Journal Homepage: - www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI: 10.21474/IJAR01/17909
DOI URL: http://dx.doi.org/10.21474/IJAR01/17909

RESEARCH ARTICLE

LOGCAT ANALYSIS OF MOBILE DEVICES TO IDENTIFY ERRORS

João Gabriel Ferreira Aguiar[1], Vinícius Gabriel Carvalho da Silva[1], Vinícius Gabriel Oliveira Pontes[1] and Pablo Augusto da Paz Elleres[2]

1. Information Systems Student.
2. Lecturer in Information Systems.

Manuscript Info

Manuscript History
Received: 17 September 2023
Final Accepted: 24 October 2023
Published: November 2023

Key words:-
LogCat, Mobile Devices, Error Analysis, Log Filtering, Log Identification

Abstract

This study seeks to address the growing complexity in analyzing mobile device logs, a challenging task for testing professionals and developers. The diversity of applications and platforms results in a wide range of LogCat records, many of which are not relevant to the error identification process. To solve this problem, we have developed a highly efficient log filtering system. This solution employs advanced text processing and semantic analysis algorithms, capable of discerning between informative messages and those that indicate genuine malfunctions. The implementation of this system has resulted in a notable reduction in the time spent sorting and identifying errors, providing a more accurate and efficient analysis of LogCat logs. The results show a significant improvement in the effectiveness of the testing process and in the final quality of the applications developed.

Introduction:-

In recent years, the exponential proliferation of mobile applications in conjunction with the diversification of platforms and operating environments has imposed a substantial challenge for testing professionals and software developers. In this scenario, the analysis of logs from mobile devices plays a critical role in identifying and resolving errors, providing valuable insights into the performance and operational integrity of applications (Log Analysis on Mobile Devices,[n.d.]).

However, the growing heterogeneity and volume of data generated by LogCat has led to information overload, making it difficult to effectively identify relevant errors amid a flood of trivial and informative messages (Log analysis on mobile devices,[n.d.]). This situation leads to inefficiency in the testing process, delaying the delivery of functional applications and increasing development costs.

To address this problem, we propose the implementation of a mobile device log filtering system, a sophisticated mechanism based on text processing algorithms and advanced semantic analysis. This system is designed to efficiently identify and isolate specific and critical log messages.

The design and development of this system aims to optimize efficiency in LogCat analysis, allowing test engineers and developers to focus on solving essential problems, to the detriment of manually sorting through a seemingly

Corresponding Author:- João Gabriel Ferreira Aguiar
Address:- Information Systems Student.

endless sea of information. In doing so, we anticipate a substantial improvement in the efficiency of the testing process and, ultimately, in the quality of the resulting software.

This study looks not only at the technical feasibility of this system, but also at the tangible impact it can have on the mobile application development cycle. By leveraging innovative approaches to log analysis, we anticipate a significant advance in the productivity and effectiveness of the mobile development industry.

Theoretical framework

Mobile devices such as smartphones and tablets are increasingly present in our daily lives. They are governed by operating systems that coordinate their functions. These operating systems, such as Android and iOS, work in different layers, from the system core to interactive applications. The mobility of these devices brings specific challenges, and log analysis is essential for diagnosing and resolving faults.

Mobile device systems

Operating systems for mobile devices are designed to be lightweight and efficient in order to guarantee good performance on devices with limited resources. They are responsible for managing memory, processing, networking, security and other aspects of the device. The most popular operating systems for mobile devices are Google's Android and Apple's iOS (GCFGlobal, [n.d.]; Operating System 2022).

Applications for mobile devices are developed to perform specific functions, such as games, social networks, messaging and so on. They are designed to be lightweight and efficient, so as to ensure good performance on devices with limited resources. In addition, apps can be developed to access specific features of the device, such as the camera, GPS, accelerometer, among others (GCFGlobal, [n.d.]; Mobile, 2021).

