Caio A. G. Xavier

Dênio A. Almeida

Nícolas V. Auler

Finnish

Smart Personal Financial Management by Leveraging Machine Learning and the Open Banking API

São Paulo, SP

2024

Finnish - Smart Personal Financial Management by Leveraging Machine Learning and the Open Banking API

Dissertation in
Computer Engineering - Engenharia de Computação

Caio A. G. Xavier
Dênio A. Almeida
Nícolas V. Auler

Advisor: Prof. Bruno C. Albertini
Academic Year: 2024

Abstract

"Finnish" is a novel personal financial management tool that leverages machine learning and the Open Banking API to redefine personal finance management. Motivated by the rapid evolution of software delivery and the complex state of personal finances in Brazil, this work aims to address the increasing need for comprehensive and user-friendly financial management solutions. Finnish distinguishes itself by offering an intuitive platform that integrates advanced analytics for personalized financial insights and recommendations, making financial management accessible to all and promoting financial literacy. Comparing Finnish against existing solutions, its potential to fill market gaps and set new standards in the personal financial management domain is revealed. This study underscores the significant role of technology in enhancing financial stability and literacy, highlighting Finnish's contribution to the broader objectives of financial inclusion and improved user experience.

Keywords: Personal Financial Management, Machine Learning, Open Banking API, Digital Banking, Software Quality, Financial Inclusion, User Experience

Resumo

"Finnish" é uma ferramenta inovadora de gestão financeira pessoal que aproveita o aprendizado de máquina e a API de Open Banking para redefinir a gestão de finanças pessoais. Motivada pela rápida evolução na entrega de software e pela complexidade do estado das finanças pessoais no Brasil, este trabalho visa abordar a crescente necessidade de soluções de gestão financeira abrangentes e amigáveis ao usuário. O Finnish se distingue por oferecer uma plataforma intuitiva que integra análises avançadas para insights financeiros personalizados e recomendações, tornando a gestão financeira acessível a todos e promovendo a literacia financeira. A análise realizada compara o Finnish com soluções existentes, revelando seu potencial para preencher lacunas no mercado e estabelecer novos padrões no domínio da gestão financeira pessoal. Este estudo sublinha o papel significativo da tecnologia na melhoria da estabilidade e literacia financeiras, destacando a contribuição do Finnish para os objetivos mais amplos de inclusão financeira e melhoria da experiência do usuário.

Palavras-chave: Gestão Financeira Pessoal, Aprendizado de Máquina, Open Banking, Bancos Digitais, Qualidade de Software, Inclusão Financeira, Experiência do Usuário

Contents

1 | Introduction

1.1. Motivation

The development of this Thesis and the creation of Finnish, the product born from it, was specially fostered by two major factors:

1.1.1. State of Software Products

After introspecting on current state of software products, it's noticeable that a lot of changes have been happening to them, for example:

- How they're delivered

  When there weren't easy processes to update software remotely, via the Internet, the level of quality in the release had to be a quality that would guarantee uptime.

  Nowadays, the policy for software has been that of releasing as soon as possible, and go patching the mistakes and adding functionality afterwards [6].

  Of course, this has its advantages, software can be shipped rapidly and demands met, however, it's a two edged blade, since it's ever more common to encounter bugs.

- How they're implemented

  For the Apollo 11 Guidance Computer in 1969, its software had to be optimized to the extreme, both in terms of performance and memory, because the hardware it ran on was limited[1].

  But since then, hardware has had significant developments, with CPU's and GPU's achieving astronomical performance.

  However, there's a case to be made that software hasn't followed this steep raise of growth, because it can reap the benefits from the hardware developments.

  In the game development industry, where optimization is still a core principle, a point was made in the biggest game developer conference in Europe, devGAMM, to state that “software has been freeriding on hardware”[4].

This trend is particularly tied to long-running pieces of software, where legacy and contracts must be kept.

The core pieces of software used in Web Development, for example, the database engines, have been in place for decades. While that introduces a concept of stability, those products were devised in a completely different world stage for technology, specially when taking into account the rapid evolution of this area[7].

Exposure to academia leads to the discovery of bleeding-edge technologies, which motivates the building of a software product to advertise the state-of-the-art in the software engineering field.

One such technology, is the one of the rising stars Github repository: TigerBettle[8]. It aims to replace Postgres and MySQL, both database engines from almost three decades ago, with a new, distributed and “1000 times faster” finance-oriented database engine. Using this technology in Finnish will be a point of research in the following stages.

Furthermore, in Brazil, the amount of transactions has been growing rapidly 1.1.2. Which adds the need for designing a system capable of suiting the ever growing demands.

That means building a product with the following core principles:

- Keeping the customer always happy

  That means no bugs, no downtime. A software you can’t complain about, not even the 1% of the customer base, all the customers have to be content.

- Respecting the customer privacy and providing security

  To be elaborated in the following chapter.

1.1.2. State of Personal Finances in Brazil

Rise of banking with the advent of digital banks

Research done by BC shows that the average number of bank accounts was 5.2 in 2022.

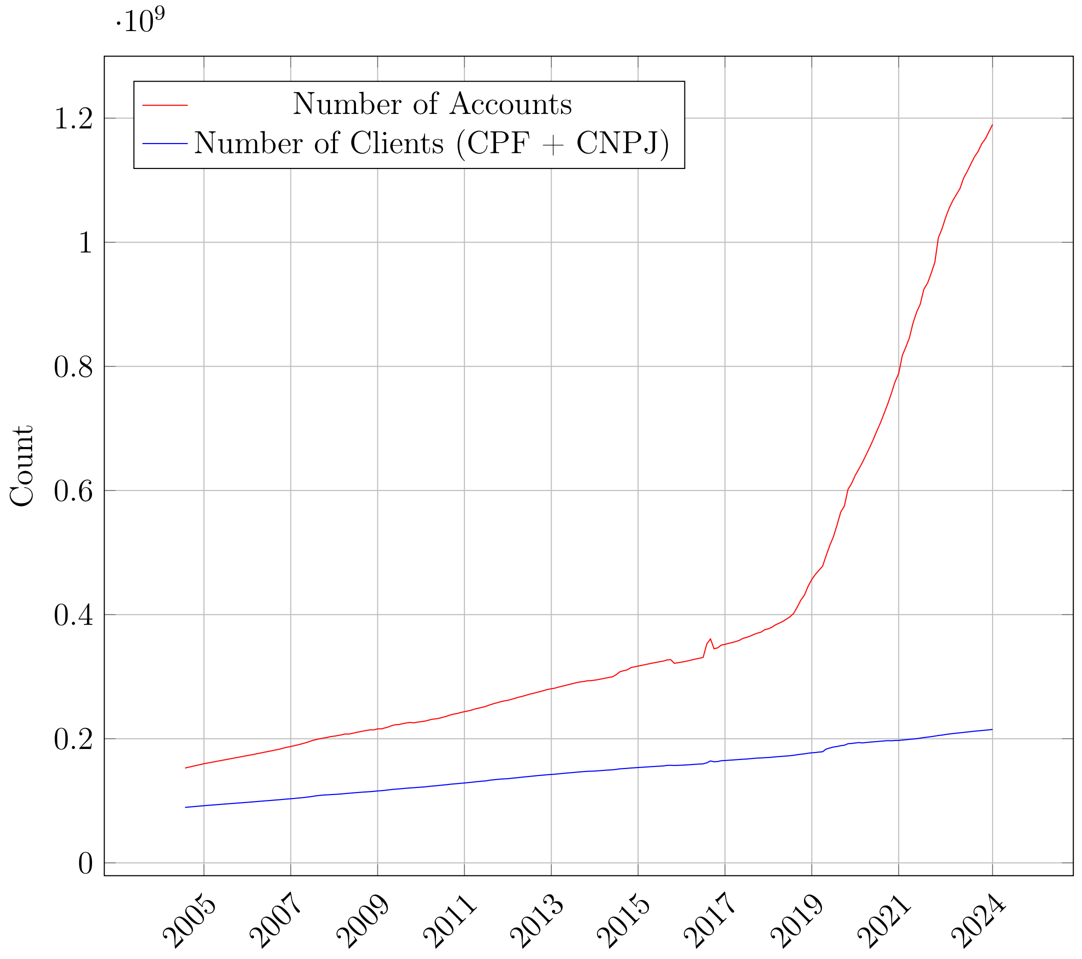


Figure 1.1: Time Series of Number of Bank Accounts and Number of Clients (CPF + CNPJ) in Brazil[3]

This shows an interesting scenario in Brazil.

- Why does a bank client need 5 different bank accounts?

  Firstly, having a diversified portfolio, helps by ensuring resilience to market fluctuations, but, also, helps with interest rates, as well as with credit and investment opportunities. The advent of digital banking has helped with this phenomenon.

  However, having lots of banks accounts makes the client more prone to mismanagement, specially when accounting for the segment without a formal education in finances or personal finances.

- This graph only shows that the population that has an account, would have on average 5 in 2022, however, not shown is that there is still 20% of the population that has an account, but didn't do any operations in 2022[9]. Why?

  One possible cause is that of unnacessibility. Speaking of digital banks, when a user creates an account through an app: the user can be used to onboarding processes, like those of creating credentials and providing necessary information and documents. But to manage finances through an app, that can be harder if there is no focus on those groups and no educational and tutorial material in the app.

  Finnish tries to solve this problem by both providing the user with some education material on financial basics, but also with an AI that will try to analyse the account and make suggestions, as well as help explain some phenomenon.

  Finnish also will have a more intuitive design and UX, to make the user feel that making operations isn't such a hard task.

Number of banking institutions growing rapidly

The number of banking institutions authorized by the Central Bank doubled from 2021 to 2022. This has been the landscape of Brazilian finances, where new, most of the times, digital institutions are appering, with focus on a specific niche, like cryptocurrencies.

But it's still to appear an institution that focuses on popularizing and making finances accessible, by allowing a client access to their finances in a holistic approach.

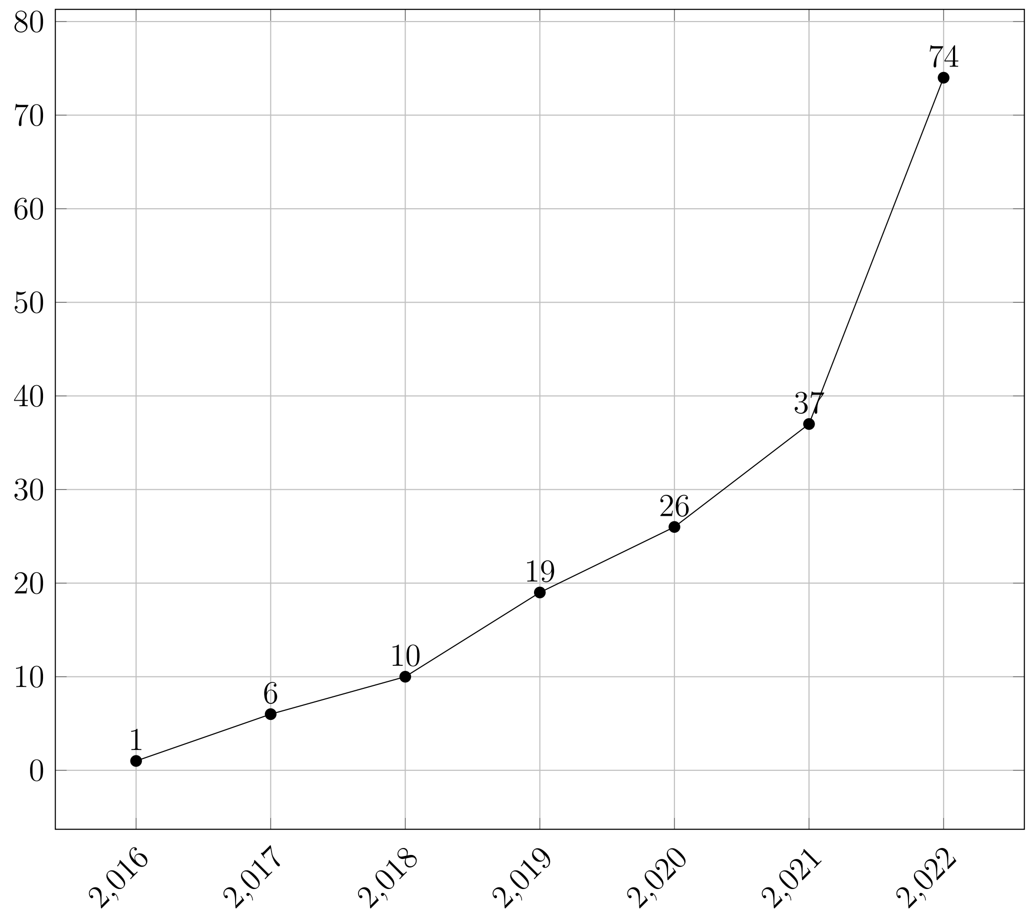


Figure 1.2: Number of financial institutions authorized by the Central Bank each year.

And this growth, also poses a challenge: are all of the institutions focusing on delivering good and secure products? Or is the race for the brazilian digital finance segment obfuscating the goals that don't directly drive revenue and market share?

Steep growth in number of transactions

The digitalization process in the payment instruments was accompanied by a change in the pattern of use of the channels of access to financial services, resulting in a continuous and substancial reduction in the use of presencial channels[5].

This payment instruments change can be seen in 1.3 and 1.4.

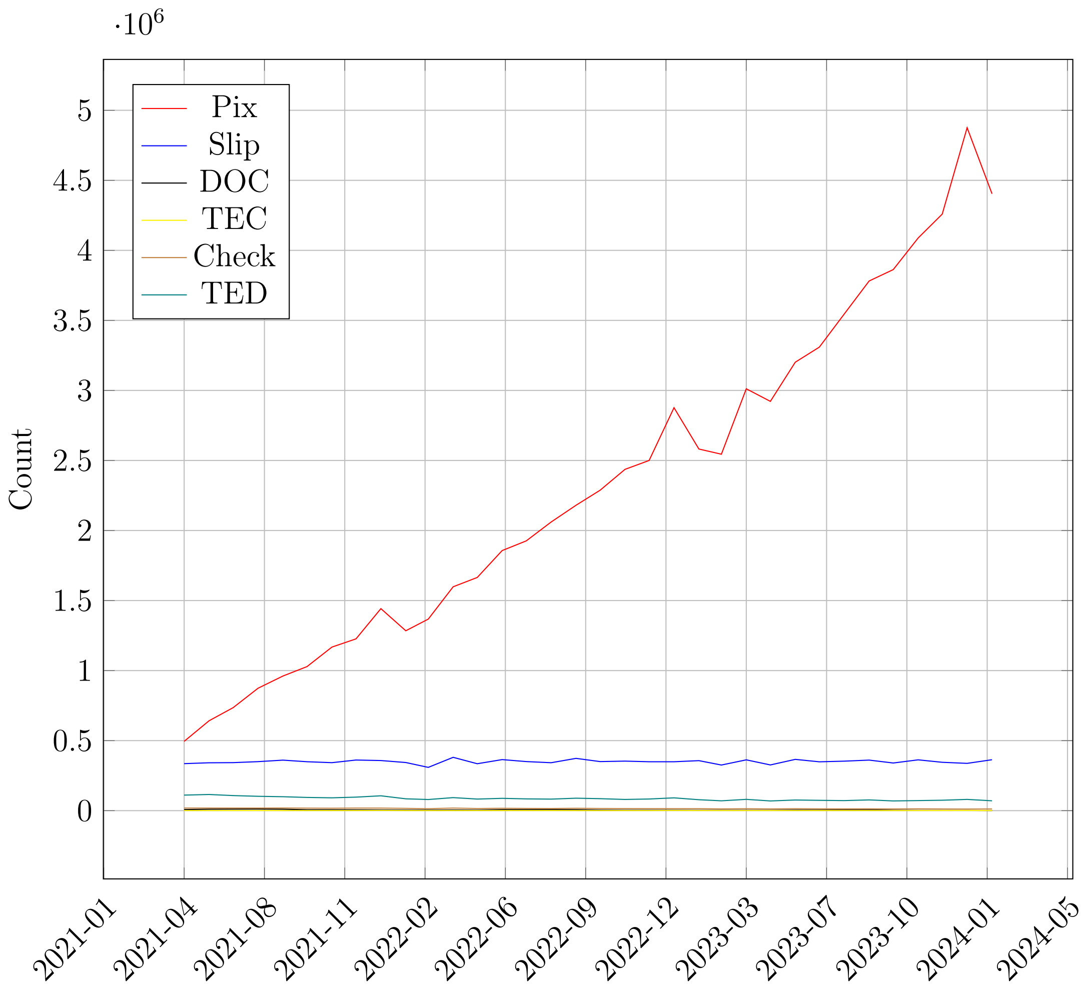


Figure 1.3: Number of monthly transferences per instrument in the last thee years[2]

