Article **Journal Applied Computing**
June 2022, Vol.6 No.18 53-61

Processor and memory performance with design patterns in a native Android application

Rendimiento del procesador y memoria con patrones de diseño en una aplicación nativa de Android

SAMPAYO-RODRÍGUEZ, Carmen Jeannette†*, GONZÁLEZ-AMBRIZ, Rosalba, GONZÁLEZ-MARTÍNEZ, Blanca Areli and ALDANA-HERRERA, Jonathan

Tecnológico Nacional de México / Instituto Tecnológico Superior de Huauchinango, Ingeniería en Sistemas Computacionales, México.

ID 1st Author: *Carmen Jeannette, Sampayo-Rodriguez* / **ORC ID:** 0000-0001-8844-6055, **CVU CONACYT ID:** 951529

ID 1st Co-author: *Rosalba, González-Ambriz* / **ORC ID:** 0000-0001-5400-9754, **CVU CONACYT ID:** 368433

ID 2nd Co-author: *Blanca Areli, Gonzalez-Martinez* / **ORC ID:** 0000-0001-7313-4497, **CVU CONACYT ID:** 368551

ID 3rd Co-author: *Jonathan, Aldana-Herrera* / **ORC ID:** 0000-0001-7992-529X, **CVU CONACYT ID:** 1234604

DOI: 10.35429/JCA.2022.18.6.53.61 Received January 30, 2022; Accepted June 30, 2022

Abstract

The main objective of this article was to develop a native Android mobile application focused on local file storage, following different design patterns to compare the performance they had in processor and RAM memory consumption. To achieve this, the design patterns MVC, MVP and MVVM were taken as a sample, and for each one a native Android mobile application was developed to compare the performance they had when executed on the same device, thus concluding which design pattern consumed less processing resources and RAM memory. It was contributed to the area of software architecture and it was possible to test the hypothesis that the use of a software architecture design pattern applied in a native Android mobile application is a factor that influences the performance of use in CPU consumption and RAM memory. The pattern that least affects the device performance between MVC Pattern, MVP Pattern and MVVM Pattern is the MVVM with just a 3.5% increase in processor work and a record of a 17.5% increase in RAM consumption.

Patterns, Processing, Mobile, Application

Resumen

El objetivo principal del presente artículo fue desarrollar una aplicación móvil nativa de Android con enfoque en almacenamiento de archivos de manera local, siguiendo diferentes patrones de diseño para comparar el rendimiento que tuvieron en el consumo de procesador y memoria RAM. Para lograrlo se tomaron como muestra los patrones de diseño MVC, MVP y MVVM, y por cada uno se elaboró una aplicación móvil nativa de Android con lo que se comparó el rendimiento que tuvieron al momento de ejecutarse sobre el mismo dispositivo, con ello se logró concluir cuál fue el patrón de diseño que consumió menos recursos de procesamiento y memoria RAM. Se aportó al área de la arquitectura de software y se pudo comprobar la hipótesis de que el uso de un patrón de diseño de arquitectura de software aplicada en una aplicación móvil nativa de Android es un factor que influye en el rendimiento de uso en el consumo de CPU y la memoria RAM. El patrón que menos afecta el rendimiento del dispositivo entre Patrón MVC, Patrón MVP y el Patrón MVVM es el MVVM con apenas un aumento del 3.5% en el trabajo del procesador y un registro de un aumento del 17.5% en consumo de RAM.

Patrones, Procesamiento, Móvil, Aplicación

Citation: SAMPAYO-RODRÍGUEZ, Carmen Jeannette, GONZÁLEZ-AMBRIZ, Rosalba, GONZÁLEZ-MARTÍNEZ, Blanca Areli and ALDANA-HERRERA, Jonathan. Processor and memory performance with design patterns in a native Android application. Journal Applied Computing. 2022. 6-18:53-61.

* Correspondence to the Author (E-mail: cjean_80@hotmail.com)
† Researcher contributing as first author.

