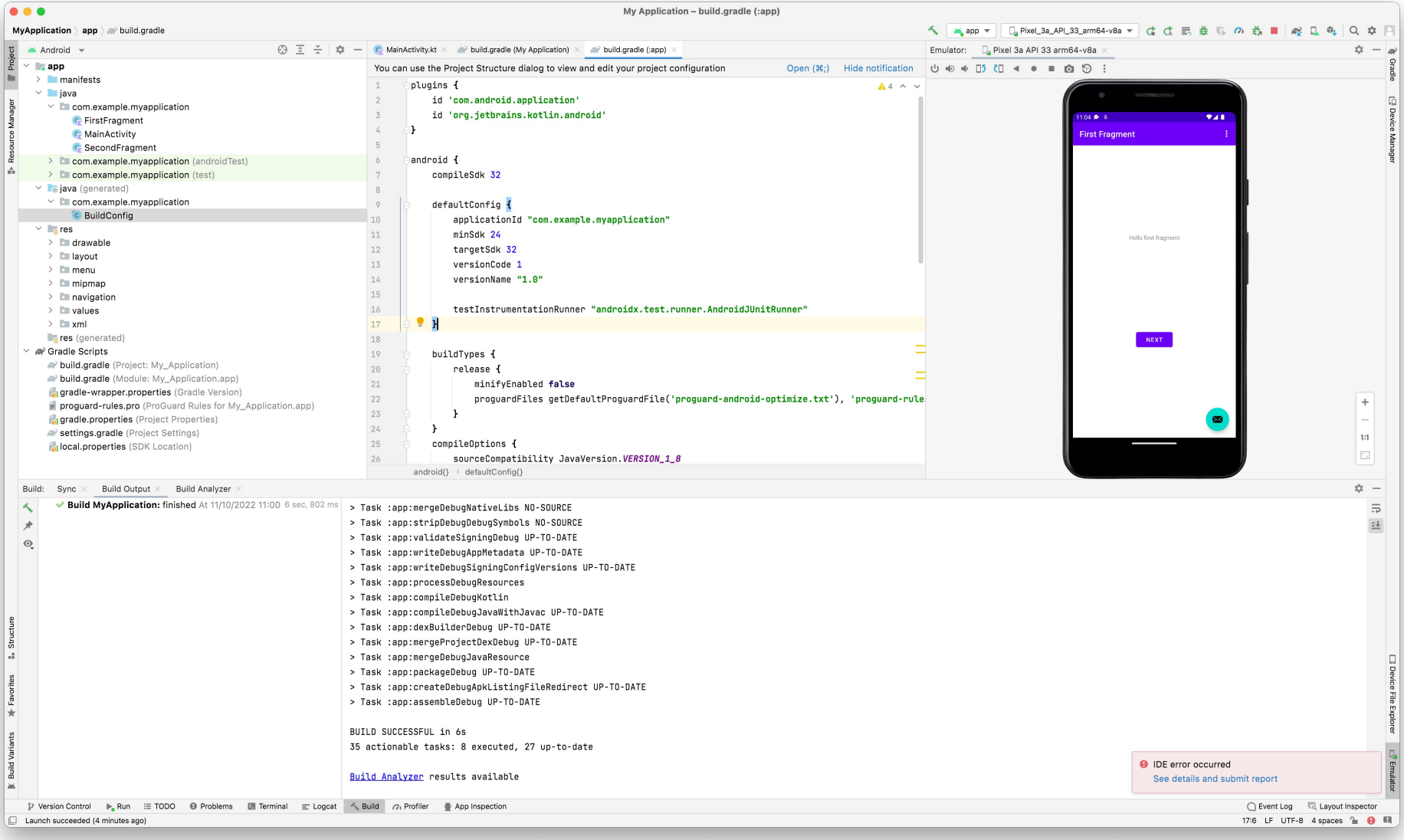
- Gestion de plugins et dépendances

- Succède à Maven/Ant

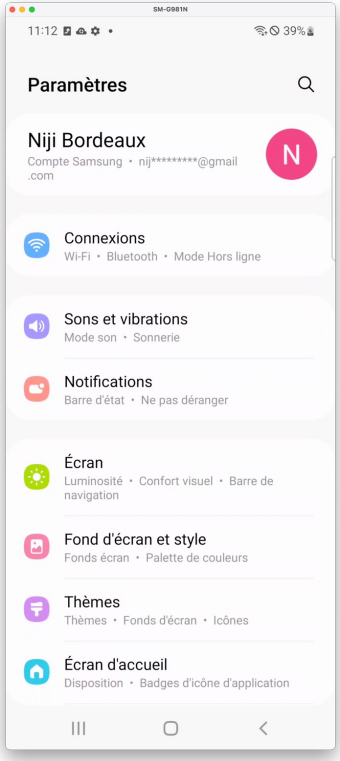
`build.gradle`

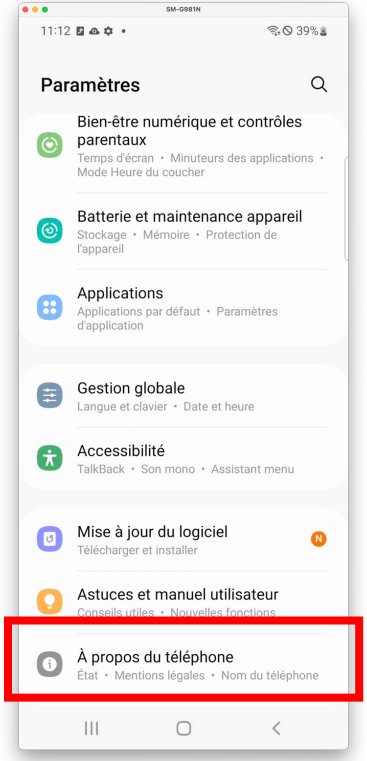

MyApplication
app
build.gradle
Project
Resource Manager
Android
app
manifests
java
com.example.myapplication
FirstFragment
MainActivity
SecondFragment
com.example.myapplication (androidTest)
com.example.myapplication (test)
java (generated)
com.example.myapplication
BuildConfig
res
drawable
layout
menu
mipmap
navigation
values
xml
res (generated)
Gradle Scripts
build.gradle (Project: My_Application)
build.gradle (Module: My_Application.app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for My_Application.app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)

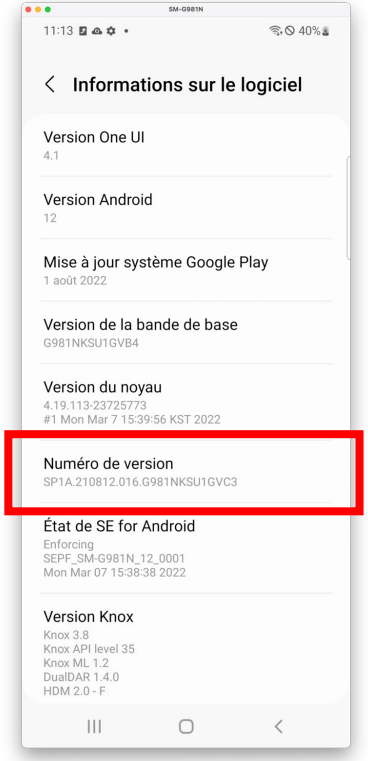



My Application – build.gradle (:app)
MyApplication 〉 app 〉 build.gradle
Android
app
manifests
java
com.example.myapplication
FirstFragment
MainActivity
SecondFragment
com.example.myapplication (androidTest)
com.example.myapplication (test)
java (generated)
com.example.myapplication
BuildConfig
res
drawable
layout
menu
mipmap
navigation
values
xml
res (generated)
Gradle Scripts
build.gradle (Project: My_Application)
build.gradle (Module: My_Application.app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for My_Application.app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
MainActivity.kt
build.gradle (My Application)
build.gradle (:app)
You can use the Project Structure dialog to view and edit your project configuration
Open (⌘;)
Hide notification
plugins {
id 'com.android.application'
id 'org.jetbrains.kotlin.android'
}
android {
compileSdk 32
defaultConfig {
applicationId "com.example.myapplication"
minSdk 24
targetSdk 32
versionCode 1
versionName "1.0"
testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
}
buildTypes {
release {
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rule
}
}
compileOptions {
sourceCompatibility JavaVersion.VERSION_1_8
android{} 〉 defaultConfig{}
Emulator:
Pixel 3a API 33 arm64-v8a
First Fragment
Hello first fragment
NEXT
Build:
Sync
Build Output
Build Analyzer
Build MyApplication: finished At 11/10/2022 11:00 6 sec, 802 ms
> Task :app:mergeDebugNativeLibs NO-SOURCE
> Task :app:stripDebugDebugSymbols NO-SOURCE
> Task :app:validateSigningDebug UP-TO-DATE
> Task :app:writeDebugAppMetadata UP-TO-DATE
> Task :app:writeDebugSigningConfigVersions UP-TO-DATE
> Task :app:processDebugResources
> Task :app:compileDebugKotlin
> Task :app:compileDebugJavaWithJavac UP-TO-DATE
> Task :app:dexBuilderDebug UP-TO-DATE
> Task :app:mergeProjectDexDebug UP-TO-DATE
> Task :app:mergeDebugJavaResource
> Task :app:packageDebug UP-TO-DATE
> Task :app:createDebugApkListingFileRedirect UP-TO-DATE
> Task :app:assembleDebug UP-TO-DATE
BUILD SUCCESSFUL in 6s
35 actionable tasks: 8 executed, 27 up-to-date
Build Analyzer results available
IDE error occurred
See details and submit report
Version Control
Run
TODO
Problems
Terminal
Logcat
Build
Profiler
App Inspection
Event Log
Layout Inspector
Launch succeeded (4 minutes ago)
17:6
LF
UTF-8
4 spaces

