INVESTLY – SIMULADOR DE RENTABILIDADE DE INVESTIMENTOS

Rayssa Roberta Rodrigues Silva[1]; Mateus Dutra Rezende Jardim[2]; Mateus de Souza Valente[3]

[1], [2], [3] Universidade de Uberaba

rayssarrsilva1@gmail.com, raul.rezende@uniube.br

Resumo

Este artigo apresenta o desenvolvimento de um sistema *web* voltado à simulação de investimentos em renda fixa. A aplicação permite ao usuário calcular a rentabilidade futura de uma aplicação e estimar o tempo necessário para atingir metas financeiras, considerando parâmetros ajustáveis como percentual do *CDI*, taxa *Selic*, taxa de administração e imposto de renda. O objetivo é oferecer uma ferramenta prática, educativa e acessível, capaz de apoiar a tomada de decisões no planejamento financeiro pessoal. A metodologia adotada baseou-se na utilização do *framework Django* e *Django REST Framework*, com banco de dados *SQLite*, autenticação *JWT* e interface responsiva desenvolvida com *HTML*, *Tailwind CSS* e *Bootstrap*. O sistema foi estruturado com base em boas práticas de arquitetura em camadas, com separação entre *models*, *views*, *serializers* e *templates*. Como resultado, obteve-se uma aplicação funcional e segura, que permite ao usuário simular diferentes cenários de investimentos e acompanhar o histórico das simulações realizadas. As funcionalidades implementadas demonstraram eficácia na tradução de cálculos financeiros complexos para uma interface clara e intuitiva. Conclui-se que a plataforma desenvolvida possui potencial significativo como recurso didático e de apoio à autonomia financeira, sobretudo em contextos educacionais ou de uso pessoal.

Palavras-chave: SIMULAÇÃO FINANCEIRA; JUROS COMPOSTOS; RENDA FIXA; PLANEJAMENTO FINANCEIRO.

1 Introdução

Grande parte da população brasileira enfrenta dificuldades ao tentar compreender o funcionamento de aplicações financeiras, especialmente em ambientes digitais. É comum que usuários se deparam com simuladores pouco didáticos, interfaces técnicas e termos financeiros como “percentual do CDI” ou “juros compostos”, sem a devida contextualização. Essa limitação na compreensão compromete a autonomia do investidor e, por consequência, a qualidade de suas decisões financeiras. Nesse contexto, o desenvolvimento de soluções tecnológicas que simplifiquem tais conceitos e tornem as simulações acessíveis a diferentes perfis de usuários torna-se uma estratégia relevante para a promoção da educação financeira.

A busca crescente por conhecimento financeiro, com vistas à conquista de maior autonomia, e o desejo de realizar aplicações como forma de garantir estabilidade e conforto pessoal a longo prazo, têm impulsionado o desenvolvimento de ferramentas

digitais voltadas à simulação de investimentos. Em um cenário econômico no qual os produtos de renda fixa se destacam pela previsibilidade e segurança, compreender o impacto de variáveis como a taxa Selic, o percentual do CDI e as taxas de administração torna-se essencial para que o investidor tome decisões mais conscientes e embasadas.

Segundo Saito e Saito (2017), a educação financeira no Brasil ainda é tratada de forma incipiente, sendo necessário o fomento de ações por parte do governo, da iniciativa privada e do terceiro setor para promover a capacitação financeira da população. Nesse contexto, a simulação de investimentos tende a assumir um papel didático e democratizador de conhecimento sobre investimentos financeiros. Estudos técnicos e acadêmicos evidenciam o papel fundamental da tecnologia como aliada na promoção da educação financeira, permitindo que usuários não especializados tenham acesso facilitado a conceitos historicamente restritos a profissionais do mercado.

