Ícone

Descrição gerada automaticamente

**FACULDADE DE TECNOLOGIA DE COTIA**

**TECNOLOGIA EM DESENVOLVIMENTO DE SOFTWARE MULTIPLATAFORMA**

**THARSEO**

**AUTOMATIZADOR DE ESTRATÉGIAS DE NEGOCIAÇÃO EM CRIPTOMOEDAS**

**ALBERTO RIBEIRO DE OLIVEIRA, ALESSANDRA SANCHES, CAIO RIBEIRO, FLAVIO DOMINGUES VIEIRA, RICHARD GUEDES RIBEIRO DA SILVA**

**Cotia – SP**

**Março/2025**

**FACULDADE DE TECNOLOGIA DE COTIA**

**THARSEO**

**AUTOMATIZADOR DE ESTRATÉGIAS DE NEGOCIAÇÃO EM CRIPTOMOEDAS**

**ALBERTO RIBEIRO DE OLIVEIRA, ALESSANDRA SANCHES, CAIO RIBEIRO, FLAVIO DOMINGUES VIEIRA, RICHARD GUEDES RIBEIRO DA SILVA**

Projeto Integrador entregue à disciplina de Laboratório de desenvolvimento Multiplataforma para o Trabalho de Graduação do curso de Tecnologia em Desenvolvimento de Software Multiplataforma da Faculdade de Tecnologia de Cotia.

**Professor (a) Orientador (a):Jefferson Pereira Macena**

**Cotia – SP**

**Março/2025**

# Projeto Integrador

**Sumário**

[Projeto Integrador 3](#__RefHeading___Toc12388_3575018764)

[1. Cronograma de Trabalho 5](#__RefHeading___Toc12390_3575018764)

[1.1. Brainstorm 11](#__RefHeading___Toc12392_3575018764)

[1.2. Gestão do projeto 12](#__RefHeading___Toc12394_3575018764)

[1.2.1. Trello (Kanban) 12](#__RefHeading___Toc12396_3575018764)

[1.2.2. Scrum 13](#__RefHeading___Toc12398_3575018764)

[1.2.3. Sprint (Reuniões semanais) 13](#__RefHeading___Toc12400_3575018764)

[2. Introdução 15](#__RefHeading___Toc12402_3575018764)

[2.1. Elicitação de requisitos 15](#__RefHeading___Toc12404_3575018764)

[3. Definição Conceitual da Solução 16](#__RefHeading___Toc12406_3575018764)

[3.1. Fundamentação Teórica 20](#__RefHeading___Toc12408_3575018764)

[3.2. Oque são criptomoedas? 21](#__RefHeading___Toc12410_3575018764)

[3.2.1. Como ter acesso ao mercado de criptoativos? 21](#__RefHeading___Toc12412_3575018764)

[3.2.2. Como funciona o mercado hoje? 22](#__RefHeading___Toc12414_3575018764)

[3.2.3. Qual o impacto das criptomoedas e como ela movimenta o mercado atual? 22](#__RefHeading___Toc12416_3575018764)

[3.2.4. Oque é a Binance? 24](#__RefHeading___Toc12418_3575018764)

[2.5.4. Regras de Negócios 24](#__RefHeading___Toc12420_3575018764)

[4. Diagrama de Casos de Uso 26](#__RefHeading___Toc12422_3575018764)

[5. Requisitos Funcionais 26](#__RefHeading___Toc12424_3575018764)

[5.1 Requisitos do Usuário 29](#__RefHeading___Toc12426_3575018764)

[6. Requisitos Não-funcionais 30](#__RefHeading___Toc12428_3575018764)

[7. Protótipo Navegável do Sistema 30](#__RefHeading___Toc12430_3575018764)

[8. Diagrama de Classes de Domínio 36](#__RefHeading___Toc12432_3575018764)

[9. Arquitetura da Solução 37](#__RefHeading___Toc12434_3575018764)

[10. Padrão Arquitetural 39](#__RefHeading___Toc12436_3575018764)

[11. C4 model - Diagrama de Contexto 40](#__RefHeading___Toc12438_3575018764)

[12. Frameworks de Trabalho 41](#__RefHeading___Toc12440_3575018764)

[12.1. Typescript 41](#__RefHeading___Toc12442_3575018764)

[12.2. Jest 41](#__RefHeading___Toc12444_3575018764)

[13.3. API Binance 42](#__RefHeading___Toc12446_3575018764)

[13.4. PostgresSQL 43](#__RefHeading___Toc12448_3575018764)

[13.5. Websocket 44](#__RefHeading___Toc12450_3575018764)

[13.6. Git 44](#__RefHeading___Toc12452_3575018764)

[13.7. Docker 45](#__RefHeading___Toc12454_3575018764)

[13.8. SonarCloud 45](#__RefHeading___Toc12456_3575018764)

[13.9. Azure 47](#__RefHeading___Toc12458_3575018764)

[13.10. Clarity 49](#__RefHeading___Toc12460_3575018764)

[13. Estrutura Base do Front End 54](#__RefHeading___Toc12462_3575018764)

[55](#__RefHeading___Toc12464_3575018764)

[14. Modelo Relacional ou Projeto de Banco de Dados NoSQL 56](#__RefHeading___Toc12466_3575018764)

[14.1 Modelo do Banco relacional 58](#__RefHeading___Toc12468_3575018764)

[15. Plano de Testes 59](#__RefHeading___Toc12470_3575018764)

[Plano de Testes Tharseo Backend v2 59](#__RefHeading___Toc12472_3575018764)

[Plano de Testes Tharseo Frontend v2 89](#__RefHeading___Toc12474_3575018764)

[16. Aspectos de segurança do Projeto 96](#__RefHeading___Toc12476_3575018764)

[17. Apropriação de Horas no Projeto 100](#__RefHeading___Toc12478_3575018764)

[18. Código da Aplicação 101](#__RefHeading___Toc12480_3575018764)

[19. Avaliação Retrospectiva 102](#__RefHeading___Toc12482_3575018764)

[20. Objetivos Estimados 102](#__RefHeading___Toc12484_3575018764)

[19.2 Objetivos Alcançados 103](#__RefHeading___Toc12486_3575018764)

[21. Lições aprendidas 103](#__RefHeading___Toc12488_3575018764)

[22. Referências 104](#__RefHeading___Toc12490_3575018764)

## Cronograma de Trabalho

|  |  |  |  |
| --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** |
| **De** | **Até** |
| 23/02/2025 | 28/02/2025 | 1. Desenvolvimento de views do fluxo de registro de usuário;  Desenvolvimento do fluxo e rotas de autenticacao no backend;  Desenvolvimento das views do fluxo de autenticação;  Implementação de token kwt e criação de payload de register e auth;  Desenvolvimento de Sidebar e alimentando a aplicação como componente;  Desenvolvimento de Componente para trazer os principais assets e quanto eles representam;  Configuração do zustand para gerenciamento de estado no frontend; | Desenvolvimento da página de login, cadastro e parte da Navegação da aplicação (frontend) funcional e início do processo de atualização do Backend entregando as partes desenvolvidas funcionais e documentadas, desenvolvimento de parte de autenticação e sistemas de segurança para autenticação, entregando tudo oque foi previsto e proposto de forma funcional. |
| 01/03/2025 | 06/03/2025 | 2. Implementação de Pipeline do Frontend (ReactTSX) no Firebase;  Implementação de seed com usuarios dos membros da equipe;  Implementado JWT e AuthGuard em rotas que necessitam autenticação;  Desenvolvimento de Interface de Requests public e private Binance;  Desenvolvimento de integração com API da Binance;  Criação do seed para registrar os assets iniciais e carteiras iniciais dos cadastros dos membros;  Desenvolvimento de view para listar ativos do usuário (dados mock);  DB: Alteração do diagrama conforme regra de negócio; | Continuidade em detalhes do front e backend, implementando rotas, funcionalidades relacionadas e login e cadastro, iniciando o processo de registro de ativos de ações e operações na aplicação, da conexão com a fonte das transações no caso a binance através das API’s que estavam previstas para suporte e funcionamento do produto. Com isso todas as etapas previstas e executadas foram entregues funcionais. |
| 07/03/2025 | 12/03/2025 | 3. Desenvolvimento de lógica, métodos e rotas para Wallet;  Desenvolvimento das rotas, dtos, services e repositories, inclusive documentadas no postman das ordens;  Desenvolvimento do Signature da Binance;  Desenvolvimento de View para configurar o formato da estratégia grid do usuário;  Implementação de rotas de páginas acessíveis e feedback de páginas em manutenção;  Desenvolvimento de View para visualizar as automações de um usuário;  Adicionado drawer para cadastrar nova automação; | Continuidade do desenvolvimento backend com verificação e desenvolvimento da logica para os métodos e rotas envolvidos na Wallet, páginas e feedbacks, desenvolvimento de pontos de segurança e validação com a binance, retomada do desenvolvimento do processo de automações de compra/venda, desenvolvimento do Frontend em Views importantes para a apliacação. Tudo oque foi previsto e proposto até a data, foi entregue funcional. |
| 13/03/2025 | 18/03/2025 | 4. Desenvolvida view para visualizar operações em aberto;  Desenvolvido readme do repositório frontend com instruções, destalhes e tecnologias;  Desenvolvimento de view para visualizar operações encerradas ou canceladas;  Refatoração de Header da page Estratégias para padronizar conforme outras páginas;  Implementação de Pipeline backend com novos requisitos para o NestJs e nova Vm;  Desenvolvimento de componente Chart para futura integração com dados de gráfico;  Desenvolvimento de módulo Asset com Controller e Service;  Documentação do Módulo Asset no Postman com rotas, exemplos de body, exemplos de retorno e descritivos;  Inclusão de Todos os Descritivos dos métodos desenvolvidos até aqui; | Continuidade focada principalmente no Frontend, com o desenvolvimento de Views relacionadas as operações dentro da aplicação, sendo elas operações em aberto, encerradas ou canceladas, atualizando também o componente Header da página para manter o mesmo padrão igual das demais páginas, foi desenvolvido também a documentação readme com as instruções de execução, detalhes e as tecnologias utilizadas no frontend, foi iniciado o desenvolvimento do componente Chart pois precisaremos dele para integração e leitura dos gráficos dos ativos, em relação ao backend foi implementado o Pipeline, o desenvolvimento dos assets e controllers faltantes, além da documentação do que foi realizado até aqui no Postman. Tudo oque foi proposto até essa etapa foi entregue funcional, sem a necessidade de ajustes e/ou correções. |
| 19/03/2025 | 24/03/2025 | 5. Desenvolvimento das rotas, métodos e logicas do backend da Entidade de Estratégias;  Resolução de método para deslogar devido a ficar com cadastro preso;  Desenvolvimento do store para gerenciar estado das estratégias no front e métodos para receber dados do back;  Desenvolvimento de form em Drawer para criar uma nova estratégia;  Desenvolvimento das rotas de tradeflow (automações);  Implementação de rota para receber todos os assets do backend;  Implementação da api e service dentro do Trade para comunicação com Backend; | Seguiu-se com o desenvolvimento Front e Back com busca do atendimento das necessidades das interações entre Back e frontend, resolvendo situações que nos geraria problemas futuros, foi desenvolvido também a parte da automação de compra e venda e funcionalidades para definição da estratégia do usuário. Tudo oque foi proposto até aqui foi entregue de forma funcional e sem necessidades de atualização e reparo. |
| 25/03/2025 | 31/03/2025 | 6. Inclusão de imagens dos logotipos das principais criptos;  Implementação de store para gerenciar estado das ordens e implementação na view para visualização de ordens em aberto;  Configuração do prometheus e exposição da rota métricas;  Criação da estrutura do Dockerfile e prometheus.yml;  Implementação do Repositório e Hospedagem do Prometheus;  Criar Fluxo protótipo mobile, web, gerar vídeo, hospedar e acrescentar o link a documentação;  Integração de Métricas e Prometheus com Grafana. Criação do Dashboard; | Atualização do frontend com foco em pontos intuitivos para o usuário, configurações e implementações de requisitos da matéria de computação em Nuvem 2, e atualizações solicitadas no Template da documentação para entrega. Tudo o que foi proposto até aqui foi entregue de forma funcional, prováveis reparos na documentação serão indicados pelo professor. |
| 01/04/2025 | 06/04/2025 | 7. Implementação do Drawer com funcionalidade para adicionar ordem manual;  Implementação do Fluxo de criar ordem manual dentro do backend integrado com a binance;  Retirada dos dados mocks da histórico de trades e integração com api trazendo os dados do server alimentando a view;  Ajuste de dashboard para trazer o ícone dos Aseets corretamente;  Desenvolver o front para que os ativos renderizem o gráfico conforme symbol e timechart;  Implementação de método para armazenar dados de preço para servir a aplicação; | Realizados ajustes no front e criação de novas funcionalidades na aplicação além do desenvolvimento de partes do front que ainda não estavam feitas para renderizar o gráfico dos ativos e método para armazenamento de preços desenvolvido no Backend. Tudo oque foi proposto até aqui foi entregue de forma funcional e sem necessidades de atualização e reparo. |
| 07/04/2025 | 12/04/2025 | 8. Engine inicial para montar lógica de ordens de Compra na corretora e ordens de venda em segundo plano com status pendente;  Atualização e correção da documentação segundo pontos apontados pelo professor e atualizações feitas no trabalho;  Implementação de rotina para verificar execução de ordens de compra e cadastro das ordens de venda a LIMIT;  Desenvolver o Botão e implementar rota do Frontend para cancelamento de todas as ordens em aberto;  Implementação de Fluxo no Backend e exposição de rota para cancelar todas as ordens em aberto;  Implementação do método para calcular e registrar profit e performance em ordens e estratégias; | Desenvolvimento Front e Back de novas funcionalidades e necessidades observadas durante o desenvolvimento, atualização e correção de documentação segundo orientação do professor no 1° Feedback e continuidade no desenvolvimento. Tudo o que foi proposto até aqui foi entregue de forma funcional e prováveis atualizações futuras serão necessárias na documentação, buscando alinhamento com a orientação do professor. |
| 13/04/2025 | 18/04/2025 | 7. Implementação testes unitários em todos métodos das classes user.service e user.controller;  Implementação de pipeline com exigência de testes unitários;  Implementação de estrutura e libs para testes unitários;  Desenvolvimento de modal para quando desativar a automação, pergunte ao usuário se deseja cancelar as ordens pendentes; | Concluído, seguindo roteiro e atualização da documentação e da aplicação |
| 19/04/2025 | 24/04/2025 | 8. Criar Visão Perfil Usuário;  Dar continuidade na Atualizaçãpo documentação no template indicado pelo professor fazendo correções e ajustes apontados na revisão do documento;  A/C Flavio: Implementar nova VM com maior capacidade para back-end/database como plano B;  Estudar e se possível Implementar Ferramentas de análise de dados e relatórios personalizados; | Concluído, seguindo com o desenvolvimento da aplicação e atualização da documentação. |
| 25/04/2025 | 30/04/2025 | 9. Dar continuidade na Atualizaçãpo documentação no template indicado pelo professor fazendo correções e ajustes apontados na revisão do documento;  Estudar e se possível gerar a Integração com outras plataformas de negociação;  Estudar método e se possível implementar Notificações em tempo real sobre eventos importantes do mercado.;  Desenvolver Aplicação de Testes unitários no backend; | Pendente atualização conforme desenvolvimento da etapa na data prevista; |
| 01/05/2025 | 06/05/2025 | 10. Dar continuidade na Atualização da documentação no template indicado pelo professor fazendo correções e ajustes apontados na revisão do documento;  Desenvolver Home LP;  Criar fluxo e método para deslogar no frontend;  Criar fluxo de notificações no frontend;  Criar fluxo de notificações no Backend;  Criar fluxo para atualizar dados do Usuario; | Processo concluído seguindo com o desenvolvimento da aplicação e atualização da Documentação. |
| 07/05/2025 | 30/05/2025 | 11. Definir e configurar tipo de servidor e suas particularidades;  Documentar configurações do servidor da aplicação | Concluído, será dado inicio no documento para apresentação. |
| 01/06/2025 | 28/01/2025 | 12. Verificações dos funcionais já implementados;  Checagem dos testes realizados;  Verificações dos servidores OK;  Elaboração dos slides para apresentação;  Apresentação final para professores | Em processo de finalização, boa parte das etapas descritas já prontas, porém a finalização será concretizada no dia da apresentação (28/06). |

### 1.1. Brainstorm

“O Brainstorming é uma dinâmica de grupo que é usada em várias empresas como umSa técnica para resolver problemas específicos, para desenvolver novas ideias ou projetos, para juntar informação e para estimular o pensamento criativo.” (Significados,2022).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BRAINSTORMING | | | | |
| 19/09/2023 à 23/09/2023 | Entre os intervalos das aulas | | Presencial | |
| Participantes |  | | | |
| Alberto Ribeiro | | | | |
| Alessandra Sanches | | | | |
| Caio Ribeiro | | | | |
| Flávio Domingues | | | | |
| Richard Guedes | | | | |
| Assunto: Elicitação de Requisitos Funcionais, Não-Funcionais, Regras de Negócios | | | |  |
| O que foi feito: | | Em pequenas reuniões, fizemos o levantamento dos requisitos funcionais, não funcionais e regras de negócio baseadas na necessidade da demanda estudada. | |  |
| Como foi feito: | | Foi feito de modo a pensar em tornar atrativa a aplicação para a sociedade, independente da idade ou classe social. | |  |
| Porque foi feito: | | Devido ao potencial atual e futuro de lucratividade com as crypto moedas, sendo um assunto muito tratado entre diversos públicos e mídias como tenência futura com a digitalização das moedas e bens. | |  |
| Onde foi feito: | | Polo da Fatec em Cotia/SP. | |  |
| Quem fará: | | Os alunos integrantes do grupo. | |  |

### 1.2. Gestão do projeto

#### 1.2.1. Trello (Kanban)

O Trello é uma plataforma de gerenciamento de projetos que utiliza um sistema de quadros, listas e cartões para organizar tarefas e colaborar em equipes. Lançado em 2011, o Trello rapidamente se tornou uma ferramenta popular devido à sua simplicidade e eficácia. Seu design intuitivo baseado em quadros permite que os usuários visualizem e controlem o progresso de suas tarefas de forma clara e interativa. Com a capacidade de personalizar os quadros, adicionar listas e atribuir cartões a membros da equipe, o Trello se destaca como uma solução versátil para gerenciamento de projetos em diversos contextos.

