Mobile App Architecture?

Dominguez Diaz Alma Juanita

January 8, 2024

What is mobile app architecture?

Mobile app architecture is the structural design and organization of a mobile application, outlining how various components and modules of the app are interconnected and work together to archive its functionality. The key components of mobile app architecture consist of the following layers:

1. User Interface (UI) Layer: The UI layer handles presenting the app to the user. It encompasses visual elements and components that users interact with, such as screens, buttons, forms, navigation menus, and graphical elements. This layer manages the app's layout and appearance.

2. Application Logic Layer: Also known as the business logic layer, this houses the app's core functionality. It includes algorithms, business rules, and processes that control the app's behavior. This layer handles user input manages data retrieval and storage and ensures the proper functioning of the app's features.

3. Data Layer: The data layer manages data storage, retrieval, and communication with external data sources. It involves databases, server APIs, and data repositories that the app interacts with. This layer ensures data integrity, security, and availability. Commonly used technologies in the data layer include databases (SQL or NoSQL), RESTful or GraphQL APIs, and caching mechanisms.

1 Types of mobile App Architecture

There are primarily five types of mobile application architecture:

1.1 Monolithic Architecture

In monolithic architecture, all components and modules of an application are tightly integrated into a single, unified unit. In a monolithic architecture, the entire application, including the user interface, application logic, and data storage, is bundled together as a single codebase and runs within a single process. Key Characteristics of Monolithic Architecture:

- In a monolithic architecture, all the code that makes up the application is part of one codebase.
- Tight Coupling: Components and modules within the monolith are tightly interconnected, meaning they are interdependent and often share resources and libraries. Changes to one part of the code can have ripple effects on other parts, making maintenance and updates more challenging.
- Single Deployment Unit: Monolithic applications are typically deployed. When you need to make changes or updates to the application, you redeploy the entire monolith, which can result in downtime during updates.
- Shared Data Storage: Data storage is centralized, and all components of the application share access to the same database or data store.

1.2 Microservices Architecture

In a microservices architecture, an application uses small, independent, and loosely coupled services that operate simultaneously to provide its functionality. Instead of building a monolithic application where all features and functions are tightly integrated into a single codebase, a microservices architecture breaks down the application into a collection of individual services, each responsible for a specific set of tasks or functionalities.

Key Characteristics of Microservices Architecture:

- Service Independence
- Services communicate with each other through well-defined, lightweight interfaces such as APIs (Application Programming Interfaces).
- Microservices are small and focused on a specific piece of functionality. Each service should perform a single task well.
- In a microservices architecture, different services can use different programming languages and technologies based on the specific requirements of the service.

- Independent Data Management.

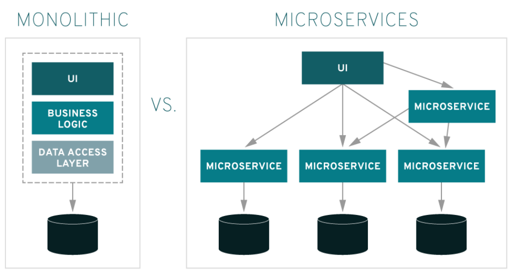

Figure 1: image about microservices architecture

1.3 Model-View-Controller (MVC)

It divides the application into three interconnected components, each with a specific role and responsibility. The primary purpose of the MVC pattern is to separate the concerns of an application, making it easier to develop, maintain, and extend.

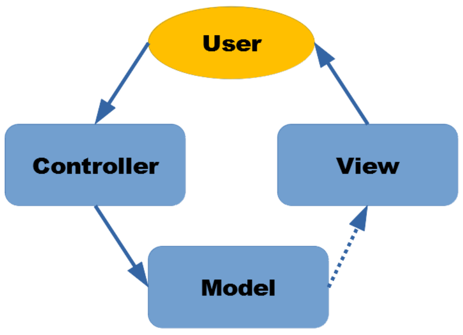

Figure 2: image about Model-View-Controller(MVC)

In app development, the model handles the data and business logic. It contains the main functions and manages the app's data. The view displays the user interface and shows data to users in a clear way. It gets information from the model and presents it nicely. The controller is like a middleman between the model and view. It gets user input, works with the model to get or change data, and decides which view to show based on what the user does.

1.4 Model-View-View Model (MVVM)

Is an architectural design pattern used primarily in software development for building user interfaces. It is an evolution of the Model-View-Controller (MVC) pattern, designed to enhance the separation of concerns and improve the testability and maintainability of code.

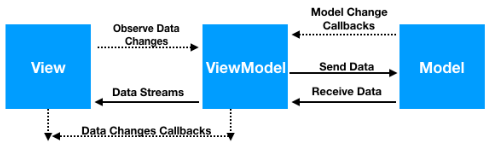

Figure 3: image about Model-View-View Model (MVVM)

1.5 Clean Architecture

Robert C. Martin introduced it provides a structured approach to building robust and adaptable software systems. The core idea of clean Architecture is to decouple distinct parts of the system and organize them in a way that prioritizes business logic and minimizes dependencies on external frameworks and technologies.