O uso de aplicações web com simulações personalizáveis, interfaces responsivas e integração a parâmetros econômicos reais reforça essa tendência, promovendo tanto a inclusão digital quanto o desenvolvimento da autonomia financeira individual. Dados da ANBIMA (2023) revelam que mais de 60% dos brasileiros não compreendem o funcionamento de produtos básicos de renda fixa, o que evidencia a necessidade de soluções acessíveis que tornem esse conhecimento mais difundido.

Dessa forma, o desenvolvimento de soluções que possibilitem o cálculo da rentabilidade futura e a projeção do tempo necessário para o alcance de metas financeiras constitui uma iniciativa relevante, tanto para o campo técnico quanto para o educacional.

A justificativa para este trabalho fundamenta-se na crescente demanda por ferramentas acessíveis que auxiliem usuários sem conhecimentos prévios na compreensão dos mecanismos de rentabilidade em investimentos de renda fixa, utilizando parâmetros ajustáveis e atualizados. Vieira et al. (2019) desenvolveram e validaram um indicador de educação financeira, destacando a relevância do tema e a necessidade de avanços na sua implementação no Brasil. Conforme Angelos (2023), a integração de tecnologias digitais ao processo de aprendizagem potencializa a eficácia da educação financeira, tornando-a mais acessível e interativa. Ao transformar fórmulas financeiras em simulações interativas, almeja-se não apenas fornecer resultados, mas também capacitar o usuário, contribuindo para sua alfabetização financeira e para o fortalecimento da tomada de decisões fundamentadas.

Este trabalho delimita-se ao desenvolvimento de um sistema voltado especificamente para a simulação de investimentos em renda fixa, considerando o percentual do CDI/Selic anual — índices comumente utilizados na remuneração de investimentos de renda passiva —, além de oferecer ao usuário uma ferramenta prática, intuitiva e educativa de apoio à tomada de decisão financeira.

2 Materiais e Métodos

O sistema Investly foi desenvolvido com o objetivo de fornecer simulações de investimentos em renda fixa e realizar o cálculo do tempo de acumulação necessário para atingir uma meta financeira. A solução foi implementada com o framework Django e o

Django REST Framework, utilizando banco de dados relacional SQLite, autenticação JWT e renderização de templates responsivos com HTML, Tailwind CSS e Bootstrap.

A arquitetura adotada segue o padrão em camadas (MVC), organizando o projeto em componentes de apresentação (views/templates), regras de negócio (views e lógica interna) e acesso a dados (models e serializers). Essa separação favorece a manutenibilidade e clareza da estrutura. O fluxo de simulação inicia com o preenchimento do formulário, passa pelas views de cálculo e retorna os resultados ao template com base nas fórmulas aplicadas.

2.1 Configuração e estrutura do projeto

O projeto foi iniciado com o comando django-admin startproject investhere e a aplicação principal foi criada com startapp investimentos. As aplicações investimentos e accounts foram adicionadas ao INSTALLED_APPS, junto com rest_framework, rest_framework_simplejwt, drf_yasg e widget_tweaks. O banco de dados padrão foi mantido como SQLite para facilitar o desenvolvimento local. As URLs foram configuradas no projeto “investhere” e no app “investimentos”, incluindo rotas da aplicação e da API JWT.

Figura 1 - inclusão dos aplicativos no settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'investimentos',
    'rest_framework',
    'rest_framework_simplejwt',
    'drf_yasg',
    'accounts',
    'widget_tweaks',
]
```

Fonte: Elaborado pelos autores (2025).

Figura 2 - Configuração das urls no project investhere

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/token/', TokenObtainPairView.as_view(), name='token_obtain_pair'),
    path('api/token/refresh/', TokenRefreshView.as_view(), name='token_refresh'),
    path('', include('investimentos.urls')),  # Inclusão das URLs do app investimentos
    path('api/docs/', schema_view.with_ui('swagger', cache_timeout=0), name='swagger-docs'),
    path('api/redoc/', schema_view.with_ui('redoc', cache_timeout=0), name='redoc-docs'),
    path('api/logout/', LogoutView.as_view(), name='logout'),
    path('accounts/', include('accounts.urls')),  # Inclusão das URLs do app accounts
]
```

Fonte: Elaborado pelos autores (2025).

