

# PAUSIFY

15/06/2025

---

Lucas Mazzolim  
Curitiba, 2025



## SUMMARY

## 1. INTRODUCTION

Pausify is a mobile application that helps users decrease their screen time. It's done by setting goals, locking in user money and giving it away if the user doesn't accomplish their goals, returning it to the user otherwise. The users choose which app they want to track, how much money they want to block, for how long the track is and for who the money goes to if they don't accomplish their goals. The focus is a simple yet personalized experience.

### 1.1 Purpose

We want to help people that are addicted to their phones, making them improve time management, productivity and life overall with a financial motivation in a gamified approach. Recent studies found that excessive screen time affects sleep quality, decreases social activity and can harm the brain<sup>1</sup>, therefore this issue is very serious and, mainly in our hyper connected society, it's a problem worth fighting.

---

1

<https://longevity.stanford.edu/lifestyle/2024/05/30/what-excessive-screen-time-does-to-the-adult-brain/>

## 2. LITERATURE SURVEY

There are apps that have a similar approach as the one of Pausify, however, none of them uses the system of financial loss to help the user reduce their screen time and none of them have the donation system as well. Notable apps that have similar proposals or solve close problems are:

- I. **Forest:** App used as a focus tool. In the app you set a time you want to focus and, while the time doesn't end, you are not allowed to close the software. If you complete your challenge you are rewarded with coins that, when accumulated, can be used to plant a real tree.
- II. **Notion:** This software is used to create to-do lists, track project progress and help companies boost their productivity with a diverse portfolio of project-management tools. It's one of the most famous apps on productivity and project management.

These two classes of products, which are used, respectively, to focus for a certain time and to track progress/ manage time better, are the two most proxime apps to Pausify. Nonetheless, these two types of software don't share the money-losing system and app-based time tracking, so Pausify creates a differentiator from its competitors.

### 3. REQUIREMENTS

#### 3.1 Functional Requirements

##### I. Screen Time Goal Management

1. The system must allow the user to define multiple screen time goals for different periods, chosen by the user.
2. For each goal, the user must be able to:
  - a. Specify the applications to be monitored.
  - b. Define the maximum usage time limit for the selected applications.
  - c. Determine the monetary value to be blocked associated with that goal.
  - d. Choose the charity organization to which the money will be sent in case of failure, or a friend.

##### II. Usage Time Monitoring and Notification

1. The system must actively monitor the usage time of the applications selected by the user.
2. The system must display a remaining time counter that the user still has before exceeding their goal.
3. The system must notify the user (via in-app push notifications) about the remaining time for their goal and the risk of losing the blocked amount.
4. The system must use the Android system API for monitoring and must be allowed to run in the background.

##### III. Payment/Donation Processing and Notification

1. The system must facilitate the secure blocking of the monetary value defined by the user.
2. The system must send a confirmation email to the user whenever a payment (money blocking) is successfully made.
3. If the user exceeds their goal, the system must automatically process the transfer of the blocked amount to the selected charity (or friend).
4. The system must send a confirmation email to the user informing that the money was sent to charity (or friend) due to not meeting the goal.

5. If the user meets their goal, the system must automatically process the return of the blocked amount to the user's account.
6. The system must send a confirmation email to the user informing that the money was successfully returned for meeting the goal.

#### **IV. Application Selection and Blocking:**

1. The system must list all applications installed on the user's device.
2. The system must allow the user to freely select and deselect which applications they wish to include in their goal monitoring.

#### **V. User Profile Management:**

1. The system must allow the user to view and edit their profile information, including:
  - a. User name.
  - b. Email address.
  - c. PIX code (PIX will be the only method for money return ).
2. The user can change their name, PIX code, and email whenever they wish

### **3.2 Non-Functional Requirements**

#### **1. Performance**

- a. Responsiveness: The application should load quickly (e.g., within 2-3 seconds on average internet/device conditions). UI transitions should be smooth and lag-free.
- b. Processing Speed: Time tracking and goal evaluation should happen in near real-time, with minimal delay in detecting app usage.
- c. Background Processing Efficiency: Monitoring apps in the background should consume minimal battery and data.