Mode debug

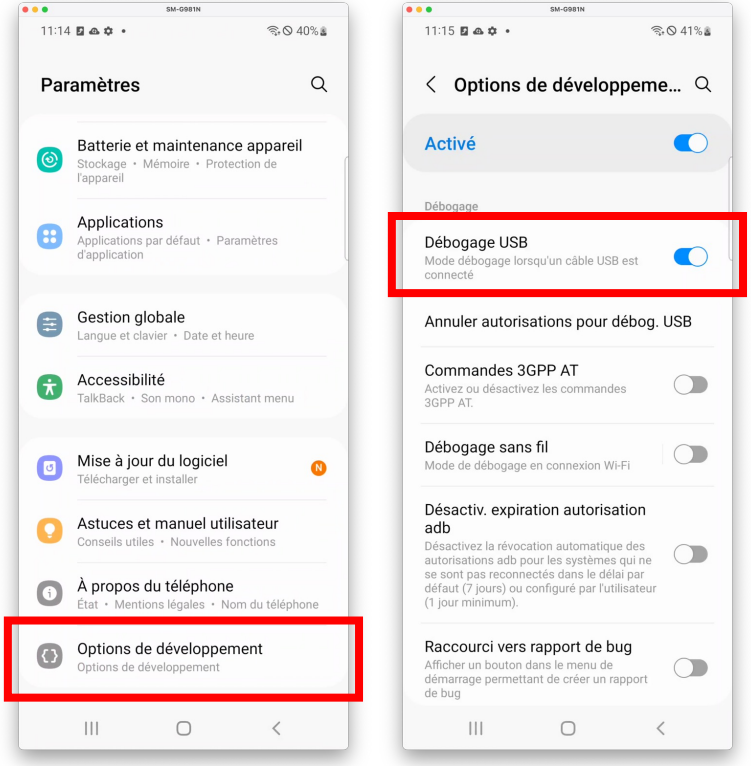


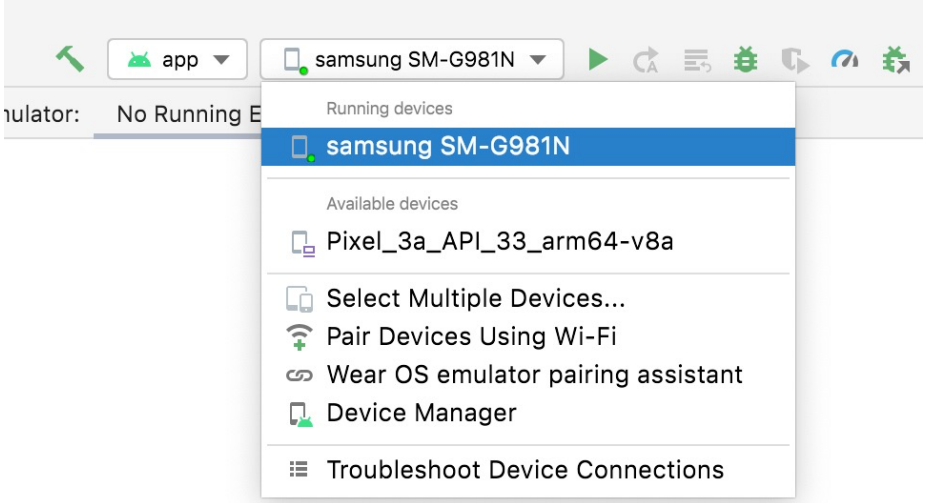




7x

Mode debug





Activity

- Activity = écran
- 1 Activity = 1 classe Java/Kotlin + 1 Layout XML

ViewBinding

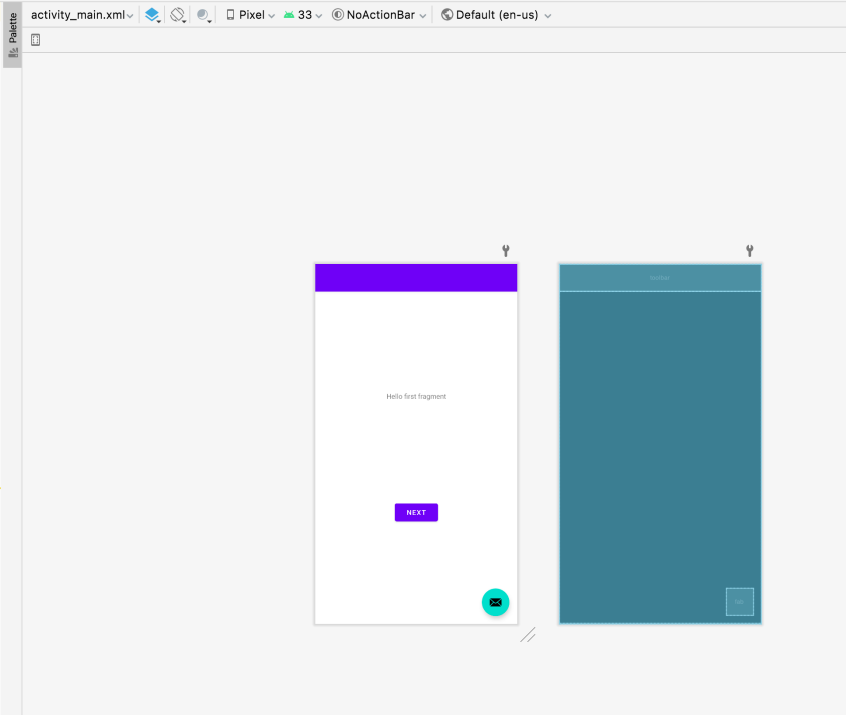
- Lien entre les vues dans le layout XML et le code applicatif Java/Kotlin

ViewBinding

```
private lateinit var appBarConfiguration: AppBarConfiguration
private lateinit var binding: ActivityMainBinding
```

ViewBinding

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout 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">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/Theme.MyApplication.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/Theme.MyApplication.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_main" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        app:srcCompat="@android:drawable/ic_dialog_email" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```



ViewBinding

```
binding.fab.setOnClickListener { view ->
    Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
        .setAction( text: "Action",  listener: null).show()
}
```

ViewBinding

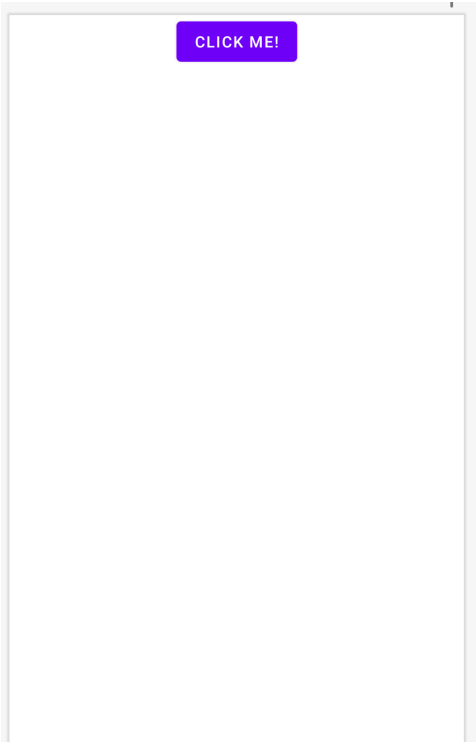
```
buildFeatures { viewBinding = true }
```

ViewBinding

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

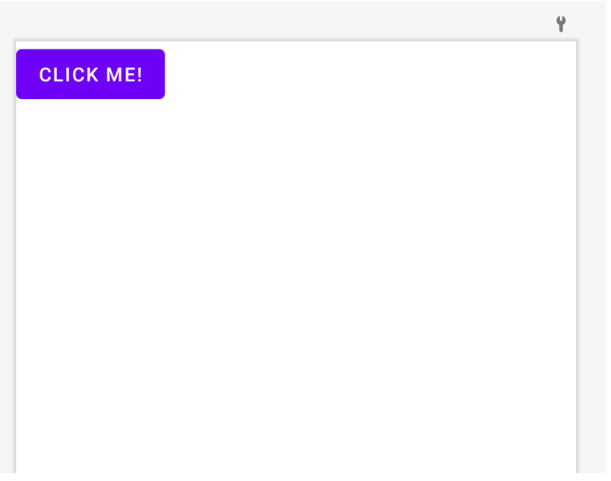
ConstraintLayout

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click me!"
    app:layout_constraintLeft_toLeftOf="@+id/root_layout"
    app:layout_constraintRight_toRightOf="@+id/root_layout"
    android:id="@+id/activity_button" />
```



ConstraintLayout

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click me!"
    app:layout_constraintLeft_toLeftOf="@+id/root_layout"
    android:id="@+id/activity_button" />
```



Accessibilité

- Toutes les langues ne se lisent pas de gauche à droite !

