MVI Architecture

Topics

- What is MVI architecture?
- Components of MVI
- Key Principles of MVI
- How does the MVI work?
- Advantages and Disadvantages of MVI

What is MVI Architecture

- MVI(Model-View-Intent) is an architecture pattern for designing and building applications, primarily in Android.
- It helps maintain a unidirectional data flow, ensuring that state management is predictable.
- **Key Goals**: Simplifying code, improving maintainability, and ensuring a reactive UI.

Components of MVI

- **Model**:
  - Represents the state of the application.
  - Holds all the data and logic of the application.

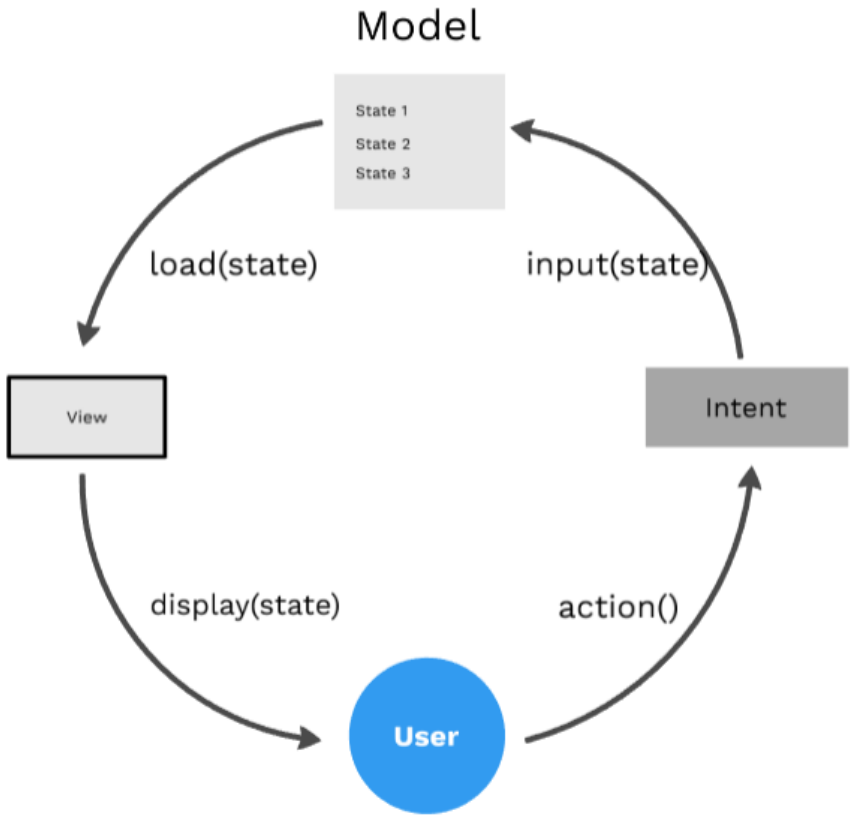
- **View**:
  - Represents the UI that displays data to the user.
  - It reacts to changes in the state from the Model.

- **Intent**:
  - Represents the user’s actions or intentions.
  - The View sends Intents to the Model, which updates the state

Key Principles of MVI

- Unidirectional Data Flow:
  - View → Intent → Model → View.
  - This ensures that all changes flow in one direction, making the state predictable.
- Immutable States:
  - The state of the model is immutable, meaning it cannot be modified directly.
- Reactive UI:
  - The View reacts to state changes and updates the UI accordingly.

How Does MVI Work



MVI cyclic flow representation

Advantages of MVI

- Maintaining state is no more a challenge with this architecture, As it focuses mainly on states.
- As it is unidirectional, Data flow can be tracked and predicted easily.
- It ensures thread safety as the state objects are immutable.
- Easy to debug, As we know the state of the object when the error occurred.
- It's more decoupled as each component fulfills its own responsibility.
- Testing the app also will be easier as we can map the business logic for each state.

Disadvantages of MVI

- It leads to lots of boilerplate code as we have to maintain a state for each user action.
- As we know it has to create lots of objects for all the states. This makes it too costly for app memory management.
- Handling alert states might be challenging while we handle configuration changes. For example, if there is no internet we will show the snackbar, On configuration change, it shows the snackbar again as its the state of the intent. In terms of usability, this has to be handled.

Thanks