#### **2. Reliability/Availability**

- a. Uptime: The backend API and database should be highly available.
- b. Data Integrity: All user data (goals, blocked money status, usage time) must be accurate and consistent, even in case of system failures. Transactions (money blocking/return/donation) must be atomic.

- c. Error Handling: The system should gracefully handle errors (e.g., network issues, API failures) and inform the user appropriately without crashing.

### **3. Security**

- a. Data Encryption: All sensitive user data (PIX codes, financial transaction details, personal information) must be encrypted at rest and in transit.
- b. Authentication & Authorization: User authentication (login) must be secure, protecting against unauthorized access. Only authenticated and authorized users should be able to access their own data.
- c. Financial Transaction Security: Integration with financial APIs must adhere to the highest security standards.
- d. Protection against Tampering: The app should have mechanisms to detect and prevent users from tampering with local time tracking or bypassing monitoring.

### **4. Usability (User Experience - UX)**

- a. Intuitiveness: The app's interface must be easy to understand and navigate, especially for setting goals and tracking progress.
- b. Learnability: New users should be able to quickly grasp how to use the core features without extensive tutorials.
- c. Accessibility: Consider features for users with disabilities (e.g., screen reader compatibility, adjustable font sizes).
- d. Consistency: The UI/UX should be consistent across all screens and flows.
- e. Feedback: The system should provide clear and immediate feedback to user actions (e.g., successful goal setting, money blocked).

### **5. Maintainability**

- a. Code Quality: The code should be well-structured, documented, and easy for other developers to understand and modify.
- b. Modularity: The system should be designed with modular components to facilitate future updates and feature additions without impacting existing functionalities.



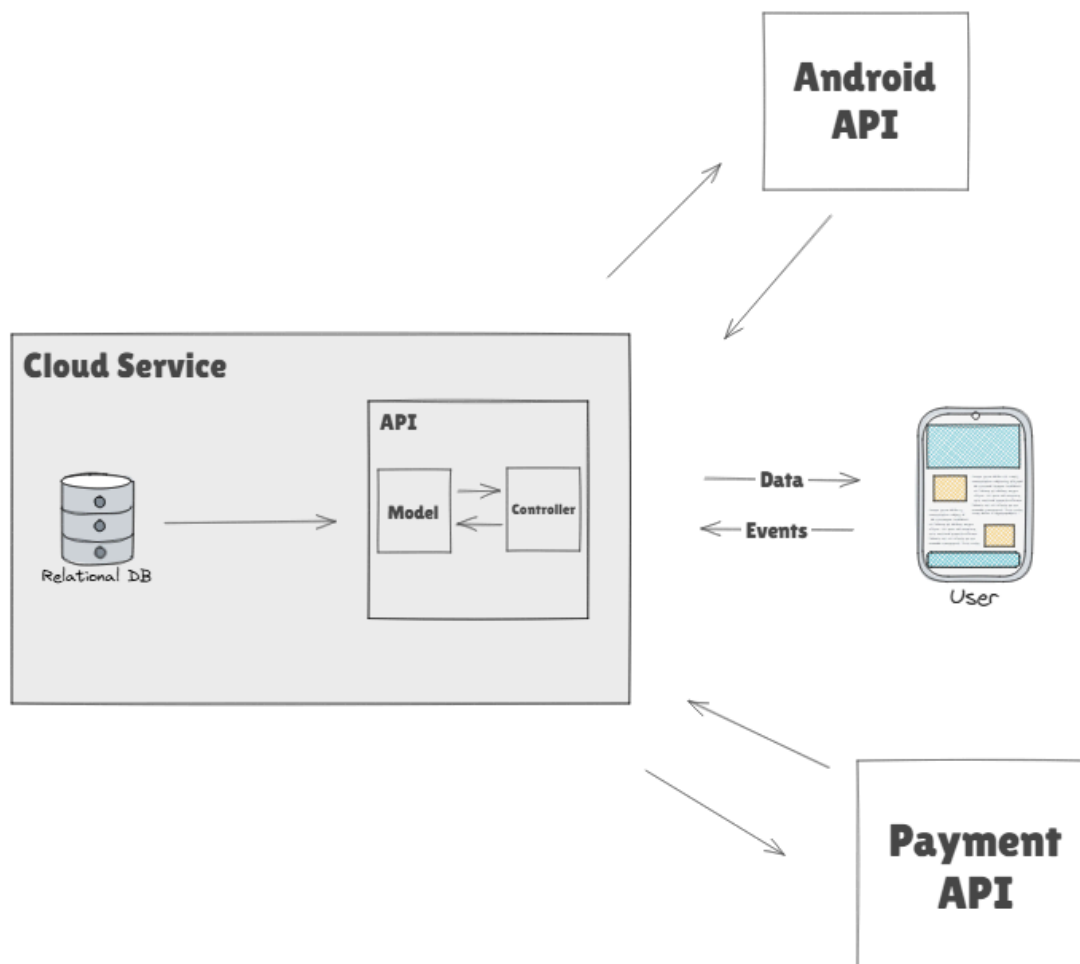
- c. Testability: The system should be designed to be easily testable (e.g., unit tests, integration tests).



