Mobile Development, Native and Cross-Platform Applications, Design Patterns

Beltran Ramirez Erik Alfonso

03/01/24

1 Fundamentals of Mobile Development

Mobile development is a branch that focuses on creating applications for mobile devices such as phones and tablets. To fully understand its fundamentals, we need to consider the following:

- **Mobile Platforms:** There are various mobile platforms, with Android and iOS being the most prominent. Each has its own ecosystem and set of development tools.
- **Specific Operating Systems:** In-depth knowledge of the specific operating systems of each platform is crucial. This involves understanding the unique features of Android and iOS, as well as best practices to leverage them.
- **Technologies and Tools:** Familiarity with mobile development technologies and tools, such as Integrated Development Environments (IDE), frameworks, and Software Development Kits (SDKs).

2 Native and Cross-Platform Applications

Mobile application development is categorized into two main types: native and cross-platform.

2.1 Native Applications

Native applications are designed and developed specifically for a particular platform, maximizing the use of platform-specific features and functionalities. In the case of Android, languages like Java or Kotlin are used, while Swift and Objective-C are common for iOS.

2.2 Cross-Platform Applications

In contrast, cross-platform applications are developed to run on multiple platforms. This is achieved using technologies like React Native, Flutter, or Xamarin, enabling more efficient development by sharing code across platforms.

3 Design Patterns for Mobile Applications

Design patterns are reusable strategies that provide solutions to common problems in software development. Some fundamental design patterns for mobile applications include:

- **MVC (Model-View-Controller):** This pattern divides the application into three main components, facilitating modularity and code maintenance.
- **MVVM (Model-View-ViewModel):** Similar to MVC, but with an additional layer (ViewModel) that manages logic and the state of the user interface. This approach is especially useful in mobile development environments.
- **Singleton Pattern:** Ensures that a class has only one instance and provides a global access point to that instance. This is beneficial for efficiently managing shared resources.
- **Observer Pattern:** Defines a one-to-many dependency between objects, so that when one object changes state, all its dependents are notified. This pattern is valuable in situations where updates to multiple components are required.

4 Sources

1. Bachi. (2023, August 28). *Discover everything about mobile development: fundamentals, tips, and trends.* FelinoHost. https://felinohost.com/desarrollo-movil/descubre-todo-sobre-el-desarrollo-movil-fundamentos-consejos-y-tendencias/
2. Zapater, S. (2022, May 28). *Hybrid or Native App: Differences and Examples.* Hiberus Blog. https://www.hiberus.com/crecemos-contigo/app-hibrida-o-nativa/
3. Txema Rodríguez. (2011, April 2). *Android Patterns: Design patterns for developing Android applications..* https://www.genbeta.com/desarrollo/android-patterns-patrones-de-diseno-para-desarrollar-aplicaciones-android