Tucandeira

Android Data Collector

Mobile App User's Guide

Tucandeira

Contents

1. Introduction

1.1 Purpose and scope

Tucandeira is a smartphone application that collects important data from the device, such as battery, memory, and general usage data. The application's goal is to help detect which device information can directly influence performance and contribute to developing solutions that enhance the user's smartphone experience. The application's name is a reference to the ant of the same name, known for its ability to collect food in large quantities and with great efficiency. This user guide will explain the process of installing and using the Tucandeira application on your device.

1.2 How/What kind of information is collected?

Tucandeira application was created with the purpose of collecting multiple relevant information about mobile device usage, such as battery data, internet connection, and running applications, both in the foreground and background, in order to analyze them more comprehensively through machine learning techniques and data mining. This data is stored in a local database within the device itself. At the end of the day, a CSV (comma-separated values) file is automatically generated and sent to a folder in Google Drive. This data collection can work uninterrupted since it is done silently, running in the background of the device. Therefore, it is ideal for the user to maintain their normal usage routine as the application will continue to collect information, and this will be more realistic for the study. Additionally, the tool aims to identify possible causes of performance issues.

1.3 Compatibility

To use the Tucandeira application, the Android device needs to be running on a minimum operating system version of 8.0 (Oreo) or higher, with at least 2GB of RAM and 150MB of free storage space (recommended). The application requires internet access to send the collected information to the project's database once a day (not mandatory).

1.4 Organization

This application was developed by the Intelligent Software (IS) group, part of the project characterized as Research, Development and Technological Innovation (RD&I), entitled Artificial Intelligence Techniques for Software Performance Analysis and Optimization (SWPERFI). Advanced data mining and AI techniques like deep learning are being used to analyze performance metrics. The project is developing innovative methods to establish correlations, verify dependencies, determine possible problems, and create a new approach using a prototype tool to optimize software performance through AI techniques. Any questions, concerns, or issues regarding the app may be resolved by contacting the SWPERFI project, and they will direct you to the IS team.

SWPERFI Website: https://swperfi.icomp.ufam.edu.br/

General Info Contact: swperfi@icomp.ufam.edu.br

Tucandeira App E-mail: swperfi-is@icomp.ufam.edu.br

2. Technical Specification

Name	Tucandeira
Description	Tucandeira is a mobile app that collects device data like battery, memory, and usage to improve smartphone performance.
Performed by	SWPERFI RD&I PROJECT
Responsible team	Intelligent Software - IS
Support	Running on a minimum operating system version of 8.0 (Oreo) or higher, with at least 2GB of RAM and 150MB of free storage space
Current version	2.00.03-alpha
Download link	Tucandeira App (http://swperfi-project.github.io/Pages-dev/TucdAndroidDataCollector-app)

3. Installation

To install the Tucandeira app on an Android device, follow the steps below:

1. Access the app file, which is stored on GitHub Pages, at the following link: Tucandeira App (http://swperfi-project.github.io/Pages-dev/TucdAndroidDataCollector-app);
2. The file is already available for public download, so it is not necessary to request permission to download it;
3. Download the application file on the Android device directly from the Google Drive app or using a web browser, once you have access to the file;
4. Make sure to allow the download of unknown files on your Android device by enabling the "Unknown sources" option in the security settings.
5. After the application file is downloaded, access the device's file management application and locate the Tucandeira application file as illustrated in Figure 1;
6. Touch the file to start the installation process;
7. The device may display a security warning message informing that you are about to install an app from an unknown source. As the app comes from a secure source, tap on "Install" to continue;
8. The installation of the application may take a few seconds, and after completion, you can open it from the device's app menu.

