PressAcademia
www.pressacademia.org/conferences
IFC
ISBN 978-625-97685-0-2
PressAcademia Procedia YEAR 2024 VOLUME 20
13TH ISTANBUL FINANCE CONGRESS, December 23, 2024 Istanbul, Turkiye.

# TRANSITION TO MODULAR ARCHITECTURE IN MOBILE FINANCE APPLICATIONS

**DOI: 10.17261/Pressacademia.2024.1917**
**PAP- V.20-2024(3)-p.10-13**

**Pınar Celdirme Kaygusuz**
Turkcell, Paycell Research and Development Center, Istanbul, Turkiye.
pinar.kaygusuz@turkcell.com.tr , ORCID: 0000-0003-3007-9963

**To cite this document**
Kaygusuz, P.C., (2024). A comparative nonparametric analysis on crypto currency exchange rate returns. PressAcademia Procedia (PAP), 20, 10-13.
**Permanent link to this document:** http://doi.org/10.17261/Pressacademia.2024.1917


## ABSTRACT

**Purpose-** As user expectations and market demands continue to evolve, the mobile Finance applications undergoes constant updates to ensure it remains both responsive to user needs and capable of delivering reliable and high-quality services. However, as applications grow in complexity, maintaining their scalability, flexibility, and manageability becomes increasingly challenging. In this context, adopting a modular software architecture emerges as a strategic solution, offering a more streamlined approach to reducing the intricacy of mobile applications while enhancing their adaptability and scalability. This paper's objective is to show clear comparison between moduler structure and monolitic application design. After discussing the benefits of moduler structure, a guideline will be given in order to transition moduler architecture.

**Mehodology-** The current structure is analyzed in dependencies aspect. The modules and how to eliminate dependencies while forming modules which can simplify development processes, improve application performance, and make updates more efficient. Furthermore, the study highlights the key advantages of this architectural shift, such as easier maintenance, faster feature deployment, and improved testing capabilities.

**Findings-**Specific criterias are outlined to guide the identification and definition of modules, ensuring that the modular design aligns with the application's objectives and delivers optimal benefits.

**Conclusion-** The study shows the positive contribution of modularization in Finance applications to support the fast changing demands.

**Keywords:** Modular structure, refactoring, dependency injection, modularization, mobile finance application
**JEL Codes:** E44, G01, G32

## 1. INTRODUCTION

Mobile finance applications are advanced systems designed to help users perform financial transactions quickly and securely. As these applications evolve with new features, they can become monolithic, making them harder to maintain, scale, and update. This complexity often leads to slower development cycles and higher operational risks [1].

Modular software architecture offers a solution by dividing an application into smaller, independent modules that focus on specific functionalities. This approach simplifies development, improves maintainability, and enhances scalability by allowing individual components to be updated or replaced without disrupting the entire system.[2]

Modular architecture refers to a design principle where a system is divided into distinct modules that can be independently replaced or modified. This approach enhances the flexibility, scalability, and sustainability of applications, especially in complex environments like financial systems. By breaking applications down into smaller, manageable components responsible for specific functionalities, modular architecture simplifies development processes and allows teams to work on different modules simultaneously without affecting the overall system. This not only reduces complexity but also streamlines updates and maintenance.[7]

Key advantages of modular architecture include rapid updates, where new features or bug fixes can be swiftly implemented; performance improvements, achieved by eliminating unnecessary dependencies; ease of maintenance, as each module can be independently updated or replaced; and improved testability, where unaltered modules can be excluded from testing when creating new versions.[5][8][4] Furthermore, defining clear boundaries and interfaces for module communication is essential for determining how modules will interact and depend on one another, ensuring seamless functionality.[9]

This study examines the modularization of a fintech application, focusing on the methods used to transition from a monolithic structure to a modular one. It highlights the benefits of modular architecture in streamlining development, improving performance, and ensuring that the application remains adaptable to evolving user needs.[6] The Methodology section provides detailed information about modular architecture

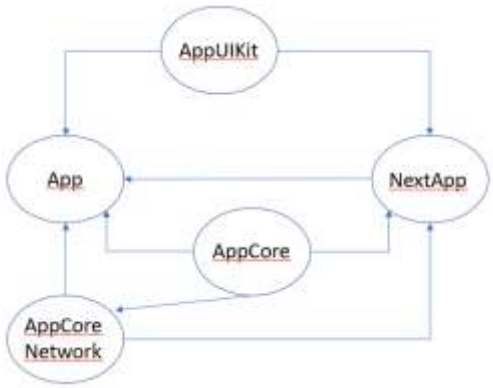
and its application in designing efficient and scalable systems, including strategies for managing dependencies. Finally, the Conclusions section consolidates the key findings of the study, evaluates its contributions to the fintech landscape, and discusses potential directions for future research.