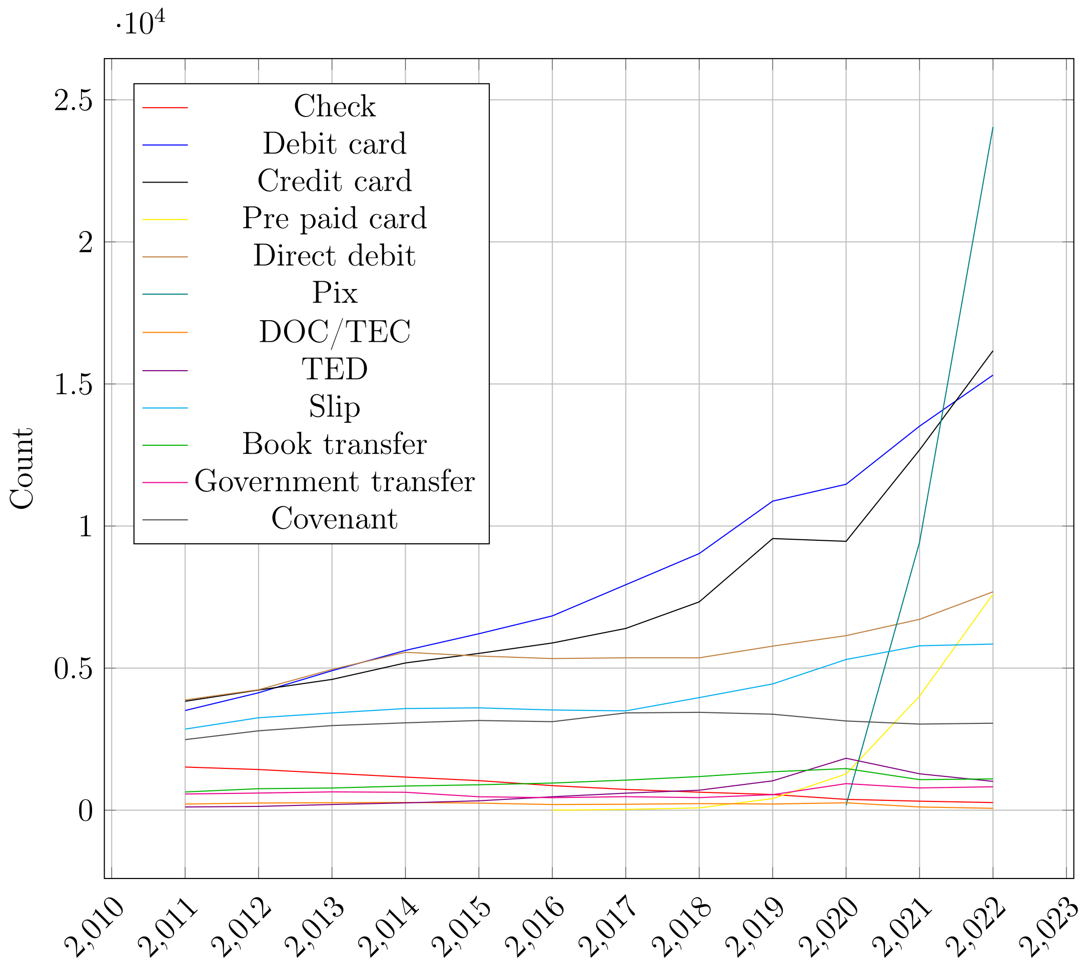


Figure 1.4: Number of yearly transactions per instrument[2]

This rise in the number of transactions shows the need for optimizing transactions in a new way, using new technology and processes.

Also, the digitalization process shows the need to have an exceptional digital support, so that clients can be satisfied, by not having a presencial channel of access, which would facilitate communication.

1.2. Objective

Our mission is to **Make personal finances accessible to all**.

Leveraging the brazilian Open Finance ecosystem, which is a tool that allows for banking institutions to exchange data among them, Finnish aims to provide the end user with these missing tools for personal finance. The app also will provide the user with educational content for finances, both tailored for their data and non-specific training.

1.3. Justificative

The development of this project is crucial for addressing a significant gap in the financial management landscape, where individuals often struggle to manage accounts across multiple banks and platforms. In today's society, financial transparency and control are essential, yet the complexity of dealing with scattered financial data makes it difficult for many users to stay on top of their personal finances. This system leverages the Open Finance framework to centralize a person's financial life into a single application, providing an innovative solution to this challenge.

The system simplifies the user's mutiple bank accounts management by integrating his financial data and offers a better vision of his finance. Furthermore, the use of a machine learning layer that provides personalized financial insights and recommendations adds value to customers, and also help them to make better decisions and improve their financial health.

1.4. Work Organization

This section consist of the following parts. The work method specifies how the project was carried out, outlining every step like requirements specification, design, implementation and testing. Although which results from a certain appraisal phase are achieved is clarified in later chapters, this section outlines the workflow utilized.

The Requirements Specification defines the project requirements, tailored to the development of the system. It includes a comprehensive list of both functional and non-functional requirements, ensuring the system meets user needs.

Noew, the Development of the Work chapter illustrates the transformation of requirements into a functional product. It is organized into subsections to facilitate understanding of the development process, including:

- Technologies Used : This subsection highlights the key technologies and tools employed during the product development, explaining their relevance and contribution to the project.
- Implementation: Here, the results of the design and implementation phases are discussed, along with justifications for the decisions made during these processes.
- Testing and Evaluation : This section outlines the testing procedures undertaken, and validation tests, tailored to the system's needs.

2 | Design

2.1. Sector Analysis

2.1.1. National Market

To provide context for the project, a study was conducted on the main existing market solutions for the previously listed issues. Among these, we observed 3 Brazilian applications, each with different scales (in terms of user numbers); and 2 foreign applications, each with distinct purposes.

Since mid-2017, Brazil has experienced a notable influx of legislative proposals aimed at regulating the flourishing fintech sector. This regulatory initiative is underpinned by the dual goals of enhancing legal security for stakeholders and stimulating a competitive landscape within the sector. The advent of these regulations is a testament to the recognition of the critical role that fintech companies play in the broader financial ecosystem, driven by rapid technological advancements and evolving consumer preferences. In response to these dynamics, Brazilian regulatory authorities have embarked on a comprehensive approach to governance, marked by the introduction of several key regulatory frameworks.

Notably, the Regulation of Investment Crowdfunding (CVM No. 588, 2018) stands as a seminal piece of legislation designed to govern the domain of crowdfunding platforms. This regulation is instrumental in establishing a legal framework that safeguards investor interests while providing a structured pathway for startups and small businesses to access alternative funding sources. Furthermore, the introduction of the Regulatory Sandbox (Complementary Law No. 182, 2021) signifies a progressive step towards fostering innovation within the fintech space. This legal provision facilitates a controlled environment wherein emerging fintech entities can test novel products and services under the temporary supervision of regulatory bodies, without being subjected to the full gamut of regulatory requirements. This approach not only accelerates the pace of innovation but also enables regulators to adapt and refine regulatory frameworks in alignment with technological advancements.

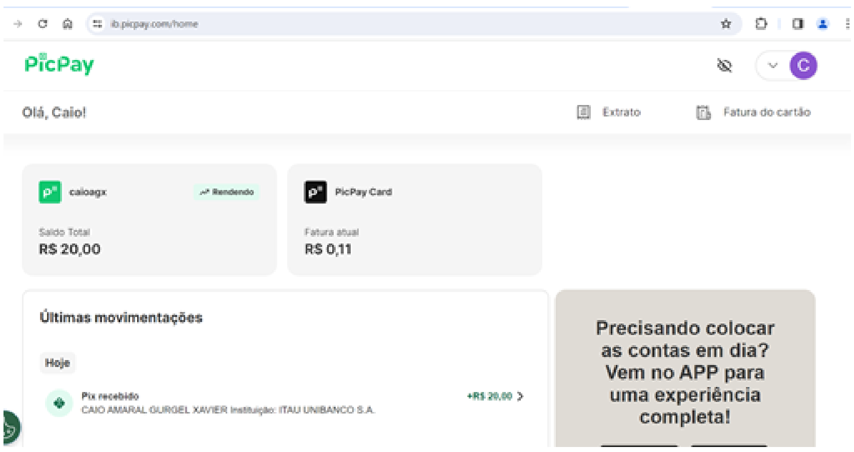
Lastly, the Open Finance (Circular 4.032, 2020) regulation emerges as a critical initiative aimed at promoting financial inclusivity and competition. By mandating the sharing of customer data among financial institutions, with explicit customer consent, this regulation paves the way for a more integrated and customer-centric financial services ecosystem. This paradigm shift towards open finance underscores the commitment of Brazilian regulators to leverage technology in enhancing financial accessibility and efficiency.

In summary, these regulatory measures collectively embody Brazil's strategic approach to fostering a resilient and competitive fintech sector, acknowledging the intricate balance between regulation and innovation in the rapidly evolving financial landscape.

These regulatory changes have allowed many companies and business models to emerge. Below, we list 3 of them, each with distinct characteristics:

Picpay (+100mn downloads)

PicPay functions primarily as a digital wallet, streamlining transactions between various parties. Its web version, however, is somewhat restricted. Through integration with Open Finance, PicPay facilitates comprehensive banking interactions for users. In the realm of personal finance, it offers free services for financial management and planning, having no paid services.



Olá, Caio!

caioagx
Saldo Total
R$ 20,00

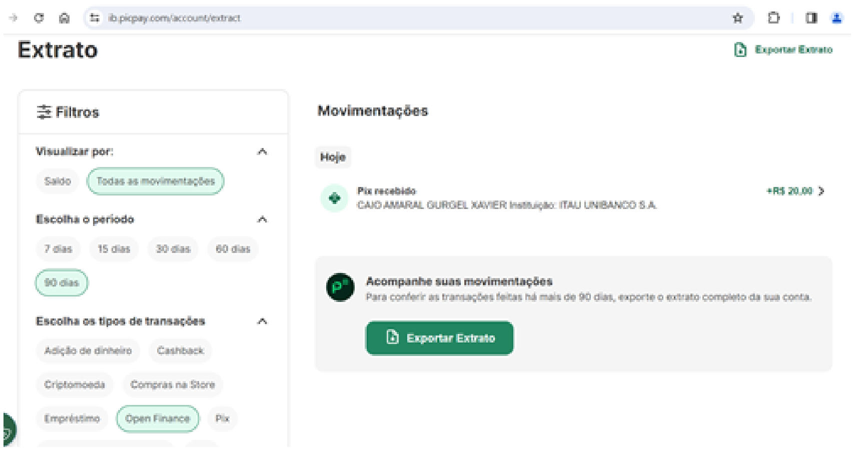
Fatura atual
R$ 0,11

Últimas movimentações

Pix recebido
CAIO AMARAL GURGEL XAVIER Instituição: ITAU UNIBANCO S.A.
+R$ 20,00

Precisando colocar as contas em dia? Vem no APP para uma experiência completa!

Figure 2.1: Picpay's web home dashboard



CAIO AMARAL GURGEL XAVIER Instituição: ITAU UNIBANCO S.A.

Para conferir as transações feitas há mais de 90 dias, exporte o extrato completo da sua conta.

Figure 2.2: View of transactions' details

Organizze (+1mn downloads)

Organizze is a comprehensive application designed to assist users with their personal financial management, similar to the concepts proposed in this project. It offers a user-friendly platform that allows for easy modifications and adjustments according to the user's financial data and preferences. However, one notable limitation is its lack of automation features, as well as the absence of AI-powered tools for the categorization of financial transactions, which could enhance the efficiency and accuracy of financial tracking.

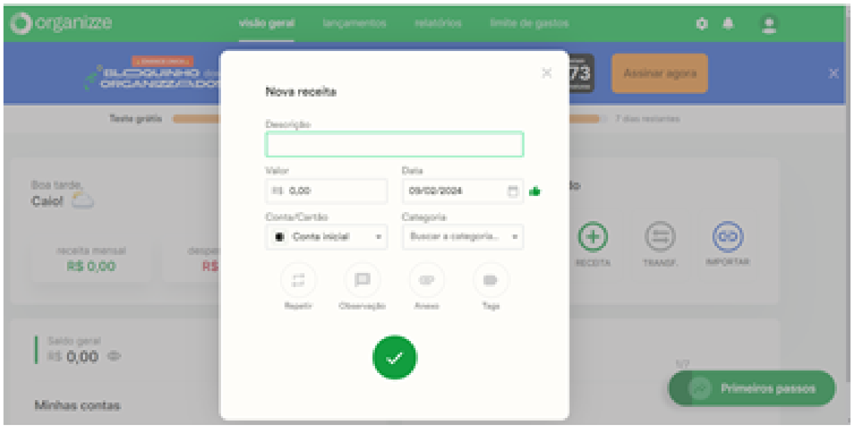


Figure 2.3: View to add financial transactions

Organizze adopts a subscription-based model. The service provides various plans, among which is a three-year subscription option priced at BRL 358.00. It also offers a 7-day free trial. This trial period is a key aspect of their service, encouraging users to experience the application's capabilities before making a commitment.

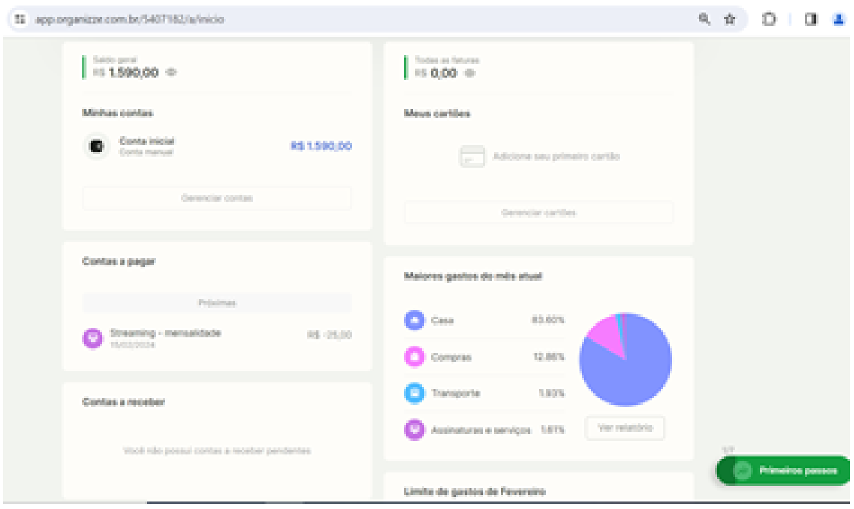


Figure 2.4: Organizze's home dashboard

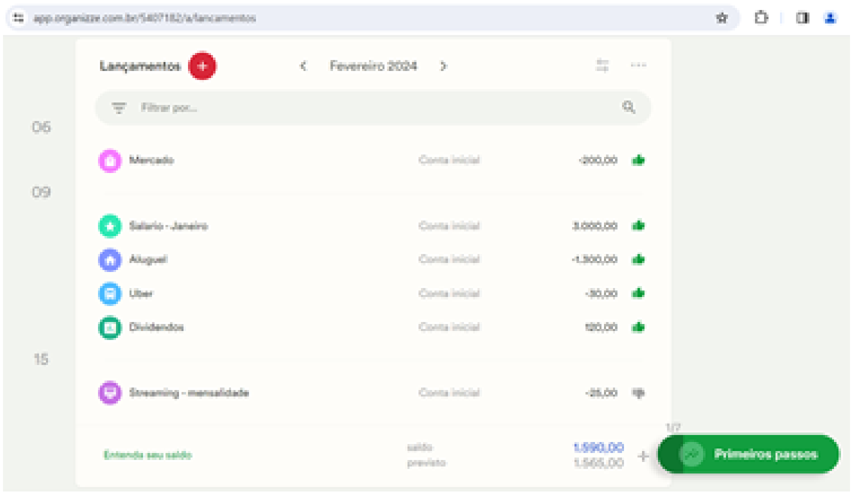
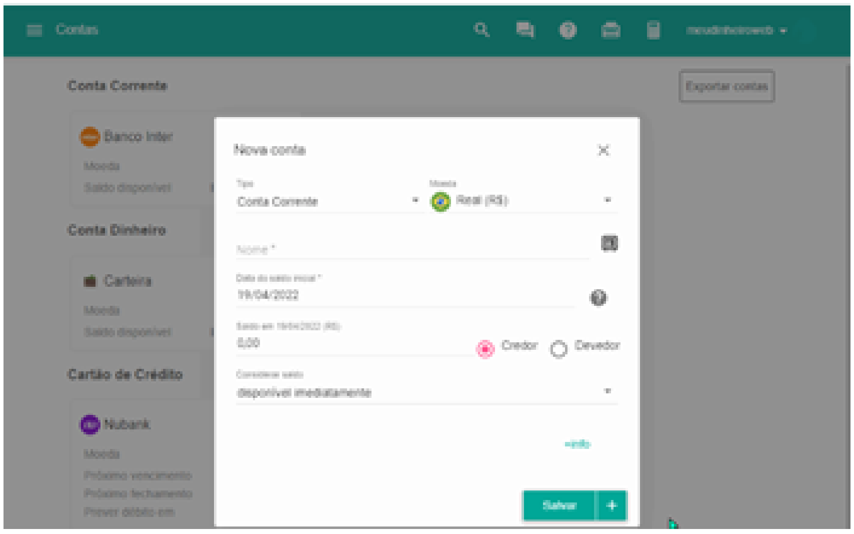


Figure 2.5: View of categorized financial statement

MeuDinheiro (+100k downloads)

MeuDinheiro's focus extends beyond individual consumers to encompass small businesses. The application provides a free version closely resembling Organizze, facilitating the inclusion of both expenses and income.