O Kanban é um método visual de gerenciamento de processos originado no Japão e popularizado pela Toyota nas décadas de 1940 e 1950. Essa abordagem se baseia em cartões (kanbans) que representam tarefas ou itens em um quadro visual. O objetivo principal do Kanban é otimizar a eficiência, minimizar o desperdício e melhorar a visibilidade do processo de produção ou gestão de projetos. As etapas do fluxo de trabalho são representadas em colunas no quadro, permitindo que os membros da equipe visualizem facilmente o status de cada tarefa. O Kanban é conhecido por sua flexibilidade e adaptabilidade, tornando-se uma ferramenta valiosa em ambientes ágeis e colaborativos.

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

#### 1.2.2. Scrum

O Scrum é um dos frameworks mais populares no mundo ágil, oferecendo uma abordagem flexível e colaborativa para o desenvolvimento de projetos complexos. Desenvolvido inicialmente para a gestão de projetos de software, o Scrum tem se expandido para diversas áreas, tornando-se uma metodologia amplamente adotada em diversos setores.

Central para o Scrum está o conceito de iteração e colaboração contínua. Os projetos são divididos em ciclos de desenvolvimento chamados "sprints", geralmente com duração de duas a quatro semanas. Cada sprint envolve um planejamento detalhado, seguido pela execução e revisão, permitindo adaptações rápidas às mudanças e feedback contínuo.

O Scrum inclui papéis claramente definidos, como o Scrum Master, responsável por garantir a aplicação correta do Scrum, o Product Owner, encarregado de definir e priorizar os requisitos do projeto, e a equipe de desenvolvimento, responsável por entregar as tarefas definidas durante o sprint.

As cerimônias do Scrum, como as reuniões diárias ("daily stand-ups") e as revisões de sprint, promovem a comunicação eficaz e a transparência, essenciais para o sucesso do projeto. Ao adotar princípios como a entrega incremental de valor, o Scrum oferece uma estrutura adaptável e eficiente para equipes que buscam maior flexibilidade e capacidade de resposta em um ambiente em constante evolução.

Definimos como Product Owner o integrante Richard e o Scrum Master o integrante Alessandra, os quais direcionaram o andamento do trabalho e dirigiam as reuniões semanais verificando em que ponto o trabalho estava e as tarefas necessárias a serem feitas para o bom andamento do trabalho.

#### 1.2.3. Sprint (Reuniões semanais)

O termo "sprint" é uma peça fundamental na metodologia ágil, especialmente no contexto do Scrum, um framework popular para o desenvolvimento de software e gerenciamento de projetos. Uma sprint é um período definido, tipicamente curto, que varia de uma a quatro semanas, durante o qual uma equipe se compromete a completar um conjunto específico de tarefas ou entregáveis.

Essas sprints são marcadas por uma intensa colaboração e foco, visando a criação de incrementos tangíveis e funcionais do produto ao final de cada ciclo. Durante a sprint, a equipe realiza reuniões diárias conhecidas como "daily stand-ups" para discutir o progresso, identificar obstáculos e ajustar estratégias conforme necessário.

O conceito de sprint proporciona uma abordagem iterativa e incremental ao desenvolvimento de projetos, permitindo que as equipes respondam rapidamente às mudanças nos requisitos ou nas prioridades do cliente. Essa prática enraizada na metodologia ágil promove a transparência, a adaptação contínua e a entrega de valor de forma consistente ao longo do ciclo de vida do projeto.

Nossas Sprints tinham como objetivo monitorar o andamento do projeto distribuindo funções e tarefas para os mais adiantados, prestando suporte aos que estavam com dificuldades, implementando as partes prontas e realizando testes, e claro documentando cada parte.

## Introdução

O Tharseo, nossa proposta de aplicação para automatizar estratégias de negociação em criptomoedas, foi criado com o objetivo primordial de automatizar negociações no mercado de criptomoedas, oferecendo aos usuários uma abordagem estratégica e eficiente permitindo que o usuário possa seguir sua rotina enquanto investe no mercado financeiro no segmento de criptomoedas.

A crescente popularidade e volatilidade do mercado de criptomoedas têm suscitado a necessidade de ferramentas inovadoras que possam auxiliar os traders na otimização de suas operações comerciais, maximizando os ganhos potenciais e gerenciando os riscos de forma inteligente. Nesse contexto, o Tharseo surge como uma solução poderosa, proporcionando uma plataforma robusta e resiliente para a automação de negociações.

Através dessa, buscamos fornecer uma visão abrangente das características, funcionalidades e processos envolvidos na utilização do Tharseo. Serão abordados aspectos como a necessidade do usuário. Buscamos dar ênfase a experiência única do usuário, explorando através de ferramentas de UX e UI, o desenvolvimento em que possamos trazer uma melhor usabilidade.

Adicionalmente, abordaremos a problemática que motivou a criação do aplicativo, destacando os desafios enfrentados pelos traders no mercado de criptomoedas e a necessidade de soluções inovadoras para enfrentá-los. Através das personas e pesquisas de campo, mostraremos os resultados alcançados e a definição de estratégias, funcionalidades e prototipação definidas através de pesquisas de campo.

### 2.1. Elicitação de requisitos

“Elicitação de requisitos é uma fase do projeto onde são extraídas informações do cliente sobre o que ele deseja que seja construído. É a fase em que o profissional de TI entende a necessidade do cliente e o orienta. É o momento de conversa com o usuário, de sentimento sobre o que este espera que seja entregue a ele.”

(DEVMEDIA, 2022).

Adentramos em uma das fases fundamentais do projeto, explorando a importância significativa e a facilitação do seu desenvolvimento, juntamente com o potencial de investimento bem-sucedido, através de uma elicitação de requisitos devidamente estruturada.

A dificuldade inicial identificada reside na adesão e motivação dos usuários em aderirem ao Tharseo seria a incerteza e insegurança a princípio de confiarem o seu dinheiro em um sistema totalmente novo, será preciso gerar pontos de identificação do projeto com o usuário e a sociedade de modo a gerarem confiança e segurança a todos que aderirem a aplicação.

Para superar essa barreira, implementamos a técnica de Brainstorming. Essa abordagem permitiu a coleta de uma variedade de opções provenientes dos diversos membros do grupo, sem avaliações críticas nesta fase inicial. O objetivo foi obter um conjunto abrangente de possíveis requisitos para subsidiar as etapas subsequentes

## Definição Conceitual da Solução

Segundo a ANBIMA(Associação Brasileira das Entidades dos Mercados Financeiro e de Capitais), no ano de 2023/2024 os investimentos dos brasileiros (pessoas físicas), somaram sucessivamente no mês de Setembro 5,5 trilhões, que é equivalente a um aumento de 9,7% em relação aos anos anteriores, a observação feita engloba os segmentos de varejos tradicionais, varejo de alta renda e private.

Foi observado que mais da metade dos investimentos dos brasileiros estavam aplicados em títulos e valores mobiliários, além de CDB's, e fundos de renda fixa.

Como podemos observar o público brasileiro está em crescente busca por investimentos e formas alternativas de renda além das atuais usuais, um dos mercados com maior liquidez e potencial de rendimentos que temos hoje são as crypto moedas, estas que tem valor variável, porém não gera a mesma insegurança que ativos e ações que podem despencar com mal planejamento das pessoas que dirigem a empresa ou por dividas que vem à tona e pegam a todos desprevenidos, Olhando para o atual potencial e para a crescente busca por investimentos dos brasileiros desenvolvemos a idéia do Tharseo de modo a facilitar a vida dos que não tem o tempo e nem o Knowhow no mercado de compra e venda de crypto moedas, onde a aplicação faz a compra e venda de forma automatizada a partir de um pré-set, e realiza a venda a partir do mínimo lucro também setado no pré set automatizando o processo e otimizando o tempo dos seus usuários.

Público-alvo:

População em geral, grandes investidores, leigos, jovens investidores, investidores experientes etc.

Problema:

Conforme matéria ANBIMA e Datafolha:

**“A 7ª edição do Raio X do Investidor Brasileiro, pesquisa que realizamos em parceria com o Datafolha, mostra que a caderneta de poupança mantém o posto de investimento mais conhecido e mais utilizado do país. Um quarto da população afirma ter recursos aplicados no produto. Os títulos privados e as moedas digitais, entretanto, vêm ganhando espaço e cresceram na preferência das pessoas pelo segundo ano consecutivo: já fazem parte das carteiras de 5% e de 4% dos brasileiros, respectivamente.**

**“Apesar de a poupança ainda ser a opção mais lembrada e acessada, vemos ao longo dos anos uma proporção cada vez maior de pessoas mencionando e investindo em outros produtos. É, inclusive, uma tendência maior entre a população mais jovem. Muitos fatores têm contribuído para isso, como a procura de investidores por aplicações mais rentáveis em momentos de queda de juros; a sofisticação do nosso mercado financeiro; e o fenômeno dos influenciadores digitais, que popularizaram a conversa sobre investimentos nas redes sociais”, afirma Marcelo Billi, nosso superintendente de Sustentabilidade, Inovação e Educação.”**

Como podemos observar através da matéria, a poupança é o investimento que tem maior índice de investimento até porque é o mais conhecido, estudos também apontam que a maior porcentagem dos que investem em crypto são os jovens que optam por investir em plataformas digitais, além de que a categoria de pessoas que mais investem são os de categoria A/B, tal índice tem relação com a falta de conhecimento dos mais velhos, ou da classe inferior a A/B, além da dificuldade de interação com os sistemas digitais. Abaixo relatórios onde tais observações e conclusões foram possíveis:

Linha do tempo

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente

Diagrama

Descrição gerada automaticamente

Olhando para o cenário apresentado na pesquisa desenvolvemos o Tharseo, na busca de gerar acessibilidade ao investimento para diversas classes e idades, gerando o conhecimento a moedas digitais para investimentos.

### 3.1. Fundamentação Teórica

O “inverno crypto” pode até assustar os que acham que a criptoeconomia se resume a criptomoedas, mas nos bastidores há um mercado trilionário de DeFi crescendo com apoio das tecnologias de blockchain, que pode representar 10% do PIB global em 2030 - US$ 16,1 trilhões, para sermos mais precisos - mesmo com as diferenças regulatórias globais (ou a ausência delas) e a redução de investimentos de risco nessas iniciativas.

Esse pode ser um bom resumo para o estudo recém-publicado pelo Boston Consulting Group (BCG), em parceria com a exchange ADDX, de Singapura, que aprofunda o entendimento das implicações da tokenização de ativos on-chain (gravadas na blockchain) para o crescimento das Finanças Descentralizadas, sua diferenciação do fracionamento de ativos na economia tradicional, as oportunidades abertas para a tokenização de ativos “ilíquidos” e o que é preciso fazer para alavancar seu crescimento.

O relatório - “Relevance of on-chain asset tokenization in ‘crypto winter’” - é assinado pelo diretor administrativo do BCG, Sumit Kumar, o cofundador do ADDX, Darius Liu, e pelos consultores Rajaram Suresh, Bernhard Kronfellner e Aaditya Kaul. Segundo os autores, “uma grande parte da riqueza do mundo hoje está bloqueada em ativos ilíquidos”. A tokenização (transformação em ativos digitais fracionáveis) pode desbloquear esses ativos, contornando as barreiras tradicionais de negociação, gerando alternativas de comercialização fora das práticas tradicionais, baixando o ticket de entrada e abrindo espaço para novos investidores e modalidades de negócio.

Em 2021, a tokenização de ativos representou US$ 2,3 bilhões, e deve chegar a US$ 5,6 bilhões em 2026, segundo o estudo. Ao incluir bens ilíquidos, daria saltos trilionários nas DeFi (veja gráfico acima). Na lista de ativos ilíquidos, o relatório menciona imóveis, recursos naturais, terrenos, commodities, infra-estrutura pública como minas e portos, arte, carros antigos e bebidas raras, que se somam a ações pré-IPO, dívidas privadas, receitas de pequenas e médias empresas, fundos privados e títulos de atacado, entre outros.

As razões para a falta de liquidez desses ativos incluem ticket médio alto para a entrada de investidores em massa, acesso restrito a grupos de elite (no caso de obras de arte e carros antigos), obstáculos regulatórios, entre outros, que dificultam a sua compra ou negociação.

O fracionamento de ativos é uma prática adotada no mercado tradicional, principalmente nos bens patrimoniais e imobiliários - por exemplo, fundos de investimento imobiliário (FIIs) e ETFs. Estão presentes nos mercados públicos, eficientes e com alta tecnologia, mas não nos mercados privados, muitas vezes manuais, lentos, e com altos custos gerais, aponta o relatório.

A tokenização de ativos on-chain nas DeFi (gráfico acima) contorna as barreiras para os ativos ilíquidos “ajudando a reimaginar o processo de encontrar e combinar investidores com oportunidades de investimento”, aponta o estudo. O “minting” (emissão) de tokens representando frações digitais de ativos físicos gravadas em um livro razão distribuído (DLT ou blockchain) oferece garantia de propriedade transparente e imutável.

Baseado no potencial da tendência de digitalização, e o aumento a cada dia da aderência da população em investimentos e carteiras digitais de crypto que o projeto Tharseo ganhou força e engajamento, pois o seu potencial de aderência com o futuro dos investimentos é grande.

### 3.2. Oque são criptomoedas?

Criptomoedas são moedas digitais que usam a criptografia para validar as transações financeiras, portanto é um tipo de dinheiro que existe somente no âmbito virtual. Essa característica também faz com que esses ativos sejam descentralizados. Ou seja, não existe regulamentação por nenhum órgão ou país específico.

#### 3.2.1. Como ter acesso ao mercado de criptoativos?

A maneira mais fácil e segura para iniciantes de se negociar criptomoedas é através de exchanges, que funcionam como corretoras de criptoativos, facilitando a compra e a venda de moedas digitais.

Dependendo da corretora, você pode acessar cotações, variações de preço e negociar uma grande variedade de moedas.

#### 3.2.2. Como funciona o mercado hoje?

A compra e venda de criptomoedas ocorre por meio das exchanges, que são corretoras de criptoativos que oferecem um home broker para realizar as operações a qualquer momento.

O preço dos ativos é determinado pela oferta e demanda, mas os investidores também podem adquirir cotas de fundos de criptoativos e ETFs.

Tela de celular com texto preto sobre fundo azul

Descrição gerada automaticamenteExemplo de uma ferramenta disponível no mercado:

#### 3.2.3. Qual o impacto das criptomoedas e como ela movimenta o mercado atual?

Ao operarem em uma rede distribuída, sem a necessidade de intermediários como bancos, as criptomoedas têm o potencial de eliminar barreiras geográficas e reduzir custos de transação, beneficiando diversos setores, inclusive serviços de contabilidade preço.

Perfil do Investidor:

De acordo com o cryptobank Monnos, os jovens são a maior parte dos investidores de criptomoedas, de acordo com dados levantados pela Instituição.   
A plataforma tem mais de 45 mil usuários, espalhados por 118 países – entre estes, 67% estão na faixa etária entre 18 e 25 anos.

Gráfico sobre compra e venda de criptomoeda no mercado atual

Gráfico inserido demonstrado a valorização ao longo dos anos (de 2011 até a presente data).

Gráfico, Histograma

Descrição gerada automaticamente

Pensando no seu tema, existe algo parecido no mercado? Como ele é?

Concorrência:

Logotipo

Descrição gerada automaticamenteLogotipo

Descrição gerada automaticamente com confiança médiaUma imagem contendo Logotipo

Descrição gerada automaticamenteLogotipo

Descrição gerada automaticamente

#### 3.2.4. Oque é a Binance?

A Binance é a maior corretora de criptomoedas do mundo, atendendo a 150 milhões de usuários registrados em mais de 180 países. Com taxas baixas e mais de 350 criptomoedas para negociar, a Binance é a corretora preferida para trades de Bitcoin, Altcoins e outros ativos virtuais.

#### 2.5.4. Regras de Negócios

“Como o próprio termo sugere, são regras que servem para definir ou restringir alguma ação nos processos de sua empresa. São declarações que irão descrever como determinadas operações devem ser realizadas e se há algum limite que precisa ser aplicado.” Heflo, 2022

A seguir listaremos as definições das regras de negócio, são elas que manterão o projeto em uma linha de raciocínio para que possamos desenvolver o projeto com o foco nos Requisitos fornecidos pelos usuários.

|  |  |
| --- | --- |
| **REGRAS DE NEGÓCIO** | |
| **Identificador** | **Descrição** |
| RN01 | O Aplicativo apresenta todas as possíveis Cryptos para compra. |
| RN02 | O aplicativo apresenta os valores das Cryptos e as suas variações gráficas. |
| RN03 | O aplicativo apresenta o histórico de valores de compra e venda das transações |
| RN04 | O aplicativo apresenta a entrada e saída de valores. |
| RN05 | O aplicativo deduz o valor de Impostos e a taxa percentual da Tharseo. |
| RN06 | O aplicativo deve ser intuitivo e ter um fluxo que o usuário consiga atingir o seu objetivo. |
| RN07 | O aplicativo deve gerar informe de rendimentos para se necessário o seu usuário declarar no imposto de renda. |

## Diagrama de Casos de Uso

|  |
| --- |
| Diagrama  O conteúdo gerado por IA pode estar incorreto. |

## Requisitos Funcionais

“Os requisitos funcionais são todos os problemas e necessidades que devem ser atendidos e resolvidos pelo software por meio de funções ou serviços”. (CUNHA, 2022).

