**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**
**DO RIO GRANDE DO SUL**
**CAMPUS RESTINGA**

# SISTEMA DE GERENCIAMENTO DE DESPESAS PESSOAIS

**BRUNO PADILHA DA CUNHA**

**Porto Alegre**
**2025**

**BRUNO PADILHA DA CUNHA**

# SISTEMA DE GERENCIAMENTO DE DESPESAS PESSOAIS: IFRS - RESTINGA

Trabalho de Conclusão de Curso apresentado junto ao Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal Educação, Ciência e Tecnologia do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Ricardo Luis dos Santos

**Porto Alegre**
**2025**

**BRUNO PADILHA DA CUNHA**

# SISTEMA DE GERENCIAMENTO DE DESPESAS PESSOAIS: IFRS - RESTINGA

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Dr. Ricardo Luis dos Santos

Aprovado em MÊS, ANO.

_____

Ricardo Luis dos Santos

_____

Membro da Banca - Dr Jezer Machado – Instituição

_____

Membro da Banca – Dr Rafael Esteves – Instituição

Dedico este trabalho a minha familia, que se fizeram presentes em toda a minha trajetória e que sempre me apoiaram, compreendendo a importância desta conquista para nossa família.

# AGRADECIMENTOS

Agradeço aos meus familiares pelo apoio e incentivo durante minha jornada acadêmica no IFRS, agradeço aos meus professores por todo o comprometimento e dedicação prestados, e principalmente ao IFRS - Campus Restinga em que passei 6 anos de minha vida, contando o tempo de universidade e do ensino médio.

*“Seja uma referência de qualidade. As pessoas não estão acostumadas a ambientes onde a excelência é esperada.”*
*(Steve Jobs)*

**RESUMO**

O aumento do endividamento no Brasil e a baixa maturidade em educação financeira reforçam a necessidade de soluções simples, gratuitas e confiáveis para o controle de gastos. Este Trabalho de Conclusão de Curso apresenta o planejamento e a primeira entrega de um sistema web de gestão financeira pessoal priorizando a privacidade desde o início. Na fase atual, não há sincronização direta com contas bancárias. O foco está em reduzir atritos como digitar tudo à mão, reunir despesas de vários meios de pagamento e conferir parcelas/comprovantes e em dar ao usuário controle sobre seus dados. O protótipo funcional está disponível online; no qual o usuário entra com sua Conta Google pelo padrão OAuth 2.0, implementado com django-allauth, cria e edita categorias de gasto e registra/consulta lançamentos com poucos cliques. O *back-end* foi desenvolvido em Django (Python) e a interface utiliza Django *Templates* com Bootstrap 5, seguindo boas práticas de segurança e em conformidade com a LGPD.

**Palavras-chave:** Finanças pessoais, Gestão de despesas, Django, OCR, NF-e.

## *ABSTRACT*

*Brazil's growing household indebtedness and low financial literacy underscore the need for simple, free-of-friction, and trustworthy expense-tracking tools. This Final Paper presents the planning and first delivery of a privacy-first, web-based personal finance system. In its current stage, the system does not synchronize directly with bank accounts. Instead, it targets common frictions in expense logging—such as manual typing, consolidating spending across multiple payment methods, and checking installments/receipts—while keeping users in control of their own data.*

*A functional prototype is already available online: users sign in with their Google Account via OAuth 2.0 (implemented with django-allauth), create and edit custom spending categories, and record or review expenses with just a few clicks. The back end is built with Django (Python), and the interface uses Django Templates and Bootstrap 5, following security best practices and complying with Brazil's data-protection law (LGPD). By reducing entry barriers and centering privacy, the system aims to provide practical support for day-to-day financial organization.*



***Keywords:*** *Personal finance, Expense management, Django, OCR, NF-e.*


.

# LISTA DE FIGURAS

# LISTA DE TABELAS

## SUMÁRIO

# 1. INTRODUÇÃO

Controlar as finanças pessoais não é luxo, é condição para estabilidade e qualidade de vida. O retrato brasileiro, porém, mostra um atraso na educação financeira agravado pelo cenário econômico. O aumento do endividamento no Brasil e a baixa maturidade em gestão de recursos reforçam a necessidade de soluções que sejam simples, gratuitas e, acima de tudo, confiáveis. Segundo dados compilados pelo Senado Federal (2025), a partir de levantamentos da CNDL e do SPC Brasil, 43,1% da população adulta encontra-se negativada. A Pesquisa de Endividamento e Inadimplência do Consumidor (PEIC), da CNC (2025), aponta um quadro semelhante: quase 80% das famílias possuem algum tipo de dívida.

Soma-se a esse cenário a facilidade de acesso ao crédito, o parcelamento em múltiplos canais e a dificuldade de organizar gastos que se pulverizam entre cartões, Pix e dinheiro vivo. Além disso, uma pesquisa da Funpresp-Jud (2024) indica que 90% dos entrevistados reconhecem a necessidade de melhorar seus conhecimentos em finanças pessoais, sendo que 63% afirmam ter apenas entendimento básico sobre o tema. Esse conjunto de evidências sugere que o problema reside menos na falta de interesse e mais nas barreiras práticas do cotidiano: a digitação manual de lançamentos, a categorização repetitiva e a dificuldade de consolidar comprovantes geram um atrito que leva ao abandono do controle financeiro.

Simplificar o registro, a categorização e a leitura das despesas é, portanto, estratégico para favorecer a adoção contínua do controle de gastos, em consonância com diretrizes oficiais de educação financeira (BCB, 2022; CVM, 2019; GOV.BR, 2018). Diante disso, este trabalho propõe o desenvolvimento de um sistema de gestão de despesas pessoais projetado para reduzir esses atritos operacionais sem abrir mão da privacidade do usuário. O diferencial da solução reside na automação da entrada de dados: além do lançamento manual, o sistema implementa a importação automática de Notas Fiscais Eletrônicas (NF-e/NFC-e) através da leitura de QR Codes e do processamento de arquivos digitais (PDF).

É importante destacar uma evolução técnica decisiva no escopo deste projeto. Inicialmente, o sistema foi concebido para incluir o reconhecimento ótico de caracteres (OCR) visando a leitura de notas fiscais impressas em papel. Entretanto, após avaliação técnica das limitações práticas dessa abordagem, como a dependência da qualidade da câmera, iluminação precária e papel amassado,

optou-se por pivotar o foco exclusivamente para documentos fiscais eletrônicos. Esta decisão privilegia a confiabilidade na extração de dados, uma vez que arquivos PDF eletrônicos (DANFE) e os dados vinculados aos QR Codes já possuem estrutura padronizada e texto selecionável, eliminando as incertezas inerentes ao processamento de imagens. A escolha pelo eixo "nota fiscal eletrônica" evita, nesta fase, a integração bancária direta, preservando a sensação de segurança do usuário, pois o dado nasce do próprio comprovante da compra emitido pela SEFAZ, sem exigir senhas de banco.

O fluxo de uso foi desenhado para ser fluido: para cadastrar uma nova despesa, o usuário escaneia o QR Code com a câmera do dispositivo ou faz o upload do arquivo PDF da nota, o sistema processa as informações para extrair automaticamente o emitente, a data, os itens e os valores. Em seguida, apresenta categorias sugeridas ou já criadas, solicitando apenas que o usuário confirme a classificação. Esse modelo reduz drasticamente o trabalho em comparação à digitação manual e transforma o ajuste pontual em aprendizado para o sistema.

## 1.1. OBJETIVOS

### 1.1.1 OBJETIVO GERAL

O objetivo geral do projeto é disponibilizar uma ferramenta web gratuita, com foco na simplicidade de uso e no respeito à privacidade, capaz de reduzir o tempo e o esforço cognitivo necessários para o acompanhamento de gastos pessoais.

### 1.1.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

- **Viabilizar a importação automática:** Implementar mecanismos de leitura de QR Code e processamento de arquivos PDF (*DANFE*) para extração de dados estruturados;
- **Oferecer categorização assistida:** Desenvolver lógica de sugestão de categorias com base no histórico e em palavras-chave, permitindo ao usuário organizar seus gastos de forma personalizada;
- **Gestão de Orçamento e Renda:** Permitir o registro de renda mensal e a definição de tetos de gastos por categoria;
- **Sistema de Alertas Proativos:** Implementar notificações automáticas (via e-mail)

quando os gastos atingirem percentuais críticos do orçamento definido (configurados por padrão em 80%, 90% e 100%);

- **Visualização de Dados:** Apresentar *dashboards* interativos que permitam ao usuário identificar padrões de consumo, picos de despesa e distribuição orçamentária;
- ***Onboarding* Guiado:** Garantir um fluxo de configuração inicial que oriente o usuário no preenchimento de perfil e preferências na primeira utilização.

A relevância deste trabalho justifica-se pela lacuna existente entre planilhas manuais (trabalhosas) e aplicativos bancários (invasivos). A avaliação do sistema considerará indicadores qualitativos e observáveis: a redução do tempo médio para registrar uma despesa após a importação, a diminuição da taxa de correções manuais ao longo do uso e a eficácia dos alertas na prevenção do estouro de orçamento. Esses sinais orientarão a evolução do software, assegurando que ele responda aos problemas reais do cotidiano financeiro.

A organização do documento segue a estrutura: O Capítulo 2 apresenta os trabalhos relacionados e sistemas concorrentes. O Capítulo 3 descreve o levantamento de necessidades e os requisitos do sistema. O Capítulo 4 detalha a solução conceitual adotada. O Capítulo 5 relata a implementação técnica, detalhando as tecnologias e algoritmos empregados na extração de dados. O Capítulo 6 traz o cronograma de desenvolvimento, e, por fim, o Capítulo 7 reúne as considerações finais, resultados obtidos e propostas para trabalhos futuros.


menor preço NFG
Produto
GASOLINA COMUM
0.83 km
R$4.04
Nota Fiscal processada em
14/02/2018 16:05
VER NO MAPA
HISTÓRICO
COMPARTILHAR
Lista
Mapa

# 2. TRABALHOS RELACIONADOS

Para a realização deste Trabalho de Conclusão de Curso (TCC), foi conduzida uma revisão da literatura e de mercado com o objetivo de verificar sistemas que possuam requisitos funcionais similares aos propostos, bem como identificar as tecnologias e modelos de negócio vigentes.

