



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SISTEMAS DE INFORMAÇÃO

Arthur da Silva

**Desenvolvimento de sistema de recomendação personalizado e otimizado de
carteira de ativos**

Florianópolis

2024

Arthur da Silva

**Desenvolvimento de sistema de recomendação personalizado e otimizado de
carteira de ativos**

Trabalho de Conclusão de Curso submetido ao curso de Sistemas de Informação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Jean Carlo Rossa Hauck Dr.

Florianópolis

2024

da Silva, Arthur
Desenvolvimento de sistema de recomendação
personalizado e otimizado de carteira de ativos / Arthur
da Silva ; orientador, Jean Carlo Rossa Hauck, 2024.
114 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro
Tecnológico, Graduação em Sistemas de Informação,
Florianópolis, 2024.

Inclui referências.

1. Sistemas de Informação. 2. Finanças. 3.
Investimento. 4. Desenvolvimento. 5. Sistema web. I.
Hauck, Jean Carlo Rossa. II. Universidade Federal de Santa
Catarina. Graduação em Sistemas de Informação. III.
Título.

Arthur da Silva

**Desenvolvimento de sistema de recomendação personalizado e otimizado de
carteira de ativos**

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel em Sistemas de Informação e aprovado em sua forma final pelo Curso de Graduação em Sistemas de Informação

Florianópolis, 08 de julho de 2024.

Coordenação do Curso

Banca examinadora

Prof. Jean Carlo Rossa Hauck, Dr.
Orientador

Prof. José Eduardo de Lucca, Me.
Universidade Federal de Santa Catarina

Prof. Ricardo Pereira e Silva, Dr.
Universidade Federal de Santa Catarina

RESUMO

Este trabalho apresenta o desenvolvimento do *ADASFINANCE*, um sistema de controle de carteira de ativos financeiros centrado nos objetivos de investimento do usuário que auxilie o investidor a monitorar seus ativos financeiros de maneira prática e simples - oferecendo uma alternativa às plataformas já existentes. A solução proposta consiste em uma aplicação web construída com as tecnologias LAMP (Linux, Apache, MySQL e PHP), além de telas minimalistas, porém objetivas, e gráficos interativos que utilizam bibliotecas e frameworks JavaScript como por exemplo Chart.js e Bootstrap 5. O sistema permite ao usuário cadastrar seus ativos financeiros e a partir disso definir objetivos de acordo com seu perfil de investimento. O sistema é capaz de indicar a respectiva ação a ser tomada para cada ativo, além de possibilitar ao usuário visualizar em formato de tabela as transações específicas por ativo, exibindo de forma gráfica as oscilações dos valores dos ativos no decorrer de determinados períodos.

Palavras-chave: Investimento; Aplicações web; PHP; MySQL; Carteira de ativos.

ABSTRACT

This paper presents the development of *ADASFINANCE*, a financial asset portfolio control system centered on user investment objectives, aimed at assisting investors in monitoring their financial assets in a practical and straightforward manner - providing an alternative to existing platforms. The proposed solution consists of a web application built with LAMP technologies (Linux, Apache, MySQL, and PHP), featuring minimalist yet objective screens and interactive charts using JavaScript libraries and frameworks such as Chart.js and Bootstrap 5. The system allows users to register their financial assets and define objectives according to their investment profile. It can suggest appropriate actions for each asset and enables users to visualize specific transactions per asset in tabular format, graphically displaying fluctuations in asset values over certain periods.

Keywords: Investment; Web applications; PHP; MySQL; Asset portfolio.

LISTA DE FIGURAS

Figura 1 - Página inicial da plataforma StatusInvest.....	24
Figura 2 - Detalhamento da carteira.....	24
Figura 3 - Detalhamento das ações da carteira.....	25
Figura 4 - Tela inicial do Fundamentei.....	26
Figura 5 - Detalhamento da carteira do Fundamentei.....	27
Figura 6 - Tela inicial do GorilaApp.....	28
Figura 7 - Tela da Minha Carteira do GorilaApp.....	28
Figura 8 - Arquitetura geral do sistema.....	39
Figura 9 - Diagrama Entidade-Relacionamento.....	41
Figura 10 - Exemplo de documento da criação da tabela de ativos.....	41
Figura 11 - Exemplo de documento da criação da tabela de usuários.....	41
Figura 12 - Exemplo de documento da criação da tabela de associativa entre usuários e ativos.....	42
Figura 13: Exemplo de documento da criação da tabela de transações.....	42
Figura 14: Exemplo de documento da criação da tabela de descrição textual das transações.....	42
Figura 15 - Classe index.php.....	43
Figura 16 - Função execute da classe Router.....	44
Figura 17 - Função routes da classe Router.....	45
Figura 18 - Função load da classe Router.....	45
Figura 19 - Classe Asset.....	46
Figura 20 - Classe ConnectionCreator.....	47
Figura 21 - Função query da classe RepositoryTrait.....	48
Figura 22 - Exemplo de insert básico da classe Transaction do módulo Repository.....	49
Figura 23 - Exemplo do controlador da classe AssetController, responsável por tratar tarefas relacionadas aos ativos.....	50
Figura 24 - Função processAssets da classe DatabasePopulator.....	52
Figura 25 - Trecho da função processAssets da classe DatabasePopulator.....	52
Figura 26 - Trecho da função processAssets da classe DatabasePopulator.....	53
Figura 27 - Trecho da função processAssets da classe DatabasePopulator.....	53
Figura 28 - Trecho da função processAssets da classe DatabasePopulator.....	53
Figura 29 - Função executInserts da classe DatabasePopulator.....	54
Figura 30 - Trecho de código da classe DatabasePopulator.....	54
Figura 32 - Exemplo de “componente” html especificado no arquivo asset-filter.html... 56	
Figura 33 - Exemplo de trechos de código que aplicam estilos em determinadas classes.....	57
Figura 34 - Trecho de código referente a implementação da função formatTable da classe main.js.....	58
Figura 35 - Trecho de código referente a implementação da função	

calculateObjectivePercentageDifference da classe main.js.....	59
Figura 36 - Trecho de código referente a implementação da função searchBySymbol da classe main.js, no exemplo estão sendo configurados os eventos “focus” e “blur” para os elementos que possuem a classe “objective-percentage”.....	60
Figura 37 - Trecho de código referente a implementação da função searchBySymbol da classe main.js.....	60
Figura 38 - Exemplo de função da classe main.js que cria um gráfico utilizando a biblioteca Chart.js.....	61
Figura 39 - Trecho de código referente a implementação da função searchBySymbol da classe main.js.....	61
Figura 40 - Tela de login e cadastro.....	62
Figura 41 - Tela de início.....	63
Figura 42 - Tela de inclusão de ativo.....	63
Figura 43 - Tela de compra e venda de ativo.....	64
Figura 44 - Tela de gráfico e transações.....	64
Figura 45 - Diagrama de deployment.....	66

LISTA DE QUADROS

Quadro 1 – Requisitos Funcionais.....	29
Quadro 2 – Requisitos Não-Funcionais.....	30
Quadro 3 – História de Usuário 01.....	33
Quadro 4 – História de Usuário 02.....	33
Quadro 5 – História de Usuário 03.....	35
Quadro 6 – Lista dos participantes do estudo.....	63
Quadro 7 – Respostas compiladas dos participantes do questionário autoral.....	69
Quadro 8 – História de Usuário 04.....	74
Quadro 9 – História de Usuário 05.....	75
Quadro 10 – História de Usuário 06.....	76
Quadro 11 – História de Usuário 07.....	78
Quadro 12 – História de Usuário 08.....	80
Quadro 13 – História de Usuário 09.....	82
Quadro 14 – História de Usuário 10.....	84
Quadro 15 – História de Usuário 11.....	85

LISTA DE TABELAS

Tabela 1 – Respostas Participante 1	65
Tabela 2 – Respostas Participante 2	66
Tabela 3 – Respostas Participante 3	66
Tabela 4 – Respostas Participante 4	67
Tabela 5 – Respostas Participante 5	67
Tabela 6 – Valor global final SUS	68

LISTA DE ABREVIATURAS E SIGLAS

CDI	Certificado de Depósito Interbancário
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
FII	Fundo de Investimento Imobiliário
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
LAMP	Linux, Apache, MySQL, PHP
ORM	Object-Relational Mapping
PDO	PHP Data Objects
PHP	Hypertext Preprocessor
SQL	Structured Query Language
SUS	System Usability Scale

SUMÁRIO

1 INTRODUÇÃO.....	14
1.1 OBJETIVOS.....	15
1.1.1 Objetivo Geral.....	15
1.1.2 Objetivos Específicos.....	15
1.2 ABORDAGEM METODOLÓGICA.....	16
2 FUNDAMENTAÇÃO TEÓRICA.....	18
2.1 DESENVOLVIMENTO WEB.....	18
2.1.1 Backend e Frontend.....	18
2.1.1.1 HTML.....	18
2.1.1.2 CSS.....	19
2.1.1.3 JavaScript.....	20
2.1.1.4 PHP.....	20
2.2 CARTEIRAS DE INVESTIMENTO.....	20
3 TRABALHOS RELACIONADOS.....	23
3.1 STATUSINVEST.....	23
3.2 FUNDAMENTEI.....	25
3.3 GORILAAPP.....	27
3.4 CONSIDERAÇÕES FINAIS.....	29
4 DESENVOLVIMENTO.....	30
4.1 LEVANTAMENTO E ANÁLISE DOS REQUISITOS.....	30
4.1.1 Histórias de Usuários.....	34
4.2 DEFINIÇÃO DA ARQUITETURA.....	38
4.3 MODELAGEM DE BANCO DE DADOS.....	40
4.4 IMPLEMENTAÇÃO.....	42
4.4.1 Camada de aplicação (backend).....	43
4.4.1.1 Módulo de roteamento.....	43
4.4.1.2 Módulo de entidades.....	45
4.4.1.3 Módulo de consulta e persistência.....	47
4.4.1.4 Módulo de controle.....	49
4.4.1.5 Módulo de serviços.....	50
4.4.1.6 Módulo de integração com APIs externas.....	51
4.4.1.6.1 HG Finance.....	52
4.4.2 Camada de apresentação (frontend).....	54
4.4.2.1 HTML e CCS.....	55
4.4.2.2 JavaScript.....	57
4.4.2.3 Chart.js.....	62
4.4.3 Telas.....	62
4.5 VISÃO DE IMPLANTAÇÃO.....	65
5 AVALIAÇÃO DA FERRAMENTA.....	66

5.1 DEFINIÇÃO DA AVALIAÇÃO.....	66
5.2 PARTICIPANTES DA AVALIAÇÃO.....	67
5.3 PROCEDIMENTOS DA AVALIAÇÃO.....	68
5.4 RESULTADOS DA AVALIAÇÃO.....	68
5.4.1 Cálculo da pontuação do Instrumento SUS e resultados.....	69
5.4.2 Resultados do questionário autoral.....	73
5.5 COMPARAÇÃO.....	76
6 CONCLUSÃO.....	78
REFERÊNCIAS.....	80
APÊNDICE I - Histórias de usuários 04 a 11.....	82
APÊNDICE II - Respostas do aplicação do SUS e do questionário autoral.....	95
APÊNDICE III - Código Fonte.....	105
APÊNDICE IV - Artigo.....	106

1 INTRODUÇÃO

No cenário financeiro contemporâneo, a busca por estratégias de investimento eficientes tornou-se imperativa diante da complexidade e da constante evolução dos mercados globais. A interconexão entre as economias, a volatilidade dos ativos e as mudanças rápidas nas condições macroeconômicas acrescentam uma camada adicional de desafios para os investidores. Existem diversas opções de investimentos nesse cenário, tanto no mercado nacional como internacional. Opções como renda variável, renda fixa, fundos imobiliários (FIIs) e criptoativos, por exemplo, podem compor o portfólio do investidor.

No que tange a presença do mercado de ações nesse cenário, o Brasil, especificamente, tem sido colocado em destaque por parte de investidores e empresas, uma vez que o país tem se apresentado como uma oportunidade para investidores externos que visam diversificar seus portfólios (Nunes *et al.*, 2005).

Os investimentos em bolsa de valores têm sido associados às incertezas e grandes riscos, quando na verdade existem teorias que podem ser aprendidas e assimiladas por qualquer pessoa que deseje diversificar seus investimentos, potencializando resultados (Scottá dos Passos; Pinheiro, 2010). Atualmente, a quantidade de informações disponíveis para os investidores é vasta. É preciso analisar diversos fatores como indicadores econômicos, relatórios de empresas, notícias geopolíticas e tendências de mercado. A complexidade do ambiente financeiro é estimulada pela rápida disseminação de informações por meio de canais digitais e redes sociais, tornando crucial a habilidade de filtrar dados relevantes e tomar decisões informadas.

Nesse sentido, a carteira de investimentos é um conjunto de ativos de investimentos criado através da negociação de títulos existentes. A utilização dos rendimentos pode ser aplicada na compra de novos títulos, bem como o uso do investimento em fundos para aumentar o tamanho da carteira ou na venda de títulos para diminuí-la (Bodie *et al.*, 2014). A carteira então, é um atributo fundamental, pois se refere a essa combinação específica de ativos, como ações, títulos e outros instrumentos financeiros detidos pelo investidor.

A diversificação é a chave para construir uma carteira robusta, distribuindo os investimentos entre diferentes classes de ativos e setores, com o objetivo de reduzir o risco, uma vez que a inclusão de vários títulos diferenciados numa carteira

permite, em geral, reduzir o risco da carteira relativamente ao risco médio dos ativos que a compõem devido à diversificação dos riscos, otimizando assim o retorno (Mendes; Abreu, 2006). A composição da carteira é essencialmente personalizada, refletindo os objetivos financeiros, o perfil de risco e as preferências do investidor. Ferramentas como GorilaAPP, Fundamentei e StatusInvest são plataformas voltadas para o acompanhamento e gestão de investimentos que oferecem diversas funcionalidades para facilitar a vida do investidor, mas são limitadas em fornecer informações objetivas em relação às ações necessárias para fazer o balanceamento da carteira do usuário.

Tendo em vista isso, a era digital trouxe consigo uma explosão de dados e uma acessibilidade sem precedentes às informações financeiras, criando um ambiente no qual a tomada de decisões de investimento exige análises precisas e ainda mais personalizadas, o que carece de disponibilidade no mercado atual. Assim, este trabalho propõe o desenvolvimento de um sistema de gestão de carteira de ativos de acordo com os objetivos do investidor, a fim de facilitar a tomada de decisão nas operações de compra e venda.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Este trabalho tem como objetivo o desenvolvimento de um sistema de controle de carteira de ativos financeiros centrado nos objetivos de investimento do usuário.

1.1.2 Objetivos Específicos

- a) analisar os trabalhos correlatos;
- b) levantar e analisar os requisitos de um sistema web de controle de carteira de ativos financeiros;
- c) modelar e implementar um sistema web de controle de carteira de ativos financeiros;
- d) avaliar o sistema web de controle de ativos financeiros desenvolvido.

1.2 ABORDAGEM METODOLÓGICA

Este trabalho consiste em uma pesquisa aplicada e tecnológica, pois foca no desenvolvimento de um sistema de software. Assim, é estabelecida uma abordagem focada no desenvolvimento de software. Na sequência, os passos metodológicos são apresentados.

Na primeira etapa, será realizada a análise dos trabalhos relacionados, outros sistemas atualmente desenvolvidos e disponíveis para uso e que permitem a gerência de carteira de ativos. A pesquisa de trabalhos relacionados consistirá em buscar e identificar sistemas e estudos existentes sobre a gestão de carteiras de ativos financeiros. Em seguida, será realizada a extração de dados desses trabalhos, em que serão coletadas informações detalhadas e específicas dos sistemas identificados, como por exemplo suas funcionalidades. Por fim, os sistemas existentes serão avaliados criticamente para identificar suas vantagens e limitações, visando incorporar as melhores práticas no novo desenvolvimento.

Em seguida, será definida a engenharia de requisitos, que consiste em coletar e analisar os requisitos de uma ferramenta de controle de carteira de ativos financeiros. Nela, será realizado o levantamento de requisitos, etapa que envolve a identificação e documentação das necessidades e expectativas do usuário, bem como a análise de requisitos, processo que visa transformar os requisitos levantados em especificações claras e detalhadas, que orientarão o desenvolvimento do sistema.

Na etapa de modelagem do sistema, após a definição da engenharia de requisitos, uma proposta de solução será apresentada sob a forma de modelos. Esta elaboração da proposta de solução consiste em representar graficamente como o sistema irá funcionar e como os requisitos serão implementados.

Por conseguinte, inicia-se a modelagem da arquitetura do sistema. Nesta fase, são definidos os componentes principais do sistema, suas interações e como eles se relacionam para atender aos requisitos identificados. Por fim, a última etapa da engenharia de requisitos consiste no detalhamento da proposta elaborada durante a modelagem do sistema. Nesta fase, todos os aspectos da solução proposta são refinados e documentados em maior profundidade, fornecendo um guia completo para o desenvolvimento do sistema. Isso inclui a especificação detalhada de cada requisito identificado, descrevendo suas funcionalidades,

comportamentos esperados, entradas e saídas, além de quaisquer restrições ou considerações especiais.

A implementação do sistema será realizada seguindo uma abordagem ágil, que permite o desenvolvimento iterativo e incremental do software, facilitando adaptações rápidas às necessidades e feedbacks dos usuários. No desenvolvimento do sistema, o foco está na construção do sistema de software com base nos requisitos e na arquitetura previamente definidos. Durante essa fase, serão realizadas atividades de codificação, integração de componentes e implementação das funcionalidades conforme especificadas na engenharia de requisitos.

Por fim, a etapa de avaliação do sistema envolve a análise crítica do software desenvolvido por meio de um estudo de caso, permitindo que potenciais usuários testem e forneçam feedback sobre o sistema. O primeiro passo na avaliação do sistema é identificar os potenciais usuários que participarão do estudo de caso. A seleção deve ser criteriosa para garantir que os participantes tenham experiência relevante e possam fornecer feedback significativo. Com os usuários identificados, a avaliação é conduzida por meio da aplicação do instrumento *SUS* e de um questionário autoral. Após a realização da avaliação, os dados coletados são analisados para identificar pontos fortes e fracos do sistema. A análise envolve a compilação e interpretação do feedback dos usuários, destacando aspectos como facilidade de uso, eficiência das funcionalidades e possíveis problemas ou limitações. Com base nesta análise, são elaboradas recomendações para melhorias no sistema, garantindo que ele atenda de forma eficaz às necessidades dos usuários finais antes de sua implementação definitiva.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os principais conceitos relacionados ao desenvolvimento deste trabalho. Inicialmente são apresentados os conceitos relacionados ao desenvolvimento de sistemas WEB, seguidos dos conceitos relacionados a carteiras de investimentos.

2.1 DESENVOLVIMENTO WEB

Uma aplicação web é um software que é acessado e executado através de um navegador de internet, utilizando tecnologias e padrões da web, como HTML, CSS e JavaScript. De acordo com Meloni (2018), a combinação de HTML para estrutura, CSS para estilo e JavaScript para comportamento permite a criação de interfaces dinâmicas e responsivas. Diferente de softwares tradicionais que precisam ser instalados no computador, as aplicações web são hospedadas em servidores remotos e podem ser acessadas de qualquer dispositivo com conexão à internet. Elas podem variar desde simples páginas de informação até complexos sistemas de gestão empresarial. Como destaca Fowler (2020), a arquitetura de aplicações web modernas frequentemente inclui componentes de frontend interativos que se comunicam com serviços de backend para fornecer funcionalidades robustas e escaláveis.

As aplicações web interagem com os usuários através de interfaces amigáveis e dinâmicas, e muitas vezes se comunicam com servidores de backend para processar dados e executar funções complexas. Exemplos comuns incluem sistemas de gerenciamento de conteúdo, plataformas de comércio eletrônico, redes sociais e ferramentas colaborativas. A principal vantagem das aplicações web é a sua acessibilidade e capacidade de atualização centralizada, permitindo melhorias e correções de forma eficiente e rápida.

2.1.1 Backend e Frontend

2.1.1.1 HTML

HTML, sigla para *Hypertext Markup Language*, que, em português, significa linguagem para marcação de hipertexto, foi inventada por Sir Tim Berners-Lee e é a linguagem padrão utilizada para criar e estruturar páginas na web (Silva, 2019). Ele define a estrutura básica de um documento web através de uma série de elementos e tags, que descrevem o conteúdo e sua organização. Esses elementos podem incluir títulos, parágrafos, links, imagens, tabelas e formulários, entre outros. Cada tag em HTML tem um propósito específico e pode ser aninhada para criar layouts complexos. HTML é uma linguagem de marcação, o que significa que ele utiliza tags para delinear diferentes partes do conteúdo, diferentemente de linguagens de programação tradicionais. Além disso, HTML trabalha em conjunto com CSS e JavaScript para estilizar e adicionar interatividade às páginas web. O conhecimento de HTML é fundamental para desenvolvedores web, pois é a base sobre a qual todas as outras tecnologias web se constroem. Com o advento do HTML5, novas funcionalidades e tags foram introduzidas, permitindo uma maior semântica e capacidades avançadas, como suporte para áudio, vídeo e gráficos interativos.

2.1.1.2 CSS

CSS, sigla para Cascading Style Sheets, é uma linguagem de estilo utilizada para descrever a apresentação de documentos escritos em HTML ou XML. Com CSS é possível criar um documento de folha de estilo que especifica as fontes, cores, espaçamentos e outras características vinculando todas as páginas que devem ter essa aparência à folha de estilo, em vez de especificar todos esses estilos repetidamente em cada documento separado. Assim, uma folha de estilo é um agrupamento de instruções de formatação que controla a aparência de várias páginas HTML de uma só vez (Meloni, 2012). As regras de estilo em CSS são aplicadas através de seletores, que apontam para os elementos HTML aos quais os estilos devem ser aplicados. Essa separação de preocupações facilita a manutenção e a atualização do design das páginas, permitindo que o mesmo conjunto de estilos seja reutilizado em múltiplas páginas. Além disso, CSS permite criar layouts responsivos, que se adaptam a diferentes dispositivos e tamanhos de tela, proporcionando uma melhor experiência ao usuário.

2.1.1.3 JavaScript

O JavaScript foi desenvolvido pela Netscape Communications Corporation, a criadora do navegador web Netscape, e foi a primeira linguagem de script para web a ser suportada por navegadores e ainda é de longe a mais popular (Meloni, 2012). JavaScript é uma linguagem de programação de alto nível, interpretada pelos navegadores (Chrome, Mozilla Firefox etc), utilizada para tornar as páginas web interativas e dinâmicas. Com JavaScript, é possível manipular elementos HTML, responder a eventos do usuário, criar animações, validar formulários e fazer requisições para servidores, possibilitando uma ampla gama de funcionalidades, desde a criação de simples scripts até o desenvolvimento de aplicações web completas e complexas.

2.1.1.4 PHP

PHP, sigla para *Hypertext Preprocessor*, é uma linguagem de programação de código aberto amplamente utilizada para o desenvolvimento web. Introduzida em 1995, ela é especialmente eficaz para a criação de páginas dinâmicas e interativas, funcionando ao lado do servidor. PHP pode ser embutido no HTML, o que facilita a integração com conteúdo web e a criação de sites robustos (Converse; Park, 2003). A linguagem é altamente versátil e compatível com diversos bancos de dados, como MySQL, PostgreSQL e SQLite, além de suportar diversos protocolos, como HTTP e FTP. Sua simplicidade e ampla documentação tornam o PHP uma escolha popular tanto para desenvolvedores iniciantes quanto para projetos complexos. Além disso, a comunidade ativa de desenvolvedores contribui com uma vasta gama de bibliotecas e frameworks, como Laravel e Symfony, que expandem suas capacidades e aceleram o desenvolvimento de aplicações.