Baseados nos requisitos de usuário, gera-se através de filtros e estudos os requisitos funcionais que irão dar base ao desenho e arquitetura do sistema e aplicativo. A importância da estruturação, reforça que o planejamento de qualquer projeto evita perda de tempo, foco e desgaste durante a sua execução. Utilizando o método Moscow para definir a prioridade dos itens dos requisitos funcionais em questão, com o complemento da ferramenta 5wh1 para que possamos definir de forma estruturada o papel e tarefa de cada requisito. A seguir a documentação dos requisitos funcionais:

|  |  |  |
| --- | --- | --- |
| **REQUISITOS FUNCIONAIS** | | |
| ***Identificador*** | ***Nome*** | ***Prioridade*** |
| **#RF01** | **Lista de criptos e seus valores de compra** | **Must Have** |
| **Descrição** | O sistema deverá apresentar cada Crypto disponível para negociação e seus valores atuais | |
| **Entradas e Pré-condições** | O Usuário para poder operar deverá ter o seu cadastro efetuado e deverá estar logado no sistema. | |
| **Saídas e Pós-Condições** | O usuário conseguirá setar o sistema com o valor de compra e após quanto deverá ser vendido após alcançado determinado valor | |
| **Fluxo de Eventos** | 1. O usuário irá fazer login no sistema; 2. O sistema retornará o menu principal com a carteira e principais moedas do usuário; 3. O usuário selecionará qual a crypto que ele vai negociar, seu valor de compra e a qual movimentação deverá ser feita a venda. 4. O sistema vai executar com as compras e vendas de acordo com o que foi pré-setado. | |
| ***Identificador*** | ***Nome*** | ***Prioridade*** |
| **#RF02** | **Carteira com valores disponíveis para compra e venda das Cryptos** | **Must Have** |
| **Descrição** | O Usuário devera a partir de determinada tela ter acesso a como se fosse um perfil, e nele deve estar apresentado a sua carteira com o valor disponível para negociar em compra e venda de Cryptos. | |
| **Entradas e Pré-condições** | O usuário deve cadastrar seus dados; O usuário deve depositar uma quantia mínima para conseguir operar com a compra e venda das cryptos; O usuário deve estar conectado na internet para atualização do mercado de cryptos em tempo real. | |
| **Saídas e Pós-Condições** | O aplicativo retornará uma atualização de valores em carteira e das cryptos antes, durante e após as transações; | |
| **Fluxo de Eventos** | 1. O usuário deve logar no sistema; 2. O usuário deverá acessar a sua carteira, caso haja valor disponível em carteira, se não o usuário também deverá realizar um depósito; 3. O sistema deve retornar os valores em carteira do usuário. | |
| ***Identificador*** | ***Nome*** | ***Prioridade*** |
| **#RF03** | **Acesso a moedas para compra e venda** | **Must Have** |
| **Descrição** | O usuário ao entrar na aba de cryptos, a princípio ela deverá disponibilizar uma aba de pesquisa e as principais cryptos para serem negociadas. | |
| **Entradas e Pré-condições** | O sistema deverá captar em tempo real as cryptos e as suas cotações a partir da API da binance. | |
| **Saídas e Pós-Condições** | Após a busca o sistema deverá apresentar a crypto e a opção de favorita-las, para não gerar mais a necessidade de pesquisa e para que as mesmas apareçam logo que logado. | |
| **Fluxo de Eventos** | 1. Após processo de login o usuário deverá acessar a zona de cryptos. 2. O sistema irá retornar a lista de cryptos, seus valores e uma barra para busca de diversas cryptos. | |
| ***Identificador*** | ***Nome*** | ***Prioridade*** |
| **#RF04** | **Histórico de transações** | **Must Have** |
| **Descrição** | O sistema deverá apresentar o histórico de transações, com a possibilidade de filtrar por período das transações. | |
| **Entradas e Pré-condições** | O usuário deverá acessar o aplicativo; O usuário deverá se logar no sistema; O usuário ao acessar o menu principal deverá acessar a aba de históricos das transações. | |
| **Saídas e Pós-Condições** | Ao acessar a aba de históricos o sistema deverá retornar as transações com data e valor de lucro. | |
| **Fluxo de Eventos** | 1. O usuário deve acessar o aplicativo. 2. O usuário deve logar no aplicativo; 3. O usuário deverá acessar a aba de históricos. | |

### 5.1 Requisitos do Usuário

Sendo assim estruturamos e definimos os requisitos de usuários listados abaixo:

“Requisitos de usuários são requisitos de mais alto nível, escritos por usuários, normalmente em linguagem natural e sem entrar em detalhes técnicos.” (Engenharia de Software Moderna, 2022)

|  |  |  |  |
| --- | --- | --- | --- |
| **REQUISITOS DE USUÁRIOS** | | |  |
| Identificador | Descrição | Prioridade | |
| UR01 | Eu como usuário deverei ter acesso a cotação de cada Crypto em tempo real. | Must Have | |
| UR02 | Eu como usuário deverei ter a minha carteira digital com os meus ativos | Must Have | |
| UR03 | Eu como Usuário deverei ter acesso as moedas disponíveis para compra e venda. | Must Have | |
| UR04 | Eu como usuário deverei ter acesso ao histórico de transações do dia com meus lucros. | Must Have | |

## Requisitos Não-funcionais

Após definidos os requisitos funcionais, daremos início ao levantamento dos requisitos não funcionais, partindo da mesma forma de estruturação:

“Os requisitos não funcionais são todos aqueles relacionados à forma como o software tornará realidade os que está sendo planejado. Ou seja, enquanto os requisitos funcionais estão focados no que será feito, os não funcionais descrevem como serão feitos. ” (CUNHA, 2022).

|  |  |
| --- | --- |
| **REQUISITOS NÃO FUNCIONAIS** | |
| **Identificador** | **Descrição** |
| RNF01 | O aplicativo terá como padrão as cores. |
| RNF02 | O aplicativo terá um logotipo. |
| RNF03 | O aplicativo possuirá tela de login e cadastro |
| RNF04 | O aplicativo terá como público-alvo o público em geral, investidores ou não, iniciantes ou experientes. |
| RNF05 | O sistema deverá ter a possibilidade de compartilhamento do aplicativo para indicar a mais usuários |

## Protótipo Navegável do Sistema

O Figma é uma ferramenta de design colaborativo baseada em nuvem, amplamente reconhecida por sua capacidade de permitir que equipes trabalhem simultaneamente em projetos de design de interface de usuário (UI) e experiência do usuário (UX). Lançado em 2016 pela Figma Inc., o Figma revolucionou a maneira como designers e equipes de design colaboram e criam interfaces digitais.

Design em Tempo Real: Uma das principais vantagens do Figma é a capacidade de editar designs em tempo real. Vários membros da equipe podem trabalhar simultaneamente no mesmo projeto, visualizando as alterações conforme são feitas. Isso elimina a necessidade de enviar arquivos e versões atualizadas manualmente.

Interface Intuitiva e Acessível: O Figma possui uma interface amigável e intuitiva, que facilita o aprendizado e a navegação para designers de diferentes níveis de habilidade. As ferramentas de design são acessíveis através de menus contextuais e barras de ferramentas bem-organizadas.

Funcionalidades de Prototipagem: Além de criar designs estáticos, o Figma suporta a criação de protótipos interativos. Isso permite que os designers criem fluxos de usuário, simulem interações e testem a usabilidade de seus designs diretamente na plataforma.

Componentes Reutilizáveis: O Figma utiliza o conceito de componentes, que são elementos de design reutilizáveis. Os componentes permitem que os designers criem bibliotecas de design consistente e atualizem automaticamente todos os elementos vinculados sempre que são modificados.

Colaboração e Comentários: A colaboração é central no Figma. Os usuários podem compartilhar seus designs com outras pessoas, permitindo que revisem, editem e comentem diretamente no projeto. Isso facilita o feedback em tempo real e melhora a eficiência da equipe.

Integração com Outras Ferramentas: O Figma oferece integrações com uma variedade de ferramentas de prototipagem, gerenciamento de projetos e desenvolvimento, como Slack, Jira, Trello e GitHub. Isso permite que equipes integrem facilmente o fluxo de trabalho de design com outros processos de desenvolvimento de software.

Eficiência de Colaboração: Equipes distribuídas ou colaboradores remotos podem trabalhar de forma síncrona no mesmo projeto, aumentando a produtividade e reduzindo o tempo de desenvolvimento.

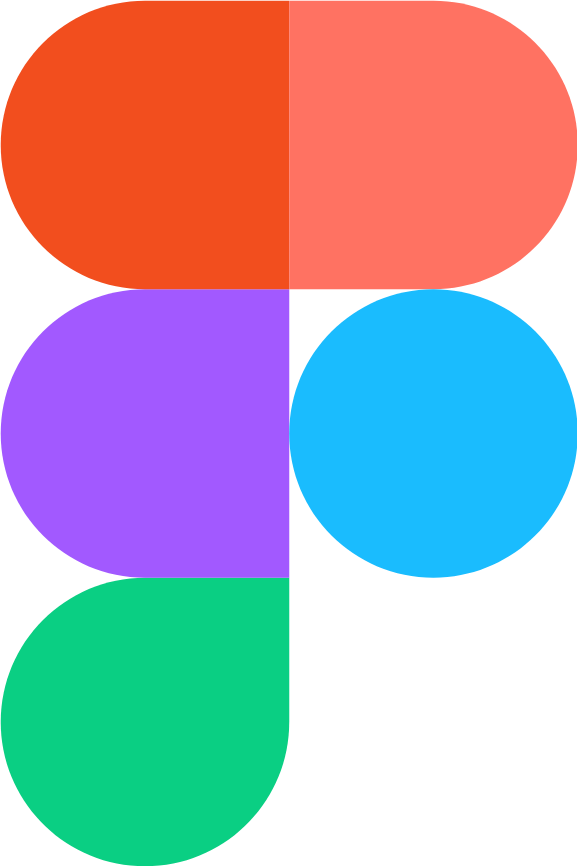
Consistência de Design: Com o uso de componentes e estilos compartilhados, o Figma ajuda a manter a consistência visual em todos os projetos, garantindo uma experiência de usuário coesa.

Feedback Iterativo: A capacidade de receber feedback instantâneo e iterar designs rapidamente melhora a qualidade final do produto e a satisfação do cliente.

Redução de Custos: Ao eliminar a necessidade de várias ferramentas separadas e simplificar o processo de design e desenvolvimento, o Figma pode ajudar a reduzir custos operacionais.

O Figma é amplamente utilizado por empresas de todos os tamanhos, desde startups até grandes corporações, devido à sua acessibilidade e recursos poderosos de colaboração. Ele se tornou uma ferramenta indispensável para equipes de design que buscam eficiência, agilidade e qualidade em seus processos de criação de interfaces digitais.

O Figma transformou a maneira como equipes de design colaboram e criam interfaces digitais, proporcionando uma plataforma robusta e intuitiva para o design de UI/UX. Com funcionalidades avançadas de colaboração, prototipagem e reutilização de componentes, o Figma continua a ser uma escolha popular entre designers e equipes de desenvolvimento que buscam eficiência, qualidade e inovação em seus projetos de design.



Através da ferramenta Figma foi feito todo o nosso processo de prototipagem, ela iniciou-se através de um estudo sobre as cores, onde utilizamos para inspiração de Layouts aplicações voltadas para o mesmo propósito da ideia inicial.

Em relação as cores utilizadas foram utilizadas tons frios para passar mais sobriedade e elegância, com a maior paleta partindo do azul para o roxo.

Gráfico, Gráfico de explosão solar

O conteúdo gerado por IA pode estar incorreto.

Abaixo imagens do protótipo gerado como ideia inicial do que tínhamos como proposta para posterior desenvolvimento do front end:

Tela de computador com fundo escuro

O conteúdo gerado por IA pode estar incorreto.

Tela de computador com texto preto sobre fundo escuro

Descrição gerada automaticamente

Tela de vídeo game

O conteúdo gerado por IA pode estar incorreto.

Tela de celular com fundo preto

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

Abaixo o link para acesso ao protótipo Web e Mobile: <https://github.com/RichardGuedesRib/tharseo_backend_v2/tree/main/docs>

[https://www.figma.com/design/Ilrsq6jhSmGQ84Ojiz0tWB/4%C2%BA-Semestre-projeto-crypto?node-id=0-1&t=mXcZTm2EZ9rVqpJv-1](https://www.figma.com/design/Ilrsq6jhSmGQ84Ojiz0tWB/4º-Semestre-projeto-crypto?node-id=0-1&t=mXcZTm2EZ9rVqpJv-1)

## Diagrama de Classes de Domínio

Diagrama, Esquemático

O conteúdo gerado por IA pode estar incorreto.

## Arquitetura da Solução

Texto

O conteúdo gerado por IA pode estar incorreto.

Atual estrutura arquitetural do nosso Backend, onde optamos por fazer em typescript e optamos por estruturar pelas classes, mantendo dentro de cada uma das pastas das classes os respectivos controllers, modules e services deixando uma organização e padrão para melhor entendimento nosso atual e para futuras manutenções uma forma mais intuitiva para quem for realizar a manutenção, abaixo a nossa organização da arquitetura do Frontend:

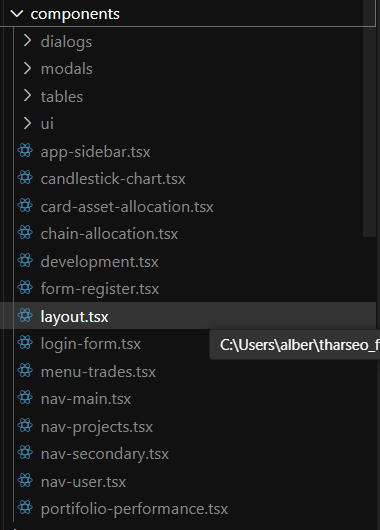
Texto

O conteúdo gerado por IA pode estar incorreto.

Onde foi organizada a estrutura por pastas, separando as API’s e conexões, as páginas desenvolvidas por tipo de Tela (como no exemplo da imagem abaixo), as models e services, os componentes criados para possível reaproveitamento, e todos os demais códigos de itens frontend da aplicação.

Interface gráfica do usuário, Texto

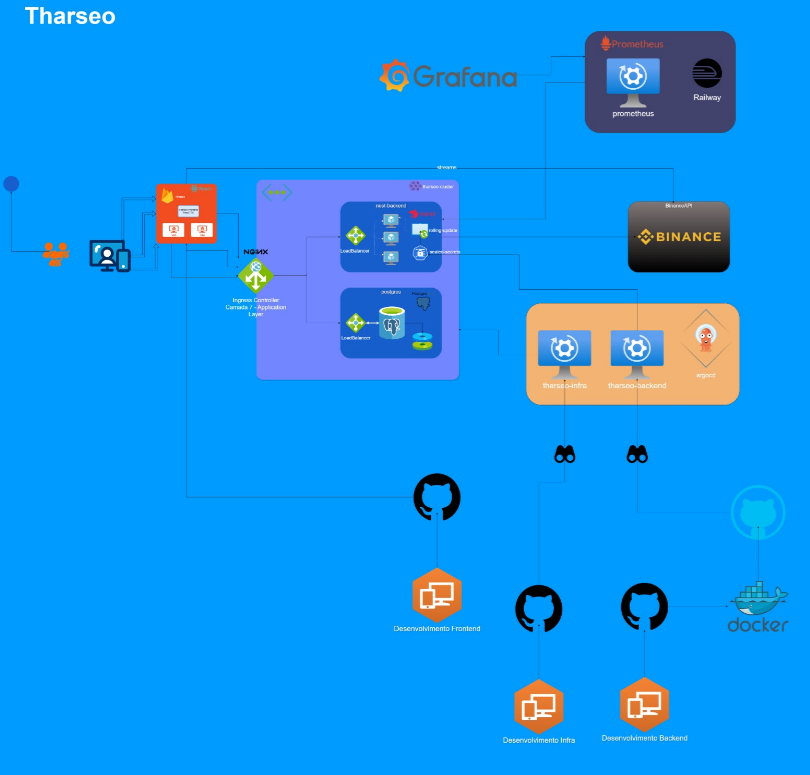
O conteúdo gerado por IA pode estar incorreto.



## Padrão Arquitetural

No projeto apresentado foi optado por trabalhar com o padrão arquitetural MVC devido ao modelo de negócio que seria desenvolvido e por ser a ferramenta que melhor nos atenderia nesse caso, os benefícios que observamos no modelo MVC também nos influenciou na escolha como a separação da lógica de negócios da interface do usuário, a questão da melhor afinidade com a escrita dos testes unitários que teriam que ser implementados no nosso projeto, e a questão da flexibilidade do padrão MVC em relação a possibilidades de frameworks, uma vez que o nosso Backend esteve em Java e depois optamos pela mudança para typescript, relativo a estrutura optamos por individualizar em pastas das classes e inserir dentro de cada uma delas os seus respectivos models, controllers, etc.

## C4 model - Diagrama de Contexto



Diagrama

O conteúdo gerado por IA pode estar incorreto.

## Frameworks de Trabalho

Nesse tópico apresentaremos todos os frameworks principais e alternativos que dão suporte e sustentação a aplicação que fora desenvolvida.

### 12.1. Typescript

TypeScript é um superconjunto tipado de JavaScript, o que significa que todo código JavaScript é um código TypeScript válido. O TypeScript adiciona muitos recursos novos além disso. TypeScript torna JavaScript mais parecido com uma linguagem fortemente tipada e orientada a objetos, semelhante a C# e Java. Isso significa que o código TypeScript tende a ser mais fácil de usar para grandes projetos e que esse código tende a ser mais fácil de entender e manter. A tipagem forte também significa que a linguagem pode (e é) pré-compilada e que variáveis ​​não podem ter valores atribuídos que estejam fora de seu intervalo declarado. Por exemplo, quando uma variável TypeScript é declarada como um número, você não pode atribuir um valor de texto a ela. Essa tipagem forte e orientação a objetos torna o TypeScript mais fácil de depurar e manter, e esses foram dois dos pontos mais fracos do JavaScript padrão

### 12.2. Jest

O Jest foi inicialmente criado para testar o framework React, também criado pelo Facebook. Porém sua implementação se tornou muito mais ampla, sendo utilizado como ferramenta de teste unitário para diversas plataformas JavaScript como Node e Redux, e até mesmo plataformas em TypeScript como Angular e Ionic.

Jest é um poderoso framework de testes em JavaScript com foco na simplicidade. Aliás, essa é a definição do site oficial. Leia este conteúdo para saber mais detalhes do que é Jest e usá-lo, principalmente, no lado front-end.

Jest e os testes unitários

A sua função está muito ligada aos testes unitários. Portanto, relembre o que são eles.

Em suma, os testes unitários aferem o código escrito, vendo o seu nível de adequação em todos os detalhes, inclusive na menor fração.

Nas linguagens orientadas a objeto, por exemplo, a menor parte de um código pode se referir a um método de uma classe. Os testes unitários são aplicados nesses métodos.

De modo geral, eles são usados para reduzir erros e, assim, ter entregas com mais qualidade e menos retrabalho.

Jest e o React

Como você leu inicialmente, Jest foi criado pelo Facebook. Portanto, ele foi pensado para testar o framework React. Mas ele acabou tendo um uso mais amplo, testando várias outras plataformas, como Node e Redux. Plataformas em TypeScript, como Angular e Ionic, também foram o seu foco.

Hoje em dia ele é compatível com o EcmaScript 5 em diante e também pode ser instalado pelo Yarn ou NPM.

