Federal University of Amazonas - UFAM

Institute of Computing - IComp

SWPerfl Project

Intelligent Hardware - IH

Suindara App – User Guide v1.2.2

Author: Pedro Victor Dos Santos Matias

Project Sponsor: Motorola Mobility

Manaus - AM, March 23, 2025

Table of Contents

Revision History

Check the latest release from the Releases Section.

Date	Name	Comment of changes
07 Mar 2025	Pedro Matias, pvsm@icomp.ufam.edu.br	**Initial version with XGboost**
18 Mar 2025	Pedro Matias, pvsm@icomp.ufam.edu.br	**Added a new pre-trained model with log data, Catboost**
23 Mar 2025	Pedro Matias, pvsm@icomp.ufam.edu.br	**Some bug fixes**

Introduction

Purpose and Scope

Suindara App is a desktop application developed with Python and PySide6 to parse and analyze Android bug reports and integrate predictive models for call drop analysis. The tool is part of the SWPERFI project and is designed for technical teams working with mobile networks and quality analysis.

Organization

This application was developed by SWPERFI project is a collaborative effort between the Federal University of Amazonas (UFAM) and an industry partner focused on applying advanced AI techniques to enhance software performance. As part of this project, the Intelligent Hardware (IH) subgroup investigates methods to identify, predict, and minimize call drops in mobile networks, with a particular emphasis on IMS (IP Multimedia Subsystem) calls like VoLTE.

Any questions, concerns, or issues regarding the app may be resolved by contacting the SWPERFI project, and they will direct you to the IH team.

Contact: swperfi@icomp.ufam.edu.br

Technical Specifications

Full name	Suindara App
Short name	Suindara App - Call Drop Log Parser & Predictor
Description	Suindara App is a desktop application built with Python and PySide6 for parsing and analyzing Android bug reports, integrating predictive models for call drop analysis. It includes a complete pipeline for parsing logs and predicting call drop occurrences, featuring SHAP explanations for model interpretability.
Performed by	SWPERFI
Responsible team	Intelligent Hardware (*Modem subgroup*) | IH-Modem
Support	Windows 10/11
Technology	Python, PySide6
Libraries	pandas, numpy, xgboost, catboost, shap, matplotlib
Current version	1.2.2
Download link	Suindara App (swperfi-project.github.io)

Installation

1. Check for the latest version in the Suindara App (swperfi-project.github.io) or GitHub releases page.
2. Download the executable via Zenodo.
3. Run SuindaraApp.exe directly. No extraction or additional setup is needed.

Note: Suindara App is distributed as a standalone executable, Python installation is not required.