2.2 CARTEIRAS DE INVESTIMENTO

Uma carteira compreende um conjunto de ativos cujos valores de mercado variam conforme as empresas que a compõem, o que pode reduzir ou aumentar o risco da carteira (KATO, 2004, apud Bach et al., 2015). A seleção de carteiras

eficientes visa maximizar o retorno esperado, e a Teoria dos Portfólios, desenvolvida por Markowitz (1952), busca gerar a máxima eficiência de uma carteira por meio da diversificação. O objetivo é obter a melhor relação entre risco e retorno, diminuindo o risco da carteira enquanto se mantém ou se aumenta o retorno (Bach et al., 2015).

Elas são estruturadas para diversificar os investimentos e, assim, minimizar riscos, equilibrando a proporção de diferentes tipos de ativos, como ações, títulos e criptomoedas. O objetivo principal de uma carteira de investimento é otimizar o retorno ajustado ao risco, garantindo que os investidores possam alcançar suas metas financeiras de curto, médio e longo prazos. Para isso, é fundamental que as carteiras sejam monitoradas e ajustadas periodicamente, conforme as condições de mercado e as necessidades dos investidores.

Ativos financeiros são os meios pelos quais os indivíduos de economias bem desenvolvidas reivindicam seus direitos sobre ativos reais, ou também reivindicações pela renda gerada pelos ativos reais (ou exigibilidades sobre renda do governo) (BODIE; KANE; MARCUS, 2014). Eles podem ser comprados, vendidos ou negociados em mercados financeiros, sendo fundamentais para a alocação de recursos e a realização de investimentos. Assim, a escolha dos ativos financeiros adequados depende dos objetivos e do perfil de risco de cada investidor.

Os tipos de ativos financeiros são variados, mas incluem principalmente os títulos de renda fixa ou de dívida, ações e derivativos (BODIE; KANE; MARCUS, 2014). As ações representam uma participação no capital social de empresas, oferecendo potencial de valorização e dividendos. De acordo com os autores, os títulos de dívida, como bonds e debêntures, são instrumentos através dos quais os investidores emprestam dinheiro a governos ou empresas, recebendo juros em troca. Derivativos, como opções e futuros, são contratos cujo valor deriva de ativos subjacentes, sendo utilizados para hedge ou especulação. Além desses, há também moedas e criptomoedas, que são ativos financeiros com valor baseado em sua demanda no mercado. Cada tipo de ativo possui características específicas que influenciam seu comportamento e o perfil de risco-retorno para os investidores.

A estratégia de rebalanceamento de carteira envolve a revisão e readequação da composição de ativos, com a finalidade de preservar a alocação planejada inicialmente, de acordo com os objetivos de investimento, a expectativa de retorno e a tolerância a riscos do investidor, e esse processo visa corrigir os desequilíbrios causados pelas oscilações de mercado, por meio da venda de ativos valorizados e

compra de ativos desvalorizados, mantendo a carteira alinhada às metas estabelecidas (Stumpf, 2024). Essa prática ajuda a controlar o risco e a maximizar o retorno ao longo do tempo, garantindo que a carteira permaneça adequada às circunstâncias e objetivos do investidor.

3 TRABALHOS RELACIONADOS

Como este trabalho propõe desenvolver um sistema de gerenciamento de ativos de acordo com os objetivos do usuário, neste capítulo são apresentados alguns sistemas com funcionalidades semelhantes disponíveis na internet. Apesar de existirem diversas e variadas ferramentas no mercado com função parecida, as três escolhidas possuem pontos fortes e fracos, assim como algumas limitações quanto a resolução do problema a ser resolvido proposto neste trabalho.

São analisados os seguintes sistemas similares: StatusInvest, Fundamentei e GorilaApp. Os sistemas avaliados foram encontrados por meio de busca na ferramenta Google por sistemas de gestão de carteiras de investimento e foram selecionados devido à sua similaridade de funcionalidades com a proposta deste trabalho.

3.1 STATUSINVEST

O StatusInvest é um site de investimentos completo e robusto, com inúmeras funcionalidades relacionadas a investimentos, que fornece um banco de dados com vários indicadores sobre ações, fundos de investimento, BDRs e mais. Nele é possível encontrar diversas informações contábeis, múltiplos das ações, distribuição de proventos e relatórios das empresas e fundos, além de acompanhar o desempenho dos principais indicadores e ativos do mercado, os eventos das empresas listadas em bolsa e informações dos títulos públicos (Vitor, 2024).

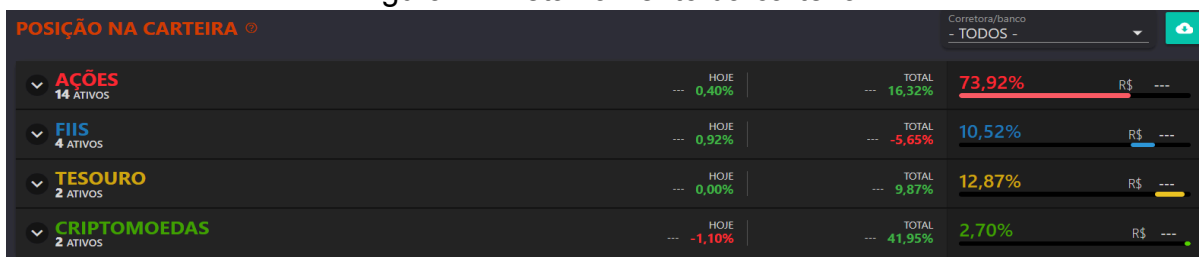
Figura 1 - Página inicial da plataforma StatusInvest



Fonte: StatusInvest

Além disso, o StatusInvest conta com diversas integrações automáticas, como por exemplo com a B3 (bolsa de valores brasileira) e exchanges de criptomoedas (como por exemplo o MercadoBitcoin).

Figura 2 - Detalhamento da carteira



Fonte: StatusInvest

Entre outras funcionalidades da plataforma, possui gráficos ilustrativos contendo seus investimentos categorizados de acordo com sua natureza (ações, tesouro, FIIS, etc), informações de rentabilidade total e parcial por dia ou período, informações de proventos, evolução de patrimônio em comparação com Ibovespa e CDI, composição da carteira de investimento e outras.

Figura 3 - Detalhamento das ações da carteira

ATIVO	PREÇO MÉDIO	PREÇO ATUAL	DIFERENÇA MÉDIO / ATUAL	QUANTIDADE	PATRIMÔNIO ATUAL	VARIÇÃO HOJE	VARIÇÃO TOTAL	% EM AÇÕES	% NA CARTEIRA
ABC4	---	R\$ 21,77	↑ 28,97%	---	---	↓ -0,91%	↑ 28,97%	5,76%	4,26%
AMER3	---	R\$ 1,14	↑ 4,59%	---	---	↑ 11,76%	↑ 4,59%	0,30%	0,22%
BBAS3	---	R\$ 51,32	↑ 20,24%	---	---	↑ 0,61%	↑ 20,24%	13,58%	10,03%
BBD4	---	R\$ 15,48	↑ 14,75%	---	---	↑ 0,06%	↑ 14,75%	4,09%	3,03%
BHIA3	---	R\$ 0,60	↓ -69,85%	---	---	↑ 1,69%	↓ -69,85%	0,32%	0,23%
BRBI11	---	R\$ 13,17	↑ 19,08%	---	---	↑ 0,00%	↑ 19,08%	3,48%	2,58%
ITSA4	---	R\$ 9,96	↑ 13,31%	---	---	↑ 0,00%	↑ 13,31%	13,17%	9,74%
ITUB4	---	R\$ 30,77	↑ 25,54%	---	---	↑ 0,07%	↑ 25,54%	8,14%	6,02%
MGLU3	---	R\$ 2,16	↓ -44,33%	---	---	↓ -5,26%	↓ -44,33%	1,14%	0,84%
MOV13	---	R\$ 10,82	↑ 56,36%	---	---	↓ -1,55%	↑ 56,36%	8,59%	6,35%

Fonte: StatusInvest

Além de poder puxar as informações de seus ativos de forma automática através das integrações disponíveis na plataforma, o usuário também pode inserir de forma manual qualquer transação que deseje.

Existem também opções pagas que exibem de forma mais completa indicadores e estatísticas de cada ativo de forma individual, além de análises feitas por especialistas que ajudam o usuário a manusear sua carteira com opções de compra e venda.

Pontos positivos:

- é uma das ferramentas mais completas do mercado para análise de ações, principalmente considerando as funcionalidades dos planos pagos;
- integrações automáticas com B3 e outras instituições financeiras.

Pontos negativos:

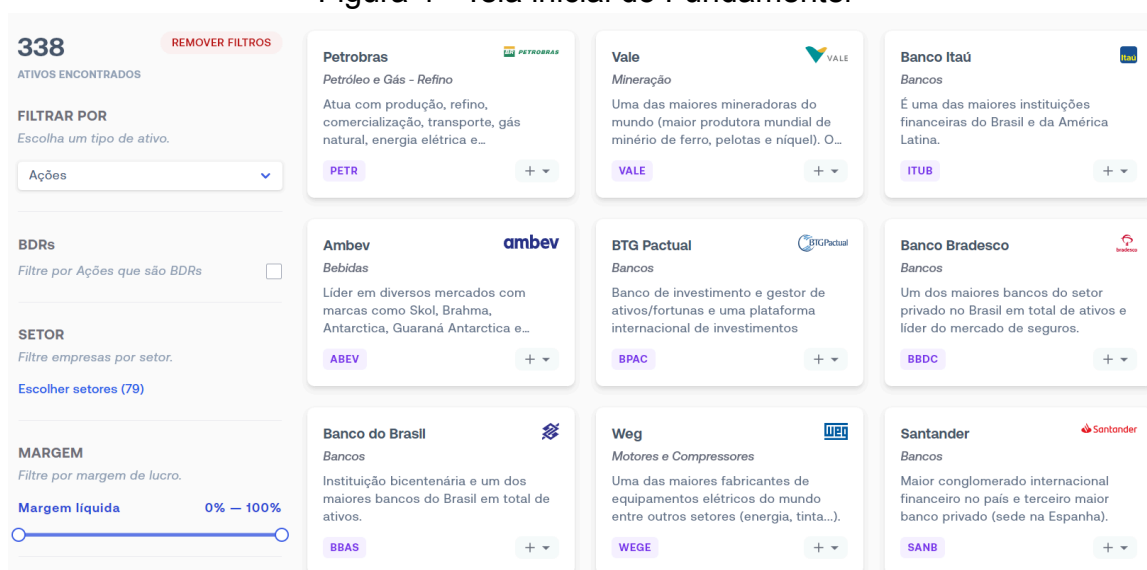
- não possui a possibilidade de edição da tabela de ativos a fim de informar sua meta de participação na carteira;
- por ter muitas funcionalidades a navegação pode ser um pouco mais complicada para usuários iniciantes e sem experiência.

3.2 FUNDAMENTEI

O Fundamentei é um site brasileiro que oferece serviços relacionados a investimentos e educação financeira, e que fornece ferramentas e recursos para

auxiliar investidores a tomarem decisões mais informadas sobre seus investimentos, além de oferecer conteúdo educacional sobre finanças pessoais e investimentos. Oferece a seus usuários, principalmente, serviços de informações com relação aos ativos listados na B3 - Brasil Bolsa Balcão S.A., The Nasdaq Stock Market e The New York Stock Exchange, não se restringindo unicamente a esta finalidade (Fundamentei, 2023). Possui uma interface amigável, além de informações na forma de tabelas e gráficos e é possível filtrar os resultados por alguns critérios, como setor e margem líquida da operação.

Figura 4 - Tela inicial do Fundamentei

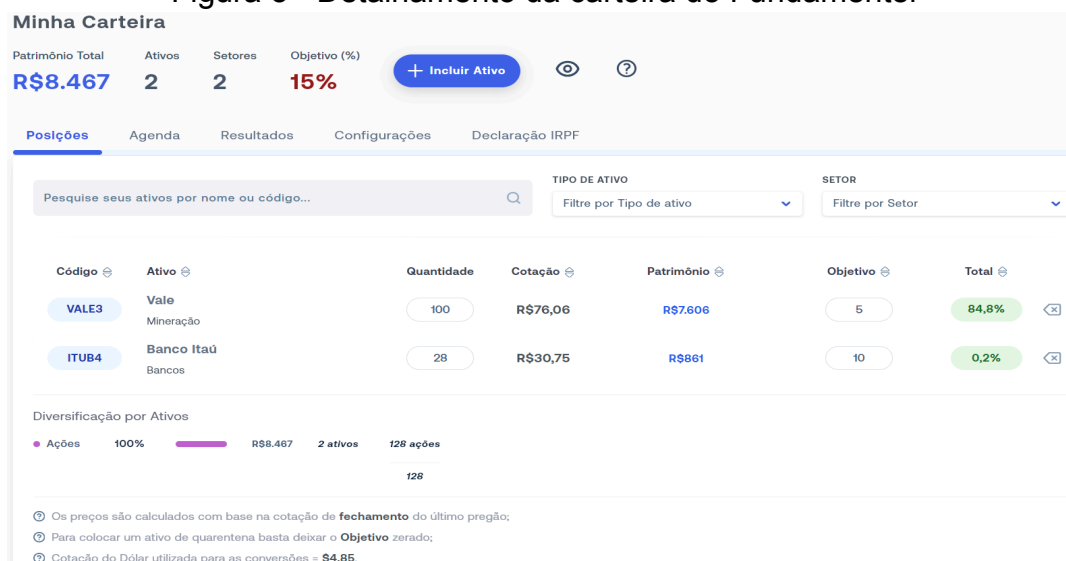


Fonte: Fundamentei

No que se refere às informações disponibilizadas, não é um site tão completo como o StatusInvest, contendo apenas com informações básicas como cotações históricas, e algumas informações contábeis, como patrimônio líquido e receita. Existe, também, a possibilidade de contratação de um plano premium, que fornece algumas informações adicionais dos ativos e a possibilidade de participação em grupos de discussão.

O diferencial do site é que ele possui uma calculadora de balanceamento de ações com diversas opções de customização. Dessa forma você pode atribuir pesos para cada ação na sua carteira e ele informa qual ação deve ser comprada para rebalancear a divisão de ativos da forma configurada.

Figura 5 - Detalhamento da carteira do Fundamentei



Fonte: Fundamentei

Pontos positivos:

- interface extremamente minimalista e polida, fácil de usar por qualquer usuário;
- permite ao usuário ponderar sua carteira com os valores relativos que deseja.

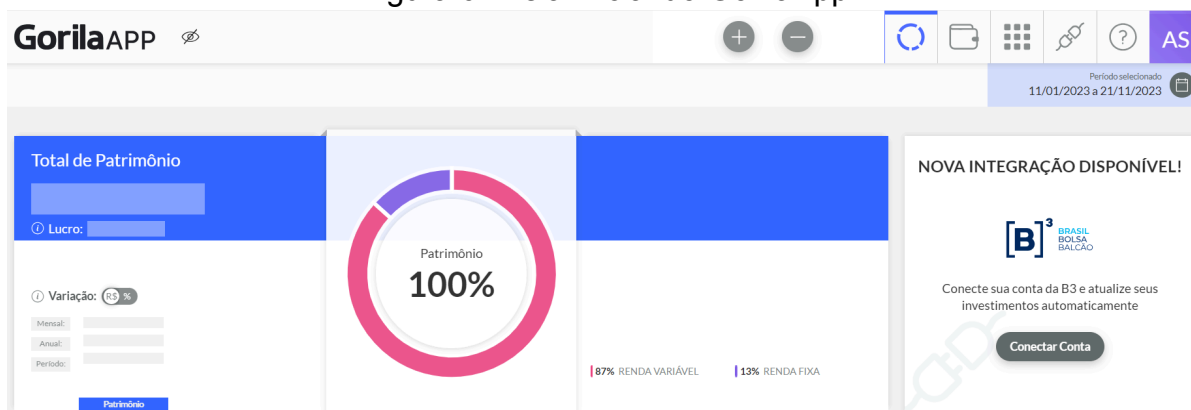
Pontos negativos:

- não possui integração automática com instituições financeiras;
- poucas informações sobre os ativos em relação aos concorrentes;
- poucas informações sobre seus ativos na planilha de exibição destes.

3.3 GORILAAPP

O GorilaAPP é uma plataforma brasileira que oferece serviços relacionados a investimentos e finanças pessoais e permite que os usuários acompanhem e analisem seus investimentos de forma integrada, consolidando informações de diferentes instituições financeiras em um só lugar. A plataforma ajuda a monitorar seus investimentos em tempo real e a otimizar sua carteira, oferecendo análises detalhadas e uma visão geral completa do seu portfólio de investimentos.

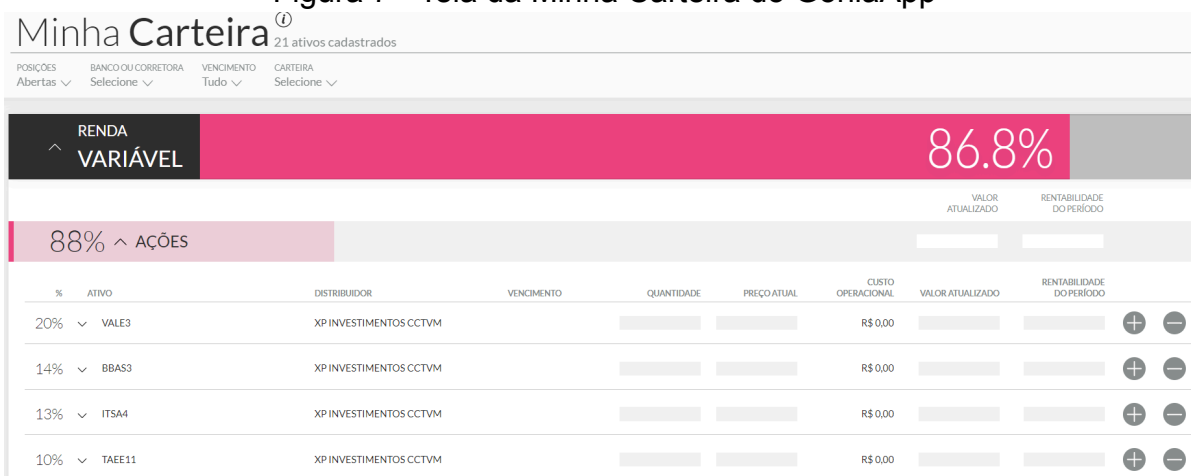
Figura 6 - Tela inicial do GorilaApp



Fonte: GorilaApp

A plataforma fornece dados confiáveis e atualizados sobre uma variedade de produtos financeiros, abrangendo desde renda fixa, ações, fundos imobiliários, ETFs, até outras alternativas de investimento. Além disso, é possível executar operações de compra e venda de ativos diretamente pelo site.

Figura 7 - Tela da Minha Carteira do GorilaApp



Fonte: GorilaApp

O aplicativo oferece uma variedade de ferramentas para os investidores, incluindo a capacidade de simular investimentos. Essa função possibilita a avaliação do desempenho da carteira em cenários econômicos diversos, auxiliando o usuário a adaptar sua estratégia diante das flutuações do mercado e a atingir seus objetivos financeiros.

O GorilaAPP ainda conta com uma seção inteira para análise de ativos, contendo informações completas para auxiliar o investidor na análise de risco e desempenho da sua carteira. Além desta seção do GorilaAPP, existe também o

GorilaFLOW (um módulo separado apenas para análise estatística e gráfica dos ativos de forma contextualizada).

Pontos positivos:

- a) interface extremamente polida e simples;
- b) possui integração com a B3 e a corretora XP.

Pontos negativos:

- a) não possui a possibilidade de edição da tabela de ativos a fim de informar sua meta de participação na carteira;
- b) extremamente limitado em funções, apenas exibe informações básicas sobre os ativos.

3.4 CONSIDERAÇÕES FINAIS

As ferramentas analisadas apresentam um conjunto relevante de funcionalidades para gestão de carteira de investimentos, além de diversos pontos positivos. No entanto, foram identificadas limitações nas ferramentas, por meio dos seus pontos negativos, que indicam a relevância do desenvolvimento proposto neste trabalho.

4 DESENVOLVIMENTO

Este capítulo apresenta a ferramenta ADAS FINANCE, um sistema de gerenciamento de ativos integrado a uma calculadora de balanceamento de investimento de acordo com os objetivos do usuário, que permite que o usuário acompanhe de forma simplificada seus investimentos, auxiliando o investidor, de acordo com seus objetivos pré estabelecidos, no processo de negociação de ações, criptomoedas e fundos imobiliários. A ferramenta ADAS FINANCE é composta por funções que permitem o usuário usufruir das funcionalidades da plataforma. São elas:

- a) configurar o valor percentual do objetivo, o que permite ao usuário definir a porcentagem de metas financeiras dentro da plataforma, ajudando no planejamento e acompanhamento dos objetivos de investimento;
- b) cadastrar, editar e deletar ativos, bem como visualizá-los, de modo que o usuário possa adicionar novos ativos financeiros, como ações, títulos e entre outros, fazer alterações ou remover ativos existentes, além de poder visualizar todos os ativos cadastrados;
- c) registrar novas transações financeiras, como compras e vendas de ativos, dentro da plataforma e;
- d) visualizar gráfico de ações e o histórico de transações, ao passo que o usuário pode acessar gráficos que mostram o desempenho das ações ao longo do tempo e revisar o histórico detalhado de todas as transações realizadas.

Inicialmente, o levantamento e análise dos requisitos são apresentados, seguido da arquitetura e descrição técnica do sistema e, por fim, uma apresentação da ferramenta desenvolvida.

4.1 LEVANTAMENTO E ANÁLISE DOS REQUISITOS

Os requisitos da ferramenta ADAS FINANCES foram identificados com base na análise dos pontos fracos e fortes das funcionalidades das ferramentas similares apresentadas no capítulo 2 e também com base na experiência do autor deste trabalho. Na sequência são apresentados os requisitos funcionais e não funcionais (Quadro 1 e Quadro 2).

Quadro 1 - Requisitos Funcionais

(continua)

#	Título do requisito	Descrição
RF01	Cadastro de usuários	O sistema deve permitir o cadastro de usuários, contemplando os dados nome de usuário, nome completo, senha e email.
RF02	Acessar o sistema	Os usuários devem ser capazes de acessar o sistema com suas credenciais (nome de usuário ou e-mail) e uma senha.
RF03	Sair do sistema	O sistema deve permitir que os usuários encerrem sua sessão até que um novo login seja realizado.
RF04	Configurar valor percentual do objetivo	O sistema deve permitir que os investidores configurem o valor percentual desejado para a ocupação de cada ação na sua carteira de investimentos.
RF05	Cadastrar ativos	Os usuários devem ser capazes de cadastrar seus ativos, informando o código ou nome do ativo, preço de compra, quantidade e peso.
RF06	Editar ativos	Os usuários devem ser capazes de editar seus ativos caso deseje; os campos preço de compra, quantidade e peso devem ser editáveis.
RF07	Deletar ativos	Os usuários devem ser capazes de excluir seus ativos.
RF08	Visualizar ativos	O sistema deve disponibilizar de forma visual as informações atualizadas dos ativos.

Quadro 1 - Requisitos Funcionais

(conclusão)

#	Título do requisito	Descrição
RF09	Adicionar transação	O sistema deve permitir que os investidores adicionem transações à sua carteira de investimentos.
RF10	Visualizar gráfico de ações	O sistema deve permitir que os investidores visualizem o desempenho dos investimentos ao longo do tempo de forma gráfica.
RF11	Visualizar histórico de transações	O sistema deve permitir que os investidores visualizem o histórico completo de transações realizadas na carteira de investimentos, que inclui compra e venda.
RF12	Recuperar informações atualizadas dos ativos	O sistema deve invocar um serviço externo para recuperar informações referentes aos ativos do usuário, como o preço atual.

Fonte: Autor

Quadro 2 - Requisitos Não-Funcionais

(continua)

#	Título do requisito	Descrição
RNF01	Sistema Web	O sistema deve ser acessado por meio de um navegador web (no mínimo Chrome versão 5).
RNF02	Integração	A aplicação deve ser capaz de se integrar a API do serviço AlphaVantage contendo informações atualizadas sobre os ativos do usuário.