## 2. METHODOLOGY AND ARCHITECTURE

### 2.1. Analysis of the Current Application

The first step is conducting a comprehensive analysis of the existing application architecture. This analysis identifies tightly coupled components and areas where modularization would provide the most benefit. It includes reviewing the application's functionality, performance metrics, and user requirements to ensure the modular design aligns with business objectives.[1]

**Figure 1: Old hierarchy**




**Key Elements to Evaluate During Analysis:**

**Functional Components**: Identify the core features of the application.

**Dependency Map:** Examine relationships between application components.[7]

**Performance Data:** Analyze the most frequently used features and performance bottlenecks.[5]

Following this evaluation, the next step is defining modular boundaries, which involves determining how to break the application into smaller, cohesive modules. These modules should work independently while seamlessly interacting with one another. This process often requires creating well-defined interfaces and communication protocols to maintain low coupling and high cohesion.[9]

### 2.2. Defining and Designing Modules

Modules should be identified and designed based on functionality. For the modular transformation of the fintech application, the proposed modules include:

**Screens Module**: Contains screens and algorithms working for the screens.

**CoreNetwork Module**: Includes the network layer, services, and request-response models.

**Infrastructure Module**: Handles core tasks like logging, linting, and managing various application managers.

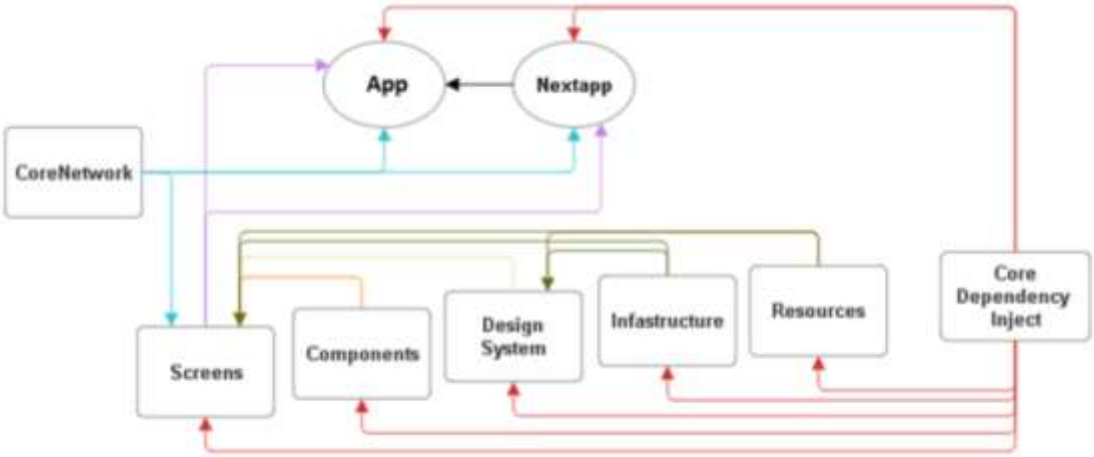
**DesignSystem Module**: Stores latest UI components.

**Resources Module**: Stores resources such as images and localization files used in the application.

**Components Module**: Contains former UI components to be used within screens.

**Core Dependency Inject Module**: Facilitates communication and integration between modules through dependency injection methods.

**Figure 2: Modules after Defining and Designing**




Once modular boundaries are established, the actual implementation of the modular architecture begins. This typically involves refactoring existing code to align with the new modular design. Some sections of the application may need to be rewritten to enhance modularity and ensure each module adheres to encapsulation and separation of concerns principles. Automated testing and continuous integration practices are also critical during this transition to provide rapid feedback and ensure the correctness of new modular components individually and as part of the broader system.[8]

## 2.3. Managing Dependencies

Managing dependencies is one of the significant advantages of adopting a modular architecture. By separating functionality into independent modules, developers can minimize the risk of one module's changes affecting others, thus reducing the likelihood of errors and increasing overall application stability. This modular approach also enables easier updates and improvements since individual modules can be modified or replaced without requiring an extensive overhaul of the entire system.[7]

## 2.4. Refactoring and Integration

Transitioning an application to a modular structure is not solely a technical undertaking; it often necessitates a significant cultural shift within the organization. This transformation requires teams to move beyond traditional development practices, embracing agile methodologies that promote flexibility, collaboration, and rapid iteration. An agile approach fosters a culture of continuous improvement and innovation, enabling teams to adapt quickly to evolving requirements and technological advancements. Such a shift is essential for ensuring that the modularization process not only achieves its technical goals but also aligns with the organization's broader objectives. For the fintech app, this cultural and technical alignment was facilitated by its existing agile framework, which provided a strong foundation for the modularization process. The organization demonstrated a comprehensive and dynamic transformation by undertaking code refactoring in parallel with ongoing feature development. This dual focus ensured that the application continued to evolve and meet user needs even as its architectural foundation was being restructured.