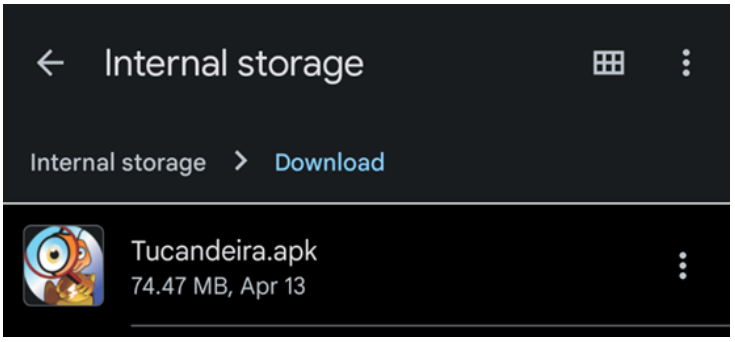

Figure 1. Example of the installer obtained through the link in this manual.

4. App overview

4.1 Splash screen

The application displays a splash screen that shows the application logo and loads some project settings.

Figure 2. Loading screen (splash screen) of the Tucandeira application.

4.2 Permissions screen

After the splash screen, the app is directed to a welcome screen, followed by a screen briefly explaining the purpose of the app and informing that it will be necessary to request certain permissions in order for the app to fulfill its purpose, as illustrated in Figure 3. These screens are displayed only on the first use after installation. Since this is an application that collects various information from the device, for security reasons, Android asks the user to grant such information in order to release the data capture.

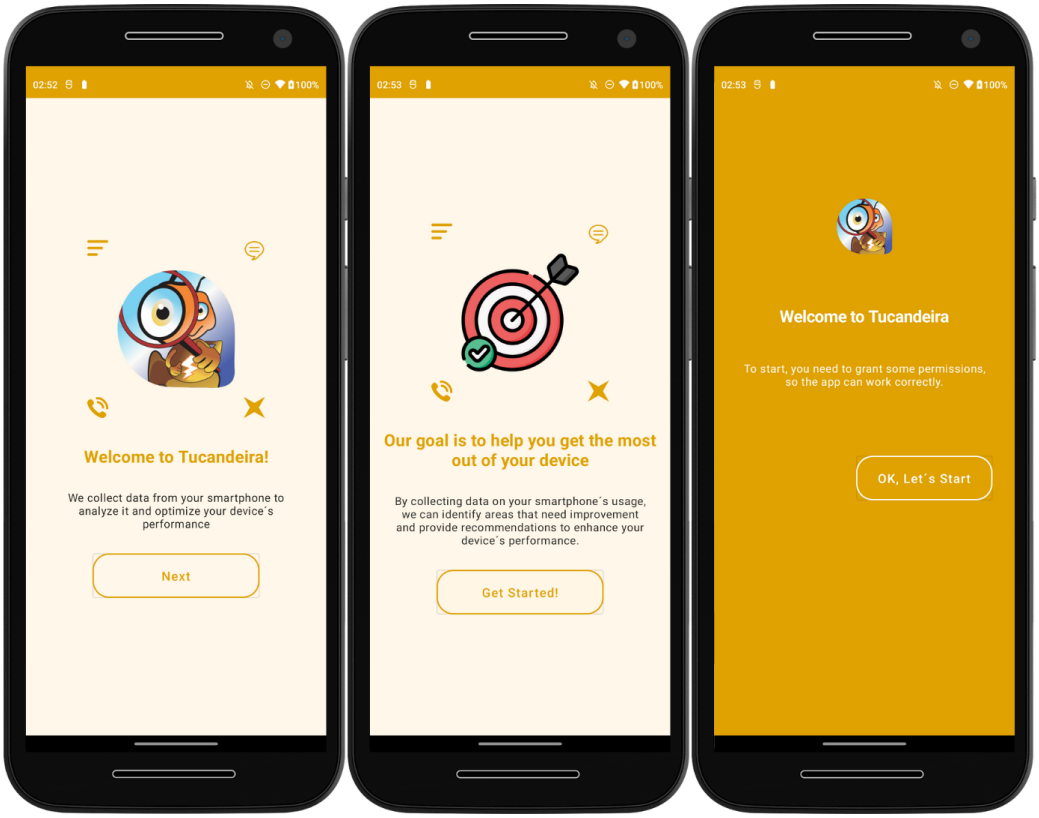

Figure 3. Initial screens welcoming the user, stating the app's purpose, and informing about permission requests.