## 4. SYSTEM ARCHITECTURE AND DESIGN

### 4.1 General System Architecture

Figure 1 - Software High Level Structure




Source: Own Authorship

With the wish of fast development, a very simple but functional architecture was set. A relational database is used to manage user data, as it's good to model structured data and can describe well the user relationship with the apps they use.

A simple RESTful API was applied to make the communication between the user and the database. The MVC style was applied in the API structure, differentiating the layers and making it easier to debug and test the application; as well making it more polished (the View layer being the data in json).

The Cloud Service is used to host the database and the API.



Along with all of this, we have a strong communication between the android API (to get user time in apps they blocked) as well as Payment API's, to empower payment both, by the user and by the application with ease and in a quick manner.

## 4.2 UI/UX

The visual part of the app was done with the focus of being easy to use, quick to understand and simple. This was done applying a community of design rules:

1. Few Colours: The app uses three main colours in HEX:
  - a. Linear gradient from **#1C1C1C** to **#666666** with the two colours set to 100% in the background;
  - b. **#FFFFFF** to icons and text;
  - c. Linear gradient from **#D812D4** to **#720970** with the last colour set to 93% opacity used for clickable icons and text.

With few exceptions to red, green and blue.

2. Few Screens: The app has just 13 screens in its first version, with the goal of fast development and ease of understanding.
3. Localization: Every screen has its name at the top or an indication at the bottom in the case of Home and Profile Screen. This helps the user know where they are and what's the next step.
4. Gestalt Principles: Gestalt Principles are used around the software to create a smooth user experience. Examples of these principles are the use of blank space, text opacity, contrast and proportion.

User flow was thought too. The app doesn't have any dead-end screen and the flow of login/signup and recovering of password was specially thought of, Given its importance.

The design of the user that's logged with one goal is the following (it's in Portuguese/PT-BR).

Figma: <https://www.figma.com/proto/SaZIOrFftZdfq2i7M8umPA/Purify?node-id=36-41&t=ZnrPKYPWHoyTNLdh-1>



### **4.3 Database Design**

### **4.4 API Design**

### **4.5 Cloud Environment**

### **4.6 PIX API**

## 5. SOURCES

The following sources were used along the project. It's a collection of useful resources that covers software programming, database design, project management and much more.

REACT NATIVE. React Native · A framework for building native apps using React. Disponível em: <<https://reactnative.dev/>>.

LUCIDCHART. What is an Entity Relationship Diagram (ERD)? Disponível em: <<https://www.lucidchart.com/pages/er-diagrams>>.