UTT

UNIVERSIDAD TECNOLÓGICA DE TIJUANA

GOBIERNO DE BAJA CALIFORNIA

Topic:

Selection of Design Patterns

By:

Arguelles Galvez Antonio

Group:

10B

Matter:

Integral Mobile Development

Teacher:

Ray Brunett Parra Galaviz

Date:

01/09/2025

The selection of design patterns is a crucial aspect of software development that helps address common design problems, ensuring that the application is scalable, maintainable, and easy to understand. Design patterns are reusable solutions to frequently occurring issues, providing best practices for software architecture.

Key Steps for Selecting Design Patterns

Understand the Problem Requirements

The first step is to thoroughly understand the requirements of the project. This includes analyzing the problem to identify the recurring design issues that a pattern could solve.

Classify the Type of Pattern Needed

Design patterns are categorized into three main types:

Creational Patterns: Deal with object creation mechanisms.

Structural Patterns: Focus on the composition of classes and objects.

Behavioral Patterns: Define how objects interact and communicate.

Analyze Potential Patterns

After identifying potential patterns, evaluate them based on the project's requirements. Consider factors such as complexity, flexibility, and scalability.

Consider the Pattern's Impact

Analyze the trade-offs, including the benefits and drawbacks, before finalizing the pattern. Ensure that the selected pattern does not introduce unnecessary complexity.

Choose a Pattern Based on the Framework

The framework you are using will influence the design pattern selection. For example, React Native works well with patterns like MVVM (Model-View-ViewModel) or Observer, which align with its component-based architecture.

MVVM (Model-View-ViewModel)

The MVVM pattern is a great fit for React Native because it provides a clear separation of concerns between the UI (View) and the business logic (Model). The ViewModel acts as a bridge between them, managing the state and logic of the application.

Structure of MVVM

Model

Manages the application's data and business logic.

Example: Fetching data from an API.

View

Represents the UI components and displays the data to the user.

Example: Screens, buttons, and forms in the app.

ViewModel

Handles the interaction between the View and Model.

Example: Manages state changes and handles user inputs.

Benefits of Using MVVM in React Native:

Improved Code Organization: Separation of concerns makes the codebase cleaner and more maintainable.

Reusable Components: Encourages reusable UI components across the application.

Easier Testing: The separation of business logic and UI makes unit testing more straightforward.

Better State Management: Works well with tools like Redux, MobX, or Zustand.

Conclusion

Selecting a suitable design pattern is essential for creating scalable and maintainable applications. For a framework like React Native, MVVM is a highly recommended design pattern as it aligns with React Native’s component-based structure. It simplifies the development process by improving state management, reusability, and testability.