2.2 Modelagem de dados

Dois modelos principais foram criados, Investimento e SimulacaoHistorico.

Investimento: representa um investimento real, com nome, categoria, valor aplicado, rentabilidade anual, data de aplicação e prazo.

Figura 3 - Definição do modelo Investimento no arquivo models.py, com atributos, categorias pré-definidas e método de cálculo da rentabilidade mensal

```
class SimulacaoHistorico(models.Model):
    usuario = models.ForeignKey(User, on_delete=models.CASCADE, null=True, blank=True)
    tipo_simulacao = models.CharField(max_length=50)  # "Investimento" ou "Necessário"
    valor_inicial = models.DecimalField(max_digits=10, decimal_places=2, null=True, blank=True)
    rentabilidade_anual = models.DecimalField(max_digits=5, decimal_places=2)
    prazo_meses = models.IntegerField(null=True, blank=True)
    taxa_administracao = models.DecimalField(max_digits=5, decimal_places=2, null=True, blank=True)
    imposto_renda = models.DecimalField(max_digits=5, decimal_places=2, null=True, blank=True)
    valor_futuro = models.DecimalField(max_digits=10, decimal_places=2, null=True, blank=True)
    valor_desejado = models.DecimalField(max_digits=10, decimal_places=2, null=True, blank=True)
    valor_maximo = models.DecimalField(max_digits=10, decimal_places=2, null=True, blank=True)
    tempo_necessario = models.IntegerField(null=True, blank=True)
    valor_total = models.DecimalField(max_digits=10, decimal_places=2, null=True, blank=True)
    data_criacao = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Simulação de {self.usuario} - {self.tipo_simulacao}"
```

Fonte: Elaborado pelos autores (2025).

SimulacaoHistorico: armazena as simulações realizadas pelos usuários, tanto de valor futuro quanto de tempo de acumulação.

Figura 4 - Estrutura da classe SimulaçãoHistorico no arquivo models

```
from django.db import models
from django.contrib.auth.models import User

class Investimento(models.Model):
    CATEGORIAS = [
        ('renda_fixa', 'Renda Fixa'),
        ('renda_variavel', 'Renda Variável'),
        ('fundo_imobiliario', 'Fundo Imobiliário'),
        ('cripto', 'Criptomoeda'),
    ]

    usuario = models.ForeignKey(User, on_delete=models.CASCADE)
    nome = models.CharField(max_length=255)
    categoria = models.CharField(max_length=20, choices=CATEGORIAS)
    valor_investido = models.DecimalField(max_digits=10, decimal_places=2)
    rentabilidade_anual = models.DecimalField(max_digits=5, decimal_places=2)
    data_aplicacao = models.DateField()
    prazo_meses = models.IntegerField()

    def rentabilidade_mensal(self):
        """Calcula a rentabilidade mensal com base na anual."""
        return round(float(self.rentabilidade_anual) / 12, 2)

    def __str__(self):
        return f"{self.nome} - {self.usuario.username if self.usuario else 'Sem usuário'}"
```

Fonte: Elaborado pelos autores (2025).

Esses modelos estão relacionados ao usuário autenticado e foram registrados via *admin.py* e migrados com os comandos padrões do *ORM* do *Django*.

2.3 *Views* e regras de negócio

A lógica de funcionamento do sistema *Investly* foi implementada por meio de *views* específicas, organizadas de acordo com as funcionalidades principais da aplicação: simulações, autenticação e histórico de uso. Conforme Ferreira (2015), a arquitetura da informação tem papel central no delineamento eficiente das funcionalidades, contribuindo para uma estrutura mais compreensível e adaptável.