When advancing to the next screen, some boxes or screens will appear requesting some permissions from the user. The permissions for camera and location can be granted by clicking on "During app use", as detailed in Figure 4. While the storage and background operation permissions for Tucandeira can simply be selected by clicking "Allow". The next permission, shown in Figure 5, is intended to assist in reading and writing CSV files generated by Tucandeira on newer versions of Android. Simply activate the "Access to manage all files" option and go back to the previous screen. The other permission, shown in Figure 5, requires a bit more time. The steps are to click "OK" on the message that appears on the app screen. After that, a new screen titled "Overlay other apps" will appear, with a list of apps. In this list, you need to locate the "Tucandeira" app, possibly having to scroll to the end of the list. Once you find the app name, simply click on it and accept the overlay permission. Once the process is complete, just go back until you reach the Tucandeira screen again.

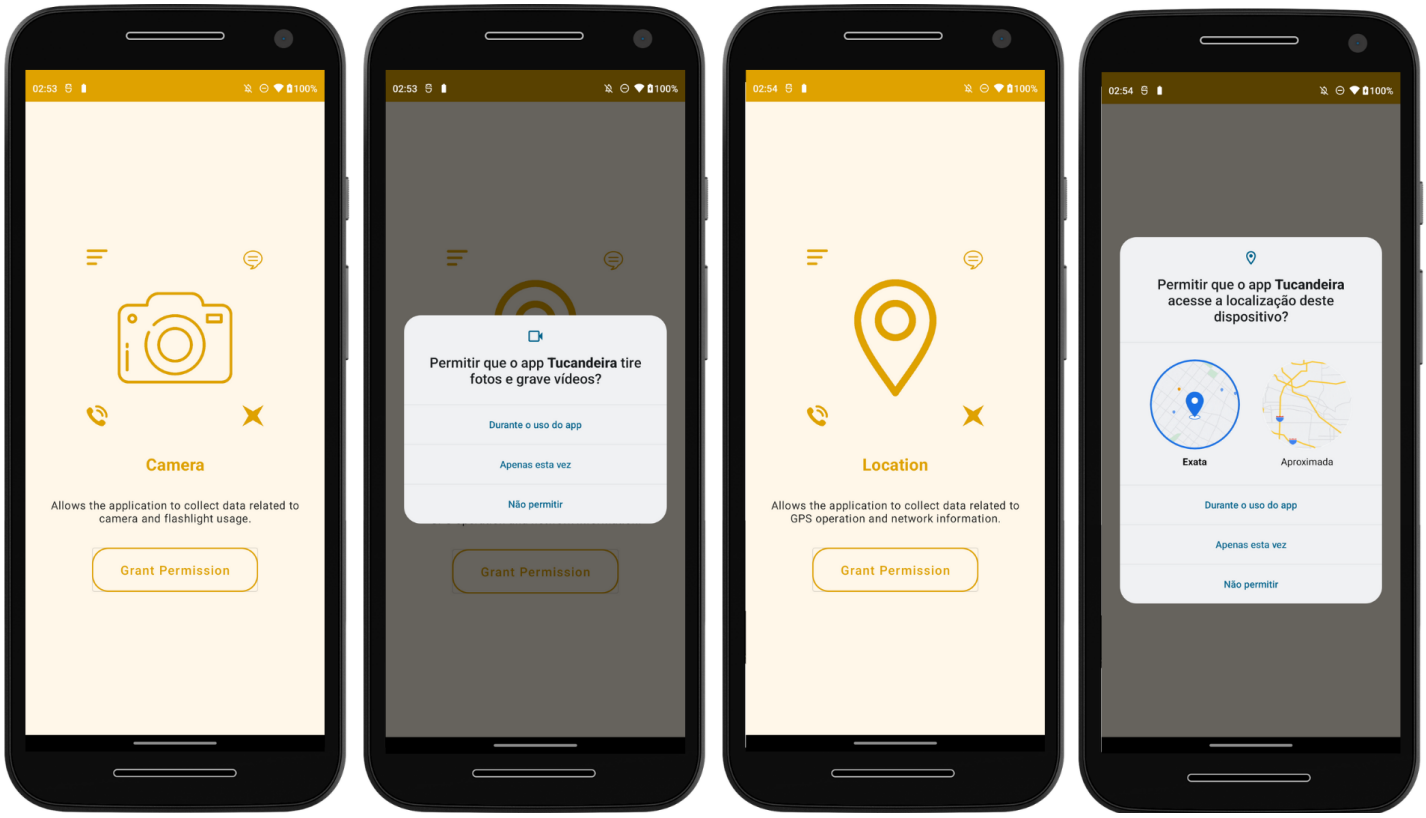

Figure 4. Required permissions for: a) Capturing photos, videos; and b) Obtaining location.