A phased transition plan was central to this success. Instead of attempting to overhaul the entire application in one massive update—a process that could disrupt user experience and introduce significant risks—the organization adopted an incremental approach. Modules were designed, integrated, and tested asynchronously, allowing the transformation to proceed smoothly and with minimal impact on the application's stability and performance. This method not only ensured a seamless user experience but also provided teams with valuable feedback at each stage, enabling continuous refinement and optimization of the modular architecture. Through this approach, the fintech company exemplified how organizations can effectively balance innovation with operational continuity during a arge-scale architectural transformation.

## 3. CONCLUSIONS

In conclusion, transitioning to a modular architecture offers numerous advantages, particularly in managing dependencies and increasing the overall agility of the application. The steps undertaken during the transition, from analyzing existing structures to defining and implementing modular components, play a decisive role in ensuring a successful transformation.

By adopting modularity, organizations can not only enhance operational efficiency but also better respond to evolving market demands and technological advancements. Modular architecture in mobile finance applications significantly simplifies dependency management. Each component in a modularized application can be developed and updated independently, introducing flexibility and speed into the software development process.

**Key Steps in the Transition Process**

**Code Refactoring**: Transforming monolithic structures into small, independent modules.

**Reducing Dependencies**: Isolating inter-module dependencies and employing dependency injection for required communication.

**Gradual Integration**: Executing the transition step-by-step without disrupting the user experience.

In summary, transitioning to modular architecture makes applications more sustainable, flexible, and manageable. This architectural approach supports the development of innovative solutions that meet user expectations.

Future work will focus on enhancing the modular architecture by implementing comprehensive automated testing for all modules. This will ensure that each module functions independently and integrates seamlessly with the broader system, reducing the risk of errors and speeding up deployment cycles. Additionally, efforts will be directed toward identifying and removing legacy code remnants that no longer align with the modular structure. Eliminating outdated code will not only improve maintainability but also enhance the overall performance and reliability of the application. These steps will help streamline development processes, maintain a clean codebase, and support the long-term scalability and sustainability of the project.

## Acknowledgement

This research is supported by the Paycell R&D Center. The authors would like to express their gratitude for the contributions from the Paycell R&D Center.

## REFERENCES

Al-Barakati, A. (2021). Leveraging Artificial Intelligence–enabled workflow framework for legacy transformation. International Journal of Advanced Computer Science and Applications, 12(12), 1–9. https://doi.org/10.14569/IJACSA.2021.0121239

Amirat, A. (2012). Generic model for software architecture evolution. In 2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT) (pp. 139–143). Kuala Lumpur, Malaysia. https://doi.org/10.1109/ACSAT.2012.79

Askhøj, C., Christensen, C. K. F., & Mortensen, N. H. (2021). Cross domain modularization tool: Mechanics, electronics, and software. Concurrent Engineering, 29(3), 221–235. https://doi.org/10.1177/1063293X211000331

Dörbecker, R., & Böhmann, T. (2013). The concept and effects of service modularity: A literature review. In Proceedings of the Annual Hawaii International Conference on System Sciences (HICSS) (pp. 1357–1366). https://doi.org/10.1109/HICSS.2013.22

Ghasemi, M., Sharafi, S. M., & Arman, A. (2015). Towards an analytical approach to measure modularity in software architecture design. Journal of Software, 10(4), 465–479.

Hasselbring, W., Wojcieszak, M., & Dustdar, S. (2021). Control flow versus data flow in distributed systems integration: Revival of flow-based programming for the industrial internet of things. IEEE Internet Computing, 25(4), 5–12. https://doi.org/10.1109/MIC.2021.3053712

Nooraei, M., & Mirzaie, M. (2023). An empirical analysis for software robustness vulnerability in terms of modularity quality. Systems Engineering, 26. https://doi.org/10.1002/sys.21686

Zheng, H., Kramer, J., & Chang, S. (2020). Auto-modularity enforcement framework using micro-service architecture. Journal of Visual Languages and Sentient Systems, 2020, 17–22.