The mobility of mobile devices brings specific challenges, such as the need to ensure data security, compatibility with different devices and operating systems, and performance optimization. Log analysis is essential for diagnosing and resolving faults, allowing developers to identify problems and improve the quality of applications (GCFGlobal, [n.d.]; Mobile, 2021).

Programming for mobile devices is a discipline that specializes in creating specific software to work on smartphones and tablets. This area of programming is responsible for bringing to life the applications we use every day, from social networks and games to productivity apps and delivery services (Mobile, 2021).

LogCat is a software debugging tool for Android devices. It allows developers to view log messages generated by the system and applications in real time. LogCat is essential for developing applications for Android devices, as it allows developers to identify and solve problems efficiently.

LogCat

LogCat allows developers to view log messages at different priority levels, such as debug, info, warning and error. It can be used in a variety of ways. For example, developers can use LogCat to debug applications in real time, monitor system and application performance, and identify security problems. In addition, LogCat can be used to collect diagnostic information for error reporting and data analysis purposes (Android Developers, 2023).

In summary, LogCat is an essential tool for Android application developers, offering visualization and filtering of log messages in real time. Its versatility makes it indispensable for solving a variety of development challenges (Android Developers, 2023)

Operating system records, known as logs, are documents in text format that store information about the sequence of events that occurred in the system. These logs are generated automatically by the operating system and play a crucial role in detecting and resolving possible system failures or problems.

Identifying errors in the system log

Analyzing operating system logs is a practice for identifying problems in operating systems and applications. These records contain information about the chronology of events, including errors and failures. This approach allows developers to identify and solve problems efficiently (HostMídia, [n.d.]).

Operating system logs can be used to identify errors at different levels, from system errors to application errors. Identifying errors in the system log can be done using log analysis tools, which allow developers to view log messages in real time (Microsoft Learn, [n.d.]).

Log analysis can help identify security problems, such as intrusion attempts and suspicious activity. In addition, log analysis can help identify performance problems, such as network bottlenecks and memory problems.

In short, fault detection through system logs is a crucial practice for diagnosing possible problems in operating systems and applications. These records provide a detailed view of the sequence of events in the operating system, ranging from normal occurrences to error and failure situations. Meticulous log analysis is an essential tool for developers, enabling them to identify and resolve issues quickly and effectively (HostMídia, [n.d.]).

The prototyping phase plays a crucial role in the product and system development cycle. By allowing engineers to test and evaluate the functioning of the product at an early stage, it contributes to reducing costs and production time, while ensuring the excellence of the end result. A variety of techniques and tools, from initial sketches to fully functional models, can be employed in this process.

Prototype

A prototype represents an initial version or preliminary edition of a product or system. Its purpose is to test functionality before mass production. This stage allows developers to identify and correct possible problems and flaws before the final product reaches the market. In addition, prototypes are valuable tools for obtaining feedback from users and presenting the product to potential investors or customers (Techopedia, [n.d.]).

According to Ulrich and Eppinger (2015), prototyping is an important tool in the product development process, as it allows developers to test and evaluate different design concepts and solutions before investing time and resources in mass production. The prototype can be used to assess the functionality, ergonomics, aesthetics and technical feasibility of the product.

In short, prototypes are undeniably important in the product development cycle, giving engineers the opportunity to evaluate different design approaches before going fully into mass production. They are crucial for identifying and correcting flaws, obtaining feedback from users and optimizing development time and costs.

Materials and Methods:-

Log Filtering System

The Filtering System developed is an application designed to improve the management and analysis of logs from mobile devices. Based on Python and adb technologies, the system operates as a multifaceted platform that offers critical functionalities to optimize the interpretation of log data and the identification of specific errors.

Technical Description of the Filtration System

The Filtering System consists of an application that operates in a Python Tkinter environment, allowing it to run on the desktop device. This application is designed to interact with the mobile device's logcat, which is a command-line utility widely used in the Android ecosystem for capturing and viewing event logs in real time.