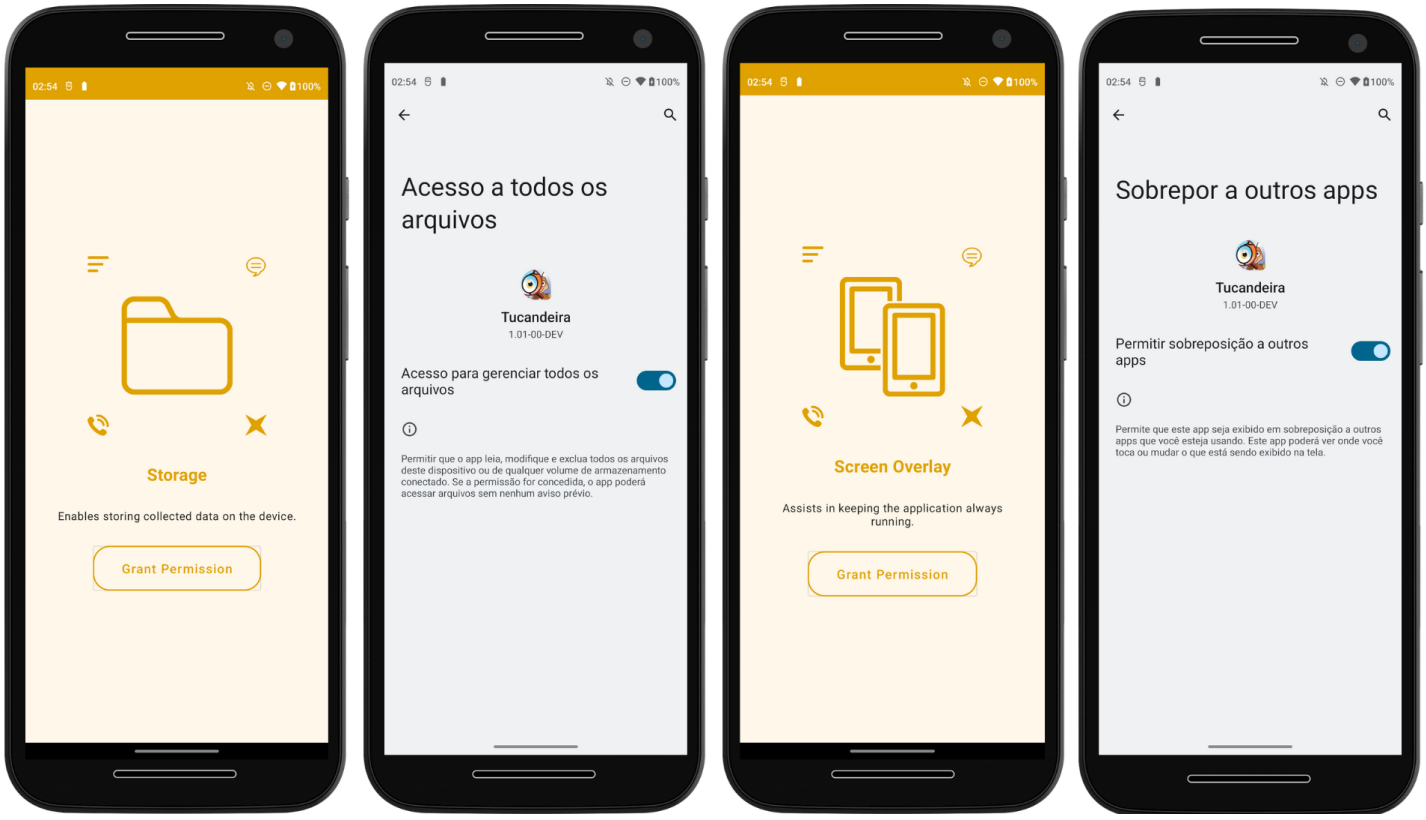

Figure 5. Required permissions for: a) Accessing device files; and b) Running Tucandeira app in the background.

Finally, a new tab of "Required permission" will appear, where you just need to click "Ok" and "Allow access to usage" before returning to the app. Once the process is completed, the user can click on the "Start App" button at the end of the screen to start the collect service, as shown in Figure 6. When closing the permissions screen, the visual part of the application is closed. However, the data collection system proceeds normally after a few seconds.

