Χρήση σύγχρονης αρχιτεκτονικής

Android

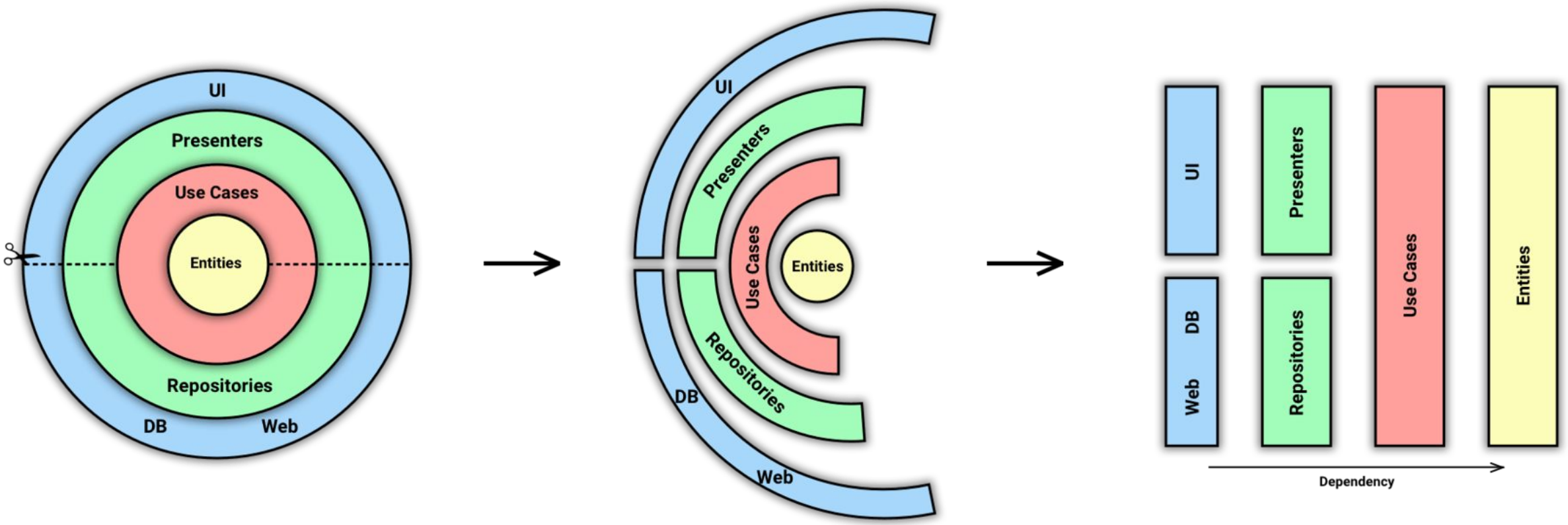
Αρχιτεκτονική & Designs Patterns

Η αρχιτεκτονική και τα design patterns μας διευκολύνουν:

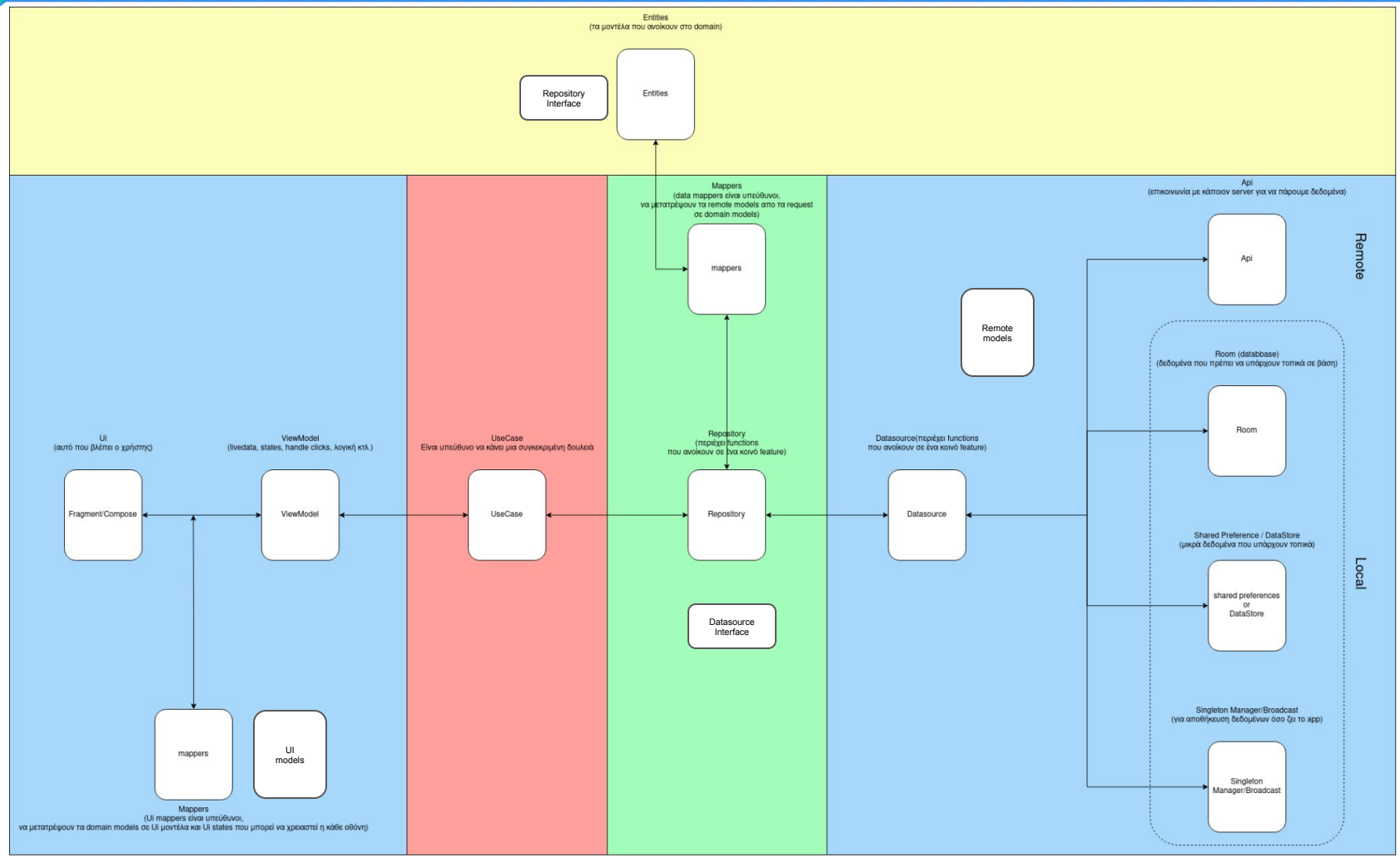
- στον αρχικό σχεδιασμό
- στη συντήρηση
- στην επέκταση
- στη σωστή λειτουργία της ομάδας
- στο automated testing

Για την αρχιτεκτονική των εφαρμογών μας χρησιμοποιούμε **MVVM** (προτείνεται από την Google) και **Clean Architecture**.