System Objectives and Scope

The primary purpose of the Filtering System is to provide testers and developers with an efficient way of analyzing and isolating errors in mobile device logs. By applying customizable filters, the system aims to make it easier to identify specific errors, as well as providing a clearer and more organized view of the event log.

Functional Specifications of the Filtering System

1. Real-Time Logcat: The system continuously monitors the device's logcat, allowing instant visualization of events as they are generated.
2. Filtered Logcat: Makes it possible to apply filters based on user-defined criteria. This means that you can select specific types of events (e.g. errors, warnings) for display.
3. Filter field: Provides an input field in which the user can specify the desired filter criteria, such as specific keywords or tags.

4. Apply Filter button: By pressing this button, the specified filter criteria are applied to the logcat, resulting in the display of only those events that match the defined criteria.
5. Stop Logcat button: Allows temporary interruption of logcat event capture, providing additional control over the display of logs.
6. Clear button: Clears the current logcat display, providing a clean screen for a new analysis.
7. Error Counter: Keeps an up-to-date record of the number of events considered to be errors, providing a quantitative metric of the severity of the problems identified.

Technologies used

The filtering system was implemented by integrating three key technologies: Python, Python-tkinter and adb. The choice of these technologies was strategically defined in order to create a robust and efficient application.

Python is a high-level, interpreted, object-oriented programming language that is widely used in software development. Its use in the Filtering System made it possible to create an efficient and scalable system, with a wide variety of libraries available for manipulating data and processing information (Python Software Foundation, 2023).

Python-tkinter is a Python library for creating graphical user interfaces. Its use in the Filtering System has made it possible to create an intuitive and easy-to-use interface for users, with advanced customization features (Python Software Foundation, 2023).

Adb is a command-line tool that allows communication with Android devices. Its integration into the Filtering System has enabled efficient manipulation and analysis of the device's log data, contributing significantly to the system's core functionality (Android Developers, 2023).

Prototype

The system's home page presents a dynamic and immersive view of the device's information. Here, LogCat is updated in real time, displaying a distinct color palette that categorizes each type of log. In addition, it is possible to identify filtered logs to improve the accuracy of information visualization.

In addition, an additional screen dedicated exclusively to "Fatal Error" incidents is provided. In this context, attention is focused on the most critical events, offering a more in-depth analysis and making it easier to take quick action to keep operations running smoothly. In this way, data is transformed into insights, elevating the user experience to a level of greater intuition and efficiency.

Coding

```
self.device_info_title_label_logcat = Label(self.left_frame_logcat, text="Informações do Dispositivo",
                                            font=("Arial", 12, "bold"))
self.device_info_title_label_logcat.grid(row=0, column=0, columnspan=2, pady=5)

self.product_name_label_logcat = Label(self.left_frame_logcat, text="Product Name:", font=("Arial", 12, "bold"))
self.product_name_label_logcat.grid(row=1, column=0, sticky="w", pady=5)

self.product_name_value_label_logcat = Label(self.left_frame_logcat, text="Aguarde...", font=("Arial", 12))
self.product_name_value_label_logcat.grid(row=1, column=1, sticky="w", pady=5)

self.carrier_label_logcat = Label(self.left_frame_logcat, text="Carrier ID:", font=("Arial", 12, "bold"))
self.carrier_label_logcat.grid(row=2, column=0, sticky="w", pady=5)

self.carrier_value_label_logcat = Label(self.left_frame_logcat, text="Aguarde...", font=("Arial", 12))
self.carrier_value_label_logcat.grid(row=2, column=1, sticky="w", pady=5)

self.android_label_logcat = Label(self.left_frame_logcat, text="Android:", font=("Arial", 12, "bold"))
self.android_label_logcat.grid(row=3, column=0, sticky="w", pady=5)

self.android_value_label_logcat = Label(self.left_frame_logcat, text="Aguarde...", font=("Arial", 12))
self.android_value_label_logcat.grid(row=3, column=1, sticky="w", pady=5)
```