Introduction

Software development has been characterized by improving more and more in shorter periods of time. Since the software crisis until today, an exponential growth has been achieved that allows having everything we know today as software.

This enormous evolution for software development has managed to create a process engineering, which defines moments and characteristic points that are vital for the correct operation of a system, however, when it comes to a more specific product such as a mobile application, there are factors to take into account ranging from the language to be used to the type of application to be made (native or hybrid) and when using generic resources, an optimal expected result is not obtained. (Pizón Bayona & Sanabria Orjuela, 2022)

There is a wide range of software architecture patterns that can be used arbitrarily in the development of any system.

Notoriously a software architecture pattern may be more efficient in the development of systems with specific features, however, there is no accurate information on when a software architecture pattern will be more efficient for the specific development of a mobile application even though the Android operating system is the most widely used by users. (Arbulu Peralta, 2022).

When it comes to mobile development there is an extensive list of ways to develop an application, from hybrid applications to native applications, and when it comes to the development phase of the application, the way to structure the code will depend a lot on whether it is a native application or not.

At the present time, knowledge about the implementation of design patterns in native mobile development on Android is scarce. Research exists, but it is mostly for desktop applications and web development.

Software development is the process of describing a sequence of activities that must be followed by a team of workers to generate a coherent set of products (Abanto Cruz & Gonzáles Ramírez, 2019) which means that it is not only about complying with the functionality of the software, but to generate a quality product and/or service that meets high level standards and for this it is necessary to know more than just the methodology of the development itself, it is necessary to be able to evaluate which architecture will be implemented since this has an influence on the quality characteristics of the system. (Orellana Chicaiza & Velastegui Mejía, 2007)

Based on studies, it has been proven that design patterns have shown a greater impact on classic software development, of them, the main software design patterns for quality mobile development are: MVC, MVVM, MVP, BLOC and VIPER and with them, there are criteria for their analysis that allow you to decide which pattern fits best according to your needs. (Abanto Cruz*, op. cit.*)

In the field of mobile development there is not enough information on studies conducted in this specific area, however, in other areas of software development it was found that the microservices architecture pattern meets the notion of "separately deployed units" which refers to the fact that each component of the architecture is implemented separately, thus making the architecture pattern understandable to other domains that require the application of new techniques that allow to face continuous software deliveries in accelerated development contexts (Mamani Rodríguez, Del Pino, Rodríguez, & Gonzales Suarez, 2020), which consequently allows overcoming problems that were previously very marked, such as maintenance or change in the functionalities of the applications, which represented a problem due to its monolithic nature. (López & Maya, 2017)

Nowadays the software architecture of a system is important to achieve a high level of quality in many project requirements and thus be able to implement evaluation models of software architectures to prevent disasters of an architectural design that does not meet the quality requirements. (Orellana Chicaiza,op. cit.)

ISSN-2531-2952

It has also been found that the most focused works in software development address issues that increase usability, which are called "usability mechanisms" and should be taken into account from the beginning of the project, the functionality of each usability mechanism is represented by application scenarios. (Rodríguez Tibocha, 2014)

Consequently, we can say, that the idea of architectural patterns can also be used independently of architecture, since design solutions are provided, this only considering that the usability requirements are given at the first moment of development and evaluated by means of architectural patterns so that usability improvements are provided in the final problem. (Idem)

Design patterns

Design patterns have a wide variety of uses that have been employed and tested in practice. They have been shown to be effective in software development to simplify the overall design of applications. Design patterns make software more reusable, which can reduce production cost and development time. Design patterns are very useful for developers and designers as they encapsulate expertise, provide a common vocabulary, and improve the documentation of software designs.

Types of design patterns:

- Delegated.
- Compound.
- Decorator.
- Mediator.
- Iterator.
- Observer.

Design patterns have had massive influence on software development. Like web applications, mobile application implementation also established some proven patterns and standards to overcome the challenges and limitations of mobile development. Most of the mobile applications were developed with low quality code and are not based on architectural design patterns.