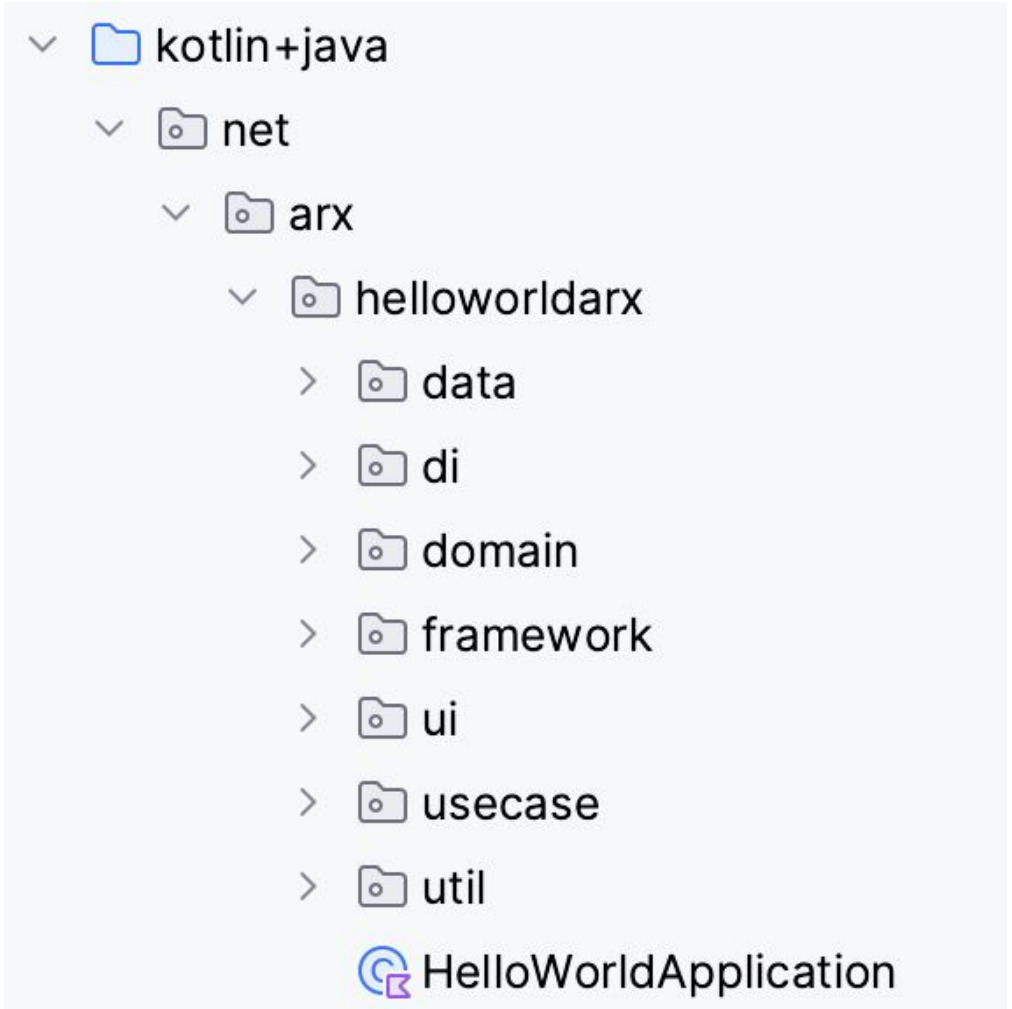
Clean Architecture

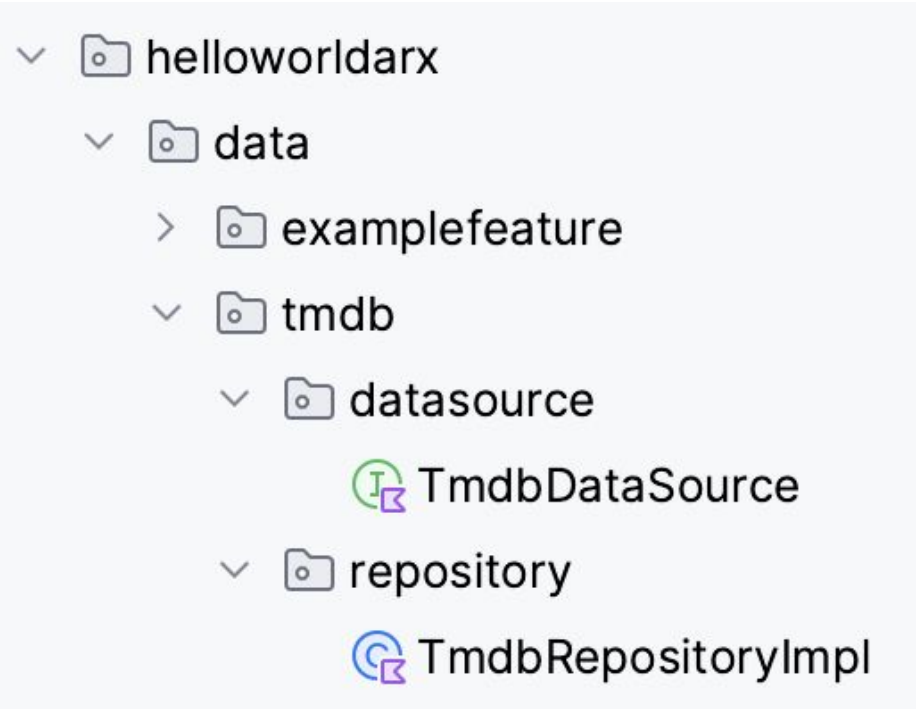


Η αρχιτεκτονική στο Android



Εφαρμογή με απλούς κανόνες





Βασικές μεθοδολογίες του Jetpack Compose

Android

Βασική δομή στο Compose

- <Feature>Screen, <Feature>Content, <Feature>ContentPreview
- State Hoisting
- Reusable composables - όπου αυτό είναι εφικτό

```kotlin
@Composable
internal fun SplashScreen(viewModel: SplashViewModel) {
    val splashStateFlowUi = viewModel.splashStateFlowUi.collectAsStateWithLifecycle()
    SplashContent(splashUiState = splashStateFlowUi)
}


@Composable
fun SplashContent(splashUiState: State<SplashUiState>) {
    // TODO: handle UI State
}
```

```kotlin
@Preview(name = "Loading State")
@Composable
private fun SplashContentLoadingStatePreview() {
    val splashUiState = remember {
        mutableStateOf(SplashUiState.LoadingUiState)
    }

    HelloWorldArxTheme {
        SplashContent(splashUiState = splashUiState)
    }
}
```