Jest e suas vantagens

As vantagens apresentadas pelo Jest são:

Rapidez e segurança: Jest pode realizar testes paralelos e com resultados confiáveis.

Cobertura de código: é possível gerar cobertura de código adicionando o Coverage, sem precisar realizar outra configuração;

Testes mais agradáveis: realizar testes com Jest é mais prático e agradável, tanto que o GitHub tem quase 4 milhões de repositórios públicos com esta tecnologia.

Prova disso são as empresas que o utilizam. Atualmente, a tecnologia é usada por empresas como Facebook, Twitter, Instagram, Airbnb e Spotify

### 13.3. API Binance

Podemos definir API como Interface de Programação de Aplicações, é um conjunto de definições e protocolos que permite a comunicação entre diferentes softwares. APIs especificam como os componentes de software devem interagir, facilitando a integração de novas funcionalidades e serviços. Elas são amplamente utilizadas para permitir que diferentes sistemas e aplicações troquem informações e funcionem em conjunto de maneira eficiente.

Tipos de API:

APIs Web: Permitem a comunicação entre servidores e clientes através da internet, utilizando protocolos como HTTP/HTTPS. São comuns em serviços web e aplicativos móveis.

APIs de Sistema Operacional: Fornecem acesso a funcionalidades do sistema operacional, permitindo que aplicações interajam com hardware e recursos do sistema.

APIs de Biblioteca: Oferecem funções e métodos pré-definidos para realizar tarefas específicas dentro de uma linguagem de programação.

APIs de Banco de Dados: Facilitam a interação entre aplicações e sistemas de gerenciamento de bancos de dados.

Componentes de uma API:

End Points: URLs específicas onde os recursos da API podem ser acessados.

Métodos HTTP: Definem a ação a ser realizada, como GET (recuperar dados), POST (enviar dados), PUT (atualizar dados) e DELETE (remover dados).

Cabeçalhos de Requisição: Contêm informações adicionais sobre a solicitação, como autenticação e formato de dados.

Corpo da Requisição: Contém os dados que são enviados para a API (usado principalmente com métodos POST e PUT).

Respostas da API: Incluem códigos de status HTTP e, frequentemente, dados no formato JSON ou XML.

Vantagens das APIs:

Interoperabilidade: Permitem que diferentes sistemas e plataformas trabalhem juntos.

Reutilização de Código: Promovem a reutilização de funcionalidades existentes, economizando tempo e esforço.

Desenvolvimento Modular: Facilitam o desenvolvimento de aplicações modulares e escaláveis.

Exemplos de Uso de APIs:

Serviços de Pagamento: APIs como PayPal e Stripe permitem que lojas online processem pagamentos.

Redes Sociais: APIs do Facebook, Twitter e Instagram permitem a integração de funcionalidades dessas plataformas em outras aplicações.

Mapas e Geolocalização: APIs do Google Maps permitem a inclusão de mapas e serviços de localização em aplicações.

APIs são ferramentas essenciais no desenvolvimento de software moderno, permitindo a criação de aplicações mais integradas, eficientes e fáceis de manter. Elas possibilitam que diferentes sistemas e aplicações se comuniquem de maneira padronizada, promovendo a interoperabilidade e a reutilização de código.

Foi utilizada a API da Binance para que fosse possível importar os valores e cotações das cryptos, utilizada também para acesso a carteira do cliente e para possibilitar a compra e a venda das cryptos, possibilitando a visualização da cotação além das transações de compras e vendas conforme setado pelo usuário.

### 13.4. PostgresSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados relacional de código aberto que oferece uma solução robusta e escalável para gerenciar e manipular dados. Com uma longa história de desenvolvimento, o PostgreSQL é conhecido por sua estabilidade, segurança e desempenho.

Desenvolvido inicialmente pela Universidade da Califórnia em Berkeley, o PostgreSQL é mantido atualmente pela comunidade global de desenvolvedores. Isso significa que o código-fonte do PostgreSQL é disponibilizado voluntariamente, permitindo que qualquer pessoa possa estudá-lo, modificá-lo e distribuí-lo sem restrição.

Alguns dos principais recursos do PostgreSQL incluem:

- Suporte a transações ACID, que garantem a consistência e integridade dos dados.

- Suporte a múltiplos mecanismos de armazenamento, incluindo o mecanismo de armazenamento de tabelas.

- Suporte a linguagens de programação como C, C++, Java, Python e Ruby.

- Ferramentas de administração e gerenciamento de banco de dados, como o pgAdmin.

O PostgreSQL é amplamente utilizado em aplicações empresariais e na web devido à sua robustez, eficiência e escalabilidade. Além disso, o PostgreSQL é compatível com uma variedade de plataformas, incluindo Windows, Linux e macOS

Em resumo, o PostgreSQL é uma solução de banco de dados relacional de código aberto que oferece estabilidade, segurança e desempenho. Com sua longa história de desenvolvimento e sua comunidade global de desenvolvedores, o PostgreSQL é uma escolha popular para aplicações empresariais e na web.

### 13.5. Websocket

WebSocket é uma tecnologia de comunicação bidirecional que permite a transmissão de dados entre um navegador web e um servidor de maneira eficiente e em tempo real. Diferente do modelo tradicional de requisições HTTP, onde o navegador faz uma solicitação e espera por uma resposta do servidor, com WebSockets, uma conexão persistente é mantida, permitindo a troca de mensagens de ambos os lados a qualquer momento.

Como Funciona

Estabelecendo a Conexão: Um WebSocket começa como uma conexão HTTP normal, com um pedido de "upgrade" sendo enviado do cliente para o servidor, solicitando a troca para o protocolo WebSocket. Uma vez aceito, a comunicação é mantida de forma persistente, e mensagens podem fluir em ambas as direções.

Bidirecionalidade: Uma vez estabelecida a conexão, tanto o cliente quanto o servidor podem enviar mensagens a qualquer momento sem a necessidade de uma solicitação explícita.

Eventos e Callbacks: A comunicação WebSocket é geralmente baseada em eventos. O cliente e o servidor podem definir manipuladores para eventos como a abertura da conexão, recebimento de uma mensagem, ou fechamento da conexão.

### 13.6. Git

O git é um sistema de controle de versão de códigos. Gerencia ramificações feitas por diferentes pessoas de um time de modo a manter a ordem e integração de todas as partes. Ou seja, mesmo que várias pessoas estejam trabalhando em um programa ao mesmo tempo, ainda é possível controlar as mudanças para evitar erros.

Sem um controle de versionamento, como o que o git propõe, pode haver o risco de pessoas acessarem diferentes versões e suas mudanças sumirem por conta da desorganização.

Além disso, o git é um sistema que permite rastrear alterações e manter um histórico dessas alterações para possibilitar voltar a uma versão estável sempre que ocorrer um problema. Ou seja, caso algum erro quebre o sistema em desenvolvimento, é viável voltar para a versão funcional saudável.

Em complemento ao que falamos, o git ajuda também na transparência e no controle das mudanças feitas. Viabiliza o conhecimento sobre quem fez o quê e a frequência dessas modificações.

Para prosseguirmos, no entanto, é importante destacar a diferença do git para o github. O github é uma plataforma de gestão de códigos que se baseia no git e adiciona um fator social, de compartilhamento e interação interpessoal. O hub é crucial para a gestão de projetos, pois ajuda a visualizar as responsabilidades e as entregas, com um controle de prazos e de detalhes.

O hub é um site aberto para todos. Então, é comum que pessoas colaborem com projetos de outras pessoas dev que nem conhecem, de outros estados ou países. Projetos open-source são disponibilizados lá e podem ser alterados por qualquer pessoa desenvolvedora. Isso só reforça o aspecto comunitário e colaborativo da área de programação.

Ou seja, enquanto o git é o sistema por trás, que cuida dos códigos e comandos e gerência as ramificações, o hub é a plataforma onde se postam os códigos.

O github é uma das plataformas baseadas no git, inclusive. Existem várias outras que tentam implementar as mesmas funções e nelas se usam os mesmos comandos git. Alguns exemplos delas: GitLab e Apache Allura.

### 13.7. Docker

O Docker é um software de código aberto usado para implantar aplicativos dentro de containers virtuais. A conteinerização permite que vários aplicativos funcionem em diferentes ambientes complexos. Por exemplo: o Docker permite executar o WordPress em sistemas Windows, Linux e macOS, sem problemas.

Docker vs Máquina Virtual

Embora o Docker e as máquinas virtuais tenham um propósito semelhante, seu desempenho, portabilidade e suporte a sistemas operacionais diferem significativamente.

A principal diferença é que os containers do Docker compartilham o sistema operacional do host, enquanto as máquinas virtuais também têm um sistema operacional convidado sendo executado no sistema host. Esse método de operação afeta o desempenho, as necessidades de hardware e o suporte do SO.

### 13.8. SonarCloud

SonarCloud é uma ferramenta de análise de código baseada em nuvem desenvolvida pela SonarSource. Ela oferece serviços de inspeção de código para identificar problemas relacionados à qualidade, segurança e desempenho em projetos de software. SonarCloud é amplamente utilizado por equipes de desenvolvimento para melhorar a qualidade do código, garantir padrões de codificação e detectar vulnerabilidades de segurança em fases iniciais do desenvolvimento.

Características Principais:

Análise de Qualidade do Código: SonarCloud analisa o código-fonte para identificar problemas de qualidade, como bugs, vulnerabilidades, duplicação de código, cobertura de testes e violações de padrões de codificação.

Integração com CI/CD: A ferramenta se integra facilmente com sistemas de Integração Contínua/Entrega Contínua (CI/CD) como Jenkins, Travis CI, Azure DevOps, GitHub Actions e Bitbucket Pipelines. Isso permite a análise automática do código em cada fase do desenvolvimento.

Suporte a Várias Linguagens: SonarCloud suporta mais de 25 linguagens de programação, incluindo Java, JavaScript, TypeScript, Python, C#, PHP, C++, Ruby e Go. Essa ampla compatibilidade facilita a análise de projetos poliglotas.

Feedback em Tempo Real: Fornece feedback contínuo aos desenvolvedores durante o processo de desenvolvimento, permitindo que problemas de qualidade e segurança sejam corrigidos imediatamente.

Dashboard e Relatórios: SonarCloud oferece um dashboard intuitivo que fornece uma visão abrangente do estado da qualidade do código. Relatórios detalhados ajudam as equipes a identificar áreas problemáticas e acompanhar o progresso ao longo do tempo.

Regra de Qualidade Personalizável: Permite que as equipes personalizem regras de qualidade de acordo com suas necessidades específicas, garantindo que o código atenda aos padrões internos e requisitos do projeto.

Análise de Ramificações e Pull Requests: Analisa ramificações e pull requests para detectar problemas antes de serem mesclados com a linha principal do código, prevenindo a introdução de novos defeitos.

Benefícios do SonarCloud:

Melhoria Contínua da Qualidade do Código: A análise contínua e o feedback em tempo real ajudam os desenvolvedores a manter um alto padrão de qualidade do código ao longo do ciclo de desenvolvimento.

Redução de Custos com Correção de Bugs: Detectar e corrigir problemas de qualidade e segurança nas fases iniciais do desenvolvimento reduz significativamente os custos associados à correção de bugs em estágios posteriores.

Maior Segurança: A identificação precoce de vulnerabilidades de segurança ajuda a proteger o software contra ameaças potenciais e a cumprir normas de segurança.

Facilidade de Uso: A interface intuitiva e a fácil integração com ferramentas de CI/CD existentes tornam o SonarCloud acessível e fácil de usar para equipes de todos os tamanhos.

Visibilidade e Transparência: Os dashboards e relatórios fornecem visibilidade sobre a qualidade do código, permitindo que todos os membros da equipe, incluindo gerentes de projeto e stakeholders, acompanhem o progresso e identifiquem áreas de melhoria.

### 13.9. Azure

O Microsoft Azure é uma plataforma destinada à execução de aplicativos e serviços, baseada nos conceitos da computação em nuvem da Microsoft. Anteriormente conhecido como Windows Azure, ele é, atualmente, um dos principais pontos de ação da empresa e de seu CEO, Satya Nadella.

Atualmente, adotado cada vez mais pelas grandes corporações, a plataforma é um importante passo dentro da jornada de transformação digital. No entanto, dúvidas e inseguranças geradas pela falta de conhecimento a respeito da computação em nuvem atrasam, em muitos casos esse processo de modernização.

Antes de entendermos como se dá a atuação do Microsoft Azure, é necessário dar um passo para trás e compreender o que é a nuvem.

A definição mais direta possível afirma que a nuvem é uma rede global de servidores virtuais que contam com as mais variadas funções. Por não ser física, essa rede consegue conectar servidores remotos espalhados globalmente, possibilitando que eles operem em conjunto.

Mas, afinal, quais são essas funções? Como já dissemos aqui, elas são múltiplas – e, por isso, a nuvem atende tão bem as necessidades do mundo corporativo moderno. Dentre elas, podemos citar o armazenamento e gerenciamento de dados e o desenvolvimento e execução de aplicativos e serviços como webmail e softwares.

A virtualidade da nuvem também permite que seus usuários consigam acessar todos os seus recursos por meio dos mais variados dispositivos. Na prática, isso traz um grande incentivo à produtividade.

Existem quatro tipos de nuvem:

Nuvem pública: modelo padrão. Os serviços são oferecidos a todos que querem comprá-los (ou simplesmente usá-los) na internet pública.

Nuvem privada: também conhecida como nuvem corporativa. Nela, os serviços são disponibilizados somente a usuários selecionados.

Nuvem de comunidade: os recursos são compartilhados entre organizações com interesses convergentes, como instituições ligadas a governos, por exemplo.

Nuvem híbrida: combinação da nuvem pública e privada; os dados e aplicativos podem ser compartilhados entre elas de acordo com a demanda. É a opção mais flexível e a classificação do Microsoft Azure.

O Microsoft Azure é a plataforma de nuvem híbrida da Microsoft. Por definição, ele concentra toda a infraestrutura de cloud computing da empresa, incluindo os seus produtos, serviços e aplicativos.

Do ponto de vista prático, há dois modelos de serviços na nuvem: IaaS (Infrastructure as a Service, ou Infraestrutura como um Serviço, em português) e PaaS (Platform as a Service, ou Plataforma como um Serviço, em português).

Microsoft Azure e IaaS

O IaaS é voltado para a infraestrutura de computação, que inclui itens como servidores e backups. O formato prioriza a diminuição de gastos com a compra e administração de servidores físicos e demais estruturas de processamento de dados. Nesse sentido, o Microsoft Azure gerencia a infraestrutura enquanto o usuário cuida de toda a configuração (incluindo a compra e instalação) do software.

Dessa maneira, além de otimizar os gastos, o IaaS elimina preocupações com a infraestrutura de TI da empresa, incentivando a inovação e garantindo a segurança dos dados e aplicativos mesmo em situações de desastre e demais interrupções.

O Microsoft Azure também funciona no modelo PaaS, que oferece ainda mais tranquilidade ao usuário. E o motivo é simples! Além de incluir todos os recursos de IaaS, ele também engloba todo o ambiente necessário para o desenvolvimento e administração dos aplicativos na nuvem.

Ou seja, o usuário só precisa se preocupar em gerenciar os apps que desenvolveu, deixando todo o resto por conta do provedor. Além disso, PaaS também oferece serviços de business intelligence (BI), gerenciamento de bancos de dados e middleware (uma camada de mediação entre diferentes tecnologias e aplicativos).

Quando tratamos de suas vantagens, podemos dizer que o Microsoft Azure facilita consideravelmente a vida dos desenvolvedores. O modelo PaaS tem os mesmos benefícios do IaaS – e mais.

Nesse formato, os destaques são as ferramentas sofisticadas de desenvolvimento que permitem o gerenciamento de todas as etapas de criação de aplicativos. Elas garantem a diminuição do tempo de codificação de novos apps e permitem que os desenvolvedores trabalhem não somente nos computadores tradicionais, mas também em dispositivos móveis. A colaboração também entra em evidência aqui: em um ambiente de desenvolvimento online, profissionais conseguem trabalhar juntos, mesmo à distância.

Graças a sua variedade de recursos, o Microsoft Azure pode ser utilizado por empresas dos mais diferentes setores – desde a saúde até o varejo, passando pela manufatura e logística.

Levando em consideração todas as possibilidades proporcionadas pela computação em nuvem, não é uma surpresa o fato de que o Microsoft Azure tem muitos serviços diferentes.

Com o Azure, é possível desenvolver e implantar apps diversos, inclusive para dispositivos móveis. Os usuários também podem migrar aplicativos, servidores e bancos de dados para a nuvem da Microsoft.

Há, ainda, a opção de trabalhar sem servidores no Microsoft Azure. Nela, os desenvolvedores conseguem produzir com mais rapidez e os custos com a infraestrutura são otimizados. Vale notar que os aplicativos sem servidor continuam contando com a possibilidade de incluir recursos de Inteligência Artificial e algoritmos de machine learning.

A computação de alto desempenho (HPC) também tem um grande aliado no Microsoft Azure. Com a nuvem da Microsoft, empresas de diferentes setores conseguem gerenciar e executar facilmente os seus fluxos de trabalho. Aqui, versatilidade e escalabilidade são palavras-chave.

Outro uso muito relevante para o Microsoft Azure é o seu papel no backup e recuperação de desastres. O mundo corporativo lida com um número crescente de dados; ter espaço para armazená-los com segurança é imprescindível. O Azure permite que as companhias aumentem a sua capacidade de armazenamento de acordo com a demanda, otimizando gastos.

Além disso, é importante notar que todas as soluções do Microsoft Azure foram projetadas para facilitar a conformidade com as leis de proteção de dados – isto é, LGPD e GDPR.

Microsoft Azure e segurança

Como citamos aqui, a segurança é um item muito importante no Microsoft Azure. A empresa investe massivamente nesse aspecto – no total, US$ 1 bilhão são empreendidos anualmente em soluções de cibersegurança na nuvem da Microsoft.

O Microsoft Azure tem camadas espessas de segurança que protegem os usuários das mais variadas ameaças. Exemplo disso são os ataques DDoS, que não são classificados exatamente como uma invasão ao sistema, mas como uma forma de impedir que seus recursos sejam acessados.

No dia a dia, todas as operações realizadas no Microsoft Azure também são altamente protegidas. No Azure Active Directory (AD), por exemplo, é possível gerenciar identidades e controlar os acessos, adotando as melhores práticas do setor. Dessa maneira, garante-se que somente as pessoas certas, sob condições específicas, consigam acessar os dados da empresa.