Quadro 2 - Requisitos Não-Funcionais

(conclusão)

#	Título do requisito	Descrição
RNF03	Usabilidade	O sistema deve ter uma interface simplificada, intuitiva e amigável, facilitando o uso dos investidores, mesmo aqueles com pouca experiência em tecnologia.
RNF04	Segurança	A ferramenta deve assegurar a proteção dos dados sob sua responsabilidade.

Fonte: Autor

Conforme pode ser visto na Tabela 1, a principal funcionalidade do sistema é, assim como nas 3 plataformas apresentadas no capítulo 2, permitir a utilização de uma tabela descritiva com os ativos do usuário, apresentando informações como ativo; preço médio; preço atual; diferença; quantidade; patrimônio atual; variação total; % em ações; % na carteira; objetivo e ação.

As principais ações consistem na notificação de compra do ativo, quando este estiver abaixo do percentual objetivo deferido pelo usuário. Além de informar a quantidade de ações, ou em valor real do quanto precisa comprar para a carteira rebalancear para o estado objetivo.

Por exemplo, se define de antemão que, por ter o perfil de investidor mais agressivo, desejo que 70% da minha carteira seja alocada em ações, 15% em renda fixa e 15% em criptomoedas. Dentro de cada classe de ativos, se define ainda que dentro dos 70% de ações, 20% será alocado em VALE. Assim, conforme os preços das ações do portfólio vão subindo ou descendo, a porcentagem referente a eles também tende naturalmente a sair do meu objetivo inicial. Assim que a VALE oscila para baixo por exemplo, e as demais ações se mantêm, ou sobem de preço, é natural que a % de VALE na minha carteira desça dos 20%, o sistema então informa o usuário a quantidade de ações da VALE que eu necessito comprar para voltar a ter 20% dela na carteira. Isso vale para todas as ações. Assim a carteira sempre fica configurada para meu objetivo e perfil de investimento.

Além das informações dos ativos ilustrados de forma gráfica, são definidos indicativos de ações a serem tomadas, além de informações completas de acordo com o perfil traçado inicialmente pelo usuário, de forma a sempre rebalancear a carteira mirando os objetivos do usuário.

4.1.1 Histórias de Usuários

Histórias de Usuários descrevem uma funcionalidade importante para um usuário ou comprador de um sistema ou software e são compostas por três aspectos: uma descrição escrita da história usada para planejamento e como um lembrete; conversas sobre a história que servem para detalhar os pormenores da história e testes que transmitem e documentam detalhes e que podem ser usados para determinar quando uma história está completa (Cohn, 2009). Nesse sentido, neste trabalho as histórias de usuário são definidas por meio dos seguintes itens:

- a) US00 - identificador único para a história de usuário específica, mudando os algarismos de acordo com a inserção de novas histórias de usuário;
- b) Critérios de aceite - conjunto de condições que um produto, recurso ou funcionalidade deve cumprir para ser considerado completo e aprovado pelo cliente ou pela equipe de desenvolvimento;
- c) Requisitos - condições que a funcionalidade descrita na história deve atender para ser considerada completa e aceita pelo cliente ou pela equipe de desenvolvimento, e;
- d) Protótipo de tela - representação visual preliminar da interface do usuário que ilustra como a funcionalidade descrita na história será apresentada e interagida pelo usuário.

Como forma de realizar a análise dos requisitos, são definidas 11 histórias de usuário para o sistema ADAS FINANCE. Dentre as histórias de usuário documentadas, são apresentadas nesta seção três exemplos nos Quadros 3 a 5. As demais histórias de usuário, devidamente documentadas, são apresentadas no Apêndice A.

Quadro 3 - História de Usuário 01

US01 - Como Investidor, eu preciso realizar o auto-cadastro para que eu possa realizar login no sistema.

Critérios de aceite:

US01.01 - Os dados obrigatórios para o cadastro de um usuário são: nome, e-mail, usuário e senha.

US01.02 - Não é permitido o cadastro de dois usuários com o mesmo e-mail.

US01.03 - O usuário é registrado no banco de dados.

Requisitos:

RF01

Protótipos de tela:

O protótipo de tela mostra uma interface de usuário para o sistema Adas Finance. No topo, há uma barra de endereço com o URL <https://www.adasfinance.com/cadastro>. O formulário de cadastro é dividido em duas colunas. A coluna da esquerda contém um espaço reservado para uma imagem, representado por um retângulo com uma 'X' diagonal. A coluna da direita contém os campos de entrada para: Nome (Arthur da Silva), Email (arthurdasilva@gmail.com), Usuário (arthurdasilva) e Senha (representada por pontos). Abaixo dos campos, há um botão azul com o texto 'CADASTRAR'. Na base do formulário, há o texto 'Já possui uma conta?' seguido por um link azul 'Cadastre-se'.

Fonte: Autor

Quadro 4 - História de Usuário 02

(continua)

US02 - Como Investidor, eu devo conseguir fazer login.

Quadro 4 - História de Usuário 02

(conclusão)

Critérios de aceite:

US02.01 - O sistema deve apresentar uma tela de login acessível a partir da página inicial que deve incluir campos para inserir o nome de usuário e senha.

US02.02 - Se o e-mail ou a senha estiverem incorretos, o sistema deve exibir uma mensagem de erro informativa ao investidor.

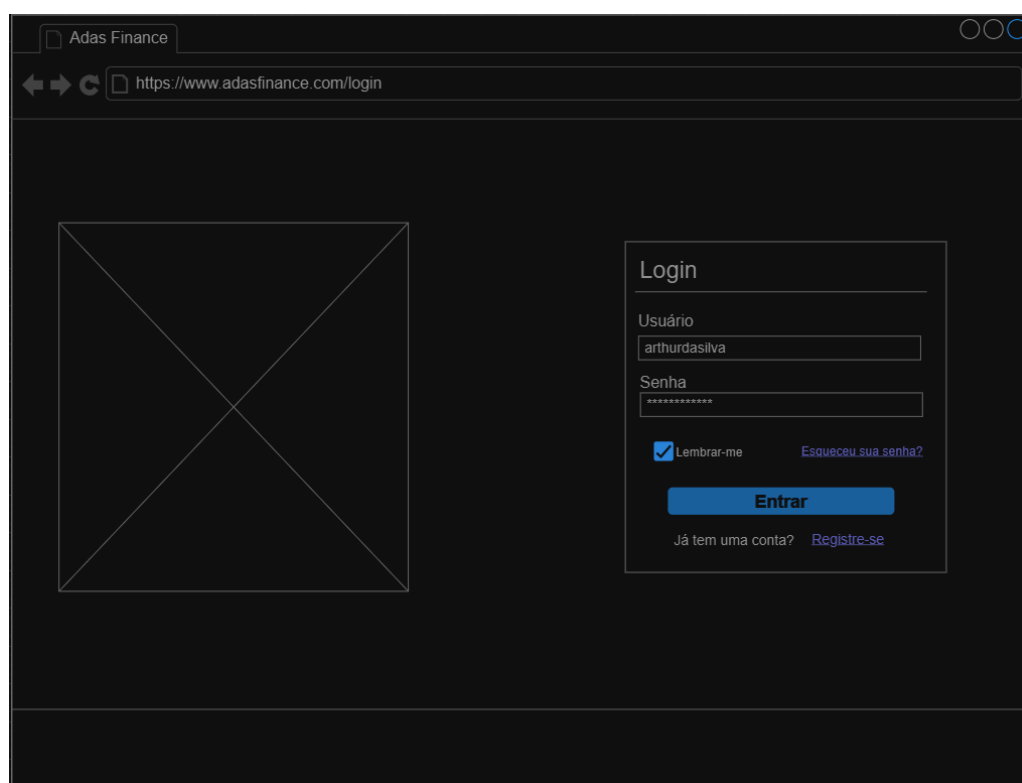
US02.03 - Após um login bem-sucedido, o investidor deve ser redirecionado para a página inicial do painel de investimentos.

US02.04 - O sistema deve oferecer uma opção de "Lembrar Me" para que o investidor possa optar por manter sua sessão aberta.

US02.05 - Deve haver uma funcionalidade de "Esqueci Minha Senha" que permita ao investidor recuperar sua senha através de um processo seguro.

Requisitos:

RF02

Protótipos de tela:

Fonte: Autor

Quadro 5 - História de Usuário 03

(continua)

US03 - Como Investidor, eu devo conseguir fazer logout.

Critérios de aceite:

US03.01 - Deverá haver um botão de logout claramente visível e acessível na página do sistema após o login.

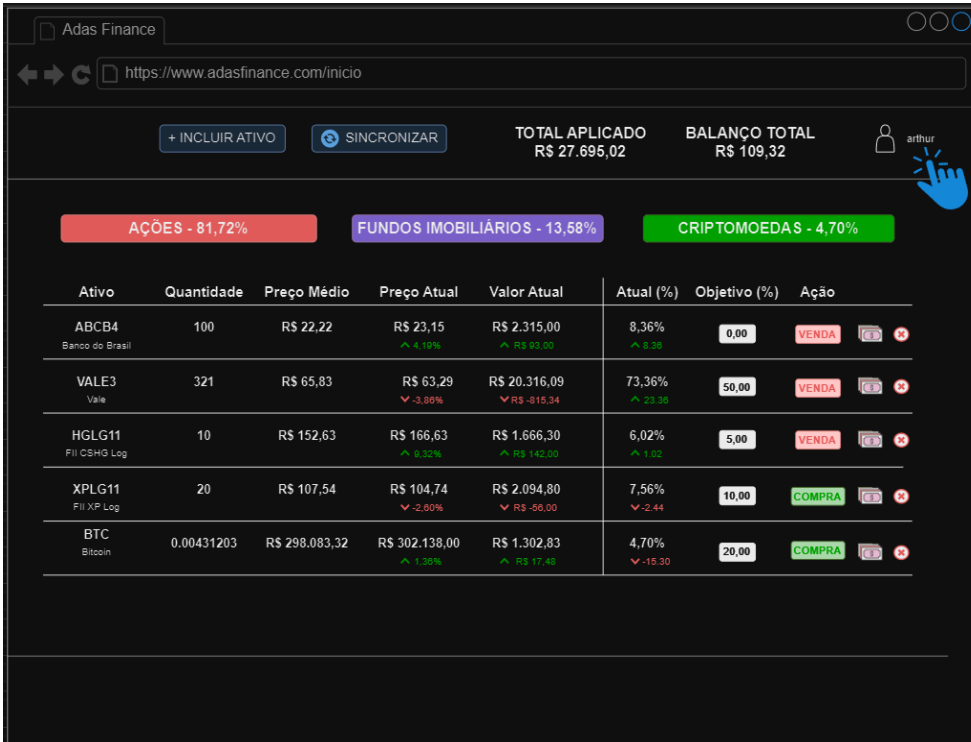
US03.02 - Ao clicar no botão de logout, o investidor deve ser imediatamente desconectado do sistema.

US03.03 - Após realizar o logout, o investidor deve ser redirecionado para a página de login do site.

Requisitos:

RF03

Protótipos de tela:



The screenshot shows a dark-themed financial dashboard. At the top, there are navigation buttons for '+ INCLUIR ATIVO' and 'SINCRONIZAR'. The main header displays 'TOTAL APLICADO R\$ 27.695,02' and 'BALANÇO TOTAL R\$ 109,32'. Below this, there are three colored bars representing asset categories: 'AÇÕES - 81,72%' (red), 'FUNDOS IMOBILIÁRIOS - 13,58%' (purple), and 'CRIPTOMOEDAS - 4,70%' (green). The main content is a table of assets with columns for 'Ativo', 'Quantidade', 'Preço Médio', 'Preço Atual', 'Valor Atual', 'Atual (%)', 'Objetivo (%)', and 'Ação'. A blue hand icon points to the user profile 'arthur' in the top right corner.

Ativo	Quantidade	Preço Médio	Preço Atual	Valor Atual	Atual (%)	Objetivo (%)	Ação
ABCB4 Banco do Brasil	100	R\$ 22,22	R\$ 23,15 ▲ 4,19%	R\$ 2.315,00 ▲ R\$ 93,00	8,36% ▲ 3,35%	0,00	VENDA
VALE3 Vale	321	R\$ 65,83	R\$ 63,29 ▼ -3,88%	R\$ 20.316,09 ▼ R\$ -315,34	73,36% ▲ -22,36%	50,00	VENDA
HGLG11 FII CSHG Log	10	R\$ 152,63	R\$ 166,63 ▲ 9,22%	R\$ 1.666,30 ▲ R\$ 142,00	6,02% ▲ 1,02%	5,00	VENDA
XPLG11 FII XP Log	20	R\$ 107,54	R\$ 104,74 ▼ -2,60%	R\$ 2.094,80 ▼ R\$ -56,00	7,56% ▼ -2,44%	10,00	COMPRA
BTC Bitcoin	0.00431203	R\$ 298.083,32	R\$ 302.138,00 ▲ 1,36%	R\$ 1.302,83 ▲ R\$ 17,48	4,70% ▼ -15,30%	20,00	COMPRA

Quadro 5 - História de Usuário 03

(conclusão)

Protótipos de tela:

Ativo	Quantidade	Preço Médio	Preço Atual	Valor Atual	Atual (%)	Objetivo (%)	Ação
ABCB4 Banco do Brasil	100	R\$ 22,22	R\$ 23,15 ▲ 4,19%	R\$ 2.315,00 ▲ R\$ 93,00	8,36% ▲ 2,95%	0,00	VENDA
VALE3 Vale	321	R\$ 65,83	R\$ 63,29 ▼ -3,88%	R\$ 20.316,09 ▼ R\$ -815,34	73,36% ▲ 23,35%	50,00	VENDA
HGLG11 FII CSHG Log	10	R\$ 152,63	R\$ 166,63 ▲ 9,22%	R\$ 1.666,30 ▲ R\$ 142,00	6,02% ▲ 1,02%	5,00	VENDA
XPLG11 FII XP Log	20	R\$ 107,54	R\$ 104,74 ▼ -2,60%	R\$ 2.094,80 ▼ R\$ -56,00	7,56% ▼ -2,44%	10,00	COMPRA
BTC Bitcoin	0.00431203	R\$ 298.083,32	R\$ 302.138,00 ▲ 1,36%	R\$ 1.302,83 ▲ R\$ 17,48	4,70% ▼ -16,30%	20,00	COMPRA

Fonte: Autor

As demais histórias de usuário são detalhadamente apresentadas no Apêndice A.

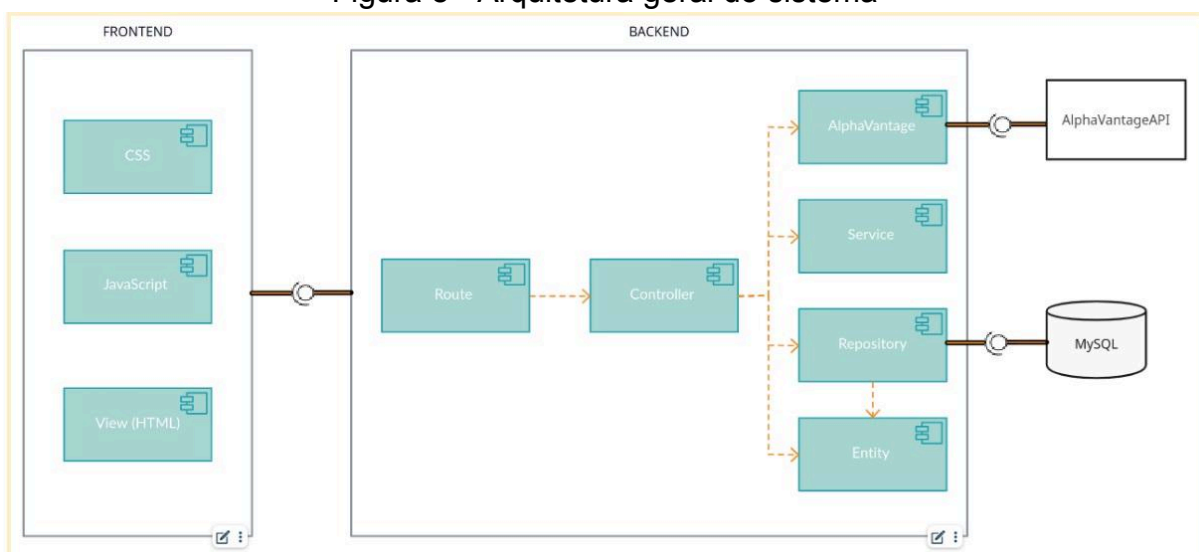
4.2 DEFINIÇÃO DA ARQUITETURA

Neste capítulo, é apresentada a arquitetura dos módulos que compõem o sistema desenvolvido, proporcionando uma visão detalhada sobre sua estrutura e funcionalidades. A compreensão da arquitetura é fundamental para identificar a interação entre os diferentes componentes e como eles se integram para formar uma solução coesa. O sistema é dividido em três partes principais: o backend (contemplando a integração com a API externa AlphaVantage), o banco de dados MySQL e o frontend. Cada um desses módulos desempenha um papel crucial no funcionamento do sistema e é descrito em detalhe ao longo deste capítulo, principalmente no tópico 3.3 (Implementação).

O módulo backend é a espinha dorsal do sistema, responsável por gerenciar a lógica de negócios, a comunicação com o banco de dados e a API externa. Ele é composto por seis módulos: Route, Controller, AlphaVantage, Service, Repository e Entity. O módulo Route define as rotas de acesso aos serviços do sistema, enquanto o Controller atua como intermediário entre as requisições do usuário e os serviços fornecidos. O módulo AlphaVantage é responsável pela integração com a API externa AlphaVantage, que fornece dados financeiros essenciais para o sistema. Já o Service implementa lógicas de negócio que dão suporte ao resto do sistema. O módulo Entity representa as entidades, que por fim são gerenciadas pelo módulo Repository, que interage com o banco de dados MySQL.

No frontend, a interação com o usuário é facilitada por tecnologias conhecidas e amplamente utilizadas no cenário de desenvolvimento web. A interface é representada pelos arquivos HTML disponíveis no módulo View, além de ser desenvolvida utilizando JavaScript, CSS, enquanto o Chart.js é utilizado para a criação de gráficos dinâmicos e interativos. Frameworks como Bootstrap e MDB (Material Design for Bootstrap) são empregados para garantir um design responsivo e uma experiência de usuário consistente e intuitiva. Esta combinação de tecnologias permite a criação de uma interface rica e funcional, que comunica eficazmente os dados e as funcionalidades fornecidas pelo backend. A seguir, cada um desses componentes será explorado em detalhes, elucidando suas características e o papel que desempenham na arquitetura geral do sistema.

Figura 8 - Arquitetura geral do sistema



Fonte: Autor

4.3 MODELAGEM DE BANCO DE DADOS

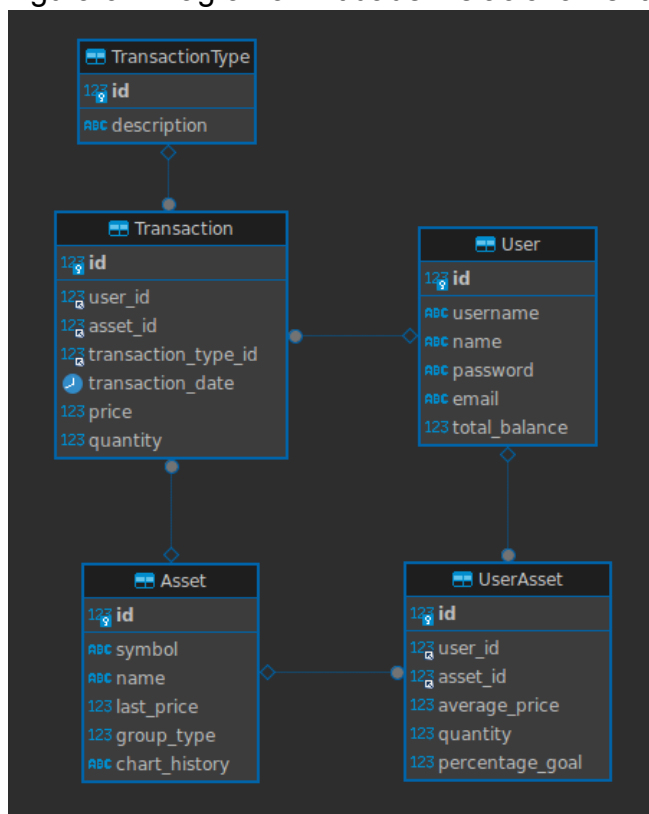
Neste desenvolvimento, foi utilizado o banco de dados MySQL, um dos sistemas de gerenciamento de banco de dados relacional mais utilizados ao redor do mundo, mantido pela Oracle Corporation. Ele permite armazenar, organizar e recuperar dados de maneira eficiente, além de ser conhecido por sua confiabilidade, desempenho e facilidade de uso, sendo uma escolha popular para aplicativos da web.

No projeto ADASFINANCE foram mapeadas tabelas, são elas:

- a) *TransactionType*, uma tabela informativa criada para fazer comparações utilizando os ids, e não strings de forma textual o tipo da transação, seja de compra ou de venda;
- b) *Transaction*, que representa as transações que cada usuário exerce de cada ativo relacionado a ele;
- c) *User*, que representa os usuários do sistema;
- d) *Asset*, que representa os ativos (ações, fundos imobiliários e criptomoedas) e
- e) *UserAsset*, que representa uma tabela associativa entre o usuário e o ativo, contendo informações sobre qual o preço médio, a quantidade e a porcentagem objetivo de determinado ativo para determinado usuário. As tabelas podem ser observadas com mais detalhes na figura abaixo.

A Figura 9 apresenta o diagrama do banco de dados da ferramenta desenvolvida. Os scripts de criação das tabelas são apresentados nas Figura 10 a 14.