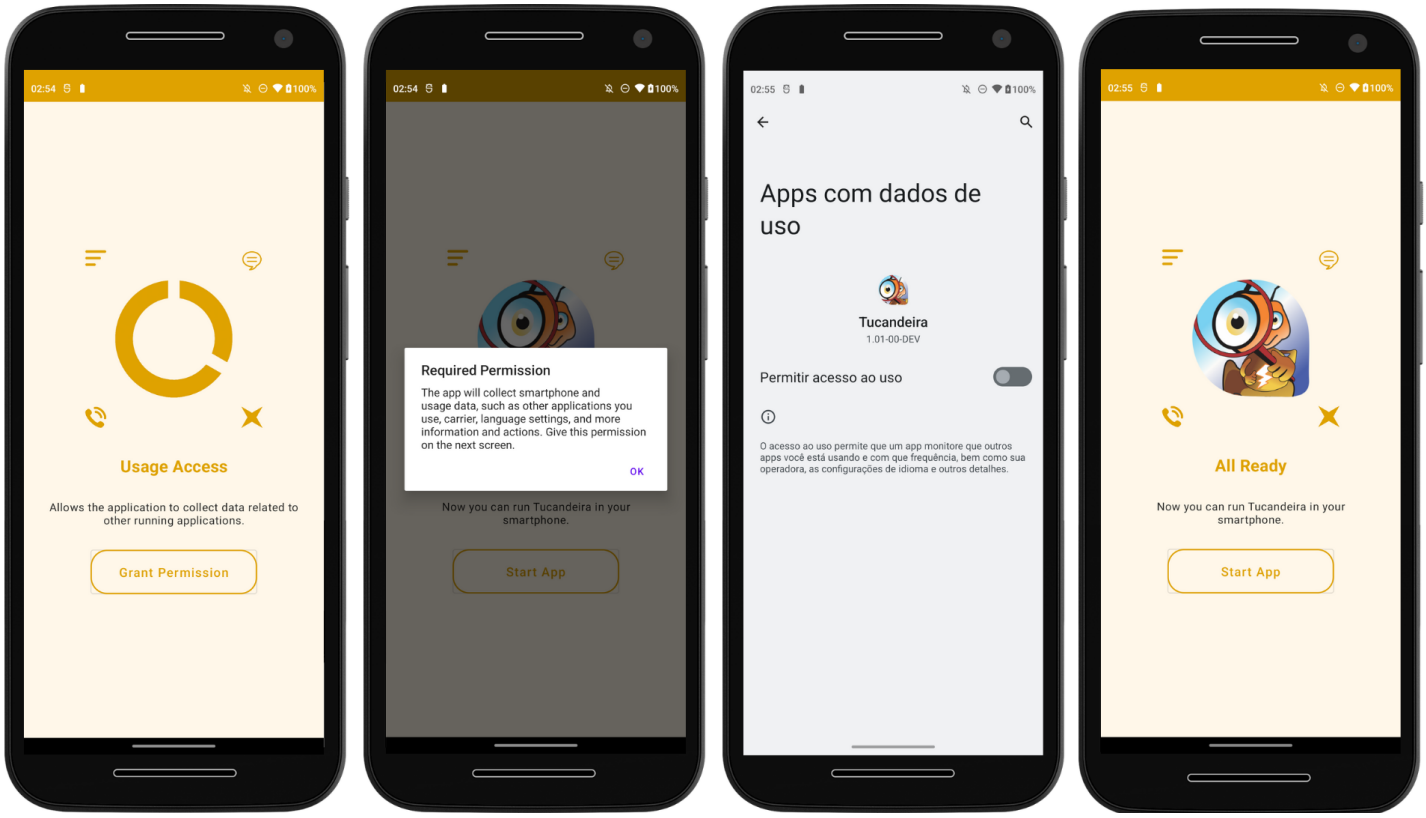

Figure 6. On the first screen, permission to manage files for reading and writing data in ROM memory. On the fourth screen, it is possible to start the app.

5. Description of application’s functions

Tucandeira provides several main functionalities that focus on collecting and storing data from mobile devices. This data is saved locally on the device itself, in the folder: "./Secondary storage/Documents/SWPerfI/Tucandeira/Logs/", and can be accessed through the file manager. The directory contains three folders:

- **Dynamic:** Refers to resources that change frequently, such as Bluetooth status, screen-on time, foreground application, and timestamp;
- **Static:** Refers to records that remain constant or vary very little, such as the manufacturer's name, kernel version, and device architecture;;
- **Apps:** Includes a csv file with the tag "background_apps", which contains records of background collection, such as the timestamp and the list of running apps. In addition, there is another csv file with the tag "app_list", which contains the list of installed apps, including the app ID, package name, app type, and SDK version.

One of the main functionalities of Tucandeira is real-time data collection, updated every second in CSV spreadsheets. These data include information such as background running apps. Additionally, other important data, known as dynamic data, are collected throughout the day and stored in a SQLite database every second. Every minute, the CSV file corresponding to the day of collection is updated with the information from dynamic collection. At the end of the day, this data is sent to Google Drive, while a copy of the CSV remains stored on the device.

6. Additional information about the collected data

More than 80 pieces of information are collected from various sources, including foreground and background apps, audio, battery, CPU, device, memory, connection (including internet, mobile data, Wi-Fi, and carrier), screen, general settings (such as Bluetooth, GPS, NFC, flashlight, and fingerprint), and calendar. Different libraries and native Android tools are used to collect static and dynamic data for each type of information. All data is stored efficiently to save space.