This code snippet is building a graphical interface using the tkinter library in Python to display information about an Android device. It creates labels to show the product name, operator ID and Android version. These labels are organized in rows and columns within a frame called self.left_frame_logcat. Each label is configured with an initial text, such as "Please wait...", which will be updated later with actual device information.

```
# Informações do técnicas do dispositivo - OK

1 usage
def get_product_name():...
1 usage
def get_android_version():...
1 usage
def get_carrier_id():...
1 usage
def get_sales_code():...
2 usages
def update_device_info():...
```

This Python code contains a series of functions that use the adb (Android Debug Bridge) command to obtain technical information from an Android device. Each function corresponds to a specific type of information, such as the product name, Android version, operator identifier and sales code. The update_device_info function calls the other functions to collect and return all this information at once. This information can be used in other parts of the program.

```
def start_logcat(log_text, filtered_text, filter_keywords):
    global logcat_running, logcat_process, command_clear_buffer

    if logcat_running:
        return

    logcat_command = "adb logcat"
    logcat_process = subprocess.Popen(logcat_command, shell=True, stdout=subprocess.PIPE,
                                      stderr=subprocess.PIPE, text=True)

    logcat_running = True

    xlog_text = log_text
    xfiltered_text = filtered_text

    def read_logcat_output():...

    logcat_thread = threading.Thread(target=read_logcat_output)
    logcat_thread.start()
```

This Python code defines a function called start_logcat that starts capturing logs from an Android device using the adb logcat command. It reads the logcat output in a separate thread and updates a graphical interface with the logs. If the line contains specific keywords, it is also displayed in another part of the interface. The code uses the subprocess module to start the logcat process and threads to read the logs asynchronously.

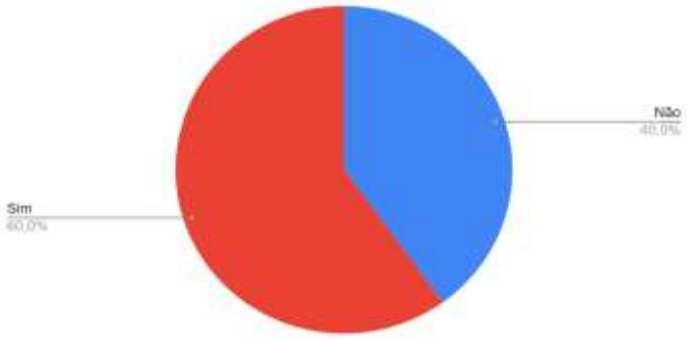
Results and Discussions:-

The testing stage lasted 10 working days, with the tool being used extensively. Five participants were selected for each evaluation phase. At the end of each iteration, five questions were asked about the system's performance, usability and possible future optimizations.

The purpose of the initial test (Test 1) was a discriminating analysis of the strengths and shortcomings of the proposed solution. As a result of the observations and results obtained, the second iteration (Test 2) was implemented as an extension focused on rectifying the shortcomings identified. This approach was designed to improve the user experience and ensure the delivery of a high-quality system.

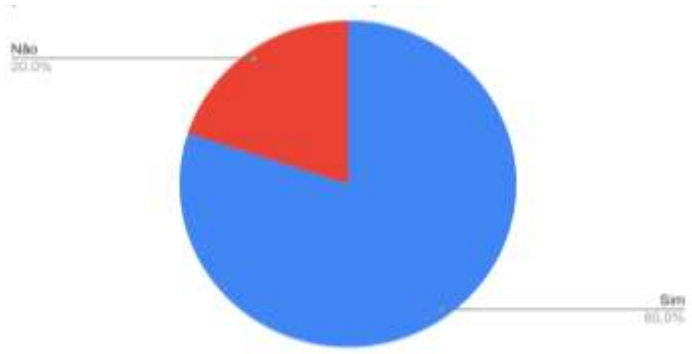
First test scenario

- Is the system easy to use and navigate to filter and view specific logs?



