Integração de Dados com *Open Finance* para Gestão Financeira Pessoal

Lucas Martins de Barros[1], **Helena Graziottin Ribeiro**[1]

[1]EXATAS - Universidade de Caxias do Sul (UCS)
Caxias do Sul – RS – Brasil

lmbarros@ucs.br, hgrib@ucs.br

Abstract. *The amount of financial institutions combined with the diversity of platforms and applications of these institutions make personal financial control increasingly challenging. Customers with multiple bank accounts, often in different financial institutions, have their financial information very dispersed, making it difficult to consolidate and analyze their own data. The objective of this work is to build a data engineering pipeline for integrating real financial data of an individual in more than one financial institution based on the Open Finance standard. A mobile application prototype was developed to display the integrated data.*

Resumo. *A quantidade de instituições financeiras combinada com a diversidade de plataformas e aplicativos próprios dessas instituições, tornam o controle financeiro pessoal cada vez mais desafiador. Clientes com múltiplas contas bancárias, muitas vezes em instituições financeiras diferentes, têm as suas informações financeiras muito dispersas, dificultando a consolidação e análise dos próprios dados. O objetivo deste trabalho foi construir um pipeline de engenharia de dados para integração de dados financeiros reais de uma pessoa física em mais de uma instituição financeira, usando como base o padrão Open Finance. Foi desenvolvido um protótipo de aplicativo móvel para exibição dos dados integrados.*

1. Introdução

A obtenção e consolidação dos dados financeiros são essenciais para a tomada de decisões de compras, investimentos e planejamento financeiro. Porém, cada instituição têm sua própria estrutura de dados, além de políticas de intercâmbio específicas.

Integrações de sistemas podem ser executadas a partir de diferentes estratégias, como mapeamentos entre os sistemas ou uso de interfaces comuns, e sua implementação pode ocorrer nas diferentes camadas de uma aplicação: de comunicação, de dados ou de apresentação [He and Xu 2014]. Uma das formas de integração através de uma interface comum é considerar o uso de padrões de representação de dados para troca de informações, ou seja, uma linguagem padrão para representação e acesso aos conjuntos de dados. Há padrões definidos por instituições e organismos reconhecidos em diferentes áreas, como a HL7 na área da saúde, *SWIFT* para trocas entre bancos estrangeiros, e o *Open Finance* para dados financeiros no Brasil. No contexto de integração, o uso de padrões de representação de dados é um recurso importante, pois reduz a complexidade do tratamento dos dados ao estabelecer um formato comum para a estrutura das

informações. A existência dessa interface comum permite reduzir esforços na conversão e minimiza erros [Doan et al. 2012].

A engenharia de dados considera o desenvolvimento de estratégias e programas para realizar a integração de dados em um ambiente de forma automatizada [Reis and Housley 2022]. Um processo de engenharia de dados pode ser definido pelo *design* e implementação de um *pipeline* de dados, um fluxo automatizado ou parcialmente automatizado que executa os processamentos de extração, transformação e carga - ETL (*Extract, Transform, Load*), que é uma das metodologias mais populares para gerar conjuntos de dados que alimentam métodos de análise específicos [Nwokeji and Matovu 2021].

O objetivo deste estudo foi prover um ambiente centralizado onde uma pessoa possa acessar e visualizar os seus dados financeiros de múltiplas instituições de forma integrada. Para tal, foi construída uma solução baseada no padrão *Open Finance*, que inclui o desenvolvimento de um *pipeline* de engenharia de dados para integração das informações. Além disso, como prova de conceito, foi desenvolvido um protótipo de aplicação móvel que, além de exibir os dados consolidados, também possibilita a categorização das transações integradas e a análise de gastos por categorias, facilitando a gestão financeira pessoal.