6.1 Application section

In the application section, we obtain information related to the application that is being run on the screen, which we call the foreground application, as well as all the applications and services running in the background. To collect information on the foreground applications, we use an external library called AppChecker, which has the ability to detect which application is currently running. As for the background processes, we use the PackageManager, an Android native application or package manager.

Classification	Feature Name	Type	Value	Description
App List	app_id	integer	Positive integer	Foreground app identifier
	package_name	char sequence	char sequence	Application package name
	app_name	char sequence	char sequence	Application Name
	app_type	integer	0	User application
			1	System application
	is_foreground	integer	0	The app runs in the background
			1	The app runs in the foreground
	app_sdk	integer	Positive integer	SDK version for the app

6.1.1 Update system

The Tucandeira has an update system that notifies the user whenever a new version of the application is available. When opening the app, if an update is available, the user will receive a notification on the app's home screen, informing them that a new version is available for download. The user can choose to download and install immediately or opt to do so later. This notification system allows users to always have access to the latest version of the application, with the latest improvements and fixes implemented by the development team.

6.2 Audio section

Using the Android AudioManager library, we collect data such as volume, ring mode, and whether any sound is being played by the device.

Classification	Feature Name	Type	Value	Description
Sound Features	ring_mode	integer	0	Ringing mode will be silent
			1	Ring mode will be silent and vibrate
			2	Ringing mode can be audible and can vibrate
	sound_level	integer	[0,...,15]	Current volume index
	playback_status	integer	0	No music track is active
			1	Music tracks are active

6.3 CPU section

To collect data on CPU, we need to access information available within the Android system, where we can find the CPU temperature, the number of cores, and the usage of each of them.