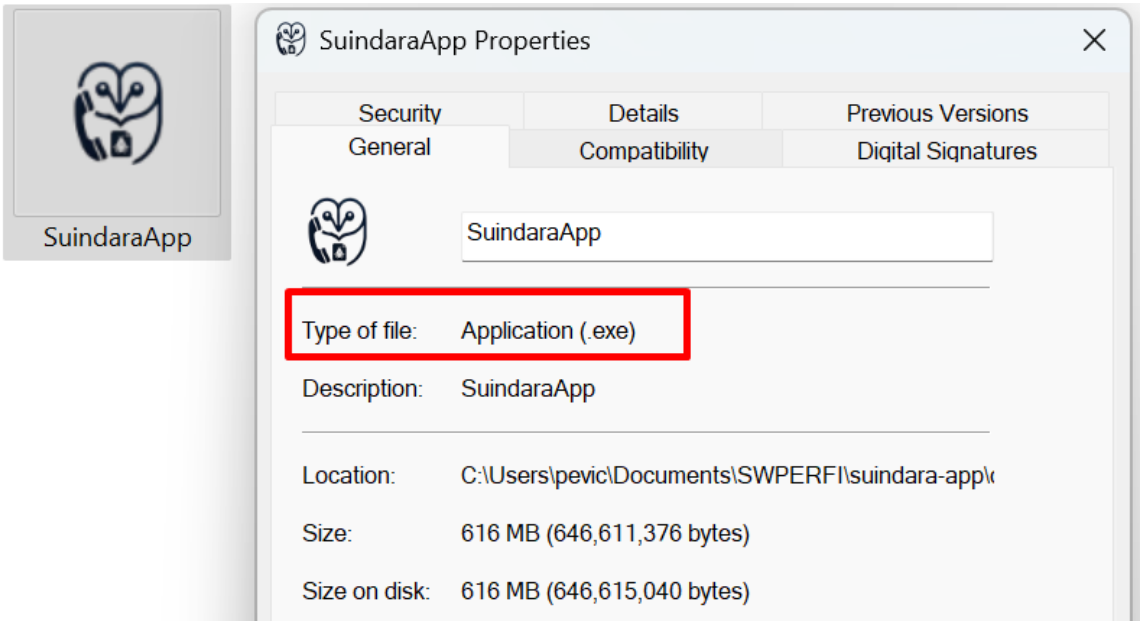


Figure 1: Example of executable file obtained through the link in this manual.

Application Overview

Splash Screen

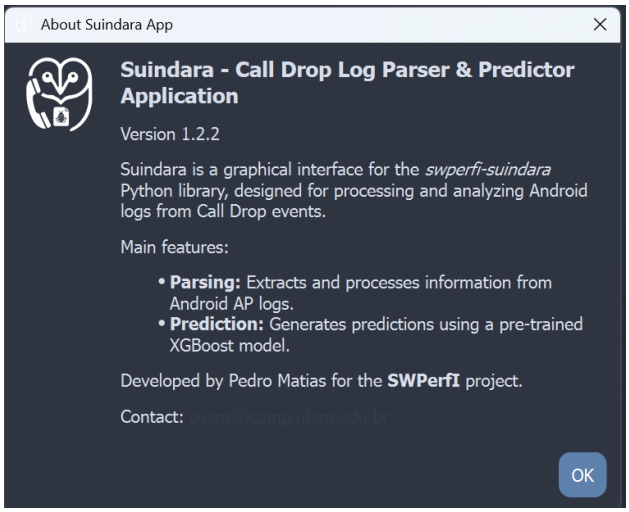
The application displays a splash screen that shows the application logo while loading the libs and the pre-trained model.

Figure 2: Loading screen (splash screen).

About Dialog

The application displays a dialog box with information about the tool.

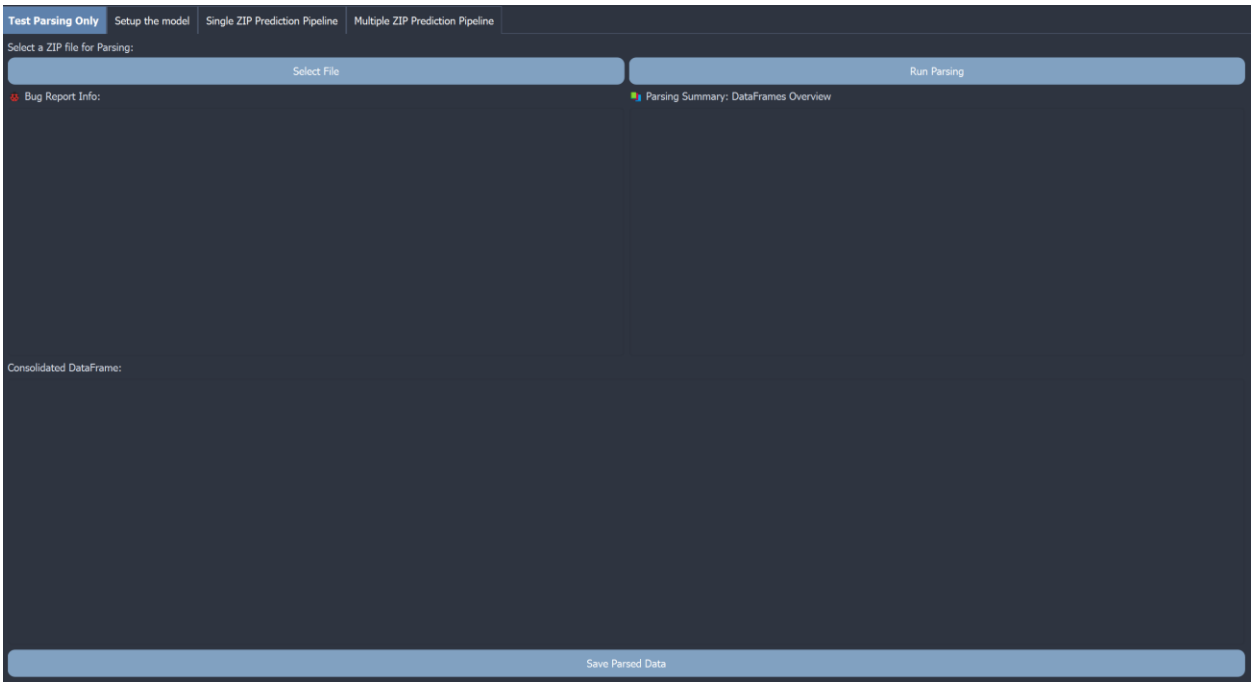
.