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click me!"
    app:layout_constraintStart_toStartOf="@+id/root_layout"
    app:layout_constraintEnd_toEndOf="@+id/root_layout"
    android:id="@+id/activity_button" />
```

Accessibilité

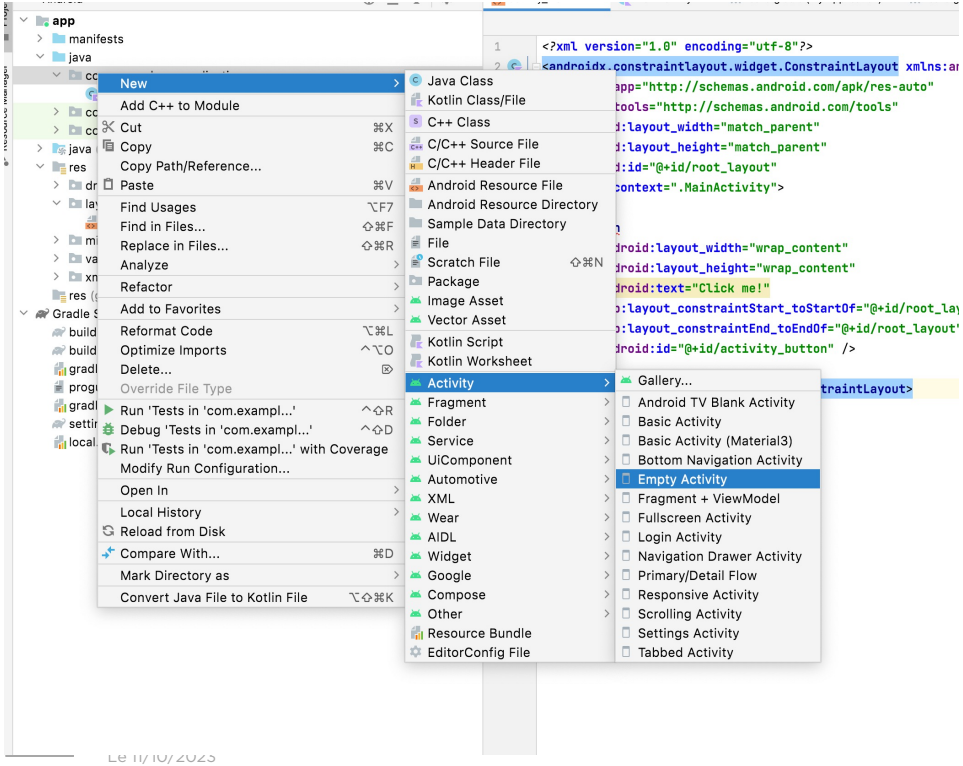
- Toutes les langues ne sont pas aussi « volumineuses »

- Welcome dear user
- Bienvenue cher utilisateur
- Willkommen lieber Benutzer

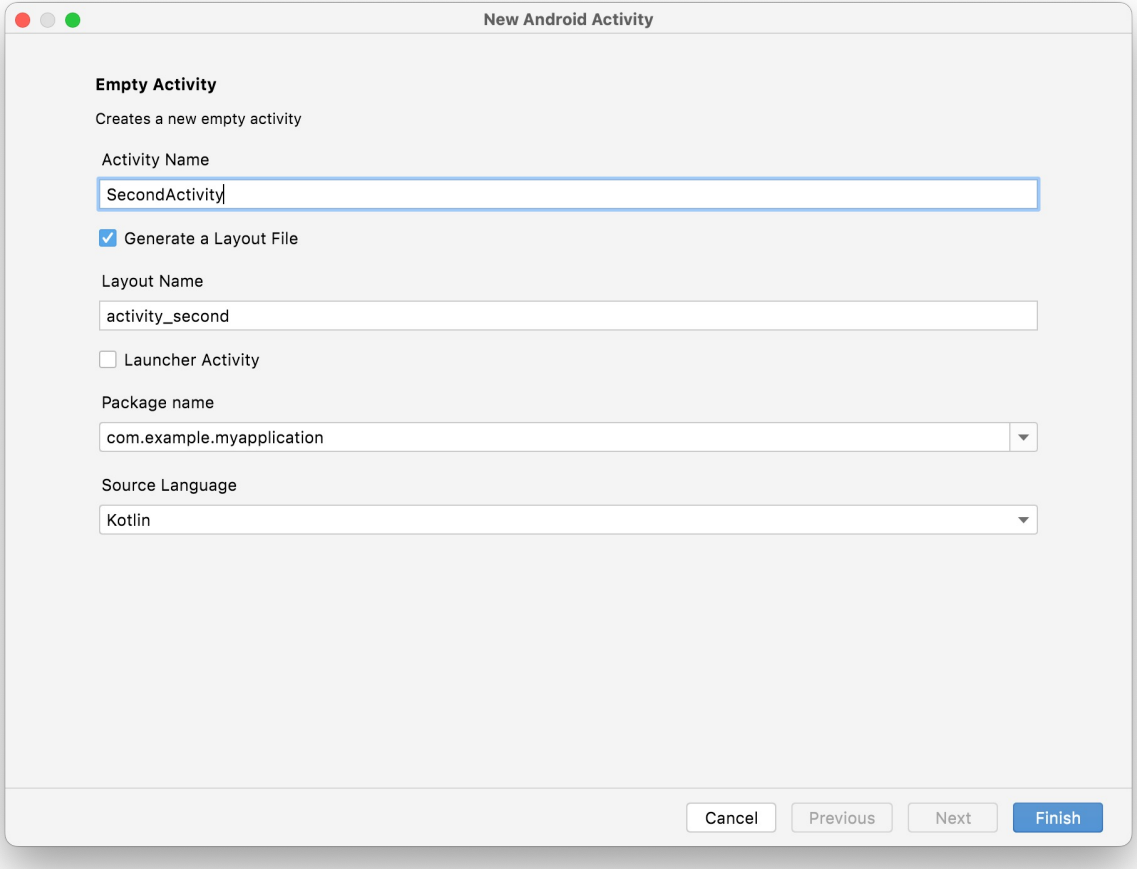
- Attention aux largeurs fixes dans les layouts

Intents

- Moyen de communiquer vers une `Activity`



Intents



Intents

```
binding.activityButton.setOnClickListener { it: View!
    val intent = Intent(applicationContext, SecondActivity::class.java)
    startActivity(intent)
}
```

Intents

Passage de paramètres

```
binding.activityButton.setOnClickListener {  it: View!
    val intent = Intent(applicationContext, SecondActivity::class.java)
    intent.putEx
    star
}
```

```
m putExtra(name: String!, value: Int)                    Intent
m putExtra(name: String!, value: Byte)                   Intent
m putExtra(name: String!, value: Char)                   Intent
m putExtra(name: String!, value: Long)                   Intent
m putExtra(name: String!, value: Float)                  Intent
m putExtra(name: String!, value: Short)                  Intent
m putExtra(name: String!, value: Double)                 Intent
m putExtra(name: String!, value: Boolean)                Intent
m putExtra(name: String!, value: Bundle?)                Intent
m putExtra(name: String!, value: String?)                Intent
m putExtra(name: String!, value: IntArray?)              Intent
Press ^. to choose the selected (or first) suggestion and insert a dot afterwards  Next Tip
```

Intents

Passage de paramètres