Este artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 apresenta o *Open Finance* como padrão de dados financeiros. A Seção 4 apresenta a *Pluggy* como plataforma de obtenção de dados. Na Seção 5 é exposta a proposta de desenvolvimento. A Seção 6 detalha o processo de desenvolvimento do *pipeline* de engenharia de dados, enquanto a Seção 7 detalha o desenvolvimento do protótipo de aplicação. A exposição dos resultados é feita na Seção 8, enquanto a Seção 9 apresenta a conclusão e os trabalhos futuros.

2. Trabalhos relacionados

Existem trabalhos e aplicativos disponíveis para gestão financeira pessoal que apresentam diferentes abordagens para organizar as finanças de um indivíduo. O trabalho de [Xavier et al. 2024] aborda uma aplicação para controle financeiro, chamada de "Finnish", que implementa uma solução semelhante à desse trabalho, utilizando a mesma plataforma fornecedora de dados. Porém, com foco em categorizações automatizadas, abrindo mão do controle e personalização do usuário, e priorizando a praticidade.

Entre os aplicativos disponíveis no mercado, o *Despezzas*[1] é uma solução de gestão financeira que permite a sincronização de transações através do *Open Finance*. Porém, o aplicativo adota um modelo de assinatura, onde a versão gratuita oferece apenas funcionalidades manuais.

Os aplicativos de algumas instituições financeiras também oferecem funcionalidades de gestão financeira. Por exemplo, a aplicação do banco *Nubank* oferece a possibilidade de categorizar transações realizadas com o cartão de crédito. No entanto, permite somente a visualização de transações realizadas através do próprio banco, sem integrar dados de cartões de outras instituições financeiras, mesmo que estejam conectadas através do *Open Finance*, o que impossibilita a consolidação das informações.

[1]Disponível em: `www.despezzas.com.br`. Último acesso em: 01/04/2025.

	Finnish	Despezzas	Nubank
Versão gratuita	(não publicado)	X	X
Criar transações manualmente	X	X	
Obter transações através do Open Finance	X	(versão paga)	
Categorização manual		X	(apenas transações Nubank)
Categorização automatizada	X	X	(apenas transações Nubank)
Gráficos de gastos	X	X	

Tabela 1. Tabela comparativa trabalhos relacionados

A Tabela 1 apresenta uma comparação das principais funcionalidades de cada um dos trabalhos relacionados. O presente trabalho propõe uma solução que se diferencia por buscar um equilíbrio entre automação e controle do usuário. A obtenção de transações através do *Open Finance* permite um fluxo automatizado de consolidação das transações, porém, com foco em categorização manual, o que possibilita o maior controle do usuário em relação as análises dos dados.

3. Padrões de dados financeiros

Incentivadas pela crise financeira de 2008, seguida por uma crescente desconfiança da população aos bancos tradicionais, as *Fintechs*, novas instituições do setor financeiro que aplicam tecnologia para serviços financeiros, cresceram e se multiplicaram. Essas empresas evoluíram a maneira como as pessoas utilizam serviços financeiros. A ascensão das *Fintechs* promoveu uma cultura de inovação e maior concorrência no setor [Buckley et al. 2016].

Esse cenário impulsionou o conceito de um sistema financeiro aberto, com dados compartilhados entre instituições. O *Open Finance* surgiu no Brasil com esse objetivo, o compartilhamento de dados financeiros de clientes do sistema financeiro entre instituições [BACEN 2024]. Para que o *Open Finance* seja efetivamente implementado, é essencial que os dados das instituições financeiras estejam padronizados de maneira consistente e confiável. Para isso, o Banco central (BACEN) definiu normas e padrões a serem seguidos pelas instituições financeiras para receber e enviar dados.

Para participar do *Open Finance*, enviando ou recebendo dados, é necessário ser uma instituição financeira autorizada pelo BACEN, além de seguir as regras e normas impostas pelo Banco Central. Essas normas impõem que o compartilhamento de dados aconteça de forma segura, garantindo a confidencialidade e a confiabilidade da informação [BACEN 2024].