Conta Corrente

Banco Inter
Moeda
Saldo disponível

Conta Dinheiro

Carteira
Moeda
Saldo disponível

Cartão de Crédito

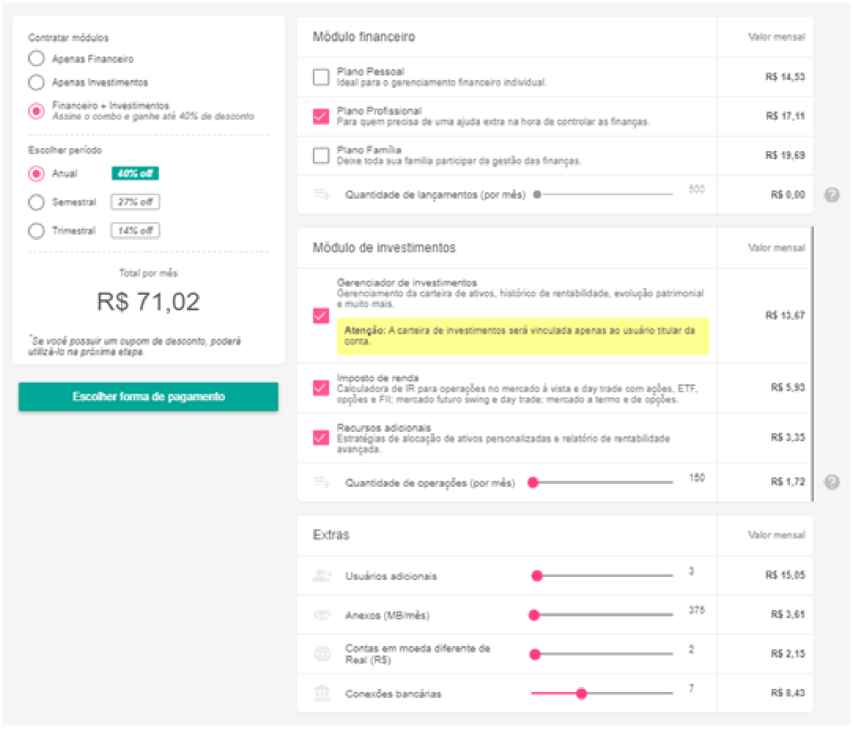
Nubank
Moeda
Próximo vencimento
Próximo fechamento
Prever débito em

Nova conta

Tipo: Conta Corrente
Moeda: Real (R$)
Nome *
Data do saldo inicial *: 19/04/2022
Saldo em 19/04/2022 (R$): 0,00 [✓ Credor] Devedor
Considerar saldo: disponível imediatamente

Figure 2.6: View to add financial transactions

For users seeking enhanced functionality, the application's premium version presents a comprehensive suite of features. This includes sophisticated tools for monitoring the progression of net worth, setting specific income and expense objectives, and integrating with bank accounts, among others. The range and depth of these features are designed to be flexible, adapting to the varied demands of users.


Contratar módulos
Apenas Financeiro
Apenas Investimentos
[✓ Financeiro + Investimentos]
Assine o combo e ganhe até 40% de desconto

Escolher período
[✓ Anual] *40% off*
Semestral *27% off*
Trimestral *14% off*

Total por mês
R$ 71,02

**Se você possuir um cupom de desconto, poderá utilizá-lo na próxima etapa*

Módulo financeiro		Valor mensal
☐ Plano Pessoal Ideal para o gerenciamento financeiro individual.		R$ 14,53
✓ Plano Profissional Para quem precisa de uma ajuda extra na hora de controlar as finanças.		R$ 17,11
☐ Plano Família Deixe toda sua família participar da gestão das finanças.		R$ 19,69
Quantidade de lançamentos (por mês)	500	R$ 0,00

Módulo de investimentos		Valor mensal
✓ Gerenciador de investimentos Gerenciamento da carteira de ativos, histórico de rentabilidade, evolução patrimonial e muito mais. **Atenção:** A carteira de investimentos será vinculada apenas ao usuário titular da conta.		R$ 13,67
✓ Imposto de renda Calculadora de IR para operações no mercado à vista e day trade com ações, ETF, opções e FII; mercado futuro swing e day trade; mercado a termo e de opções.		R$ 5,93
✓ Recursos adicionais Estratégias de alocação de ativos personalizadas e relatório de rentabilidade avançada.		R$ 3,35
Quantidade de operações (por mês)	150	R$ 1,72

Extras		Valor mensal
Usuários adicionais	3	R$ 15,05
Anexos (MB/mês)	375	R$ 3,61
Contas em moeda diferente de Real (R$)	2	R$ 2,15
Conexões bancárias	7	R$ 8,43

Figure 2.7: View of plans and additional features

Comparative Table

Upon analyzing each competitor within the local landscape, it is possible to summarize their characteristics in the table presented below. This comparison is instrumental for advancing with the project.

	Organizze	**Picpay**	**MeuDinheiro**
Charts with income/expenses evolution	x		x
Extract categorization	x	x	x
Openfinance Licensing	x	x	x
AI recommendation tools			
Automatic bank's data extraction		x	
Manual insert of expenses/income data	x		x
Web app	x	x	x
Mobile app	x	x	x
Paid service	x		x

Table 2.1: Comparative list of peers and its features

2.1.2. International Market

The fintech sector outside Brazil presents a dynamic and varied landscape, shaped by diverse regulatory approaches and market maturity levels. In developed markets, regulations such as open banking have spurred innovation in personal financial management (PFM) apps, enabling them to offer more comprehensive services by accessing bank data with user consent. This regulatory environment fosters a competitive ecosystem where fintech startups and traditional financial institutions vie to provide user-centric solutions.

In this context, two companies stand out: one that is emerging, with the integration of users' bank accounts, and another that is more established in the market for personal finance management.

Plaid (+100k downloads)

Plaid is an American company operating in seven countries, with approximately 100,000 users, acting as a technical intermediary between financial institutions and users. It is capable of consolidating financial data from various sources and categorizing transactions.

Mint

Mint is the closest global peer to Organizze, boasting 25 million users, with trend charts on the evolution of expenses. Its most significant distinction comes from possessing AI tools for expenditure recommendations.

2.2. Ideas grooming

At first, emerging technologies were considered, aiming for an application with more features and market differentiators. Among them, DREX was taken into consideration. To that end, the group contacted a collaborator from BTG Pactual bank, who was selected to work at Bacen, Banco Central do Brasil, and assist in the development of DREX.

The contact was made, and unfortunately, he said that the technology is not expected to be deployed in the production environment by the end of the year, when our dissertation needs to be done. Additionally, the group of people from banks who are developing and testing DREX is very restricted, thus diminishing the prospects for this group to leverage this tool. DREX would be a highlight for the application, as it would bring clients closer to a decentralized ecosystem, where purchases could be safer and available at any time of the day. For instance, it envisions that through smart contracts, a person looking to buy a car from someone else would receive ownership simultaneously as their money is transferred, without involving third parties and ensuring security for both parties.

3 | Requirements Specification

The developed project implements a personal financial management application that connects the user's accounts via the Open Finance provided by the Pluggy platform.In addition, there is a machine learning layer that provides financial insights and recommendations.This system is accessed via a web application and an Android mobile application.Therefore, based on this scope, the requirements are listed below.

3.1. Functional Requirements

- The system must allow users to set their personal data, username and password for registration;
- The system must allow users to change their password if it was forgotten or if it wants;
- The system must allow users to add or to delete its accounts from any bank;
- The system must allow users to add non-trackable transactions e.g, cash ones;
- The system must encourage users to create stronger passwords on sign-up phase;
- The system must show financial incomes and outcomes in dashboards e.g, pie charts for month expenses;
- The system must give expenses suggestions in the app based on the user history;
- The system must categorize users expenses e.g, food related expenses.

3.2. Non Functional Requirements

- The system shall be safe:
  - The system shall encrypt the passwords;
  - The users sessions will be expirablle;

  - The system shall have captchas for sign in or sign up;
  - The server will have a Rate-limit;
  - The system shall support a two factors authentication;
- The system shall work for users privacy, since it will respect the LGPD;
- The system shall be available 24/7;
- The system shall implement a machine learning layer for categorization.

4 | Implementation

4.1. Used Technologies

4.1.1. Server

In the development of the Finnish Server, the general programming process of separating Hypermedia API's from Data API's was followed.

Internally, a website written with HTMX is being used for testing purposes. This is where the Hypermedia API fits in.

The server is written as a Rest service in HTTP 1.1, which is written in Async Rust, on top an Async database engine, called SQLx, running on Postgresql.

It's the server responsibility to integrate with the external clients being used for Emailing, Webhooks, Open-Finance integration and Proof-of-Work captcha verification, as well as providing ergonomic contracts to the Mobile client, which consumes the Server's Data API's.

Infrastructure

The server is deployed to a Nixos private server, which allows us to have Infrastructure as code, by defining reproducible linux services, instead of the more traditional route of containers.

Having our private server also allowed us to add Observability services, like a Grafana Beyla collector that sends Metrics and Traces to a Grafana Cloud dashboard.

4.1.2. Mobile

From the Mobile side, the application was developed using the Kotlin programming language, in the Android Studio developing environment. Through its native resources, it was easier to develop interfaces and features. And, the user interface was developed using

Figma, always trying to do a friendly interface and easy-to-use.

The developing process, actually, started with a web application version, which served as a starting point for the mobile layer. So, the mobile views were inspired in the web ones, trying to maintain a similarity between platforms.

4.1.3. Categorization

In the development of the categorization system, we initially implemented DistilBERT, a pre-trained language model optimized for efficiency, to classify transaction descriptions. However, due to issues with non-tokenized transaction data in Portuguese, we transitioned to a custom lexogram-based approach, where specific patterns within transaction descriptions were used as features. We then utilized XGBoost, a powerful gradient boosting algorithm, to enhance classification accuracy. Real transaction data was retrieved through Pluggy's API and mapped to predefined categories for training and testing the model.

Model Server

In order for the model to talk to the rest of the Finnish ecosystem, we decided to deploy it as a server, since this entire project is being run on a very limited timeline, and also with only 3 developers. So, to avoid us having to make compromises in the actual server of the application, like writing it in Python FastAPI instead of Rust Axum, and avoid having to add unnecessary complexity to the Model, like writing it in Rust, instead of Python, we chose to write only the Model in Python, and, to make it communicate with our Rust server, we turned it into a server of its own, so the Rust one could act as client.

4.2. Project and Implementation

4.2.1. Server

The server's implementation followed a "provide the most, as early as possible" mentality.

Firstly, a manual usage of the HTMX website was developed, so users could already input their expenses and track them through the website.

Then, the focus shifted to the Data API, since the Mobile app became the main priority. It's important to note that the Mobile app came after we started implementing the server.

After the API's were implemented for the mobile app, integration with Open-Finance began, so users could automatically track their expenses.

From then, the focus shifted to testing, architecting and restructuring for webhook integration and good database layouts, also taking categorization model needs in consideration.

Finally, we integratedour model into the server flows, so we can both categorize users transactions and improve our model with the new data (and hopefully, user feedback).

This was done by providing our Model as an API, a server. And our Rust code would act as the client for this Model service, and provide a processed set of API's back to the Mobile.

In the next Categorization section we'll describe in more detail this Model server, but, in the optics of the Rust server, we consume the Model API via a categorization service and a training service.

The first API is served transparently to the Mobile, since the categorization is sent inside of other flows, like the transaction listing.

And the second is explicitely called when a user requests a re-categorization of a transaction, which causes the model to be re-trained.

Infrastructure

The DevOps concern of Finnish started with a Bleeding-Edge approach, not only were we using a Infrastructure-as-Code model, we were using a "Infrastructure-generated-from-Rust-macros" model, which doens't even have a term for it yet.

But that quickly became too restrictive, since I often had to open issues in the cloud provider github libraries for Rust, and ultimately we decided that also investing time in becoming contributors to this open source innovation wasn't going to go well with the extremely short time to develop this Project.

Moreover, the need for having Observability was growing, and, again, having to use that through the lens of Shuttle (this experimental approach), was going to be harder than just spawning an OpenTelemetry Collector in a private Linux instance.

So we migrated to a private server in Digital Ocean, running Nixos. We first had to Nixos-infect a regular Ubuntu server. Then, we could develop our Nix reproducible environments, with the server and Grafana Beyla auto-instrumented collector and regular Linux Systemd services.

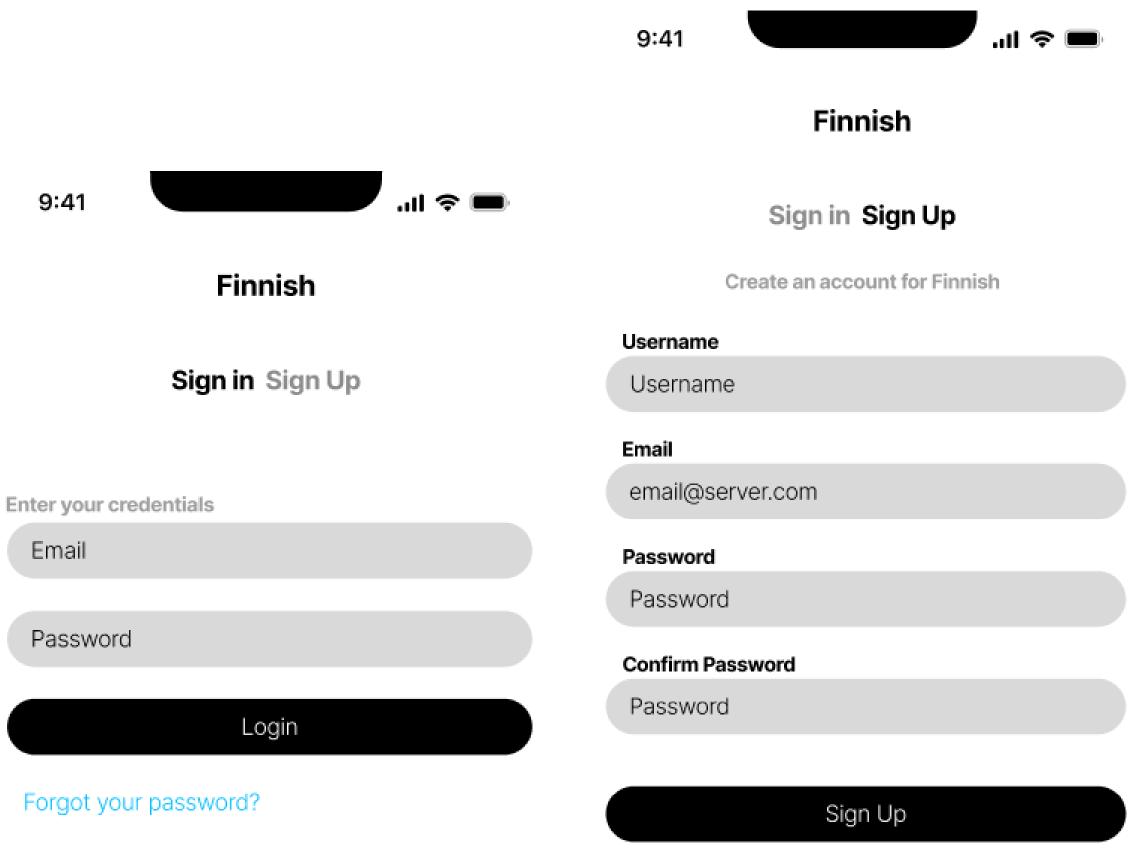
This allows for ultimate flexibility, and responsability, while we also remain at a Cutting-

Edge field of DevOps.

So, we deployed the Postgres database, the Rust server, the Grafana collector, as well as the Model server (which is a Python FastAPI server) to this NixOS machine, via nix-configured files.