Figura 9 - Diagrama Entidade-Relacionamento



Fonte: Autor

Figura 10 - Exemplo de documento da criação da tabela de ativos

```

CREATE TABLE `Asset` (
  `id` int NOT NULL AUTO_INCREMENT,
  `symbol` varchar(255) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `last_price` double NOT NULL,
  `group_type` int NOT NULL,
  `chart_history` text,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1462 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
    
```

Fonte: Autor

Figura 11 - Exemplo de documento da criação da tabela de usuários

```

CREATE TABLE `User` (
  `id` int NOT NULL AUTO_INCREMENT,
  `username` varchar(255) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `password` varchar(255) NOT NULL,
  `email` varchar(255) DEFAULT NULL,
  `total_balance` double DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
    
```

Fonte: Autor

Figura 12 - Exemplo de documento da criação da tabela de associativa entre usuários e ativos

```

CREATE TABLE `UserAsset` (
  `id` int NOT NULL AUTO_INCREMENT,
  `user_id` int DEFAULT NULL,
  `asset_id` int DEFAULT NULL,
  `average_price` double NOT NULL,
  `quantity` double NOT NULL,
  `percentage_goal` decimal(10,2) DEFAULT '0.00',
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  KEY `asset_id` (`asset_id`),
  CONSTRAINT `UserAsset_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `User` (`id`),
  CONSTRAINT `UserAsset_ibfk_2` FOREIGN KEY (`asset_id`) REFERENCES `Asset` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=86 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

Fonte: Autor

Figura 13: Exemplo de documento da criação da tabela de transações

```

CREATE TABLE `Transaction` (
  `id` int NOT NULL AUTO_INCREMENT,
  `user_id` int DEFAULT NULL,
  `asset_id` int DEFAULT NULL,
  `transaction_type_id` int DEFAULT NULL,
  `transaction_date` timestamp NOT NULL,
  `price` float NOT NULL,
  `quantity` double NOT NULL,
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  KEY `asset_id` (`asset_id`),
  KEY `transaction_type_id` (`transaction_type_id`),
  CONSTRAINT `Transaction_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `User` (`id`),
  CONSTRAINT `Transaction_ibfk_2` FOREIGN KEY (`asset_id`) REFERENCES `Asset` (`id`),
  CONSTRAINT `Transaction_ibfk_3` FOREIGN KEY (`transaction_type_id`) REFERENCES `TransactionType` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=79 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

Fonte: Autor

Figura 14: Exemplo de documento da criação da tabela de descrição textual das transações

```

CREATE TABLE `TransactionType` (
  `id` int NOT NULL AUTO_INCREMENT,
  `description` varchar(50) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

Fonte: Autor

4.4 IMPLEMENTAÇÃO

O desenvolvimento deste trabalho é dividido em algumas etapas, sendo as principais: modelagem dos dados e construção da estrutura modelo entidade relacionamento; desenvolvimento da camada de aplicação e integração com API externa e desenvolvimento da camada de apresentação (interface com o usuário).

4.4.1 Camada de aplicação (backend)

Neste capítulo, serão apresentadas informações referentes à implementação da camada de aplicação da ferramenta. A linguagem escolhida para o desenvolvimento do backend foi PHP, primeiramente pela familiaridade e experiência prévia do autor com a tecnologia, e em segundo lugar pelos benefícios próprios da linguagem que são adequados ao ambiente necessário para o desenvolvimento da solução, como por exemplo simplicidade, facilidade de uso, alta disponibilidade de documentação e suporte da comunidade, desempenho, segurança, manutenção e escalabilidade e entre outros. Vale destacar que não foi utilizado nenhum framework, de forma que todos os recursos necessários para o correto funcionamento da aplicação foram desenvolvidos do zero.

4.4.1.1 Módulo de roteamento

Como não foi utilizado nenhum framework no desenvolvimento, foi necessária a criação de um módulo específico para tratar do roteamento das requisições externas (chamadas pelo frontend). Esse módulo é composto basicamente pela classe **Router.php**, responsável pelo cadastramento e tratamento das rotas que são aceitas pela aplicação.

A aplicação inicia através da classe **index.php**, que basicamente carrega o autoloader do Composer, responsável por fazer a carga automática das classes PHP do projeto sem a necessidade de incluir manualmente arquivos com “require” ou “include”; chama a função nativa do **session_start**, usada para iniciar uma nova sessão ou retomar uma sessão existente, e por fim chama a função estática da classe **Router execute**, conforme a figura abaixo.

Figura 15 - Classe index.php

```
1 <?php
2
3 use AdasFinance\Route\Router;
4
5 require_once '../vendor/autoload.php';
6
7 session_start();
8 Router::execute();
9
10 ?>
```

Fonte: Autor

A função **execute**, por sua vez, recupera as rotas cadastradas (através da função **routes** que será especificada a seguir: recupera o método do protocolo HTTP e a path da requisição, utilizando as classes nativas da linguagem “Request” e “Uri”; inclui o cabeçalho da página (caso seja necessário); verifica se a rota existe; faz validações referentes à sessão do usuário, como por exemplo se o usuário já está logado e tenta ir para a rota de “login”, se o usuário não está logado e tenta ir para a “home” etc. Após a validação é feito o respectivo redirecionamento, como por exemplo redirecionar o usuário para a tela “home”, ou redirecionar o usuário para a tela de “login”, ações respectivas aos exemplos de verificações citadas anteriormente; inicializa um elemento chamado “\$router” e chama uma função com o mesmo nome da variável (explicado no próximo parágrafo), e inclui o footer da página (caso seja necessário).

Figura 16 - Função execute da classe **Router**

```

80 public static function execute()
81 {
82     try {
83         $routes = self::routes();
84         $requestMethod = Request::get();
85         $uri = Uri::get('path');
86
87         self::includeHeader($uri);
88
89         if (!isset($routes[$requestMethod]) || !array_key_exists($uri, $routes[$requestMethod])) {
90             throw new Exception('Route not found');
91         }
92
93         self::handleAuthenticationRedirects($requestMethod, $uri);
94         $router = $routes[$requestMethod][$uri];
95
96         if (!is_callable($router)) {
97             throw new Exception("Route {$uri} is not callable");
98         }
99
100        $router();
101
102        self::includeFooter($uri);
103    } catch (\Throwable $th) {
104        echo $th->getMessage();
105    }
106 }
107

```

Fonte: Autor

A função **routes**, chamada pela função anterior, apenas retorna um array de funções chaveadas por algum dos quatro métodos HTTP utilizados (GET, POST, PUT e DELETE) e o path da requisição, que especificam qual a classe Controller e seu respectivo método encarregado pelo tratamento da requisição. Cada um dos registros chama a função **load**, especificada a seguir.

Figura 17 - Função routes da classe Router

```

46 public static function routes(): array
47 {
48     return [
49         'get' => [
50             '/login' => fn () => self::load('UserController', 'login'),
51             '/signup' => fn () => self::load('UserController', 'signup'),
52             '/logout' => fn () => self::load('UserController', 'logout'),
53             '/home' => fn () => self::load('UserAssetController', 'home'),
54             '/search_by_symbol' => fn () => self::load('AssetController', 'searchBySymbol'),
55             '/chart_history' => fn () => self::load('AssetController', 'chartHistory'),
56         ],
57         'post' => [
58             '/login_submit' => fn () => self::load('UserController', 'loginSubmit'),
59             '/signup_submit' => fn () => self::load('UserController', 'signupSubmit'),
60             '/user_asset_goal_percentage' => fn () => self::load('UserAssetController', 'userAssetGoalPercentage'),
61             '/user_asset_save' => fn () => self::load('UserAssetController', 'userAssetSave'),
62             '/register_transaction' => fn () => self::load('UserAssetController', 'registerTransaction'),
63             '/synchronize_user_assets' => fn () => self::load('UserController', 'synchronizeUserAssets'),
64         ],
65         'put' => [
66         ],
67         'delete' => [
68             '/user_asset_delete' => fn () => self::load('UserAssetController', 'userAssetDelete'),
69         ],
70     ];
71 }

```

Fonte: Autor

Por fim, a função **load** tem como objetivo carregar dinamicamente um controlador e executar uma ação específica com base nos parâmetros fornecidos pela função anterior, conforme a figura abaixo.

Figura 18 - Função load da classe Router

```

23 public static function load(string $controller, string $action)
24 {
25     try {
26         $controllerNamespace = self::CONTROLLER_NAMESPACE . '\\' . $controller;
27         $action .= "Action";
28
29         if (!class_exists($controllerNamespace)) {
30             throw new Exception("Controller {$controller} not found");
31         }
32
33         $controllerInstance = new $controllerNamespace;
34
35         if (!method_exists($controllerInstance, $action)) {
36             throw new Exception("Action {$action} not found in Controller {$controller}");
37         }
38
39         $data = json_decode(file_get_contents("php://input"), true) ?? $_REQUEST;
40
41         $controllerInstance->$action((object) $data);
42     } catch (\Throwable $sth) {
43         echo $sth->getMessage();
44     }
45 }

```

Fonte: Autor

4.4.1.2 Módulo de entidades

O módulo de entidades representa e manipula os dados da aplicação. Ele inclui classes de entidades que correspondem às tabelas do banco de dados,

encapsulando algumas lógicas de negócios e regras de validação. O módulo também auxilia a persistência de dados, oferecendo métodos “getters” e “setters” para gerenciamento das entidades a serem persistidas/recuperadas. Além disso, o módulo é independente da interface do usuário e das classes controladoras, fornecendo uma forma simples de acesso e manipulação de dados. As classes de entidade deste desenvolvimento são: Asset, Transaction, User e UserAsset, que contêm métodos específicos para as operações e lógica de negócio associadas.

Figura 19 - Classe Asset

```
5
6 class Asset {
7
8     private $id;
9     private $symbol;
10    private $name;
11    private $lastPrice;
12    private $groupType;
13    private $chartHistory;
14
15    public function __construct(
16        ?int $id = null,
17        ?string $symbol = null,
18        ?string $name = null,
19        ?float $lastPrice = null,
20        ?int $groupType = null,
21        ?string $chartHistory
22    ) {
23        $this->id = $id;
24        $this->symbol = $symbol;
25        $this->name = $name;
26        $this->lastPrice = $lastPrice;
27        $this->groupType = $groupType;
28        $this->chartHistory = $chartHistory;
29    }
30
31    public static function createFromParams(stdClass $params): self
32    {
33        return new self(
34            self::convertToType($params->id, 'int'),
35            self::convertToType($params->symbol, 'string'),
36            self::convertToType($params->name, 'string'),
37            self::convertToType($params->lastPrice, 'double'),
38            self::convertToType($params->groupType, 'int'),
39            self::convertToType($params->chartHistory, 'string'),
40        );
41    }
42
43    private static function convertToType($value, string $type)
44    {
45        if (!is_null($value)) {
46            settype($value, $type);
47        }
48
49        return $value;
50    }
51 }
```

Fonte: Autor

Exemplo que representa a entidade “**Asset**”. Além dessas funções, ainda fornece “getters” e “setters” de todos os atributos, com suas respectivas validações e regras caso se faça necessário.

4.4.1.3 Módulo de consulta e persistência

O módulo de consulta e persistência, denominado no projeto como "Repository", funciona como uma camada intermediária entre o modelo (entidades) e o banco de dados, centralizando e encapsulando as operações de persistência e consulta de dados. Ele realiza operações CRUD (Create, Read, Update, Delete) e gerencia a conexão com o banco de dados (através da classe nativa do PHP chamada "PDO"), fornecendo uma interface limpa e desacoplada para acesso aos dados. Isso facilita a manutenção, melhora a legibilidade e permite mudanças na lógica de acesso a dados sem impactar diretamente os módulos de entidades ou controladores.

Em PHP, é comum que um repositório utilize um ORM (*Object-Relational Mapping*): uma técnica de programação que permite mapear objetos de uma linguagem de programação para tabelas de um banco de dados relacional. Os ORMs mais conhecidos da linguagem são o Eloquent (do framework Laravel) e o Doctrine (do Symfony). Como o autor optou pela não utilização de frameworks no projeto, foram criadas consultas SQL diretas para interagir com o banco de dados.

Figura 20 - Classe ConnectionCreator

```
1 <?php
2 namespace AdasFinance\Service;
3
4 require_once '/home/arthur/Projects/AdasFinance/public/config.php';
5 use PDO;
6
7 class ConnectionCreator
8 {
9     public static $connection;
10
11     public static function getConnection()
12     {
13         if (!isset(self::$connection)) {
14             self::$connection = new PDO('mysql:host=' . DB_HOST . ';dbname=' . DB_NAME, DB_USER, DB_PASS);
15             self::$connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
16         }
17
18         return self::$connection;
19     }
20 }
21
22 ?>
```

Fonte: Autor

Representação da classe responsável por encapsular a conexão com o banco de dados (MySQL).

As classes do módulo de consulta e persistência implementam uma interface, chamada **RepositoryInterface**, que define a implementação do método **query** identificado na imagem abaixo:

Figura 21 - Função query da classe **RepositoryTrait**

```
16 private PDO $pdo;  
17  
18 public function query($sql, $params = [])  
19 {  
20     try {  
21         $this->pdo = ConnectionCreator::getConnection();  
22         $stmt = $this->pdo->prepare($sql);  
23         $stmt->execute($params);  
24  
25         $results = $stmt->fetchAll(PDO::FETCH_CLASS);  
26         return [  
27             'status' => 'success',  
28             'data' => $results  
29         ];  
30     } catch (PDOException $err) {  
31         return [  
32             'status' => 'error',  
33             'data' => $err->getMessage()  
34         ];  
35     }  
36 }  
37  
38
```

Fonte: Autor

O trecho de código acima define o método que executa as consultas. Primeiramente, ele obtém uma conexão com o banco de dados através do método estático **getConnection**. Em seguida, prepara a consulta SQL fornecida (\$sql) usando o método **prepare** e executa a consulta com os parâmetros fornecidos (\$params). Se a execução for bem-sucedida, ele busca todos os resultados como instâncias de classe e retorna um array com o status 'success' e os dados resultantes. Caso ocorra uma exceção (PDOException), ele captura o erro e retorna um array com o status 'error' e a mensagem de erro correspondente.

Figura 22 - Exemplo de insert básico da classe **Transaction** do módulo **Repository**

```

11 public function insert(Transaction $transaction)
12 {
13     $sql = "INSERT
14     INTO
15     Transaction(
16     user_id,
17     asset_id,
18     transaction_type_id,
19     transaction_date,
20     price,
21     quantity
22     )
23     VALUES(
24     :userId,
25     :assetId,
26     :transactionTypeId,
27     :transactionDate,
28     :price,
29     :quantity
30     )";
31
32     $params = [
33     ':userId' => $transaction->getUserId(),
34     ':assetId' => $transaction->getAssetId(),
35     ':transactionTypeId' => $transaction->getTransactionTypeId(),
36     ':transactionDate' => $transaction->getTransactionDate(),
37     ':price' => $transaction->getPrice(),
38     ':quantity' => $transaction->getQuantity(),
39     ];
40
41     return $this->query($sql, $params);
42 }

```

Fonte: Autor

4.4.1.4 Módulo de controle

O módulo de controlador (Controller) atua como intermediário entre a interface do usuário (camada de apresentação) e o resto da aplicação. Ele gerencia a lógica de fluxo da aplicação ao processar requisições recebidas, decidir o que deve ser feito e delegar tarefas aos módulos responsáveis e por fim à visualização.

O controlador recebe os dados da requisição, como parâmetros de URL ou dados de formulários, valida-os e aciona os módulos para o processamento dos dados conforme necessário. Após a manipulação dos dados, o controlador seleciona a visualização apropriada para apresentar os resultados ao usuário. Alguns dos controladores implementados aplicam regras de negócio e coordenam as transações entre o modelo e a visualização, garantindo que a aplicação funcione de maneira coesa e eficiente.

No modelo proposto neste trabalho, cada rota invoca seu respectivo método denominado pelo nome da rota em camelcase acrescido do sufixo "**Action**". Essa configuração é definida na classe "**Router**", mencionada na seção 3.4.1.1. desse documento.

Figura 23 - Exemplo do controlador da classe **AssetController**, responsável por tratar tarefas relacionadas aos ativos

```

19 public function __construct()
20 {
21     $this->assetRepository = new AssetRepository();
22     $this->userRepository = new UserRepository();
23 }
24
25 public function searchBySymbolAction($parameters)
26 {
27     $result = $this->assetRepository->searchByQuery($parameters->symbol, $_SESSION['user']->getId());
28
29     echo json_encode($result);
30 }
31
32 public function chartHistoryAction($parameters)
33 {
34     $chartHistory = AlphaVantageApiService::getChartHistory($parameters->symbol);
35
36     if ($chartHistory) {
37         $asset = new Asset(null, $parameters->symbol, null, null, null, $chartHistory);
38         $this->assetRepository->update($asset);
39     } else {
40         $chartHistory = $this->assetRepository->getChartHistoryBySymbol($parameters->symbol)['data'][0]->chart_history;
41     }
42
43     echo json_encode($chartHistory);
44 }
45

```

Fonte: Autor

No exemplo acima, a função **searchBySymbolAction** invoca o repositório um objeto da classe **AssetRepository** e chama a função **searchByQuery**, que com um determinado string retorna os ativos que possuem similaridade de nome ou código com o string fornecido. Essa função é chamada quando o usuário vai cadastrar um ativo através da função “Incluir Ativo” da interface.

Já a função **chartHistoryAction** invoca uma classe estática de um serviço do módulo de integração com API externa, responsável por fazer uma requisição REST para a API do AlphaVantage a fim de obter os dados históricos de determinado ativo através do seu símbolo. A função ainda faz uma validação que verifica se a requisição foi bem sucedida e conseguiu recuperar os dados históricos. Caso positivo, convoca o repositório **AssetRepository** e atualiza as informações obtidas através do método update. Caso negativo, recupera através do mesmo repositório as últimas informações registradas neste campo.

4.4.1.5 Módulo de serviços

O módulo de serviços, pode ser considerado um conjunto genérico de utilidades que fornecem suporte e funcionalidades auxiliares para outras partes do sistema. Ele contém classes com métodos estáticos que oferecem diversas funções úteis, como manipulação de strings, operações matemáticas, validações de dados, ou interações com APIs externas. Esses serviços são projetados para ser

reutilizáveis e independentes, promovendo a reutilização de código e reduzindo a duplicação. Além disso, centralizar funções comuns em um módulo de serviços facilita a manutenção e a atualização, pois quaisquer melhorias ou correções podem ser feitas em um único lugar, beneficiando todo o sistema.

Neste módulo estão contidas três classes, sendo elas:

- a) **AssetTransactionManager**: responsável por gerenciar as transações de ativos do usuário (como criação da transação, registro de compra e venda de ativos, cálculo de preço médio etc);
- b) **CamelCaseConverter**: responsável por converter um objeto com atributos em formato snake case para o formato camelcase;
- c) **ConnectionCreator**: responsável por obter conexão com banco de dados.

4.4.1.6 Módulo de integração com APIs externas

Este módulo encapsula toda a lógica necessária para interagir com APIs de terceiros, como envio de requisições HTTP, tratamento de respostas e gerenciamento de autenticação. Para isso, foi utilizada a função nativa do php “file_gets_content”, responsável por fazer requisições HTTP simples ao fornecer uma URL como argumento e retornando o conteúdo da resposta.

Ao centralizar as integrações, o módulo facilita a manutenção e a escalabilidade, permitindo que mudanças nas APIs externas sejam gerenciadas em um único ponto. Além disso, promove a reutilização de código, garantindo que diferentes partes do sistema possam aproveitar as funcionalidades fornecidas pelas APIs sem duplicação de esforço.

Dentro deste módulo ainda, foram usadas duas APIs externas, uma para popular a tabela Assets inicialmente com informações sobre os ativos, e a outra sendo utilizada em tempo de execução sempre que necessário, requisitando informações em tempo real sobre os ativos.

Vale destacar que ambas as APIs possuem limites de requisições diárias nos planos gratuitos, o que exige adaptações nas integrações para gerenciar eficientemente o uso das chamadas. Essas adaptações incluem a otimização do código para reduzir a necessidade de múltiplas requisições consecutivas e a

priorização das requisições mais importantes para garantir que os limites não sejam excedidos.

4.4.1.6.1 HG Finance

O HG Finance é uma API que fornece dados da bolsa de valores brasileira B3, preço de ações, dividendos, índices, cotação do dólar e várias outras moedas. API fácil de implementar, desde 2009 no mercado, simples de implementar, com bibliotecas em PHP, Ruby e JavaScript (em breve).

Foi utilizada com o objetivo de popular a tabela Assets, contendo todos os ativos do Ibovespa, sendo eles ações ou fundos imobiliários. Por ser uma API paga, foi utilizado apenas o período de testes de 7 dias da aplicação, que oferecia a opção de fazer chamadas individuais por símbolo de ativo, ou chamadas de até cinco símbolos por vez, de forma a reduzir o número de chamadas. A integração é resumida apenas em uma classe chamada **DatabasePopulator**, que resumidamente se limita às seguintes etapas:

1. Faz uma chamada para obter a lista de todas as siglas de todas as ações e todos os fundos imobiliários disponíveis:

Figura 24 - Função **processAssets** da classe DatabasePopulator

```
14
15 public function processAssets() {
16     // Faz a chamada para obter a lista de siglas
17     $ticker_list_url = 'https://api.hgbrasil.com/finance/ticker_list';
18     $ticker_list_response = $this->callAPI($ticker_list_url);
19
20     if (!$ticker_list_response || !isset($ticker_list_response['results'])) {
21         return false;
22     }
23
24     $results = $ticker_list_response['results'];
25
```

Fonte: Autor

2. Agrupa os ativos retornados em grupos de 5:

Figura 25 - Trecho da função **processAssets** da classe DatabasePopulator

```
25
26     // Agrupa as siglas em grupos de 5
27     $groups = array_chunk($results, 5);
28
```

Fonte: Autor

3. Para cada grupo, obtém os detalhes de cada um dos 5 ativos em uma única requisição:

Figura 26 - Trecho da função **processAssets** da classe DatabasePopulator

```

33 // Para cada grupo de siglas, faz a chamada para obter os detalhes
34 foreach ($groups as $group) {
35     $symbols = implode(',', $group);
36     $stock_price_url = "https://api.hqbrasil.com/finance/stock_price?key=" . $this->api_key . '&symbol=' . $symbols;
37     $stock_price_response = $this->callAPI($stock_price_url);
38
39     if (!$stock_price_response || !isset($stock_price_response['results'])) {
40         continue;
41     }
42
43     $asset_data = $stock_price_response['results'];
44

```

Fonte: Autor

4. Traduz os dados obtidos e extrai as informações relevantes para a carga na tabela Assets (preço, nome, tipo, último preço etc.):

Figura 27 - Trecho da função **processAssets** da classe DatabasePopulator

```

45 foreach ($asset_data as $symbol => $data) {
46     // Traduz o tipo de grupo
47     $group_type = isset($data['kind']) ? $this->translateGroupType($data['kind']) : 0;
48     $data['price'] = isset($data['price']) ? $data['price'] : 0;
49     $data['name'] = isset($data['name']) ? str_replace(["", " ", " ", " "], "", $data['name']) : "";
50

```

Fonte: Autor

5. Para cada ativo, escreve uma linha de INSERT com os dados referentes ao ativo em questão e armazena em um array chamado **\$insert_queries**:

Figura 28 - Trecho da função **processAssets** da classe DatabasePopulator

```

51 // Prepara o texto do insert
52 $insert_queries[] = "INSERT INTO Asset (symbol, name, last_price, group_type) VALUES ('{$symbol}', '{$data['name']}', {$data['price']}, {$group_type})";
53
54 $queries_executed++;
55

```

Fonte: Autor

6. Repete as etapas 3, 4 e 5 para todos os grupamentos definidos anteriormente;
7. Chama a função **executeInserts**, responsável por abrir uma conexão com o banco de dados e executar o texto de cada elemento contido no array **\$insert_queries** como uma query INSERT no banco:

Figura 29 - Função **executeInserts** da classe DatabasePopulator

```

86     private function executeInserts($insert_queries) {
87         $pdo = ConnectionCreator::getConnection();
88         $count = 1;
89         try {
90             foreach ($insert_queries as $query) {
91                 // Executa o insert no banco de dados
92                 $stmt = $pdo->prepare($query);
93                 $stmt->execute([]);
94                 $stmt->fetchAll(PDO::FETCH_CLASS);
95                 echo "$count - Sucesso no insert: $query \n";
96                 $count++;
97             }
98         }
99     }
100     } catch (PDOException $err) {
101         echo "Erro no insert: $query \n";
102     }
103     die;
104 }
105 }

```

Fonte: Autor

Todo esse processo é realizado via linha de comando, executando o PHP diretamente através do terminal. É atribuída a uma variável a chave utilizada para fazer a autenticação com a API do HG Finance; após isso é criada uma instância da classe e por fim acionado o método **processAssets**, que inicia todo o processo detalhado acima.

Figura 30 - Trecho de código da classe DatabasePopulator