O mercado de aplicativos de gestão financeira pessoal no Brasil é consolidado, contando com diversas soluções robustas. Para posicionar o sistema desenvolvido (BpCash), realizou-se uma análise comparativa com os principais players do setor: Mobills, Organizze, GuiaBolso (referência histórica) e o programa governamental Nota Fiscal Gaúcha (referência técnica).

A Tabela 1 apresenta uma síntese comparativa das funcionalidades, destacando os diferenciais da proposta deste trabalho em relação à gratuidade e à privacidade.

**Tabela 1** **SÍNTESE COMPARATIVA**

| **Funcionalidade** | **Sistema Proposto (BpCash)** | **Mobills** | **Organizze** | **GuiaBolso (Legado)** |
|---|---|---|---|---|
| **Modelo de Custo** | Gratuito | Freemium | Freemium | Gratuito |
| **Plataforma** | Web Responsivo | App Nativo / Web | App Nativo / Web | App Nativo |
| **Registro Manual** | Sim | Sim | Sim | Sim |

| | | | | |
|---|---|---|---|---|
| **Sincronização Bancária** | **Não** (Foco em privacidade) | Sim (Premium) | Não | Sim (Nativo) |
| **Gestão de Cartão/Contas** | Não | Sim | Sim | Sim |
| **Importação de Nota (PDF)** | **Sim** (Diferencial) | Não | Não | Não |
| **Leitura de QR Code (NF-e)** | **Sim** (Diferencial) | Não | Não | Não |
| **Alertas de Orçamento** | **Sim** (E-mail personalizado) | Sim (Premium) | Sim (Premium) | Apenas Push |
| **Autenticação** | Google / E-mail | Vários | Vários | Vários |

Fonte: próprio autor(2025).

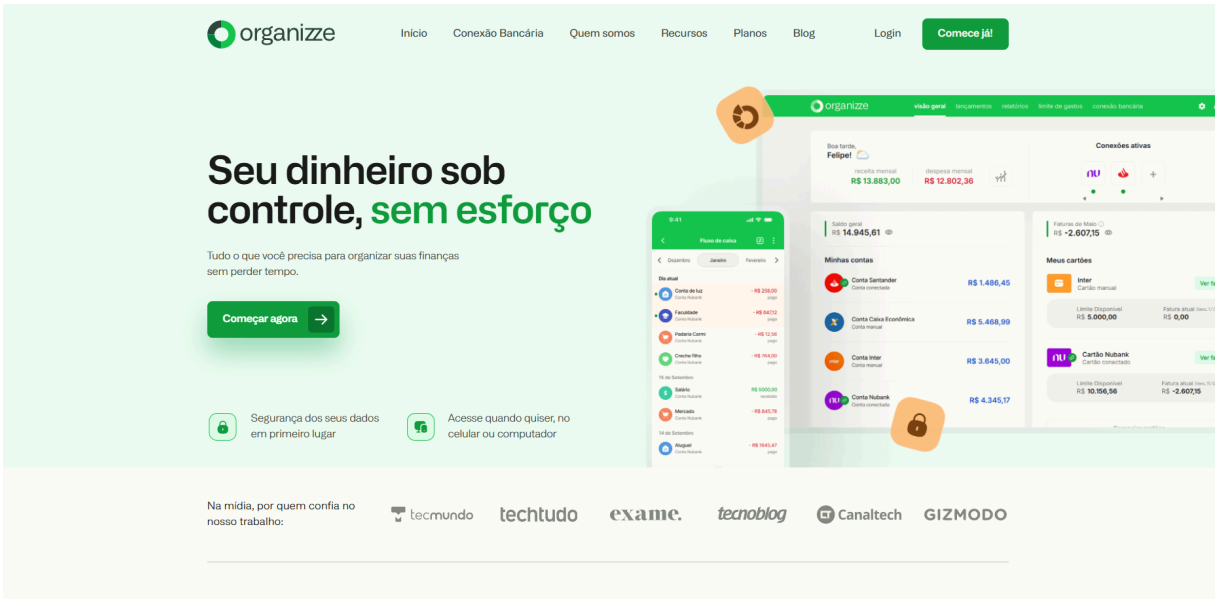
## 2.1. MOBILLS E ORGANIZZE (MODELO *FREEMIUM*)

O Mobills e o Organizze são as atuais referências de mercado no Brasil. Ambos operam sob o modelo de negócios *Freemium*, termo que define serviços que oferecem funcionalidades básicas gratuitamente (*Free*), enquanto cobram uma assinatura pelo acesso a recursos avançados (*Premium*), como relatórios personalizados e integração bancária automática. O Mobills se destaca por ser um dos mais completos, oferecendo controle de despesas, gerenciamento de cartões de crédito e planejamento de orçamento. A automação do registro de transações é feita via integração bancária, funcionalidade restrita aos assinantes pagos. Já o Organizze, foca na simplicidade e na experiência do usuário (*UX*), com uma interface limpa. Permite categorização de gastos e criação de metas. Diferencia-se

por não exibir anúncios mesmo na versão gratuita, mas também limita a automação aos planos pagos.

**Figura 1. SITE DO MOBILLS**


**Figura 2. SITE DO ORGANIZZE**




## 2.2. GUIABOLSO (DESCONTINUADO)

O GuiaBolso foi pioneiro no Brasil ao popularizar a sincronização automática

com contas bancárias, eliminando a necessidade de registro manual. Sua principal proposta de valor era a automação total e a análise de perfil de crédito. Apesar de seu sucesso técnico, o aplicativo foi descontinuado após ser adquirido pela PicPay, em um movimento estratégico de consolidação de mercado e advento da integração bancária direta (Open Finance). Sua trajetória evidencia a alta demanda por automação, mas também ressalta os riscos e a complexidade de depender de modelos de negócio que centralizam credenciais bancárias sensíveis dos usuários.


**Figura 3. APP DESATIVADO DO GUIABOLSO**

## 2.3. NOTA FISCAL GAÚCHA (MENOR PREÇO)

No âmbito governamental, o programa Nota Fiscal Gaúcha possui um

aplicativo (Menor Preço) que utiliza os dados das notas fiscais eletrônicas para pesquisa de valores. Embora seu objetivo não seja o controle de despesas pessoais, ele demonstra a viabilidade técnica de extrair e processar dados estruturados (emitente, produtos e valores) de notas fiscais eletrônicas em larga escala, servindo de inspiração técnica para o mecanismo de captura de dados desenvolvido neste trabalho.


**Figura 4. APP MENOR PREÇO**

## 2.4. SISTEMA PROPOSTO (BPCASH)

O sistema proposto (BpCash) diferencia-se dos concorrentes comerciais por ser totalmente gratuito e, sobretudo, por não exigir integração bancária. A opção arquitetural foi tratar o registro de despesas por meio de dados fiscais públicos (Notas Fiscais Eletrônicas), atuando como um "meio-termo" entre a digitação manual exaustiva e a entrega de senhas bancárias a terceiros.

Para viabilizar essa automação sem OCR de papel (que apresenta limitações de qualidade de imagem), o sistema implementa duas estratégias de captura de dados "nascidos digitais":

1. **Leitura de QR Code:** Captura via câmera do dispositivo ou *upload* de imagem, com decodificação da *url* de consulta da SEFAZ e posterior *scraping* dos dados estruturados (emitente, data, totais).
2. **Importação de Arquivo (PDF):** Processamento direto do arquivo DANFE, utilizando técnicas de mineração de texto para extrair tabelas de itens com precisão superior ao reconhecimento visual.

Adicionalmente, o sistema democratiza funcionalidades geralmente restritas a planos *Premium*, como os alertas proativos de orçamento, enviando notificações por e-mail quando o usuário atinge patamares críticos (80%, 90% e 100%) de seus limites definidos. A interface, construída como *Web* Responsiva, garante acesso universal sem a barreira de instalação de um aplicativo nativo.

# 3. LEVANTAMENTO DE REQUISITOS

A definição dos requisitos deste sistema não se baseou apenas na experiência empírica do autor, mas foi conduzida principalmente por uma análise de lacunas realizada sobre os concorrentes de mercado mapeados no capítulo anterior. Ao identificar que ferramentas como Mobills e Organizze restringem a automação a planos pagos e que soluções como o extinto GuiaBolso exigiam o compartilhamento de senhas bancárias, estabeleceu-se a premissa fundamental do projeto: oferecer automação de entrada de dados gratuita e segura, sem dependência de integração bancária (*Open Finance*).

Dessa forma, a elicitação dos requisitos funcionais guiou-se pela viabilidade técnica de extrair dados de documentos fiscais públicos. O sistema foi projetado para operar com uma abordagem híbrida de importação, priorizando documentos "nascidos digitais" em detrimento do reconhecimento ótico (OCR) de notas físicas amassadas, cuja complexidade e taxa de erro se mostraram inviáveis durante a prototipagem.

Para mitigar o trabalho manual do usuário, uma das principais queixas em apps gratuitos, definiu-se como requisito crítico a implementação de uma categorização automática baseada em heurísticas. O sistema deve confrontar o nome do estabelecimento extraído da nota (ex: "Zaffari", "Shell", "Panvel") com um dicionário interno de palavras-chave, sugerindo automaticamente classificações como "Mercado", "Vestuário" ou "Farmácia". Diferente de um cadastro manual simples, essa funcionalidade transforma o sistema em um assistente ativo.