4.2.2. Mobile

First, for the mobile development, the design was made in Figma where it was created layouts and user flows based on usability principles. At frontend end designs finalization, it started with mobile app development and simultaneously integrating the same with backend as APIs for endpoints that were getting prepared. The intended screens are showed below in Figures 4.1, 4.2 and 4.3.



(a) SignIn screen. (b) SignUp screen.

Figure 4.1: Figma SignUp and SignIn screens

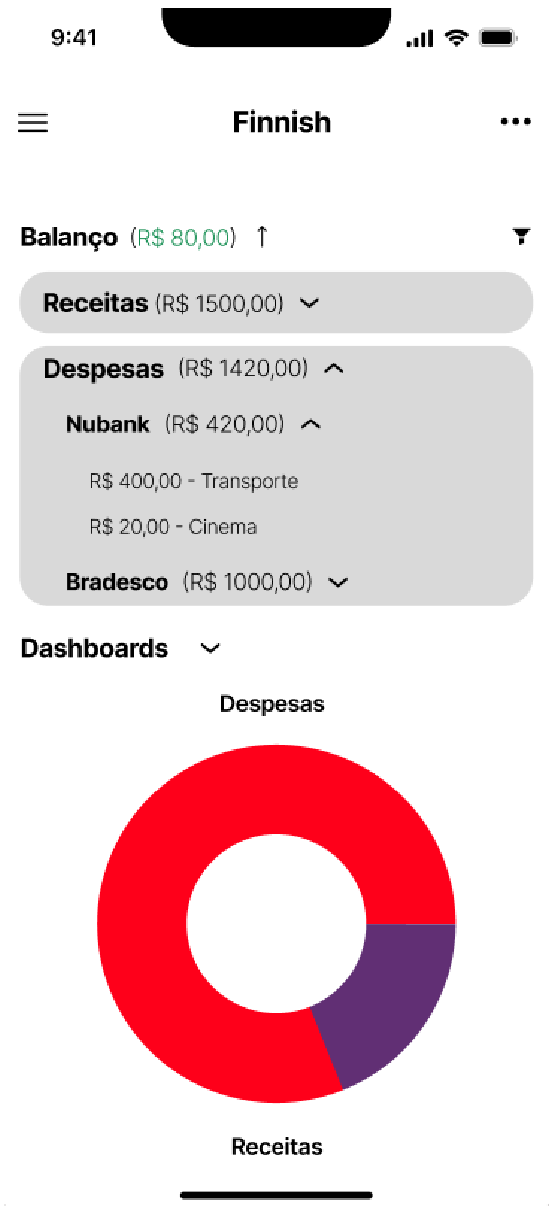
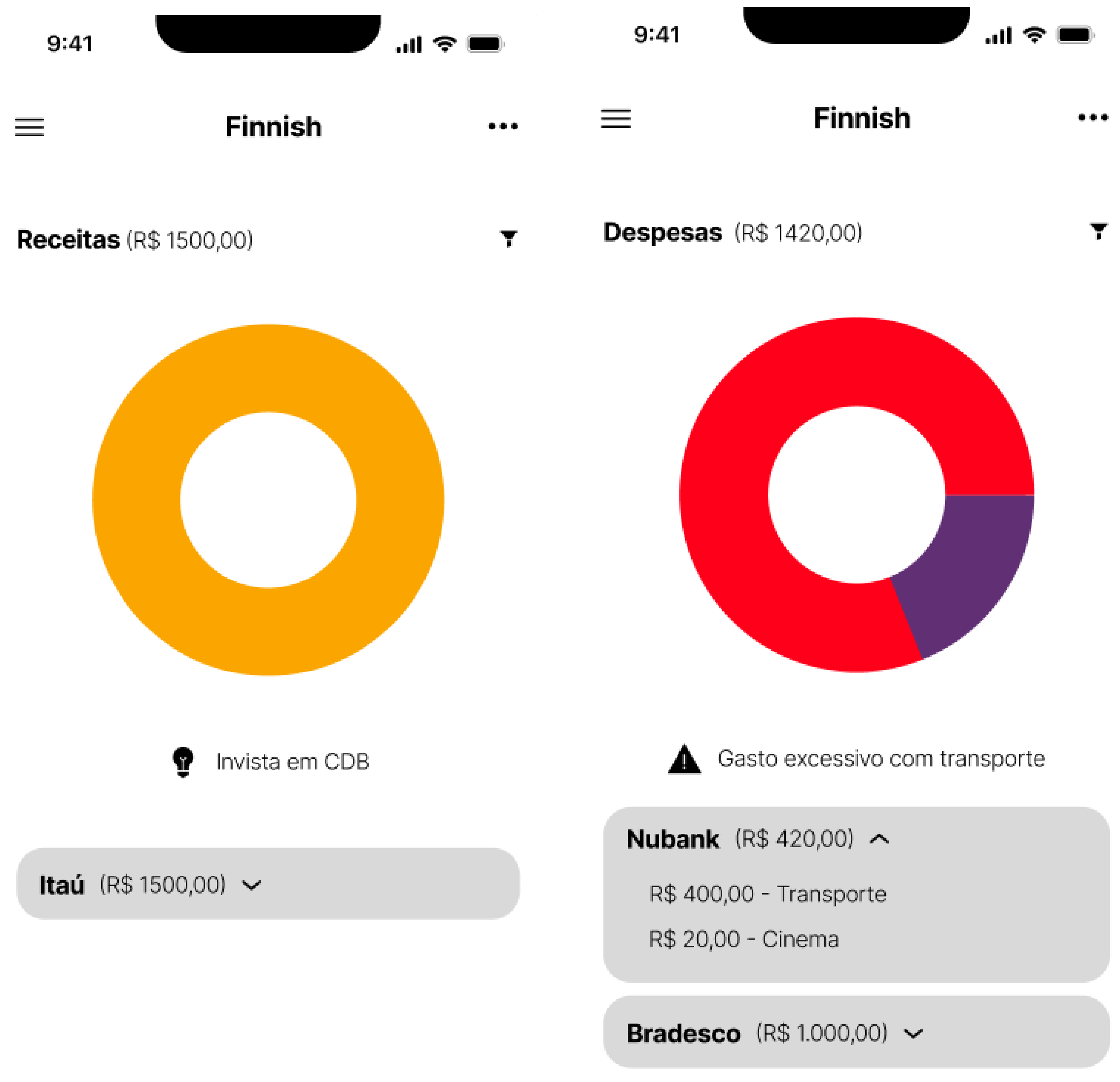


Figure 4.2: Figma - Home screen



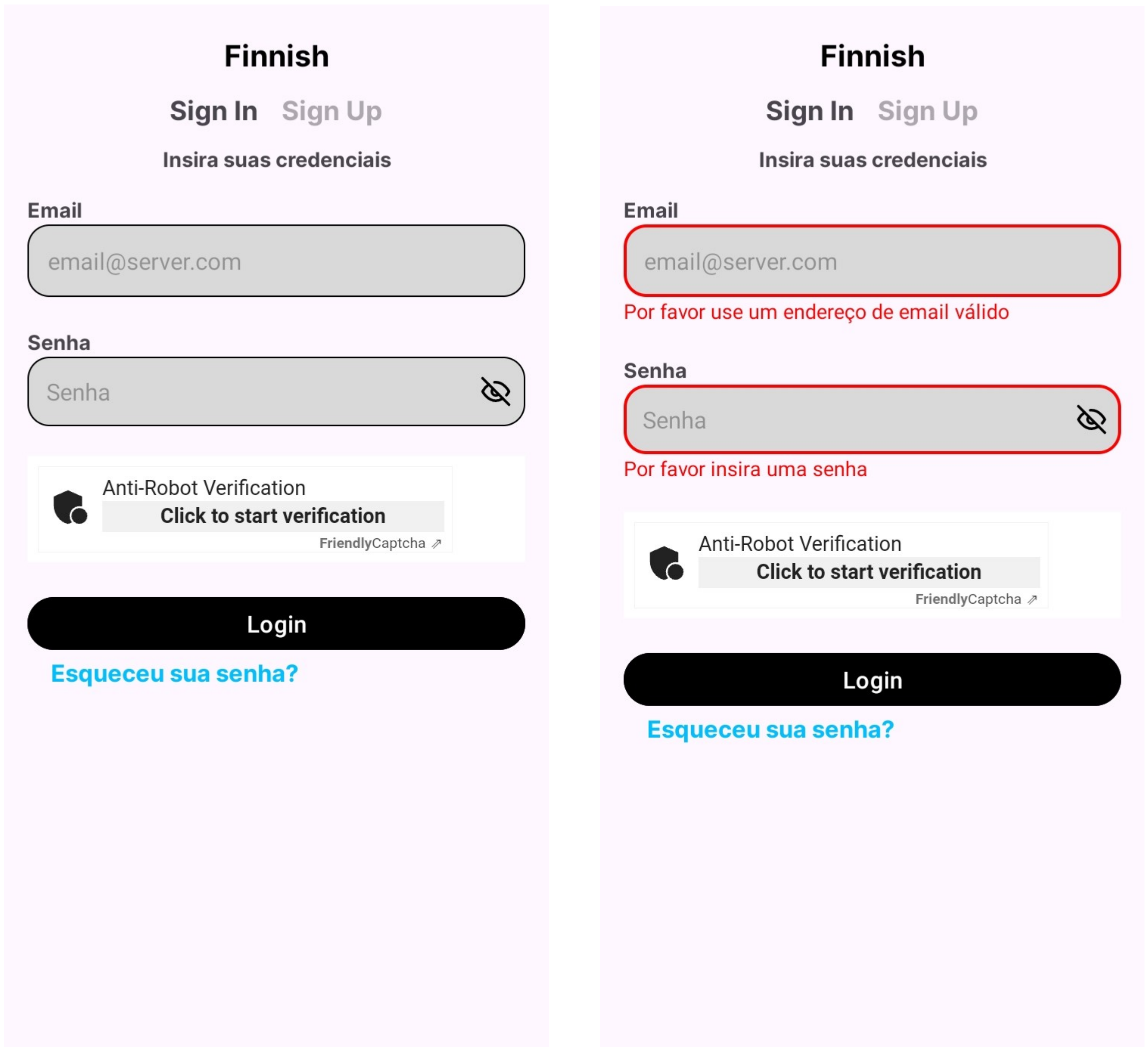
(a) Incomes screen. (b) Outcomes screen.

Figure 4.3: Figma Incomes and Outcomes screens

Each functionality added was tested to ensure it worked as intended before moving on to the next feature. This iterative process helped maintain a quality throughout development. The integration with Pluggy, which provides financial data, is the final step. It worked on incorporating their widget into the app to enable seamless access to users' financial information. But it is still needed to display expenses from banks got from the widget related to Open Finance.

Now, it is described how the application was set. The app has the Sign In, Sign Up, Forgot My Password, Change Password screens, a screen for entering the confirmation email, a screen for entering the MFA code, Home screen, financial incomes screen, financial output screen. Still within the Home screen, there is a dialog box for adding untrackable expenses and one for editing them.

The app starts with the Sign In screen, for entering email and password, as well as a dynamic captcha. There is a button to actually Sign In and the data is sent to the server. If they are incorrect, the wrong fields are outlined in red and an error message is shown under each field and the captcha is restarted. The captcha must be entered too. The screens developed for this can be seen in Figure 4.4.



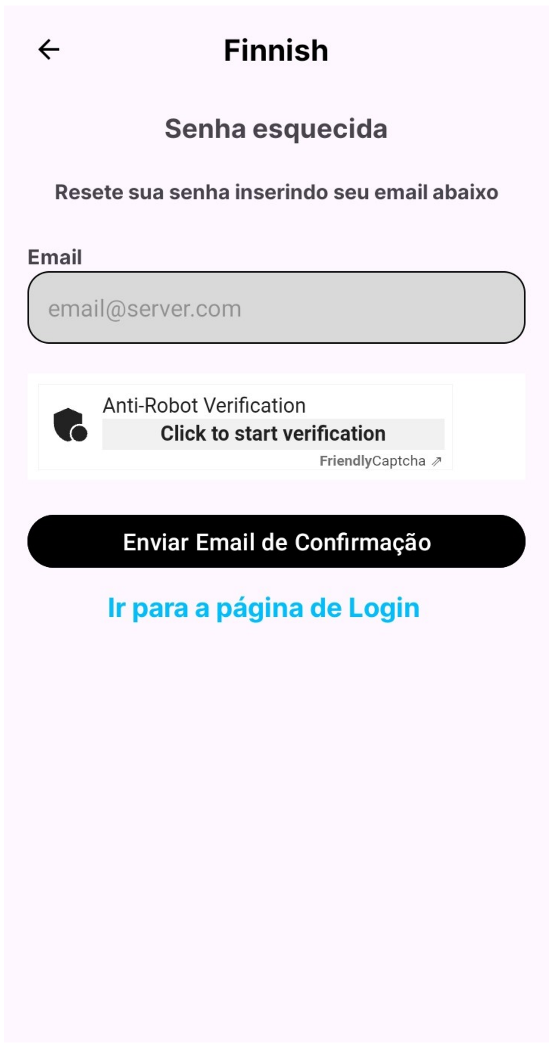
Por favor use um endereço de email válido

Por favor insira uma senha

(a) SignIn App. (b) SignIn validations screen.

Figure 4.4: SignIn and validations screen developed

If the user has forgotten the password, there is a field with the text 'Esqueceu Sua Senha?' which takes the user to a screen where they enter their password reset email. This screen basically has the field and the button to send the email to the server, and a button to return to the Sign In screen. Once the user has received the message in the registered email, they are taken to a screen on the web where they register their new password. This can be seen in Figure 4.5.



Finnish

Senha esquecida

Resete sua senha inserindo seu email abaixo

Email

email@server.com

Anti-Robot Verification

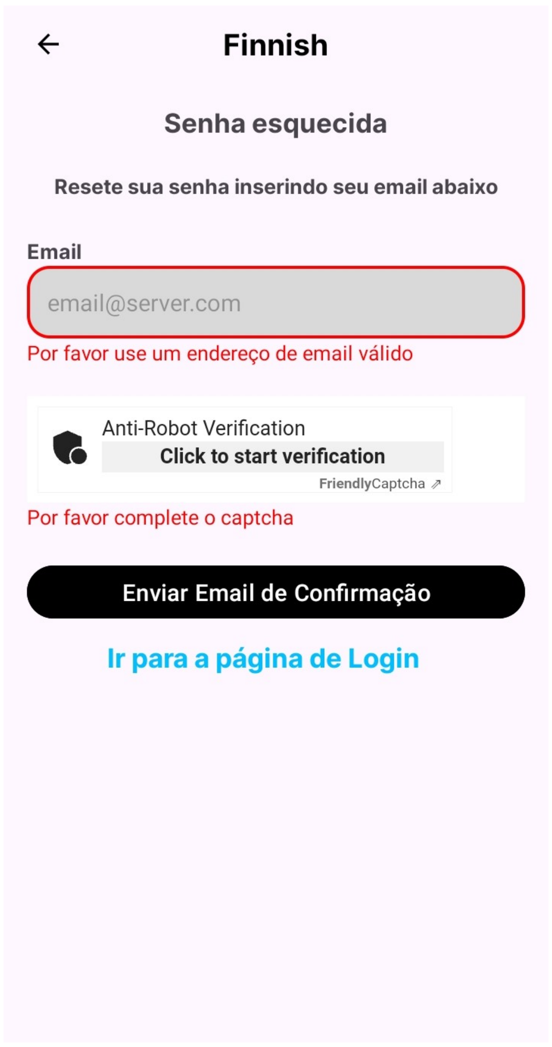
Click to start verification

FriendlyCaptcha

Enviar Email de Confirmação

Ir para a página de Login

(a) Forgot password screen.



Finnish

Senha esquecida

Resete sua senha inserindo seu email abaixo

Email

email@server.com

Por favor use um endereço de email válido

Anti-Robot Verification

Click to start verification

FriendlyCaptcha

Por favor complete o captcha

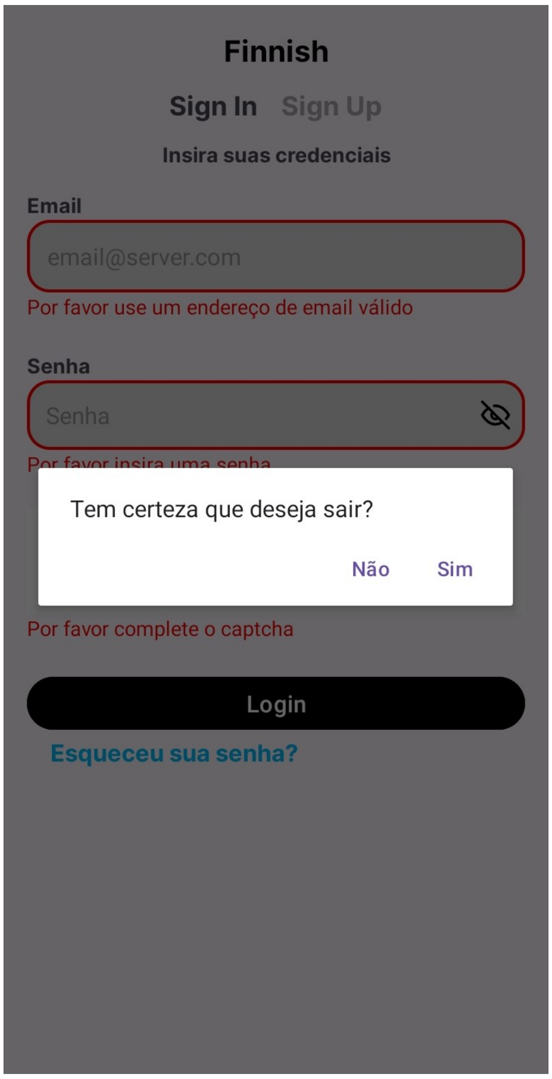
Enviar Email de Confirmação

Ir para a página de Login

(b) Forgot password validations screen.