A documentação técnica, localizada na "Área do Desenvolvedor" no site do *Open Finance*, especifica todos padrões a serem seguidos pelas instituições financeiras participantes. Alguns exemplos de padrões impostos são os campos obrigatórios e opcionais de cada *endpoint* das API's, seus tipos e como formatá-los, convenções de nomenclatura, envio de cabeçalhos e códigos de resposta HTTP [BACEN 2024].

4. Obtenção de dados financeiros

A obtenção dos dados por meio do *Open Finance* é limitada a instituições financeiras, portanto pessoas físicas ou instituições de outras áreas não têm acesso aos dados. Para resolver esse problema, surgiram *fintechs* especializadas em fornecer esses dados do *Open*

Finance para o público que não possui acesso. Essas *fintechs* são instituições financeiras autorizadas pelo BACEN que têm o objetivo de prover os dados em um local centralizado, tanto para pessoas físicas que querem acessar seus próprios dados, quanto para empresas não diretamente do setor financeiro que buscam construir aplicações usando os dados do *Open Finance*. Além de disponibilizar os dados para esse público que não teria acesso, também eliminam a necessidade de extração de cada instituição individualmente, já que elas agrupam os dados das diferentes instituições, fazendo o trabalho mais intenso de integração.

Uma das plataformas, que tem como um dos objetivos fornecer os dados para esse público sem acesso, é a *Pluggy*[2], uma *fintech* paulistana especializada em *Open Finance*. Segundo a própria, eles possuem diversos produtos e funcionalidades, alguns exemplos são a inicialização de pagamentos, que utiliza as contas do cliente cadastradas na *Pluggy* para realizar pagamentos através do *Open Finance* e a portabilidade de crédito, que possibilita o cliente analisar e solicitar a portabilidade de crédito para outras instituições financeiras [Pluggy 2024].

A *Pluggy* também fornece um produto chamado "*MeuPluggy*", que é focado nos próprios clientes do setor financeiro que querem agrupar suas informações de diferentes contas em um ambiente único. O *MeuPluggy* pode ser usado tanto para consulta desses dados, através da API da instituição, quanto para visualização desses dados, através do próprio produto [Pluggy 2024].

É importante ressaltar que, embora esses dados não sejam acessados diretamente pelas API's do *Open Finance* das instituições, eles ainda têm origem nessas fontes, mantendo grande semelhança com os campos definidos pelo BACEN. No entanto, a *Pluggy* também possui integração direta com algumas instituições que ainda não aderiram ao *Open Finance*. As diferenças em relação aos campos da *Pluggy* com os do *Open Finance* se dá pelas transformações feitas pela *Pluggy*, que podem ser necessárias para, por exemplo, integrar campos "extras" de instituições do *Open Finance*, como previsto pelo princípio de extensibilidade do BACEN, ou até mesmo acomodar particularidades de instituições não participantes do *Open Finance* [Pluggy 2024, BACEN 2024].

5. Proposta

A proposta de desenvolvimento consiste na implementação de um pipeline de integração de dados financeiros e um protótipo de aplicação para acesso e gerenciamento dessas informações. Os dados de transações de conta corrente e cartão de crédito foram extraídos de três instituições financeiras: *Nubank*, XP e Bradesco. O protótipo de aplicação permite a visualização desses dados e oferece funcionalidades para categorização e análise dos gastos por categorias, que são definidas pelo próprio usuário.

O desenvolvimento se beneficia do *Open Finance* como padrão adotado pelas instituições para padronização dos dados financeiros. A obtenção dos dados ocorre por meio da *Pluggy*, que permite os acesso aos dados do *Open Finance* através do "MeuPluggy", em uma API única que consolida os dados das três instituições.