Developments of a mobile application with the right design pattern can effectively synchronize the user interface with data models and business logic, this will influence how your source code should look like. There are several architecture design patterns for mobile development. In the table below we show the following design patterns for mobile applications. (Abanto Cruz, op. cit.)

Platform	Design patterns
iOS	Abstract Factory, Adapter, Factory Method, Template, Singleton, MVC
Android	MVC, Model View View-Model, Model View Presenter, BLOC Viper

Table 1 Design patterns and platforms
Source: Own elaboration

Usability pattern

The aspects that these patterns deal with basically refer to user interfaces, so they are also called interface patterns or interaction design patterns. The most widely used pattern concept in software development is the design pattern used particularly in the object-oriented paradigm, which in this context refers to a description of classes and objects working together to solve a particular problem.

According to the degree of abstraction, there are several possible classifications of patterns, some proposed categories are:

- Design patterns.
- Architecture patterns.
- Analysis patterns.
- Creation patterns.
- Behavioral patterns.
- Requirements modeling patterns.
- Organizational patterns.
- Programming patterns.

ISSN-2531-2952

Software architecture

In the book "Software Engineering. A Practical Approach", by author Roger S. Pressman, an internationally recognized authority and PhD in Engineering Physics from the University of Connecticut, USA, the following definition of software architecture is given: "The software architecture of a program or computer system is the structure of the system structures, which comprises the software components, the properties of those externally visible components, and the relationships between them".

Design patterns

They provide a proven and documented solution to software development problems that are subject to similar contexts. Design patterns facilitate the reuse of successful software architectures and designs; they are oriented to how to organize the source code and how the different parts of the software interact. (Melgar Sasieta, 2020)

Design patterns and their relationship in software architecture

Since design patterns are applied to different solutions and can be implemented in different architectures, the best features of each pattern can be extracted and applied to the best points of an architecture to achieve a model that completely fits our needs.

MVC

- Its foundation is the separation of the code into three different layers, bounded by their responsibility, in what are called Models, Views and Controllers, or what is the same, Model, Views & Controllers. It is a design pattern used in all modern web applications (Toapanta, Palacios, Chito & De la Torre, 2022).

MVVM

- MVVM, Model View ViewModel, is a design pattern that aims to separate the user interface part (hence the V for View) from the business logic part (hence the M for Model), thus making the visual part totally independent. The other component is the ViewModel, which is the part that will interact as a bridge between the View and the Model.

MVP

- MVP (Model View Presenter View) is another design pattern that aims to separate the view layer from the logic, perform unit tests, and write cleaner code, etc.

BLOC

- BLoC stands for Business Logic Component.

- The Google team's goal in designing this pattern was code reuse between their mobile applications, using Flutter with Dart, and web, using Angular Dart.

VIPER

- In VIPER we find more subdivisions than in other architecture patterns, since each of its components must be responsible for a single task.

- VIPER stands for: View, Interactor, Presenter, Entity and Router.

Separately Deployed Units

Usability requirement

- The main factors to be considered when talking about usability are the ease of learning, the effectiveness of use and the satisfaction with which people are able to perform their tasks when using the product, all of which rest on the foundations of user-centered design.

ISSN-2531-2952

Types of design patterns

- Delegated: uses inheritance to delegate operations on dependencies with static characteristics.
- Composite: uses hierarchies to handle primary and composite objects in a uniform way.
- Decorator: allows combining features in a dynamic way.
- Mediator: coordinates communication between objects of different classes.
- Iterator: performs traversals on composite objects independently of their implementation.
- Observer: creates a one-to-many dependency between objects so that all objects involved can be automatically updated.

Monolithic application

A monolithic application refers to a software application in which the user interface layer, business logic and data access layer are combined in the same program and on the same platform.

The research contributes to the area of software architecture, shows the results of the comparison of applying design patterns in software development, measuring memory and processor performance parameters of a native Android mobile application, focused on storing files locally.