Figure 4.5: Forgot password validations screen developed

There is a popup that asks if the user really wants to leave the app by pressing the return button, show in Figure 4.6.


Finnish

Sign In Sign Up

Insira suas credenciais

Email

email@server.com

Por favor use um endereço de email válido

Senha

Senha

Por favor insira uma senha

Tem certeza que deseja sair?

Não Sim

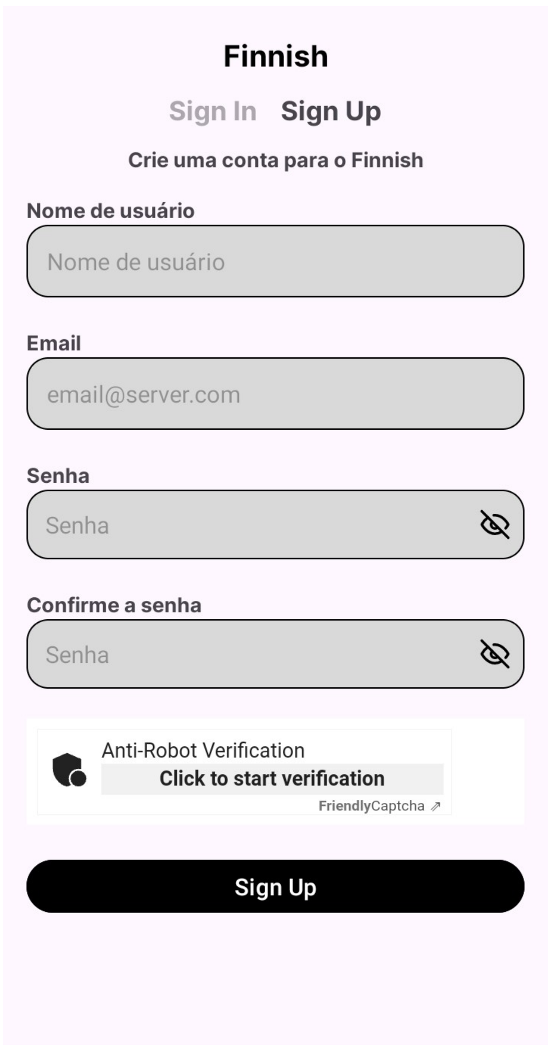
Por favor complete o captcha

Login

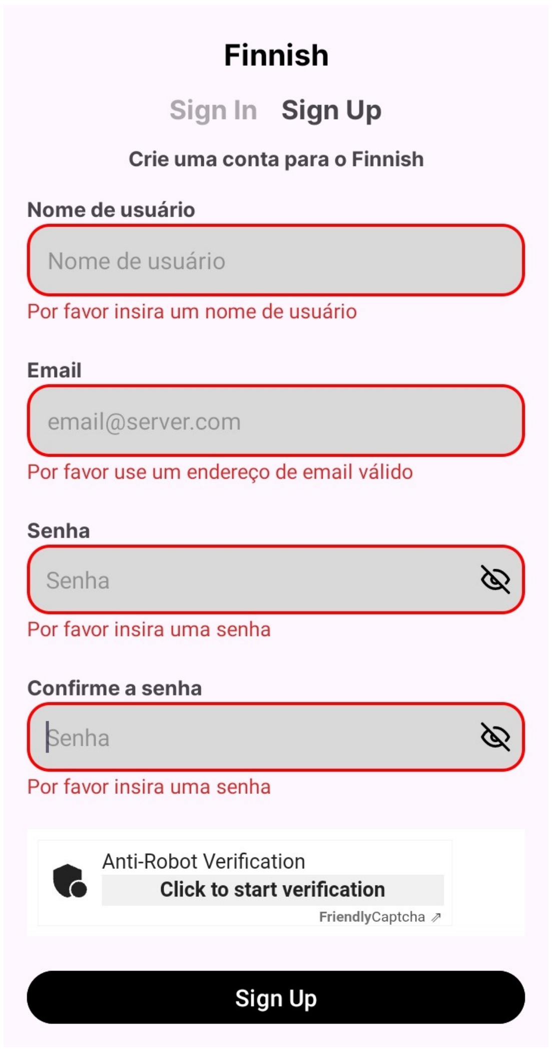
Esqueceu sua senha?

Figure 4.6: Leave App Feature

Returning to the Sign In screen, there is also a field with the text ’Sign Up’, which takes the user to the screen for registering their data. On this screen, there are fields for registering its name, email, password, password confirmation, and a captcha. As well as a button to send data to the server. Empty fields, weak passwords, or non-matching passwords are validated when data is sent. Just like on the Sign In screen, this validation changes the fields, outlining those that are incorrect in red, and showing the respective error message, be it a weak password or an empty field, for example, shown in Figure 4.7.



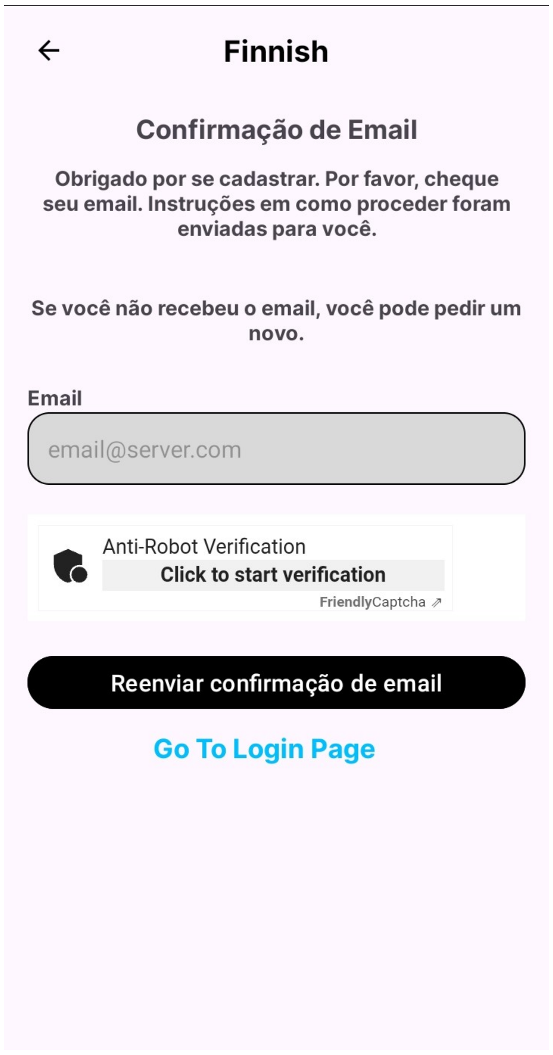
(a) SignUp screen.


Finnish
Sign In
Sign Up
Crie uma conta para o Finnish
Nome de usuário
Nome de usuário
Por favor insira um nome de usuário
Email
email@server.com
Por favor use um endereço de email válido
Senha
Senha
Por favor insira uma senha
Confirme a senha
Senha
Por favor insira uma senha
Anti-Robot Verification
Click to start verification
FriendlyCaptcha
Sign Up

(b) SignUp validations screen.

Figure 4.7: SignUp and validations screen developed

Returning to the Sign In screen, when the data entered is valid, the user may be taken to some screens depending on their status. Firstly, the user must have confirmed their email. This confirmation is sent to the email made in Sign Up when the user registers correctly. If the user has not yet confirmed, the screen to be shown is for entering the confirmation email, with the send button. There is also a button to return to the Sign In screen, and it all can be seen in Figure 4.8.



Finnish

Confirmação de Email

Obrigado por se cadastrar. Por favor, cheque seu email. Instruções em como proceder foram enviadas para você.

Se você não recebeu o email, você pode pedir um novo.

Email

email@server.com

Anti-Robot Verification

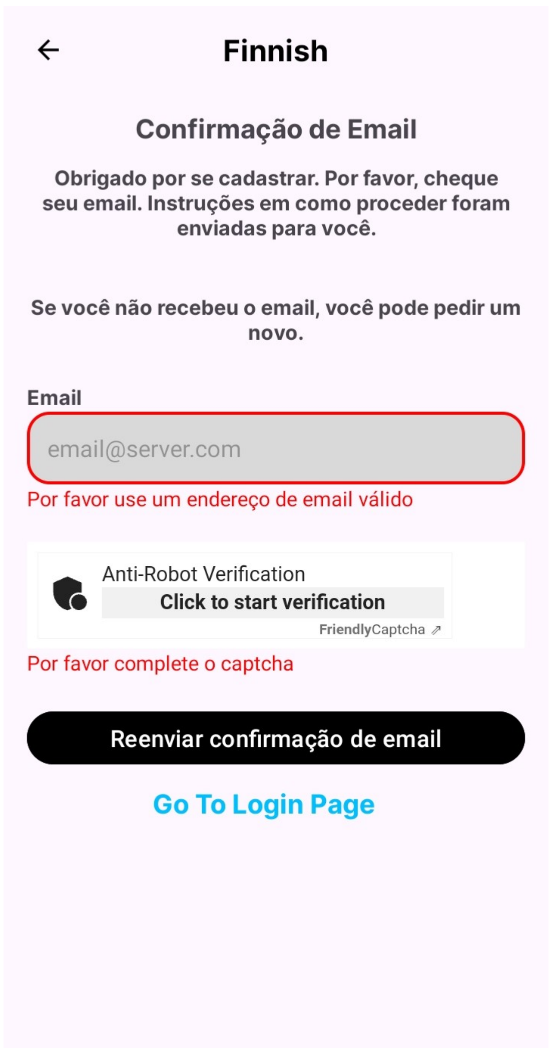
Click to start verification

FriendlyCaptcha

Reenviar confirmação de email

Go To Login Page

(a) Email Confirmation screen.



Finnish

Confirmação de Email

Obrigado por se cadastrar. Por favor, cheque seu email. Instruções em como proceder foram enviadas para você.

Se você não recebeu o email, você pode pedir um novo.

Email

email@server.com

Por favor use um endereço de email válido

Anti-Robot Verification

Click to start verification

FriendlyCaptcha

Por favor complete o captcha

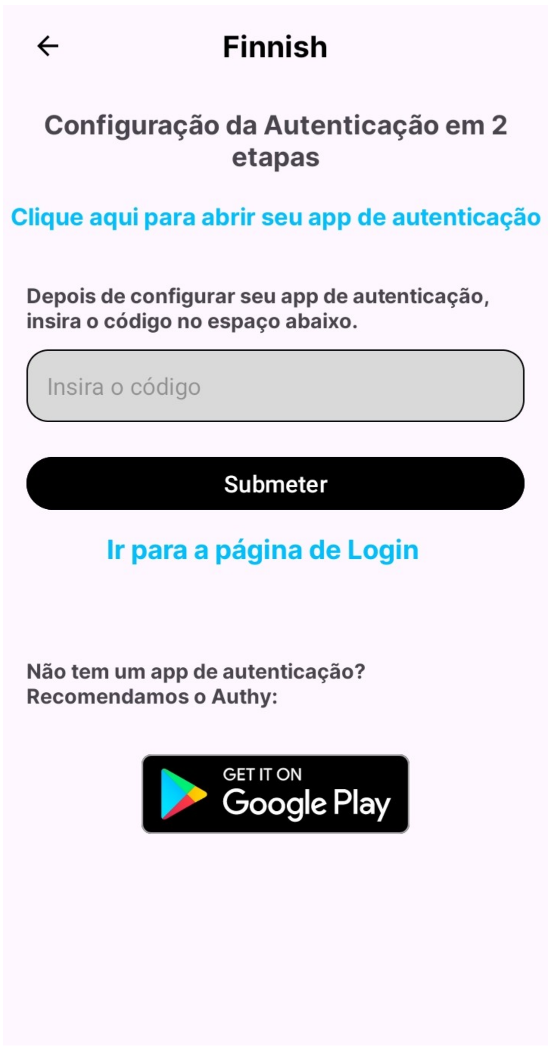
Reenviar confirmação de email

Go To Login Page

(b) Email Confirmation validations screen.

Figure 4.8: SignUp and validations screen developed

Now, if the user's email is confirmed, the screen presented to the user when they Sign In is the second authentication factor screen. In it, the user is presented with a field for inserting the token, a widget that takes the user to the Play Store to download a recommended authentication application, and a link that takes the user to the authentication application that they have installed on their device. After inserting the token, the field will go through the same validation mentioned above, and if the server returns a failure, the field will be outlined in red and the error text will be displayed. This is shown in Figures 4.9 and 4.10.



Finnish

Configuração da Autenticação em 2 etapas

Clique aqui para abrir seu app de autenticação

Depois de configurar seu app de autenticação, insira o código no espaço abaixo.

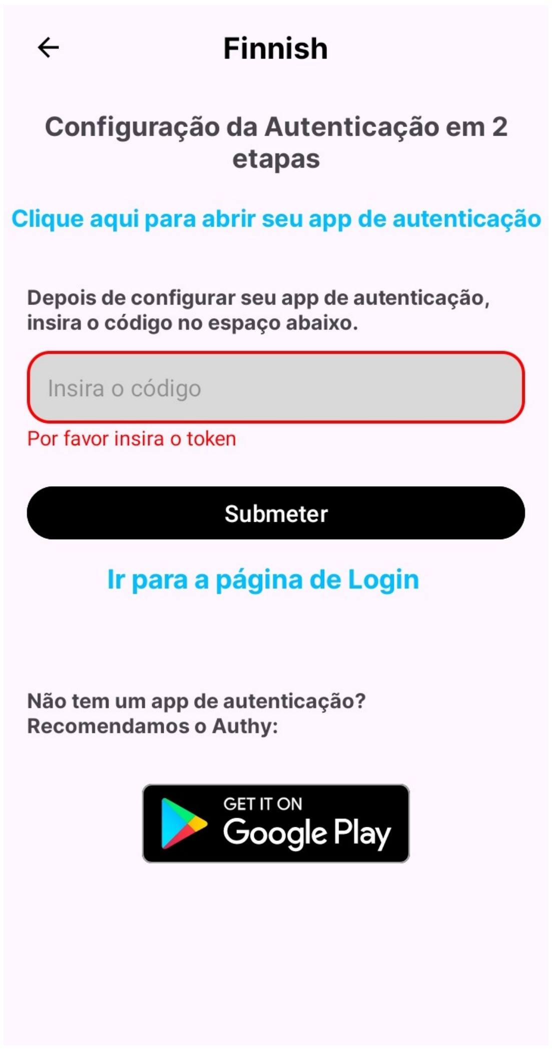
Insira o código

Submeter

Ir para a página de Login

Não tem um app de autenticação? Recomendamos o Authy:

(a) MFA screen.



Finnish

Configuração da Autenticação em 2 etapas

Clique aqui para abrir seu app de autenticação

Depois de configurar seu app de autenticação, insira o código no espaço abaixo.

Insira o código

Por favor insira o token

Submeter

Ir para a página de Login

Não tem um app de autenticação? Recomendamos o Authy:

(b) MFA validations screen.