A solução inclui uma API desenvolvida em *Python*, utilizando *Flask*, responsável por executar o pipeline de ETL e intermediar a comunicação entre o protótipo de aplicação

[2]Disponível em: `www.pluggy.ai`. Último acesso em: 01/04/2025.

e o banco de dados em nuvem. O protótipo foi desenvolvido em *Swift*, com um banco de dados local para armazenamento temporário e toda a comunicação com o banco de dados em nuvem através da API.

Os serviços em nuvem utilizados incluem o *Azure SQL Database*, onde são armazenados os dados extraídos pelo pipeline de ETL e os dados gerenciadas pelo protótipo, assim como o *Azure Web App*, serviço que hospeda a API.

6. Desenvolvimento da Integração

O desenvolvimento da integração foi separado em três etapas sequenciais: a preparação da *Pluggy* (Seção 6.1), desenvolvimento da API (Seção 6.2) e desenvolvimento do *pipeline* de ETL (Seção 6.3).

6.1. Preparação *Pluggy*

Conforme representado pelo diagrama da Figura 1, o processo para preparar o ambiente responsável por disponibilizar os dados foi:

1-2. **Adicionar conta:** que consiste em conectar uma conta bancária que irá compartilhar os dados através do *Open Finance*. O fluxo para conectar a conta pode variar dependendo da instituição. Foram adicionadas três contas bancárias: "Nubank", "Bradesco" e "XP Banking". O fluxo de autenticação dessas três contas foram iguais, primeiramente foi solicitado o CPF do titular e, após a inserção do CPF, é mostrado um código QR que direciona para o aplicativo da instituição, onde é realizada a autorização do compartilhamento dos dados com a *Pluggy*.

3. **Criar time:** o time da *Pluggy* se trata de um grupo administrado por um ou mais membros que contém as aplicações e conectores.

4. **Adicionar conector:** conectores são responsáveis por fazer a conexão entre os dados das instituições com o time. O conector utilizado foi o "*MeuPluggy*". A seleção de conector ainda não garante o acesso dos dados para o time, serve somente para especificar quais conectores estarão disponíveis para as aplicações do time.

5. **Criar aplicação:** a aplicação é responsável pelo acesso aos dados, a ela são atribuídas as contas dos conectores permitidos no time da aplicação. Além de ser responsável pela permissão de acesso às contas, também disponibiliza uma credencial (*Client ID* e *Client Secret*), que é utilizada para a extração dos dados através da API da *Pluggy*.

6. **Acesso da aplicação ao "*MeuPluggy*":** através do *preview* da aplicação disponibilizada pela *Pluggy*, é possível adicionar as contas daqueles conectores permitidos no time da aplicação. Neste caso, foi adicionada a conta do "*MeuPluggy*", herdando todas as contas bancárias que foram adicionadas nas etapas 1-2.

Com todas as etapas de preparação da *Pluggy*, o resultado é uma credencial de aplicação (*Client ID* e *Client Secret*) que tem acesso às três contas bancárias diferentes. Com essa credencial é possível realizar requisições na API da *Pluggy* para acessar os dados, padronizados pelo *Open Finance*, dessas contas.

6.2. Desenvolvimento da API

A API foi desenvolvida em *Python* utilizando o *framework Flask* e hospedada em um *Azure Web App*. Os *endpoints* implementados permitem o gerenciamento das entidades de categorias e transações.

