
HE IG VD
IICT
Institut des
Technologies de
l'information et de
la communication

HEIG VD | IICT Institut des Technologies de l'information et de la communication

# Développement d'applications *Android*

## Laboratoire 4

*Architecture MVVM, utilisation d'une base de données Room et d'un RecyclerView*

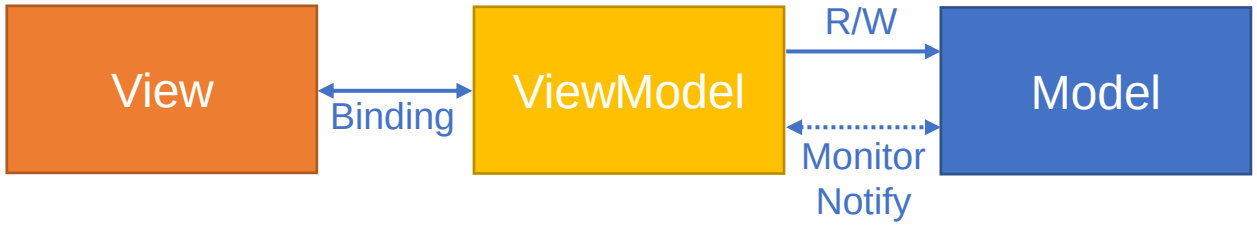
Fabien Dutoit

Elliot Ganty

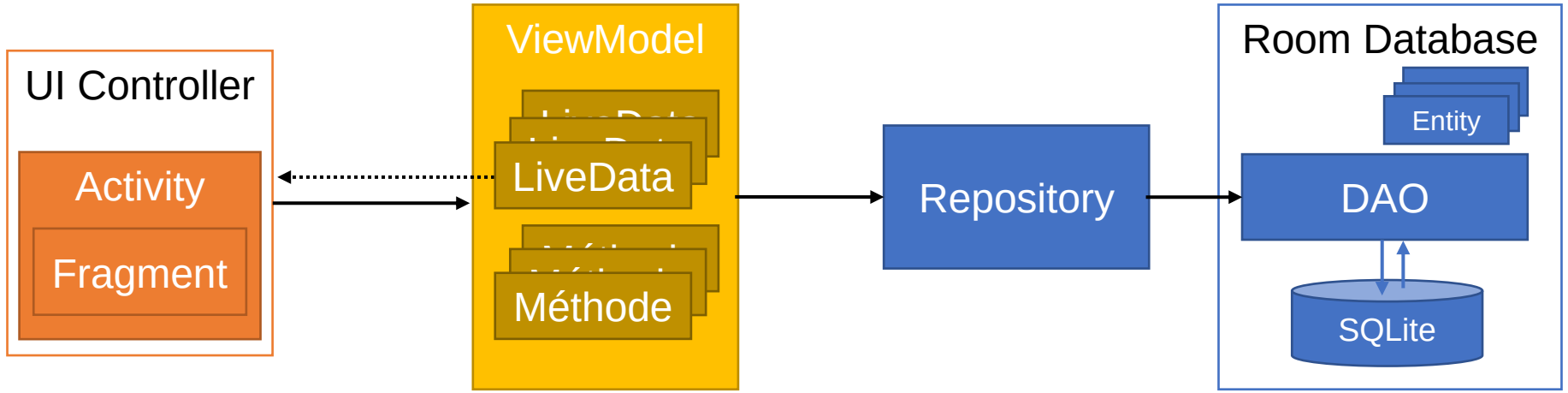
Hes·so Haute Ecole Spécialisée de Suisse occidentale

# Rappel - MVVM

Architecture MVVM :




Appliquée sur *Android :*



# Base de données

- Utilisation d'*Android Room* pour gérer la base de données
- Les modèles sont fournis (Entities), relation One-to-One (0…1)

