Modularization

The goal is to separate functionality into independent, interchangeable modules.

- Faster build times.
- Fine-grained dependency control.
- Improve reusability across other apps.
- Improves the ownership & the quality of the codebase.
- Stricter boundaries when compared to packages.
- Encourages Open Source of the newly created libraries.
- Makes Instant Apps & Dynamic Features possible (improving discoverability).

Feature Modules: Keeping it Clean.

- Modules are user-facing functionality
- We don't have package names like activities/fragments/adapters
    - Each module should contain its own presentation, domain, data & data sources layers (if they require it).

Base Modules: Avoid a single Base Module.

- Stay away from a single Base/Core Module. Don't be afraid of splitting the base functionality of your app so feature modules can pick which base functionality they need.
- All of these base modules will never be user-facing and will only be used by 1 or multiple Modules.
- Presentation contains mostly View/Keyboard extension functions & some goodies for ViewModel like the SingleLiveEvent from google samples.

Gradle:

- apply from: `$rootDir/common-android-library.gradle` we can ended up with a concise and small mymodule.gradle

Dynamic Feature Module

- No need multiple APKs
- Reduce size of APK that user needs to download
- Provide functionality only needed
- Instant delivery
- Android App Bundle
    - Base Module *1: `com.android.application`
    - Library Module *n: `com.android.library`
    - Dynamic Feature Module *n: `com.android.dynamic-feature`
- Play Core Library

    - Required by Base Module.
    - Used to request download Dynamic Feature Module during App runtime

- Split APKs service (Google Play)

    - .aab --> Base APK + Dynamic feature APKs + Configuration APKs

- Dynamic Delivery service (Google Play)

    - Deliver APKs based on user device information