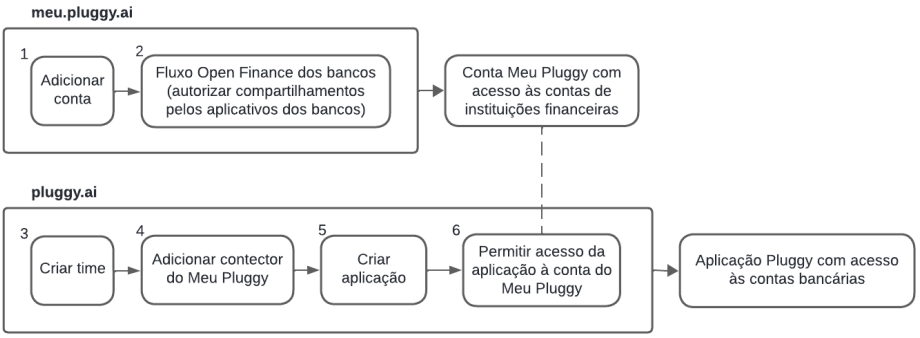

Figura 1. Fluxograma *Pluggy*

A entidade de categoria possui operações completas de CRUD (*Create*, *Read*, *Update*, *Delete*). Já a entidade de transação não permite criação nem exclusão, sendo possível modificar apenas os atributos gerenciados pelo protótipo. Além disso, a API conta com um *endpoint* para acionar o *pipeline* de ETL e outro para fornecer todos os dados necessários para atualização do banco de dados local do protótipo de aplicação, incluindo tanto categorias quanto transações.

6.3. Desenvolvimento do *pipeline* de dados

O *pipeline* de dados, como exposto na Seção 6.2, é acionado através do *endpoint* da API. Toda vez que é realizada uma requisição para o *endpoint*, são instanciados três objetos: um para extração, que será mostrado na Seção 6.3.1; para transformação, da Seção 6.3.2; e para carga, Seção 6.3.3.

6.3.1. Extração

Todos dados extraídos da solução são provenientes da *Pluggy*, através de requisições no *endpoint* "*/transactions*". Os dados das transações são retornados em formato de *JSON*, seguindo o esquema da Tabela 2, que mostra o nome do campo, seu tipo e uma descrição. A saída da etapa de extração é uma lista com as transações, em formato de dicionários, de todas as contas configuradas, tanto cartão de crédito quanto conta corrente das três contas.

A ideia inicial da extração seria realizar uma primeira carga completa e as seguintes serem apenas incrementais, interrompendo a extração na primeira transação com data mais antiga que a última carga realizada e fazendo o *merge* pelo "Id" da transação. Porém, durante o desenvolvimento, foi observado que algumas transações estavam sendo duplicadas. Apesar de conterem Id's e descrições diferentes, tratavam-se de uma única operação, que aparecia somente uma vez no extrato. De acordo com a própria *Pluggy*, em sua documentação, as transações podem ser deletadas e substituídas com um novo Id caso algum campo, como a descrição, seja modificado pela instituição financeira.

Por esse motivo, decidiu-se não realizar extrações incrementais, e sim sempre realizar a extração de todas as transações disponíveis na *Pluggy*, dispensando assim a necessidade de controlar as transações que foram deletadas e re-inseridas pela *Pluggy*. Outra possibilidade seria receber um gatilho vindo da *Pluggy* para excluir as transações

Campo	Tipo	Descrição
id	string	ID da transação, criada pela Pluggy.
date	date	Data da transação, ISO8601 (UTC time).
description	string	Descrição fornecida pela instituição financeira.
descriptionRaw	string	Opcional. Descrição crua fornecida pela instituição financeira.
amount	number	Valor da transação.
amountInAccountCurrency	number	Opcional. Valor da transação na moeda da conta.
balance	number	Opcional. Saldo da conta após a transação.
currencyCode	string	Moeda da transação.
category	string	Somente para assinaturas Pro. Categoria da transação provida pela Pluggy.
accountId	string	ID da conta bancária associada à transação.
providerCode	string	Opcional. Código da transação do provedor.
status	string	Status, “PENDING” ou “POSTED”.
type	string	Tipo, “DEBIT” ou “CREDIT”.
paymentData	object	Opcional. Dados relacionados ao pagamento/transferência.
creditCardMetadata	object	Opcional. Dados relacionados à transação do cartão de crédito.
merchant	object	Opcional. Dados relacionados ao comerciante associado com a transação.
operationType	string	Opcional. Tipo da operação, “TED”, “DOC”, “PIX”...
providerId	string	Opcional. Identificador do provedor da transação.

