

Clean
vs.
MVVM
vs.
MVI

Architectures in Android Development

SWIPE



Esmaeeil Moradi



Choosing the Right Architecture for Your App

Selecting the right architecture is crucial for building scalable, maintainable, and robust Android apps. Below, we compare Clean Architecture, MVVM, and MVI, focusing on their strengths, use cases, and layers.

KEEP SWIPING

Esmaeetil Moradi



1. Clean Architecture: Separation of Concerns at Scale

- **Layers:**
- Domain Layer: Contains business logic, entities, and use cases.
- Data Layer: Handles data sources (local DB, network APIs).
- Presentation Layer: Manages UI components (Activities, Fragments, Composables).
- **Key Features:**
- ☒ Framework Independence: Business logic remains untouched by external libraries.
- ☒ High Testability: Each layer can be tested in isolation.
- ☒ Scalability: Ideal for large teams and complex projects.
- **When to Use?**
- Enterprise apps with intricate business rules (e.g., banking, e-commerce).
- Long-term projects requiring flexibility and modularity.

KEEP SWIPING



2. MVVM (Model-View-ViewModel): Streamlining UI Logic

- **Layers:**
 - Model: Manages data and business logic.
 - View: Displays UI and observes ViewModel.
 - ViewModel: Mediates between View and Model, exposing data via LiveData/StateFlow.
- **Key Features:**
 - ☒ Jetpack Integration: Works seamlessly with LiveData, Data Binding, and ViewModel.
 - ☒ Reduced Boilerplate: Simplifies UI updates with reactive programming.
 - ☒ Lifecycle Awareness: Survives configuration changes.
- When to Use?
 - Apps with dynamic UIs (e.g., social media, news feeds).
 - Teams familiar with Android Jetpack.

KEEP SWIPING



3. MVI (Model-View-Intent): Predictable State Management

- **Layers:**
- Model: Represents a single immutable UI state.
- View: Renders state and sends user Intents (e.g., button clicks).
- ViewModel: Processes intents, updates state, and emits new state.
- **Key Features:**
- ☒ Unidirectional Flow: State changes follow a strict Intent → Model → View cycle.
- ☒ Immutable State: Prevents side effects by creating new state instances.
- ☒ Debugging Ease: Single source of truth simplifies tracking state transitions.
- **When to Use?**
- Apps with complex state logic (e.g., forms, real-time dashboards).
- Projects prioritizing predictability and testability.

KEEP SWIPING



Layer Comparison

Layer	Clean Architecture	MVVM	MVI
Model	Domain entities, data access	Data and business logic	Immutable State
Domain	Core business logic	Not explicitly defined	Not explicitly defined
Data	Handles data sources	Part of Model	Part of Model
Presentation	UI components	View + ViewModel	View + ViewModel + Intents



Key Takeaways

- Clean Architecture separates the entire app into layers (Domain, Data, Presentation).
- MVVM/MVI focus only on the Presentation Layer, splitting it into View, ViewModel, and (for MVI) Intents.
- You can combine Clean with MVVM/MVI: Use Clean for overall structure and MVVM/MVI for UI logic.

KEEP SWIPING



Architecture Comparison

Table

08/10

Feature	Clean	MVVM	MVI
Focus	Full app structure	UI logic separation	State management
Complexity	High	Medium	Medium/High
Best For	Large-scale projects	Dynamic UIs	Complex state flows
Reactive Support	Optional	Built-in (Jetpack)	Built-in
Learning Curve	Steep	Moderate	Steep



When to Use Which?

- 🏢 Clean + MVVM/MVI: For enterprise apps needing scalability (e.g., finance apps).
- 🎨 MVVM: For UI-centric apps using Jetpack (e.g., social media).
- 🔄 MVI: For apps with intricate state transitions (e.g., form-heavy apps).

KEEP SWIPING



**"Architectures are tools, not goals.
Choose based on your project's
needs, team expertise, and
scalability requirements."**

 **Let's Discuss!**

Which architecture do you prefer, and why? Share your thoughts below! 

COMMENT BELOW

Esmaeeil Moradi