```
class SecondActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_second)
        intent.getStringExtra( name: "key")
    }
}
```

Code de la séance

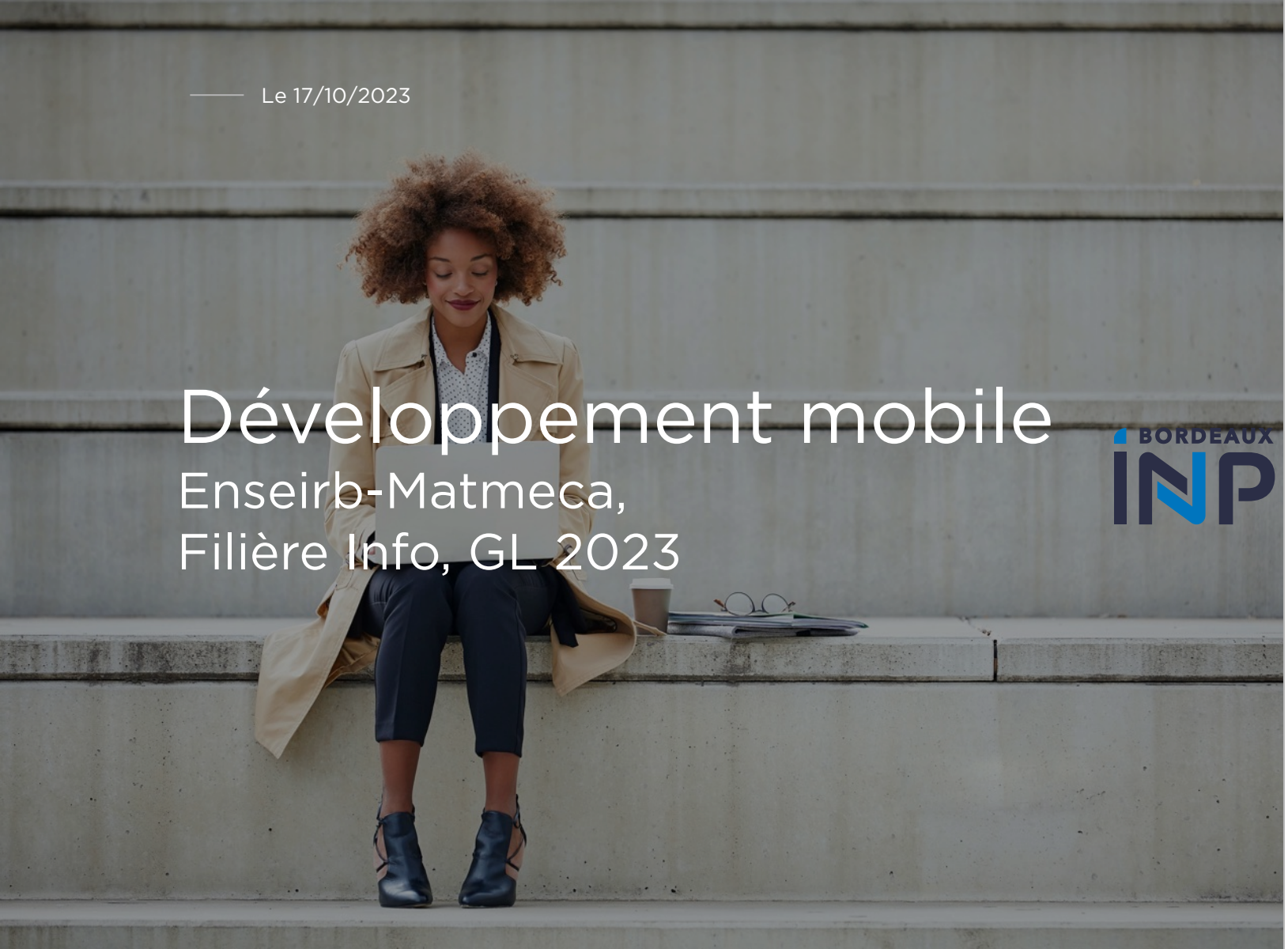
Le code est disponible sur Github

https://github.com/mpeysale-niji/seance_2_2023

Marc PEYSALE
Mobile Team Lead
M : +33 6 31 77 86 65
@ : marc.peysale@niji.fr

niji.fr

Le 17/10/2023



Développement mobile

Enseirb-Matmeca,
Filière Info, GL 2023

BORDEAUX INP Enseirb-Matmeca

Sommaire

1. RecyclerView
2. OKHTTP
3. GSON

TP précédent

https://marc.peysale.com/Seance_2.pdf

https://github.com/mpeysale-niji/seance_2_2023

TP précédent

Création d’un projet Android Studio
Découverte de Gradle
Activity
ViewBinding
Intents

RecyclerView

Affichage de listes/grilles
Eléments répétés

RecyclerView

Adapter pour gérer le contenu de la liste

Pattern de délégation

RecyclerViewAdapter

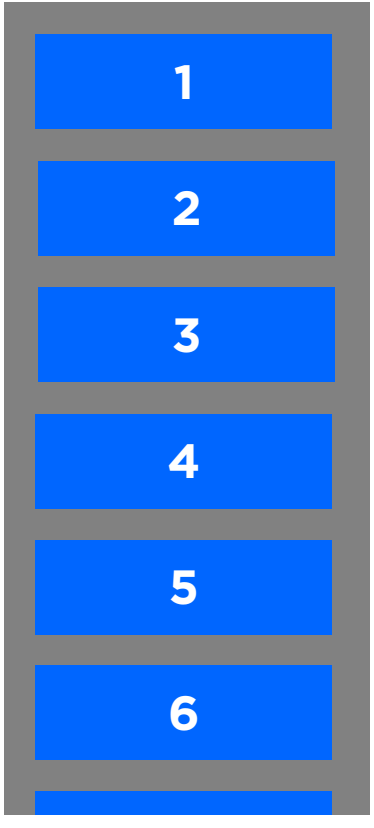
```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ListViewHolder {

}

override fun onBindViewHolder(holder: ListViewHolder, position: Int) {

}

override fun getItemCount(): Int {

}
```

RecyclerView

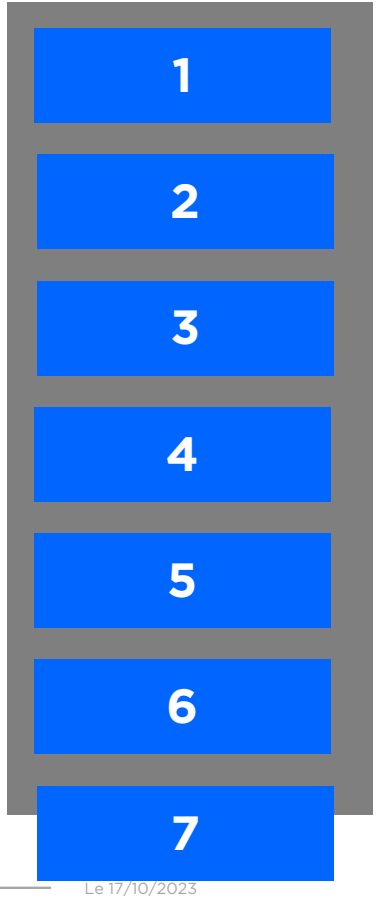
RecyclerView = Données + Flow d’affichage

Adapter + LayoutManager

RecyclerView - Recyclage



RecyclerView - Recyclage



RecyclerView - Recyclage

ViewHolder : mécanisme de rétention de cellule en mémoire

On n’instancie pas autant de cellules que d’éléments dans l’Adapter

On réutilise les cellules autant que possible

RecyclerView

Time to practice!

Networking

OKHTTP

Librairie HTTP pour Android
Permet de consommer des API facilement

Permissions & AndroidManifest.xml

Par défaut, Android bloque tout aux créateurs d'applications

Nécessité de demander des autorisations
Exemple : accès à Internet

```
<uses-permission
   android:name="android.permission.INTERNET">
```

AndroidManifest.xml

Fichier de configuration d’une app :

Liste toutes les Activity, l’Activity à lancer au démarrage, les permissions etc...

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.recyclerview">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="RecyclerView"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.RecyclerView"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="RecyclerView"
            android:theme="@style/Theme.RecyclerView.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

OKHTTP

Ajout de la dépendance dans le `build.gradle`

```
implementation "com.squareup.okhttp3:okhttp:4.9.3"
```

Puis Gradle Sync

Exemple basique

```
val url = URL(“https://example.com/api“)

val request = Request.Builder().url(url).build()

val response = client.newCall(request).execute()

result = response.body?.string()
```

Exemple basique

```
val url = URL(“https://example.com/api“)

val request = Request.Builder().url(url).build()

val response = client.newCall(request).execute()

result = response.body?.string()
```

Un appel réseau, ça prend du temps !

Exemple asynchrone

```
val url = URL(“https://example.com/api“)

val request = Request.Builder().url(url).build()


val response = client.newCall(request.enqueue(object : Callback {
   override fun onFailure(call: Call, e: IOException) {

   }

   override fun onResponse(call: Call, response: Response) {

   }
})
```

Exemple asynchrone

```
val url = URL("https://example.com/api")

val request = Request.Builder().url(url).build()

val response = client.newCall(request.enqueue(object : Callback {
    override fun onFailure(call: Call, e: IOException) {

    }

    override fun onResponse(call: Call, response: Response) {

    }
})
```

Seuls les callbacks sont appelés sur le Main Thread

GSON

Le retour de l'API est une String

On a besoin de parser cette String et de manipuler des objets métier

GSON

GSON : Librairie Google permettant de gérer du JSON

Ex :

```
gson.fromJson(myJSONString, MyClass::class.java)
```

GSON

implementation 'com.google.code.gson:gson:2.8.5'

Marc PEYSALE
Mobile Team Lead
M : +33 6 31 77 86 65
@ : marc.peysale@niji.fr

niji.fr

Le 24/10/2023

Développement mobile

Enseirb-Matmeca,
Filière Info, GL 2023

BORDEAUX INP Enseirb-Matmeca

TP précédent

RecyclerView
- Adapters
- ViewHolders

OKHTTP

Sommaire

1. GSON
2. Debugging
3. Obtenir des données d'une API
4. Material Design
5. Layouts

GSON

Le retour de l'API est une String

On a besoin de parser cette String et de manipuler des objets métier

GSON

GSON : Librairie Google permettant de gérer du JSON

Ex :