Com relação à privacidade, o sistema adota requisitos estritos de soberania do usuário. Além da premissa de não coletar credenciais financeiras, foi estabelecido como requisito funcional a existência de um mecanismo de exclusão total de conta, permitindo que o usuário remova permanentemente todos os seus registros financeiros e pessoais do banco de dados a qualquer momento.

A organização do desenvolvimento seguiu uma abordagem incremental e evolutiva, estruturada em um Product Backlog. Diferente de projetos que entregam todas as funcionalidades apenas no final, optou-se por construir o sistema em camadas de complexidade crescente.

Inicialmente, foram desenvolvidas as estruturas fundamentais (Autenticação, Gestão de Perfil e CRUDs de Despesas), necessárias para garantir a estabilidade e a persistência dos dados. Uma vez consolidada essa base técnica, o foco do desenvolvimento voltou-se integralmente para a implementação dos diferenciais de valor: a importação de Notas Fiscais Eletrônicas, a inteligência de categorização e o sistema de alertas. Essa estratégia permitiu validar a arquitetura básica antes de introduzir a complexidade dos algoritmos de extração de dados e processamento assíncrono.

A Tabela 2 apresenta o Product Backlog consolidado, ordenado por prioridade de negócio.

***Tabela 2.*** ***PRODUCT BACKLOG***

| ID | Método | Eu, como Usuário, quero… | Para que... | Prioridade |
|---|---|---|---|---|
| **01** | **Acesso** | Fazer login via Google (OAuth 2.0). | Eu acesse o sistema com um clique, sem decorar senhas. | **Crítica** |
| **02** | **Acesso** | Criar conta com e-mail e senha. | Eu tenha uma alternativa caso não queira usar redes sociais. | **Alta** |

| | | | | |
|---|---|---|---|---|
| **03** | ***Onboarding*** | Ser guiado na configuração inicial. | Eu defina minha Renda e Preferências de Alerta logo no primeiro acesso. | **Crítica** |
| **04** | **Importação** | Importar despesas via QR Code (Câmera/Imagem). | O sistema busque os dados na SEFAZ automaticamente. | **Crítica** |
| **05** | **Importação** | Importar despesas via Upload de PDF (DANFE). | Eu registre notas recebidas por e-mail ou baixadas no celular. | **Crítica** |
| **06** | **Interface** | Visualizar indicador de carregamento. | Eu saiba que o sistema está processando a nota e não travou. | **Alta** |
| **07** | **Assitência** | Ter categorias sugeridas automaticamente. | O sistema preencha "Mercado" ao identificar "Zaffari" ou "Carrefour". | **Crítica** |
| **08** | **Gestão** | Criar categorias "rápidas" durante a despesa. | Eu cadastre uma nova categoria (ex: "Jogos") sem perder os dados da despesa atual. | **Média** |

| | | | | |
|---|---|---|---|---|
| **09** | **Gestão** | Revisar dados importados antes de salvar. | Eu garanta que o valor e a categoria estão corretos. | **Alta** |
| **10** | **Despesas** | Cadastrar despesas manualmente. | Eu registre gastos informais (ex: feira, pagamentos em dinheiro). | **Média** |
| **11** | **Despesas** | Listar despesas com paginação. | Eu navegue pelo histórico sem carregar uma lista infinita. | **Média** |
| **12** | **Despesas** | Excluir ou Editar despesas lançadas. | Eu corrija erros de digitação ou remova lançamentos duplicados. | **Média** |
| **13** | **Receitas** | Cadastrar Renda Fixa (Salário). | O sistema calcule meu orçamento base mensal. | **Alta** |
| **14** | **Receitas** | Cadastrar Receitas Variáveis (Extras/Bônus). | Eu registre entradas esporádicas que aumentam meu saldo no mês. | **Média** |
| **15** | **Orçamento** | Configurar gatilhos de alerta (80%, | Eu escolha quão rígido quero que o monitoramento | **Alta** |

| | | | | |
|---|---|---|---|---|
| | | 90%, 100%). | seja. | |
| **16** | **Notificação** | Receber e-mails de alerta formatados. | Eu visualize claramente o quanto já consumi do meu orçamento. | **Alta** |
| **17** | **Dashboard** | Visualizar saldo disponível em tempo real. | Eu tome decisões rápidas de compra ("posso gastar hoje?"). | **Alta** |
| **18** | **Dashboard** | Analisar gastos por Categoria (Gráfico). | Eu identifique ofensores do orçamento (ex: estou gastando muito em "Lazer"). | **Média** |
| **19** | **Privacidade** | Excluir minha conta permanentemente. | Eu tenha soberania sobre meus dados, removendo tudo se desejar sair. | **Crítica** |
| **20** | **Mobile** | Acessar o sistema pelo celular (Responsivo). | Eu registre despesas na rua, no momento em que elas acontecem. | **Crítica** |

Fonte: Próprio autor.

# 4. SOLUÇÃO CONCEITUAL E ARQUITETURA

Após identificar as necessidades do Sistema de Gerenciamento de Despesas Pessoais e as limitações das ferramentas existentes, definiu-se uma solução arquitetural baseada no padrão MVT (*Model-View-Template*), uma variação moderna do clássico MVC (*Model-View-Controller*) adaptada para o desenvolvimento web. A solução prioriza a separação de responsabilidades, a modularidade e a escalabilidade, permitindo a evolução incremental do sistema.

A autenticação foi projetada para delegar a gestão de credenciais a provedores externos confiáveis (via protocolo OAuth 2.0), eliminando a necessidade de armazenar senhas localmente e aumentando a segurança do usuário. O sistema foi concebido para operar em múltiplos dispositivos (*desktop, tablet, smartphone*) através de uma interface *web* responsiva, garantindo acessibilidade universal sem a barreira de instalação de aplicativos nativos.

## 4.1. DIAGRAMA DE CLASSES

O diagrama de classes reflete a estrutura real do código, incluindo particularidades do framework Django. A principal distinção arquitetural é a separação entre User e Usuario:

- **User:** Classe nativa do Django (django.contrib.auth.models.User) que gerencia autenticação, credenciais e permissões.
- **Usuario:** Modelo de extensão (perfil) com atributos específicos do domínio financeiro, conectado ao User via relacionamento OneToOne.

Essa abordagem segue o padrão recomendado pela documentação do Django para estender o modelo de usuário sem modificar a classe base de autenticação (DJANGO, 2025c).

O diagrama implementado é composto pelas seguintes classes principais:

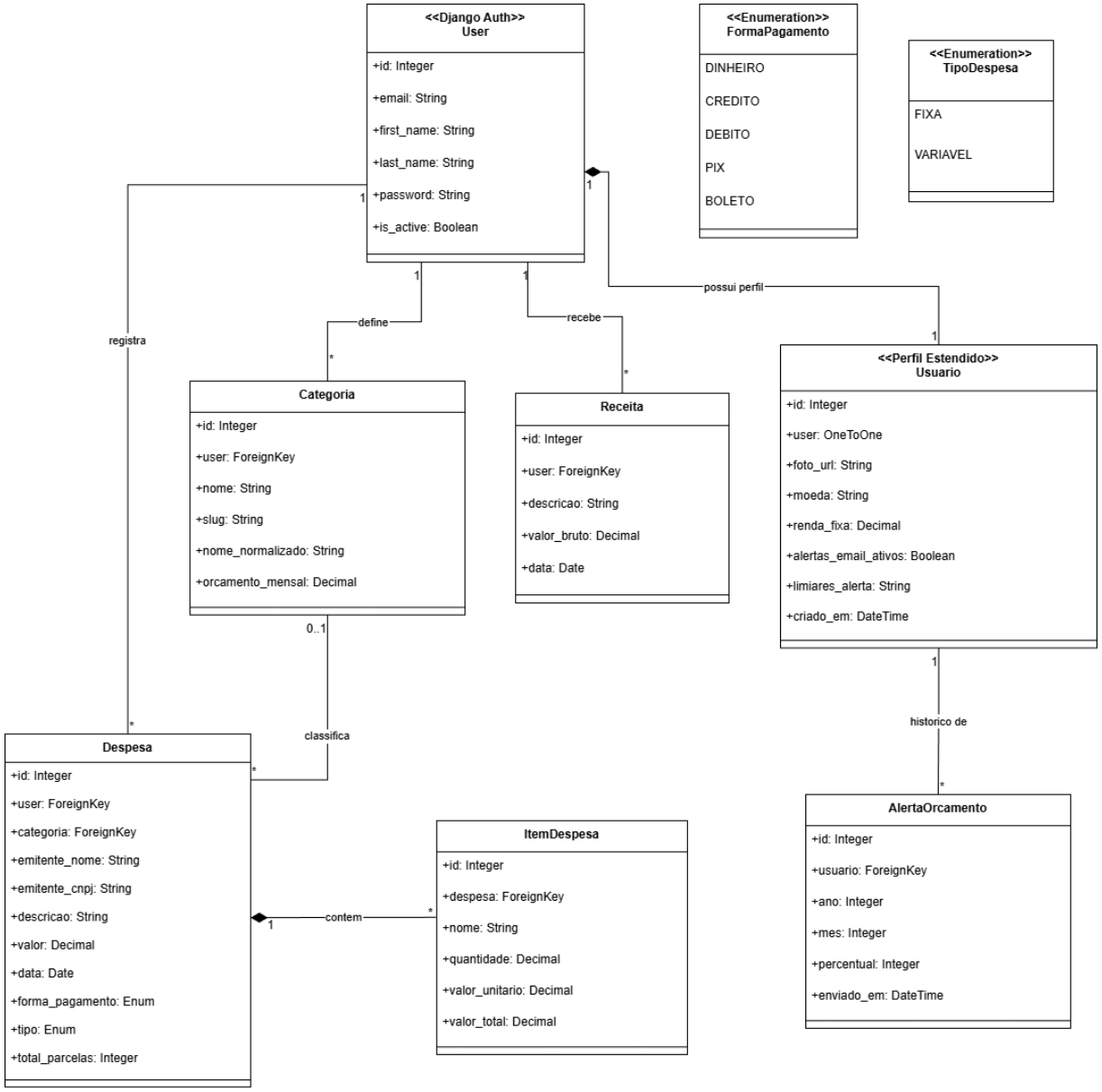
A entidade *User* é nativa do Django e gerencia credenciais de acesso, enquanto a entidade Usuario estende esse cadastro para armazenar preferências do domínio financeiro, como foto do usuário e configurações de alerta. Esta separação mantém a responsabilidade de autenticação isolada das regras de negócio.