Tabela 2. Esquema de dados original

que foram excluídas da base de dados (que trocaram de Id), porém, foi definido sempre realizar a extração completa, devido ao baixo volume de dados, de menos de duas mil transações combinando as três instituições.

6.3.2. Transformação

A classe de transformação recebe os dados de transações da *Pluggy* de forma bruta, em formato de lista de dicionários, e os transforma em um *DataFrame* da biblioteca *Pandas*. O esquema final dos dados, após as transformações, pode ser visto na Tabela 3, que apresenta o nome do campo original da *Pluggy*, o nome após renomeação e uma descrição das transformações aplicadas.

Pluggy	Solução	Descrição
id	id	
description	ds_transacao	Transformada. Para transações Pix originadas do Nubank, concatena o “documentNumber” do pagador ou recebedor, parâmetro do objeto “paymentData”, para auxiliar na identificação de transações.
type	cod_tipo	
	cod_metodo	Coluna calculada. “Recebimento” caso cod_tipo seja “CREDIT”; “Pix” quando ds_transacao contém a palavra “Pix”; “Crédito” quando creditCardMetaData contenha valor; “Débito” caso não tenha se encaixado em nenhuma regra anterior.
date	dt_transacao	
amount	vl_transacao	Transformada. Multiplica por -1 caso o cod_tipo seja “DEBIT”.
accountId	cod_conta	
	bool_novo	Informação gerenciada pelo protótipo. Padrão: “TRUE”.
	bool_ignorar	Informação gerenciada pelo protótipo. Padrão: “FALSE”.
	id_categoria	Informação gerenciada pelo protótipo. Padrão: “Sem Categoria”.
	etl_date	Data de inserção da transação.

Tabela 3. Esquema de dados após transformações

Após as transformações, as linhas são ordenadas de forma decrescente pela data de transação. Com todas as transformações prontas, a saída desta etapa é um *DataFrame* com os dados sanitizados e prontos para serem carregados no banco de dados da solução.

6.3.3. Carga

Como mencionado na Seção 6.3.1, todas as transações são extraídas em todas execuções do *pipeline* da solução. Desta forma, para realizar a carga dos novos dados não é possível realizar somente uma inserção. Também não é possível deletar todos os dados já carregados no banco e substituir pelos novos, já que os atributos gerenciados pelo protótipo, como as categorias, seriam perdidos.

Para resolver o problema, é realizado um processo de *merge* personalizado. Esse processo pode ser dividido em três etapas principais:

1. Verifica se existe algum registro no banco de dados que não existe nos novos dados extraídos da *Pluggy*, comparando pelo "Id".
2. Caso exista algum registro que se encaixe nesse caso, realiza uma busca por semelhança entre as novas transações da extração (as transações vindas da *Pluggy* que ainda não existem no banco de dados), procurando uma transação equivalente, que tenha o mesmo valor, data, hora e conta, porém com um Id diferente.
    (a) Se existir uma transação equivalente, realiza um *merge*, substituindo os atributos (incluindo o "Id") da transação do banco de dados pelos atributos da nova transação. Os únicos atributos que não são substituídos pelos novos são os atributos gerenciados pelo protótipo (que não são provenientes da extração dos dados).
    (b) Se não existir nenhuma transação equivalente entre as extraídas pela *Pluggy*, a transação do banco de dados é deletada.
3. Os registros novos, que não se encaixam em uma atualização de atributos, são adicionados na tabela. Enquanto os registros vindos da *Pluggy*, que já existem no banco de dados (que possuem um Id que já consta na tabela do banco) não são inseridos.