```

107     $api_key = ██████████;
108     $processor = new DatabasePopulator($api_key);
109     $processor->processAssets();

```

Fonte: Autor

4.4.2 Camada de apresentação (frontend)

A camada de apresentação serve como uma interface visual através da qual os usuários interagem com a aplicação. Ela é responsável por exibir dados ao usuário e capturar suas entradas para serem processadas pelo sistema. Foi utilizado apenas JavaScript puro, CSS, HTML, Bootstrap 5, MDB (Material Design for Bootstrap), e Chart.js. Esta camada foca em fornecer uma experiência de usuário

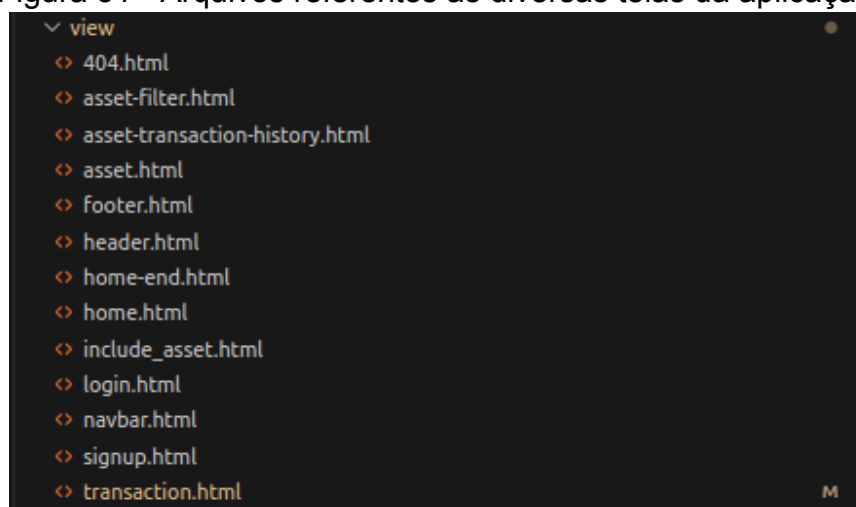
fluida e intuitiva, sem a complexidade adicional de frameworks de frontend mais avançados. De forma resumida: o HTML estrutura a página, o CSS estiliza os elementos, e o JavaScript adiciona interatividade e dinamismo às páginas.

Por ter pouca experiência com desenvolvimento frontend e ter optado por não utilizar frameworks JavaScript mais complexos como React, Angular e Vue, a camada de apresentação ficou dividida apenas em quatro “módulos”, sendo eles um módulo responsável por conter os arquivos HTML chamado de **View**, um módulo contendo apenas uma classe CSS, chamada **main.css**, um módulo contendo apenas uma classe JavaScript chamada **main.js** e um módulo chamado imagens que contém as imagens utilizadas pela aplicação.

4.4.2.1 HTML e CCS

As telas da aplicação ficaram armazenadas no módulo denominado View. São arquivos HTML que compõem as três telas principais: login, cadastro e home. Tentou-se isolar o máximo possível do código PHP das telas, porém por não utilizar nenhum framework não foi possível desacoplar totalmente o módulo do backend da aplicação. Alguns arquivos contêm "componentes" que são reutilizados em parte do código. Esses componentes incluem cabeçalhos, rodapés e menus de navegação, proporcionando uma consistência visual e funcional em toda a aplicação. A reutilização de componentes também facilita a manutenção e a atualização do sistema, pois as alterações podem ser feitas em um único lugar. Apesar das limitações, essa abordagem permitiu uma organização modular do projeto, separando, na medida do possível, a lógica de apresentação da lógica de negócios.

Figura 31 - Arquivos referentes às diversas telas da aplicação.



Fonte: Autor

Figura 32 - Exemplo de “componente” html especificado no arquivo asset-filter.html

```

1 <div id="asset-filter" class="mt-3">
2   <div class="nav nav-pills nav-justified mb-3 d-grid gap-3 d-md-flex col-8 mx-auto">
3     <button
4       type="button"
5       class="btn nav-link active col-4"
6       data-mdb-ripple-init
7       data-mdb-ripple-color="dark"
8       id="acoes"
9       data-group-id="1">
10      <i class="fa-solid fa-circle-dollar-to-slot fa-lg me-2"></i>
11      Ações -
12    </button>
13    <button
14      type="button"
15      class="btn nav-link active col-4"
16      data-mdb-ripple-init
17      data-mdb-ripple-color="dark"
18      id="fiis"
19      data-group-id="2">
20      <i class="fa-solid fa-house-chimney fa-lg me-2"></i>
21      Fundos Imobiliários -
22    </button>
23    <button
24      type="button"
25      class="btn nav-link active col-4"
26      data-mdb-ripple-init
27      data-mdb-ripple-color="dark"
28      id="criptos"
29      data-group-id="3">
30      <i class="fa-brands fa-btc fa-lg me-2"></i>
31      Criptomoedas -
32    </button>
33  </div>
34 </div>

```

Fonte: Autor

Já o CSS ficou todo contido na classe main.css, conforme mostra o trecho de código da figura abaixo.

Figura 33 - Exemplo de trechos de código que aplicam estilos em determinadas classes.

```
1  .main-background {
2    background: #55a8c9;
3  }
4
5  .text-input {
6    height: 4em !important;
7  }
8
9  .icon-container {
10   display: flex;
11 }
12
13 .icon-container div {
14   margin-right: 12px;
15 }
16
17 .icon-container i {
18   font-size: 22px;
19 }
20
21 .objective-percentage {
22   border: 1px solid #e5e5e5;
23   border-radius: 20px;
24   height: 32px;
25   width: 60px;
26   outline: none;
27   transition: border-color 0.3s ease;
28   font-weight: 500;
29   font-size: .875rem;
30   line-height: 1.25rem;
31   text-align: center;
32 }
```

Fonte: Autor

4.4.2.2 JavaScript

É representado pela classe main.js, e entre suas atribuições principais pode-se citar como exemplos:

- a) formatar tabela: a função **formatTable** percorre todas as linhas e colunas de uma tabela, adicionando estilos e ícones de acordo com os dados presentes;

Figura 34 - Trecho de código referente a implementação da função **formatTable** da classe **main.js**

```
8 function formatTable() {
9   let rows = document.querySelectorAll('tr');
10  rows.forEach(function(row) {
11    if (!row.classList.contains("asset-transaction-history")) {
12      addAssetIcon(row);
13    }
14  });
15
16  let columns = document.querySelectorAll('td');
17  columns.forEach(function(column) {
18    column.classList.add('text-center');
19  });
20
21  spans = document.querySelectorAll('.difference');
22  spans.forEach(function(span) {
23    let spanValue = convertToFloat(span.innerText);
24    let spanParentNode = span.parentNode;
25    let icon = spanParentNode.querySelector('i');
26
27    if (spanValue > 0) {
28      spanParentNode.classList.add('text-success');
29      icon.classList.add('fa-caret-up');
30    } else if (spanValue == 0) {
31      spanParentNode.classList.add('text-secondary');
32      icon.classList.add('fa-minus');
33    } else {
34      spanParentNode.classList.add('text-danger');
35      icon.classList.add('fa-caret-down');
36    }
37  });
38
39  let transactionsType = document.querySelectorAll('.transaction-type');
40  transactionsType.forEach(function(transactionType) {
41    const textContent = transactionType.childNodes[0].nodeValue.trim();
42    if (textContent == "COMPRA") {
43      transactionType.closest('tr').style.color = "green";
44      transactionType.childNodes[1].classList.add("fa-up-long");
45    } else {
46      transactionType.closest('tr').style.color = "red";
47      transactionType.childNodes[1].classList.add("fa-down-long");
48    }
49  });
50 }
```

Fonte: Autor

- b) calcular a diferença entre objetivo e percentual atual: a função **calculateObjectivePercentageDifference** calcula a diferença percentual entre valores e aplica estilos de acordo com o resultado, além de exibir alertas em caso de limites excedidos;

Figura 35 - Trecho de código referente a implementação da função **calculateObjectivePercentageDifference** da classe main.js

```

52 function calculateObjectivePercentageDifference() {
53   let inputs = document.querySelectorAll('.objective-percentage');
54   let objectivePercentageSum = 0;
55
56   inputs.forEach(function(input) {
57     let td = input.closest('td');
58     let previousTd = td.previousElementSibling;
59     let nextTd = td.nextElementSibling;
60
61     let spanCurrentPercentage = previousTd.querySelector('.primary-value span');
62     let spanDifference = previousTd.querySelector('.second-value span');
63     let innerSpan = spanDifference.querySelector('span');
64
65     let icon = previousTd.querySelector('.second-value i');
66     let badge = nextTd.querySelector('.badge');
67
68     let currentPercentage = convertToFloat(spanCurrentPercentage.textContent);
69     let objectivePercentage = convertToFloat(input.value);
70     let percentageDiff = (currentPercentage - objectivePercentage).toFixed(2);
71
72     objectivePercentageSum += parseFloat(objectivePercentage);
73     innerSpan.textContent = percentageDiff;
74
75     spanDifference.classList.remove('text-success', 'text-secondary', 'text-danger');
76     icon.classList.remove('fa-caret-up', 'fa-minus', 'fa-caret-down');
77     badge.classList.remove('badge-success', 'badge-secondary', 'badge-danger');
78
79     if (percentageDiff > 0) {
80       spanDifference.classList.add('text-success');
81       icon.classList.add('fa-caret-up');
82       badge.classList.add('badge-danger');
83       badge.innerText = "Venda";
84     } else if (percentageDiff == 0) {
85       spanDifference.classList.add('text-secondary');
86       icon.classList.add('fa-minus');
87       badge.classList.add('badge-secondary');
88       badge.innerText = "Neutro";
89     } else {
90       spanDifference.classList.add('text-danger');
91       icon.classList.add('fa-caret-down');
92       badge.classList.add('badge-success');
93       badge.innerText = "Compra";
94     }
95   });
96
97   if (objectivePercentageSum > 100) {
98     showAlertContainer();
99     return false;
100  }
101
102  closeAlertContainer();
103  return true;
104 }

```

Fonte: Autor

- c) adicionar “listeners” de eventos: a função **addEventListener** adiciona diversos ouvintes de eventos, como por exemplo para inputs de objetivo de porcentagem e autocomplete de ativos, garantindo funcionalidades como sincronização de dados e interação com o usuário;

Figura 36 - Trecho de código referente a implementação da função `searchBySymbol` da classe `main.js`, no exemplo estão sendo configurados os eventos “focus” e “blur” para os elementos que possuem a classe “objective-percentage”.

```

106 async function addEventListeners() {
107   let objectivePercentageInputs = document.querySelectorAll('.objective-percentage');
108
109   objectivePercentageInputs.forEach(function(input) {
110     let originalValue = input.value;
111
112     input.addEventListener('focus', function() {
113       originalValue = input.value;
114     });
115
116     input.addEventListener('blur', async function() {
117       if (input.value !== originalValue) {
118         const container = input.parentNode;
119         const spinner = container.querySelector('.spinner');
120
121         if (calculateObjectivePercentageDifference()) {
122           input.style.display = 'none';
123           spinner.style.display = 'block';
124
125           const userAssetId = input.getAttribute('data-user-asset-id');
126           const newValue = input.value;
127
128           setTimeout(async () => {
129             await saveNewObjectivePercentageValue(userAssetId, newValue);
130             input.style.display = '';
131             spinner.style.display = 'none';
132           }, 1000);
133         }
134       }
135     });
136   });

```

Fonte: Autor

- d) gerenciar máscaras de inputs: funções como **numberMask**, **removeNumberMask**, **doublePriceMask** e outras são responsáveis por formatar e remover máscaras de inputs, como as utilizadas nos campos referentes a preços e quantidades, garantindo a consistência dos dados inseridos;

Figura 37 - Trecho de código referente a implementação da função `searchBySymbol` da classe `main.js`

```

633 const numberMask = (input) => {
634   if (input.value.trim() === '') {
635     input.value = 0;
636   }
637
638   const inputId = input.id;
639
640   switch (inputId) {
641     case 'average-price':
642     case 'average-price-transaction':
643       doublePriceMask(input);
644       break;
645     case 'objective-percentage':
646       objectivePercentageMask(input);
647       break;
648     case 'quantity':
649     case 'quantity-transaction':
650       doubleMask(input);
651       break;
652   }
653 }

```

Fonte: Autor

- e) gerar gráficos: a função **createChart** faz requisições para obter dados históricos de ativos e gera gráficos interativos de acordo com o período selecionado, permitindo análises visuais da performance do investimento;

Figura 38 - Exemplo de função da classe main.js que cria um gráfico utilizando a biblioteca Chart.js.

```

757 async function createChart(symbol) {
758     try {
759         const response = await fetch(`http://localhost:8000/chart_history?symbol=${symbol}`);
760         console.log(response);
761         if (!response.ok) {
762             throw new Error(`HTTP error! status: ${response.status}`);
763         }
764
765         const data = await response.json();
766
767         dados = JSON.parse(data);
768         chartSymbol = symbol;
769         refreshChart('12', dados);
770     } catch (error) {
771         console.error('Erro ao fazer requisição:', error);
772     }
773 }

```

Fonte: Autor

- f) Interagir com o backend e com API Externa: funções como **searchBySymbol** e **synchronizeUserAssets** fazem requisições HTTP para o backend da aplicação e a API externa do AlphaVantage, respectivamente buscando ativos por símbolo e sincronizando informações dos ativos do usuário, possibilitando a integração com serviços externos.

Figura 39 - Trecho de código referente a implementação da função searchBySymbol da classe main.js