```
gson.fromJson(myJSONString, MyClass::class.java)
```

GSON

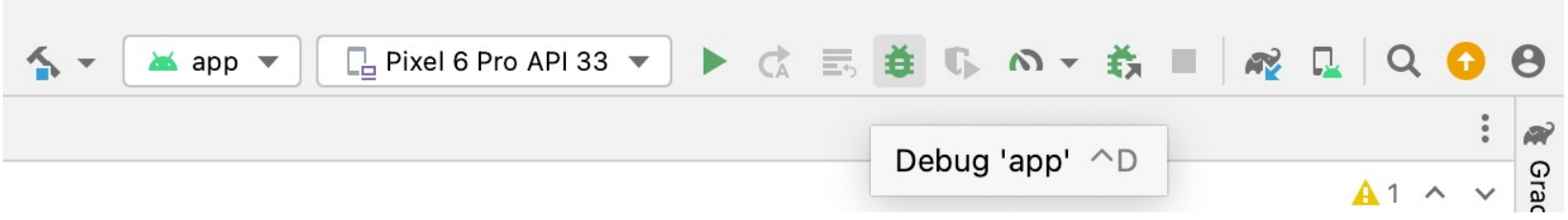
implementation ("com.google.code.gson:gson:2.10")

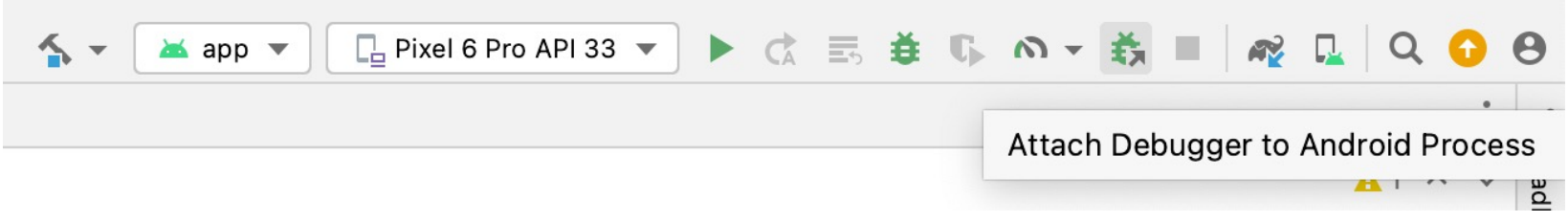
Debugging

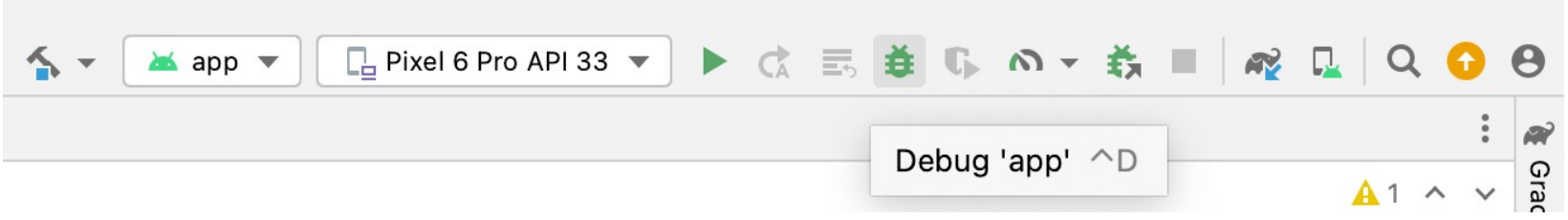
```
34      private fun performNetworkCall() {
35
36          val client = OkHttpClient()
37
38
39          val url = URL( spec: "https://example.com/api")
40          val request : Request = Request.Builder().url(url).build()
41          client
42              .newCall(request)
43              .enqueue(object: Callback {
44              override fun onFailure(call: Call, e: IOException) {
45                  Log.i( tag: "OKHTTP",  msg: "OnFailure: ${e.localizedMessage}")
46              }
47              override fun onResponse(call: Call, response: Response) {
48                  Log.i( tag: "OKHTTP",  msg: "OnSuccess")
49                  Log.i( tag: "OKHTTP", response.networkResponse.toString())
50              }
51          })
52
53      }
54
55  }
```

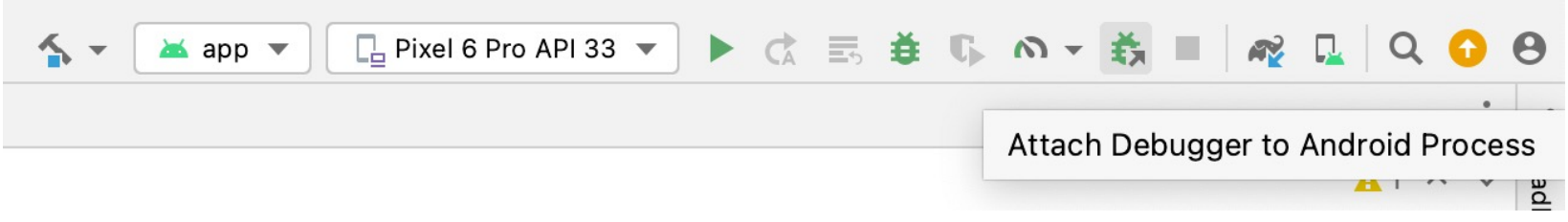
Debugging

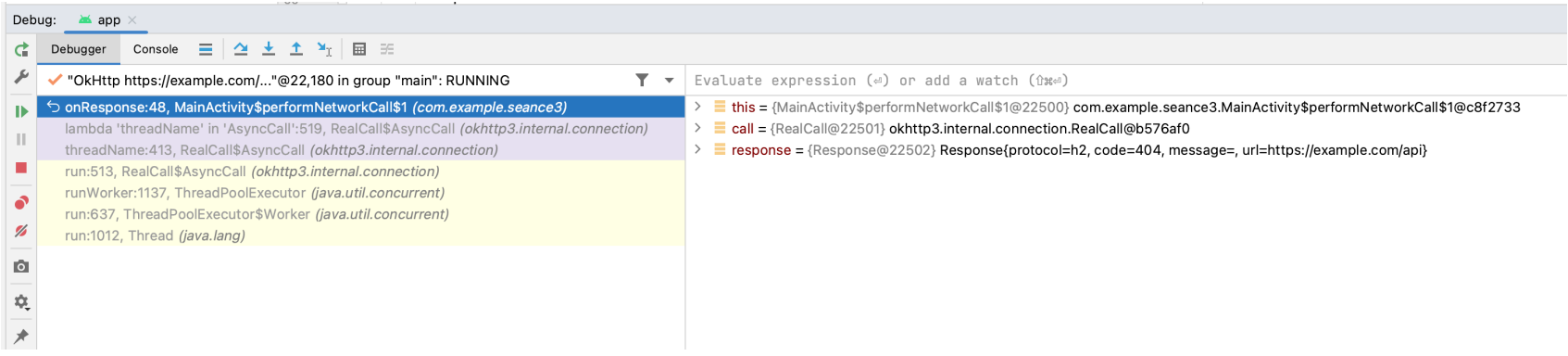
```
34      private fun performNetworkCall() {
35
36          val client = OkHttpClient()
37
38
39          val url = URL( spec: "https://example.com/api")
40          val request : Request = Request.Builder().url(url).build()
41          client
42              .newCall(request)
43              .enqueue(object: Callback {
44              override fun onFailure(call: Call, e: IOException) {
45                  Log.i( tag: "OKHTTP",  msg: "OnFailure: ${e.localizedMessage}")
46              }
47              override fun onResponse(call: Call, response: Response) {
48                  Log.i( tag: "OKHTTP",  msg: "OnSuccess")
49                  Log.i( tag: "OKHTTP", response.networkResponse.toString())
50              }
51          })
52
53      }
54
55  }
```

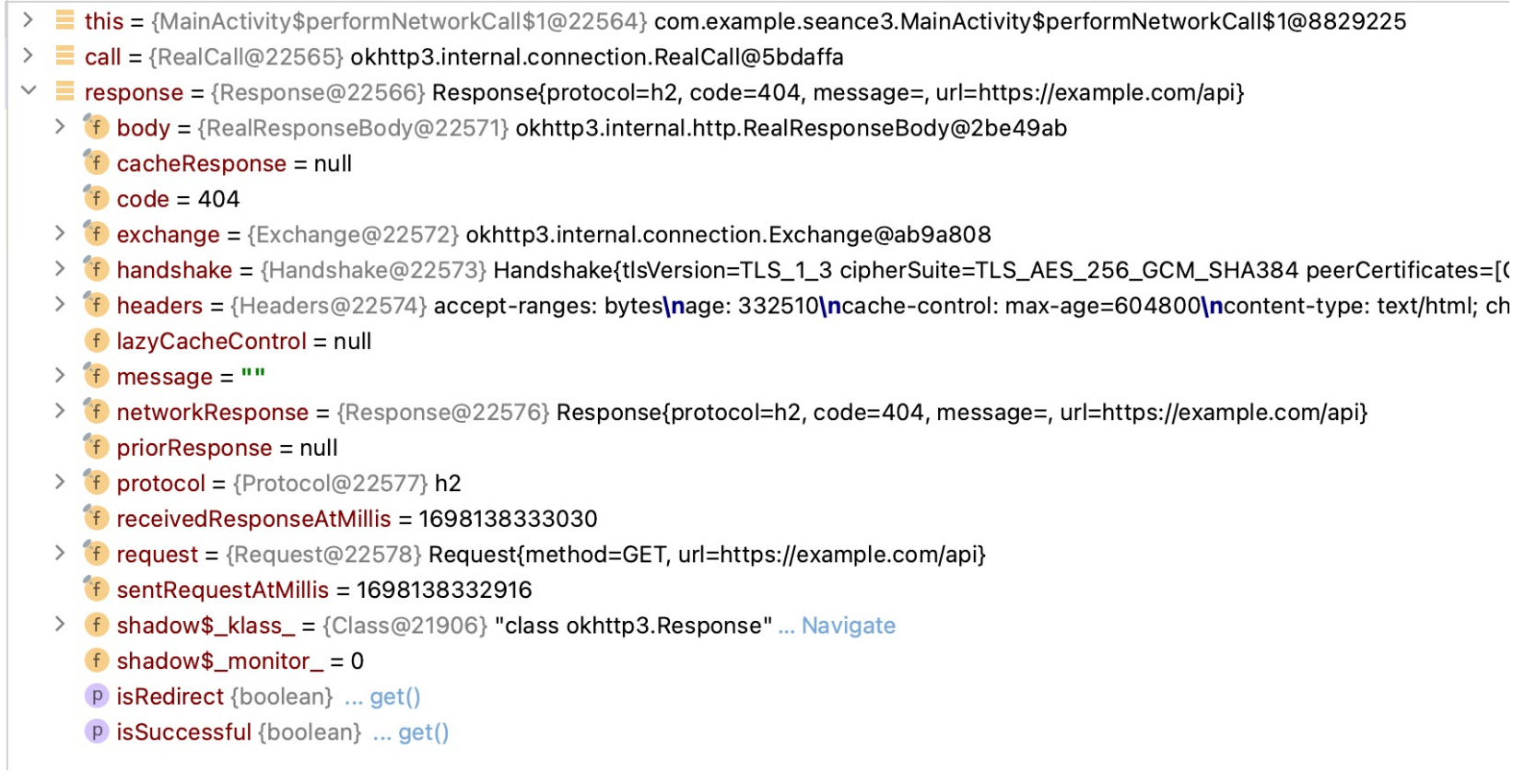

app
Pixel 6 Pro API 33
Debug 'app' ^D
1


app
Pixel 6 Pro API 33
Attach Debugger to Android Process


app
Pixel 6 Pro API 33
Debug 'app' ^D
1


app
Pixel 6 Pro API 33
Attach Debugger to Android Process


Debug:
app
Debugger
Console
"OkHttp https://example.com/..."@22,180 in group "main": RUNNING
Evaluate expression (⏎) or add a watch (⇧⌘⏎)
onResponse:48, MainActivity$performNetworkCall$1 (com.example.seance3)
lambda 'threadName' in 'AsyncCall':519, RealCall$AsyncCall (okhttp3.internal.connection)
threadName:413, RealCall$AsyncCall (okhttp3.internal.connection)
run:513, RealCall$AsyncCall (okhttp3.internal.connection)
runWorker:1137, ThreadPoolExecutor (java.util.concurrent)
run:637, ThreadPoolExecutor$Worker (java.util.concurrent)