Figure 4.9: MFA validations screen developed

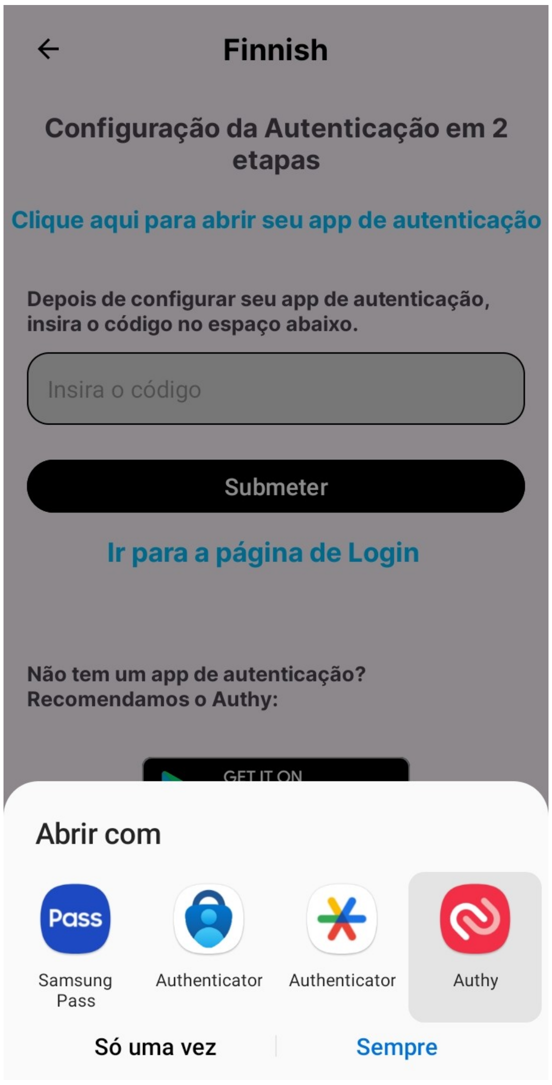
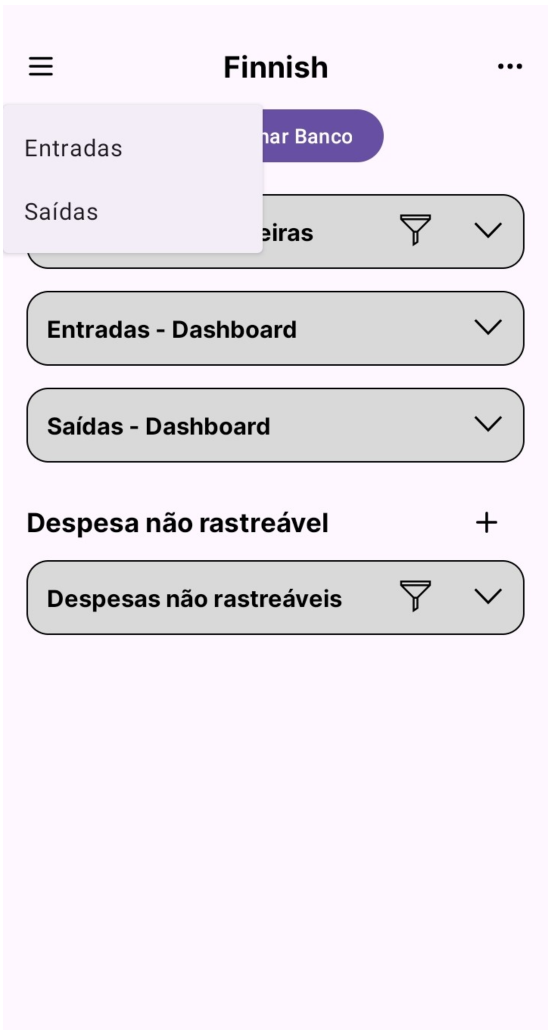
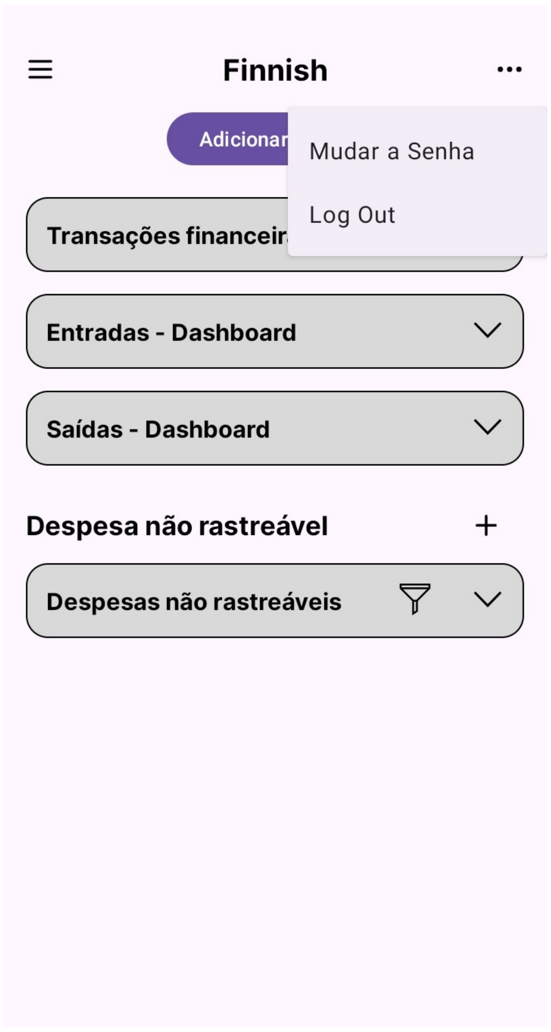

Finnish
Configuração da Autenticação em 2 etapas
Clique aqui para abrir seu app de autenticação
Depois de configurar seu app de autenticação, insira o código no espaço abaixo.
Insira o código
Submeter
Ir para a página de Login
Não tem um app de autenticação? Recomendamos o Authy:
GET IT ON
Abrir com
Samsung Pass
Authenticator
Authenticator
Authy
Só uma vez
Sempre

Figure 4.10: MFA open local authenticator

Finally, if the user has already confirmed their email and has also completed the second authentication factor, they are taken to the Home screen. On this main screen, it will have a sandwich style menu on the left, from which the user can go to the Inputs and Outputs screen. On the right there is another menu, where it can go to the screen to change its password, or log out. If it chooses to log out, a popup is shown if it really wants to log out. For the other options, it is taken to the selected screen, and it can be seen in Figures 4.11, 4.12, 4.13, 4.14, 4.15, and 4.16.

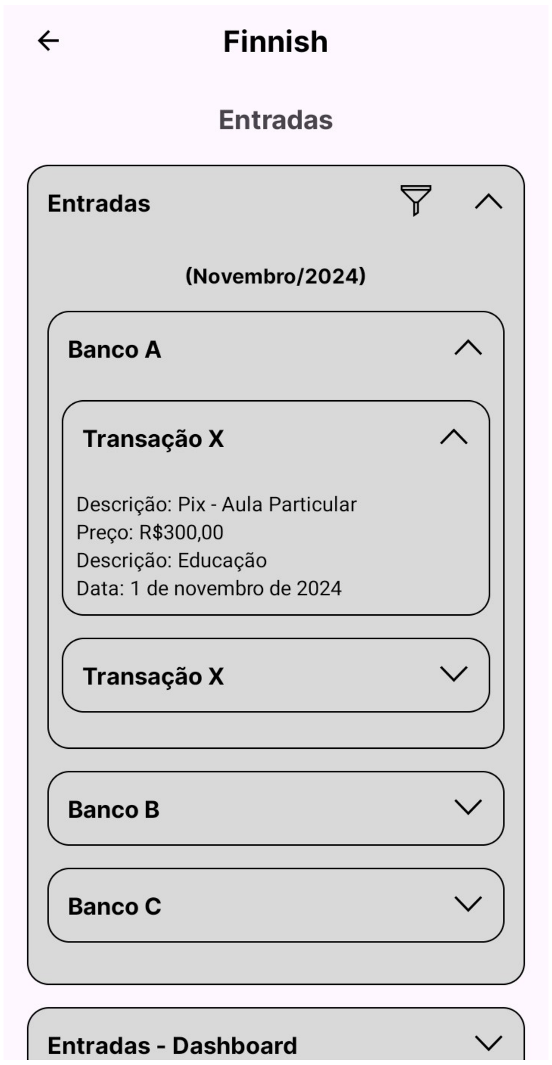


(a) Menu Incomes - Outcomes.



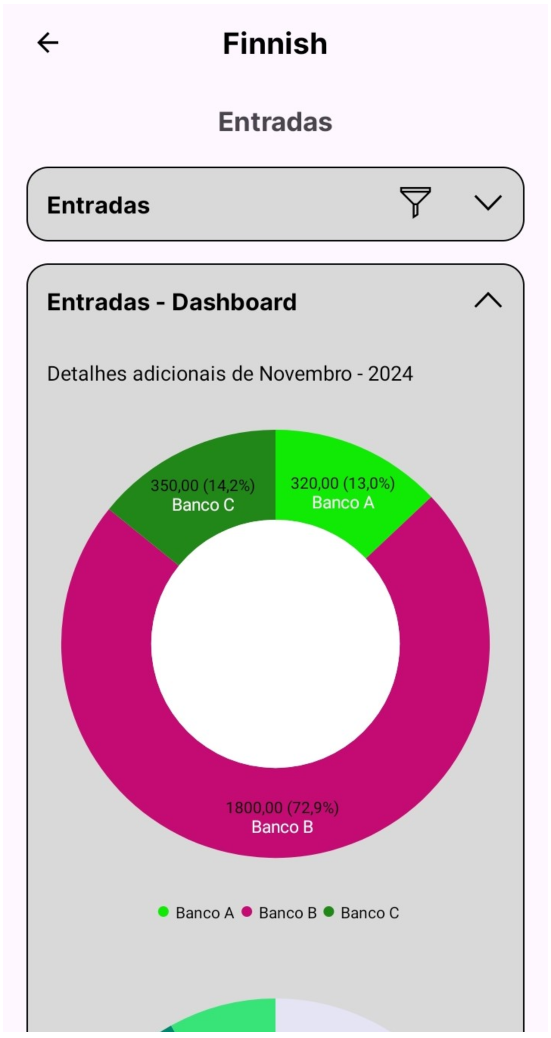
(b) Menu Change Password - Log Out.

Figure 4.11: Menus for changing screens

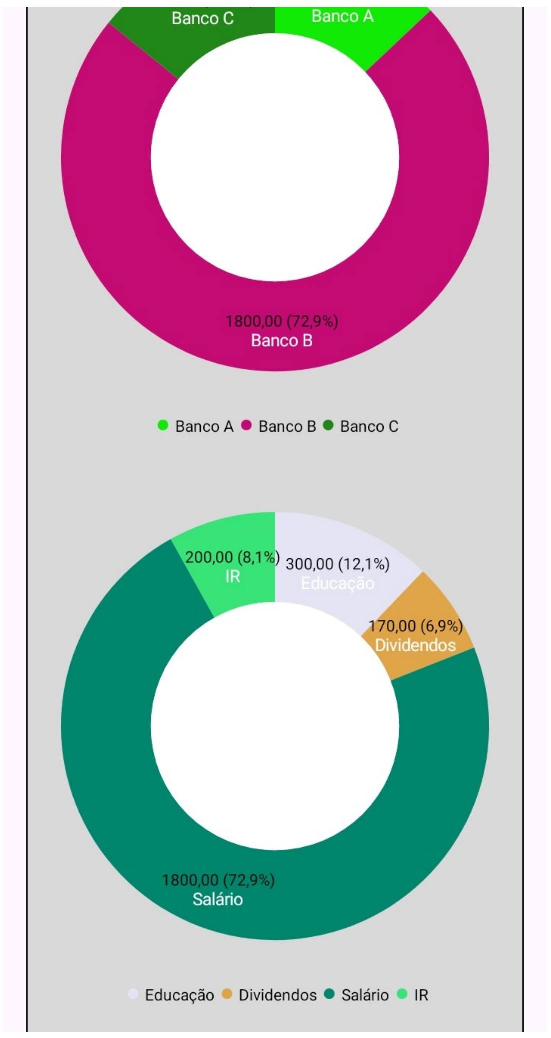


Descrição: Pix - Aula Particular
Preço: R$300,00
Descrição: Educação
Data: 1 de novembro de 2024

Figure 4.12: Incomes Details screen developed

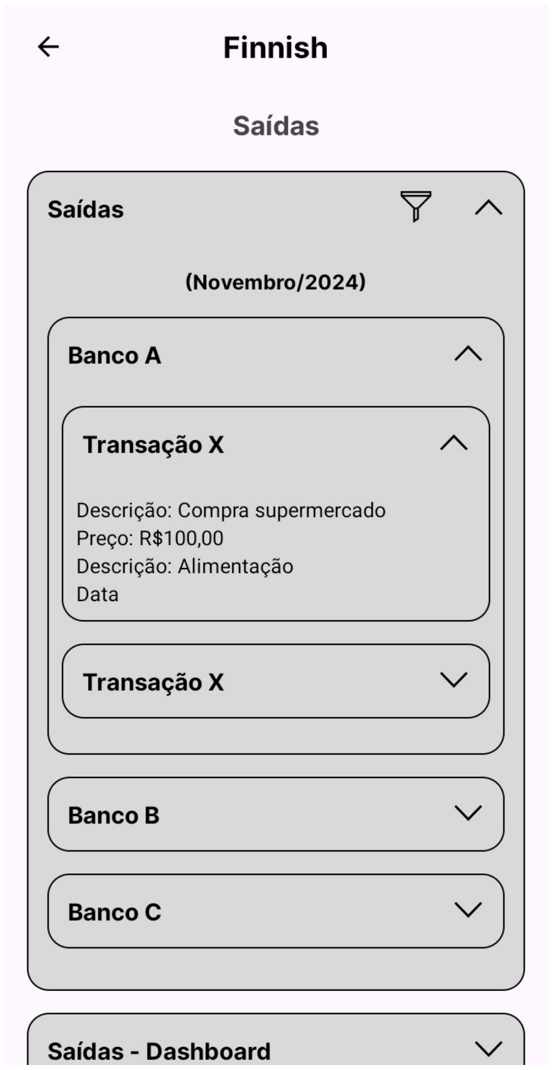


(a) Incomes Dashboard 1 screen.



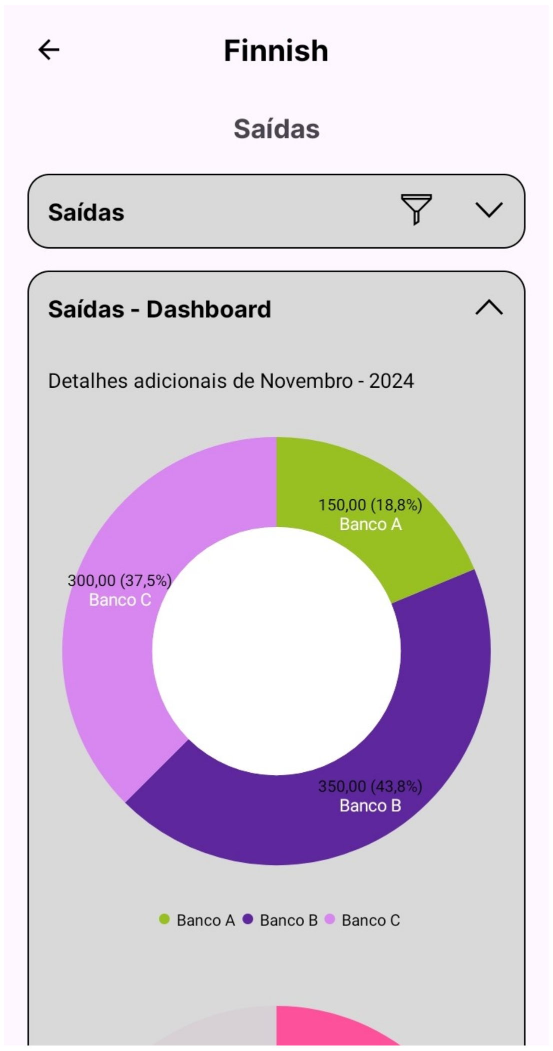
(b) Incomes Dashboard 2 screen.