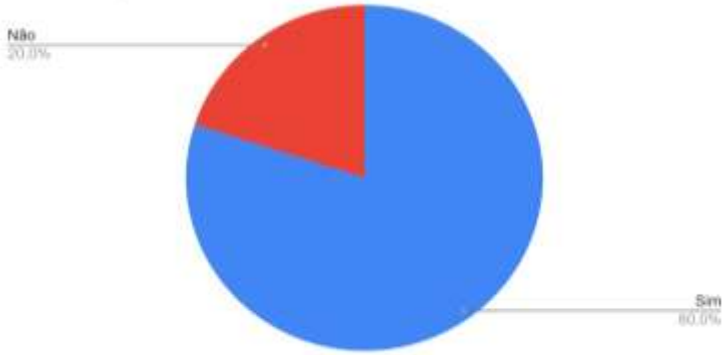
Result: 3 / 60% Yes - 2 / 40% No

- Have you noticed a significant reduction in the time needed to identify and analyze specific errors using the system compared to conventional methods?



Result: 4 / 80% Yes - 1 / 20% No

- Was the information in the filtered logs presented clearly and legibly?



Result: 4 / 80% Yes - 1 / 20% No

- Did the system adequately meet the specific requirements of your log analysis context?

Result: 5 / 100% Yes - 0 / 0% No

- Do you have any additional feedback on the system?

Result: 3 / 60% Yes - 2 / 40% No

Second test scenario

- During the second test, the system continued to be easy to use and navigate to filter and view specific logs?

Result: 5 / 100% Yes - 0 / 0% No

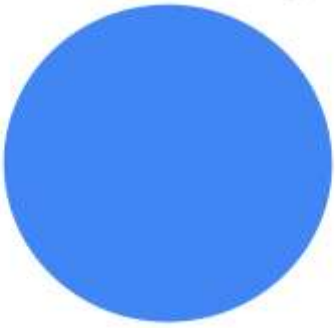
- Did you notice a significant reduction in the time needed to identify and analyze specific errors during the second test, compared to conventional methods?

Result: 5 / 100% Yes - 0 / 0% No

- Was the information in the logs filtered during the second test presented clearly and legibly?

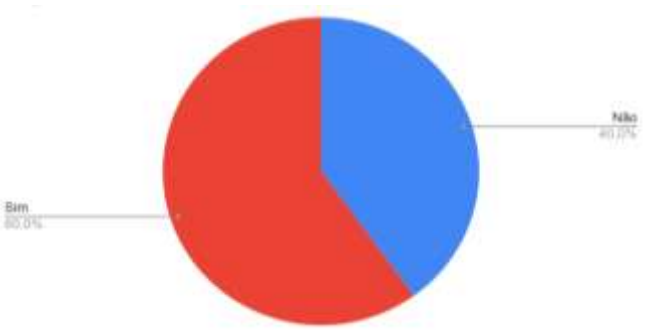
Result: 5 / 100% Yes - 0 / 0% No

- Did the system adequately meet the specific requirements of the log analysis context during the second test?



Result: 5 / 100% Yes - 0 / 0% No

- Do you have any additional feedback on the system after the second test?



Result: 3 / 60% Yes - 2 / 40% No

Discussion of scenarios:-

The results of the tests provided an insight into the performance and usability of the log analysis system. In Test 1, we observed a positive reception, but also identified areas that require refinement.

During the first scenario, most participants found the system easy to use and navigate to filter and view specific logs, representing encouraging progress. However, the identification of specific errors revealed some variations, indicating that there is still room for optimization. The clarity and readability of the information in the filtered logs were rated favorably, suggesting a good direction in the design of the interface.