Classification	Feature Name	Type	Value	Description
CPU info	cpu_usage	float	Positive float	CPU usage percentage
	cpu_temperature	float	Positive float	CPU temperature in °C
	core_numbers	integer	Positive integer	Number of CPU cores
	up_time	double	Positive double	Time in seconds since last reset
	sleep_time	double	Positive double	Time in seconds since the last reboot in which the device was turned

				off
	frequency_core	float	Positive float	Frequency of use of each CPU core

6.4 Battery section

Regarding the battery, we collect data ranging from static factory values, such as battery technology, to dynamic data such as level, health, charging mode, temperature, current, voltage, power, and capacity. These data are collected with the help of BatteryManager.

Classification	Feature Name	Type	Value	Description
Battery Info	battery_level	integer	[0,...,100]	Battery level
	battery_health	integer	0	Cold
			1	Discharged
			2	Good
			3	Overheat
			4	Over voltage
			-1	Unknown
	battery_charging_status	integer	1	Charging
			2	Discharging
			3	Not charging
			4	Full charged
			-1	Unknown
	battery_connection _status	integer	0	Battery
			1	Charging
			2	USB
	battery_temperature	real	Float	Battery temperature in Celsius
	battery_current	real	Float	Battery current in mA
	battery_voltage	real	Float	Battery voltage in Volts

	battery_power	real	Float	Instantaneous battery power in watts/sec
	saving_mode	integer	0	Power saving enabled
			1	Power saving off

6.5 Device section

Regarding the device, we have a unique identifier generated for each smartphone, which should be useful in the data analysis stages, static information such as brand, model, product, version, and Android architecture. We also add data such as root mode, USB debugging, and permission to install outside of Google Play.

Classification	Feature Name	Type	Value	Description
Device Info	device_id	char sequence	char sequence	Device unique identifier
	date	char sequence	char sequence	Current date in MM/DD/YYYY format
	country_code	char sequence	char sequence	Country code (abbreviation)
	language	char sequence	char sequence	Device language
	continent	char sequence	char sequence	Returns the continent code
	local_time	char sequence	char sequence	Represents the international device reference time standard
	time_zone	char sequence	char sequence	Time zone where the device is located
	device_model	char sequence	char sequence	Device model
	device_manufacturer	char sequence	char sequence	Device manufacturer name
	device_brand	char sequence	char sequence	Device brand name
	device_product	char sequence	char sequence	Represents the product name of the device
	device_sku	char sequence	char sequence	Unique code assigned to each device model by the manufacturer
	android_version	real	Float	Android version number
	android_api	integer	Positive Integer	API version number
	device_arch	char sequence	char sequence	Device architecture
	device_rooted	integer	Positive Integer	Indicates whether the device has

				super user privileges
	kernel_version	char sequence	char sequence	Device kernel version
	usb_debug	integer	Positive Integer	Indicates whether debug mode is enabled
	unknown_src	integer	Positive Integer	Indicates whether or not the sources for installing apps are verified by Google
	battery_presence	char sequence	char sequence	Indicates if there is a battery connected
	battery_scale	integer	Positive Integer	Refers to the device's maximum battery capacity
	battery_tech	char sequence	char sequence	Technology used in the battery
	core_numbers	core_numbers	Positive Integer	Device core number
	battery_capacity	real	Float	Battery capacity
	ram_capacity	integer	Positive Integer	RAM capacity (in bytes)
	rom_capacity	integer	Positive Integer	ROM capacity (in bytes)
	gpu_version	real	Float	Device GPU version

6.6 Connection section

Regarding the connection, we will divide it into four segments. The first, more generic, checks if the device has any type of internet connection, using the ConnectivityManager. Then, we check if this connection is via mobile data. If so, we will verify if the connection is 3G, 4G or 5G, as well as the possible use of roaming and its upload and download rates, with the help of NetworkInfo. With regards to WiFi, and with the help of WiFiManager, we collect the signal strength, speed, upload and download rates, as well as detect if the router or access point is active for other devices. Last but not least, we have carrier data, where we collect its name, and codes such as MCC and MNC, which also identify it better, with the help of TelephonyManager.