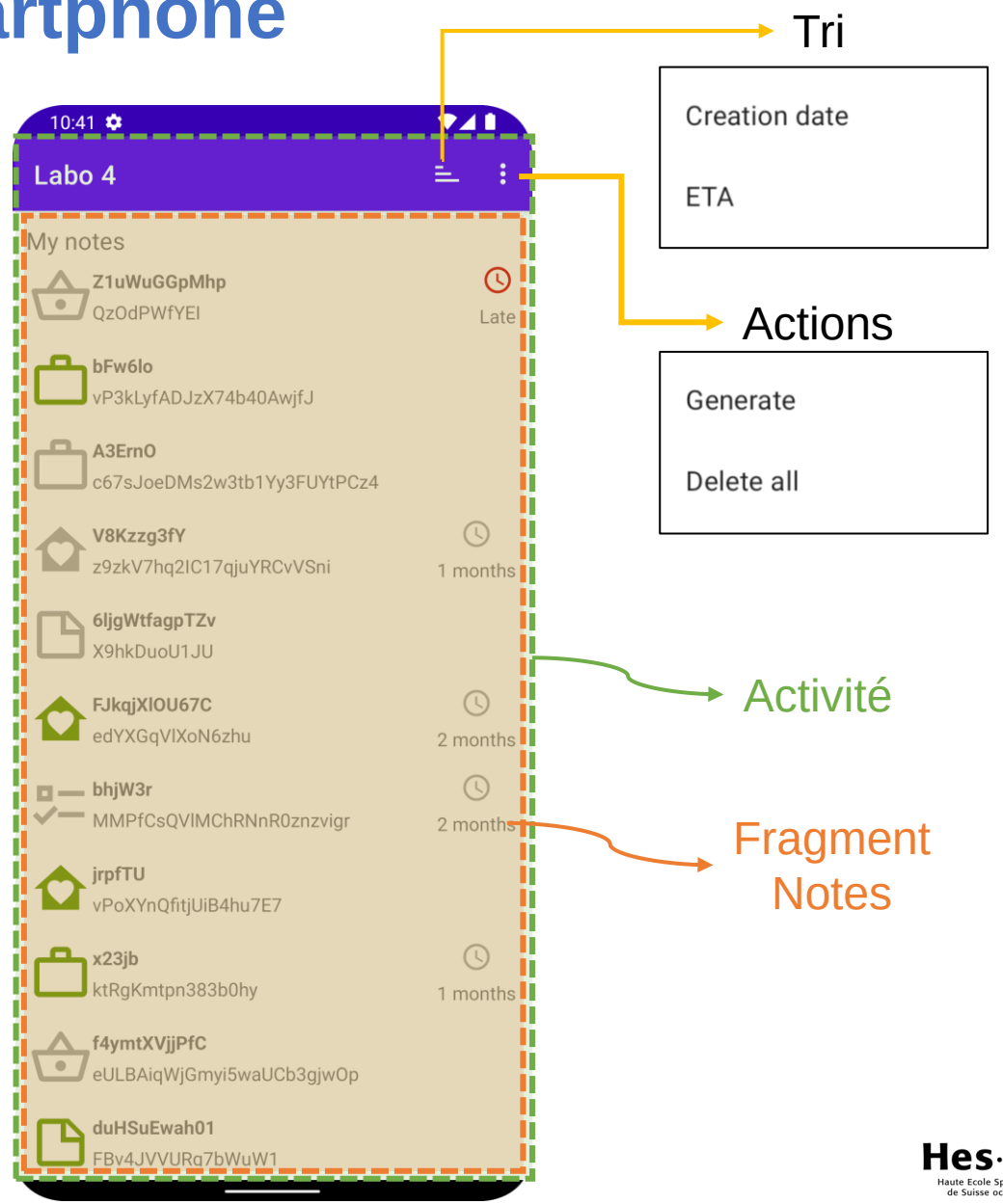
```
@Entity
data class Note (
    @PrimaryKey(autoGenerate = true) var noteId : Long?,
    var state : State,
    var title : String,
    var text : String,
    var creationDate : Calendar,
    var type : Type
)

@Entity
data class Schedule (
    @PrimaryKey(autoGenerate = true) var scheduleId : Long?,
    var ownerId : Long,
    var date : Calendar
)
```

```
data class NoteAndSchedule (
    @Embedded val note: Note,
    @Relation(
        parentColumn = "noteId",
        entityColumn = "ownerId"
    )
    val schedule: Schedule?
)

enum class State { IN_PROGRESS, DONE }

enum class Type { NONE, TODO, SHOPPING, WORK, FAMILY }
```