Δυναμικά μεγέθη

Καθώς υπάρχει πληθώρα σε μεγέθη οθονών, πρέπει να σχεδιάζουμε τα UI μας με τη χρήση δυναμικών sizes (στους modifiers) ως προς το width της οθόνης (πχ fillMaxWidth, fillMaxHeight, aspectRatio) και να αποφεύγουμε τα στατικά dp.

Αυτό θα έχει ως αποτέλεσμα, το UI να φαίνεται ομοιόμορφα στους χρήστες ανεξάρτητα με τις οθόνες που έχουν.

*Ιδανικά πρέπει να σχεδιάζουμε UI για tablets ή Foldables

Βασικά UI States

Για την καλύτερη διαχείριση του UI αλλά και για να βελτιώσουμε το User Experience δημιουργούμε τα παρακάτω βασικά UI States για την κάθε οθόνη, με τη χρήση των sealed classes

- **Loading**, ο χρήστης θα πρέπει να βλέπει έναν Loader όταν γίνεται το βασικό request της οθόνης.
- **Default Content (Data)**, ο χρήστης θα βλέπει τα προβλεπόμενα δεδομένα στην οθόνη.
- **Error**, ο χρήστης θα βλέπει στην οθόνη ένα γενικό μήνυμα λάθους.
- **Empty**, σε περίπτωση που η οθόνη αποτελείται από μια λίστα (πχ Movies Category) η οποία είναι άδεια, ο χρήστης θα πρέπει να δει κάτω από τον τίτλο ένα μήνυμα “δεν βρέθηκε περιεχόμενο”.

```
sealed interface SplashUiState {
    class DefaultUiState(val splashUiItem: String) : SplashUiState
    data object ErrorUiState : SplashUiState
    data object LoadingUiState : SplashUiState
}

data class SplashUiItem(
    val text: String
)
```

LiveData, State, StateFlow

Για την επικοινωνία (lifecycle aware) των UI States (Default, Loading, Error κτλ) απο το ViewModel προς το UI μπορούμε να κάνουμε χρήση των παρακάτω:

- **LiveData**, χρήση πιο πολύ με τα fragments & xml
- **State**, χρήση με Compose ακούει στις αλλαγές και κάνει recomposition μόνο στα απαραίτητα Composables
- **StateFlow**, για πιο σύνθετα δεδομένα τα οποία πρέπει να εμφανίζονται με flow

Διαχείριση Ui State στο Compose

```
@Composable
fun SplashContent(splashUiState: State<SplashUiState>) {
    when (val currentState = splashUiState.value) {
        is SplashUiState.DefaultUiState -> {
            // TODO: default content
        }

        SplashUiState.ErrorUiState -> {
            // TODO: show error state - in cases that we might have an error
        }

        SplashUiState.LoadingUiState -> {
            // TODO: loading
        }
    }
}
```

Βασικά βήματα υλοποίησης

Όταν έχουμε ένα feature για να υλοποιήσουμε μπορούμε να ξεκινήσουμε με τα παρακάτω βήματα:

1. Ανάλυση και καταγραφή των requirements & designs,
2. Σχεδίαση των βασικών λειτουργιών σε UML (χαρτί, draw.io κτλ)
3. Προγραμματισμός, ξεκινώντας με τα δεδομένα που έχουμε σίγουρα (πχ αν δεν έχουμε ακόμα το backend έτοιμο, μπορούμε να ξεκινήσουμε με το UI με τα αντίστοιχα UI models)
   a. Πάντα πριν κάνουμε commit βεβαιωνόμαστε ότι το project κάνει compile και δεν crasharει με το άνοιγμα
   b. Κάνουμε μικρά commits
4. Δοκιμάζουμε την εφαρμογή σε διάφορες συσκευές (Android versions, screen size) και σε Release builds.

THANK YOU
QA