As transações financeiras são representadas pelas entidades Despesa e Receita. A Despesa armazena detalhes do gasto (emitente, valor, data) e se relaciona com ItemDespesa, que detalha os produtos adquiridos quando há

importação de nota fiscal. Para organização, as despesas são agrupadas pela entidade Categoria (ex: Mercado, Lazer), que possui um limite de orçamento mensal. Por fim, a entidade AlertaOrcamento registra o histórico de notificações enviadas ao usuário para evitar duplicidade.

A Figura 4 apresenta o Diagrama de Classes, refletindo as entidades e relacionamentos implementados.

**Figura 4. DIAGRAMA DE CLASSES**




Fonte: Próprio autor.

## 4.2. FLUXO DE IMPORTAÇÃO DE NOTA FISCAL ELETRÔNICA

O processo de importação utiliza duas técnicas computacionais principais: a leitura ótica de códigos bidimensionais (*QR Codes*) e a extração automatizada de dados *web* (*Web Scraping*).

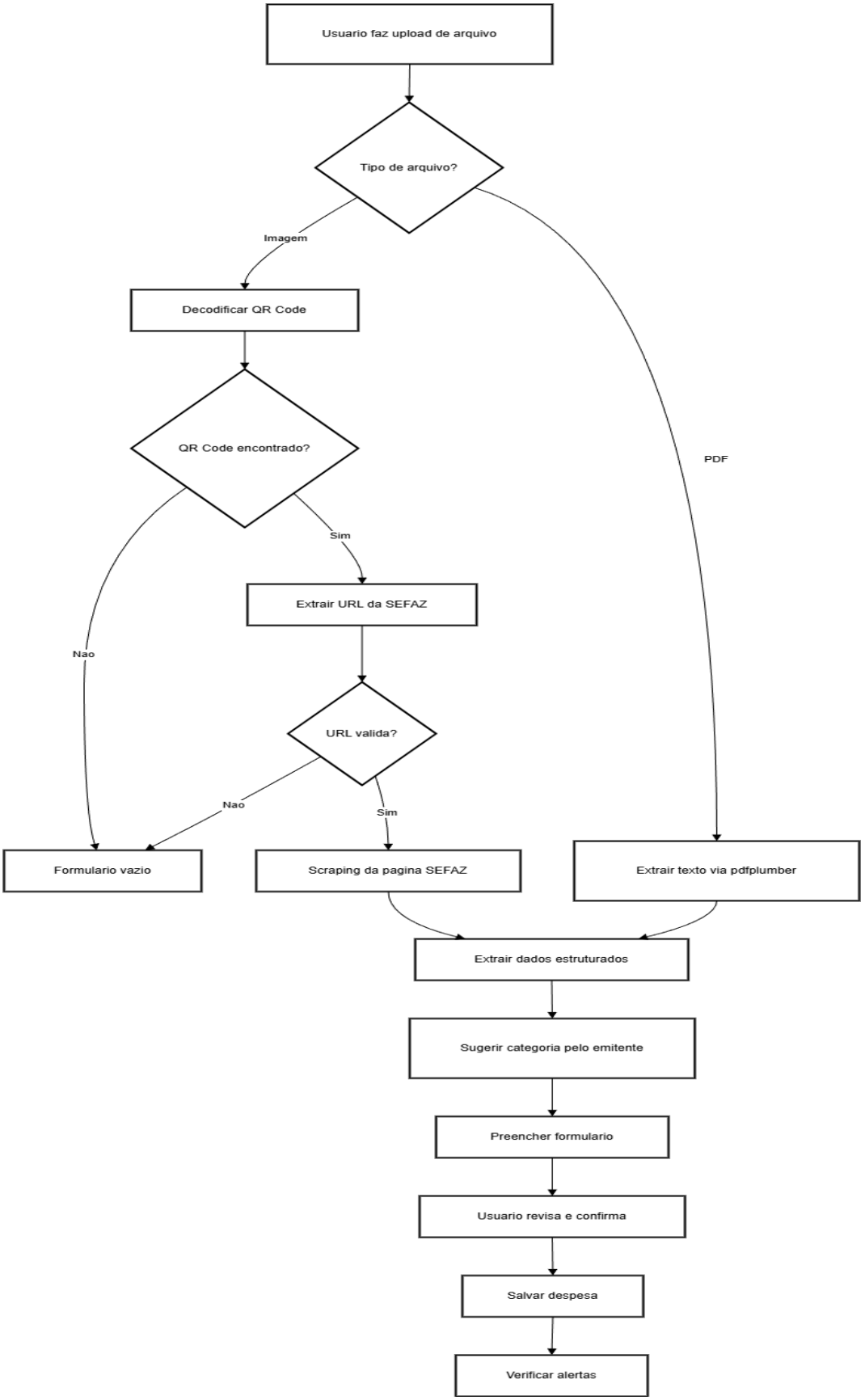
O *Web Scraping* é definido como a técnica de extração de dados de sites através de *software* automatizado (MITCHELL, 2018). No contexto deste trabalho, o *scraping* é utilizado para acessar a página pública da Nota Fiscal Eletrônica na Secretaria da Fazenda. Diferente de uma API formal, a página HTML é processada pelo sistema, que identifica e captura informações específicas (CNPJ, data, valores e itens) contidas em tags *HTML* estruturadas, transformando dados visuais em registros de banco de dados.

A Validação do *QRCode* ocorre em etapas sucessivas. Inicialmente, bibliotecas de visão computacional (como OpenCV e ZBar) localizam e decodificam o padrão visual da imagem em uma string de texto. O sistema então aplica validações conceituais sobre essa string: verifica se ela possui o formato de uma *URL* válida, se aponta para um domínio governamental confiável (sefaz.rs.gov.br) e se contém os parâmetros necessários para a consulta da nota. Esta verificação é crucial para garantir que apenas *QR Codes* fiscais legítimos sejam processados, descartando imagens inválidas ou tentativas de injeção de *URLs* maliciosas.

O processo também implementa validação de segurança contra ataques SSRF, bloqueando URLs que apontem para endereços IP internos antes de realizar a requisição externa. A predição de categoria utiliza um dicionário de palavras-chave que mapeia nomes de estabelecimentos conhecidos para categorias sugeridas.

O fluxograma a seguir detalha esse processo integrado, desde o upload até o preenchimento do formulário.

**Figura 5. FLUXOGRAMA IMPORTAÇÃO NFE**




Fonte: Próprio autor.

## 4.3. FLUXO DE ALERTAS DE ORÇAMENTO

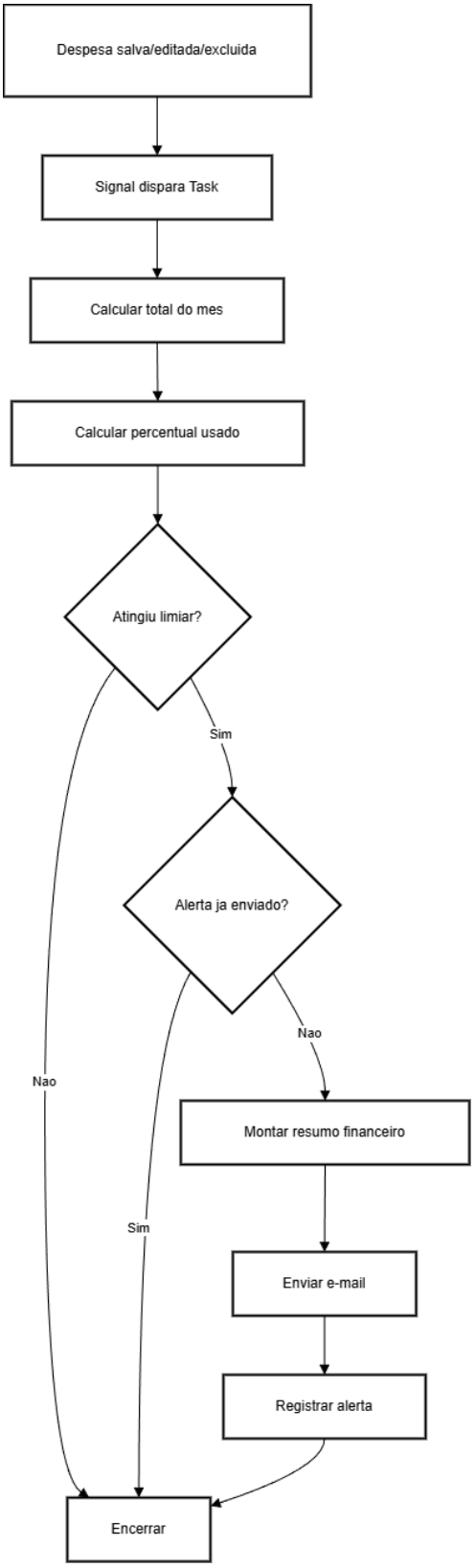
O sistema de alertas opera seguindo o padrão comportamental ***Observer*** (ou *Publish-Subscribe*), onde alterações no estado financeiro do usuário disparam automaticamente a verificação de regras de negócio.