Além disso, o Microsoft Azure também garante a segurança da rede por meio de controles fortes e nativos da plataforma, como o Firewall do Azure. Outra ferramenta importante é o Azure Key Vault, cuja função é armazenar e gerenciar senhas. É interessante notar que, antes de realizar qualquer ação nele, é preciso passar por um processo de autenticação.

### 13.10. Clarity

Clarity é uma ferramenta de análise de comportamento de usuários desenvolvida pela Microsoft, focada em fornecer insights detalhados sobre como os visitantes interagem com um site. Lançada em 2020, Clarity é uma solução gratuita que oferece gravações de sessões, mapas de calor e diversas métricas para ajudar os proprietários de sites a entender melhor a experiência do usuário e identificar áreas para melhorias.

Características Principais:

Gravações de Sessões: Clarity permite que os proprietários de sites assistam gravações de sessões dos usuários, mostrando exatamente como eles navegam pelo site, onde clicam, quanto tempo passam em cada página e outros comportamentos detalhados.

Mapas de Calor: Clarity gera mapas de calor que visualizam onde os usuários clicam, movem o mouse e rolam. Esses mapas ajudam a identificar padrões de interação e áreas de maior interesse ou frustração.

Métricas e Análises: A ferramenta fornece uma variedade de métricas de engajamento, como tempo na página, taxa de rolagem, e cliques mortos (cliques em áreas não interativas). Essas métricas ajudam a avaliar a eficácia do design do site e a usabilidade.

Detecção de Problemas: Clarity pode identificar automaticamente problemas de usabilidade, como cliques rápidos ou hesitações, que podem indicar confusão ou frustração dos usuários.

Filtros e Segmentação: Os usuários podem filtrar e segmentar dados de sessões e mapas de calor por diversos critérios, como dispositivo, navegador, origem de tráfego e comportamento específico, para obter insights mais detalhados e específicos.

Privacidade e Segurança: Clarity é projetado para ser compatível com regulamentos de privacidade, como o GDPR. Ele anonima automaticamente dados sensíveis e permite a exclusão de dados pessoais dos usuários conforme necessário.

Benefícios do Clarity:

Insights Detalhados: Com gravações de sessões e mapas de calor, Clarity oferece uma visão detalhada de como os usuários interagem com o site, permitindo a identificação de problemas e oportunidades de melhoria.

Facilidade de Uso: A interface intuitiva de Clarity facilita a configuração e o uso, tornando-a acessível para desenvolvedores, designers e proprietários de sites sem necessidade de treinamento extensivo.

Gratuito: Clarity é uma ferramenta gratuita, tornando-a uma opção atrativa para pequenas empresas e sites pessoais que desejam obter insights sobre a experiência do usuário sem incorrer em custos adicionais.

Compatibilidade com Outras Ferramentas: Clarity pode ser usado em conjunto com outras ferramentas de análise, como Google Analytics, para proporcionar uma visão mais abrangente do comportamento do usuário.

Melhoria da Experiência do Usuário: Ao entender melhor como os usuários interagem com um site, os proprietários podem fazer ajustes informados no design e na funcionalidade para melhorar a experiência do usuário e aumentar a satisfação e conversão.

13.11. Owasp

OWASP, que significa "Open Web Application Security Project" (Projeto Aberto de Segurança em Aplicações Web), é uma organização sem fins lucrativos focada em melhorar a segurança de software. Fundada em 2001, a OWASP oferece uma ampla gama de recursos, ferramentas, e documentação para ajudar desenvolvedores, arquitetos de software e organizações a entender e mitigar os riscos de segurança em aplicações web.

Principais Objetivos e Atividades da OWASP

Educação e Conscientização:

OWASP Top Ten: Uma lista das dez vulnerabilidades mais críticas em aplicações web. Esta lista é amplamente reconhecida e utilizada como um padrão de referência para segurança de aplicações.

Documentação: Diversos guias, livros e artigos sobre melhores práticas em segurança de software.

Treinamentos e Conferências: OWASP organiza eventos e conferências globais para promover a troca de conhecimento e treinamento em segurança de software.

Ferramentas e Projetos:

OWASP ZAP (Zed Attack Proxy): Uma das ferramentas de segurança de aplicações mais populares, usada para encontrar vulnerabilidades em aplicações web.

OWASP Dependency-Check: Ferramenta que verifica componentes e bibliotecas de software para identificar vulnerabilidades conhecidas.

OWASP ASVS (Application Security Verification Standard): Um framework de verificação de segurança para aplicações, que fornece um conjunto de requisitos para o desenvolvimento seguro.

Comunidade e Colaboração:

Capítulos Locais: OWASP tem capítulos regionais ao redor do mundo que organizam encontros, palestras e workshops para promover a segurança de software em nível local.

Projetos Open Source: A OWASP mantém uma série de projetos de código aberto focados em diferentes aspectos da segurança de software.

Importância da OWASP

A OWASP é uma referência crucial para profissionais de segurança da informação, desenvolvedores e gestores de TI. Suas publicações e ferramentas são amplamente utilizadas para ajudar a identificar, entender e remediar vulnerabilidades em aplicações web. A OWASP também promove a importância da segurança no ciclo de desenvolvimento de software (SDLC) e incentiva a adoção de práticas de desenvolvimento seguro desde as fases iniciais de um projeto.

Recursos Populares da OWASP

OWASP Top Ten:

Fornece uma visão geral das falhas de segurança mais críticas que afetam as aplicações web.

Atualizado periodicamente para refletir as ameaças emergentes.

OWASP ZAP:

Uma ferramenta de proxy interceptador que ajuda a encontrar vulnerabilidades em aplicações web durante o teste de segurança.

OWASP ASVS:

Um conjunto de normas e requisitos que ajudam a definir e alcançar níveis de segurança apropriados para aplicações web.

Auto Release: O número de versão em auto release refere-se à prática de incrementar e gerenciar automaticamente os números de versão de um software sempre que uma nova release é feita. Isso é crucial para rastrear mudanças, melhorias e correções em um projeto de software de forma sistemática e compreensível. A numeração de versão segue frequentemente o padrão de Versionamento Semântico (SemVer), que é amplamente adotado na indústria de software.

Versionamento Semântico (SemVer)

O Versionamento Semântico (SemVer) é um sistema de numeração de versão que utiliza um formato MAJOR.MINOR.PATCH, onde:

MAJOR: Incrementado quando há mudanças incompatíveis com versões anteriores.

MINOR: Incrementado quando novas funcionalidades são adicionadas de maneira compatível com versões anteriores.

PATCH: Incrementado quando são feitas correções de bugs de maneira compatível com versões anteriores.

Exemplo de Números de Versão

1.0.0: Primeira versão estável.

1.1.0: Nova funcionalidade adicionada de forma compatível.

1.1.1: Correção de bug.

2.0.0: Mudança incompatível com versões anteriores.

Como Funciona no Auto Release

Quando você implementa auto release, o número de versão do seu projeto é automaticamente incrementado com base nas mudanças detectadas. Isso é feito através da análise de commits, mensagens de commit ou tags para determinar se a mudança é uma correção de bug, uma nova funcionalidade ou uma alteração significativa que quebra a compatibilidade com versões anteriores.

Ferramentas para Auto Release

Ferramentas como semantic-release podem automatizar todo o processo de versionamento e lançamento. Elas analisam as mensagens de commit para determinar o tipo de mudança e incrementam o número de versão apropriado. Além disso, essas ferramentas podem gerar changelogs, criar tags de versão no repositório e até publicar pacotes em registries (como npm, PyPI, etc.).

## Estrutura Base do Front End

## Tela de celular com fundo preto O conteúdo gerado por IA pode estar incorreto.Tela de computador com letras brancas em fundo preto O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto O conteúdo gerado por IA pode estar incorreto.Tela de computador com letras e números em fundo preto O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Aplicativo O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Aplicativo O conteúdo gerado por IA pode estar incorreto.

## Modelo Relacional ou Projeto de Banco de Dados NoSQL

“Modelagem de banco de dados é o processo de levantamento, análise, categorização e exploração de todos os dados e tipos de informações que irão sustentar uma aplicação. “ (Escola Superior de Redes, 2020)

Após a estruturação e estudo dos requisitos, iniciamos uma etapa muito importante na construção de qualquer aplicação: A Modelagem do Banco de Dados.

Listamos os requerimentos e através de análises, listamos as entidades e atributos que seriam pertinentes a base da aplicação. Após formalização das entidades e atributos, suas devidas cardinalidades, foram formados os relacionamentos de forma a agilizar a gestão de dados do banco de dados.

Com a estrutura do BD formada, foram iniciadas as normalizações baseadas nas prioridades de seus conceitos:

1. Primeira Forma Normal de Codd:

• Possuir somente dados atômicos;

• Não há grupos de atributos repetidos;

• Todas as entidades possuem uma chave primária;

• Relação não possui atributos multivalorados.

2. Segunda Forma Normal de Codd:

• Estar na 1FN;

• Todos os atributos não-chaves são funcionalmente dependentes de todas as partes da chave primária;

• Não existem dependências parciais.

3. Terceira Forma Normal de Codd:

• Estar na 2FN;

• Nenhuma coluna não-chave depende de outra coluna não-chave.

### 14.1 Modelo do Banco relacional

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

## Plano de Testes

### Plano de Testes Tharseo Backend v2

**Introdução**

O **Tharseo Backend v2** é uma API robusta, escalável e eficiente, projetada para suportar uma ampla variedade de recursos para a plataforma Tharseo. Esta API serve como o núcleo do sistema Tharseo, lidando com lógica de negócios, gerenciamento de banco de dados, autenticação e comunicação com clientes front-end e serviços externos. O Tharseo Backend v2 utiliza tecnologias modernas como **NestJS**, **Prisma ORM**, **Autenticação JWT** e **PostgreSQL** para fornecer um ambiente seguro, confiável e de alto desempenho para a plataforma.

O objetivo desse plano de testes, é elencar a cobertura dos métodos e fluxos de forma granular proporcionando segurança ao desenvolver e a prevenção de surpresas ao mudar determinados pontos do código que são usados por diversos métodos e repassar isso para produção.

**Métodos dos Testes**

Para realização dos testes a nível backend, utilizaremos o framework Jest e Supertest, que são bem conhecidos no mercado e apresentam um nível de confiança aceitável pela comunidade.

No momento, temos como alvo os controllers e suas rotas, serviços e seus métodos, classes e seus métodos.

Temos um total de testes unitários de 22 suítes com total de 162 testes. Também temos 4 suítes com 20 testes de integração (e2e).

Os testes unitários serão responsáveis em testar cada ponto da aplicação, em casos de sucesso e casos de falha.

Já os testes e2e, testarão o fluxo da integração, desde a rota recebida, regra de negócio validada, registro em banco de dados e retorno da resposta para o requisitante.

**Tipo de Testes**

* Testes Unitários - (Jest - NextJs)
* Testes de integração (e2e) - (SuperTest NextJs)

**Estratégia de Validação**

Para validação optamos em duas formas:  
 Para os testes unitários, ao desenvolver, podemos chamar os testes através de npm run test para testar localmente.

Para adicionar um duplo check, nossa pipeline tem como requisito antes de efetuar o deploy para produção, rodar todos os testes unitários e em caso de falha, o processo de deploy é abortado.

Para os testes de integração, os requisitos para rodar localmente, é instanciar um banco de dados postgres definindo as credenciais iguais as do arquivo env.test. Os testes serão iniciados e o feedback através do terminal.

Também adicionamos um duplo check em nossa pipeline, durante o deploy, o GitActions emula um banco de dados, em seguida, efetuamos um push de nossa estrutura do banco de dados para esse banco do GitActions, e ele é instanciado e permite nossa interação. Dessa forma, os testes e2e podem ser validados e em caso de falha o deploy é abortado.

**Cobertura dos Testes**

* 162 testes unitários
* 20 testes e2e

**Execução e Descrição dos Testes Unitários**

**AuthController**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 1 | auth.controller.spec.ts | Instancia | deve instanciar o auth.controller | sucesso | deve ser instanciado |
| 2 | auth.controller.spec.ts | signUp | deve criar e retornar um novo usuário | sucesso | deve retornar o usuário criado |
| 3 | auth.controller.spec.ts | signUp | deve retornar um erro se o usuario nao for criado | falha | deve retornar uma BadRequestException |
| 4 | auth.controller.spec.ts | signIn | deve retornar um payload de autenticação | sucesso | deve retornar um login com sucesso |
| 5 | auth.controller.spec.ts | signIn | deve retornar UnauthorizedException se o login falhar | falha | deve retornar UnauthorizedException |

**AuthGuard**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 6 | auth.guard.spec.ts | AuthGuard | deve instanciar o auth.guard | sucesso | deve ser instanciado |
| 7 | auth.guard.spec.ts | canActivate | deve retornar true se o token for válido | sucesso | deve retornar true |
| 8 | auth.guard.spec.ts | canActivate | deve lançar UnauthorizedException se o token não for encontrado | falha | deve retornar UnauthorizedException |
| 9 | auth.guard.spec.ts | canActivate | deve lançar UnauthorizedException se o token for inválido | falha | deve lançar UnauthorizedException |
| 10 | auth.guard.spec.ts | extractTokenFromHeader | deve extrair corretamente o token da header | sucesso | deve retornar o token |
| 11 | auth.guard.spec.ts | extractTokenFromHeader | deve retornar undefined se a header de authorization estiver ausente | falha | deve retornar o undefined |
| 12 | auth.guard.spec.ts | extractTokenFromHeader | deve retornar undefined se o tipo não for "Bearer | falha | deve retornar o undefined |

**AuthGuard**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 13 | auth.service.spec.ts | AuthService | deve instanciar o auth.service | sucesso | deve ser instanciado |
| 14 | auth.service.spec.ts | signUp | deve criar um usuário com sucesso | sucesso | deve retornar o usuário criado |
| 15 | auth.service.spec.ts | signUp | deve lançar erro ao falhar ao criar usuário | falha | deve retornar BadRequestException |
| 16 | auth.service.spec.ts | signIn | deve realizar login com sucesso | sucesso | deve retornar payload de sucesso no login |
| 17 | auth.service.spec.ts | signIn | deve lançar UnauthorizedException quando email não encontrado | falha | deve retornar UnauthorizedException |
| 18 | auth.service.spec.ts | signIn | 'eve lançar UnauthorizedException quando senha estiver incorreta | falha | deve retornar UnauthorizedException |

**BinanceApiController**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 19 | binanceapi.controller.spec.ts | BinanceapiController | deve instanciar o binanceapi.controller | sucesso | deve ser instanciado |
| 20 | binanceapi.controller.spec.ts | check | deve retornar statusCode 200 e success | sucesso | deve retornar no body Ping Sucesso |

**BinanceApiService**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 21 | binanceapi.service.spec.ts | BinanceapiService | deve instanciar o binanceapi.service | sucesso | deve ser instanciado |
| 22 | binanceapi.service.spec.ts | getAllOrders | deve retornar statusCode 200 e successdeve buscar todas as ordens com sucesso | sucesso | deve buscar ordens para BTCUSDT |
| 23 | binanceapi.service.spec.ts | getAllOrders | deve lançar exceção se der erro ao buscar ordens | Falha | deve retornar InternalServerErrorException |
| 24 | binanceapi.service.spec.ts | newOrder | deve executar uma ordem MARKET com sucesso | sucesso | deve retornar response a ordem mocada |
| 25 | binanceapi.service.spec.ts | newOrder | deve executar uma ordem LIMIT com sucesso | sucesso | deve retornar response a ordem mocada |
| 26 | binanceapi.service.spec.ts | newOrder | deve lançar BadRequestException se tipo de ordem for inválido | Falha | deve retornar InternalServerErrorException |
| 27 | binanceapi.service.spec.ts | newOrder | deve lançar InternalServerErrorException se houver erro na execução | Falha | deve retornar InternalServerErrorException |
| 28 | binanceapi.service.spec.ts | cancelOrder | deve cancelar uma ordem com sucesso | sucesso | deve retornar response a ordem mocada |
| 29 | binanceapi.service.spec.ts | cancelOrder | deve lançar exceção ao falhar no cancelamento da ordem | Falha | deve retornar no response: Erro ao cancelar ordem |
| 30 | binanceapi.service.spec.ts | cancelOpenOrders | deve cancelar todas as ordens abertas com sucesso | sucesso | deve retornar no response a ordem cancelada |
| 31 | binanceapi.service.spec.ts | cancelOpenOrders | deve lançar exceção ao falhar no cancelamento de ordens abertas | Falha | deve retornar no response: Erro ao cancelar ordem |
| 32 | binanceapi.service.spec.ts | getPriceMarket | deve retornar o preço de mercado com sucesso | sucesso | deve retornar o preco do ativo (28000.50) |
| 33 | binanceapi.service.spec.ts | getPriceMarket | deve lançar exceção ao falhar ao buscar preço de mercado | Falha | deve retornar Erro ao obter preço de mercado para BTCUSDT |
| 34 | binanceapi.service.spec.ts | checkOrder | deve verificar o status da ordem com sucesso | sucesso | deve retornar o mock de resposta da ordem |
| 35 | binanceapi.service.spec.ts | checkOrder | deve lançar exceção ao falhar ao verificar a ordem | Falha | deve retornar Erro ao verificar a ordem... |

**PrismaService**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 36 | prisma.service.spec.ts | PrismaService | deve instanciar o prisma.service | sucesso | deve ser instanciado |

**Engine-TharseoController**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 37 | engine-tharseo.controller.spec.ts | EngineTharseoController | deve instanciar o engine-tharseo.controller | sucesso | deve ser instanciado |
| 38 | engine-tharseo.controller.spec.ts | startEngine | deve chamar startEngineTharseo no controller | sucesso | deve retornar statuscode 200 na rota |
| 39 | engine-tharseo.controller.spec.ts | checkOrders | deve chamar checkOrders no controller | sucesso | deve retornar statuscode 200 na rota |

**Engine-TharseoService**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 40 | engine-tharseo.service.spec.ts | EngineTharseoService | deve instanciar o engine-tharseo.service | sucesso | deve ser instanciado |
| 41 | engine-tharseo.service.spec.ts | getOpenOrders | deve retornar as ordens de venda pendentes | sucesso | retornar ordens do mock |
| 42 | engine-tharseo.service.spec.ts | getActivesTradeflow | deve retornar os tradeflows ativos | sucesso | retonar tradeflows do mock |
| 43 | engine-tharseo.service.spec.ts | sendExchangeOrder | deve enviar a ordem para a Exchange | sucesso | retornar ordem enviada do mock |
| 44 | engine-tharseo.service.spec.ts | createOrder | deve criar uma nova ordem | sucesso | retornar ordem criado do mock |