run:1012, Thread (java.lang)
this = {MainActivity$performNetworkCall$1@22500} com.example.seance3.MainActivity$performNetworkCall$1@c8f2733
call = {RealCall@22501} okhttp3.internal.connection.RealCall@b576af0
response = {Response@22502} Response{protocol=h2, code=404, message=, url=https://example.com/api}


this = {MainActivity$performNetworkCall$1@22564} com.example.seance3.MainActivity$performNetworkCall$1@8829225
call = {RealCall@22565} okhttp3.internal.connection.RealCall@5bdaffa
response = {Response@22566} Response{protocol=h2, code=404, message=, url=https://example.com/api}
body = {RealResponseBody@22571} okhttp3.internal.http.RealResponseBody@2be49ab
cacheResponse = null
code = 404
exchange = {Exchange@22572} okhttp3.internal.connection.Exchange@ab9a808
handshake = {Handshake@22573} Handshake{tlsVersion=TLS_1_3 cipherSuite=TLS_AES_256_GCM_SHA384 peerCertificates=[(
headers = {Headers@22574} accept-ranges: bytes\nage: 332510\ncache-control: max-age=604800\ncontent-type: text/html; ch
lazyCacheControl = null
message = ""
networkResponse = {Response@22576} Response{protocol=h2, code=404, message=, url=https://example.com/api}
priorResponse = null
protocol = {Protocol@22577} h2
receivedResponseAtMillis = 1698138333030
request = {Request@22578} Request{method=GET, url=https://example.com/api}
sentRequestAtMillis = 1698138332916
shadow$_klass_ = {Class@21906} "class okhttp3.Response" ... Navigate
shadow$_monitor_ = 0
isRedirect {boolean} ... get()
isSuccessful {boolean} ... get()

The Meal DB API

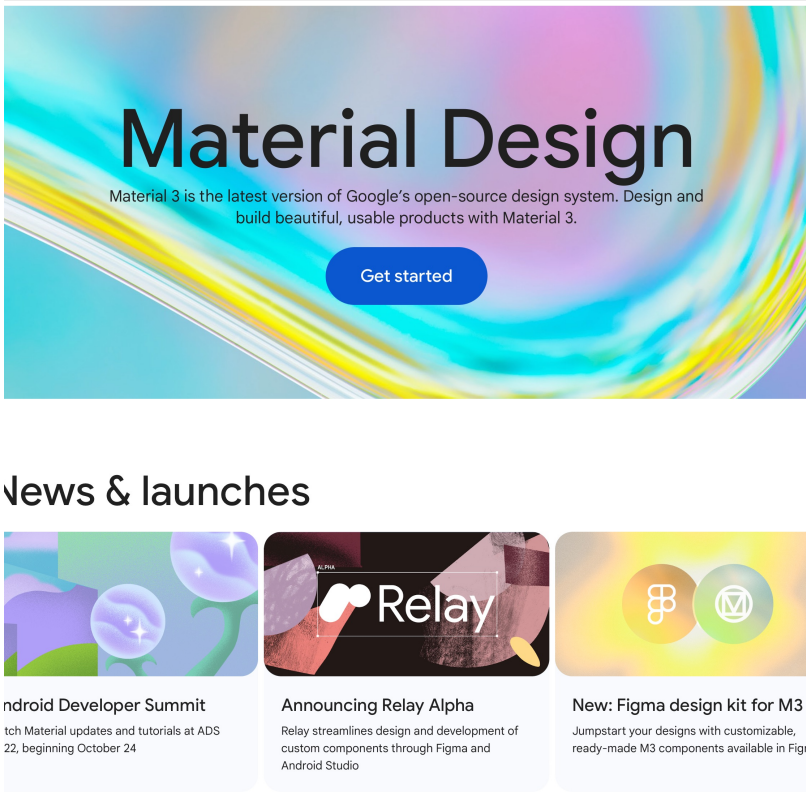
www.themealdb.com/api/json/v1/1/categories.php

Material Design

Spec de design Google

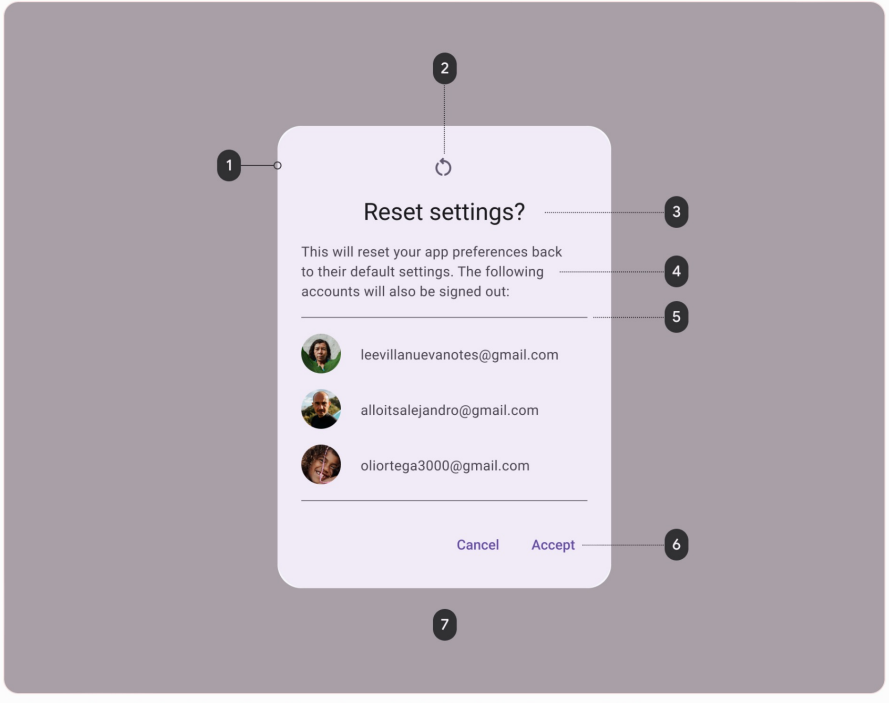
Lancée avec Android 5

material.io



Material Design

Basic dialogs



1. Container 2. Icon (optional) 3. Headline 4. Supporting text 5. Divider (optional) 6. Text button 7. Scrim

Material Design

https://m3.material.io/components

https://m3.material.io/develop/android/mdc-android

Material Design

Material Design Components (MDC)

Alerts

Alerts

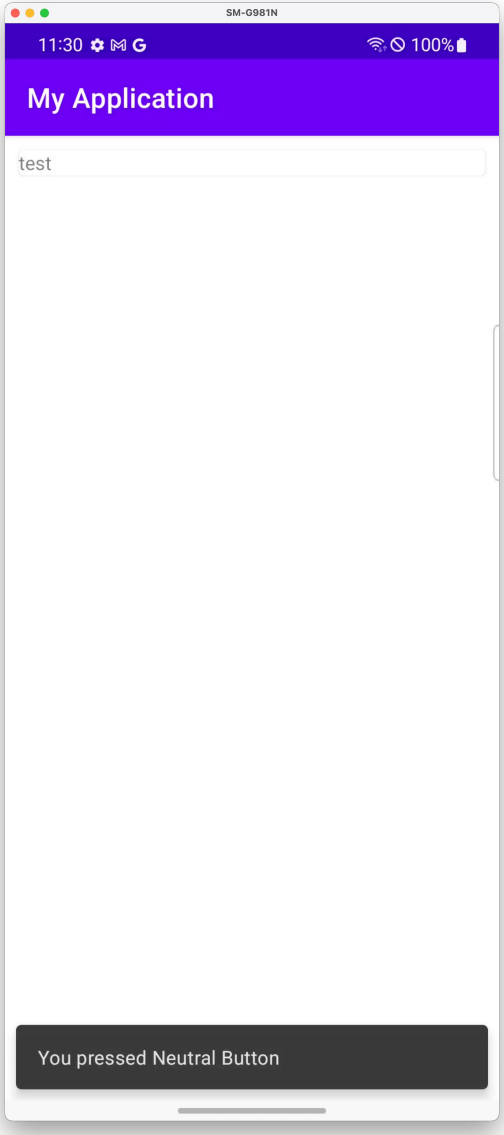
```
MaterialAlertDialogBuilder(this)
    .setTitle("Title")
    .setMessage("Message")
    .setNeutralButton("Annuler") { dialog, which ->
        // Respond to neutral button press
    }
    .setNegativeButton("Supprimer") { dialog, which ->
        // Respond to negative button press
    }
    .setPositiveButton("Valider") { dialog, which ->
        // Respond to positive button press
    }
    .show()
```

Snackbar

Composant de feedback
Non intrusif

Possibilité d’ajouter une action

Snackbar



Loaders

Donner du feedback à l’utilisateur quand une action prend du temps

Loaders