Main screen

The main window provides the following tabs:

- **Test Parsing Only**



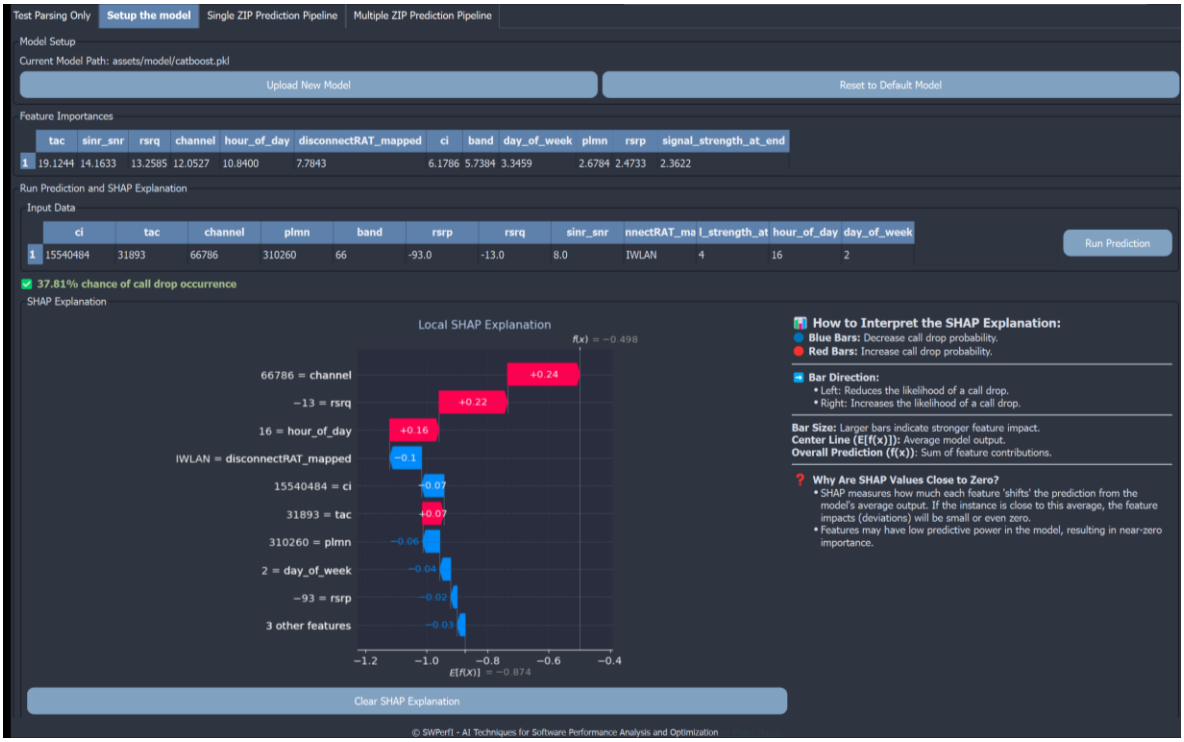
This tab is designed for users who want to exclusively test the **parsing functionality** without triggering the prediction pipeline. It is ideal for inspecting the extracted log data from an Android bug report (.zip file) before proceeding to any predictive analysis.

Key Elements:

- **Select File**: Choose a ZIP file containing Android bug reports.
- **Run Parsing**: Executes the parsing process, extracting metadata and call-related logs.
- **Bug Report Info**: Displays extracted metadata from the bug report (e.g., device info, log summary).
- **Parsing Summary**: Presents an overview of generated DataFrames during parsing.
- **Consolidated DataFrame**: Shows a unified DataFrame combining the extracted information.
- **Save Parsed Data**: Exports the consolidated data to a CSV file for further analysis.

This tab is particularly useful for validating whether the bug report contains the expected logs before running any call drop predictions.