O mecanismo deve ser reativo e assíncrono: sempre que uma despesa é registrada, alterada ou removida (evento de mudança de estado), o sistema aciona um processo em segundo plano. Este processo recalcula o total de gastos do mês corrente e compara o valor acumulado com os limiares percentuais definidos pelo usuário.

Caso um limiar (ex: 90% do orçamento) seja atingido ou superado, o sistema consulta o histórico para verificar se este evento específico já foi notificado no período atual. Se for um novo evento, uma notificação é gerada e enviada via e-mail. Esta arquitetura desacopla a lógica de transação (salvar despesa) da lógica de notificação (enviar alerta), garantindo que o tempo de resposta da interface não seja prejudicado pelo processamento dos avisos.

Os limiares padrão são 80%, 90% e 100% do orçamento mensal, podendo ser personalizados pelo usuário durante o onboarding ou nas configurações.

**Figura 6. FLUXOGRAMA IMPORTAÇÃO NFE**




**Fonte: Próprio autor**

# 5. IMPLEMENTAÇÃO

Com a conclusão da fase de planejamento conceitual e levantamento de requisitos, inicia-se a etapa de desenvolvimento prático do sistema. O objetivo desta fase é a construção do Mínimo Produto Viável (MVP), uma versão funcional que engloba as funcionalidades essenciais: autenticação de usuários via Google e o ciclo completo de CRUD (criação, leitura, atualização e exclusão) de categorias e despesas.

Esta seção detalha as escolhas tecnológicas que deram suporte ao projeto e apresenta o sistema desenvolvido até o momento.

## 5.1 Tecnologias utilizadas

Para a criação do sistema de gerenciamento de despesas pessoais, foram utilizadas diversas tecnologias, cada uma com um propósito específico. Entre as tecnologias utilizadas, podem ser citados o framework Django para o uso da linguagem Python, o Bootstrap 5 para o design das telas, o Google Cloud Platform para o serviço de autenticação OAuth 2.0, e o Visual Studio Code como editor de código. A linguagem Python foi escolhida por sua sintaxe clara, vasta biblioteca padrão e um ecossistema robusto para desenvolvimento web, sendo a base para o framework Django.

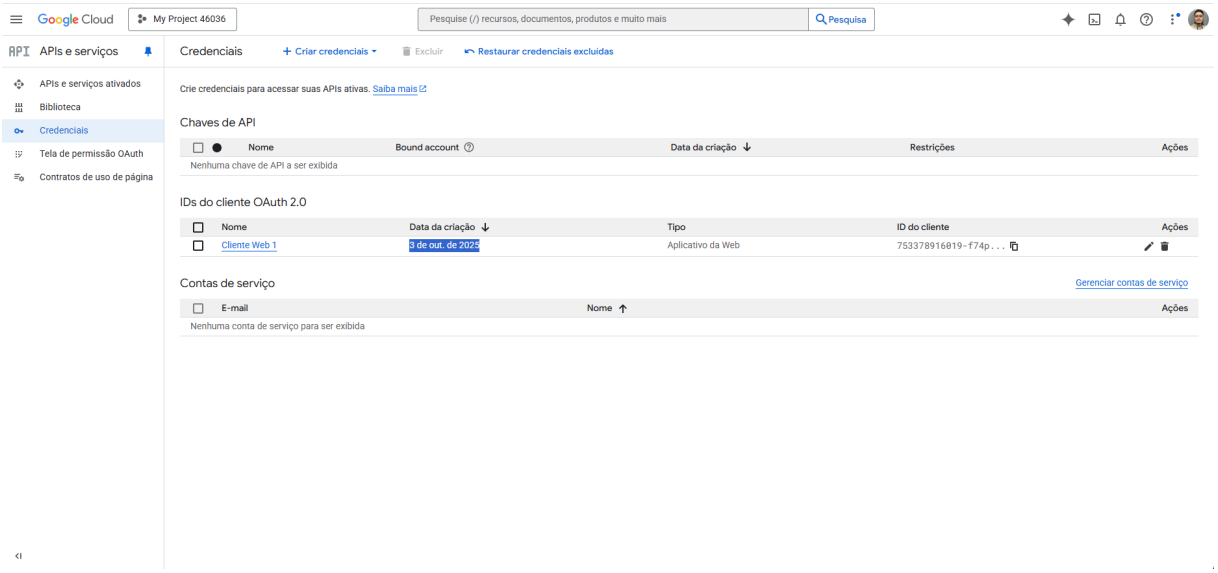
O framework Django 5 é um framework web de alto nível baseado na arquitetura MVT (*Model-View-Template*). Ele foi escolhido por sua filosofia de "baterias incluídas", que fornece componentes prontos e seguros para tarefas comuns, como ORM (Mapeamento Objeto-Relacional) para interação com o banco de dados, sistema de roteamento, motor de *templates* e um painel administrativo, o que acelera significativamente o desenvolvimento.

A implementação do front-end do sistema foi desenvolvida com o framework de CSS Bootstrap 5. Este framework facilita o desenvolvimento da interface, possibilitando que o sistema tenha um design responsivo e moderno, sem a necessidade de uma codificação extensiva de estilos CSS. Os componentes do Bootstrap foram utilizados como base e personalizados com CSS próprio (custom.css) para adequar a identidade visual do projeto .

Para a funcionalidade de autenticação social, foi utilizada a biblioteca django-allauth . Ela foi escolhida por ser uma solução completa e segura para lidar

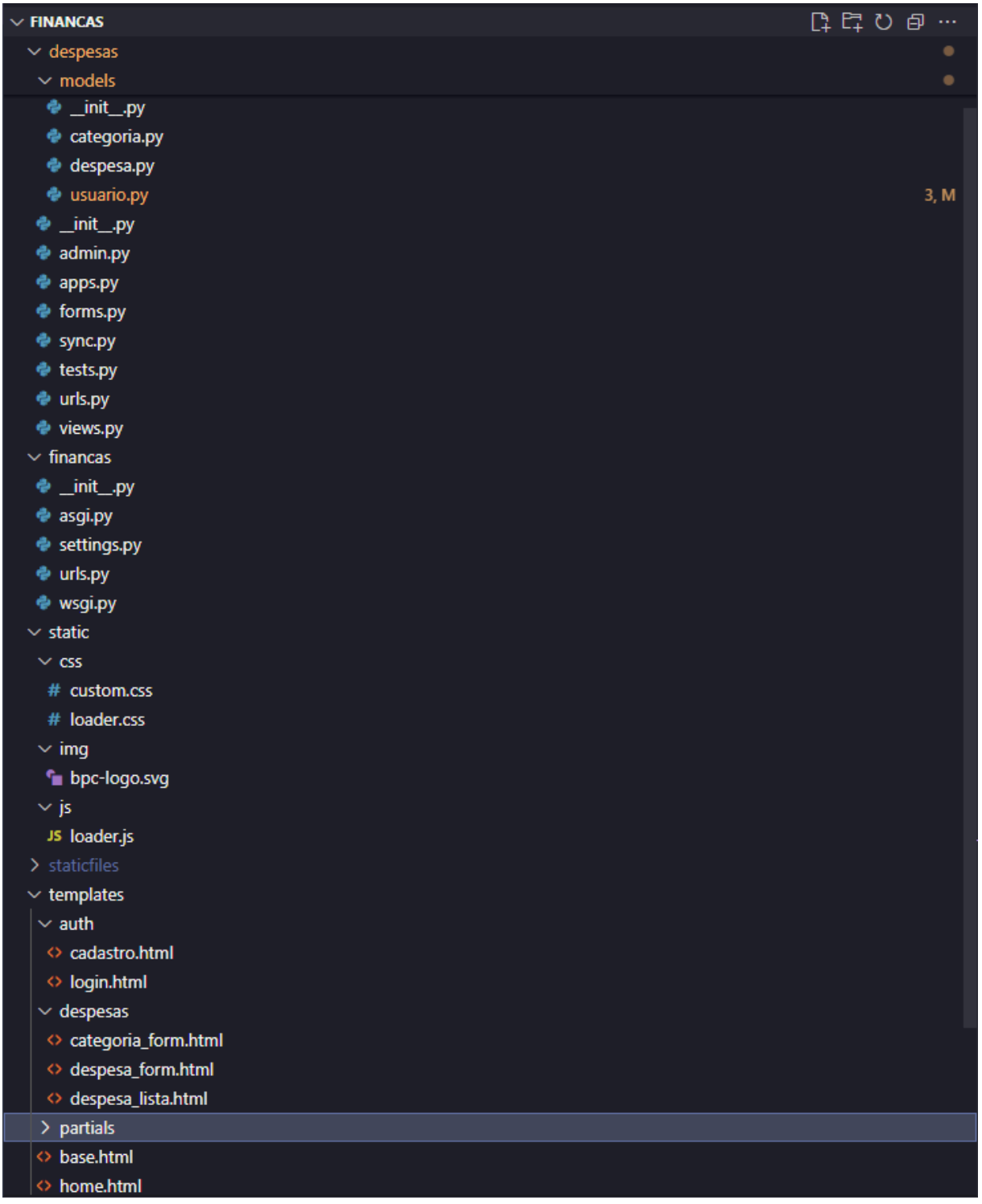
com registros e *logins* locais e sociais. Para a integração com o Google, foi necessário configurar um projeto no Google Cloud Platform, onde as credenciais do cliente OAuth 2.0 (ID do cliente e Chave secreta) foram geradas, como pode ser observado na Figura 5.

**Figura 7. Console Google Cloud**




Todo o desenvolvimento, front-end e *back-end*, foi feito através do editor de texto Visual Studio Code, que oferece integração com o terminal, controle de versão e um ecossistema rico de extensões que facilitam o desenvolvimento em Python e Django.