```
<com.google.android.material.progressindicator.CircularProgressIn
dicator
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Champs texte

Champs texte

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textField"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/label">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
    />

</com.google.android.material.textfield.TextInputLayout>
```

R.java

```
setContentView(R.layout.activity_main)

 recyclerView = findViewById(R.id.recycler_view)
```

R.java

Classe statique auto-générée par le SDK Android

Comprend l'ensemble des Assets du projet :
-String
-Layout
-Images (Drawable)
-Id

Layouts

ConstraintLayout

- Tous les éléments sont positionnés par rapport à des contraintes

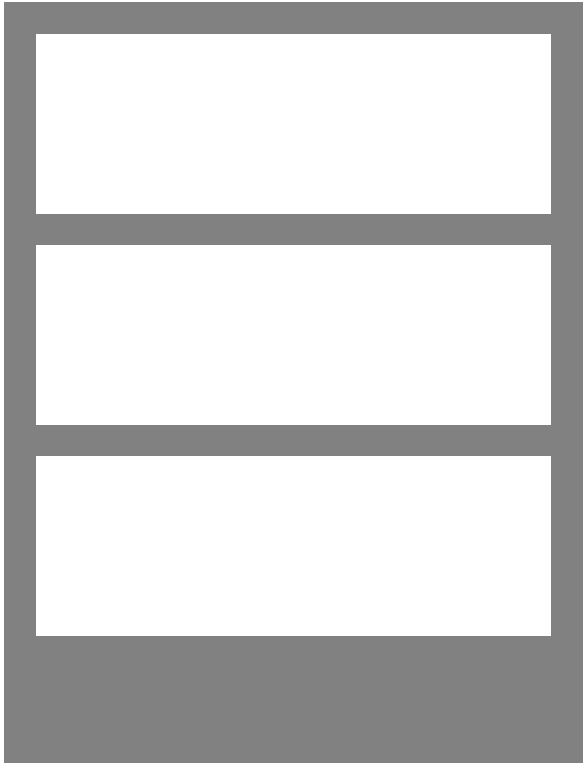
Layouts

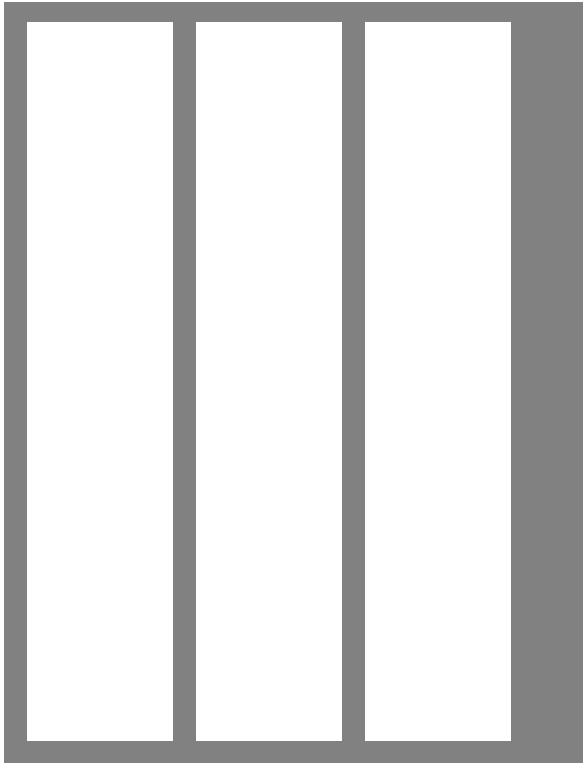
LinearLayout

- Dessine les enfants les uns après les autres selon une orientation : verticale ou horizontale

Layouts

LinearLayout





Layouts

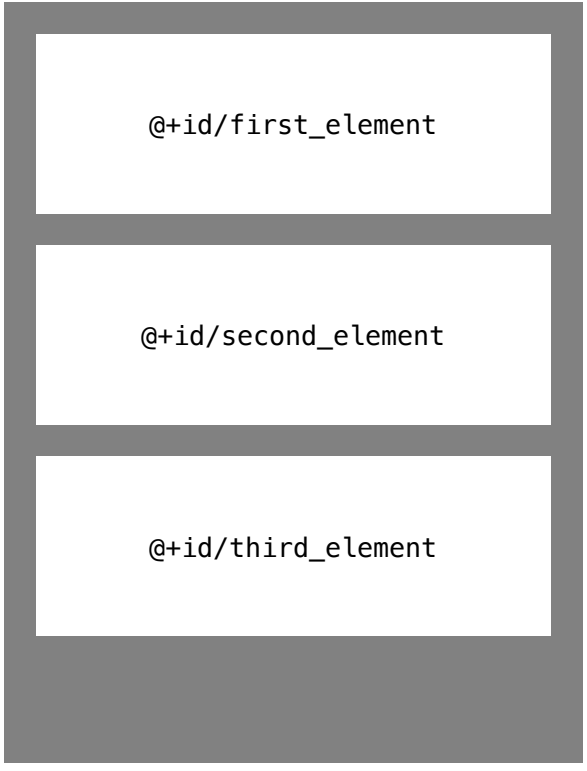
RelativeLayout

-Positionne les enfants les uns par rapport aux autres

```
layout_below
layout_above
center_in_parent
alignParentTop/Bottom…
```

Layouts

RelativeLayout




```
android:layout_alignParentTop="true »
android:layout_alignParentStart="true »
android:layout_alignParentEnd="true"
```

```
android:layout_alignParentStart="true »
android:layout_alignParentEnd="true"
android:layout_below="@+id/first_element »
```

```
android:layout_alignParentStart="true »
android:layout_alignParentEnd="true"
android:layout_below="@+id/second_element"
```

Marc PEYSALE
Mobile Team Lead
M : +33 6 31 77 86 65
@ : marc.peysale@niji.fr

niji.fr

Le 30/10/2023

Développement mobile

Enseirb-Matmeca,
Filière Info, GL 2023

BORDEAUX INP Enseirb-Matmeca

Sommaire

1. Persistance des données
2. Fragments
3. TabLayout

Persistance des données

- Besoin de conserver une partie de l'état de l'app entre 2 lancements
  - **Ex :** jetons d'authentification, réglages utilisateur etc...

Persistance des données

- Bases de données locales
  - Room
  - Realm
  - SQLite

Persistance des données

- SharedPreferences
  - Stockage de type clé/valeur
  - Fichier stocké sur le téléphone
  - Facilement accessible : pas de données sensibles

Persistance des données

SharedPreferences

- Transactions
  - Même notion que les transactions en SQL
  - Ensemble de modifications sauvegardées en même temps

Persistance des données

SharedPreferences

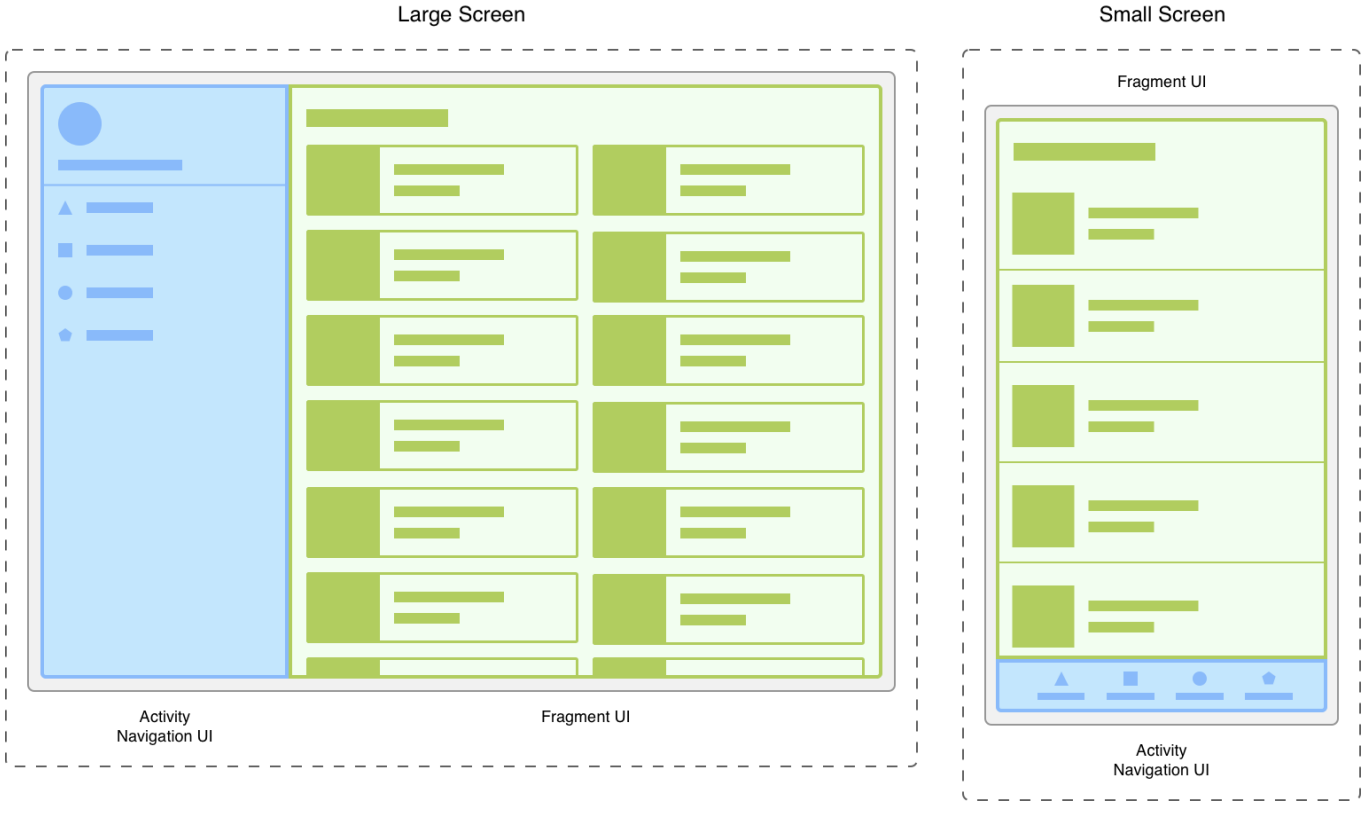
Lecture des données :