As principais *views* desenvolvidas incluem:

- *simulacao_investimento_view*: realiza o cálculo do valor futuro de um investimento, considerando fatores como CDI, Selic, prazo, taxa de administração e imposto de renda;
- *calcular_investimento_necessario_view*: estima o tempo necessário para atingir uma meta financeira com aportes mensais constantes;
- *historico_simulacoes_view*: exibe o histórico de simulações realizadas por cada usuário autenticado;
- *excluir_simulacao* e *excluir_todas_simulacoes*: permitem, respectivamente, a exclusão individual ou total dos registros salvos;
- *InvestimentoViewSet*: estrutura um *endpoint* protegido responsável pelas operações *CRUD* (*Create, Read, Update, Delete*) do modelo *Investimento* via API.

As funcionalidades de simulação utilizam fórmulas baseadas em juros compostos e equações logarítmicas para fornecer resultados precisos. Toda a lógica de negócio, bem como o controle de persistência dos dados, foi implementada no *backend*, garantindo integridade, segurança e centralização das regras do sistema.

2.4 *Templates* e interface

A camada de apresentação do sistema foi construída com base em arquivos *HTML* localizados no diretório *templates/investimentos*. Os principais *templates* utilizados são:

- *base.html*: estrutura base do sistema, contendo barra de navegação, cabeçalho, rodapé e *container* responsivo;
- *simulacao_form.html* e *simulacao_resultado.html*: responsáveis pela entrada de dados e exibição dos resultados de simulações de investimentos;
- *calcular_form.html* e *calcular_resultado.html*: voltados para o cálculo do tempo de acumulação de capital;
- *historico.html*: exibe de forma organizada o histórico de simulações realizadas pelo usuário;

- *login.html* e *registro.html*: formulários estilizados com *Bootstrap* para autenticação de usuários;
- *menu_principal.html*: tela inicial de navegação com acesso centralizado às funcionalidades principais.

Os formulários foram integrados ao pacote widget_tweaks, permitindo a customização dos campos diretamente nos templates por meio da inserção de classes CSS específicas. Essa abordagem proporcionou maior flexibilidade visual e compatibilidade com bibliotecas de estilo como Tailwind CSS e Bootstrap, sem a necessidade de sobrecarga no backend.

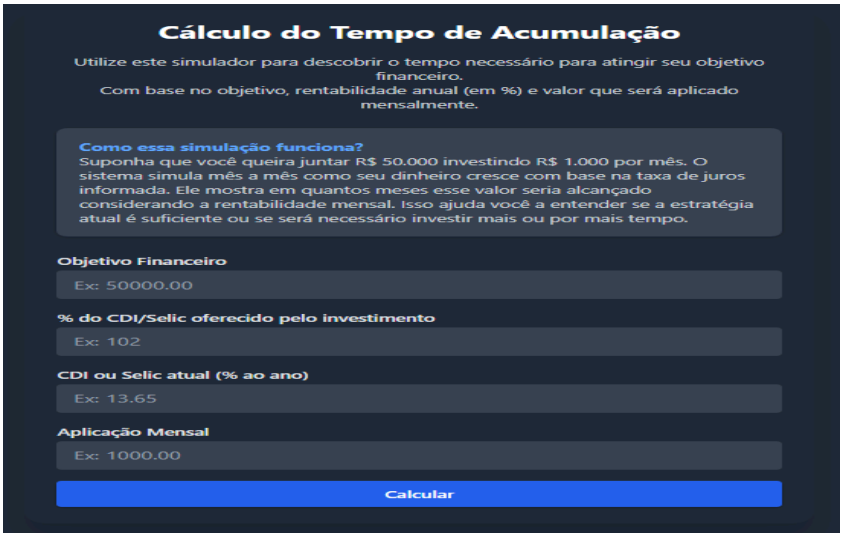
A escolha por Tailwind CSS e Bootstrap deve-se à combinação entre controle granular de estilos (fornecido pelo Tailwind) e componentes prontos de interface (disponibilizados pelo Bootstrap), otimizando o tempo de desenvolvimento, a responsividade e a padronização visual do sistema.