In addition to referring to design patterns, where it is explained what they are and how they are implemented, along with what a usability pattern is, how it relates to software architecture and how it involves design patterns. The methodology used to carry out the analysis is presented along with tables and graphs. Subsequently, the analysis of the results obtained and the conclusions are presented.

Methodology to be developed

The classical research methodology (Hernández Sampieri, Fernández Collado, & Baptista, 2014) was employed, consisting of the following steps:

- Scope and focus of the research.
- Hypothesis.
- Research design.
- Sample selection.
- Data collection.
- Data preparation.

Scope and focus of the investigation

A descriptive type of research with a quantitative approach was carried out.

Definition of the Research Hypothesis

H_0: The use of a software architecture design pattern applied in a native Android mobile application is a factor influencing usage performance in CPU and RAM consumption.

H_1: The use of design patterns is not a factor when measuring the performance in CPU and RAM consumption of a native mobile application on Android.

Research Design

The research was designed with a quantitative and experimental approach.

- Develop 3 test applications focused on local file storage, with the same functionality applying a different design pattern.
- Measure the processor and RAM performance when running the application on a mobile device with the following characteristics.

ISSN-2531-2952

Feature	Description
Type (Physical / Emulators)	Physical
Brand	Xiaomi
Model	Redmi 9 - M2004J19G
Range (high, medium, low)	Medium
OS Type and Version	Android 10 MIUI 12.0.2
CPU (Model and Capacity)	Helium G60
RAM Memory	4 GB
Storage (Capacity)	64 GB
Battery Type	5020 mAh Fast loading 18W

Table 2 Characteristics of the device to be evaluated
Source: Own elaboration

Sample selection

The three types of design patterns most frequently encountered in application development were chosen:

- MVC Model View Controller.
- MVP Model ViewPresenter Model.
- MVVM Model ViewModel.

Data Collection

The freely downloadable AIDA64 mobile application was used to obtain the processor and RAM performance data of the device at the time of execution of the native mobile test application.

Data preparation

A table with the data obtained and graphs showing the comparison between processing performance and RAM and the three design patterns are presented.

Results

Pilot Test - Local Storage Application.

- Data collection and processing.

Device	Pattern		
	MVC	MVP	MVVM
Xiaomi Redmi 9	TR1.1	TR1.2	TR1.3

Table 3 Comparison of data
Source: Own elaboration

Type of monitoring on the device		
Before, during and after execution	Element	Result
	CPU	In the CPU element, logging was started taking as 100% the value of 2000MHz. The logging was started with an original consumption of 25% and an increase to 57.9% was recorded and when the application was closed it reached a consumption of 25%.
	RAM memory	In the RAM memory element, logging started with a value of 42.5% as 100% and when the application was closed it reached a consumption of 25% 3754MB. The log started with an original consumption of 65.13% and increased to 69.36% and when the application was closed it reached a consumption of 69.09%.
	Battery	Battery usage was not affected
	Data Use	The application does not consume data

Table 4 Table of results 1.1 - TR1
Source: Own elaboration

Type of monitoring on the device		
Before, during and after execution	Element	Result
	CPU	In the CPU element, logging was started taking as 100% the value of 2000MHz. The logging was started with an original consumption of 25% and an increase to 46.85% was recorded and when the application was closed it reached a consumption of 25%.
	RAM memory	In the RAM memory element, logging was started taking as 100% the value 3754MB. Logging started with an original consumption of 66.43% and increased to 73.36% and when the application was closed it reached a consumption of 69.97%.
	Battery	Battery usage was not affected.
	Data Use	The application does not consume data

Table 5 Table of results 1.2 - TR1.2
Source: Own elaboration

ISSN-2531-2952

Type of monitoring on the device		
Before, during and after execution	Element	Result
	CPU	In the CPU element, logging was started taking as 100% the value of 2000 MHz. The logging was started with an original consumption of 25% and an increase to 42.5% was recorded and when the application was closed, the consumption reached 25%.
	RAM memory	In the RAM memory element, logging was started taking as 100% the value 3754MB. The log was started with an original consumption of 65.37% and an increase to 68.56% was recorded and when the application was closed it reached a consumption of 65.30%.
	Battery	Battery usage unaffected
	Data Use	The application does not consume data