- **Setup the model**



This tab acts as a **model configuration and analysis playground** where users can:

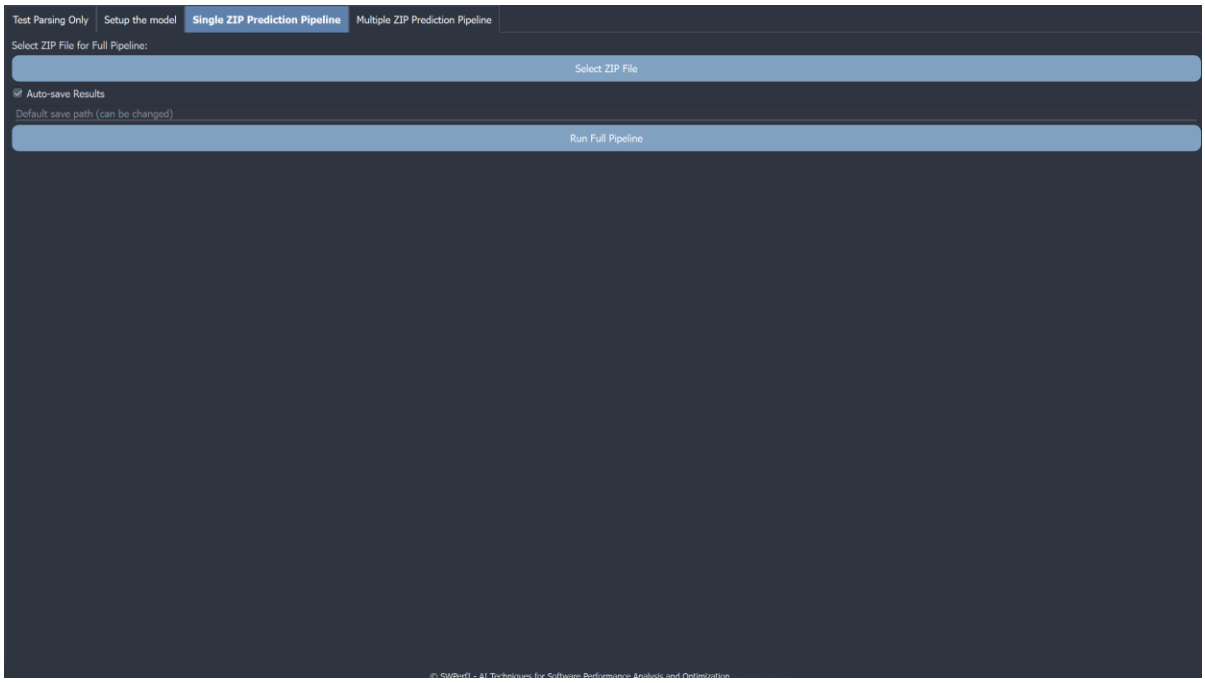
Key Functionalities:

- **Model Setup:**
  - Shows the current model path and allows users to upload a custom model file (.pkl format).
  - Option to reset to the default model distributed with Suindara App.
- **Feature Importances:**
  - Displays the importance ranking of features used by the predictive model.
  - Helps users understand which features contribute most to call drop predictions.
- **Input Data Section:**
  - Editable table where users can modify input values for features.
  - Designed to simulate different network conditions and analyze their impact.
- **Prediction and SHAP Explanation:**
  - On clicking **Run Prediction**, the model provides a probability of call drop occurrence.
  - Generates a **Local SHAP Explanation** via a waterfall chart, illustrating how each feature contributed to the predicted outcome.
- **SHAP Interpretation Guide:**

- The right panel explains how to interpret SHAP plots, including color coding and bar directions.
- Highlights the impact magnitude of each feature on the prediction.

This tab is ideal for users aiming to perform local explainability analysis, test different input scenarios, and better understand model behavior on specific cases.

- **Single ZIP Prediction Pipeline**



This tab allows users to process a **single ZIP file** containing Android bug reports through a complete pipeline, including parsing and prediction.