# Interface souhaitée – Smartphone

MainActivity, accueillant :

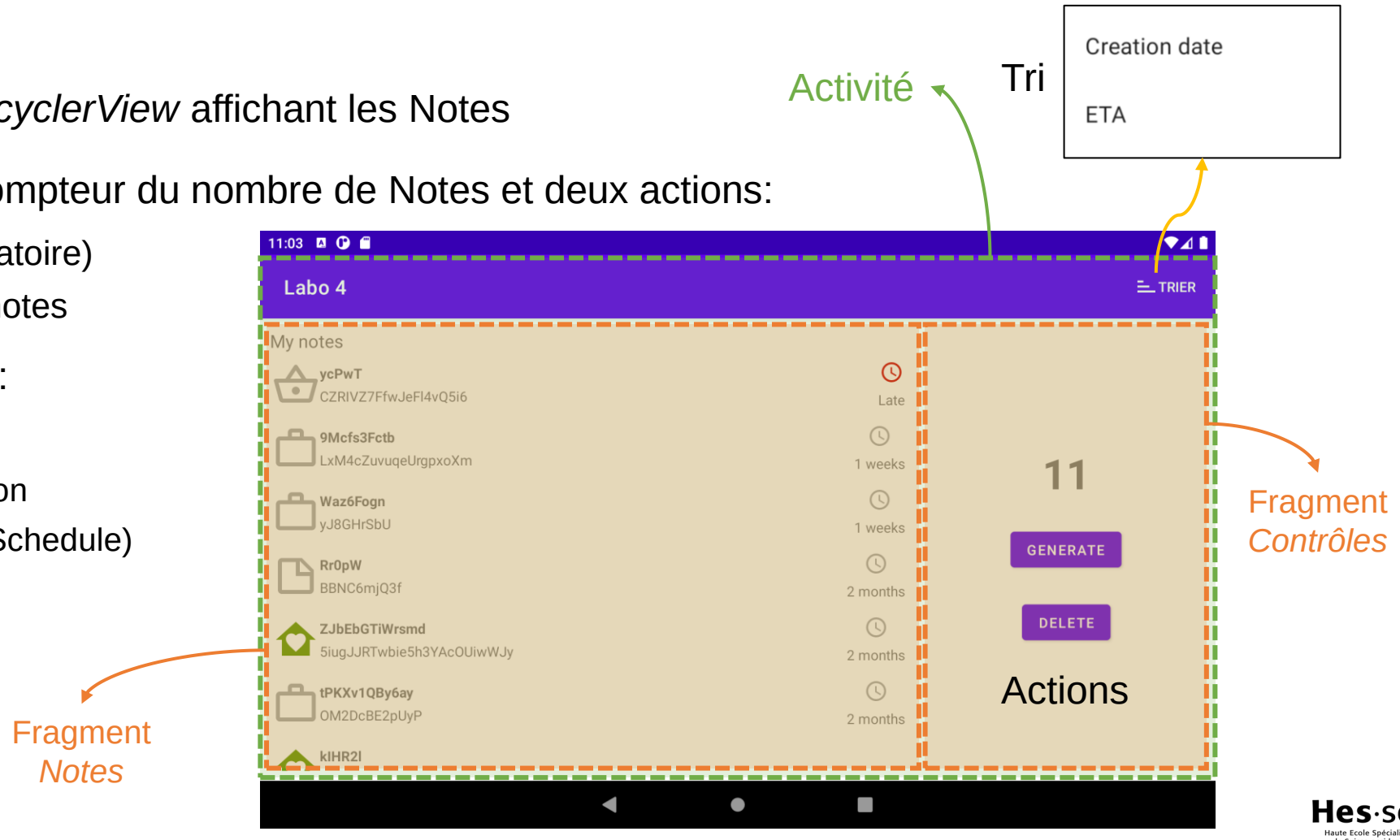
- Le *Fragment* avec la *RecyclerView* affichant les Notes
- Un Menu avec 4 options:
  - Tri des Notes
    - Par date de création
    - Par date prévue (Schedule)
  - Générer et stocker une note (aléatoire)
  - Supprimer toutes les notes
- Le *ViewModel* offrant
  - L'accès aux *LiveData* alimentant la *RecyclerView*
  - Les méthodes permettant le tri et les actions (ajout/suppression) sur les Notes