De acordo com Sarmento e Souza et al. (2004), a arquitetura da informação em ambientes digitais exerce papel central na eficiência da navegação e na clareza da interação usuário-sistema. Com isso, foi possível alcançar uma interface responsiva, clara e funcional, alinhada aos princípios de usabilidade e experiência do usuário.

Figura 5 - Interface de simulação da rentabilidade futura

Fonte: Elaborado pelos autores (2025).

Figura 6 - Interface de simulação de previsibilidade de tempo para alcance da meta financeira



Fonte: Elaborado pelos autores (2025).

Figura 7 - Interface visual do histórico de simulações

Fonte: Elaborado pelos autores (2025).

2.5 *Serializers* e autenticação *JWT*

Para estruturar a comunicação entre cliente e servidor, foi implementado o InvestimentoSerializer com o Django REST Framework. Esse componente converte instâncias do modelo em formatos como JSON e inclui o campo adicional rentabilidade_mensal, calculado a partir da taxa anual informada.

O uso de serializers centraliza a lógica de transformação dos dados, garantindo consistência e integridade nas operações da API, além de facilitar integrações com sistemas externos.

A autenticação foi implementada com JSON Web Tokens (JWT), por meio do pacote rest_framework_simplejwt, utilizando os endpoints TokenObtainPairView e TokenRefreshView. As views protegidas foram configuradas com a permissão IsAuthenticated, assegurando acesso exclusivo a usuários autenticados.

O funcionamento do JWT inicia com a autenticação do usuário, que ao inserir suas credenciais válidas, recebe um token de acesso e um token de refresh. O token de acesso tem validade temporária e é enviado nas requisições protegidas como forma de comprovar a identidade do usuário. O token de refresh permite a renovação automática da sessão, garantindo segurança sem comprometer a experiência do usuário. Essa abordagem segue boas práticas de segurança em aplicações web, mantendo a confidencialidade e integridade das informações financeiras tratadas.

Figura 8 - Estrutura da classe criada no serializer

```python
from rest_framework import serializers
from .models import Investimento

class InvestimentoSerializer(serializers.ModelSerializer):
    rentabilidade_mensal = serializers.SerializerMethodField()

    class Meta:
        model = Investimento #chama o nome da classe, modelo criado
        fields = '__all__' #inclui todos os campos

    @property
    def get_rentabilidade_mensal(self, obj):
        return obj.rentabilidade_mensal()
```

Fonte: Elaborado pelos autores (2025).

2.6 Equações utilizadas

As funcionalidades centrais do sistema Investly estão fundamentadas em fórmulas clássicas da matemática financeira, com foco na capitalização composta e na previsão de metas de investimento com aportes mensais constantes. A seguir, detalham-se as equações aplicadas em cada funcionalidade:

Valor Futuro com Juros Compostos (sem aportes mensais):
$VF = P \times (1 + r)^n$
VF: Valor Futuro do investimento
P: Valor aplicado inicialmente
r: Taxa de juros mensal líquida
n: Prazo da aplicação em meses

Conversão da taxa anual em taxa mensal:
r = (1 + CDI_anual / 100)^(1 / 12) - 1

Rentabilidade líquida mensal ajustada:
r_líquido = r × (Percentual_CDI / 100) × (1 - Taxa_Adm / 100) × (1 - IR / 100)

Tempo necessário para atingir uma meta com aportes mensais:
n = log[(FV × r / PMT) + 1] / log(1 + r)
n: Quantidade de meses para alcançar a meta
FV: Valor desejado (meta financeira)
PMT: Valor do aporte mensal
r: Taxa de juros mensal líquida

3 Resultados (ou resultados esperados)

O sistema *Investly* foi concebido com o objetivo de oferecer ao usuário duas simulações centrais: (1) o cálculo do valor futuro de um investimento, considerando parâmetros como percentual do *CDI*, taxa de administração e prazo; e (2) a estimativa do tempo necessário para alcançar um valor-alvo financeiro a partir de aportes mensais recorrentes. Ambas as funcionalidades foram desenvolvidas com base em fórmulas clássicas de juros compostos e expressões logarítmicas, assegurando projeções consistentes e fundamentadas do ponto de vista matemático.