**HealthController**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 45 | health.controller.spec.ts | HealthController | deve instanciar o health.controller | sucesso | deve ser instanciado |
| 46 | health.controller.spec.ts | check | deve retornar que o servidor está operacional | sucesso | deve retornar: Application is up and running |
| 47 | health.controller.spec.ts | check | deve retornar um InternalServerErrorException indicando que não está operacional | Falha | deve retornar InternalServerErrorException |

**OrderController**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 48 | order.controller.spec.ts | OrderController | deve instanciar o order.controller | sucesso | deve ser instanciado |
| 49 | order.controller.spec.ts | create | deve criar um pedido | sucesso | deve retornar o pedido criado |
| 50 | order.controller.spec.ts | create | deve lançar um NotFoundException se o usuario nao for encontrado | falha | deve retornar NotFoundException |
| 51 | order.controller.spec.ts | findAll | deve retornar uma lista de orders | sucesso | deve retornar as ordens do mock |
| 52 | order.controller.spec.ts | findOne | deve retornar um order | sucesso | deve retornar a ordem mocada |
| 53 | order.controller.spec.ts | findOne | deve retornar um NotFoundException se o order nao for encontrado | falha | deve retornar NotFoundException |
| 54 | order.controller.spec.ts | findOne | deve retornar um NotFoundException se o usuario nao for encontrado | falha | deve retornar NotFoundException |
| 55 | order.controller.spec.ts | update | deve atualizar uma ordem | sucesso | deve retornar a ordem atualizada do mock |
| 56 | order.controller.spec.ts | update | deve retornar um NotFoundException se o order nao for encontrado | falha | deve retornar NotFoundException |
| 57 | order.controller.spec.ts | update | deve retornar um NotFoundException se o usuario nao for encontrado | falha | deve retornar NotFoundException |
| 58 | order.controller.spec.ts | cancelOpenOrders | deve cancelar todas as ordens abertas | sucesso | deve retornar {success: true,  message: 'Ordens canceladas com sucesso',} |
| 59 | order.controller.spec.ts | cancelOpenOrders | deve retornar um NotFoundException se o usuario nao for encontrado' | falha | deve retornar NotFoundException |

**OrderService**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 60 | order.service.spec.ts | OrderService | deve instanciar o order.service | sucesso | deve ser instanciado |
| 61 | order.service.spec.ts | create | deve criar uma order com sucesso | sucesso | deve retornar a ordem do mock criada |
| 62 | order.service.spec.ts | create | deve lançar NotFoundException se asset não for encontrado | Falha | deve retornar NotFoundException |
| 63 | order.service.spec.ts | create | deve lançar NotFoundException se user não for encontrado | Falha | deve retornar NotFoundException |
| 64 | order.service.spec.ts | create | deve lançar ServiceUnavailableException se status da ordem não for FILLED | Falha | deve retornar ServiceUnavailableException |
| 65 | order.service.spec.ts | create | deve lançar erro se prisma.order.create falhar | Falha | deve retornar Erro ao salvar |
| 66 | order.service.spec.ts | createOnDatabase | deve criar uma order com sucesso | sucesso | deve retornar a ordem do mock criada |
| 67 | order.service.spec.ts | createOnDatabase | deve lançar NotFoundException se asset nao for encontrado | Falha | deve retornar NotFoundException |
| 68 | order.service.spec.ts | createOnDatabase | deve lançar NotFoundException se user nao for encontrado | Falha | deve retornar NotFoundException |
| 69 | order.service.spec.ts | findAll | deve retornar todas as orders de um usuario | sucesso | deve retornar as ordens do mock |
| 70 | order.service.spec.ts | findAll | deve retornar um array vazio se o usuario nao tiver orders | sucesso | deve retornar um array vazio |
| 71 | order.service.spec.ts | findAll | deve retornar NotFoundException se a order nao for encontrada | Falha | deve retornar NotFoundException |
| 72 | order.service.spec.ts | update | deve atualizar uma order | sucesso | deve retornar a ordem atualizada do mock |
| 73 | order.service.spec.ts | update | deve retornar NotFoundException se a order nao for encontrada | Falha | deve retornar NotFoundException |
| 74 | order.service.spec.ts | updateIdPairOrder | deve atualizar uma order com atribuição de pairOrderId | sucesso | deve retornar a ordem atualizada do mock |
| 75 | order.service.spec.ts | getPendingOrdersCreated | deve retornar uma lista de orders pendentes | sucesso | deve retornar as ordens do mock |
| 76 | order.service.spec.ts | cancelOpenOrders | deve cancelar ordens abertas do usuário e atualizar o status para CANCELADO | sucesso | deve retornar as ordens atualizadas com status CANCELADO do mock |
| 77 | order.service.spec.ts | updateOrderFromCheckExchange | deve atualizar a ordem e a estratégia se order tiver result, strategyId e performance | sucesso | deve retornar a ordem atualizada do mock |

**StrategyController**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 78 | strategy.controller.spec.ts | StrategyController | deve instanciar o strategy.controller | sucesso | deve ser instanciado |
| 79 | strategy.controller.spec.ts | create | deve criar uma strategy | sucesso | deve retornar a strategy mocada |
| 80 | strategy.controller.spec.ts | create | deve retornar um erro se o usuario nao for encontrado | Falha | deve retornar NotFoundException |
| 81 | strategy.controller.spec.ts | findAll | deve retornar todas as strategies | sucesso | deve retornar as strategys do mock |
| 82 | strategy.controller.spec.ts | findAll | deve retornar um erro se o usuario nao for encontrado | Falha | deve retornar NotFoundException |
| 83 | strategy.controller.spec.ts | findOne | deve retornar uma strategy | sucesso | deve retornar a strategy do mock |
| 84 | strategy.controller.spec.ts | findOne | deve retornar um NotFoundException se a strategy nao for encontrada | Falha | deve retornar NotFoundException |
| 85 | strategy.controller.spec.ts | findOne | deve retornar um erro se o usuario nao for encontrado | Falha | deve retornar NotFoundException |
| 86 | strategy.controller.spec.ts | update | deve atualizar uma strategy | sucesso | deve retornar a strategy atualizada do mock |
| 87 | strategy.controller.spec.ts | update | deve retornar um NotFoundException se a strategy nao for encontrada | Falha | deve retornar NotFoundException |
| 88 | strategy.controller.spec.ts | update | deve retornar um erro se o usuario nao for encontrado | Falha | deve retornar NotFoundException |

**StrategyService**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 89 | strategy.service.spec.ts | StrategyService | deve instanciar o strategy.service | sucesso | deve ser instanciado |
| 90 | strategy.service.spec.ts | create | deve criar uma nova strategy com sucesso | sucesso | deve retornar a strategy mocada |
| 91 | strategy.service.spec.ts | create | deve lançar um erro se o usuário não for encontrado | Falha | deve retornar NotFoundException |
| 92 | strategy.service.spec.ts | findAll | deve retornar todas as strategies do usuario que requisitou | sucesso | deve retornar as strategies mocadas |
| 93 | strategy.service.spec.ts | findAll | deve retornar NotFoundException se o usuário nao for encontrado | Falha | deve retornar NotFoundException |
| 94 | strategy.service.spec.ts | findOne | deve retornar uma strategy pelo id e pelo usuário | sucesso | deve retornar a strategy do |
| 95 | strategy.service.spec.ts | findOne | deve retornar NotFoundException se o usuario nao for encontrado | Falha | deve retornar NotFoundException |
| 96 | strategy.service.spec.ts | findOne | deve retornar NotFoundException se a strategy nao for encontrada | Falha | deve retornar NotFoundException |
| 97 | strategy.service.spec.ts | update | deve atualizar uma strategy pelo id e pelo usuário | sucesso | deve retornar a strategy atualizada do mock |
| 98 | strategy.service.spec.ts | update | deve retornar NotFoundException se o usuario nao for encontrado | Falha | deve retornar NotFoundException |
| 99 | strategy.service.spec.ts | update | deve retornar NotFoundException se a strategy nao for encontrada | Falha | deve retornar NotFoundException |

**AssetController**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 100 | asset.controller.spec.ts | AssetController | deve instanciar o asset.controller | sucesso | deve ser instanciado |
| 101 | asset.controller.spec.ts | create | deve criar um novo ativo | sucesso | valor retornado deve ser igual ao esperado |
| 102 | asset.controller.spec.ts | findAll | deve retornar todos os ativos | sucesso | retornar lista de ativos mocados |
| 103 | asset.controller.spec.ts | findOne | deve retornar o ativo pelo ID | sucesso | retornar asset igual ao mocado |
| 104 | asset.controller.spec.ts | findOne | deve lançar exceção se o ativo não existir | Falha | retornar NotFoundException |
| 105 | asset.controller.spec.ts | update | deve atualizar o ativo se ele existir | sucesso | retornar asset atualizado conforme mock |
| 106 | asset.controller.spec.ts | update | deve lançar exceção se o ativo não existir | Falha | retornar NotFoundException |
| 107 | asset.controller.spec.ts | remove | deve remover o ativo se ele existir | sucesso | retornar asset igual ao mocado |
| 108 | asset.controller.spec.ts | remove | deve lançar exceção se o ativo não existir | Falha | retornar NotFoundException |

**AssetService**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 109 | asset.service.spec.ts | Instancia | deve instanciar o asset.service | sucesso | deve ser instanciado |
| 110 | asset.service.spec.ts | create | deve criar e retornar um novo ativo | sucesso | deve criar um asset |
| 111 | asset.service.spec.ts | findAll | deve retornar uma lista de ativos | sucesso | deve retornar uma lista de assets |
| 112 | asset.service.spec.ts | findOne | deve retornar um ativo pelo id | sucesso | deve retornar o asset |
| 13 | asset.service.spec.ts | findOne | deve retornar null se o ativo não for encontrado | Falha | deve retornar null |
| 114 | asset.service.spec.ts | update | deve atualizar e retornar o ativo | sucesso | deve retornar o asset atualizado |
| 115 | asset.service.spec.ts | remove | deve remover e retornar o ativo removido | sucesso | deve retornar o asset removido |

**TradeflowController**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 116 | tradeflow.controller.spec.ts | TradeflowController | deve instanciar o tradeflow.controller | sucesso | deve ser instanciado |
| 117 | tradeflow.controller.spec.ts | Create | deve criar um novo TradeFlow com sucesso | sucesso | deve retornar o tradeflow do mock |
| 118 | tradeflow.controller.spec.ts | findAll | deve retornar uma lista de tradeflows | sucesso | deve retornar uma lista de tradeflows |
| 119 | tradeflow.controller.spec.ts | findOne | deve retornar um tradeflow | sucesso | deve retornar o tradeflow do mock |
| 120 | tradeflow.controller.spec.ts | findOne | deve retornar uma exception se o tradeflow nao for encontrado | Falha | deve retornar NotFoundException |
| 121 | tradeflow.controller.spec.ts | update | deve atualizar um tradeflow | sucesso | deve retornar o tradeflow atualizado do mock |
| 122 | tradeflow.controller.spec.ts | update | deve retornar uma exception se o tradeflow nao for encontrado | Falha | deve retornar NotFoundException |
| 123 | tradeflow.controller.spec.ts | remove | deve remover um tradeflow com sucesso | sucesso | deve retornar o tradeflow excluido |
| 124 | tradeflow.controller.spec.ts | remove | deve lançar NotFoundException se o tradeflow não for encontrado | Falha | deve retornar NotFoundException |
| 125 | tradeflow.controller.spec.ts | remove | deve lançar UnauthorizedException se o usuário não for o dono do tradeflow | sucesso | deve retornar UnauthorizedException |

**TradeflowService**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 126 | tradeflow.service.spec.ts | TradeflowService | deve instanciar o tradeflow.service | sucesso | deve ser instanciado |
| 127 | tradeflow.service.spec.ts | Create | deve criar um novo tradeflow com sucesso | sucesso | deve retornar o tradeflow criado do mock |
| 128 | tradeflow.service.spec.ts | Create | deve lançar NotFoundException se a estratégia não for encontrada | Falha | deve retornar NotFoundException |
| 129 | tradeflow.service.spec.ts | Create | deve lançar NotFoundException se o ativo não for encontrado | Falha | deve retornar NotFoundException |
| 130 | tradeflow.service.spec.ts | findAll | deve retornar todos os tradeflows do usuário | sucesso | deve retornar os tradeflows do mock |
| 131 | tradeflow.service.spec.ts | findOne | deve retornar o tradeflow se encontrado | sucesso | deve retornar o tradeflow do mock |
| 132 | tradeflow.service.spec.ts | findOne | deve lançar NotFoundException se o tradeflow não for encontrado | Falha | deve retornar NotFoundException |
| 133 | tradeflow.service.spec.ts | update | deve atualizar o tradeflow se existir e pertencer ao usuário | sucesso | deve retornar o tradeflow atualizado do mock |
| 134 | tradeflow.service.spec.ts | update | deve lançar NotFoundException se o tradeflow não existir | Falha | deve retornar NotFoundException |
| 135 | tradeflow.service.spec.ts | update | deve lançar UnauthorizedException se o tradeflow não pertencer ao usuário | Falha | deve retornar UnauthorizedException |
| 136 | tradeflow.service.spec.ts | remove | deve remover o tradeflow se existir e pertencer ao usuário | sucesso | deve retornar o tradeflow removido do mock |
| 137 | tradeflow.service.spec.ts | remove | deve lançar NotFoundException se o tradeflow não existir | Falha | deve retornar NotFoundException |
| 138 | tradeflow.service.spec.ts | remove | deve lançar UnauthorizedException se o tradeflow não pertencer ao usuário | Falha | deve retornar UnauthorizedException |

**UserController**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 139 | user.controller.spec.ts | UserController | deve instanciar o user.controller | sucesso | deve ser instanciado |
| 140 | user.controller.spec.ts | getAllUsers | deve retornar dois usuários | sucesso | deve retornar dois usuarios do mock |

**UserService**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 141 | user.service.spec.ts | UserService | deve instanciar o user.service | sucesso | deve ser instanciado |
| 142 | .service.spec.ts |  | deve criar um novo usuário e retornar os dados sem a senha | sucesso | deve retornar o usuário criado |
| 143 | user.service.spec.ts | getUserByEmail | deve retornar um usuário se encontrado por email | sucesso | deve retornar o usuário do mock |
| 144 | user.service.spec.ts | getUserByEmail | deve retornar null se o usuário não for encontrado | Falha | deve retornar null |
| 145 | user.service.spec.ts | getAllUsers | deve retornar uma lista de usuários | sucesso | deve retornar um array de usuários do mock |
| 146 | user.service.spec.ts | getUserById | deve retornar um usuário com wallets e credential se encontrado por ID | sucesso | deve retornar o usuário do mock |
| 147 | user.service.spec.ts | getUserById | deve retornar null se o usuário não for encontrado | Falha | deve retornar null |

**WalletController**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 148 | wallet.controller.spec.ts | WalletController | deve instanciar o wallet.controller | sucesso | deve ser instanciado |
| 149 | wallet.controller.spec.ts | createOrUpdate | deve chamar walletService.createOrUpdate com os parâmetros corretos | sucesso | deve retornar a wallet criada do mock |
| 150 | wallet.controller.spec.ts | findAll | deve retornar uma lista de carteiras do usuário | sucesso | deve retornar uma lista de wallets do mock |
| 151 | wallet.controller.spec.ts | findOne | deve retornar uma carteira específica do usuário pelo ID | sucesso | deve retornar a wallet mocada |
| 152 | wallet.controller.spec.ts | update | deve chamar walletService.update com os parâmetros corretos | sucesso | deve retornar a wallet atualizada do mock |
| 153 | wallet.controller.spec.ts | remove | deve chamar walletService.remove com o ID correto | sucesso | deve retornar a wallet removida mocada |

**WalletService**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Método/Função** | **Objetivo** | **Caso** | **Resultado** |
| 154 | wallet.service.spec.ts | WalletService | deve instanciar o wallet.service | sucesso | deve ser instanciado |
| 155 | wallet.service.spec.ts | createOrUpdate | deve criar uma nova wallet se não existir | sucesso | deve retornar uma wallet criada pelo mock |
| 156 | wallet.service.spec.ts | createOrUpdate | deve atualizar uma carteira existente | sucesso | deve atualizar uma wallet existente com o mock |
| 157 | wallet.service.spec.ts | createOrUpdate | deve lançar uma excessão se um usuário não for encontrado | falha | deve retornar NotFoundException |
| 158 | wallet.service.spec.ts | createOrUpdate | deve lançar uma excessão se um ativo nao for encontrado | falha | deve retornar NotFoundException |
| 159 | wallet.service.spec.ts | updateWallet | deve atualizar e retornar uma wallet | sucesso | deve retornar uma wallet atualizada com o mock |
| 160 | wallet.service.spec.ts | createWallet | deve criar e retornar uma nova wallet | sucesso | deve criar uma wallet mocada |
| 161 | wallet.service.spec.ts | findAll | deve retornar todas as carteiras do usuário | sucesso | deve retornar uma lista de wallets mocada |
| 162 | wallet.service.spec.ts | findOne | deve retornar se existir, uma carteira do usuário | sucesso | deve retornar a wallet mocada de um usuário |
| 163 | wallet.service.spec.ts | findOne | deve lançar uma excessão se a wallet não pertencer ao usuário | falha | deve retornar NotFoundException |