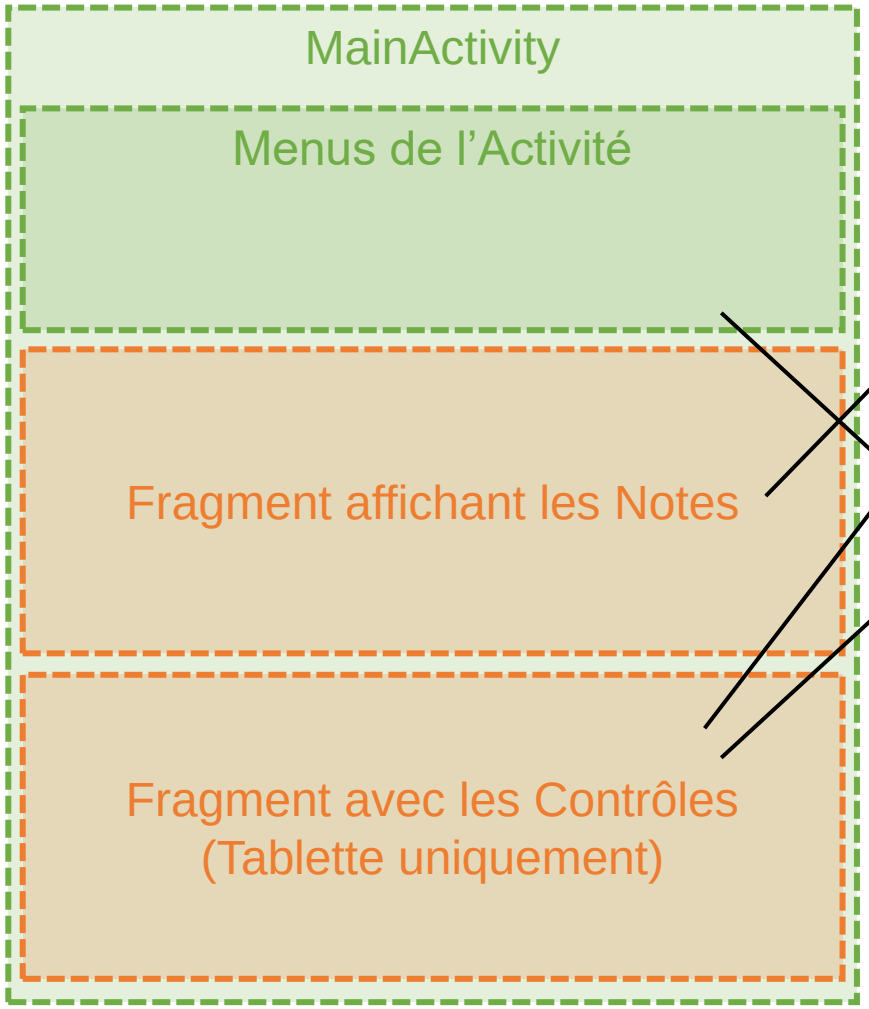
# Interface souhaitée – Tablette (configuration large)

MainActivity, accueillant :

- Le *Fragment* avec la *RecyclerView* affichant les Notes
- Un *Fragment* avec un compteur du nombre de Notes et deux actions:
  - Générer une note (aléatoire)
  - Supprimer toutes les notes
- Un Menu avec 2 options:
  - Tri des Notes
    - Par date de création
    - Par date prévue (Schedule)
- Le même *ViewModel*



# ViewModel – API




```
class NotesViewModel(private val repository: DataRepository) : ViewModel() {

    val allNotes = repository.allNotes //: LiveData<List<NoteAndSchedule>>

    val countNotes = repository.countNotes //: LiveData<Int>

    fun generateANote() { … }

    fun deleteAllNote() { … }

}
```

