Design Patterns For Mobile Apps

Gamboa López Jafet Jeshua
4A
Universidad Tecnológica de Tijuana
Tijuana, México
0322103712@ut-tijuana.edu.mx

***Abstract*—This document is an explication of What do design patterns mobile apps consist of?**

***Index Terms*—mobile, app, design, pattern**

I. Introduction

A design pattern is a reusable approach to solving a common design problem. This solution has been utilized and refined by designers and developers over time to address a specific requirement in an app's user interface. Mobile design patterns can include various elements, such as:

Navigation
Input validation
Data display
Layout

Using well-established design patterns can make an app's interface more intuitive and familiar to users, as they may have interacted with similar design patterns in other apps. Additionally, design patterns save designers and developers time and effort! Through these, they won't have to reinvent the wheel whenever they encounter challenges in mobile design.

II. Model View Controller (MVC) Architecture

MVC is a design model that separates an application into three interacting parts: Model, View, and Controller. This separation allows for better code design and modularization.

Model: Represents application data and business logic.

View: Displays data to the user.

Controller: Processes user input and controls data flow between Model and View.

III. Model View Presenter (MVP) Architecture

MVP is a new architecture that separates an application into three parts: Model, View, and Presenter. This is similar to MVC but puts more responsibility on the Teacher to manage the interaction between Model and View.

Model: Manages data and business logic.

View: Represents the user interface.

Designer: Acts as an intermediary processing user input and updating the View and Model.

IV. Model View View Model (MVVM) Architecture

MVVM is a design model widely used in mobile development, especially in frameworks like Android's Jetpack. Its purpose is to separate the application into three parts: Model, View, and ViewModel.

Model: Represents data and business logic.

View: Represents the user interface.

ViewModel: Acts as an interface between the Model and the View, which contains the reference logic.

V. VIPER Architecture

VIPER stands for View, Interactor, Presenter, Entity, and Router. VIPER is primarily based at the clean architecture ideas, which purpose to separate the concerns of different layers of the utility. Each layer has a single duty and communicates with different layers through properly-defined interfaces.

View: This is the consumer interface layer, wherein the perspectives and look at controllers are defined. The view is chargeable for showing the information provided by way of the presenter and forwarding the person moves to the presenter.

Presenter: This is the presentation layer, where the good judgment for formatting and imparting the records is defined. The presenter is liable for fetching the records from the interactor, reworking it right into a suitable layout for the view, and updating the view hence. The presenter additionally handles the consumer movements acquired from the view and calls the router to navigate to other screens.

Interactor: This is the enterprise good judgment layer, where the common sense for manipulating the data and interacting with external services is described. The interactor is accountable for gaining access to the facts from the service layer, acting any vital operations on it, and returning it to the presenter. The interactor additionally communicates with the entity layer to store and retrieve the information fashions.

Entity: This is the information layer, wherein the data models and systems are described. The entity is responsible for representing the data in a constant and coherent manner throughout the software. The entity layer also can encompass records get entry to gadgets (DAOs) or repositories that summary the information of records patience and retrieval.

Router: This is the navigation layer, where the logic for routing and transitioning among different monitors is defined. The router is chargeable for developing and providing the view controllers, passing any vital facts to them, and coping with any dependencies or configurations. The router also communicates with the presenter to get hold of the navigation requests and execute them.

VI. Singleton Method Design Pattern

The singleton policy ensures that there is only one instance of a class and provides global access. This is especially useful when you want to manage a single instance of an object or control access to a delayed object.

VII. Factory Method Design Pattern

The Factory Method model defines an interface for creating an object but allows subclasses to modify the type of the created object. Especially useful when you need to create objects with a common interface but different functionality.

VIII. Observer Method Design Pattern

The observer structure defines one to many dependencies between objects, so when one object changes its state, all its dependents are automatically notified and updated. This is useful for scheduling distributed events.

IX. Dependency Injection (DI) Method Design Pattern

Dependency Injection is a method of providing class dependencies from the outside, rather than creating them in the class. It improves code modularity and testability by making classes independent of their dependencies.

X. Adapter Method Design Pattern

The adapter configuration allows you to use the interface of an existing class as a link to a new one. It is often used to work with others without modifying the source code of existing classes.

XI. Strategy Method Design Pattern

The strategy model defines a family of algorithms, contains each of them, and provides them with flexibility. It allows you to select the appropriate algorithm at runtime. This example is useful when you want to provide different options for a task.

XII. Composite Method Design Pattern

A composite pattern allows you to arrange objects in a tree structure to represent a part-of-the-whole structure. This is helpful when you have to deal with individual objects and sets of objects accurately.

References

[1] anujpato5vy, "Design patterns for mobile development," GeeksforGeeks, 04-Jan-2024. [Online]. Available: https://www.geeksforgeeks.org/design-patterns-for-mobile-development/. [Accessed: 01-Feb-2024]