**Execução e Descrição dos Testes de Integração (E2E)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Núm.** | **Classe** | **Objetivo** | **Fluxos** | **Caso** | **Resultado** |
| 1 | App.e2e-spec.ts | Verificar se a aplicação inicia e fica disponível sem erros | GET /health/check  → Aciona: check – HealthController | sucesso | deve receber na rota status code 200 e a mensagem: Application is up and running |
| 2 | Auth.e2e-spec.ts | Registrar um novo usuário | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – userService  → Aciona: create – prismaService  → Registra: User – PostgreSQL | sucesso | deve registrar um usuário e receber de volta o registro com o payload processado |
| 3 | Auth.e2e-spec.ts | Registrar um novo usuário com e-mail já existente | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – userService | falha | deve receber status code 400 com a mensagem Erro ao cadastrar usuário |
| 4 | Auth.e2e-spec.ts | Fazer Login na Aplicação com Credenciais Corretas | POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – userService  → Aciona: findFirst – prismaService  → Aciona: compare – bcrypt | sucesso | deve receber um payload com dados principais, token e expiração do token |
| 5 | Auth.e2e-spec.ts | Fazer Login na Aplicação com Credenciais Incorretas | POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – userService  → Aciona: findFirst – prismaService  → Busca: User – PostgreSQL  → Aciona: compare – bcrypt | falha | deve receber Unauthorized com um JSON com campos: message, error e statusCode |
| 6 | Strategy.e2e-spec.ts | Criar uma nova estratégia com Sucesso | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/strategy  → Aciona: create – StrategyController  → Aciona: create – StrategyService  → Aciona: getUserById – UserService  → Aciona: create – PrismaService  → Cria: Strategy – PostgreSQL | sucesso | deve receber um payload com detalhes da estratégia criada |
| 7 | Strategy.e2e-spec.ts | Criar uma nova estratégia sem sucesso | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/strategy  → Aciona: create – StrategyController | falha | deve receber um status code 400 com payload com propriedades: message, error e statusCode com a message = ‘Bad Request’ |
| 8 | Strategy.e2e-spec.ts | Consultar todas estratégias do usuário com sucesso | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/strategy  → Aciona: create – StrategyController  → Aciona: create – StrategyService  → Aciona: getUserById – UserService  → Aciona: create – PrismaService  → Cria: Strategy – PostgreSQL  GET /v1/strategy  → Aciona: findAll – StrategyController  → Aciona: findAll – StrategyService  → Aciona: getUserById – UserService  → Aciona: findMany – PrismaService  → Busca: Strategy – PostgreSQL | sucesso | deve receber a lista de estratégias do usuário ou uma lista vazia |
| 9 | Strategy.e2e-spec.ts | Consultar uma estratégia do usuário pelo Id com sucesso | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/strategy  → Aciona: create – StrategyController  → Aciona: create – StrategyService  → Aciona: getUserById – UserService  → Aciona: create – PrismaService  → Cria: Strategy – PostgreSQL  GET /v1/strategy/{id}  → Aciona: findOne – StrategyController  → Aciona: findOne – StrategyService  → Aciona: getUserById – UserService  → Aciona: findUnique – PrismaService  → Busca: Strategy – PostgreSQL | sucesso | deve receber a estratégia que foi consultada pelo id |
| 10 | Strategy.e2e-spec.ts | Deve atualizar uma estratégia pelo id e body com sucesso | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/strategy  → Aciona: create – StrategyController  → Aciona: create – StrategyService  → Aciona: getUserById – UserService  → Aciona: create – PrismaService  → Cria: Strategy – PostgreSQL  PATCH /v1/strategy/{id}  → Aciona: update – StrategyController  → Aciona: update – StrategyService  → Aciona: findUnique – PrismaService  → Aciona: update – PrismaService  → Busca: Strategy – PostgreSQL | sucesso | deve receber a estratégia que foi consultada pelo id |
| 11 | Tradeflow.e2e-spec.ts | Deve criar uma nova automação com sucesso | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/tradeflow  → Aciona: create – TradeflowController  → Aciona: create – TradeflowService  → Aciona: findOne – StrategyService  → Aciona: getUserById – UserService  → Aciona: findUnique – PrismaService  → Busca: Strategy – PostgreSQL  → Aciona: findOne – AssetService  → Aciona: findUnique – PrismaService  → Busca: Asset – PostgreSQL  → Aciona: create – PrismaService  → Cria: Tradeflow – PostgreSQL | sucesso | deve receber a automação criada no payload da requisição |
| 12 | Tradeflow.e2e-spec.ts | Deve criar uma nova automação sem sucesso – faltando o token | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/tradeflow  → Aciona: create – TradeflowController | falha | deve receber UnauthorizedException como resposta contendo no payload message, error e statusCode |
| 13 | Tradeflow.e2e-spec.ts | Deve criar uma nova automação sem sucesso – Asset não encontrado | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/tradeflow  → Aciona: create – TradeflowController  → Aciona: create – TradeflowService  → Aciona: findOne – StrategyService  → Aciona: getUserById – UserService  → Aciona: findUnique – PrismaService  → Busca: Strategy – PostgreSQL  → Aciona: findOne – AssetService  → Aciona: findUnique – PrismaService  → Busca: Asset – PostgreSQL | falha | deve receber NotFoundException como resposta contendo no payload message, error e statusCode |
| 14 | Tradeflow.e2e-spec.ts | Deve criar uma nova automação sem sucesso – Strategy não encontrado | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/tradeflow  → Aciona: create – TradeflowController  → Aciona: create – TradeflowService  → Aciona: findOne – StrategyService  → Aciona: getUserById – UserService  → Aciona: findUnique – PrismaService  → Busca: Strategy – PostgreSQL | falha | deve receber NotFoundException como resposta contendo no payload message, error e statusCode |
| 15 | Tradeflow.e2e-spec.ts | Deve criar uma nova automação sem sucesso – Strategy não encontrado | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/tradeflow  → Aciona: create – TradeflowController  → Aciona: create – TradeflowService  → Aciona: findOne – StrategyService  → Aciona: getUserById – UserService  → Aciona: findUnique – PrismaService  → Busca: Strategy – PostgreSQL | falha | deve receber NotFoundException como resposta contendo no payload message, error e statusCode |
| 16 | Tradeflow.e2e-spec.ts | Deve retornar uma lista de Tradeflows de um usuário com sucesso | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/tradeflow  → Aciona: create – TradeflowController  → Aciona: create – TradeflowService  → Aciona: findOne – StrategyService  → Aciona: getUserById – UserService  → Aciona: findUnique – PrismaService  → Busca: Strategy – PostgreSQL  → Aciona: findOne – AssetService  → Aciona: findUnique – PrismaService  → Busca: Asset – PostgreSQL  → Aciona: create – PrismaService  → Cria: Tradeflow – PostgreSQL  GET /v1/tradeflow  → Aciona: findAll – TradeflowController  → Aciona: findAll – TradeflowService  → Aciona: getUserById – UserService  → Aciona: findMany – PrismaService  → Busca: Tradeflow – PostgreSQL | sucesso | deve receber uma lista de tradeflows |
| 17 | Tradeflow.e2e-spec.ts | Deve retornar uma lista vazia para um usuário sem tradeflows com sucesso | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  GET /v1/tradeflow  → Aciona: findAll – TradeflowController  → Aciona: findAll – TradeflowService  → Aciona: getUserById – UserService  → Aciona: findMany – PrismaService  → Busca: Tradeflow – PostgreSQL | sucesso | deve receber uma lista vazia |
| 18 | Tradeflow.e2e-spec.ts | Deve retornar um tradeflow de um usuário pesquisado pelo id com sucesso | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  POST /v1/tradeflow  → Aciona: create – TradeflowController  → Aciona: create – TradeflowService  → Aciona: findOne – StrategyService  → Aciona: getUserById – UserService  → Aciona: findUnique – PrismaService  → Busca: Strategy – PostgreSQL  → Aciona: findOne – AssetService  → Aciona: findUnique – PrismaService  → Busca: Asset – PostgreSQL  → Aciona: create – PrismaService  → Cria: Tradeflow – PostgreSQL  GET /v1/tradeflow/{id}  → Aciona: findOne – TradeflowController  → Aciona: findOne – TradeflowService  → Aciona: getUserById – UserService  → Aciona: findUnique – PrismaService  → Busca: Tradeflow – PostgreSQL | sucesso | deve receber o tradeflow referente ao id |
| 19 | Tradeflow.e2e-spec.ts | Deve retornar NotFoundException para um tradeflow não existente | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  GET /v1/tradeflow/{id}  → Aciona: findOne – TradeflowController  → Aciona: findOne – TradeflowService  → Aciona: getUserById – UserService  → Aciona: findUnique – PrismaService  → Busca: Tradeflow – PostgreSQL | falha | deve receber NotFoundException para o id inexistente |
| 20 | Tradeflow.e2e-spec.ts | Deve retornar UnauthorizedException para um tradeflow não existente | POST /auth/signup  → Aciona: signUp – AuthController  → Aciona: signUp – AuthService  → Aciona: createUser – UserService  → Aciona: create – PrismaService  → Registra: User – PostgreSQL  POST /auth/signin  → Aciona: signIn – AuthController  → Aciona: getUserByEmail – UserService  → Aciona: findFirst – PrismaService  → Busca: User – PostgreSQL  → Aciona: compare – Bcrypt  GET /v1/tradeflow/{id}  → Aciona: findOne – TradeflowController | falha | deve receber UnauthorizedException para o id inexistente |

### Plano de Testes Tharseo Frontend v2

**Introdução**

Utilizamos da tecnologia Cypress para realizar o teste do frontend nos principais fluxos utilizados pelos usuários como cadastro, login, navegação por diversas partes funcionais da aplicação etc. a partir do algoritmo o Cypress automatizou clicando e preenchendo dados nos campos e botões definidos no algoritmo, gerando automação e facilidade nos testes, de modo que ao ser modificada a aplicação gera somente a necessidade de nova execução do algoritmo sem a necessidade de teste manual do QA, garantindo que as modificações feitas em atualizações não irão impactar no que já estava funcional.

**Métodos dos Testes**

O Cypress é uma ferramenta moderna de testes end-to-end (E2E) voltada para aplicações web. Ele permite simular interações reais de usuários com a interface da aplicação, garantindo que os fluxos funcionem corretamente. Diferente de outras ferramentas, o Cypress roda diretamente no navegador, junto com a aplicação, o que proporciona acesso completo ao DOM, console e rede, permitindo testes mais precisos e rápidos.

Os testes são escritos em JavaScript ou TypeScript e seguem uma estrutura clara e encadeada. Um teste básico geralmente envolve visitar uma URL, interagir com elementos da página (como campos de texto e botões) e realizar asserções para verificar se o comportamento esperado foi alcançado. Por exemplo, um teste de login pode incluir comandos como cy.visit() para acessar a página, cy.get() para selecionar elementos, cy.type() para preencher campos e cy.click() para acionar botões. As verificações são feitas com cy.should(), que valida o estado da aplicação após as interações.

Uma das principais vantagens do Cypress é sua capacidade de esperar automaticamente por elementos estarem disponíveis antes de interagir com eles, eliminando a necessidade de comandos manuais de espera. Além disso, a ferramenta oferece uma interface visual que mostra cada etapa do teste em tempo real, facilitando a depuração e o entendimento do fluxo de execução.

Em resumo, o método de testes do Cypress é baseado em comandos encadeados, observação do DOM e espera inteligente, proporcionando uma abordagem confiável, legível e eficiente para garantir a qualidade de aplicações web.

**Estratégia de Validação**

Optamos pelo teste dos principais fluxos e funcionalidades da nossa aplicação garantindo a funcionalidade nos processos de cadastro, login, e das principais funcionalidades que já estão funcionais na nossa aplicação, segue abaixo o plano de testes realizados através do Cypress:

|  |  |  |  |
| --- | --- | --- | --- |
| **Cenário** | **Caso de teste** | **BDD** | **Status** |
| 001-Cadastro | CT001.001 - Cadastro com e-mail inválido | Dado que eu esteja na Tela de cadastro Quando preencher o campo e-mail com um e-mail invalido sem @ Então o sistema deve apontar o erro de e-mail inválido | Passou |
| CT001.002 - Cadastro com e-mail válido | Dado que eu esteja na Tela de cadastro Quando preencher o campo e-mail com um e-mail valido sem @ Então o sistema deve deixar eu dar continuidade no processo de cadastro | Passou |
| CT001.003- Cadastro um senha que não atende os requisitos de complexidade | Dado que eu esteja na Tela de cadastro Quando preencher o campo de senha com determinada senha que não atenda os requisitos determinados  Então o sistema deve apontar o erro de senha inválida | Passou |
| CT001.004 - Cadastro um senha que atende os requisitos de complexidade | Dado que eu esteja na Tela de cadastro Quando preencher o campo de senha com determinada senha atenda os requisitos determinados  Então o sistema deve permitir que prossigamos com o processo de cadastro | Passou |
| CT001.005 - Cadastro de um número celular invalido | Dado que eu esteja na Tela de cadastro Quando preencher o campo de numero de celular invalido Então o sistema deve acusar número invalido | Passou |
| CT001.006 - Cadastro de um número celular valido | Dado que eu esteja na Tela de cadastro Quando preencher o campo de numero de celular invalido Então o sistema deve permitir que sigamos com o processo de cadastro | Passou |
| CT001.007 - Teste de cadastro com um e-mail que já foi cadastrado | Dado que eu esteja na Tela de cadastro Quando for finalizar o cadastro utilizando um e-mail já cadastrado Então o sistema deve apontar erro do processo de cadastro | Passou |
| 002-Login | CT002.001 - login com e-mail inválido | Dado que eu esteja na tela de login Quando preencher usuário com e-mail sem @ Então deve apontar a mensagem de erro no processo de Login | Passou |
| CT002.002 - login com e-mail válido | Dado que eu esteja na tela de login Quando preencher usuário com e-mail válido Então deve seguir com o processo de Login | Passou |
| CT002.003 - login com senha inválida | Dado que eu esteja na tela de login Quando o campo senha com uma senha inválida Então o sistema deve apontar senha incorreta | Passou |
| CT002.004 - login com senha válida | Dado que eu esteja na tela de login Quando o campo senha com uma senha válida Então o sistema deve permitir a continuidade de login com um e-mail e senha respectivamente corretos | Passou |
| 003-Navegação | CT003.001 - Fluxo de Navegação Carteira | Dado que eu esteja na tela de navegação da carteira Quando ao clicar no botão colunas e nos seus sub botões flegando ou desflegando Então o sistema deve permitir a variação do número de colunas da aplicação | Passou |
| CT003.002 - Fluxo de Navegação Estratégias | Dado que eu esteja na tela de navegação das estratégias Quando ao clicar no botão colunas e nos seus sub botões flegando ou desflegando Então o sistema deve permitir a variação do número de colunas da aplicação | Passou |
| Dado que eu esteja na tela de navegação das estratégias Quando ao clicar no botão adicionar estratégias o sistema deve permitir o preenchimento do formulário para cadastro de uma nova estratégia | Passou |
| CT003.003 - Fluxo de Navegação Trade | Dado que eu esteja na tela de navegação de trade Quando ao clicar no botão colunas e nos seus sub botões flegando ou desflegando Então o sistema deve permitir a variação do número de colunas da aplicação | Passou |
| Dado que eu esteja na tela de navegação de trade Quando ao clicar no botão Guia Rápido Então ele não deve funcionar por ainda não ter sido implementado | Bloqueado |
| Dado que eu esteja na tela de navegação de trade Quando ao clicar no botão Automações Então ele não deve funcionar por ainda não ter sido implementado | Bloqueado |
| Dado que eu esteja na tela de navegação de trade Quando ao clicar no botão Backtesting Então ele não deve funcionar por ainda não ter sido implementado | Bloqueado |
| Dado que eu esteja na tela de navegação de trade Quando ao clicar no botão Adicionar Automação Então ele não deve funcionar por ainda não ter sido implementado | Bloqueado |
| CT003.004 - Fluxo de Navegação Em andamento | Dado que eu esteja na tela de navegação do Em andamento Quando ao clicar no botão colunas e nos seus sub botões flegando ou desflegando Então o sistema deve permitir a variação do número de colunas da aplicação | Passou |
| CT003.005 - Fluxo de Navegação Histórico | Dado que eu esteja na tela de navegação de histórico Quando ao clicar no botão colunas e nos seus sub botões flegando ou desflegando Então o sistema deve permitir a variação do número de colunas da aplicação | Passou |

Códigos utilizados disponíveis no GIT: <https://github.com/RichardGuedesRib/tharseo_frontend_v2>, para execução do teste abrir o código no VS code e executar o comando npm start para instalar as dependências necessárias para rodar o Cypress, e após esse comando e a instalação das extensões necessárias utilizar o comando npx cypress open.

Vídeos evidenciando os testes (não anexado na tarefa devido ao tamanho do arquivo, então optamos por hospedar o vídeo no youtube e disponibilizar o link para acesso):

Teste de cadastro:

<https://youtu.be/nuhJqyoR0t0?si=7Voa40hMQuhY0EMl>

Teste de cadastro com e-mail já cadastrado:

<https://youtu.be/XSOwZFuERwY?si=2pw8wKCWDlvyGy61>

Teste de login:

<https://youtu.be/580YBzm9jUs?si=JXt27hG7mWkHEY49>

Teste de navegação Carteira:

<https://youtu.be/R_AblGRLfl8?si=sfAA8pHm1Jy8w-SW>

Teste de navegação Estratégias:

<https://youtu.be/13bQGqg8GyM?si=-uAuiesRk5njdJjK>

Teste de navegação Trade:

<https://youtu.be/L3z0P_y3n-8?si=qKsPqFRzoG2mNvPO>

Teste de navegação Em Andamento:

<https://youtu.be/PbhaihYdSXc?si=_i_1qDEiheCPxVKl>

Teste de navegação Histórico:

<https://youtu.be/CPkQPRwUZOg?si=0YzIl6I0xh81dcSv>

## Aspectos de segurança do Projeto

A seguir apresentamos a análise de segurança da aplicação web Tharseo (https://tharseocripto.web.app/) baseada em dois escaneamentos DAST realizados com OWASP ZAP, sendo o primeiro do dia 19/06 e um segundo do dia 21/06, demonstrando melhorias significativas na postura de segurança da aplicação.

**Conceitos de Segurança de Software Aplicados**

**DAST (Dynamic Application Security Testing)**

Conceito: Análise de segurança em tempo de execução que testa a aplicação do ponto de vista externo, simulando ataques reais.

Aplicação no Projeto: - Utilização do OWASP ZAP para scanner automatizado - Testes realizados contra a aplicação em produção - Detecção de problemas de configuração de segurança.

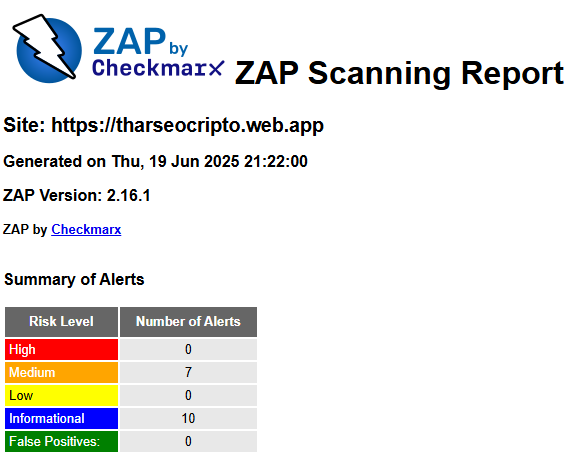
**SAST vs DAST**

SAST (não aplicado neste caso): Análise estática do código fonte DAST (aplicado): Análise dinâmica da aplicação em execução.