Figure 4.13: Incomes Dashboards screen developed

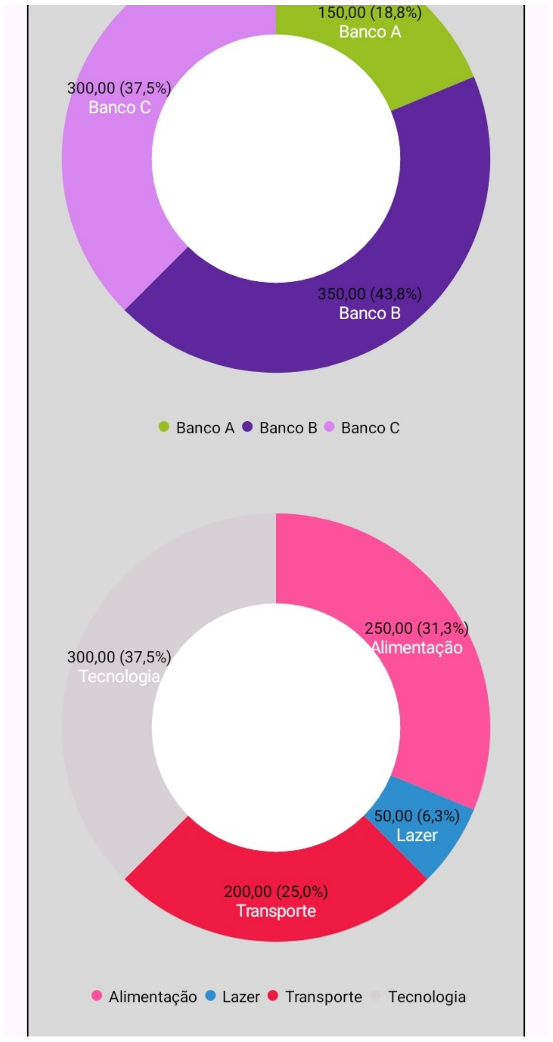


Descrição: Compra supermercado
Preço: R$100,00
Descrição: Alimentação
Data

Figure 4.14: Outcomes Details screen developed

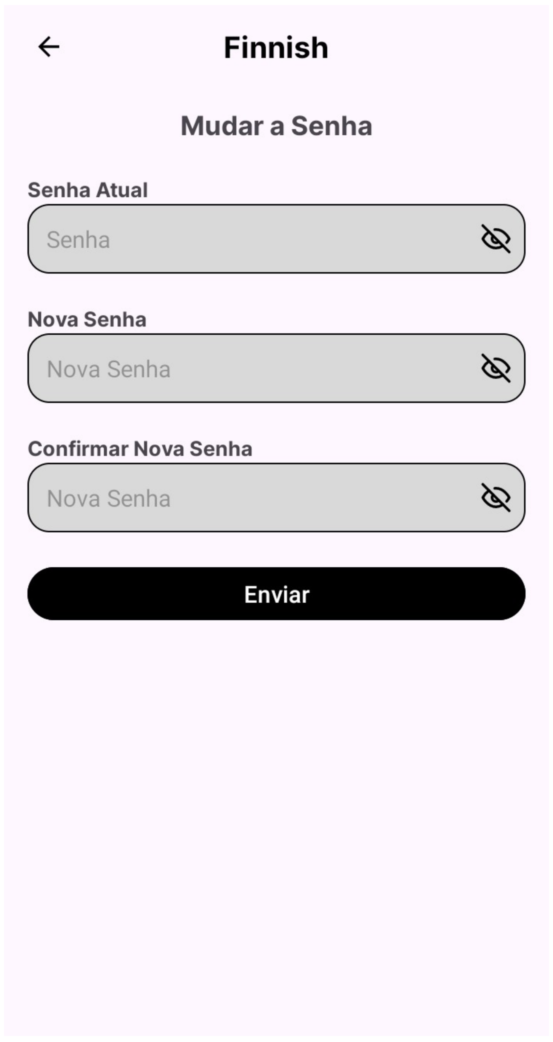


(a) Outcomes Dashboard 1 screen.

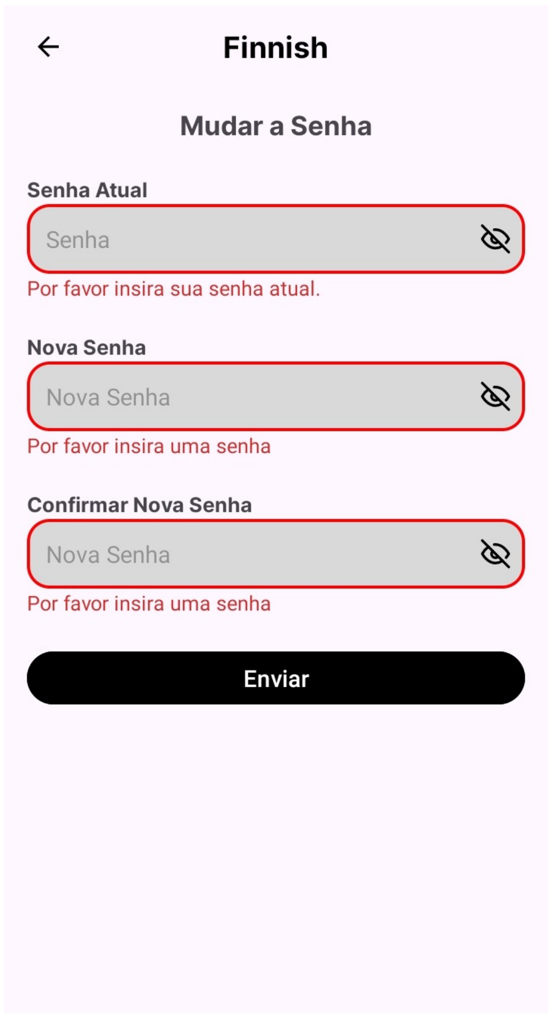


(b) Outcomes Dashboard 2 screen.

Figure 4.15: Outcomes Dashboards screen developed



(a) Change Password screen.



(b) Change Password validations screen.

Figure 4.16: Change Password screen developed

Still on the Home screen, there is a button at the top that directs the user to a Pluggy widget on the screen and they can add a bank of their choice, according to what is available. Once added, the screen loads the new information that will come from the server. Further down, the user will find two areas referring to data from the banks they have already added. It can see through each bank and each bank transaction has a brief description. In this case, this information is arranged in larger retractable Input and Output boxes. Then there are slightly smaller boxes for each bank, also retractable. And then, within each bank, each transaction also is in this format. This screen can be seen in Figure 4.17.

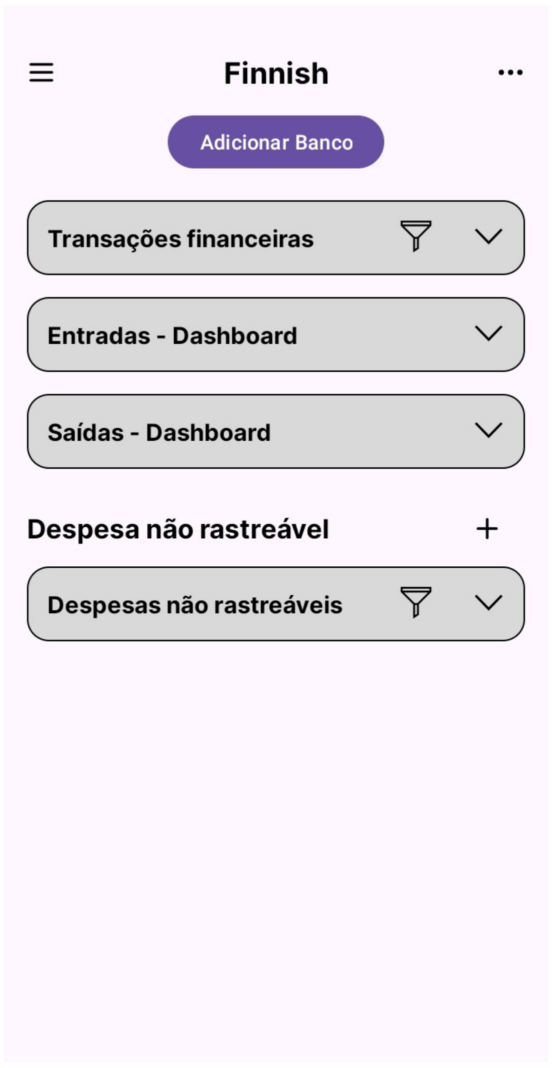
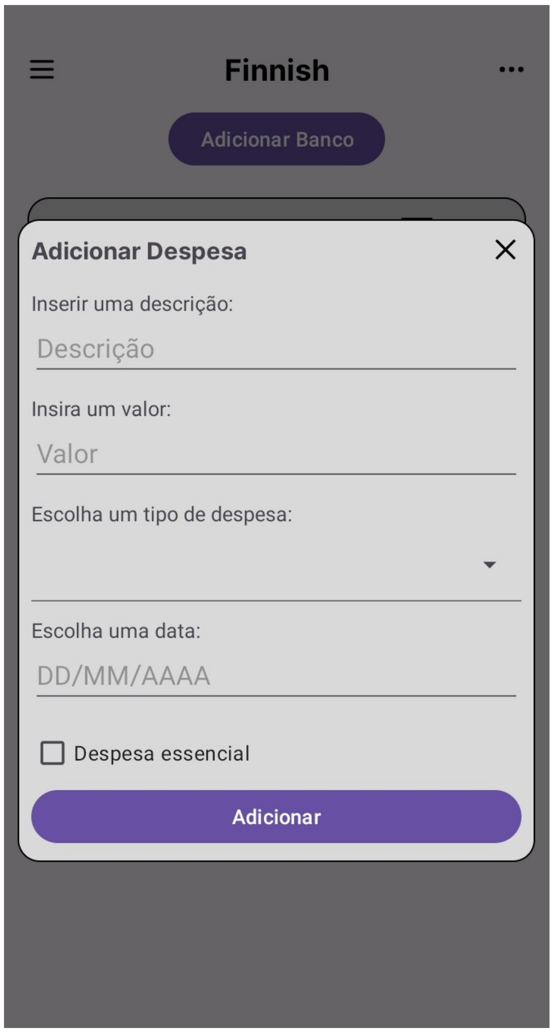


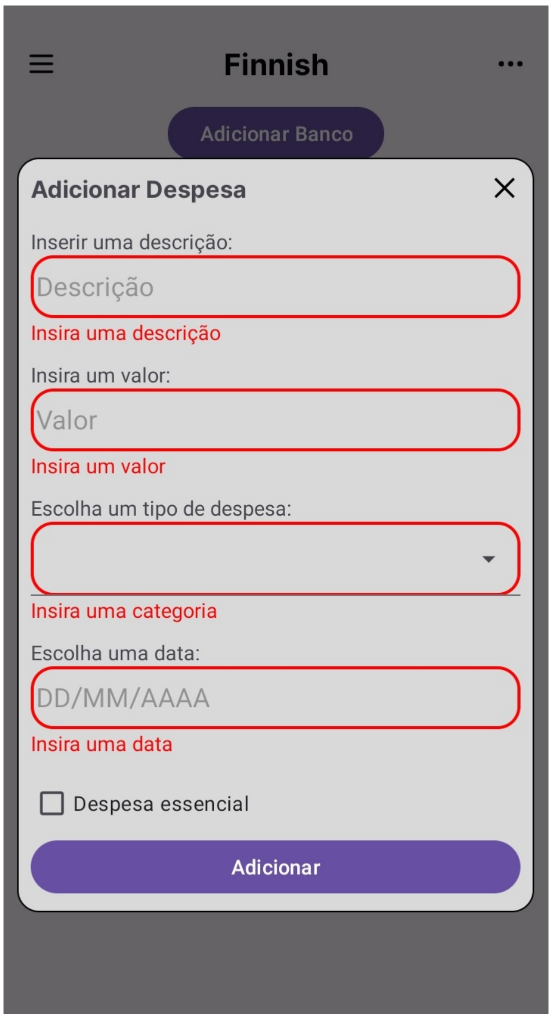
Figure 4.17: Home screen developed

Below, there is a similar arrangement of Input and Output information, but this time arranged in graphs. And, the user will see graphs such as the proportion of Incomes or Outcomes of each bank in relation to the whole, and the proportion according to the categories.

Finally, further down there is a space where the user can enter non-traceable expenses. There is a '+' button, which makes a dialog box appear on the screen, in which the user can enter the expense information and add. Here the fields also undergo validation. This type of expense is arranged in a similar way to the previous ones, however, there is also a pencil icon, which, when pressed, allows the user to edit the specific transaction or even delete it. The fields undergo a validation too, and when the user execute one of this actions, a popup is shown in the screen asking if it really wants to execute. And, a popup related to the server response is shown in the screen showing if the action had success or not. In the larger untrackable expenses box, there is also a filter icon that allows the user to filter these expenses by month. It can be seen in Figures 4.18, 4.19, and 4.20.

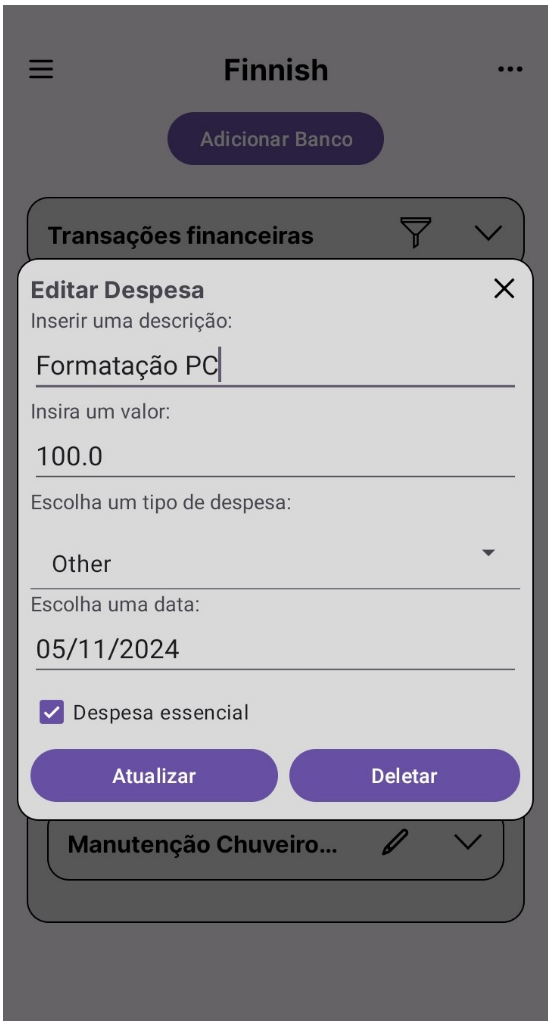


(a) Add untrackable expense dialog.

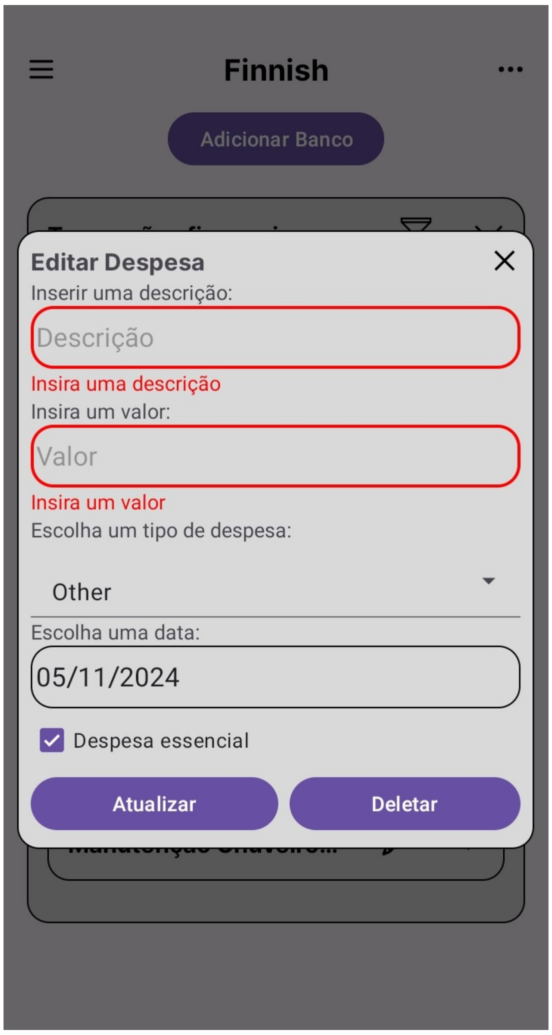


(b) Add untrackable expense dialog validations.

Figure 4.18: Add untrackable expenses dialog

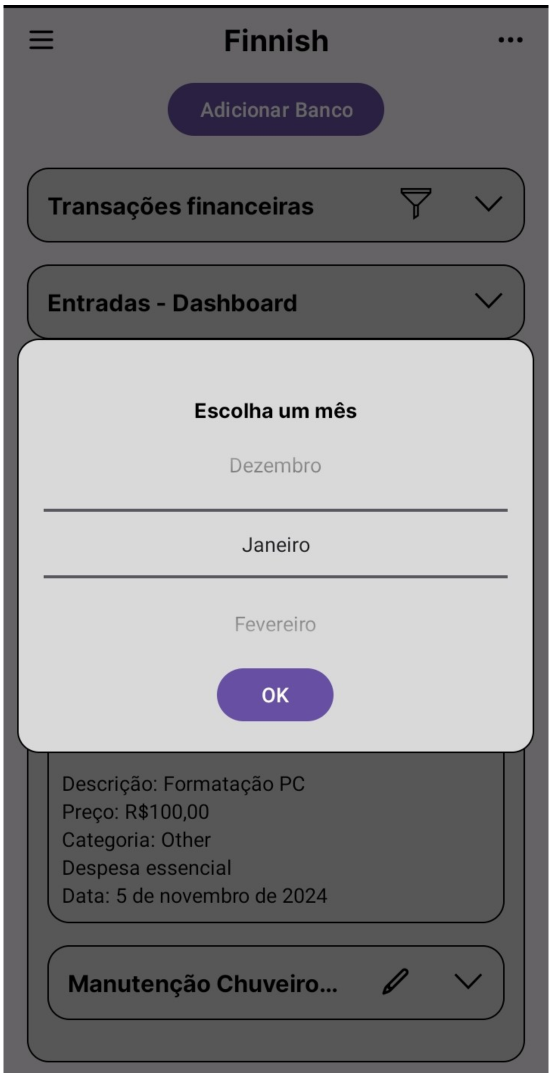


(a) Add untrackable expense dialog.