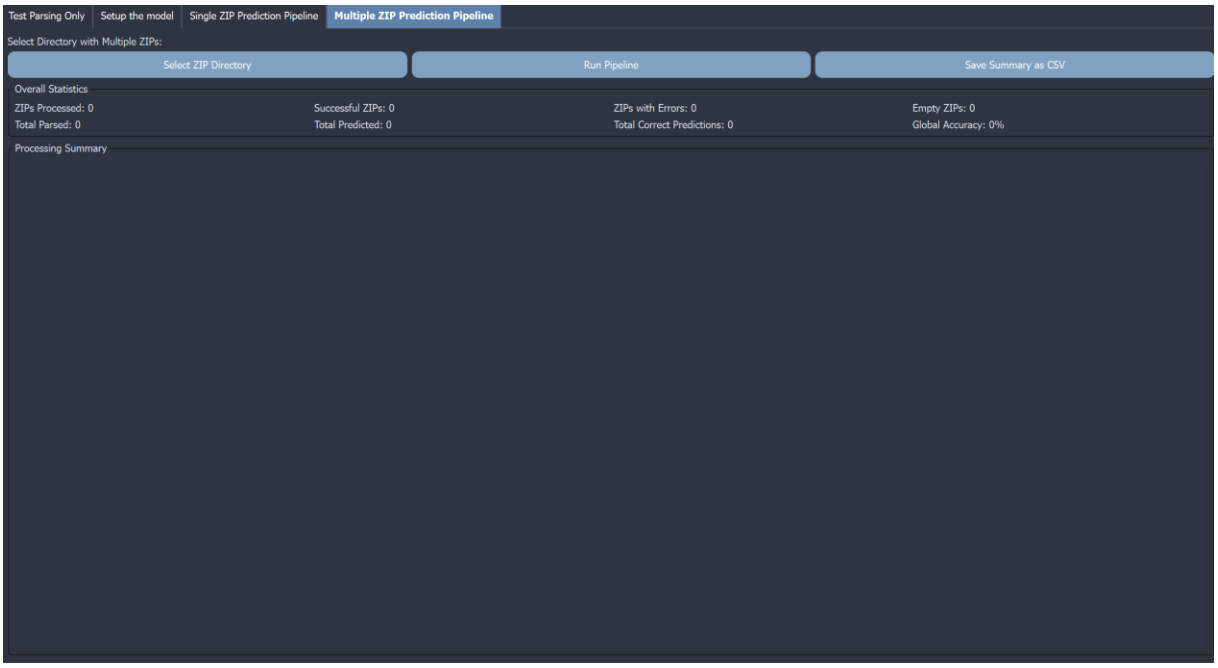
Key Functionalities:

- **ZIP Selection:**
  - Choose a single .zip file for processing.
  - The selected file is displayed at the top.
- **Pipeline Summary Information:**
  - Displays a summary after processing:
    - ZIP filename
    - Status (e.g., Success)
    - Number of Parsed Calls
    - Number of Predicted Calls
    - Correct Predictions

        - Local Accuracy (percentage)
- **Parsed DataFrame Section:**
    - Displays the parsed log data extracted from the ZIP file.
    - Users can view or export this table.
- **Full Prediction DataFrame Section:**
    - Shows the prediction output, including features used, predicted call drop probability, and binary classification.
    - Users can view or save this table for further analysis.

This tab does not include SHAP explanations.
It focuses on providing fast, end-to-end processing for a single file, combining both parsing and model prediction.

- **Multiple ZIP Prediction Pipeline**



This tab enables users to process multiple ZIP files from a directory in a batch mode, automating the parsing and prediction pipeline across several bug reports.
Key Functionalities:

- **Select ZIP Directory:**
    - Choose a folder containing multiple ZIP files with Android bug reports.
- **Run Pipeline:**
    - Automates the entire pipeline (parsing + prediction) for all ZIPs in the selected directory.
- **Overall Statistics Panel:**
    - Displays a global overview:
        - Number of ZIPs processed.

        - Count of successful ZIPs, ZIPs with errors, and empty ZIPs.
        - Aggregated metrics such as total parsed calls, total predictions, total correct predictions, and global accuracy.
- **Processing Summary Table:**
    - Shows individual results for each ZIP file:
        - Status (e.g., Success, Empty after processing)
        - Parsed Calls
        - Predicted Calls
        - Correct Predictions
        - Local Accuracy per ZIP
- **CSV Export:**
    - Allows exporting the summary as a CSV file via the **Save Summary as CSV** button.

Parsed and predicted tables are not displayed individually here.
All outputs are automatically saved in the original ZIP directory for convenience.