Classification	Feature Name	Type	Value	Description

Connection Info	network_mode	integer	0	No connection
			1	there is a connection
	wifi_status	integer	0	wifi disconnected
			1	wifi connected
	wifi_intensity	real	Float	Wi-Fi strength in decibels
	wifi_speed	real	Float	Wi-Fi link speed in Mbps
	wifi_ap	integer	0	WiFi router off
			1	WiFi Router Enabled
	wifi_rx	real	Float	Current receive link speed in Mbps
	wifi_tx	real	Float	Current transmission link speed in Mbps
	mobile_mode	char sequence	“2G”, “3G”, “4G”, “5G”	mobile connection type
			“0”	mobile network off
			“-1”	Unknown mobile network mode
	mobile_status	integer	0	Unknown mobile network mode
			1	connected mobile network
	mobile_roaming	integer	0	Disconnected mobile roaming
			1	connected mobile roaming
	mobile_rx	real	Float	Current receive link speed in Mbps
	mobile_tx	real	Float	Current transmission link speed in Mbps
	network operator	char sequence	Mobile network operator name	Mobile network operator name
	mcc	integer	Positive integer	Country code on SIM card
	mnc	integer	Positive integer	SIM card mobile network code

6.7 Memory section

Regarding memory, whether it's RAM or ROM, we need to access the ActivityManager again or the Android memory manager, respectively, to find out how much memory we have free or in use on the phone.

Classification	Feature Name	Type	Value	Description
Memory Info	ram_usage	real	Float	RAM usage in bytes
	ram_free	real	Float	RAM free in bytes
	rom_usage	real	Float	ROM usage in bytes
	rom_free	real	Float	RAM free in bytes

6.8 Screen of the device section

Regarding the device screen, we collect several general information from Android and the DisplayManager library, where we collect the brightness level of the phone, whether it is in automatic or manual mode, and how long the smartphone stays on without pauses.

Classification	Feature Name	Type	Value	Description
Screen Info	screen_status	integer	0	screen off
			1	screen on
	bright_level	integer	[0,...,255]	screen brightness level
	orientation	integer	0	screen in landscape orientation
			1	Screen in portrait orientation
	bright_mode	integer	0	Manual screen brightness mode
			1	Automatic screen brightness mode
	screen_on_time	integer	Positive integer	Time in seconds that the screen is on uninterruptedly

6.9 General section

In the general settings, we have information about Bluetooth, GPS and its use by other apps, NFC, flashlight, screen orientation, and use of fingerprint. For this, we use various native tools of Android, such as BluetoothAdapter, LocationManager, PowerManager, Nfc, Camera, and BiometricManager. Finally, using the Calendar and the Locale library, it is possible to collect data about the device's location and also about the time to assist in the analysis of the data.

Classification	Feature Name	Type	Value	Description
General Info	device_id	char sequence	char sequence	Device unique identifier
	date_time	char sequence	char sequence	Date and time of data collection
	flashlight	integer	0	flashlight off
			1	flashlight on
	airplane_mode	integer	0	airplane mode off
			1	airplane mode on
	fingerprint	integer	0	digital printing available
			1	digital printing unavailable
	foreground_app	integer	Positive integer	Foreground app identifier
	bluetooth_status	integer	10	Bluetooth off
			13	Bluetooth turning off
			11	Bluetooth turning on
			0	Bluetooth disconnected
			3	Bluetooth disconnecting
			2	Bluetooth connected
			1	Bluetooth connecting
			12	Bluetooth on
			-1	Unavailable

	bluetooth_name	char sequence	char sequence	The name of the connected bluetooth device
	nfc	integer	0	NFC off
			1	NFC on
			-1	Unknown
	gps_status	integer	0	GPS off
			1	GPS on
	gps_activity	integer	0	GPS not used
			1	GPS in usage