```

430 async function searchBySymbol(symbol) {
431     const url = `http://localhost:8000/search by symbol?symbol=${symbol}`;
432
433     try {
434         const response = await fetch(url);
435
436         if (response.ok) {
437             const data = await response.json();
438             const resultArray = data.data.map(
439                 item => ({
440                     id: item.id,
441                     symbol: item.symbol,
442                     name: item.name,
443                     lastPrice: item.last_price
444                 })
445             );
446
447             return resultArray;
448         } else {
449             throw new Error('Erro na solicitação');
450         }
451     } catch (error) {
452         console.error('Falha na solicitação:', error);
453     }
454 }
455

```

Fonte: Autor

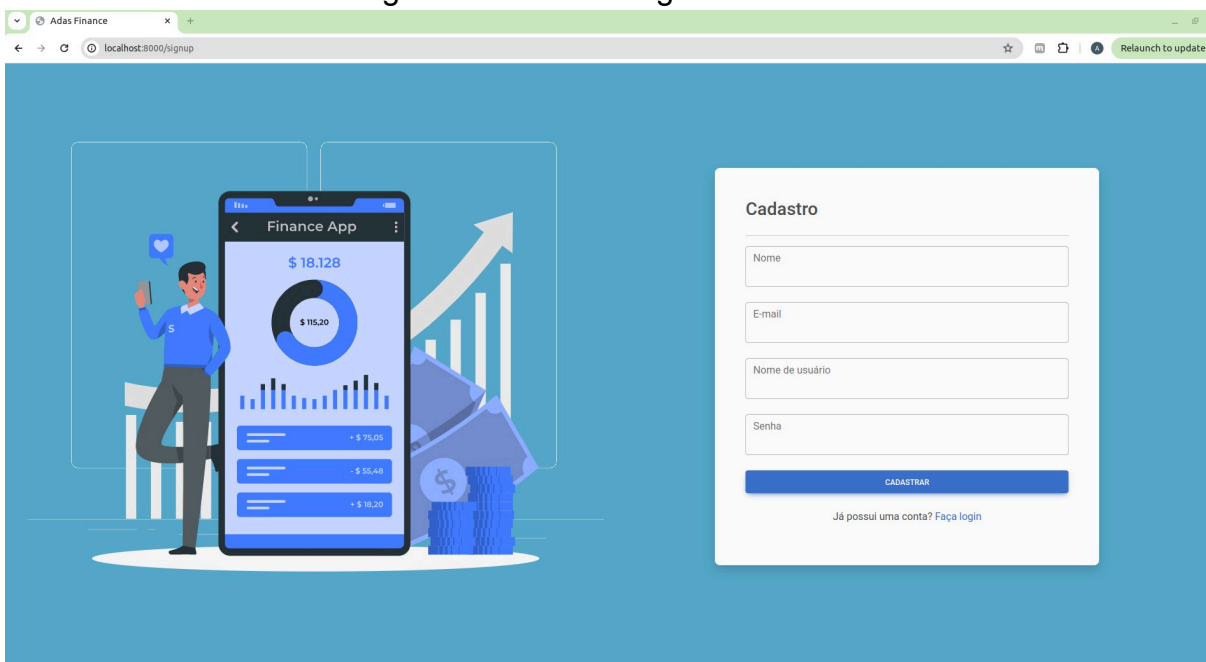
4.4.2.3 Chart.js

O Chart.js é utilizado para adicionar gráficos e visualizações de dados à camada de apresentação. Com ele, é possível criar gráficos interativos e responsivos que ajudam os usuários a interpretar informações de maneira clara e visualmente atraente. Foi utilizado para exibir os dados históricos do ativo na seção de detalhamento de ativo do usuário.

4.4.3 Telas

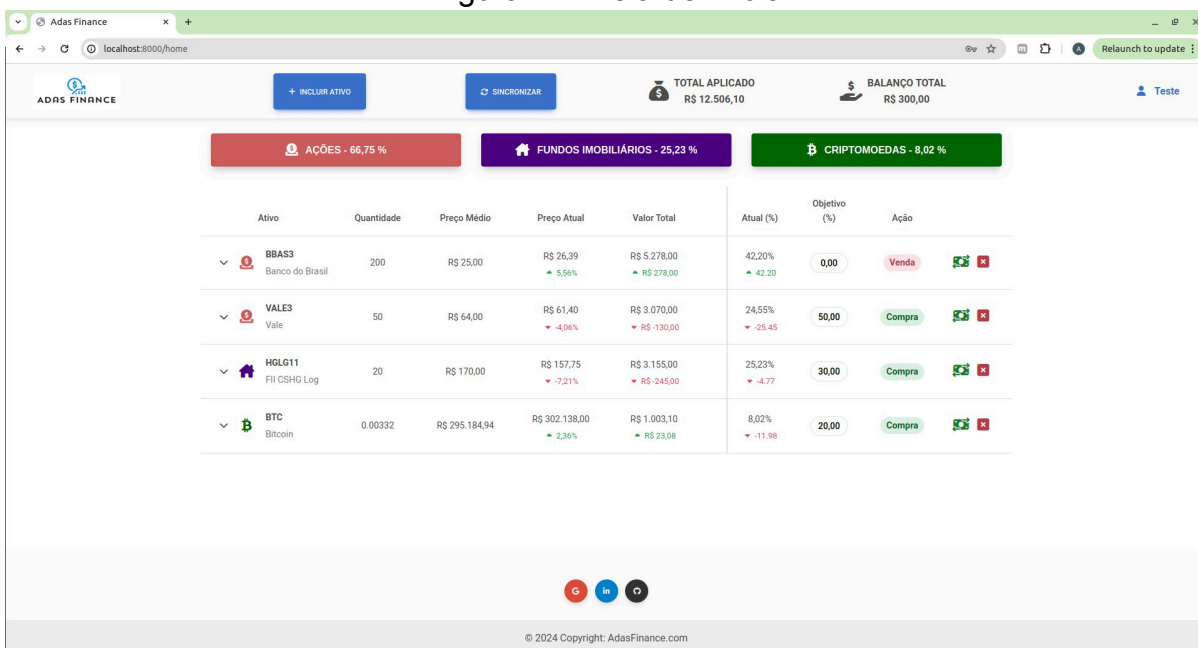
As principais telas da ferramenta podem ser vistas nas figuras abaixo.

Figura 40 - Tela de login e cadastro



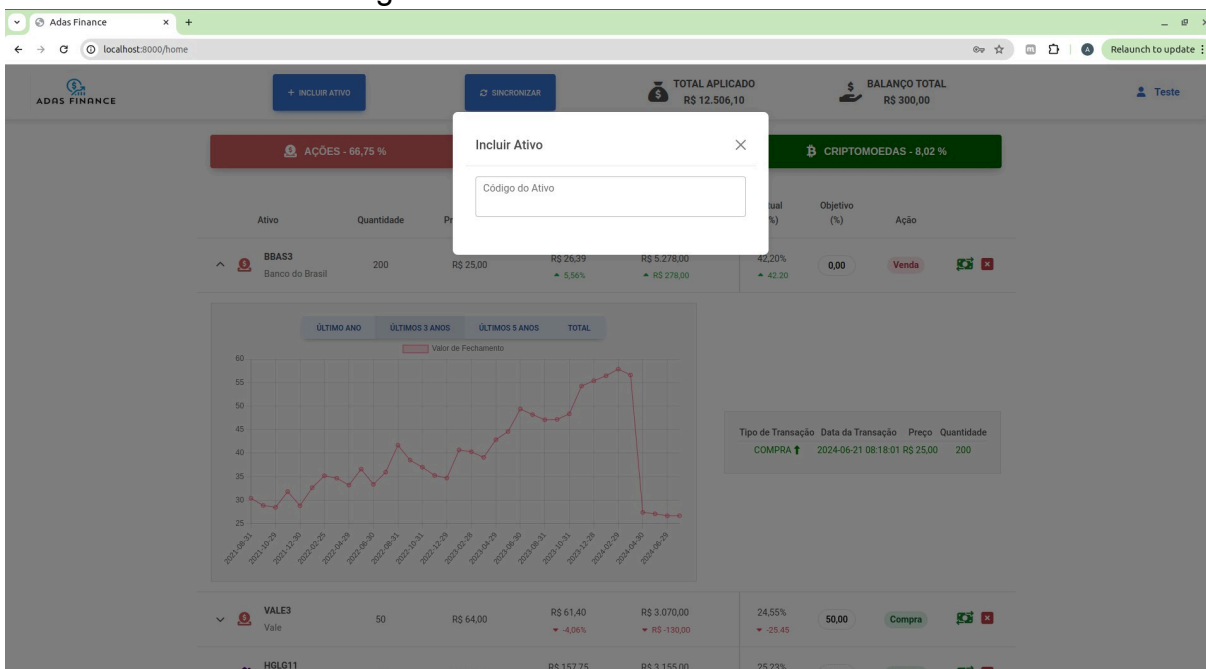
Fonte: Autor

Figura 41 - Tela de início



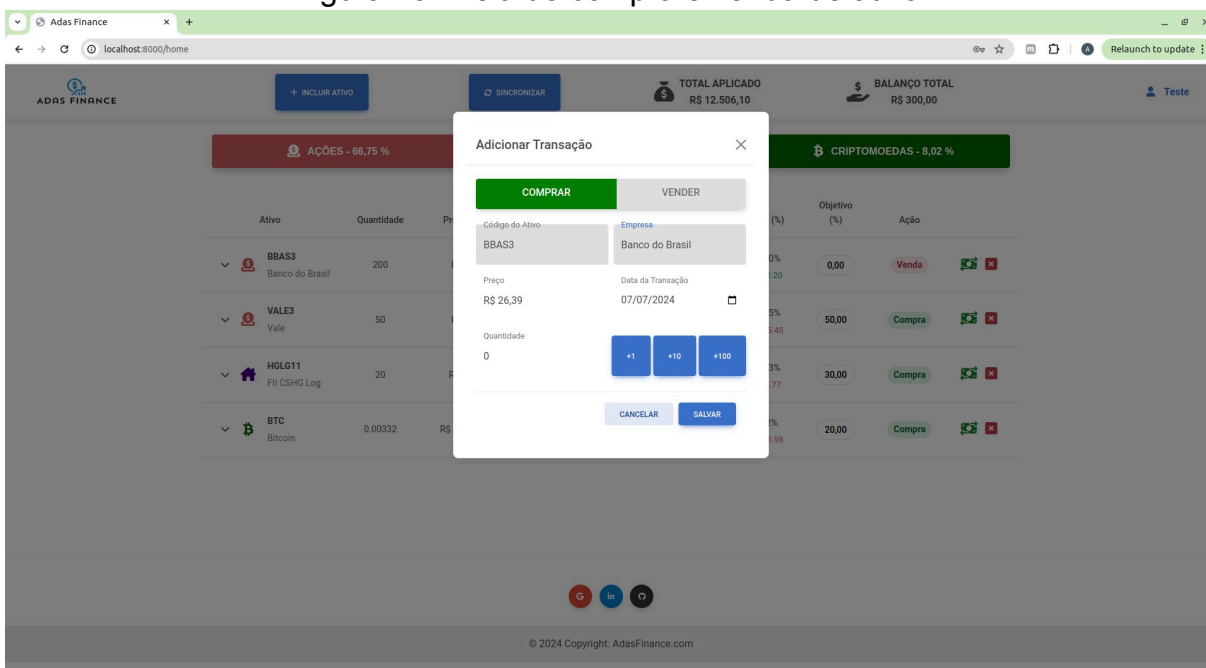
Fonte: Autor

Figura 42 - Tela de inclusão de ativo



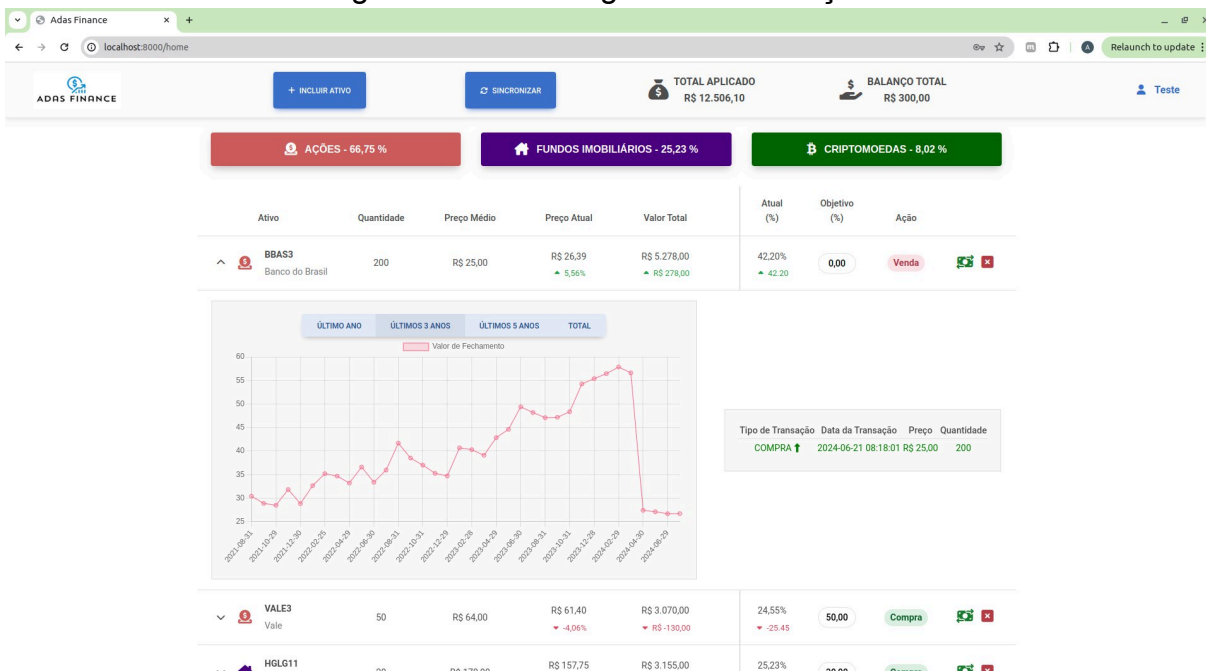
Fonte: Autor

Figura 43 - Tela de compra e venda de ativo



Fonte: Autor

Figura 44 - Tela de gráfico e transações



Fonte: Autor

4.5 VISÃO DE IMPLANTAÇÃO

No desenvolvimento deste trabalho, foi empregada a arquitetura LAMP, que consiste na integração dos softwares Linux, Apache, MySQL e PHP (Figura 39). Esta escolha pela familiaridade do autor com as ferramentas, amplamente reconhecida e utilizada na comunidade de desenvolvimento web.

O sistema operacional Linux foi selecionado pela sua estabilidade, segurança e eficiência na gestão de recursos. Este ambiente operacional oferece um suporte sólido para os demais componentes da arquitetura, garantindo um desempenho confiável e consistente.

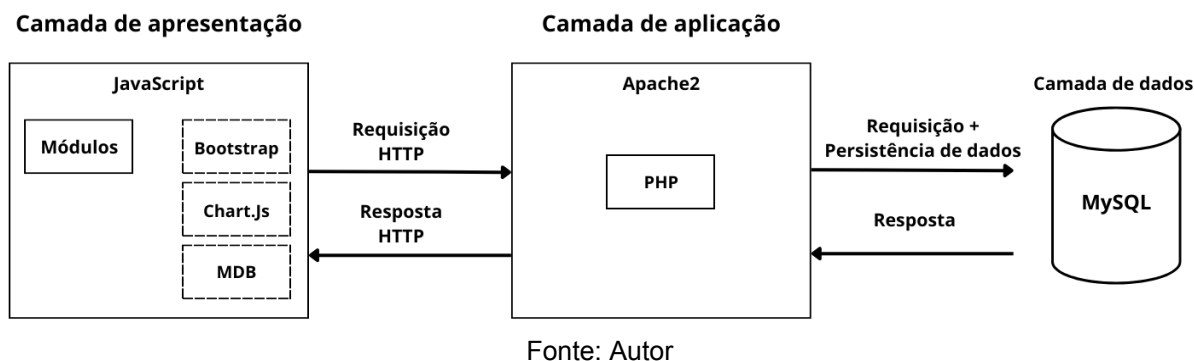
No contexto da arquitetura LAMP, o Apache2 serve como o servidor web responsável por receber e processar as requisições HTTP dos usuários. Ele interpreta as solicitações enviadas pelo frontend, entrega o conteúdo solicitado, como páginas HTML, imagens e arquivos, e executa scripts PHP quando necessário.

Para o gerenciamento de banco de dados, o MySQL foi utilizado devido à sua capacidade de lidar com os dados de maneira eficiente e segura. Sua integração com PHP facilita a execução de operações complexas de manipulação e consulta de dados, essencial para o funcionamento dinâmico da aplicação desenvolvida.

A linguagem de programação PHP foi escolhida por sua ampla adoção e facilidade de integração com o servidor Apache e o banco de dados MySQL. PHP possibilita a criação de páginas web dinâmicas e interativas, essenciais para a experiência do usuário. Sua sintaxe acessível e a vasta disponibilidade de suporte da comunidade e documentação contribuem para um desenvolvimento ágil e eficiente.

A escolha da arquitetura LAMP se mostrou adequada e eficaz para o desenvolvimento deste trabalho, oferecendo uma base sólida e confiável para a implementação das funcionalidades requeridas, além de garantir a escalabilidade e manutenção futura da aplicação.

Figura 45 - Diagrama de deployment



É possível acessar o código fonte da aplicação a partir do link disponibilizado no Apêndice C.

5 AVALIAÇÃO DA FERRAMENTA

Neste capítulo é apresentada uma avaliação da usabilidade e utilidade da ferramenta ADAS FINANCE realizada com a participação de potenciais usuários. Um instrumento de coleta de dados é definido a partir da aplicação do instrumento *System Usability Scale (SUS)* e de três perguntas adicionais. São convidados potenciais usuários como participantes da avaliação, com diferentes níveis de experiência no mercado financeiro, sendo que todos possuem contato prévio com o mercado financeiro e realizam investimentos.

5.1 DEFINIÇÃO DA AVALIAÇÃO

O estudo tem por objetivo avaliar a usabilidade e utilidade da ferramenta ADAS FINANCE sob o ponto de vista de potenciais usuários no contexto da gestão de carteiras de investimento.

Para avaliar a usabilidade da ferramenta é utilizado o *System Usability Scale (SUS)*, que foi desenvolvido por John Brooke e é uma escala simples de dez itens que oferece uma visão sobre a avaliação de três classes de medida de usabilidade: efetividade, eficiência e satisfação (Brooke, 1996). O *SUS* utiliza a escala Likert, o que significa que para cada afirmação o usuário deve assinalar um de cinco parâmetros, que variam entre 1 - Discordo Totalmente e 5 - Concordo Totalmente. As afirmações utilizadas são:

- a) Eu acho que gostaria de usar essa aplicação com frequência;
- b) Eu acho o sistema desnecessariamente complexo;
- c) Eu achei o sistema fácil de usar;
- d) Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema;
- e) Eu acho que as várias funções do sistema estão muito bem integradas;
- f) Eu acho que o sistema apresenta muitas inconsistências;
- g) Eu imagino que as pessoas aprenderão a usar esse sistema rapidamente.
- h) Eu achei o sistema complicado de usar.
- i) Eu me senti confiante ao usar o sistema.
- j) Eu precisei aprender várias coisas novas antes de usar o sistema.

O *SUS* consiste em uma ferramenta valiosa de avaliação, sendo robusta e confiável, além de ser uma ferramenta disponibilizada gratuitamente (Brooke, 1996).

Para avaliar a utilidade da ferramenta, além do *SUS*, são definidas três perguntas autorais para complementar a avaliação. São elas:

- k) Quais funcionalidades faltaram na ferramenta?
- l) De 0 a 10, o quão útil é a ferramenta para gerenciamento de carteira?
- m) Indique sugestões de melhoria para a ferramenta.

Todas as perguntas são reunidas em um único instrumento de avaliação na forma de um questionário a ser aplicado aos participantes por meio de entrevistas individuais realizadas após a utilização da ferramenta.

5.2 PARTICIPANTES DA AVALIAÇÃO

Os participantes do estudo de avaliação da ferramenta são selecionados utilizando amostragem por conveniência. São convidados como potenciais usuários da ferramenta pessoas conhecidas do autor, que possuam experiência prévia com gestão de carteiras pessoais de investimento e com formação em ensino superior de diferentes áreas do conhecimento. Foram convidados 5 participantes e todos aceitaram participar do estudo. O Quadro 6 apresenta a lista dos participantes do estudo.

Quadro 6 - Lista dos participantes do estudo

Participantes	Formação	Tempo de experiência
---------------	----------	----------------------

		com mercado financeiro
1	Direito	5 anos
2	Psicologia	1 ano
3	Medicina	2 anos
4	Sistemas de Informação	9 anos
5	Engenharia de Controle e Automação	8 anos

Fonte: Autor

5.3 PROCEDIMENTOS DA AVALIAÇÃO

A execução da avaliação foi realizada de maneira híbrida, permitindo a participação tanto online quanto presencialmente. Cada participante teve acesso à plataforma por meio de um código disponibilizado no GitHub, facilitando o acesso remoto e garantindo que todos os interessados pudessem participar independentemente de sua localização. Aqueles que preferiram ou necessitaram participar presencialmente puderam utilizar o sistema diretamente no notebook do autor, assegurando que todas as funcionalidades e aspectos técnicos fossem devidamente monitorados e suportados. Esta abordagem mista visou maximizar a conveniência e acessibilidade para os participantes, bem como garantir a integridade dos dados coletados durante a avaliação. A avaliação foi cuidadosamente planejada para que, independentemente do método de acesso escolhido, todos tivessem uma experiência uniforme e eficaz, contribuindo de maneira significativa para os resultados finais da pesquisa.

5.4 RESULTADOS DA AVALIAÇÃO

Os resultados da avaliação foram divididos em 3 etapas principais: apresentação do cálculo da pontuação do instrumento SUS, os números obtidos e as contribuições feitas pelos participantes por meio do questionário autoral.

5.4.1 Cálculo da pontuação do Instrumento *SUS* e resultados

De acordo com Brooke (1996), o *SUS* produz um único número que representa uma medida composta da usabilidade geral do sistema em estudo. Por ser uma escala Likert, o *SUS* possui itens inversos, que são perguntas formuladas de forma negativa para garantir que os participantes leiam e respondam com atenção sobre cada item. Nesse sentido, para calcular a pontuação *SUS*, primeiro foi somado as contribuições de pontuação de cada item de cada participante, a depender se o item é inverso ou não. O cálculo foi realizado da seguinte forma:

- Para os itens ímpares (inversos), a contribuição da pontuação é a posição da escala menos 1.
- Para os itens pares, a contribuição é 5 menos a posição da escala.
- A contribuição da pontuação de cada item, independente de ser inverso ou não, variará de 0 a 4.

Em seguida, foi feita a soma das contribuições dos itens e multiplicada por 2,5 para obter o valor global do *SUS* de cada participante. As pontuações finais do *SUS* variam de 0 a 100. É possível ver os resultados globais individuais obtidos a seguir.

Tabela 1 - Respostas Participante 1

(continua)

Item	Escala					Contribuição item
	1	2	3	4	5	
1				X		4-1 = 3
2	X					5-1 = 4
3					X	5-1 = 4
4	X					5-1 = 4
5					X	5-1 = 4
6		X				5-2 = 3
7				X		4-1 = 3
8	X					5-1 = 4
9					X	5-1 = 4
10	X					5-1 = 4

Tabela 1 - Respostas Participante 1

(conclusão)

Item	Escala					Contribuição item
	1	2	3	4	5	
SOMA x 2,5						92,5

Fonte: Autor

Tabela 2 - Respostas Participante 2

Item	Escala					Contribuição item
	1	2	3	4	5	
1				X		4-1 = 3
2	X					5-1 = 4
3				X		4-1 = 3
4		X				5-2 = 3
5					X	5-1 = 4
6	X					5-1 = 4
7				X		4-1 = 3
8		X				5-2 = 3
9				X		4-1 = 3
10			X			5-3 = 2
SOMA x 2,5						80

Fonte: Autor

Tabela 3 - Respostas Participante 3

(Continua)

Item	Escala					Contribuição item
	1	2	3	4	5	
1				X		4-1 = 3
2	X					5-1 = 4
3					X	5-1 = 4
4	X					5-1 = 4
5					X	5-1 = 4
6	X					5-1 = 4

Tabela 3 - Respostas Participante 3

(conclusão)

Item	Escala					Contribuição item
	1	2	3	4	5	
7					X	5-1 = 4
8	X					5-1 = 4
9				X		4-1 = 3
10	X					5-1 = 5
SOMA x 2,5						97,5

Fonte: Autor

Tabela 4 - Respostas Participante 4

Item	Escala					Contribuição item
	1	2	3	4	5	
1				X		4-1 = 3
2	X					5-1 = 4
3					X	5-1 = 4
4		X				5-2 = 3
5				X		4-1 = 3
6	X					5-1 = 4
7					X	5-1 = 4
8		X				5-2 = 3
9				X		4-1 = 3
10		X				5-2 = 3
SOMA x 2,5						85

Fonte: Autor

Tabela 5 - Respostas Participante 5

(continua)

Item	Escala					Contribuição item
	1	2	3	4	5	
1					X	5-1 = 4
2	X					5-1 = 4
3					X	5-1 = 4

Tabela 5 - Respostas Participante 5

(conclusão)

Item	Escala					Contribuição item
	1	2	3	4	5	
4	X					5-1 = 4
5				X		4-1 = 3
6	X					5-1 = 4
7					X	5-1 = 4
8	X					5-1 = 4
9			X			3-1 = 2
10	X					5-1 = 4
SOMA x 2,5						92,5

Fonte: Autor

A fim de saber o valor global final do instrumento aplicado, foi realizada uma média aritmética dos valores obtidos de cada participante, como pode ser visto na tabela abaixo.

Tabela 6 - Valor global final SUS

Participante	Soma	Pontuação SUS
1	37	92,5
2	32	80
3	39	97,5
4	34	85
5	37	92,5
Total	-	447,5
Média e valor global final	-	89,5

Fonte: Autor

De acordo com Bangor, Kortum e Miller (2008), produtos que são pelo menos aceitáveis têm pontuação SUS acima de 70, com produtos melhores pontuando entre 70 e 80 da escala e produtos verdadeiramente superiores com pontuação superior a 90. Em relação à avaliação do resultado do valor global do

SUS dos 5 participantes, obteve-se escore de 89,5, muito próximo ao valor de referência 90, que aponta um *produto verdadeiramente superior*. Assim, pode-se afirmar que a ferramenta desenvolvida pelo autor possui níveis aceitáveis - e quase excelentes - de eficácia, eficiência e satisfação preconizados pelo SUS.

5.4.2 Resultados do questionário aural

Em relação às respostas do questionário aural, obteve-se os seguintes resultados compilados:

Quadro 7 - Respostas compiladas dos participantes do questionário aural
(continua)

Participante	Perguntas		
	Quais funcionalidades faltaram na ferramenta?	Indique sugestões de melhoria para a ferramenta.	De 0 a 10, o quão útil é a ferramenta para gerenciamento de carteira?
1	Gráficos mais completos, integrações automáticas, comodidades de uso	Botão sincronizar podia ser automático	8
2	Ferramentas gráficas mais detalhadas, além de um espaçamento de tempo menor (1 mês, 3 meses, 6 meses, etc) para conseguir ver a curto prazo.	Melhorar a interação do usuário com o gráfico das ações	9

Quadro 7 - Respostas compiladas dos participantes do questionário autoral
(continuação)

Participante	Perguntas		
	Quais funcionalidades faltaram na ferramenta?	Indique sugestões de melhoria para a ferramenta.	De 0 a 10, o quão útil é a ferramenta para gerenciamento de carteira?
3	Talvez uma função para comprar ou vender a partir de um valor pré-estabelecido. Uma ordem de compra ou venda ao atingir um determinado valor.	Na minha opinião: Adicionar as ferramentas previamente citadas poderia auxiliar, porém acabaria poluindo visualmente a plataforma. O que prejudica muito quem é iniciante e dificulta a adaptação. Se implementadas, não seria bom colocar na página principal/de entrada, e sim em alguma aba a parte.	8.5
4	Integração com CEI para puxar os dados automaticamente. CDBs, tesouro direto e rentabilidade desses ativos. Comparação de rentabilidade com Ibovespa e similares no mesmo período	Implementar as funcionalidades citadas acima. Principalmente integrar com CEI	9

Quadro 7 - Respostas compiladas dos participantes do questionário autoral
(conclusão)

Participante	Perguntas		
	Quais funcionalidades faltaram na ferramenta?	Indique sugestões de melhoria para a ferramenta.	De 0 a 10, o quão útil é a ferramenta para gerenciamento de carteira?
5	Informar data de última sincronização com banco, rentabilidade total da carteira	Sincronização automática dos valores dos ativos. Balancear a carteira é uma tarefa bastante comum na vida de um investidor e, por falta de tecnologias especializadas, muitos investidores acabam utilizando planilhas. Essas acabam sendo mais propensas a erro do que um sistema especializado de TI. A ferramenta pode ser testada automaticamente. Por poder ser utilizada por inúmeros usuários, erros são mais fáceis de serem identificados.	9

Fonte: autor

Diante do exposto, é possível inferir que a ferramenta atual apresenta algumas limitações que podem dificultar a gestão eficiente da carteira de investimentos. Os gráficos disponíveis carecem de detalhes, não permitindo uma análise aprofundada da evolução dos ativos a curto prazo. Outra deficiência observada é a ausência de funcionalidades avançadas, como a possibilidade de estabelecer ordens de compra ou venda com base em valores pré-definidos, o que poderia facilitar a tomada de decisões e a execução de operações. Adicionalmente,

a ferramenta não oferece comparações entre a rentabilidade da carteira do usuário e benchmarks como o Ibovespa ou outros índices, dificultando a avaliação do desempenho dos investimentos.

Por fim, a ferramenta não fornece informações importantes, como a data da última sincronização com o banco de dados e a rentabilidade total da carteira, prejudicando o acompanhamento e a análise dos investimentos. Essas limitações evidenciam a necessidade de melhorias na ferramenta, a fim de proporcionar uma experiência mais completa e eficiente aos usuários.

Contudo, a ferramenta ainda assim apresentou ser eficiente e útil aos usuários testantes, alcançando escore de 89,5 no *SUS*, e boas notas no questionário autoral, como é orientado por Bangor, Kortum e Miller (2008). Nesse sentido, é possível citar alguns pontos fortes da ferramenta, como por exemplo a usabilidade simples e intuitiva, ideal para usuários com pouca experiência, além do balanço total da carteira, uma vez que este não é comum ser encontrado nas plataformas concorrentes, e a possibilidade do usuário de definir seus objetivos para cada ativo de forma individual.

O resultado obtido, de maneira geral, reflete em uma avaliação positiva, mas que não pode ser considerada perfeita. Assim, a análise detalhada das áreas de deficiência apontadas pode orientar esforços direcionados para aprimorar a experiência do usuário e, conseqüentemente, elevar a qualidade e eficácia do produto.

5.5 COMPARAÇÃO

	Ferramenta Desenvolvida	Ferramentas Avaliadas		
	AdasFinance	StatusInveste	GorillaApp	Fundamentei
Interface	Minimalista	Mais completa; robusta	Simple	Simple
Complexidade	Pouca	Alta	Pouca	Pouca
Integração	Integração com AlphaVantage	Integração com mercado BITCOIN, B3, FoxBit	Integração com B3	Pouca integração
Funcionalidade	Calculadora de	Fornece mais	Relatório por	Calculadora

	investimento; não possui relatórios e dados de evolução patrimonial	indicadores para análise mais detalhada de ativos; Relatório por período, de proventos	período, de proventos	de pesos; não possui relatório e dados de evolução patrimonial
Utilidade	Boa para iniciantes	Investidores mais experientes	Boa para iniciantes	Boa para iniciantes

Fonte: autor

6 CONCLUSÃO

Neste trabalho de conclusão de curso foi apresentado um sistema de controle de carteira de ativos financeiros centrado nos objetivos de investimento do usuário desenvolvido com o propósito de monitorar os ativos por meio de funções de maneira clara e objetiva, tendo assim alcançado êxito no cumprimento do objetivo principal.

A ferramenta foi testada por cinco participantes, e os resultados obtidos foram satisfatórios quando comparados com a literatura. Entre os objetivos específicos, cabe destacar que todos foram atingidos, desde a análise de trabalhos correlatos até a avaliação final do sistema, o que indica que a metodologia utilizada foi adequada e os critérios de avaliação foram bem estabelecidos. Assim, conforme a avaliação, a ferramenta sugere que o sistema é eficaz e cumpre suas funções conforme esperado. Portanto, os resultados positivos corroboram a eficácia da ferramenta e a viabilidade de sua implementação em contextos similares.

A ferramenta atual de gestão de investimentos apresenta limitações que dificultam uma análise detalhada e a tomada de decisões eficientes. Os gráficos oferecidos são superficiais, impossibilitando uma avaliação precisa da evolução dos ativos a curto prazo. Além disso, a ausência de funcionalidades avançadas, como ordens de compra ou venda baseadas em valores pré-definidos, limita a eficácia da gestão da carteira. A ferramenta também não permite comparações entre a rentabilidade da carteira e benchmarks como o Ibovespa, o que dificulta a avaliação de desempenho. A falta de informações sobre a data da última sincronização e a rentabilidade total da carteira também prejudica o acompanhamento dos investimentos.

Apesar dessas limitações, a ferramenta mostrou ser eficiente e útil, alcançando uma pontuação de 89,5 no SUS e boas avaliações em questionários específicos. Entre seus pontos fortes estão a usabilidade simples e intuitiva, adequada para usuários com pouca experiência, e a função de balanço total da carteira, não comum em plataformas concorrentes. Além disso, permite que os usuários definam objetivos individuais para cada ativo. Esses aspectos positivos

indicam uma avaliação geral favorável, mas destacam a necessidade de melhorias nas áreas identificadas para aprimorar a experiência do usuário e a eficácia do produto.

Assim, os elementos do sistema a serem aperfeiçoados revelam a necessidade de um aprimoramento constante, principalmente em relação às ferramentas gráficas e sua interação com o usuário, além das sugestões de implementação de integrações com CEI e de sincronização. Estas melhorias potenciais podem aumentar ainda mais a usabilidade e funcionalidade do sistema, proporcionando uma experiência mais intuitiva e eficiente para os usuários finais. Portanto, futuras versões da ferramenta devem focar nesses aspectos para alcançar um nível superior de excelência e satisfação do usuário.

REFERÊNCIAS

- BACH, T. M.; SILVA, W. V. da; KUDLAWICZ, C.; MARQUES, S. Eficiência das Companhias Abertas e o Risco versus Retorno das Carteiras de Ações a partir do Modelo de Markowitz. **Revista Evidenciação Contábil & Finanças**, [S. l.], v. 3, n. 1, p. 34–53, 2015. Disponível em: <https://periodicos.ufpb.br/index.php/recfin/article/view/21312>. Acesso em: 12 jun. 2024.
- BANGOR, Aaron; KORTUM, Philip T.; MILLER, James T. An Empirical Evaluation of the System Usability Scale. **Intl. Journal of Human-Computer Interaction**, [s. l.], v. 24, n. 6, p. 574-594, 30 jul. 2008. Disponível em: <https://doi.org/10.1080/10447310802205776>. Acesso em: 6 jun. 2024.