Espera-se, como resultado, a entrega de simulações precisas e adaptáveis a diferentes cenários financeiros, contribuindo diretamente para o planejamento pessoal e a tomada de decisões estratégicas por parte do usuário. Na primeira simulação, o sistema estima o montante que o capital inicial poderá atingir ao longo do tempo, de acordo com os parâmetros fornecidos. Na segunda, calcula-se a quantidade de períodos (em meses) necessários para atingir determinada meta financeira, com base em um valor fixo aplicado mensalmente. Tais simulações permitem ao usuário explorar diferentes estratégias de investimento, promovendo uma compreensão mais ampla sobre o impacto dos juros compostos no longo prazo.

Além disso, o sistema registra o histórico de simulações realizadas, vinculando cada entrada ao respectivo usuário autenticado. Essa funcionalidade possibilita a análise de estratégias previamente adotadas, bem como o acompanhamento da evolução das metas financeiras ao longo do tempo. Dessa forma, a aplicação não se limita à geração de respostas numéricas pontuais, mas atua como uma ferramenta contínua de apoio à educação financeira e à organização pessoal do usuário.

Do ponto de vista técnico, destaca-se a correta implementação das fórmulas financeiras, a adoção de autenticação baseada em *JSON Web Tokens* (*JWT*) e a interface responsiva construída com *HTML*, *Tailwind CSS* e *Bootstrap*, fatores que contribuem significativamente para a segurança, usabilidade e confiabilidade da aplicação. A integração desses elementos técnicos garante a eficácia do sistema e sua aderência à proposta inicial.

Em síntese, os resultados esperados consolidam a proposta de um sistema web funcional, seguro e extensível, capaz de atender tanto usuários iniciantes quanto aqueles com maior familiaridade com o universo dos investimentos. A metodologia adotada no desenvolvimento do *Investly* revelou-se eficaz, resultando em uma solução alinhada às demandas atuais de planejamento financeiro, com potencial de evolução contínua a partir do *feedback* dos usuários e do aprofundamento do domínio temático.

4 Discussão

O desenvolvimento do sistema Investly proporcionou a consolidação de diversas práticas relacionadas ao desenvolvimento de aplicações web modernas, com foco na simulação financeira. A partir das funcionalidades implementadas, especialmente as simulações de rentabilidade futura e de tempo para alcance de metas financeiras, foi possível perceber o quanto ferramentas interativas e responsivas podem contribuir de forma direta para a autonomia do usuário na tomada de decisão em investimentos.

A arquitetura em camadas, aliada à separação de responsabilidades entre models, views, serializers e templates, evidenciou uma abordagem coerente com os padrões de projeto recomendados na literatura de engenharia de software. Ferreira (2015) destaca que aplicações com estrutura organizacional bem definida tendem a ser mais sustentáveis e eficientes, aspecto observado na implementação deste projeto. Ademais, Soares e Oliveira (2023) apontam que recursos digitais com função pedagógica, como jogos ou simuladores, têm se mostrado eficazes na mediação de conteúdos econômicos, corroborando a proposta do Investly.

A inserção de mecanismos de autenticação com JWT contribuiu diretamente para a segurança dos dados, demonstrando alinhamento com boas práticas no desenvolvimento de sistemas protegidos e segmentados por usuário.

