NAMA : GHINAN SHABRINA ZAHIDAH

NIM : 2300018193

Pola Arsitektur Android

Pola arsitektur Android adalah struktur yang mengatur cara komponen aplikasi berkomunikasi dan terhubung. Ini berfungsi sebagai panduan untuk mengorganisir kode, memisahkan tanggung jawab, dan mengatur aliran data, sehingga aplikasi lebih terstruktur, mudah dikembangkan, dan dapat diuji.

1. MVC (Model-View-Controller)

- **Gambaran**: Pola klasik yang membagi aplikasi jadi 3 bagian. Cocok buat proyek kecil.
- **Cara kerja**: User berinteraksi dengan tampilan → Controller terima input → Controller hubungi Model → Model proses data → Controller update tampilan.
- **Plus**: Gampang dipelajari, cocok buat pemula.
- **Minus**: Activity/Fragment sering kebanyakan tugas, sulit di-test, dan komponennya ketergantungan tinggi.

2. MVP (Model-View-Presenter)

- **Gambaran**: Penyempurnaan MVC, Presenter jadi otak yang mengatur logika, View cuma tampil aja.
- **Cara kerja**: View terima input → Presenter proses → Model eksekusi data → Presenter update View lewat interface.
- **Plus**: Presenter bisa di-test (tanpa Android), tanggung jawab lebih jelas.
- **Minus**: Bikin banyak interface manual, harus atur lifecycle sendiri.

3. MVVM (Model-View-ViewModel)

- **Gambaran**: Pola modern pakai sistem reactive, UI otomatis update kalo data berubah.
- **Cara kerja**: User aksi di View → ViewModel proses → Model kerja → ViewModel update LiveData/Flow → View auto berubah.
- **Plus**: UI lebih responsif, ViewModel tahan rotation, didukung resmi Google.
- **Minus**: Bisa bocor memori kalo salah, butuh belajar LiveData/Flow dulu.

4. MVI (Model-View-Intent)

- **Gambaran**: Pola ketat dengan aliran data satu arah. UI sepenuhnya ditentukan oleh state yang tetap.
- **Cara kerja**: User aksi jadi "Intent" → ViewModel proses jadi State baru → View render ulang sesuai state.
- **Plus**: State mudah diprediksi, debugging gampang, aman dari race condition.
- **Minus**: Banyak kode repetitif, terlalu ribet buat proyek kecil.

5. Clean Architecture

- **Gambaran**: Arsitektur berlapis dengan dependency ketat, fokus pada independensi dari framework.
- **Cara kerja**: Lapisan Presentation terima input → Domain Layer (Use Cases) olah logika → Data Layer ambil data → balik ke UI.
- **Plus**: Tiap lapisan bisa di-test, fleksibel ganti framework, maintainable buat tim besar.
- **Minus**: Berlebihan buat aplikasi kecil, butuh banyak setup, learning curve curam.

Kesimpulan singkat:

- **MVC**: Oke buat belajar, tapi cepat outgrow.
- **MVP**: Lebih terpisah, tapi bikin kode lebih banyak.
- **MVVM**: Pilihan modern dengan dukungan terbaik saat ini.
- **MVI**: Cocok buat aplikasi kompleks yang butuh state management ketat.
- **Clean Architecture**: Untuk aplikasi besar dan tim, tapi jangan dipaksain buat yang sederhana.