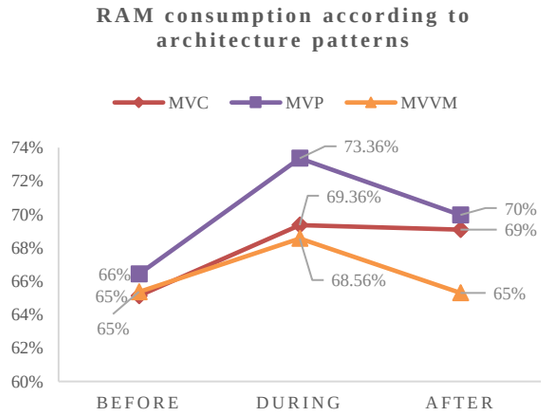
Table 6 Table of results 1.3 - TR1.3
Source: Own elaboration

Analysis of results

For the realization of this research, the development of a native Android mobile application in the Java programming language was carried out. This application was planned applying functionalities that are directly related to data storage and persistence, which would be a relatively significant and completely valid level of stress to analyze. The application was built from scratch three times varying one specific element, the architecture pattern. In the following tables you can see the analysis of the resource consumption of both RAM and CPU, information that will serve as a parameter to measure the performance of the application on the device in general.

Analysis of RAM consumption RAM memory			
Monitoring in execution	Before	During	After
MVC	65%	69.36%	69%
MVP	66%	73.36%	70%
MVVM	65%	68.56%	65%

Table 7 Analysis of RAM memory consumption
Source: Own elaboration

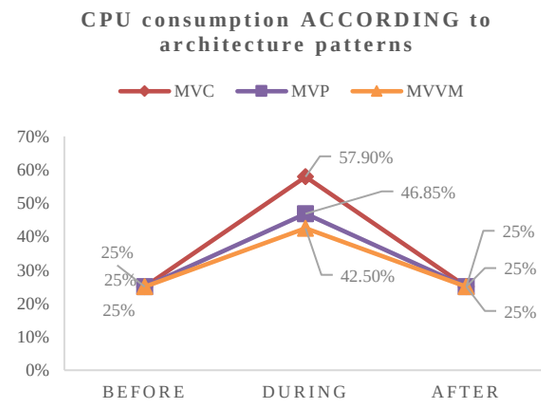


Graphic 1 RAM consumption according to architecture patterns
Source: Own elaboration

In the same way as the previous graphs, CPU resource consumption was measured with the same application on three different occasions, varying the architecture pattern at the time of development. As we can observe in Table 2 CPU consumption analysis, and confirm in Graph 2: CPU consumption according to architecture patterns, it was clearly noticed that the pattern that offers a lower performance is the MVC, since it is the one that increased the most with respect to the other two. However, we can also notice that, at runtime, the pattern that offers a more optimal performance and without consuming so many resources is the same as in the previous results, the MVVM pattern.

Analysis of CPU consumption			
Monitoring in execution	Before	During	After
MVC	25%	57.90%	25%
MVP	25%	46.85%	25%
MVVM	25%	42.50%	25%

Table 8 Analysis of CPU consumption
Source: Own elaboration



Graphic 2 CPU consumption according to architecture patterns
Source: Own elaboration

ISSN-2531-2952

Acknowledgment

We thank the Tecnológico Nacional de México/Instituto Tecnológico Superior de Huauchinango for the opportunity and support to conduct and publish this research.

Financing

Prototype: this work has been financed by TecNM/ITSH [SPeI-SR-001-2022].

Conclusions

In conclusion, the results obtained from the pilot tests validate the hypothesis, and confirm that choosing a specific software architecture pattern, such as the MVVM, when developing a mobile application will have an impact that will be directly related to both the performance of the software when consuming resources on the mobile device and its scalability when developing and building it.