A principal contribuição deste trabalho reside na capacidade de transformar cálculos financeiros tradicionalmente complexos em experiências acessíveis, visuais e interativas. Isso reforça não apenas o potencial da tecnologia como facilitadora do entendimento financeiro, mas também sua função pedagógica na formação de um usuário mais consciente e preparado para tomar decisões baseadas em dados.

Ao conectar os resultados obtidos com os objetivos iniciais do projeto, percebe-se que a solução atende à proposta de oferecer uma ferramenta funcional e didática, que alia robustez técnica à experiência do usuário.

Em comparação com simuladores financeiros oferecidos por instituições bancárias, como Caixa Econômica Federal e Nubank, o Investly diferencia-se por permitir maior personalização dos parâmetros de simulação, interface leve e foco educacional explícito. Enquanto simuladores bancários frequentemente restringem a interação a produtos próprios ou exigem login institucional, o Investly adota uma abordagem aberta e neutra, voltada à compreensão dos fundamentos do investimento em si, e não à venda de produtos.

Adicionalmente, observa-se que o caráter interativo do sistema, somado à simplicidade visual e à liberdade de testes, aproxima-se de estratégias de gamificação, ainda que de forma implícita. O incentivo ao usuário a experimentar diferentes cenários, modificar variáveis e comparar resultados reforça o engajamento contínuo e a aprendizagem autodirigida, características valorizadas em ambientes digitais com propósitos educacionais. Essa abordagem, mesmo sem elementos lúdicos explícitos, pode ser considerada um caminho viável para ampliar a adesão e o envolvimento do público-alvo.

5 Conclusão

O desenvolvimento do sistema Investly possibilitou a consolidação de práticas fundamentais do desenvolvimento de aplicações web, com ênfase em simulações de investimentos em renda fixa. As funcionalidades implementadas — especialmente as projeções de rentabilidade futura e tempo para alcance de metas financeiras — demonstraram o potencial de ferramentas digitais interativas no fortalecimento da autonomia do usuário em processos de tomada de decisão.

A integração entre cálculos financeiros e uma interface acessível ampliou a usabilidade da aplicação, favorecendo a compreensão de conceitos complexos e sua aplicação prática. A personalização de parâmetros, como percentual do CDI, prazo e valor-alvo, conferiu à plataforma flexibilidade para diferentes perfis e estratégias de investimento.

A adoção de autenticação baseada em JSON Web Tokens (JWT) demonstrou alinhamento com boas práticas de segurança digital, assegurando o controle de acesso e a proteção dos dados sensíveis (BUCKO et al., 2023). Arquiteturalmente, a separação entre models, views, serializers e templates reforçou a organização do projeto conforme preconiza a engenharia de software contemporânea.

Do ponto de vista pedagógico, o sistema reforça o papel da tecnologia na formação de um usuário mais consciente. Saito e Saito (2017) destacam que iniciativas digitais ampliam a autonomia do cidadão frente a decisões financeiras. Para Vieira et al. (2019), instrumentos como indicadores de educação financeira são essenciais à mensuração dessa efetividade. Angelos (2023) acrescenta que tecnologias educacionais bem estruturadas têm alto potencial transformador, e Costa e Silva (2021) destacam que plataformas digitais favorecem inclusão e autonomia econômica.

Conclui-se, portanto, que o Investly se apresenta como uma solução funcional, segura e didática, com forte aplicabilidade prática na educação financeira e no planejamento pessoal, e com potencial de expansão para públicos diversos e novos recursos. Entre as perspectivas futuras, destacam-se a integração com APIs públicas para atualização automática do CDI e da Selic, a inclusão de um comparador entre produtos de diferentes instituições financeiras e o desenvolvimento de uma versão mobile ou Progressive Web App (PWA), ampliando ainda mais o alcance e a acessibilidade da plataforma.

Referências

ANBIMA – ASSOCIAÇÃO BRASILEIRA DAS ENTIDADES DOS MERCADOS FINANCEIRO E DE CAPITAIS. *Raio X do investidor brasileiro 2023.* Disponível em: https://www.anbima.com.br. Acesso em: 15 maio 2025.