- BODIE, Zvi; KANE, Alex; MARCUS, Alan J. **Fundamentos de investimentos**. Tradução de Beth Honorato. 9ª ed. Porto Alegre, AMGH Editora, 2014.
- BROOKE, John. SUS: A 'Quick and Dirty' Usability Scale. In: JORDAN, Patrick W.; THOMAS B.; MCCLELLAND, Ian Lyall; WEERDMEEESTER, Bernard. **Usability Evaluation In Industry**. London: CRC Press, 1996.
- COHN, Mike. **User Stories Applied for Agile Software Development**. Addison-Wesley, 2009.
- CONVERSE, T.; PARK, J. **PHP: a bíblia**. Gulf Professional Publishing, 2003.
- FOWLER, M. **Patterns of Enterprise Application Architecture**. Addison-Wesley, 2020.
- FUNDAMENTEI. **Termos de Serviço**. 2023. Disponível em: <https://fundamentei.com/terms>. Acesso em: 04 jun. 2024.
- MARKOWITZ, H. M. Seleção de Carteiras. **The Journal of Finance**, v. 7, n. 1, p. 77-91, 1952. doi:10.2307/2975974.
- MENDES, Victor; ABREU, Margarida. **Cultura financeira dos investidores e diversificação das carteiras**. Instituto Superior de Economia e Gestão - DE Working papers 11/2006/DE/CISEP. Lisboa: ISEG – Departamento de Economia, 2006. Disponível em: <http://hdl.handle.net/10400.5/863>. Acesso em: 03 dez. 2023
- MELONI, Julie; KYRNIN, Jennifer. **HTML, CSS, and JavaScript All in One**. 3ª ed. Sams Publishing, 2018.
- MySQL**. Disponível em: <https://www.mysql.com/>. Acesso em: 25 maio 2024.
- NUNES, Maurício S.; COSTA JUNIOR, Newton C. A. da; MEURER, Roberto. A relação entre o mercado de ações e as variáveis macroeconômicas: uma análise econométrica para o Brasil. **Revista Brasileira de Economia**, Rio de Janeiro, v. 59,

n. 4, p. 585-607, dez. 2005. Disponível em: <http://dx.doi.org/10.1590/s0034-71402005000400004>. Acesso em: 03 dez. 2023.

ROBBINS, J. N. **Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics**. 4th ed. Sebastopol: O'Reilly Media, 2012.

SCOTTÁ DOS PASSOS, Vinicius C.; PINHEIRO, Juliano L. Estratégias De Investimento Em Bolsa De Valores: Uma Pesquisa Exploratória Da Visão Fundamentalista De Benjamin Graham. **Revista Gestão & Tecnologia**. 2010. Disponível em: <https://doi.org/10.20397/2177-6652/2009.v9i1.233>. Acesso em 07 jul. 2024.

SILVA, M.S. **HTML5: A linguagem de marcação que revolucionou a web**. 2ª ed. Novatec Editora, 2019.

STUMPF, K. **Rebalanceamento de carteira: o que é, vantagens e como fazer**. 2024. Disponível em: <https://www.topinvest.com.br/rebalanceamento-de-carreira/#:~:text=O%20rebalanceamento%20de%20carteira%20%C3%A9,estimada%20e%20minimiza%C3%A7%C3%A3o%20de%20riscos>. Acesso em: 07 jul. 2024.

VITOR, M. **Status Invest: é a melhor plataforma de análise gratuita?**. 2024. Disponível em: <https://www.mobills.com.br/blog/investimentos/status-invest/>. Acesso em: 07 jul. 2024.

APÊNDICE I - Histórias de usuários 04 a 11

Quadro 8 - História de Usuário 04

US04 - Como Investidor, eu devo conseguir acessar a página de início.

Critérios de aceite:

US04.01 - Após o login bem-sucedido, o investidor deve ser redirecionado automaticamente para a página de início.

US04.02 - As informações exibidas na página de início, como o desempenho dos ativos, devem ser atualizadas em tempo real ou com uma frequência aceitável para garantir que o investidor tenha acesso a dados recentes.

US04.03 - Todas as informações e funções devem estar funcionando e disponíveis ao usuário.

Requisitos:

RF02

Protótipos de tela:

The screenshot shows the Adas Finance dashboard. At the top, there are navigation buttons: '+ INCLUIR ATIVO' and 'SINCRONIZAR'. To the right, it displays 'TOTAL APLICADO R\$ 27.695,02' and 'BALANÇO TOTAL R\$ 109,32'. Below this, there are three colored boxes representing asset categories: 'AÇÕES - 81,72%' (red), 'FUNDOS IMOBILIÁRIOS - 13,58%' (purple), and 'CRIP TOMOEDAS - 4,70%' (green). The main part of the dashboard is a table with the following columns: Ativo, Quantidade, Preço Médio, Preço Atual, Valor Atual, Atual (%), Objetivo (%), and Ação. The table lists five assets: ABCB4 (Banco do Brasil), VALE3 (Vale), HGLG11 (FII CBHG Log), XPLG11 (FII XP Log), and BTC (Bitcoin). Each row includes a 'VENDA' or 'COMPRA' button and a small icon for the asset.

Ativo	Quantidade	Preço Médio	Preço Atual	Valor Atual	Atual (%)	Objetivo (%)	Ação
ABCB4 Banco do Brasil	100	R\$ 22,22	R\$ 23,15 ▲ 4,19%	R\$ 2.315,00 ▲ R\$ 62,00	8,36% ▲ 0,36	0,00	VENDA
VALE3 Vale	321	R\$ 65,83	R\$ 63,29 ▼ -3,89%	R\$ 20.316,09 ▼ R\$ -315,34	73,36% ▲ 23,36	50,00	VENDA
HGLG11 FII CBHG Log	10	R\$ 152,63	R\$ 166,63 ▲ 9,22%	R\$ 1.666,30 ▲ R\$ 162,00	6,02% ▲ 1,32	5,00	VENDA
XPLG11 FII XP Log	20	R\$ 107,54	R\$ 104,74 ▼ -2,60%	R\$ 2.094,80 ▼ R\$ -56,00	7,56% ▼ -2,44	10,00	COMPRA
BTC Bitcoin	0.00431203	R\$ 298.083,32	R\$ 302.138,00 ▲ 1,36%	R\$ 1.302,83 ▲ R\$ 17,46	4,70% ▼ -15,30	20,00	COMPRA

Fonte: Autor

Quadro 9 - História de Usuário 05

(continua)

US05 - Como Investidor, eu devo conseguir configurar o valor objetivo.

Critérios de aceite:

US05.01 - O investidor deve conseguir acessar a funcionalidade de configuração do valor objetivo a partir da página de detalhes da ação ou da página de gerenciamento de carteira.

US05.02 - Deve haver um botão ou link claramente identificado para configurar o valor objetivo.

US05.03 - O sistema deve permitir ao investidor inserir o valor percentual desejado para a ocupação da ação na carteira.

US05.04 - O campo para inserir o valor percentual deve ser claro e intuitivo, permitindo a entrada de valores numéricos entre 0% e 100%.

US05.05 - A seção deve ser atualizada para refletir o novo valor objetivo configurado.

US05.06 - O investidor deve poder editar o valor objetivo a qualquer momento, seguindo o mesmo processo de configuração inicial.

US05.07 - Deve ser possível visualizar e modificar o valor objetivo já configurado sem necessidade de reconfiguração completa.

US05.08 - Se a configuração do valor objetivo impactar outras áreas ou cálculos na carteira de investimentos, o sistema deve fornecer feedback em tempo real sobre essas mudanças.

Requisitos:

RF04

Quadro 9 - História de Usuário 05

(conclusão)

Protótipos de tela:

The screenshot shows the Adas Finance web application. At the top, there are navigation buttons for '+ INCLUIR ATIVO' and 'SINCRONIZAR'. The main header displays 'TOTAL APLICADO R\$ 27.695,02' and 'BALANÇO TOTAL R\$ 109,32' next to a user profile icon for 'arthur'. Below this, three colored bars represent asset categories: 'AÇÕES - 81,72%' (red), 'FUNDOS IMOBILIÁRIOS - 13,58%' (purple), and 'CRIPOMOEDAS - 4,70%' (green). The main content is a table with the following columns: Ativo, Quantidade, Preço Médio, Preço Atual, Valor Atual, Atual (%), Objetivo (%), and Ação.

Ativo	Quantidade	Preço Médio	Preço Atual	Valor Atual	Atual (%)	Objetivo (%)	Ação
ABCB4 Banco do Brasil	100	R\$ 22,22	R\$ 23,15 ▲ 4,16%	R\$ 2.315,00 ▲ R\$ 93,00	8,36% ▲ 8,36	0,00	VENDA
VALE3 Vale	321	R\$ 65,83	R\$ 63,29 ▼ -3,86%	R\$ 20.316,09 ▼ R\$ -815,34	73,36% ▲ 23,36	50,00	VENDA
HGLG11 FII CSHG Log	10	R\$ 152,63	R\$ 166,63 ▲ 9,22%	R\$ 1.666,30 ▲ R\$ 142,00	6,02% ▲ 1,02	5,00	VENDA
XPLG11 FII XP Log	20	R\$ 107,54	R\$ 104,74 ▼ -2,60%	R\$ 2.094,80 ▼ R\$ -86,00	7,56% ▼ -2,44	10,00	COMPRA
BTC Bitcoin	0,00431203	R\$ 298.083,32	R\$ 302.138,00 ▲ 1,36%	R\$ 1.302,83 ▲ R\$ 17,46	4,70% ▼ -15,30	20,00	COMPRA

The image shows two zoomed-in views of the 'Ação' column from the main table. The left view shows the 'Atual (%)' field with a value of 8,36% and a change of ▲ 8,36. The 'Objetivo (%)' field is currently set to 0,00 and is being interacted with by a hand cursor. The 'Ação' field is 'VENDA'. The right view shows the same fields for the 'VALE3' asset, where the 'Objetivo (%)' is set to 50,00 and the 'Ação' is 'VENDA'. For the 'XPLG11' asset, the 'Objetivo (%)' is 10,00 and the 'Ação' is 'COMPRA'. For the 'BTC' asset, the 'Objetivo (%)' is 20,00 and the 'Ação' is 'COMPRA'.

Fonte: Autor

Quadro 10 - História de Usuário 06

(continua)

US06 - Como Investidor, eu devo conseguir cadastrar um ativo.

Quadro 10 - História de Usuário 06

(continuação)

Critérios de aceite:

US06.01 O investidor deve conseguir acessar a funcionalidade de inserção de ativos a partir da seção específica para gerenciamento de ativos na página principal da carteira de investimentos.

US06.02 - Deve haver um botão claramente identificado como "Incluir Ativo".

US06.03 - O sistema deve validar que todos os campos obrigatórios foram preenchidos.

US06.04 - Os valores numéricos, como quantidade e preço de compra, devem ser validados para garantir que são números positivos.

US06.05 - A data da compra deve ser uma data válida e não pode ser uma data futura.

US06.06 - O ativo inserido deve ser salvo de forma persistente no banco de dados, garantindo que ele seja mantido mesmo após o logout e novo login do investidor.

US06.07 - A interface da carteira de investimentos deve ser atualizada para mostrar o novo ativo, incluindo todas as informações relevantes (nome, tipo, quantidade, preço de compra, valor atual, etc.).

US06.08 - Gráficos devem refletir a inclusão do novo ativo.

US06.09 - Após a inserção, o investidor deve ter a opção de editar ou remover o ativo inserido, com funcionalidades claras para essas ações.

Requisitos:

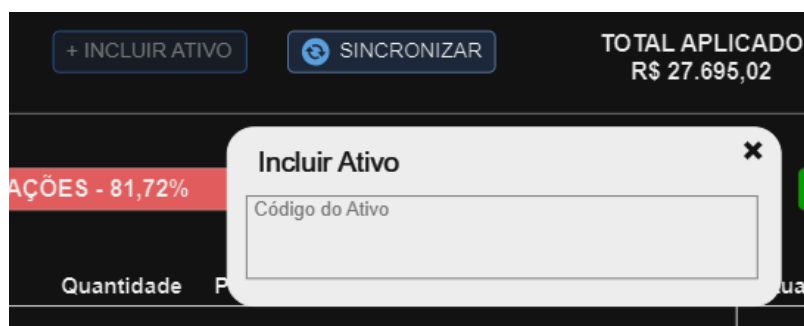
RF05, RF06, RF08, RF10

Quadro 10 - História de Usuário 06

(conclusão)

Protótipos de tela:

Ativo	Quantidade	Preço Médio	Preço Atual	Valor Atual	Atual (%)	Objetivo (%)	Ação
ABCB4 Banco do Brasil	100	R\$ 22,22	R\$ 23,15 ▲ 4,19%	R\$ 2.315,00 ▲ R\$ 93,00	8,36%	10,00	VENDA
VALE3 Vale	321	R\$ 65,83	R\$ 63,29 ▼ -3,88%	R\$ 20.316,09 ▼ R\$ -815,34	73,36%	50,00	VENDA
HGLG11 FII CSHG Log	10	R\$ 152,63	R\$ 166,63 ▲ 9,23%	R\$ 1.666,30 ▲ R\$ 140,00	6,02%	5,00	VENDA
XPLG11 FII XP Log	20	R\$ 107,54	R\$ 104,74 ▼ -2,60%	R\$ 2.094,80 ▼ R\$ -56,00	7,56%	10,00	COMPRA
BTC Bitcoin	0.00431203	R\$ 298.083,32	R\$ 302.138,00 ▲ 1,36%	R\$ 1.302,83 ▲ R\$ 17,46	4,70%	20,00	COMPRA



Fonte: Autor

Quadro 11 - História de Usuário 07

(continua)

US07 - Como Investidor, eu devo conseguir deletar um ativo.

Critérios de aceite:

US07.01 - O investidor deve conseguir acessar a funcionalidade de exclusão de ativos a partir de uma seção específica para gerenciamento de ativos na página principal da carteira de investimentos.

Quadro 11 - História de Usuário 07

(continuação)

Critérios de aceite:

US07.02 - Deve haver um botão claramente identificado como "Excluir Ativo" próximo a cada ativo listado na carteira.

US07.03 - Após a confirmação, o sistema deve remover o ativo selecionado da carteira do investidor de forma segura.

US07.04 - A interface da carteira de investimentos deve ser atualizada imediatamente para refletir a remoção do ativo.

US07.05 - Gráficos da carteira devem ser atualizados para refletir a mudança.

US07.06 - O sistema deve garantir que a deleção do ativo seja persistente, garantindo que ele não reapareça após o logout e novo login do investidor.

Requisitos:

RF07, RF08, RF10

Protótipos de tela:

Ativo	Quantidade	Preço Médio	Preço Atual	Valor Atual	Atual (%)	Objetivo (%)	Ação
ABCB4 Banco do Brasil	100	R\$ 22,22	R\$ 23,15 ▲ 4,19%	R\$ 2.315,00 ▲ R\$ 93,00	8,36% ▲ 2,95%	10,00	VENDA
VALE3 Vale	321	R\$ 65,83	R\$ 63,29 ▼ -3,88%	R\$ 20.316,09 ▼ R\$ -815,34	73,36% ▲ 23,35%	50,00	VENDA
HGLG11 FII CSHG Log	10	R\$ 152,63	R\$ 166,63 ▲ 9,22%	R\$ 1.666,30 ▲ R\$ 142,00	6,02% ▲ 1,02%	5,00	VENDA
XPLG11 FII XP Log	20	R\$ 107,54	R\$ 104,74 ▼ -2,60%	R\$ 2.094,80 ▼ R\$ -58,00	7,56% ▼ -2,44%	10,00	COMPRA
BTC Bitcoin	0.00431203	R\$ 298.083,32	R\$ 302.138,00 ▲ 1,36%	R\$ 1.302,83 ▲ R\$ 17,48	4,70% ▼ -15,30%	20,00	COMPRA

Quadro 11 - História de Usuário 07

(conclusão)

Protótipos de tela:

Atual (%)	Objetivo (%)	Ação	
8,36% ▲ 8.36	10,00	VENDA	Excluir Ativo [Carteira] [X]
73,36% ▲ 23.36	50,00	VENDA	[Carteira] [X]
6,02% ▲ 1.02	5,00	VENDA	[Carteira] [X]
7,56%	10,00	VENDA	[Carteira] [X]

Fonte: Autor

Quadro 12 - História de Usuário 08

(continua)

US08 - Como Investidor, eu devo conseguir adicionar transação de compra ou venda.

Critérios de aceite:

US08.01 - O investidor deve conseguir acessar a funcionalidade de adicionar transações a partir de uma seção específica para gerenciamento de transações na página principal da carteira de investimentos.

US08.02 - Deve haver um botão claramente identificado como "Adicionar Transação".

US08.03 - O sistema deve validar que todos os campos obrigatórios foram preenchidos.

US08.04 - Os valores numéricos, como quantidade e preço unitário, devem ser validados para garantir que são números positivos.

US08.05 - A data da transação deve ser uma data válida e não pode ser uma data futura.

US08.06 - Para transações de venda, o sistema deve verificar que o investidor possui uma quantidade suficiente do ativo para vender.

Quadro 12 - História de Usuário 08

(continuação)

Critérios de aceite:

US08.07 - A transação inserida deve ser salva de forma persistente no banco de dados, garantindo que ela seja mantida mesmo após o logout e novo login do investidor.

US08.08 - A interface da carteira de investimentos deve ser atualizada para mostrar o impacto da nova transação, incluindo ajustes na quantidade de ativos e no valor total da carteira.

Requisitos:

RF09, RF10, RF11

Protótipos de tela:

Ativo	Quantidade	Preço Médio	Preço Atual	Valor Atual	Atual (%)	Objetivo (%)	Ação
ABCB4 <small>Banco do Brasil</small>	100	R\$ 22,22	R\$ 23,15 ▲ 4,19%	R\$ 2.315,00 ▲ R\$ 93,00	8,36% ▲ 8,36%	0,00	VENDA
VALE3 <small>Vale</small>	321	R\$ 65,83	R\$ 63,29 ▼ -3,88%	R\$ 20.316,09 ▼ R\$ -815,34	73,36% ▲ 23,38%	50,00	VENDA
HGLG11 <small>FII CSHG Log</small>	10	R\$ 152,63	R\$ 166,63 ▲ 9,32%	R\$ 1.666,30 ▲ R\$ 142,00	6,02% ▲ 1,02%	5,00	VENDA
XPLG11 <small>FII XP Log</small>	20	R\$ 107,54	R\$ 104,74 ▼ -2,60%	R\$ 2.094,80 ▼ R\$ -56,00	7,56% ▼ -2,44%	10,00	COMPRA
BTC <small>Bitcoin</small>	0.00431203	R\$ 298.083,32	R\$ 302.138,00 ▲ 1,38%	R\$ 1.302,83 ▲ R\$ 17,48	4,70% ▼ -15,30%	20,00	COMPRA

Quadro 12 - História de Usuário 08

(conclusão)

Protótipos de tela:

Atual (%)	Objetivo (%)	Ação
8,36% ▲ 8.36	0,00	VENDA
73,36% ▲ 23.36	50,00	VENDA
6,02% ▲ 1.02	5,00	VENDA

Fonte: Autor

Quadro 13 - História de Usuário 09

(continua)

US09 - Como Investidor, eu devo conseguir ver o gráfico de ações.

Critérios de aceite:

US09.01 - O investidor deve conseguir acessar a funcionalidade de visualização de gráficos de ações a partir da página de detalhes do ativo.

US09.02 - Deve haver um botão claramente identificado como "^" próximo ao nome do ativo.

US09.03 - Os dados do gráfico devem ser atualizados em tempo real ou com uma frequência aceitável para refletir as últimas informações de mercado.

US09.04 - Na parte superior do quadrante do gráfico, deve haver uma barra de opções de tempo, permitindo a visualização dos períodos: Último Ano, Últimos 3 Anos, Últimos 5 Anos e Total.

Requisitos:

RF08, RF10

Quadro 13 - História de Usuário 09

(continuação)

Protótipos de tela:

Adas Finance

https://www.adasfinance.com/inicio

+ INCLUIR ATIVO SINCROINIZAR

TOTAL APLICADO R\$ 27.695,02 BALANÇO TOTAL R\$ 109,32 arthur

AÇÕES - 81,72% FUNDOS IMOBILIÁRIOS - 13,58% CRIPTOMOEDAS - 4,70%

Ativo	Quantidade	Preço Médio	Preço Atual	Valor Atual	Atual (%)	Objetivo (%)	Ação
ABCB4 Banco do Brasil	100	R\$ 22,22	R\$ 23,15 ▲ 4,19%	R\$ 2.315,00 ▲ R\$ 93,00	8,36% ▲ 8,36	0,00	VENDA
VALE3 Vale	321	R\$ 65,83	R\$ 63,29 ▼ -3,86%	R\$ 20.316,09 ▼ R\$ -815,34	73,36% ▲ 23,36	50,00	VENDA
HGLG11 FII CSHG Log	10	R\$ 152,63	R\$ 166,63 ▲ 9,22%	R\$ 1.666,30 ▲ R\$ 142,00	6,02% ▲ 1,02	5,00	VENDA
XPLG11 FII XP Log	20	R\$ 107,54	R\$ 104,74 ▼ -2,60%	R\$ 2.094,80 ▼ R\$ -56,00	7,56% ▼ -2,44	10,00	COMPRA
BTC Bitcoin	0.00431203	R\$ 298.083,32	R\$ 302.138,00 ▲ 1,36%	R\$ 1.302,83 ▲ R\$ 17,48	4,70% ▼ -15,30	20,00	COMPRA

AÇÕES - 81,72% FUNDOS IMOBILIÁRIOS - 13,58% CRIPTOMOEDAS - 4,70%

Ativo	Quantidade	Preço Médio	Preço Atual	Valor Atual	Atual (%)	Objetivo (%)	Ação
ABCB4 Banco do Brasil	100	R\$ 22,22	R\$ 23,15 ▲ 4,19%	R\$ 2.315,00 ▲ R\$ 93,00	8,36% ▲ 8,36	0,00	VENDA
VALE3 Vale	321	R\$ 65,83	R\$ 63,29 ▼ -3,86%	R\$ 20.316,09 ▼ R\$ -815,34	73,36% ▲ 23,36	50,00	VENDA
HGLG11 FII CSHG Log	10	R\$ 152,63	R\$ 166,63 ▲ 9,22%	R\$ 1.666,30 ▲ R\$ 142,00	6,02% ▲ 1,02	5,00	VENDA
XPLG11 FII XP Log	20	R\$ 107,54	R\$ 104,74 ▼ -2,60%	R\$ 2.094,80 ▼ R\$ -56,00	7,56% ▼ -2,44	10,00	COMPRA
BTC Bitcoin	0.00431203	R\$ 298.083,32	R\$ 302.138,00 ▲ 1,36%	R\$ 1.302,83 ▲ R\$ 17,48	4,70% ▼ -15,30	20,00	COMPRA

Quadro 13 - História de Usuário 09

(conclusão)

Protótipos de tela:



Fonte: Autor

Quadro 14 - História de Usuário 10

(continua)

US10 - Como Investidor, eu devo conseguir ver o histórico de transações.

Critérios de aceite:

US10.01 - O investidor deve conseguir acessar o histórico de transações do ativo no mesmo local onde está localizado o gráfico dessa ação..

US10.02 - O sistema deve exibir uma lista completa de transações realizadas, incluindo: Tipo de Transação (Compra ou Venda); Data da Transação; Valor Total da Transação e Quantidade.

US10.03 - O histórico de transações deve ser atualizado em tempo real ou com frequência aceitável para refletir as transações mais recentes.

Requisitos:

RF08, RF11

Quadro 14 - História de Usuário 10

(conclusão)

Protótipos de tela:



Fonte: Autor

Quadro 15 - História de Usuário 11

(continua)

US11 - Como Investidor, devo conseguir sincronizar os preços dos ativos com os preços (mais atualizados) da API do AlphaVantage.

Critérios de aceite:

US11.01 - O investidor deve conseguir acessar a funcionalidade de sincronização de preços dos ativos com a API do AlphaVantage na parte superior da página principal da carteira de investimentos.

US11.02 - Deve haver um botão claramente identificado como "Sincronizar".

Quadro 15 - História de Usuário 11

(conclusão)

Critérios de aceite:

US11.03 - O sistema deve se comunicar de forma eficiente com a API do AlphaVantage para obter os preços mais atualizados dos ativos na carteira do investidor.

US11.04 - Qualquer alteração significativa nos preços deve ser refletida imediatamente na interface do usuário.

Requisitos:

RF12

Protótipos de tela:

Fonte: Autor

APÊNDICE II - Respostas do aplicação do SUS e do questionário autoral

Participante 1

- Instrumento SUS

(continua)

Item	Escala				
	1 - Discordo totalmente	2 - Discordo Parcialmen te	3 - Neutro	4 - Concordo parcialment e	5 - Concordo totalmente
1) Eu acho que gostaria de usar essa aplicação com frequência				X	
2) Eu acho o sistema desnecessariamente complexo	X				
3) Eu achei o sistema fácil de usar					X
4) Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema	X				
5) Eu acho que as várias funções do sistema estão muito bem integradas					X
6) Eu acho que o sistema apresenta muitas inconsistências		X			
7) Eu imagino que as pessoas aprenderão a usar esse sistema rapidamente				X	
8) Eu achei o sistema complicado de usar	X				

- Instrumento *SUS*

(conclusão)

Item	Escala				
	1 - Discordo totalmente	2 - Discordo Parcialmen te	3 - Neutro	4 - Concordo parcialmen te	5 - Concordo totalmente
9) Eu me senti confiante ao usar o sistema					X
10) Eu precisei aprender várias coisas novas antes de usar o sistema	X				

Fonte: autor

- Questionário autoral

Perguntas abertas	Resposta abertas
Quais funcionalidades faltaram na ferramenta?	Gráficos mais completos, integrações automáticas, comodidades de uso
Indique sugestões de melhoria para a ferramenta.	Botão sincronizar podia ser automático
Pergunta fechada	Resposta fechada
De 0 a 10, o quão útil é a ferramenta para gerenciamento de carteira?	8

Fonte: autor

Participante 2

- Instrumento SUS

(continua)

Item	Escala				
	1 - Discordo totalmente	2 - Discordo Parcialmen te	3 - Neutro	4 - Concordo parcialment e	5 - Concordo totalmente
1) Eu acho que gostaria de usar essa aplicação com frequência				x	
2) Eu acho o sistema desnecessariamente complexo	x				
3) Eu achei o sistema fácil de usar				x	
4) Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema		x			
5) Eu acho que as várias funções do sistema estão muito bem integradas					x
6) Eu acho que o sistema apresenta muitas inconsistências	x				
7) Eu imagino que as pessoas aprenderão a usar esse sistema rapidamente				x	
8) Eu achei o sistema complicado de usar		x			
9) Eu me senti confiante ao usar o sistema				x	

- Instrumento *SUS*

(conclusão)

Item	Escala				
	1 - Discordo totalmente	2 - Discordo Parcialmen te	3 - Neutro	4 - Concordo parcialmen te	5 - Concordo totalmente
10) Eu precisei aprender várias coisas novas antes de usar o sistema			x		

Fonte: autor

- Questionário autoral

Perguntas abertas	Resposta abertas
Quais funcionalidades faltaram na ferramenta?	Ferramentas gráficas mais detalhadas, além de um espaçamento de tempo menor (1 mês, 3 meses, 6 meses, etc) para conseguir ver a curto prazo.
Indique sugestões de melhoria para a ferramenta.	Melhorar a interação do usuário com o gráfico das ações
Pergunta fechada	Resposta fechada
De 0 a 10, o quão útil é a ferramenta para gerenciamento de carteira?	9