(b) Edit untrackable expense dialog validations.

Figure 4.19: Edit untrackable expenses dialog



Descrição: Formatação PC
Preço: R$100,00
Categoria: Other
Despesa essencial
Data: 5 de novembro de 2024

Figure 4.20: Filter Month Picker Dialog

4.2.3. Categorization

To develop a model tailored to the project's categorization needs, we started with a comprehensive review of available technologies and research on AI-based categorization models. After evaluating different model architectures, we settled on the XGBoost algorithm due to its effectiveness in handling structured, tabular data, as well as its high performance with relatively low computational requirements. XGBoost's ability to capture complex patterns through gradient boosting made it particularly suitable for our task, where a mix of numerical, categorical, and text-derived features are essential for accurate transaction classification.

Initially, we considered using pre-trained language models, such as DistilBERT—a lighter, faster version of BERT that maintains much of BERT's language understanding. We hypothesized that DistilBERT's deep language representation could be leveraged to classify transaction descriptions directly, given its pre-training on a large corpus. Testing with DistilBERT on simple, fictional transactions like "Bought groceries," "Monthly salary," and "Paid electricity bill" yielded promising accuracy. However, when we tested it with real transaction data, particularly descriptions in Portuguese without consistent word separation (e.g., "ELCSS-EXTRAFARMA"), performance declined significantly. DistilBERT's reliance on tokenized and well-separated text limited its effectiveness with our unstructured, concatenated transaction descriptions.

To address these challenges, we pivoted to a data-driven approach using structured data from the Pluggy API. Pluggy provides comprehensive transaction data, including fields like description, amount, balance, and category, but the raw transaction descriptions varied across financial institutions. This variation presented both a challenge and an opportunity: while we couldn't rely solely on descriptions for categorization, we could integrate other features (like amount, balance, and manually extracted patterns from descriptions) into a model more suited to structured inputs.

For example, Pluggy provides transaction data in the following format:

Listing 4.1: Transaction example

```
{
    "id": "59cccc5c-1c12-497d-8f8c-b0a60d166b80",
    "description": "VENCIMENTO RENDA FIXA - LCI IPCA F 3 A BANCO
    INTER SA",
    "descriptionRaw": null,
    "currencyCode": "BRL",
    "amount": 137.15,
    "amountInAccountCurrency": null,
    "date": "2024-08-09T03:00:00.000Z",
    "category": "Fixed income",
    "categoryId": "03020000",
    "balance": 178.17,
    "accountId": "7e577092-c366-4f8d-a435-49b3025eb15d",
    "providerCode": null,
    "status": "POSTED",
    "paymentData": null,
    "type": "CREDIT",
    "creditCardMetadata": null,
    "acquirerData": null,
    "merchant": null,
    "createdAt": "2024-09-26T01:01:45.781Z",
    "updatedAt": "2024-09-26T01:01:45.781Z"
}
```

After preprocessing the data, we mapped Pluggy's diverse transaction categories to our predefined 12 target categories: 'Restaurants,' 'Shopping,' 'Services,' 'Entertainment,' 'Groceries,' 'Salary,' 'Interest Income,' 'Utilities,' 'Pharmacy,' 'Transfer,' 'Transport,' and 'Others.' This mapping helped standardize the dataset, ensuring consistent labels across transactions from different banks.

We chose XGBoost due to its robustness with tabular data and its ability to capture non-linear relationships through ensemble learning. Unlike language models that require clean, tokenized text, XGBoost is well-suited to structured datasets and can handle both categorical and numerical features efficiently. Additionally, XGBoost allowed us to integrate various features:

- Numerical features like `amount` and `balance`.
- Lexogram-based indicators to detect the presence of specific patterns within transaction descriptions.
- Category mappings that reclassify inconsistent labels into our target categories.

XGBoost's gradient-boosting approach also provided control over model complexity, allowing us to avoid overfitting while achieving high classification accuracy on our mapped categories.

To enhance classification reliability, we introduced a confidence threshold in the prediction script. The model outputs a confidence score for each prediction, and if this score falls below 0.7, we revert to the transaction's original category (provided by Pluggy) for certain predefined categories, such as 'Entertainment' and 'Utilities.' This fallback mechanism ensures that the model doesn't misclassify transactions with uncertain predictions, prioritizing the original data when confidence is low.

The initial model development, testing, and optimization were conducted in a Jupyter Notebook environment. This allowed us to iteratively experiment with feature engineering and hyperparameter tuning. Once we finalized the model, we transitioned it into a production-ready Python script, converting the Jupyter code into standalone .py files. This final version processes JSON transaction inputs, maps categories where needed, and prints both the predicted category and confidence level for seamless deployment.

This structured, multi-stage approach—from initial research and model selection to iterative testing, confidence calibration, and final deployment—allowed us to develop a robust, efficient categorization model that aligns with real-world transaction data intricacies.

To support the model's deployment and provide real-time categorization, a server was implemented using FastAPI, a lightweight and efficient Python framework for building APIs. The server includes two primary endpoints to streamline interaction with the model. The first endpoint receives a list of transactions as input, processes each transaction to predict its category using the XGBoost model, and returns the predicted categories along with confidence levels for each transaction. This functionality allows batch processing of transactions, improving scalability and efficiency. The second endpoint focuses on model maintenance by handling the retraining process. It identifies new transactions added to the dataset, triggers a script to map these transactions to predefined categories, and executes the model retraining pipeline. By integrating these two functionalities, the server ensures the model remains up-to-date while providing accurate and seamless transaction categorization for users.

One significant limitation of the current implementation is that the model operates globally, applying the same categorization logic across all users without accounting for individual preferences or behaviors. While this approach simplifies implementation and leverages shared patterns across the dataset, it overlooks the nuanced, user-specific context that often influences transaction categorization. For instance, a user might frequently categorize recurring transactions like "Netflix" under "Utilities" instead of "Entertainment," or prioritize specific lexicon patterns unique to their spending habits. By treating all users as a homogeneous group, the model risks misclassifying transactions that deviate from generalized patterns, potentially undermining its reliability for personalized financial insights.

4.3. Tests

Since good software was always at the core of this undertaking, writing tests has always been at the tasks board.

When it comes to the server side, the use of *llvm-cov*, unit-testing and integration testing is currently deployed. This tool allows visualization of the areas that are tested and untested, and reports code-coverage.

Development isn't being run in a test-driven philosophy, but instead using Patch Coverage reports in an informative way.

There are future plans of implementing “proptests” (property-based testing) on more sensitive features in the server side.

Also, in the mobile side, testing was a fundamental aspect of the development process. Each component of the application underwent testing to ensure reliability. This included software testing to evaluate functionality. Integration testing was done, which checked the interaction between the mobile app and the backend, as well as with Pluggy’s API.

5 | Final Considerations

This final topic presents the final remarks about this project, while pointing out future perspectives that could improve upon the current state of the project.

5.1. Compromises

5.1.1. Mobile

In the application, the cross-platform development was not possible. Therefore, Apple's own device such as a Macbook was not achieved for iOS development, and more time would be needed in the development of this front, which contains some differences in implementation. Likewise, it was not possible to deliver a more user-friendly interface, due to the learning curve to learn mobile development taking longer than expected, taking away time available to dedicate to the user interface.

On the other hand, it was possible to develop an app that integrates with the server to authenticate, search for financial transactions, and register untraceable expenses. The app also integrates with the Friendly Captcha API, where it receives proof in code to be sent to the server for authentication, and there is integration with the Pluggy API. In the latter case, the integration consisted of the widget that allows the user to choose the bank to search for and display their data.

5.1.2. Backend Server

The first glaring compromise that needed to be made in the server-side is that, since this project was not made as part of an actual banking institution, in Brazil, there is no way to directly connect to the Open Finance network.

This meant that an intermediary had to be used, in this case, Pluggy was that Open Finance provider, but we could have gone with other competitors, like Belvo.

Having this constraint impacted, not only on budget, but, also on external dependencies.

Another compromise that needed to be made was to the robustness of the databases's accountability patterns, both functional and non-functional.

Firstly, having to sync with Pluggy caused the server to have a Webhook sync, as well a time-sync. Medling with these transactions would have been easier with an Accountability-minded database, which we'll mention on Future Perspectives.

5.1.3. Infrastructure

Also, storing these transactions in our databases could maybe profit with some database encryption, in case our infra was compromised.

Another infrastructure compromise was not deploying our own instance of Grafana for observability, and instead going with the Grafana Cloud option.

That isn't ideal, specially long-term, due to the configurability of self-hosting it.

Finally, everything infra-related was on an extreme budget, since our cost-target was $0. Of course, we couldn't reach this target, but we wanted to not have to worry with funding for this project. Although, even with this tight budget, running on the lowest possible hardware configs, our server still met decent performance, due to using good code, and using Rust for the server.

5.1.4. Categorization

The development of the AI model required balancing key compromises, particularly when pre-trained language models like DistilBERT struggled to process unstructured transaction descriptions in Portuguese. To address this, the team opted for XGBoost, which, while less specialized in natural language processing, allowed the integration of numerical features, pattern detection, and category mapping. This compromise resulted in a robust categorization system that effectively handles diverse transaction data, fully integrated with the Pluggy API.

5.2. Future Perspectives

The application has significant potential to evolve into a multiplatform solution, expanding its accessibility and user base. By leveraging Kotlin Multiplatform, it could support both Android and iOS, requiring adaptations to the current Android structure while integrating iOS-specific features. Beyond compatibility, the application can undergo visual enhancements to improve user engagement and usability. New transaction views could provide users with diverse perspectives on their financial data, while deeper AI integration could unlock advanced functionalities such as spending insights, real-time alerts for unusual transactions, and personalized budgeting suggestions. These enhancements would rely on a more dynamic server communication with the categorization layer, enabling a more interactive experience.

The categorization model has room for advancement, particularly in terms of personalization and flexibility. A key improvement involves introducing user-specific customization, allowing the model to learn and adapt to individual spending behaviors. This could be achieved through a user-level customization layer that refines global model predictions based on historical user data, delivering more accurate and tailored insights. Letting users create their own custom categories would make the system much more flexible, allowing them to organize transactions in a way that makes sense for their personal financial needs. Integrating these custom categories into the retraining pipeline would ensure that the model remains relevant as user needs evolve. Additionally, refining the processing of unstructured data, such as noisy or concatenated transaction descriptions, would enhance the system's robustness and accuracy, ensuring reliability across a wide variety of inputs.

On the backend, strengthening the accounting database architecture is a priority to ensure the system can handle future demands. Adopting Tigerbeetle's advanced database technology could provide the necessary foundation for managing sensitive financial data with precision and scalability. Tigerbeetle's innovative testing methods and focus on financial accuracy make it a great option for improving the current architecture, resolving compromises made during the initial development. This change would not only make the system more robust but also set it up for long-term growth and reliability.

Infrastructure improvements are another critical area for future development. Vertical scaling through increased investment in computing resources would support more self-hosted services, reducing reliance on external providers and enhancing system control. Rewriting performance-critical components, such as the categorization model, in Rust instead of Python, would allow for direct embedding into the backend server, improving processing speed and reducing latency. Deploying modern tools to streamline infrastructure management, such as alternatives to the archived "nix-devops," could further optimize the development and deployment process. Although current substitutes are in their infancy, adopting these tools as they mature would significantly enhance the project's operational efficiency.

Together, these advancements form a comprehensive roadmap for the project's future, addressing current limitations while preparing for scalability, flexibility, and user-centric innovation. By pursuing these developments, the project can grow into an advanced financial management tool that meets the needs of a growing and diverse user base.

5.3. Conclusion

The **Finnish** project represents a pivotal step forward in the realm of personal financial management, offering an innovative solution that bridges cutting-edge technology and the growing need for financial literacy and accessibility. By harnessing the power of machine learning and Open Banking APIs, Finnish successfully integrates advanced analytics into an intuitive platform that promises to transform the way users interact with their finances.

Achievements and Potential

Despite the challenges encountered, the project demonstrates remarkable achievements:

1. **Technological Integration**: Seamless incorporation of advanced tools like XG-Boost for transaction categorization, alongside state-of-the-art Rust and Python frameworks, has set a solid foundation for future scalability.
2. **User-Centric Approach**: With features tailored to simplifying financial tasks and improving user experience, Finnish addresses a significant gap in the market for accessible, user-friendly financial tools.
3. **Resilience in Development**: Adopting creative solutions to overcome limitations, such as leveraging the Pluggy API and maintaining a low-cost infrastructure, highlights the team's ability to deliver under constraints.

Broader Impact

Finnish aligns itself with the global shift towards financial inclusion and digital empowerment. By addressing critical pain points such as fragmented financial data, lack of financial education, and cumbersome user interfaces, the project contributes to making personal finance management not only accessible but also enjoyable. Moreover, Finnish's focus on educating users and providing actionable insights fosters long-term financial stability and literacy.

Future Horizons

The possibilities for Finnish's growth are boundless:

- **Personalized Financial Insights**: Future iterations could incorporate user-specific customization, tailoring recommendations to individual spending patterns and habits.
- **Multi-Platform Expansion**: Extending compatibility to iOS and web platforms would broaden the app's reach, making it an indispensable tool for diverse user bases.
- **Advanced AI Features**: Deeper integration of AI for real-time fraud detection, financial health tracking, and automated financial planning could redefine the standards of personal finance applications.

Finnish is not just a project—it is a vision for a smarter, more inclusive financial future. The commitment to innovation, coupled with an unwavering focus on user needs, positions Finnish as a transformative force in personal finance. This journey, though just beginning, already reflects the potential to become a benchmark for technological excellence and user empowerment in the financial technology landscape.

5.4. Live Preview and Open-Source

The Finnish project emphasizes transparency and collaboration, aligning with the open-source ethos. Below are the repositories and live platforms associated with the project:

GitHub Repositories

- **Server (Rust API)**: Explore the backend codebase powering Finnish's financial analytics and transaction processing. `https://github.com/finnish-app/server`
- **Infrastructure (NixOS Server Code)**: Review the server infrastructure and deployment setup. `https://github.com/finnish-app/infra`

Live Platforms

- **Website**: Visit the official Finnish platform to learn more about the project and access its features. `https://fina.center`
- **Dashboard and CI/CD Monitoring**: Access Finnish's live dashboards and continuous integration pipelines for real-time insights into system operations and builds.
  - BuildBot Dashboard: `https://buildbot.fina.center`
  - Grafana Dashboard: `https://grafana.fina.center`

Bibliography

[1] C. Averill. A brief analysis of the apollo guidance computer. *arXiv preprint arXiv:2201.08230*, 2022.

[2] Banco Central do Brasil e Instituições. Estatísticas de meios de pagamentos. `https://www.bcb.gov.br/estatisticas/spbadendos`, 2022. Accessed: 2024-03-14.

[3] Banco Central do Brasil e Instituições. Estatísticas de relacionamentos, cpfs e cnpjs envolvidos. `https://www.bcb.gov.br/acessoinformacao/ccsestatisticas`, 2024. Accessed: 2024-03-14.

[4] J. Blow. Preventing the collapse of civilization. In *Preventing the Collapse of Civilization*, Moscow, 2019. devGAMM. https://www.youtube.com/watch?v=ZSRHeXYDLko.

[5] B. Central. Evolução de meios digitais para realização de transações de pagamento no brasil. *Relatório de Economia Bancária*, 2022.

[6] C. Parnin, E. Helms, C. Atlee, H. Boughton, M. Ghattas, A. Glover, J. Holman, J. Micco, B. Murphy, T. Savor, et al. The top 10 adages in continuous deployment. *IEEE Software*, 34(3):86–95, 2017.

[7] TigerBeetle. Problem of current db solutions. `https://github.com/tigerbeetle/tigerbeetle/blob/main/docs/HISTORY.md`, 2022. Accessed: 2024-03-14.

[8] TigerBeetle. Github repository. `https://github.com/tigerbeetle/tigerbeetle`, 2024. Accessed: 2024-03-14.

[9] Valor - Globo. Bancarização avança, mas uso de conta é limitado. `https://valor.globo.com/financas/noticia/2023/10/27/bancarizacao-avanca-mas-uso-de-conta-e-limitado.ghtml`, 2023. Accessed: 2024-03-14.

List of Figures

List of Tables