Description of application’s functions

This tool serves as a preliminary platform to evaluate the performance of basic machine learning models trained to classify call drops. These models generate predictions indicating whether a call was dropped or not, along with associated probabilities. By processing Bug2Go logs, extracting relevant parameters, and applying the models, the tool aims to provide classification for the call events parsed, and a user-friendly interface to visualize and interpret these results, aiding in the evaluation of model performance.

Major features:

- Complete Pipeline: Automates parsing and prediction in a single flow.
- Batch Mode: Supports directories with multiple ZIPs.
- Overall Statistics Panel: Consolidates global metrics for batch processing.
- SHAP Explanation (Setup Model tab only).

Dataset and Features

Parsed Data

The parsing process extracts various fields from Android bug report logs (.zip with AP logs), generating structured information about the call session, device parameters, and network metrics.

Parsed Columns:

FIELD	DESCRIPTION	TYPE	TYPICAL RANGE / VALUES
CI	Cell Identity of the serving cell	Integer / Categorical	e.g., 196887827, 15540484
TAC	Tracking Area Code of the LTE network area	Integer / Categorical	e.g., 10007, 31893
CHANNEL	Channel number used by the serving cell	Integer / Categorical	e.g., 9820, 66786
PLMN	Public Land Mobile Network identifier (MCC + MNC)	String	e.g., 310410, 724050
BAND	Frequency band used by the serving cell	Integer / Categorical	e.g., 30, 66
RSRP	Reference Signal Received Power	Float	-140 dBm to -44 dBm
RSRQ	Reference Signal Received Quality	Float	-20 dB to -3 dB
SINR_SNR	Signal-to-Interference-plus-Noise Ratio	Float	0 to 30
DISCONNECTRAT_MAPPED	Radio Access Technology at the time of call disconnection	String	e.g., LTE, IWLAN, NR
SIGNAL_STRENGTH_AT_END	Signal strength category at call termination	Integer	0 (Poor) to 5 (Excellent)
HOUR_OF_DAY	Hour of the day when the call was disconnected	Integer	0 to 23
DAY_OF_WEEK	Day of the week (0 = Monday, ..., 6 = Sunday)	Integer	0 to 6

These columns are generated by parsing the logs present in the Android .zip bug report. They represent the contextual and radio conditions under which the call took place.

Prediction Inputs

The prediction pipeline (CatBoost or XGBoost) uses a **subset of the parsed data** as input features. These columns are preprocessed and fed into the model to predict the likelihood of a call drop.

Used Features for Prediction:

FEATURE	DESCRIPTION
CI	Cell Identity
TAC	LTE area identifier
CHANNEL	Serving cell channel number
PLMN	Mobile network identifier (MCC+MNC)
BAND	Frequency band
RSRP	Signal power
RSRQ	Signal quality
SINR_SNR	Signal-to-noise ratio
DISCONNECTRAT_MAPPED	RAT type at call drop
SIGNAL_STRENGTH_AT_END	Signal strength category at the end of the call
HOUR_OF_DAY	Call termination hour
DAY_OF_WEEK	Day of the week

> ⚠ *Note: Both CatBoost and XGBoost models handle categorical features internally. For CatBoost, categorical variables are passed natively. For XGBoost, label encoding is applied before model inference.*

Model Output:

After inference, the model generates the following additional columns:

OUTPUT	DESCRIPTION	TYPE	TYPICAL VALUES
DROP_PROBABILITY	Probability of call drop occurrence (model output)	Float	0.0 to 1.0
PREDICTED_CALL_DROP	Model binary classification result (True = drop, False = no drop)	Boolean	True / False

Notes

- Some features are derived or mapped during the parsing phase, such as *disconnectRAT_mapped* (based on RAT codes from Android Telecom framework).
- Null or missing values are handled internally by the parsing and prediction pipeline.
- The app is designed to be compatible with Android bug reports from Motorola devices but may also work with reports from other manufacturers, depending on log structure.
- Recommended to use ZIPs generated from Motorola devices.
- The predictive model is optimized for call drop events in IMS/VoLTE networks.

Project Links Info

- SWPERFI Project: https://swperfi.icomp.ufam.edu.br
- Desktop App Repo: https://github.com/swperfi-project/suindara-app-desktop-ext
- Python SDK Repo: https://github.com/swperfi-project/swperfi-suindara-lib
- UI Code Repo: https://github.com/swperfi-project/suindara-app-desktop-dev