7. Desenvolvimento do Protótipo

O protótipo de aplicação móvel foi desenvolvido em *Swift*, utilizando *SwiftUI* para a construção da interface. O protótipo de aplicativo oferece funcionalidades que possibilitam a categorização das transações e a marcação de transações irrelevantes como "ignoradas", para situações como as de operações de transferências entre contas do próprio usuário, que teriam seu valor contabilizado tanto como despesa quanto receita. Ademais, as transações que ainda não foram categorizadas são agrupadas em uma seção de "Novas transações", para fácil identificação do que precisa ser categorizado.

Após a autenticação via API, o aplicativo realiza uma carga completa dos dados, sincronizando as informações do banco de dados em nuvem com o banco de dados local. As alterações realizadas pelo usuário, como a categorização e a edição de categorias são sincronizadas com a API de forma assíncrona, durante a execução do aplicativo.

8. Resultados

O pipeline de dados funcionou conforme o esperado, permitindo a extração completa das transações, tanto de cartão de crédito quanto de conta corrente, das três instituições financeiras integradas. Para validação, os dados extraídos foram comparados com as faturas de cartão de crédito e os extratos bancários gerados nos aplicativos das instituições em

um mês específico. Embora as descrições das transações não sejam sempre idênticas aos extratos, elas contêm a mesma informação essencial, permitindo a identificação de cada movimentação.

O protótipo foi estruturado em três telas principais, representadas na Figura 2.

- **Relatório:** exibe um resumo mensal das transações, incluindo a soma de receitas e despesas (desconsiderando as transações ignoradas), controle das transações "novas" (ainda não categorizadas), análise de despesas por categoria e uma listagem completa das transações do mês.
- **Categorização:** permite que o usuário percorra cada transação nova, atribuindo uma categoria e marcando-a como oculta, se necessário.
- **Gerenciamento de categorias:** possibilita a criação, edição e remoção de categorias personalizadas pelo usuário.