In the performance tests, it was found that the pattern that most affected the performance of the device at the time of application execution was the MVC, since it increased the processor work by 32% together with a 4.3% increase in RAM consumption, while the pattern that least affected the performance of the device was the MVVM with only a 3.5% increase in processor work and a 17.5% increase in RAM consumption.

However, it is important to highlight that the MVP pattern has an interesting resource consumption, since it registers a 21.8% increase in processor work, which shows that it increases in less quantity than the MVC, but with a 7.36% RAM consumption, that is, more than the MVC. This situation is decisive at the time of development, because, in addition to the functional and non-functional requirements, it must be taken into account whether the type of application to be developed needs to be optimized to process a large amount of information or to be optimized for the exhaustive use of RAM memory.

References

Abanto Cruz, J. A., & Gonzáles Ramírez, O. F. (2019). Análisis comparativo de patrones de diseño de software para el desarrollo de aplicaciones móviles de calidad: Una revisión sistemática de la literatura. *Repositorio de la Universidad Peruana Unión*. Lima. URL: https://repositorio.upeu.edu.pe/handle/20.500.12840/2525

Arbulu Peralta, M. (2022). Propuesta de aplicación móvil para el Minimarket Porton y su efecto en la Satisfacción del cliente. Repositorio Institucional de la UTP. Perú. URL: https://repositorio.utp.edu.pe/handle/20.500.12867/5510

Hernández Sampieri, R., Fernández Collado, C. y Baptista, P. (2014). Metodología de la investigación. 6ª. ed. México: Mc Graw-Hill.

López, D., & Maya, E. (2017). Arquitectura de software basada en microservicios para desarrollo de aplicaciones web. *Repositorio de RedCLARA*. Ecuador. URL: https://dspace.redclara.net/handle/10786/1277

Mamani Rodríguez, Z., Del Pino, Rodríguez, L., & Gonzales Suarez, J. C. (2020). Arquitectura basada en Microservicios y DevOps para una ingeniería de software continua. *Revista de Investigación Industrial Data*. 23(2). 141-149. URL: https://revistasinvestigacion.unmsm.edu.pe/index.php/idata/article/view/17278. DOI: https://doi.org/10.15381/idata.v23i2.17278

Melgar Sasieta, H.A. & Huari Casas, M.R. (2020). Revisión sistemática sobre generadores de código fuente y patrones de arquitectura. Tesis Maestría. *Repositorio de la Pontificia Universidad Católica del Perú*. San Miguel. URL: https://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/16457

Orellana Chicaiza, A. C., & Velastegui Mejía, V. A. (2007). Evaluación de la arquitectura de software de aplicaciones en producción. Tesis Ingeniería. *Repositorio Digital Institucional de la Escuela Politécnica Nacional*. Quito. URL: http://bibdigital.epn.edu.ec/handle/15000/414

ISSN-2531-2952

Pinzón Bayona, G., & Sanabria Orjuela, Y. G. (2022). Desarrollo de una aplicación móvil para traductor de lenguaje de señas mediante el uso de servicios web. *Repositorio Institucional de la Universidad Católica de Colombia*. Colombia. URL: https://repository.ucatolica.edu.co/handle/10983/26989

Pressman, R. (2021). Ingeniería del Software, un Enfoque Práctico. 9ª. ed. McGraw Hill. México.

Rodríguez Tibocha, F. D. (2014). Obtención y uso de patrones para la implementación de funcionalidades de usabilidad en aplicaciones web. Tesis Doctoral. *Archivo Digital Universidad Politécnica de Madrid*. Madrid. URL: https://oa.upm.es/33528/

Toapanta, W. V. C., Palacios, J. D. R. G., Chito, A. M. T., & De la Torre, L. A. (2022). Aplicación web–móvil para la gestión de productores agropecuarios del gobierno autónomo descentralizado del cantón mocha. *Universidad y Sociedad*, 14(3), 487-492. URL: https://rus.ucf.edu.cu/index.php/rus/article/view/2889/2842.

ISSN-2531-2952