**Figura 8. Estrutura do projeto no Visual Studio Code**

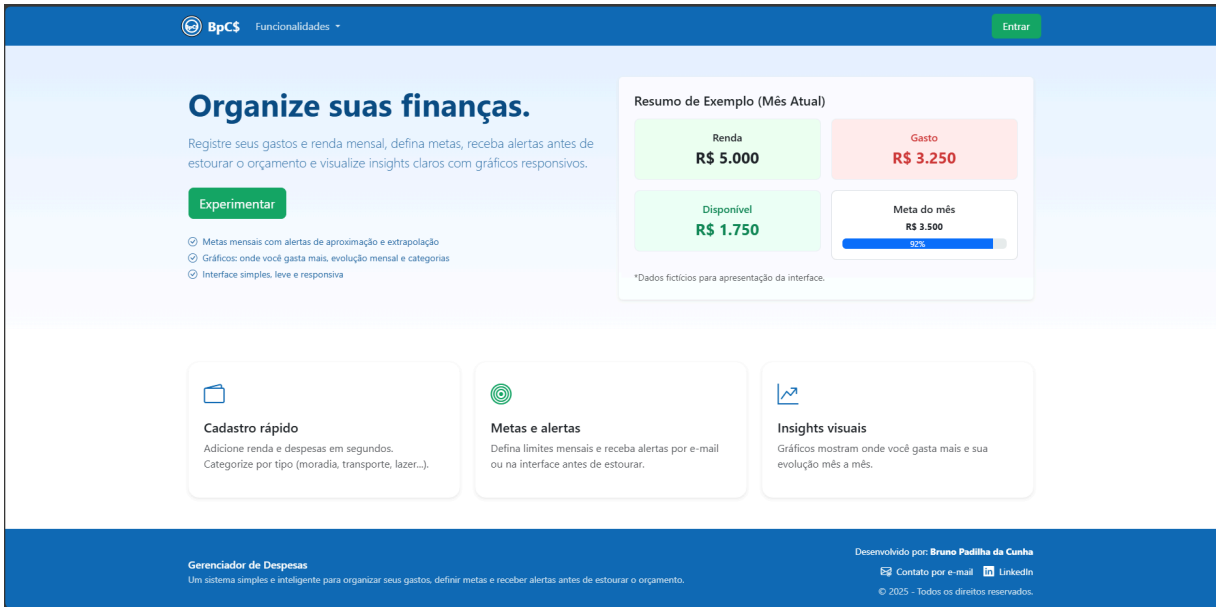



Na fase atual de desenvolvimento, o banco de dados utilizado é o SQLite . Por ser um banco de dados baseado em arquivo e já integrado nativamente ao Django, ele simplifica a configuração do ambiente de desenvolvimento local, sendo uma escolha prática para a prototipagem e o desenvolvimento do MVP.

## 5.2 Sistema desenvolvido

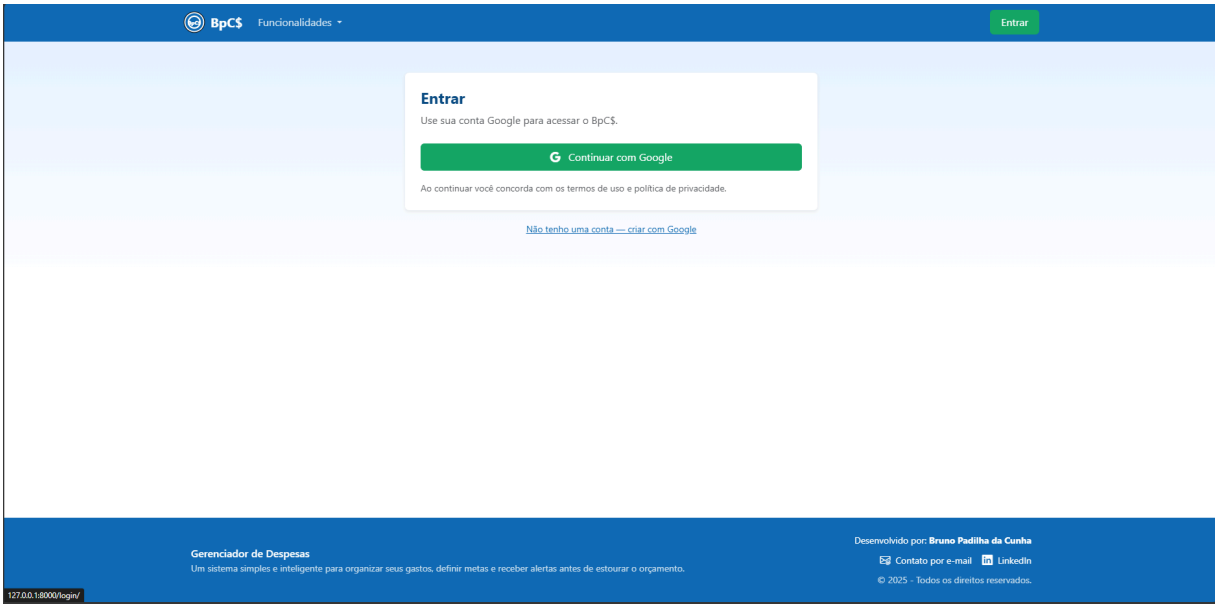
Com o uso das tecnologias citadas, foi possível realizar o desenvolvimento do Mínimo Produto Viável (MVP) do sistema (Gerenciador de Despesas). A tela inicial (*landing page*) do sistema pode ser observada na Figura 7.

**Figura 9. Home Page**




Foi desenvolvido o módulo de controle de usuários, com foco exclusivo na autenticação social via Google, eliminando a necessidade de um cadastro tradicional com e-mail e senha. Ao acessar a tela de "Entrar", o usuário é direcionado para o fluxo de autenticação do Google.

**Figura 10. Tela de *Login***




Este fluxo é iniciado por um link no *template login*.html, que utiliza as *template* tags do django-allauth para gerar a URL de autenticação correta, como demonstrado no código da Figura 9.

**Figura 11. Código da configuração de fluxo do OAuth 2.0**

```
templates > auth > login.html > ...
1  {% extends "base.html" %}
2  {% load socialaccount %}
3  {% provider_login_url 'google' process='login' as google_login_url %}
4  {% provider_login_url 'google' process='signup' as google_signup_url %}
5
6
7  {% block content %}
8  <section class="py-5 hero-blue">
9    <div class="container">
10     <div class="row justify-content-center">
11       <div class="col-md-7 col-lg-6">
12         <div class="card border-0 shadow-sm">
13           <div class="card-body p-4">
14             <h1 class="h4 fw-bold mb-2 text-brand-blue-900">Entrar</h1>
15             <p class="text-body-secondary mb-4">Use sua conta Google para acessar o BpC$.</p>
16             <a href="{{ google_login_url|default:'/accounts/google/login/?process=login' }}"
17               class="btn btn-brand-green w-100 py-2">
18               <i class="bi bi-google me-2"></i> Continuar com Google
19             </a>
20
21             <p class="small text-body-secondary mt-3 mb-0">
22               Ao continuar você concorda com os termos de uso e política de privacidade.
23             </p>
24           </div>
25         </div>
26         <div class="text-center mt-3">
27           <a href="{{ google_signup_url|default:'/accounts/google/login/?process=signup' }}"
28             class="small text-brand-blue">
29             Não tenho uma conta — criar com Google
30           </a>
31         </div>
32       </div>
33     </div>
34   </div>
35 </section>
36 {% endblock %}
```

Para que o django-allauth funcione corretamente, foi necessário configurar os *INSTALLED_APPS*, *AUTHENTICATION_BACKENDS* e as configurações do provedor "google" no arquivo *settings.py* do Django.

**Figura 12. Configuração do django-allauth e provedor Google no settings.py**

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "django.contrib.sites",
    "despesas.apps.DespesasConfig",
    "allauth",
    "allauth.account",
    "allauth.socialaccount",
    "allauth.socialaccount.providers.google",
]

SITE_ID = 1
SOCIALACCOUNT_LOGIN_ON_GET = True

AUTHENTICATION_BACKENDS = [
    "django.contrib.auth.backends.ModelBackend",
    "allauth.account.auth_backends.AuthenticationBackend",
]

LOGIN_REDIRECT_URL = "/"
LOGOUT_REDIRECT_URL = "/"

SOCIALACCOUNT_PROVIDERS = {
    "google": {
        "SCOPE": ["openid", "email", "profile"],
        "AUTH_PARAMS": {"prompt": "consent"},
    }
}
```

O roteamento das URLs de autenticação (/accounts/google/*login*/) é gerenciado pelo django-allauth, que é incluído no urls.py principal do projeto

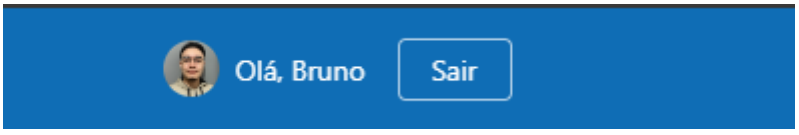
**Figura 13. Inclusão das rotas do allauth no urls.py**