```
getSharedPreferences(SharedPreferencesKey, Context.MODE_PRIVATE)
    .getString(SavedValueKey, null).let {
        // Do something with `it`
}
```

Ecriture des données :

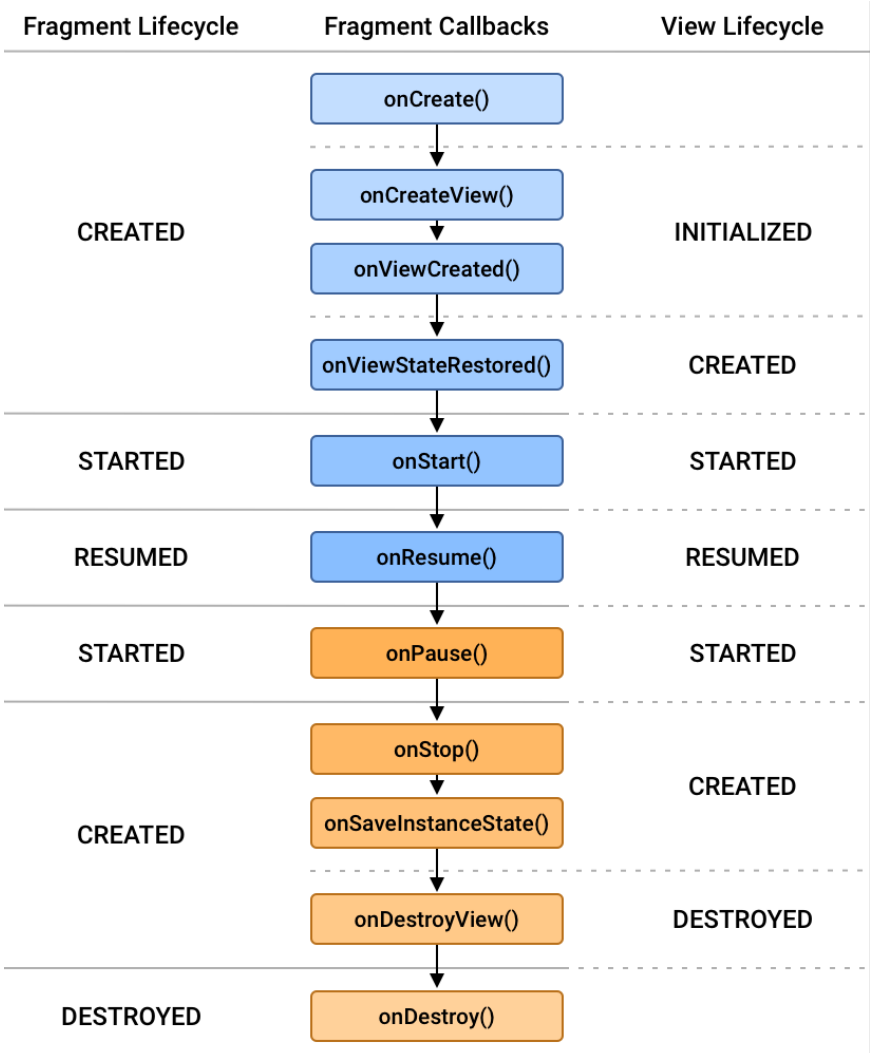
```
getSharedPreferences(SharedPreferencesKey, Context.MODE_PRIVATE).edit().apply {
    putString(SavedValueKey, myStringValue)
    // Other edits…
    apply() // Save transaction
}
```

Fragments



Fragments

Cycle de vie d'un Fragment



Fragments

FragmentManager

FragmentManager est la classe chargée d'effectuer des actions au niveau des fragments de votre application, comme les ajouter, les supprimer ou les remplacer, et les ajouter à la pile "Retour".

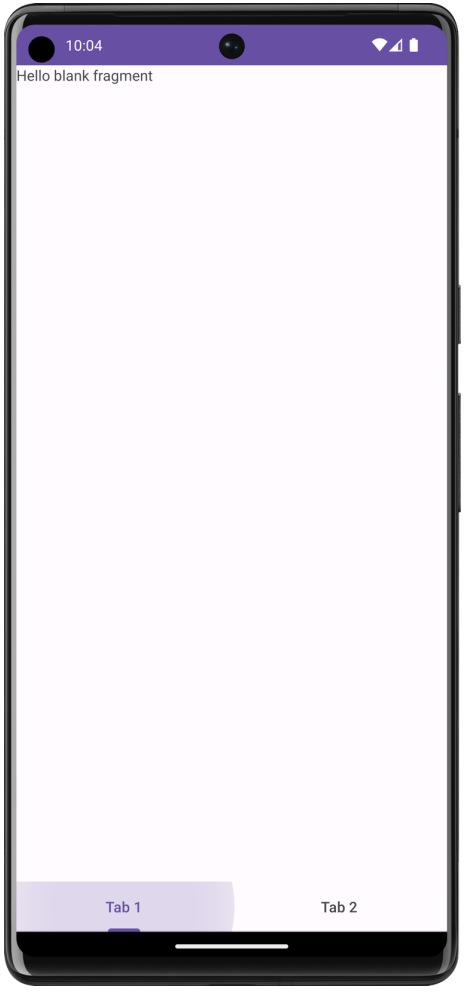
Chaque **AppCompatActivity** dispose de son **supportFragmentManager** permettant de gérer les transactions sur les Fragments.

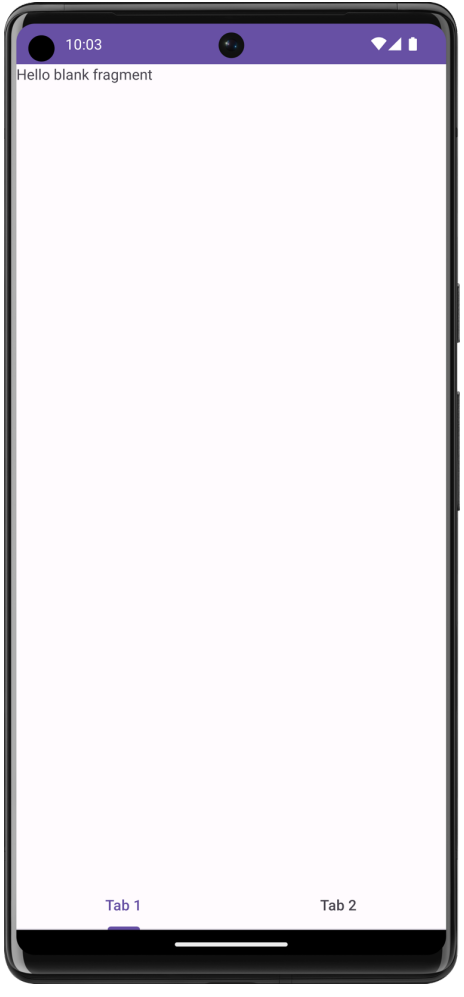
Exemple dans **onCreate** :

```
supportFragmentManager
    .beginTransaction()
    .replace(R.id.fragment_container, MyFragment.instance())
    .commit()
```

TabLayout

- Gestion de la navigation par onglets





TabLayout

Time to code!

Marc PEYSALE
Mobile Team Lead
M : +33 6 31 77 86 65
@ : marc.peysale@niji.fr

niji.fr