# TP - Introduction au développement natif : Kotlin

Karim Benchekroun

2024

## 1 TP : Développement d'une application de capsules temporelles avec Kotlin

### 1.1 Objectifs pédagogiques

— Maîtriser les fondamentaux du développement Android avec Kotlin
— Implémenter une architecture MVC
— Manipuler les composants natifs (MediaStore, notifications)
— Gérer la persistance des données et le stockage de médias
— Appliquer les bonnes pratiques de versioning avec Git
— Respecter une charte graphique existante (Figma)

**Note** Il n'est pas attendu de reproduire exactement les maquettes Figma, mais de respecter les Material Design guidelines. Toutes les fonctionnalités imaginées dans les maquettes ne sont pas nécessairement attendues. L'accent est mis sur la qualité du code et la bonne implémentation des fonctionnalités principales : la création de capsules par un utilisateur et leur visualisation par ce même utilisateur.

### 1.2 Prérequis

— Connaissances en Programmation Orientée Objet
— Bases du langage Kotlin (types, classes, coroutines) (cf. cours)
— Compréhension de l'architecture Android (cf. cours)
— Maquettes Figma réalisées lors du TP précédent
— Android Studio installé

### 1.3 Étape 1 : Configuration du projet

#### 1.3.1 Création du projet Android Studio

1. Lancez Android Studio et sélectionnez "New Project"
2. Choisissez "Empty Activity" comme template

3. Configurez le projet :
   — Name : "Remember"
   — Package name : "com.votreprenom.remember"
   — Language : Kotlin
   — Minimum SDK : API 24 (Android 7.0)

**Note** Si vous n'avez jamais configuré d'émulateur Android, suivez le guide officiel : Configuration d'un émulateur Android.

### 1.3.2 Organisation des fichiers

Créez la structure de dossiers suivante dans votre projet :

```
app/src/main/java/com/votreprenom/remember/
    models/
    views/
        fragments/
        adapters/
    controllers/
    services/
    utils/
```

Pour créer cette structure :

1. Dans Android Studio, faites un clic droit sur le package principal
2. Sélectionnez "New > Package" pour chaque dossier
3. Renommez selon la structure ci-dessus

**Astuce** Activez "Show File Extensions" dans Android Studio : Settings > Editor > General > Appearance > Show file extensions.

### 1.3.3 Configuration Git

1. Initialisez Git si ce n'est pas déjà fait :

   ```
   git init
   ```

2. Créez un fichier .gitignore adapté à Android. Vous pouvez utiliser le modèle suivant : Android.gitignore sur GitHub
3. Effectuez votre premier commit :

   ```
   git add .
   git commit -m "Initial commit: Project setup"
   ```

**Ressources complémentaires** Pour une configuration plus détaillée de votre environnement de développement Android, suivez le guide officiel : Guide Android Studio

**Points de vérification** Avant de passer à l'étape suivante, assurez-vous que :
- Votre projet compile sans erreur
- La structure de dossiers est en place
- Git est initialisé avec un .gitignore approprié
- Vous pouvez lancer l'application sur l'émulateur Android

**Problèmes courants et solutions**
- **Erreur Gradle** : File > Invalidate Caches / Restart
- **Erreur SDK** : Tools > SDK Manager pour installer les composants manquants
- **Émulateur lent** : Activer la virtualisation matérielle dans le BIOS

## 1.4 Étape 2 : Implémentation des modèles de données

### 1.4.1 Création du modèle Capsule

Créez un fichier `Capsule.kt` dans le dossier models qui devra contenir :
- Un identifiant unique
- Un titre et une description
- Les dates de création et de déverrouillage
- Un état de verrouillage
- Une collection de médias (images/vidéos)

**Ressources**
- Documentation Data Classes Kotlin
- Documentation UUID Android

### 1.4.2 Création du modèle User

Créez un fichier `User.kt` contenant les informations minimales :
- Identifiant unique
- Nom d'utilisateur
- Email
- Collection de capsules

### 1.4.3 Gestion des dates et du verrouillage

Implémentez un `CapsuleManager` qui gérera :
- La vérification du statut de verrouillage
- La comparaison des dates
- La mise à jour du statut des capsules