```
pesas > urls.py > ...
from django.contrib import admin
from django.urls import path, include
from . import views as v

urlpatterns = [
    path("", v.home, name="home"),
    path("login/", v.pagina_login, name="login"),
    path("cadastro/", v.pagina_cadastro, name="signup"),
    path("accounts/", include("allauth.urls")),
    path("despesas/", v.listar_despesa, name="listar_despesa"),
    path("despesas/nova/", v.criar_despesa, name="despesa_criar"),
    path("despesas/<int:pk>/editar/", v.editar_despesa, name="despesa_editar"),
    path("categorias/nova/", v.criar_categoria, name="categoria_criar"),
]
```

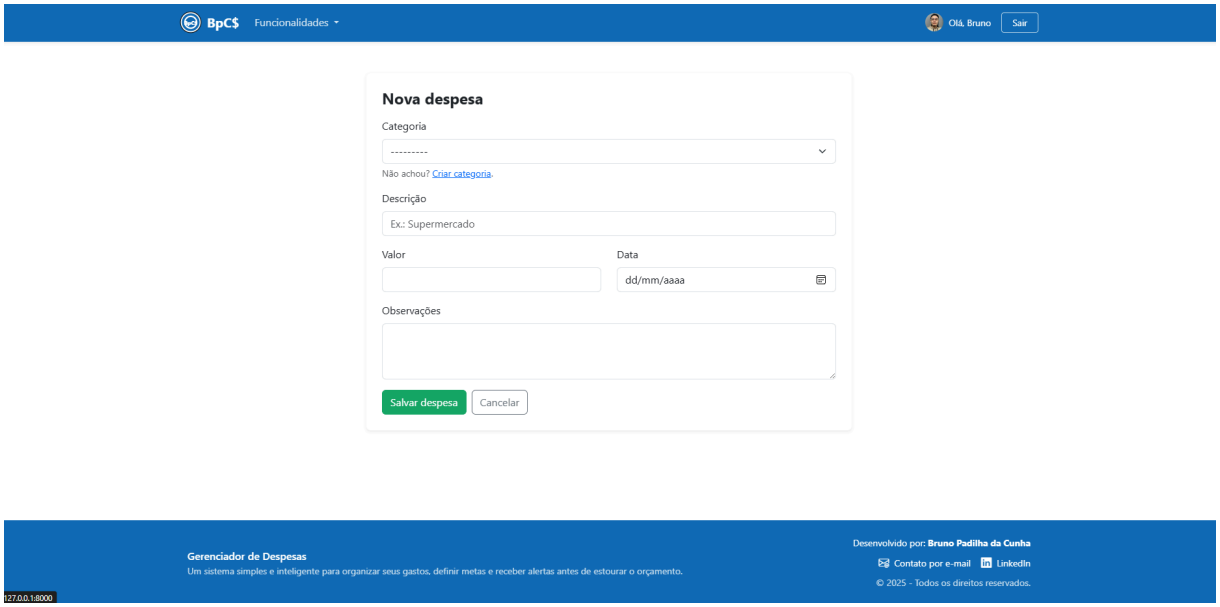
Após a autenticação bem-sucedida, o django-allauth cria o usuário no banco de dados (aproveitando o *model user* nativo do Django ) e o redireciona para a página inicial, agora logado. A interface atualiza, exibindo o nome e a foto do usuário no cabeçalho.

**Figura 14. Layout do sistema autenticado, exibindo nome e foto do usuário**



Como o principal objetivo do sistema é o gerenciamento de gastos, a funcionalidade central do MVP é a criação de despesas. O usuário, uma vez logado, pode acessar o formulário para registrar uma "Nova despesa".

**Figura 15. Tela de cadastro de nova despesa**




Esta tela é servida por uma view e rota específica (despesas/nova/), conforme definido no arquivo urls.py (visto anteriormente na Figura 11). O formulário permite ao usuário selecionar uma categoria (previamente cadastrada por ele), adicionar uma descrição, valor, data e observações. Esta funcionalidade, juntamente com a listagem de despesas e o cadastro de categorias, compõe o núcleo funcional do MVP.

Um aspecto central do sistema é dar ao usuário um jeito simples de organizar os próprios gastos. Isso é feito pelo *model* de Categoria (**Figura 14**). Para garantir privacidade e separação dos dados, essa tabela se liga ao *User* nativo do Django por uma ForeignKey (DJANGO, 2025c). Na prática, cada categoria pertence a um único usuário: quem criou é o único que vê e edita.

**Figura 16. Model de categoria**


Um desafio importante foi tratar duplicidades de um modo intuitivo, por exemplo, impedir que o mesmo usuário cadastre “Alimentação” e “alimentacao” como se fossem categorias diferentes. A solução foi aplicar validação em camadas, do banco até a interface, para cobrir tanto o uso normal quanto casos de corrida.

No models.py (**Figura 14**), o *model* Categoria tem o campo ‘nome_normalizado’. Ele guarda uma versão simplificada do nome: tudo em minúsculas e sem acentos (via slugify). Assim, “Alimentação” e “alimentacao” viram a mesma coisa para comparação. Para fechar a porta a duplicatas no nível do banco, há uma UniqueConstraint (DJANGO, 2025d) sobre ['*user*', 'nome_normalizado'], garantindo unicidade por usuário.

Para dar retorno imediato ao usuário, a validação principal acontece no CategoriaForm. Esse formulário é um ModelForm (DJANGO, 2025b) personalizado que recebe o *user* no __init__ (vindo da *view*). No clean_nome, o formulário normaliza o texto do mesmo jeito que a *model* faz e consulta se aquele usuário (self.*user*) já tem uma categoria com o mesmo nome_normalizado. Se tiver, dispara uma *ValidationError* (DJANGO, 2025a) com uma mensagem amigável, tudo antes de tentar salvar.

**Figura 17. ModelForm de categoria**

```python
class CategoriaForm(forms.ModelForm):
    class Meta:
        model = Categoria
        fields = ["nome"]
        widgets = {
            "nome": forms.TextInput(
                attrs={"class": "form-control", "placeholder": "Ex.: Alimentação"}
            ),
        }
        labels = {"nome": "Nome da categoria"}

    def __init__(self, *args, **kwargs):
        self.user = kwargs.pop("user", None)
        super().__init__(*args, **kwargs)

    def clean_nome(self):
        nome = (self.cleaned_data.get("nome") or "").strip()
        if not nome:
            raise forms.ValidationError("Informe um nome para a categoria.")
        normalizado = slugify(nome, allow_unicode=False)
        if self.user and Categoria.objects.filter(
            user=self.user, nome_normalizado=normalizado
        ).exists():
            raise forms.ValidationError("Você já possui uma categoria com esse nome.")
        return nome

    def save(self, commit=True):
        obj = super().save(commit=False)
        if self.user and not obj.user_id:
            obj.user = self.user
        if commit:
            obj.save()
        return obj
```

A *view* criar_categoria amarra o fluxo. Ela é protegida por *@login_required* (***DJANGO, 2025c***), então só usuários autenticados acessam. Ao receber um *POST*, instancia o CategoriaForm com *request.POST* e o *user=request.user*. O salvamento roda dentro de um *with transaction.atomic()* (***DJANGO, 2025e***) e de um *try...except IntegrityError*. Isso cobre o caso raro de *race condition*: se duas requisições escaparem da validação do formulário ao mesmo tempo, a *Unique Constraint* do banco vai barrar a duplicata, o erro é capturado e vira feedback claro na interface, sem quebrar a aplicação.

**Figura 18. View de Categoria**

```python
@login_required
def criar_categoria(request):
    if request.method == "POST":
        form = CategoriaForm(request.POST, user=request.user)
        if form.is_valid():
            try:
                with transaction.atomic():
                    form.save()
            except IntegrityError:
                form.add_error("nome", "Você já possui uma categoria com esse nome.")
            else:
                messages.success(request, "Categoria criada com sucesso.", extra_tags="categoria")
                return redirect("listar_despesa")
    else:
        form = CategoriaForm(user=request.user)

    return render(request, "despesas/categoria_form.html", {"form": form})
```

Por fim, o *template* categoria_form.html, renderizado por essa view, usa componentes de formulário do Bootstrap 5 ***(BOOTSTRAP, 2025)*** para exibir campos e mensagens de erro de forma limpa, responsiva e consistente, bom em telas grandes e também no celular.

**Figura 19. *Template* do formulário de categoria**

```html
{% extends "base.html" %}

{% block content %}
<div class="container py-4">
    <div class="row justify-content-center">
        <div class="col-md-6 col-lg-5">
            <div class="card shadow-sm border-0">
                <div class="card-body p-4">
                    <h1 class="h5 fw-bold text-brand-blue-900 mb-3">Nova categoria</h1>

                    {% if form.non_field_errors %}
                    <div class="alert alert-danger">
                        {{ form.non_field_errors }}
                    </div>
                    {% endif %}

                    <form method="post" novalidate>
                        {% csrf_token %}

                        <div class="mb-3">
                            <label class="form-label">{{ form.nome.label }}</label>
                            {{ form.nome }}
                            {% if form.nome.errors %}
                            <div class="text-danger small mt-1">
                                {{ form.nome.errors|striptags }}
                            </div>
                            {% endif %}
                        </div>

                        <div class="d-flex gap-2">
                            <button class="btn btn-brand-green">Salvar</button>
                            <a href="{% url 'listar_despesa' %}" class="btn btn-outline-secondary">Cancelar</a>
                        </div>
                    </form>

                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

# 6. CRONOGRAMA

O cronograma para o desenvolvimento do trabalho abrange as seguintes etapas consideradas essenciais, desde o desenvolvimento das funcionalidades centrais até a implantação e defesa final. Abaixo se encontra o detalhamento das etapas, atividades e os prazos que compõem esse cronograma.

**Implementação da leitura OCR de NFs e NFe's (4 semanas):**

Será feita a implementação da lógica para a leitura das notas fiscais, abrangendo tanto a captura por fotos ou arquivos de imagem (OCR) quanto a importação direta de arquivos XML (NFe).

**Alterações e ajustes apontados na sessão prévia (1 semana):**

Haverá a revisão e alteração de todos os pontos que forem trazidos pela banca na sessão prévia, com o intuito de que o trabalho esteja bem elaborado e estruturado.

**Adição do conteúdo ao documento (1 semana):**

Paralelamente ao desenvolvimento das funcionalidades, o corpo do documento do TCC será continuamente atualizado para refletir as novas implementações, diagramas e resultados.