O projeto utilizou exclusivamente DAST, que é apropriado para: - Validar configurações de produção - Detectar vulnerabilidades de infraestrutura - Testar controles de segurança implementados - Verificar políticas CSP e headers de segurança.

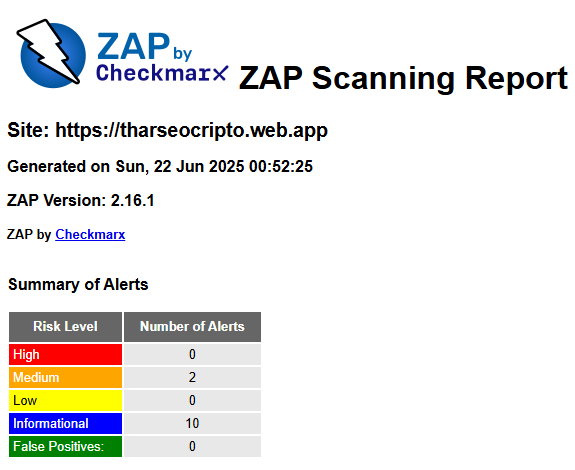
**Análise Comparativa dos Escaneamentos**

**Primeiro Scan (19/06)**



**Vulnerabilidades Críticas:** 7 Medium, 0 High - CSP: Failure to Define Directive (3 instâncias) - CSP: Wildcard Directive (3 instâncias) - CSP: script-src unsafe-inline (3 instâncias) - CSP: style-src unsafe-inline (3 instâncias) - Hidden File Found (2 instâncias) - Proxy Disclosure (8 instâncias) - Sub Resource Integrity Missing (9 instâncias).

**Segundo Scan (22/06)**

  
**Vulnerabilidades Críticas:** 2 Medium, 0 High - Proxy Disclosure (9 instâncias) - Sub Resource Integrity Missing (3 instâncias).

**Melhoria Mensurável**

Redução de 71% nas vulnerabilidades Medium (de 7 para 2 tipos) Redução de 50% nas instâncias críticas (de ~30 para ~12 instâncias).

**Evidências das Vulnerabilidades**

**Vulnerabilidades Corrigidas**

Content Security Policy (CSP) - Corrigido.

**Evidência Inicial (19/06):**

Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline'

https://cdnjs.cloudflare.com https://www.googletagmanager.com   
https://www.google-analytics.com; style-src 'self' 'unsafe-inline'   
https://fonts.googleapis.com https://cdnjs.cloudflare.com; [...]

**Problemas Identificados:** - form-action directive ausente (CWE-693) - img-src com wildcard https: (CWE-693) - script-src e style-src com unsafe-inline (CWE-693)

**Status no Segundo Scan:** Todas as vulnerabilidades CSP foram eliminadas

**Hidden Files - Corrigido**

**Evidência Inicial:** - https://tharseocripto.web.app/.\_darcs (200 OK) - https://tharseocripto.web.app/BitKeeper (200 OK)

**Risco:** Exposição de arquivos de controle de versão (CWE-538) Status: Arquivos não mais acessíveis no segundo escaneamento.

**3.2 Vulnerabilidades Persistentes**

**Proxy Disclosure**

**Descrição:** Exposição de informações do servidor proxy Varnish CWE: 204 (Information Exposure) WASC: 45 Instâncias: 8→9 (leve aumento devido ao novo arquivo security.js)

**Sub Resource Integrity – Melhorado Parcialmente**

Redução: 9→3 instâncias Problema Persistente: Links para Google Fonts sem atributo integrity

<link href="https://fonts.googleapis.com/css2?family=Poppins..." rel="stylesheet" crossorigin="anonymous" referrerpolicy="no-referrer">

CWE: 345 (Insufficient Verification of Data Authenticity)

**Correções Implementadas**

**Content Security Policy**

**Implementação:** Política CSP robusta corrigindo: - Adição da diretiva form-action - Remoção de wildcards desnecessários - Eliminação de unsafe-inline onde possível.

**Proteção Contra Clickjacking**

**Evidência da Correção:** Criação do arquivo security.js

// 🛡️ PROTEÇÃO CONTRA CLICKJACKING   
(function () {   
 'use strict';   
   
 // Impede que o site seja carregado em iframe malicioso   
 if (window.top !== window.self) {   
 window.top.location = window.self.location;   
 }   
   
 // Proteção adicional contra frame busting   
 try {   
 if (window.parent && window.parent !== window) {   
 window.parent.location = window.location;   
 }   
 } catch (e) {   
 console.log('Frame busting protection active');   
 }   
})();

**Limpeza de Arquivos Sensíveis**

**Ação:** Remoção/bloqueio de arquivos de controle de versão (.darcs, BitKeeper) Resultado: Eliminação completa da vulnerabilidade "Hidden File Found".

**Conclusão**

A análise demonstra uma evolução significativa na postura de segurança da aplicação Tharseo entre os dois escaneamentos, com:

* 71% de redução nas vulnerabilidades de nível médio;
* Eliminação completa das vulnerabilidades CSP críticas;
* Implementação proativa de proteções contra clickjacking;
* Higienização do ambiente de produção.

O projeto demonstra maturidade em segurança defensiva e aplicação efetiva de práticas DAST para identificação e correção de vulnerabilidades em ambiente de produção.

## Apropriação de Horas no Projeto

|  |  |  |
| --- | --- | --- |
| **Histórico de apropriação de horas** | | |
| **Data do registro** | **Atividade** | **Quantidade de horas** |
| 23/02/2025 | Atividade 1 do cronograma | 20 horas |
| 01/03/2025 | Atividade 2 do cronograma | 23 horas |
| 07/03/2025 | Atividade 3 do cronograma | 18 horas |
| 13/03/2025 | Atividade 4 do cronograma | 20 horas |
| 19/03/2025 | Atividade 5 do cronograma | 15 horas |
| 25/03/2025 | Atividade 6 do cronograma | 25 horas |
| 01/04/2025 | Atividade 7 do cronograma | 15 horas |
| 07/04/2025 | Atividade 8 do cronograma | 18 horas |
| 13/04/2025 | Atividade 9 do cronograma | 18 horas |
| 19/04/2025 | Atividade 10 do cronograma | 15 horas |
| 25/04/2025 | Atividade 11 do cronograma | 16 horas |
| 01/05/2025 | Atividade 12 do cronograma | 13 horas |
| 07/05/2025 | Atividade 13 do cronograma | 40 horas |
| 01/06/2025 | Atividade 14 do cronograma | 15 horas |

## Código da Aplicação

Acesso ao código da aplicação está disponível nos links:

<https://github.com/RichardGuedesRib/tharseo_backend_v2>;

<https://github.com/RichardGuedesRib/tharseo_frontend_v2>;

Quanto ao vídeo, o mesmo só poderá ser realizado ao término total do projeto, sendo assim esse tópico ainda passará por atualizações.

Link para acesso a aplicação:

[https://tharseocripto.web.app](https://tharseocripto.web.app/)

**EM DESENVOLVIMENTO SERÁ COMPLETADO QUANDO CONCLUÍDO O VÍDEO PARA ENTREGA DIA 08**

Informe também o **endereço do *site* de sua aplicação**. É importante observar que, no ambiente fornecido para a avaliação, a base de dados deverá apresentar exemplos de teste previamente cadastrados que permitam visualizar o correto funcionamento do sistema. Indique também as **credenciais de acesso**, para **todos os perfis de usuários** da aplicação, que devem ser **empregadas pelos avaliadores**.

Indique o ***link*** para acesso ao **vídeo de apresentação de seu projeto**. Espera-se a produção de um vídeo sintético de, **no máximo**, **5 minutos**, no formato **MP4**, apresentando o projeto e a solução desenvolvida.>

## Avaliação Retrospectiva

Foi um semestre extremamente desafiador, diversas necessidades de entregas , optamos por alterar toda a linguagem da aplicação, sendo necessário o aprendizado da linguagem enquanto desenvolvia, tivemos a oportunidade de aprender diversidade de tipos de teste e implementa-lo na nossa aplicação (Jest e Cypress), tivemos a oportunidade através da matéria de computação em Nuvem 2, tornar a infra da nossa aplicação mais robusta, com maior disponibilidade e escalabilidade, sendo essas propriedades extremamente necessárias dada ao tipo da nossa aplicação, sendo outro desafio a gestão e distribuição de tempo entre o projeto integrador e as demais demandas de entregas, ao término podemos observar que foi desafiador, porém gerou grande evolução em todos e maturidade com um ambiente real de trabalho, onde deve-se gerir o tempo e escolher oque deve ser priorizado por importância e prazos.

## Objetivos Estimados

O objetivo do presente trabalho consiste na concepção de uma aplicação destinada a viabilizar e incentivar o usuário a empreender e investir no mercado de Crypto moedas, visando a automação do processo de compra e venda, gerando lucro e satisfação ao cliente, otimizando o seu tempo para que o mesmo tempo que seria gasto avaliando e analisando o mercado para que esse tempo seja aproveitado com outras atividades, gerando satisfação e lucratividade ao usuário.

A aplicação encontra sua motivação primordial no público em geral pois muitas das pessoas tem como dificuldade para investimento o tempo e a falta de conhecimento de como operar, o Tharseo tem como objetivo automatizar o processo para os que não tem tempo para dedicar ao mercado financeiro, mas também dando suporte aos leigos que não conhecem do mercado, gerando lucro e satisfação a todo tipo de usuário, desde o que domina o mercado até o mais leigo.

A proposta do grupo consiste no desenvolvimento de um aplicativo que compre uma fração das criptos à partir de um pré-set na aplicação e após determinada valorização o sistema faça a sua venda de forma automatizada rendendo lucro à partir da diferença do valor de compra para o valor da venda, sendo destinado do lucro um percentual simbólico a empresa Tharseo, e o demais ao seu usuário.

Com uma ampla adesão e participação, almejamos obter destaque não apenas nacional, mas também internacional, contribuindo significativamente para que as pessoas se tornem independentes no seu planejamento de investimentos e consequentemente promovendo a marca Tharseo.

## 19.2 Objetivos Alcançados

Ao término do projeto temos uma aplicação funcional que cumpre com os objetivos definidos a princípio em relação a funcionalidade de automação de compras e venda de crypto moedas, acreditamos que os próximos passos para futuras melhorias da aplicação seja a implementação de IA como auxilio na tomada de decisões, avaliando tendências gráficas, verificando padrões de quedas e subidas de kandles e sendo maior auxílio para o leigo traçar as suas estratégias de investimentos, pois hoje um dos pontos chaves para a aplicação operar é o traçar de estratégias, que a principio pode não ser tão simples para uma pessoa leiga em investimentos operar.

## Lições aprendidas

|  |  |  |
| --- | --- | --- |
|  | **Retrospectiva (Lições Aprendidas)** | |
|  | **Descrição da Lição** | **Classificação** |
| 1 | “Ninguém é incapaz, e o esforço é o segredo do sucesso” | Positivo |
| 2 | A automação de testes de software é muito boa, mas não substitui totalmente os testes manuais | Positivo |
| 3 | Resiliência é sempre necessário para atender a um cliente volátil, vivemos isso semestre a semestre com as mudanças de tecnologias exigidas | Positivo |
| 4 | O projeto deve ser acompanhado de perto por todos os integrantes para não gerar desvios no processo de produção do projeto | Negativo |
| 5 | Nem sempre a melhor solução é a que cabe no bolso em relação a infraestrutura | Negativo |
| 6 | Distribuição igual de tarefas não sobrecarrega e não desgasta ninguém | Positivo |

## Referências

1. ANBIMA. \*\*Volume investido por brasileiros cresce 9,7% em 2023 e chega a R$ 5,5 trilhões\*\*. Disponível em: https://www.anbima.com.br/pt\_br/imprensa/volume-investido-por-brasileiros-cresce-9-7-em-2023-e-chega-a-r-5-5-trilhoes.html. Acesso em: 03 maio 2024.

2. ANBIMA. \*\*Raio-X do investidor brasileiro chega à sétima edição com novos levantamentos sobre estresse financeiro e apostas online\*\*. Disponível em: https://www.anbima.com.br/pt\_br/noticias/raio-x-do-investidor-brasileiro-chega-a-setima-edicao-com-novos-levantamentos-sobre-estresse-financeiro-e-apostas-online-8A2AB2AE8F2C5D7F018F2CFA8A016764-00.html. Acesso em: 03 maio 2024.

3. ANBIMA. \*\*Raio-X do investidor brasileiro\*\*. Disponível em: https://www.anbima.com.br/pt\_br/especial/raio-x-do-investidor-brasileiro.html. Acesso em: 03 maio 2024.

4. CUNHA, Fernando. \*\*Requisitos funcionais e não funcionais: o que são\*\*. Disponível em: https://mestresdaweb.com.br/tecnologias/requisitos-funcionais-e-nao-funcionais-o-que-sao. Acesso em: 03 maio 2024.

5. DEVMEDIA. \*\*Elicitação de requisitos: levantamento de requisitos e técnicas de elicitação\*\*. Disponível em: https://www.devmedia.com.br/elicitacao-de-requisitos-levantamento-de-requisitos-e-tecnicas-de-elicitacao. Acesso em: 03 maio 2024.

6. ENGENHARIA DE SOFTWARE MODERNA. \*\*Capítulo 3\*\*. Disponível em: engsoftmoderna.info/cap3.html. Acesso em: 04 maio 2024.

7. FLANAGAN, David. \*\*JavaScript: o guia definitivo\*\*. Bookman.

8. CARVALHO, Vinicius. \*\*MySQL: Comece com o principal banco de dados\*\*. Casa do Código.

9. ROCKETSEAT. \*\*WebSocket: o que é e quando usar\*\*. Disponível em: https://blog.rocketseat.com.br/websocket-o-que-e-e-quando-usar/#:~:text=WebSocket%20%C3%A9%20uma%20tecnologia%20de,eficiente%20e%20em%20tempo%20real. Acesso em: 03 maio 2024.

10. SOMOSTERA. \*\*Comandos Git\*\*. Disponível em: https://blog.somostera.com/desenvolvimento-web/comandos-git#:~:text=O%20git%20%C3%A9%20um%20sistema,as%20mudan%C3%A7as%20para%20evitar%20erros. Acesso em: 10 maio 2024.

11. HOSTINGER. \*\*O que é Docker?\*\*. Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-docker#:~:text=O%20Docker%20%C3%A9%20um%20software,Linux%20e%20macOS%2C%20sem%20problemas. Acesso em: 12 maio 2024.

12. MICROSOFT AZURE. \*\*O que é Azure?\*\*. Disponível em: https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-azure/?&ef\_id=k\_Cj0KCQjw0\_WyBhDMARIsAL1Vz8v2KJnUBU9uMLb2OLjgZfrMdjCoRiw5\_LaxFhbQ1OEu3vvfsuya0voaAq6xEALw\_wcB\_k&OCID=AIDcmmzmnb0182\_SEM\_k\_Cj0KCQjw0\_WyBhDMARIsAL1Vz8v2KJnUBU9uMLb2OLjgZfrMdjCoRiw5\_LaxFhbQ1OEu3vvfsuya0voaAq6xEALw\_wcB\_k&gad\_source=1&gclid=Cj0KCQjw0\_WyBhDMARIsAL1Vz8v2KJnUBU9uMLb2OLjgZfrMdjCoRiw5\_LaxFhbQ1OEu3vvfsuya0voaAq6xEALw\_wcB. Acesso em: 15 maio 2024.

13. NORD RESEARCH. \*\*O que são criptomoedas, como funcionam e por que investir\*\*. [S. l.], 19 jun. 2024. Disponível em: https://www.nordinvestimentos.com.br/blog/o-que-sao-criptomoedas/#:~:text=A%20compra%20e%20venda%20de,considerando%20o%20pre%C3%A7o%20do%20ativo. Acesso em: 21 jun. 2024.

14. NUNES, Melissa. \*\*Home Broker Rico: passo a passo para comprar e vender!\*\*. [S. l.], 12 maio 2023. Disponível em: https://www.idinheiro.com.br/investimentos/home-broker-rico/. Acesso em: 21 jun. 2024.

15. BOMFIM, Ricardo. \*\*Bitcoin fica estável após 3 quedas seguidas e em meio a novas compras da MicroStrategy\*\*. São Paulo, 22 jun. 2024. Disponível em: https://valor.globo.com/financas/criptomoedas/noticia/2024/06/20/bitcoin-fica-estvel-aps-3-quedas-seguidas-e-em-meio-a-novas-compras-da-microstrategy.ghtml. Acesso em: 21 jun. 2024.

16. FELIX, César. \*\*Como começar a investir em criptomoedas?\*\*. [S. l.], 9 jul. 2023. Disponível em: https://exame.com/future-of-money/como-comecar-a-investir-em-criptomoedas/. Acesso em: 21 jun. 2024.

17. \*\*Moedas digitais e economia global: o impacto das criptomoedas nas finanças tradicionais\*\*. [S. l.], 19 out. 2023. Disponível em: https://www.sensio.com.br/blog/moedas-digitais#:~:text=Impacto%20das%20criptomoedas%20nas%20finan%C3%A7as%20tradicionais,-As%20criptomoedas%20t%C3%AAm&text=Ao%20operarem%20em%20uma%20rede,inclusive%20servi%C3%A7os%20de%20contabilidade%20pre%C3%A7o. Acesso em: 21 jun. 2024.

18. ANDRADE, Jenne. \*\*Pesquisa revela o perfil dos investidores globais de criptomoedas\*\*. [S. l.], 24 mar. 2022. Disponível em: https://einvestidor.estadao.com.br/criptomoedas/perfil-investidor-criptomoedas/. Acesso em: 21 jun. 2024.

19. \*\*A principal plataforma de negociação de criptomoedas do Brasil\*\*. [S. l.], 2024. Disponível em: https://www.crypto.com/br. Acesso em: 21 jun. 2024.

20. NUNES, Melissa. \*\*As 11 melhores corretoras de criptomoedas no Brasil\*\*. [S. l.], 22 jun. 2024. Disponível em: https://www.idinheiro.com.br/investimentos/corretoras/melhores-corretoras-criptomoedas/. Acesso em: 21 jun. 2024.

21. **POSTGRESQL**. Disponível em: https://www.postgresql.org. Acesso em: 29 mar. 2025.