All participants agreed that the system adequately met the specific requirements of the log analysis context, which is a positive indication that we are on the right track. The additional feedback provided by the testers was a valuable resource for us to refine the system and improve the user experience.

In Test 2, after implementing improvements, we saw significant progress. The system was unanimously considered easy to use and efficient for filtering and viewing logs. The testers also noticed a notable reduction in the time needed to identify and analyze specific errors, which is an important step towards the desired efficiency.

The clear and legible presentation of the information in the filtered logs was well received, indicating that we are progressing in the right direction. All testers agreed that the system met the specific requirements of the log analysis context during this second test, signaling a positive evolution compared to Test 1.

However, it is important to note that there is still a group of testers who chose not to provide additional feedback after the second test. This may indicate that the system fully met the expectations of these participants, but it may also suggest that there is still room for further improvement.

In summary, the test results indicate promising progress, but also show that the system is in a constant process of evolution. The valuable feedback from professional testers has been essential in directing our refinement efforts. This is a significant step in the quest to deliver a high-quality solution to the developer and tester community, in line with the expectations and demands of the mobile log analysis context.

Final considerations

In this study, we demonstrated the importance of dealing with information overload in the logs, highlighting the need for a system that filters and presents only data that is relevant to the user.

Throughout the study, we explored fundamental concepts, including LogCat, techniques for identifying errors in the system log and application prototypes. These fundamentals provided the basis for the development of our system.

The results obtained during the test phase were very encouraging. Testers reported significant improvements in viewing specific logs and a notable reduction in the time spent sorting through logs. This validates the effectiveness of our system in simplifying the process of identifying errors on mobile devices.

In short, this study has shown that the approach adopted is a promising solution for dealing with the complexity of logs on mobile devices. The practical application of this system can have a significant positive impact on the efficiency and accuracy of error identification, thus benefiting the developer and tester community.

References:-

("Log analysis on mobile devices", [n.d.])
Log analysis in mobile devices ([n.d.]). In Anais do Simpósio Brasileiro de Sistemas Multimídia e Web.
(GCFGlobal, [n.d.])
Basic Computing: Operating systems for mobile devices ([n.d.]). Gcfglobal.org. Retrieved October 23, 2023, from https://edu.gcfglobal.org/pt/informatica-basica/os-sistemas-operacionais-para-dispositivos-moveis/1/
(Operating System, 2022)
What is an operating system? What is its function? (2022, August 11)). Retrieved October 23, 2023, from https://www.buscape.com.br/notebook/conteudo/o-que-e-sistema-operacional
(Mobile, 2021)
Mobile, T. (2021, June 3). Tablet operating system: key points you should know. Tec Mobile. https://www.tecmobile.com.br/blog/sistema-operacional-tablets/
(Android Developers, 2023)
Android Developers (2023). Android Developers. Retrieved October 23, 2023, from https://developer.android.com/studio/debug
(HostMídia, [n.d.])
Operating system logs - what are they and what are they for? ([n.d.]). HostMídia. Retrieved October 23, 2023, from https://www.hostmidia.com.br/blog/logs-do-sistema-operacional/
(Microsoft Learn, [n.d.])
Monitoring and diagnostic guidelines ([n.d.]). Microsoft.com. Retrieved October 23, 2023, from https://learn.microsoft.com/pt-br/azure/architecture/best-practices/monitoring
(Techopedia, [n.d.])
(Techopedia, [S.d.]). Techopedia.com. Retrieved October 23, 2023, from https://www.techopedia.com/definition/3887/prototype
(Ulrich & Eppinger, 2015)
Ulrich, K. T., & Eppinger, S. D. (2015). Product design and development. McGraw-Hill Education.
(Python Software Foundation, 2023)
Python Software Foundation. (2023). Tkinter - Python interface to Tcl/Tk - Python 3.12.0 documentation. Retrieved October 23, 2023, from https://docs.python.org/3/library/tkinter.html.