**Ressources**
- Documentation LocalDateTime
- Documentation Java Time Android

### 1.4.4 Persistance des données

Pour la sérialisation et la persistance des modèles, vous pouvez utiliser :
— Room Database pour le stockage local
— Gson/Moshi pour la sérialisation JSON

**Ressources**
— Guide Room Database
— Documentation Room Entity

## 1.5 Étape 3 : Logique métier et Controllers

### 1.5.1 Création des Controllers

Implémentez les controllers suivants :
— `CapsuleController` : Gestion des opérations CRUD sur les capsules
— `UserController` : Gestion des opérations liées à l'utilisateur
— `MediaController` : Gestion des opérations sur les médias

**Responsabilités du CapsuleController**
— Création et modification des capsules
— Vérification des dates de déverrouillage
— Coordination avec le service de notification
— Gestion des erreurs et validation des données

**Responsabilités du UserController**
— Gestion du profil utilisateur
— Validation des données utilisateur
— Gestion des préférences utilisateur

**Responsabilités du MediaController**
— Gestion des permissions d'accès aux médias
— Traitement des images et vidéos via MediaStore
— Gestion du stockage des médias

### 1.5.2 Gestion de l'état de l'application

Implémentez un `AppController` central qui :
— Coordonne les différents controllers
— Gère le cycle de vie de l'application
— Maintient l'état global de l'application

### 1.5.3 Services de données

Créez les services nécessaires pour :
— La persistance des données avec Room
— La gestion du stockage local

— Les opérations asynchrones avec les Coroutines

**Ressources**
— Documentation Activity
— Guide Coroutines
— Documentation Context

## 1.6 Étape 4 : Interface utilisateur avec Jetpack Compose

### 1.6.1 Mise en place de la navigation

Créez une navigation par onglets comprenant :
— Liste des capsules
— Création de capsule
— Profil utilisateur

**Ressources**
— Documentation Navigation Compose
— Documentation BottomNavigation

### 1.6.2 Création des vues principales

Implémentez les composables suivants :
— `CapsuleListScreen` : Affichage et filtrage des capsules
— `CreateCapsuleScreen` : Formulaire de création
— `CapsuleDetailScreen` : Visualisation détaillée
— `ProfileScreen` : Gestion du profil

### 1.6.3 Implémentation des composants réutilisables

Créez les composants suivants :
— Sélecteur de médias personnalisé
— Sélecteur de date de déverrouillage
— Carte de prévisualisation de capsule

**Ressources**
— Documentation Compose Layout
— Documentation Compose Components
— Material Design 3 pour Compose

## 1.7 Étape 5 : Fonctionnalités avancées

### 1.7.1 Sélection et stockage des médias

Utilisez les frameworks natifs pour :
— La sélection de photos et vidéos via MediaStore
— Le stockage local des médias dans le stockage interne de l'application

**Ressources**
- Documentation MediaStore
- Guide Stockage Android

#### 1.7.2 Gestion des notifications

Implémentez un gestionnaire de notifications pour :
- Programmer les notifications de déverrouillage
- Gérer les permissions
- Personnaliser le contenu des notifications

**Ressources**
- Documentation Notifications
- Guide Permissions Android

### 1.8 Livrables attendus

- Repository GitHub privé
- Ajoutez `@karim1349` comme collaborateur
- README.md avec : nom/prénom du binôme, fonctionnalités, difficultés, captures d'écrans de l'application

### 1.9 Critères d'évaluation

- **Fonctionnel**
  - Implémentation des fonctionnalités principales
  - Gestion des erreurs
  - Fluidité de l'expérience utilisateur
- **Interface**
  - Cohérence stylistique avec les maquettes Figma
  - Note : une reproduction exacte n'est pas exigée, l'accent est mis sur le respect des Material Design guidelines
- **Technique**
  - Qualité et lisibilité du code
  - Architecture MVC respectée
  - Historique Git propre avec commits conventionnels, et charge de travail équitable entre les membres du binôme

### 1.10 Ressources complémentaires

- Documentation Android
- Tutoriels officiels Jetpack Compose
- Documentation Kotlin
- Guide Git
- Convention de nommage des commits
- Material Design Guidelines

— Android Developers Courses
— Guide Android Studio