# Intégration du plugin KSP

Pour utiliser *KSP*, on doit ajouter un plugin aux fichiers *build.gradle.*

## build.gradle (projet)

```
plugins {
    alias(libs.plugins.android.application) apply false
    alias(libs.plugins.jetbrains.kotlin.android) apply false
    alias(libs.plugins.devtools.ksp) apply false
}
```

## Maven Central

| | Version | Vulnerabilities | Repository | Usages | Date |
|---|---|---|---|---|---|
| 2.1.x | 2.1.0-Beta1-1.0.25 | | Central | 0 | Sep 18, 2024 |
| | 2.0.21-1.0.25 | | Central | 0 | Oct 11, 2024 |
| | 2.0.21-RC-1.0.25 | | Central | 0 | Oct 01, 2024 |
| | 2.0.20-1.0.25 | | Central | 0 | Sep 05, 2024 |
| | 2.0.20-1.0.24 | | Central | 0 | Aug 23, 2024 |

## build.gradle (app)

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.jetbrains.kotlin.android)
    alias(libs.plugins.devtools.ksp)
}
```

## libs.versions.toml

```
[versions]
kotlin = "2.0.21"
ksp = "2.0.21-1.0.25"

[plugins]
jetbrains-kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref = "kotlin" }
devtools-ksp = { id = "com.google.devtools.ksp", version.ref = "ksp" }
```

**Version de *Kotlin***

**Version du plugin KSP**

src: https://mvnrepository.com/artifact/com.google.devtools.ksp/com.google.devtools.ksp.gradle.plugin

# Android Room - Dépendances

*Android Room* est présent dans un ensemble de librairies qui doivent être ajoutées en dépendance dans notre projet.

build.gradle (app)

```
dependencies {
    [...]
    // Room components
    implementation(libs.room.runtime)
    implementation(libs.room.ktx)
    ksp(libs.room.compiler)
    androidTestImplementation(libs.room.testing)
}
```

libs.versions.toml

```
[versions]
room = "2.6.1"

[libraries]
room-runtime = {group = "androidx.room", name = "room-runtime", version.ref = "room"}
room-ktx = {group = "androidx.room", name = "room-ktx", version.ref = "room"}
room-compiler = {group = "androidx.room", name = "room-compiler", version.ref = "room" }
room-testing = {group = "androidx.room", name = "room-testing", version.ref = "room"}
```

# ViewModel & LiveData - Dépendances

*Les LiveData et les ViewModels sont des composants d'Android Jetpack* qui doivent être également ajoutées en dépendance dans notre projet.

libs.versions.toml

```
[versions]
lifecycle = "2.8.6"
activity = "1.9.3"
fragment = "1.8.4"

[libraries]
androidx-lifecycle-viewmodel-ktx = {group = "androidx.lifecycle", name = "lifecycle-viewmodel-ktx", version.ref = "lifecycle"}
androidx-lifecycle-livedata-ktx = {group = "androidx.lifecycle", name = "lifecycle-livedata-ktx", version.ref = "lifecycle"}
androidx-lifecycle-common-java8 = {group = "androidx.lifecycle", name = "lifecycle-common-java8", version.ref = "lifecycle"}
androidx-activity-ktx = {group = "androidx.activity", name = "activity-ktx", version.ref = "activity"}
androidx-fragment-ktx = {group = "androidx.fragment", name = "fragment-ktx", version.ref = "fragment"}
```

build.gradle (app)

```
dependencies {
    [...]
    // Lifecycle components
    implementation(libs.androidx.lifecycle.viewmodel.ktx)
    implementation(libs.androidx.lifecycle.livedata.ktx)
    implementation(libs.androidx.lifecycle.common.java8)
    // ViewModels
    implementation(libs.androidx.activity.ktx)
    implementation(libs.androidx.fragment.ktx)
}
```

HE VD
IG

HAUTE ÉCOLE
D'INGÉNIERIE
ET DE GESTION
DU CANTON
DE VAUD