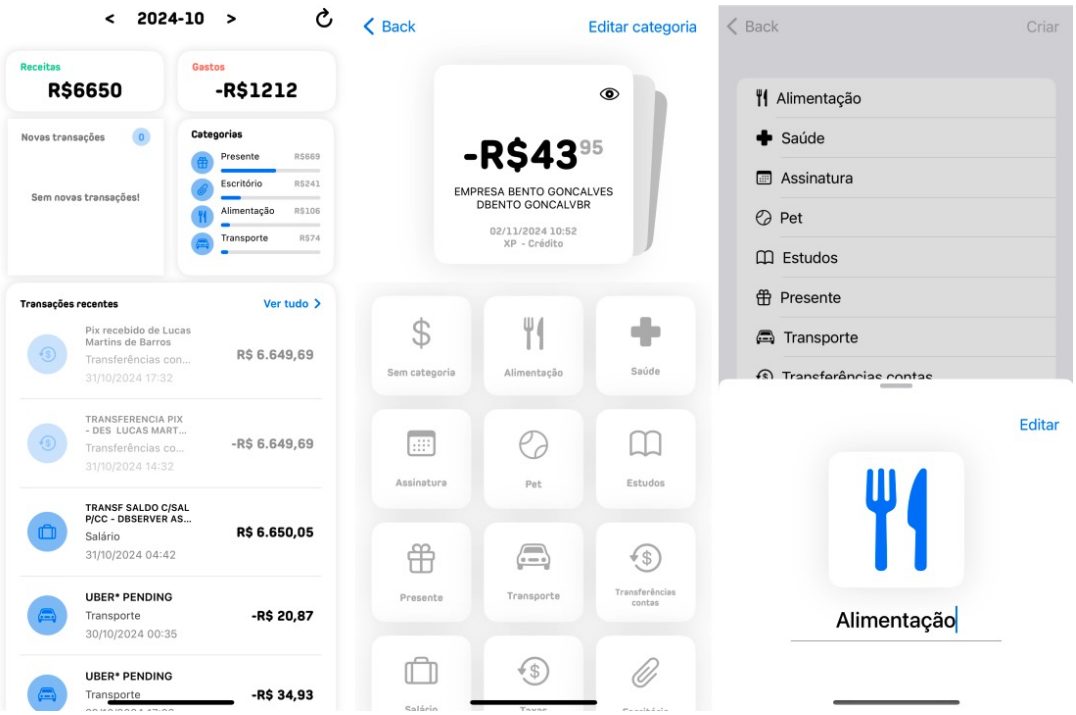

Figura 2. Telas do protótipo

9. Conclusão e Trabalhos Futuros

O padrão *Open Finance* é um grande facilitador para a integração dos dados, definindo uma interface para obter os dados diretamente das instituições, e definindo um padrão entre as instituições financeiras, o que permite reduzir a quantidade de transformações necessárias para integração das informações. Porém, mesmo que seja um padrão, existem diferenças nas implementações do padrão nas diferentes instituições, o que torna necessário o desenvolvimento de um adaptador específico para cada instituição. Observou-se que a padronização entre as instituições é limitada aos nomes dos campos e seus tipos. Os campos de texto livre, como a descrição da transação, não tem seu conteúdo padronizado. Isso fez com que se tornasse necessário, por exemplo, realizar uma transformação para padronizar a descrição das transações de um dos bancos. Outra dificuldade durante a

integração dos dados foi com o *Id* das transações, vindo da *Pluggy*, que pode ser alterado de acordo com mudanças de outros campos, como a descrição. Esse problema foi contornado realizando um processo de *merge* personalizado que se assemelha à uma carga completa.

A integração de dados financeiros provenientes de múltiplas instituições permite a criação de soluções que vão além do controle financeiro pessoal baseado na categorização de despesas. Esse tipo de *pipeline* pode ser utilizado para outras abordagens, como a análise de fluxos de caixa, previsões financeiras e monitoramento de padrões de consumo ao longo do tempo. Além disso, sua aplicação não se restringe a indivíduos, podendo ser adaptada para o controle financeiro de empresas, auxiliando na consolidação de contas, no acompanhamento de receitas e despesas e no planejamento financeiro de longo prazo.

Entre os possíveis trabalhos futuros, destacamos a expansão da solução para um modelo multiusuário, que iria exigir uma versão paga da *Pluggy* e um fluxo de autenticação na *Pluggy* diretamente pelo protótipo. Além disso, seria possível monitorar, além de transações, informações como saldos de contas e dados sobre investimentos, que podem ser extraídas através da *Pluggy* utilizando o mesmo fluxo já implementado.

Referências

[BACEN 2024] BACEN (2024). Open finance. `https://openfinancebrasil.org.br/` [Acesso em: 19 de abr. de 2024.].

[Buckley et al. 2016] Buckley, R., Arner, D., and Barberis, J. (2016). The evolution of fintech: A new post-crisis paradigm? *Georgetown Journal of International Law*, 47:1271–1319.

[Doan et al. 2012] Doan, A., Halevy, A., and Ives, Z. (2012). *Principles of Data Integration*. Morgan Kaufmann.

[He and Xu 2014] He, W. and Xu, L. D. (2014). Integration of distributed enterprise applications: A survey. *IEEE Transactions on Industrial Informatics*, 10(1):35–42.

[Nwokeji and Matovu 2021] Nwokeji, J. and Matovu, R. (2021). *A Systematic Literature Review on Big Data Extraction, Transformation and Loading (ETL)*.

[Pluggy 2024] Pluggy (2024). Pluggy. `www.pluggy.ai/` [Acesso em: 18 de mai. de 2024].

[Reis and Housley 2022] Reis, J. and Housley, M. (2022). *Fundamentals of Data Engineering: Plan and Build Robust Data Systems*. O'Reilly.

[Xavier et al. 2024] Xavier, C. A. G., de Almeida, D. A., and Auler, N. V. (2024). Finnish smart personal financial management by leveraging machine learning and the open banking api. USP - Trabalho de Conclusão de Curso.