Fonte: autor

Participante 3

- Instrumento *SUS*

(Continua)

Item	Escala				
	1 - Discordo totalmente	2 - Discordo Parcialmen te	3 - Neutro	4 - Concordo parcialment e	5 - Concordo totalmente
1) Eu acho que gostaria de usar essa aplicação com frequência				X	
2) Eu acho o sistema desnecessariamente complexo	X				
3) Eu achei o sistema fácil de usar					X
4) Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema	X				
5) Eu acho que as várias funções do sistema estão muito bem integradas					X
6) Eu acho que o sistema apresenta muitas inconsistências	X				
7) Eu imagino que as pessoas aprenderão a usar esse sistema rapidamente					X
8) Eu achei o sistema complicado de usar	X				
9) Eu me senti confiante ao usar o sistema				X	

- Instrumento *SUS*

(conclusão)

Item	Escala				
	1 - Discordo totalmente	2 - Discordo Parcialmen te	3 - Neutro	4 - Concordo parcialment e	5 - Concordo totalmente
10) Eu precisei aprender várias coisas novas antes de usar o sistema	X				

Fonte: autor

- Questionário autoral

Perguntas abertas	Resposta abertas
Quais funcionalidades faltaram na ferramenta?	Talvez uma função para comprar ou vender a partir de um valor pré-estabelecido. Uma ordem de compra ou venda ao atingir um determinado valor. Associar um gráfico do valor de cada ação em função do tempo também poderia ajudar.
Indique sugestões de melhoria para a ferramenta.	Na minha opinião: Adicionar as ferramentas previamente citadas poderia auxiliar, porém acabaria poluindo visualmente a plataforma. O que prejudica muito quem é iniciante e dificulta a adaptação. Se implementadas, não seria bom colocar na página principal/de entrada, e sim em alguma aba a parte.
Pergunta fechada	Resposta fechada
De 0 a 10, o quão útil é a ferramenta para gerenciamento de carteira?	8.5

Fonte: autor

Participante 4

- Instrumento *SUS*

(continua)

Item	Escala				
	1 - Discordo totalmente	2 - Discordo Parcialmen te	3 - Neutro	4 - Concordo parcialment e	5 - Concordo totalmente
1) Eu acho que gostaria de usar essa aplicação com frequência				x	
2) Eu acho o sistema desnecessariamente complexo	x				
3) Eu achei o sistema fácil de usar					x
4) Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema		x			
5) Eu acho que as várias funções do sistema estão muito bem integradas				x	
6) Eu acho que o sistema apresenta muitas inconsistências	x				
7) Eu imagino que as pessoas aprenderão a usar esse sistema rapidamente					x
8) Eu achei o sistema complicado de usar		x			
9) Eu me senti confiante ao usar o sistema				x	

- Instrumento *SUS*

(conclusão)

Item	Escala				
	1 - Discordo totalmente	2 - Discordo Parcialmen te	3 - Neutro	4 - Concordo parcialment e	5 - Concordo totalmente
10) Eu precisei aprender várias coisas novas antes de usar o sistema		x			

Fonte: autor

- Questionário autoral

Perguntas abertas	Resposta abertas
Quais funcionalidades faltaram na ferramenta?	Integração com CEI para puxar os dados automaticamente. CDBs, tesouro direto e rentabilidade desses ativos. Comparação de rentabilidade com Ibovespa e ETC no mesmo período
Indique sugestões de melhoria para a ferramenta.	Implementar as funcionalidades citadas acima. Principalmente integrar com CEI
Pergunta fechada	Resposta fechada
De 0 a 10, o quão útil é a ferramenta para gerenciamento de carteira?	9

Fonte: autor

Participante 5

- Instrumento *SUS*

(continua)

Item	Escala				
	1 - Discordo totalmente	2 - Discordo Parcialment e	3 - Neutro	4 - Concordo parcialment e	5 - Concordo totalmente
1) Eu acho que gostaria de usar essa aplicação com frequência					X
2) Eu acho o sistema desnecessariamente complexo	X				
3) Eu achei o sistema fácil de usar					X
4) Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema	X				
5) Eu acho que as várias funções do sistema estão muito bem integradas				X	
6) Eu acho que o sistema apresenta muitas inconsistências	X				
7) Eu imagino que as pessoas aprenderão a usar esse sistema rapidamente					X
8) Eu achei o sistema complicado de usar	X				
9) Eu me senti confiante ao usar o sistema			X		

- Instrumento *SUS*

(conclusão)

Item	Escala				
	1 - Discordo totalmente	2 - Discordo Parcialmen te	3 - Neutro	4 - Concordo parcialment e	5 - Concordo totalmente
10) Eu precisei aprender várias coisas novas antes de usar o sistema	X				

Fonte: autor

- Questionário autoral

Perguntas abertas	Resposta abertas
Quais funcionalidades faltaram na ferramenta?	Informar data de última sincronização com banco, rentabilidade total da carteira, histórico de operações
Indique sugestões de melhoria para a ferramenta.	Sincronização automática dos valores dos ativos
Pergunta fechada	Resposta fechada
De 0 a 10, o quão útil é a ferramenta para gerenciamento de carteira?	9/10 Balancear a carteira é uma tarefa bastante comum na vida de um investidor e, por falta de tecnologias especializadas, muitos investidores acabam utilizando planilhas. Essas acabam sendo mais propensas a erro do que um sistema especializado de TI. A ferramenta pode ser testada automaticamente. Por poder ser utilizada por inúmeros usuários, erros são mais fáceis de serem identificados.

Fonte: autor

APÊNDICE III - Código Fonte

O código fonte da aplicação está disponível em:
<https://github.com/arthur0696A/AdasFinance>

APÊNDICE IV - Artigo

Desenvolvimento de sistema de recomendação personalizado e otimizado de carteira de ativos

Arthur da Silva

Universidade Federal de Santa Catarina (UFSC) Departamento de Informática e Estatística — INE, Florianópolis, SC — Brasil

arthurdasilva96@gmail.com

Abstract. *This work presents the development of ADASFINANCE, a financial asset portfolio management system focused on the user's investment goals, offering a practical and simple alternative to existing platforms. The solution is a web application based on LAMP (Linux, Apache, MySQL, and PHP) with minimalist and objective interfaces and interactive charts using Chart.js and Bootstrap 5. The system allows users to register financial assets, set goals according to their investment profile, and suggests actions to be taken for each asset. Additionally, it provides table views of transactions per asset and graphically displays value fluctuations over time.*

Resumo. *Este trabalho apresenta o desenvolvimento do ADASFINANCE, um sistema de controle de carteira de ativos financeiros focado nos objetivos de investimento do usuário, oferecendo uma alternativa prática e simples às plataformas existentes. A solução é uma aplicação web baseada em LAMP (Linux, Apache, MySQL e PHP) com interfaces minimalistas e objetivas e gráficos interativos usando Chart.js e Bootstrap 5. O sistema permite ao usuário cadastrar ativos financeiros, definir objetivos conforme seu perfil de investimento e sugere ações a serem tomadas para cada ativo. Além disso, oferece visualização em tabela das transações por ativo e exibe graficamente as oscilações dos valores ao longo do tempo.*

1. Introdução

No cenário financeiro contemporâneo, a busca por estratégias de investimento eficientes é essencial devido à complexidade e à constante evolução dos mercados globais. A interconexão entre as economias, a volatilidade dos ativos e as rápidas mudanças nas condições macroeconômicas apresentam desafios adicionais para os investidores. Diversas opções de investimentos, como renda variável, renda fixa, fundos imobiliários (FIIs) e criptoativos, podem compor o portfólio do investidor.

O mercado de ações no Brasil destaca-se como uma oportunidade para investidores externos diversificarem seus portfólios (Nunes et al., 2005). Embora investimentos em bolsa de valores sejam frequentemente associados a riscos, existem teorias que podem ser aprendidas para potencializar resultados (Scottá dos Passos; Pinheiro, 2010). Analisar fatores como indicadores econômicos, relatórios de empresas, notícias geopolíticas e tendências de mercado é crucial no ambiente financeiro atual.

A diversificação é fundamental para construir uma carteira robusta, reduzindo o risco e otimizando o retorno (Mendes; Abreu, 2006). Ferramentas como GorilaAPP, Fundamentei e StatusInvest ajudam na gestão de investimentos, mas possuem limitações em fornecer orientações específicas para o balanceamento da carteira. A era digital trouxe uma explosão de dados e uma acessibilidade sem precedentes às informações financeiras, exigindo análises precisas e personalizadas. Este trabalho propõe o desenvolvimento de um sistema de gestão de carteira de ativos alinhado aos objetivos do investidor, facilitando a tomada de decisões de compra e venda (Bodie et al., 2014).

2. Conceitos importantes

2.1 Carteira de investimentos

Uma carteira de investimentos compreende um conjunto de ativos cujos valores de mercado variam conforme as empresas que a compõem, podendo aumentar ou reduzir o risco da carteira (Kato, 2004, apud Bach et al., 2015). A seleção de carteiras eficientes visa maximizar o retorno esperado, e a Teoria dos

Portfólios de Markowitz (1952) busca gerar máxima eficiência por meio da diversificação, equilibrando risco e retorno (Bach et al., 2015).

Estruturadas para diversificar investimentos e minimizar riscos, as carteiras equilibram a proporção de diferentes tipos de ativos, como ações, títulos e criptomoedas. O objetivo principal é otimizar o retorno ajustado ao risco, permitindo que os investidores alcancem suas metas financeiras de curto, médio e longo prazos. É essencial monitorar e ajustar periodicamente as carteiras conforme as condições de mercado e as necessidades dos investidores.

2.2 Rebalanceamento de carteira

O rebalanceamento de carteira é uma estratégia que consiste em revisar e ajustar a composição dos ativos para manter a alocação inicial, conforme os objetivos de investimento, a expectativa de retorno e a tolerância ao risco do investidor. Este processo corrige os desequilíbrios causados pelas flutuações do mercado, vendendo ativos que se valorizaram e comprando aqueles que se desvalorizaram, mantendo assim a carteira em conformidade com as metas estabelecidas (Stumpf, 2024). Essa prática é fundamental para controlar o risco e maximizar o retorno ao longo do tempo, garantindo que a carteira continue adequada às circunstâncias e objetivos do investidor.

3. Solução proposta

A solução proposta neste trabalho é um sistema de controle de carteira de ativos financeiros focado nos objetivos de investimento do usuário, desenvolvido para monitorar ativos de maneira clara e objetiva. A aplicação foi construída utilizando a stack tecnológica LAMP (Linux, Apache, MySQL e PHP), complementada por interfaces minimalistas e gráficos interativos criados com Chart.js e Bootstrap 5. O sistema permite que os usuários registrem seus ativos financeiros, definam metas conforme seu perfil de investimento e recebam sugestões de ações a serem tomadas para cada ativo. Além disso, a ferramenta oferece visualização em tabelas das transações específicas por ativo e exibe graficamente as flutuações de valores ao longo do tempo. Com essa abordagem, busca-se proporcionar uma alternativa prática e eficiente às plataformas existentes, facilitando a tomada de decisões de investimento.

4. Implementação

A implementação da solução proposta foi basicamente realizada em quatro etapas, que serão aprofundadas a seguir. São elas: modelagem dos dados, o desenvolvimento do backend (camada de aplicação), a integração com uma API de dados sobre ativos financeiros externa e a criação da interface de usuário.

4.1 Modelagem de dados

A modelagem dos dados foi feita utilizando o banco de dados MySQL, um sistema de gerenciamento de banco de dados relacional de código aberto, amplamente utilizado em aplicações web e empresariais. Ele utiliza a linguagem SQL (Structured Query Language) para acessar e gerenciar os dados armazenados, permitindo operações como consultas, inserções, atualizações e exclusões de registros. Foram definidas quatro tabelas para a criação da estrutura da aplicação, sendo elas:

- *TransactionType*, uma tabela informativa criada para fazer comparações utilizando os ids, e não strings de forma textual o tipo da transação, seja de compra ou de venda;
- *Transaction*, que representa as transações que cada usuário exerce de cada ativo relacionado a ele;
- *User*, que representa os usuários do sistema;
- *Asset*, que representa os ativos (ações, fundos imobiliários e criptomoedas) e
- *UserAsset*, que representa uma tabela associativa entre o usuário e o ativo, contendo informações sobre qual o preço médio, a quantidade e a porcentagem objetivo de determinado ativo para determinado usuário. As tabelas podem ser observadas com mais detalhes na figura abaixo.

4.2 Desenvolvimento

O desenvolvimento da ferramenta no que diz respeito ao backend (camada de aplicação) foi modularizada de forma a permitir a manutenção, legibilidade, escalabilidade do sistema, sem a utilização de nenhum framework, de forma que foram criados módulos operacionais específicos para encapsular e isolar algumas ações repetitivas. Os principais módulos são:

- **Módulo de Entidades:** define as classes e estruturas de dados que representam os elementos fundamentais do sistema, como usuários, ativos e transações. As entidades são mapeadas para tabelas do banco de dados e incluem validações e relacionamentos.
- **Módulo de Consulta e Persistência:** responsável por realizar operações de leitura e gravação no banco de dados.
- **Módulo de Controle:** implementa a lógica de negócio do sistema, orquestrando a interação entre os diferentes módulos. Processa as entradas dos usuários, aplica regras de negócio e coordena as operações necessárias para atender às solicitações.
- **Módulo de Roteamento:** gerencia as rotas de acesso às diferentes funcionalidades do sistema. Define os endpoints e mapeia as URLs para os controladores correspondentes, facilitando a navegação e o direcionamento das requisições dos usuários.
- **Módulo de Serviços:** fornece funcionalidades auxiliares e independentes que suportam a operação do sistema, como serviços de utilidades. Estes serviços são reutilizáveis e podem ser invocados por outros módulos conforme necessário.

4.3 Integração com API externa

Neste projeto, foram incorporadas duas APIs externas para consultas de dados. A primeira, a HG Finance, foi utilizada para preencher a tabela Assets, que inclui todos os ativos do Ibovespa, como ações e fundos imobiliários. Devido ao fato de ser uma API paga, foi utilizado apenas o período de teste gratuito de 7 dias, durante o qual era possível realizar chamadas individuais por símbolo de ativo ou chamadas de até cinco símbolos simultaneamente, visando minimizar o número de chamadas.

A segunda API utilizada foi a AlphaVantage, utilizada para obter informações atualizadas sobre os preços dos ativos, ou dos preços dos ativos em determinados períodos, função útil para a montagem dos gráficos que são exibidos aos usuários no detalhamento do ativo. Também foi utilizada a versão gratuita da ferramenta, assim, grande parte das funcionalidades foram limitadas e o número de requisições diárias impediu de serem feitas algumas funcionalidades como a integração

automática, que permitiria o usuário ter sempre o último preço do ativo de forma síncrona, sem precisar acionar o botão “sincronizar” na tela inicial.

4.4 Criação de interface com o usuário

A criação de uma interface com o usuário é crucial para garantir uma experiência intuitiva e eficaz na gestão de investimentos. Nesse sentido, foi desenvolvida uma interface que não apenas seja visualmente atraente, mas também fácil de navegar, com elementos de design minimalista que orientem os usuários de forma clara e concisa. Foram incorporados feedbacks visuais interativos, como gráficos que melhoram a compreensão e a tomada de decisões dos investidores.

Foram criadas basicamente apenas três telas, sendo uma de login, uma de cadastro de usuário e a outra a tela principal da aplicação. Dentro da tela principal ainda foram criados alguns componentes de forma a melhorar a usabilidade e fornecer ferramentas gráficas para o usuário contendo informações importantes para análise e acompanhamento dos ativos, por exemplo.

Também foram criadas funcionalidades dentro da tela inicial de inclusão, deleção, compra e venda de ativo, tabela de transações, cálculos de porcentagem objetivo e balanço total.

5. Conclusão e trabalhos futuros

A ferramenta de gestão de investimentos foi testada por cinco participantes, com resultados satisfatórios comparados à literatura existente. Todos os objetivos específicos, desde a análise de trabalhos correlatos até a avaliação final do sistema, foram alcançados, indicando a adequação da metodologia utilizada e dos critérios de avaliação. A avaliação positiva sugere que o sistema é eficaz e cumpre suas funções conforme o esperado, corroborando sua eficácia e viabilidade para implementação em contextos similares.

No entanto, a ferramenta atual apresenta limitações significativas que dificultam uma análise detalhada e a tomada de decisões eficientes. Os gráficos oferecidos são superficiais, impossibilitando uma avaliação precisa da evolução dos ativos a curto prazo. A ausência de funcionalidades avançadas, como ordens de compra ou venda baseadas em valores pré-definidos, e a incapacidade de comparar a rentabilidade da carteira com benchmarks como o Ibovespa limitam a eficácia da

gestão da carteira. Além disso, a falta de informações sobre a data da última sincronização e a rentabilidade total da carteira prejudica o acompanhamento dos investimentos.

Apesar dessas limitações, a ferramenta mostrou ser eficiente e útil, alcançando uma pontuação de 89,5 no SUS e boas avaliações em questionários específicos. Seus pontos fortes incluem uma usabilidade simples e intuitiva, adequada para usuários com pouca experiência, e a função de balanço total da carteira, que não é comum em plataformas concorrentes. A possibilidade de definir objetivos individuais para cada ativo também foi destacada positivamente. Esses aspectos positivos indicam uma avaliação geral favorável, mas ressaltam a necessidade de melhorias nas áreas identificadas para aprimorar a experiência do usuário e a eficácia do produto.

Para trabalhos futuros, será essencial focar no aprimoramento das ferramentas gráficas e na interação com o usuário, além de implementar funcionalidades avançadas como ordens de compra e venda baseadas em valores pré-definidos. A integração com plataformas como o CEI e a melhoria na sincronização de dados também serão prioridades, visando fornecer informações mais precisas e em tempo real. Adicionalmente, a inclusão de benchmarks permitirá uma avaliação mais detalhada do desempenho da carteira. Essas melhorias potencializarão a usabilidade e a eficácia do sistema, proporcionando uma experiência mais intuitiva e satisfatória para os usuários finais, e possibilitando que a ferramenta alcance um nível superior de excelência.

Referências

AlphaVantage. Disponível em: <https://www.alphavantage.co/>.

BACH, T. M.; SILVA, W. V. da; KUDLAWICZ, C.; MARQUES, S. Eficiência das Companhias Abertas e o Risco versus Retorno das Carteiras de Ações a partir do Modelo de Markowitz. **Revista Evidenciação Contábil & Finanças**, [S. l.], v. 3, n. 1, p. 34–53, 2015. Disponível em: <https://periodicos.ufpb.br/index.php/recfin/article/view/21312>. Acesso em: 12 jun. 2024.

BODIE, Zvi; KANE, Alex; MARCUS, Alan J. **Fundamentos de investimentos**. Tradução de Beth Honorato. 9ª ed. Porto Alegre, AMGH Editora, 2014.

HG Finance. Disponível em: <https://hgbrasil.com/status/finance>.

MARKOWITZ, H. M. Seleção de Carteiras. **The Journal of Finance**, v. 7, n. 1, p. 77-91, 1952. doi:10.2307/2975974.

MENDES, Victor; ABREU, Margarida. **Cultura financeira dos investidores e diversificação das carteiras**. Instituto Superior de Economia e Gestão - DE Working papers 11/2006/DE/CISEP. Lisboa: ISEG – Departamento de Economia, 2006. Disponível em: <http://hdl.handle.net/10400.5/863>. Acesso em: 03 dez. 2023.

NUNES, Maurício S.; COSTA JUNIOR, Newton C. A. da; MEURER, Roberto. A relação entre o mercado de ações e as variáveis macroeconômicas: uma análise econométrica para o Brasil. **Revista Brasileira de Economia**, Rio de Janeiro, v. 59, n. 4, p. 585-607, dez. 2005. Disponível em: <http://dx.doi.org/10.1590/s0034-71402005000400004>. Acesso em: 03 dez. 2023.

SCOTTÁ DOS PASSOS, Vinicius C.; PINHEIRO, Juliano L. Estratégias De Investimento Em Bolsa De Valores: Uma Pesquisa Exploratória Da Visão Fundamentalista De Benjamin Graham. **Revista Gestão & Tecnologia**. 2010. Disponível em: <https://doi.org/10.20397/2177-6652/2009.v9i1.233>. Acesso em 07 jul. 2024.

STUMPF, K. **Rebalanceamento de carteira: o que é, vantagens e como fazer**. 2024. Disponível em: <https://www.topinvest.com.br/rebalanceamento-de-carteira/#:~:text=O%20rebalanceamento%20de%20carteira%20%C3%A9,estimada%20e%20minimiza%C3%A7%C3%A3o%20de%20riscos>. Acesso em: 07 jul. 2024.