BUCKO, Ahmet; VISHI, Kamer; KRASNIQI, Bujar; REXHA, Blerim. *Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History.* Computers, Basel, v. 12, n. 4, p. 78, 2023. Disponível em: https://www.mdpi.com/2073-431X/12/4/78. Acesso em: 20 maio 2025.

COSTA, André Luiz; SILVA, Mariana Gomes da. *Tecnologia e educação financeira: o papel das plataformas digitais na democratização do conhecimento financeiro.* Revista de Educação e Tecnologia, São Paulo, v. 16, n. 2, p. 45-60, 2021.

DECIMAL MODULE – Python Documentation. *decimal — Decimal fixed point and floating point arithmetic.* Disponível em: https://docs.python.org/3/library/decimal.html. Acesso em: 02 junho 2025.

DJANGO SOFTWARE FOUNDATION. *Django (versão 4.x): The Web framework for perfectionists with deadlines.* Disponível em: https://www.djangoproject.com. Acesso em: 02 junho 2025.

FELDMAN, Scott et al. *Django REST framework: Web APIs for Django made easy.* Disponível em: https://www.django-rest-framework.org. Acesso em: 02 junho 2025.

FERREIRA, Guilherme Fischmann. *Arquitetura da informação no desenvolvimento de aplicação web.* Revista de Sistemas e Computação, Brasília, v. 5, n. 1, p. 10–20, 2015. Disponível em: https://bdm.unb.br/bitstream/10483/11044/1/2015_GuilhermeFischmannFerreira.pdf. Acesso em: 20 maio 2025.

JWT.IO. *JSON Web Tokens – Introduction to JWT.* Disponível em: https://jwt.io/introduction. Acesso em: 02 junho 2025.

MATH MODULE – Python Documentation. *math — Mathematical functions.* Disponível em: https://docs.python.org/3/library/math.html. Acesso em: 02 junho 2025.

PYTHON SOFTWARE FOUNDATION. *Python Programming Language – Versão 3.x.* Disponível em: https://www.python.org. Acesso em: 02 junho 2025.

RENDER. *Deploy a Django app – Render Documentation.* Disponível em: https://render.com/docs/deploy-django. Acesso em: 02 junho 2025.

SAITO, Juliana S.; SAITO, Ricardo. *A educação financeira no Brasil: um panorama a partir da Estratégia Nacional de Educação Financeira.* Revista de Administração Pública, Rio de Janeiro, v. 51, n. 6, p. 1002–1022, nov./dez. 2017. Disponível em: https://www.scielo.br/j/rap/a/XhqxBt4Cr9FLctVvzh8gLPb. Acesso em: 20 maio 2025.

SOARES, Viller Contarato; OLIVEIRA, Daniel de. *Jogos digitais em educação financeira: uma intermediação entre o mundo econômico e o mundo digital.* Revista Rease, v. 1, n. 1, p. 1–15, 2023. Disponível em: https://periodicorease.pro.br/rease/article/download/10370/4180/15770. Acesso em: 20 maio 2025.

SQLITE. *SQLite Database Engine – Lightweight, serverless SQL database engine.* Disponível em: https://www.sqlite.org. Acesso em: 02 junho 2025.

TAILWIND LABS. *Tailwind CSS – Rapidly build modern websites.* Disponível em: https://tailwindcss.com. Acesso em: 02 junho 2025.

VIEIRA, Fabiano P.; LOPES, Victor A.; MENDES, Wagner C. *Indicador de educação financeira: uma proposta para o contexto brasileiro.* Estudos Econômicos, São Paulo, v. 49, n. 4, p. 753–780, 2019. Disponível em: https://www.scielo.br/j/es/a/jpbGbNLJfVHBppfvQmVfH9R. Acesso em: 20 maio 2025.