**Implementação da adição/visualização das receitas (1 semana):**

Desenvolvimento das views, formulários e *templates* necessários para que o usuário possa registrar, visualizar, editar e excluir suas fontes de renda.

**Implementação de *dashboard* personalizados (2 semanas):**

Criação de visualizações gráficas e painéis de controle, utilizando bibliotecas de gráficos, para que o usuário possa analisar seus padrões de gasto, distribuição por categoria e balanço mensal.

**Implementação dos alertas (2 semanas):**

Desenvolvimento do sistema de notificações para alertar o usuário quando ele estiver próximo de atingir os limites de orçamento definidos por categoria.

**Abertura do sistema para testes (2 semanas):**

Fase crítica em que o sistema será disponibilizado para validação. Contaremos com o auxílio de alunos voluntários do curso de Análise e Desenvolvimento de Sistemas (ADS) do IFRS - Campus Restinga, que testarão as funcionalidades em busca de falhas, inconsistências e melhorias de usabilidade.

**Correção de *bugs* (2 semanas):**

Período concorrente à fase de testes, dedicado à correção contínua de todos os *bugs* e inconsistências reportados pelo grupo de testadores.

**Atualização do diagrama de classes (1 semana):**

O diagrama de classes do projeto será revisado e atualizado para refletir a estrutura final do sistema, incluindo os novos modelos implementados.

**Produtização do sistema (1 semana):**

Preparação do sistema para o ambiente de produção, incluindo configurações de servidor, banco de dados, gestão de arquivos estáticos e o deploy final da aplicação.

**Defesa pública da Sessão final (1 semana):**

Após a conclusão de todas as etapas de desenvolvimento, documentação, testes e correções, a defesa final do trabalho será agendada para uma data a partir de 12 de Dezembro.

| **Atividades** | **Outubro** | **Novembro** | **Dezembro** |
|---|---|---|---|
| Implementação da leitura OCR de NF’s e NFe's | X | X | |
| Alterações e ajustes apontados na sessão prévia | X | X | |
| Adição do conteúdo ao documento | X | X | X |
| Implementação da adição/visualização das receitas | | X | |

| | | | |
|---|---|---|---|
| Implementação de *dashboard* personalizados | X | X | |
| Implementação dos alertas | | X | |
| Abertura do sistema para testes | | X | |
| Correção de *bugs* | | X | |
| Atualização do diagrama de classes | | X | X |
| Produtização do sistema | | | X |
| Defesa pública da Sessão final | | | X |

# 7. CONSIDERAÇÕES FINAIS

O Sistema de Gerenciamento de Despesas Pessoais foi desenvolvido como alternativa simples e confiável para organizar gastos sem exigir acesso a contas bancárias. O protótipo permite registrar despesas, classificá-las e consultar o histórico, criando bases para acompanhamento de orçamento.

As funcionalidades foram priorizadas a partir de necessidades comuns de usuários que valorizam privacidade. O MVP, construído em Django com autenticação via Google e interface responsiva, adota uma arquitetura enxuta e preparada para incrementos.

O principal diferencial está no registro por comprovantes: leitura de notas (OCR) e importação de XML de NF-e/NFC-e, reduzindo digitação e coletando apenas o estritamente necessário, em conformidade com a LGPD.

Entre os limites estão a sensibilidade do OCR à qualidade da imagem e a heterogeneidade dos layouts de XML. Como próximos passos, destacam-se: inclusão de receitas, orçamentos com alertas, *dashboards*, exportação/backup e testes com usuários; em etapa posterior, podem ser avaliadas integrações financeiras opcionais. Assim, o projeto entrega uma base sólida para evolução gradual sem abrir mão da simplicidade e da confiança do usuário.

# REFERÊNCIAS BIBLIOGRÁFICAS


ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 6023: informação e documentação: referências, elaboração. Rio de Janeiro: ABNT, 2002.

BOOTSTRAP. Bootstrap. Disponível em: https://getbootstrap.com/. Acesso em: 23 out. 2025.

BANCO CENTRAL DO BRASIL (BCB). ***Orçamento pessoal (Folhetos – Série II: Finanças Pessoais).*** Brasília: BCB, 2022. Disponível em: <Orçamento Pessoal>. Acesso em: 26 out. 2025.

BRASIL. COMISSÃO DE VALORES MOBILIÁRIOS (CVM). ***Programa Bem-Estar Financeiro: Módulo 3 — Controle financeiro***. Rio de Janeiro: CVM, 2018. Disponível em:<apostila-03.pdf>. Acesso em: 26 out. 2025.

BOOTSTRAP. **Forms**. Bootstrap v5.0 Docs, 2025. Disponível em: https://getbootstrap.com/docs/5.0/forms/overview/. Acesso em: 23 out. 2025.

CONFEDERAÇÃO NACIONAL DO COMÉRCIO DE BENS, SERVIÇOS E TURISMO (CNC). ***Pesquisa de Endividamento e Inadimplência do Consumidor (Peic) – agosto de 2025.*** CNC, 2025. Disponível em: <Relatório Peic ago25>. Acesso em: 26 out. 2025.

COMISSÃO DE VALORES MOBILIÁRIOS (CVM); ASSOCIAÇÃO BRASILEIRA DE PLANEJADORES FINANCEIROS. TOP: ***Planejamento financeiro pessoal***. Rio de Janeiro: CVM; Planejadores Financeiros, 2019. Disponível em: <livro_TOP_planejamento_financeiro_pessoal.pdf>. Acesso em: 26 out. 2025.

DJANGO-ALLAUTH. Django-allauth Documentation. Read the Docs, 2025. Disponível em: https://django-allauth.readthedocs.io/en/latest/. Acesso em: 11 out. 2025.

DJANGO SOFTWARE FOUNDATION. The web framework for perfectionists with deadlines. Django Software Foundation, 2025. Disponível em: https://www.djangoproject.com. Acesso em: 10 out. 2025.

DJANGO. **Validation in forms**. Django documentation, 2025a. Disponível em: https://docs.djangoproject.com/en/stable/ref/forms/validation/. Acesso em: 23 out. 2025.

DJANGO. **Creating forms from models**. Django documentation, 2025b. Disponível em: https://docs.djangoproject.com/en/stable/topics/forms/modelforms/. Acesso em: 23 out. 2025.

DJANGO. **Using the Django authentication system**. Django documentation, 2025c. Disponível em: https://docs.djangoproject.com/en/stable/topics/auth/default/. Acesso em: 23 out. 2025.

DJANGO. **Model Meta options: constraints**. Django documentation, 2025d. Disponível em: https://docs.djangoproject.com/en/stable/ref/models/options/#constraints. Acesso em: 23 out. 2025.

DJANGO. **Controlling database transactions**. Django documentation, 2025e. Disponível em: https://docs.djangoproject.com/en/stable/topics/db/transactions/. Acesso em: 23 out. 2025.

FUNPRESP-JUD. **90% dos brasileiros admitem ter necessidade de educação financeira**. Disponível em: https://www.funprespjud.com.br/90-dos-brasileiros-admitem-ter-necessidade-de-educacao-financeira/. Acesso em: 12 out. 2025.

**OAuth 2.0** ***Authorization Framework*****.** Disponível em: https://auth0.com/docs/authenticate/protocols/oauth. Acesso em: 12 out. 2025.

HTML. ***W3Schools HTML Tutorial*****.** Disponível em: https://www.w3schools.com/html/. Acesso em: 23 out. 2025.

INSTITUTO REÚNA. **Educação Financeira: A chave para um futuro igualitário**. Disponível em: https://o.institutoreuna.org/educacao-financeira/. Acesso em: 11 out. 2025.

JAVASCRIPT. ***MDN Web Docs: JavaScript.*** Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em: 23 out. 2025.

MOBILLS. **O aplicativo de controle financeiro mais bem avaliado do Brasil**. Mobills, 2025. Disponível em: https://www.mobills.com.br. Acesso em: 11 out. 2025.

ORGANIZZE. **Controle financeiro pessoal e empresarial**. Organizze, 2025. Disponível em: https://www.organizze.com.br. Acesso em: 11 out. 2025.

PYTHON SOFTWARE FOUNDATION. ***Welcome to Python.org.*** Disponível em: https://www.python.org/. Acesso em: 23 out. 2025.

SENADO FEDERAL. **Educação financeira: prevenção de dívidas começa na escola**. Info Matérias, 2025. Disponível em: https://www12.senado.leg.br/noticias/infomaterias/2025/09/educacao-financeira-prevencao-de-dividas-comeca-na-escola. Acesso em: 10 out. 2025.

CONFEDERAÇÃO NACIONAL DE DIRIGENTES LOJISTAS (CNDL). **Inadimplência de pessoas físicas bate novo recorde e atinge 71,86 milhões de consumidores no país, aponta CNDL e SPC Brasil. Brasília: CNDL**, 15 out. 2025. Disponível em: <CNDL - Confederação Nacional de Dirigentes Lojistas/>. Acesso em: 23 out. 2025.

LUCENA, Francisco. **App GuiaBolso ajuda a manter suas finanças organizadas no iOS**. TudoCelular, 2014. Disponível em: App GuiaBolso ajuda a manter suas finanças organizadas no iOS - Tudocelular.com. Acesso em: 10 out. 2025.

RIATO, Giovanna. **Conheça a história do GuiaBolso, o aplicativo de finanças pessoais com mais de 1 milhão de usuários**. Projeto Draft, 2015. Disponível em: Conheça a história do GuiaBolso, o aplicativo de finanças pessoais com mais de 1 milhão de usuários. Acesso em: 16 out. 2025.

W3C. ***Cascading Style Sheets (CSS).*** Disponível em: https://www.w3.org/Style/CSS/. Acesso em: 23 out. 2025.