**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

**Pós-graduação *Lato Sensu* em Arquitetura de *Software* Distribuído**

Projeto Integrado

Relatório Técnico

Sistema Gerenciador de Investimento

de Renda Variável

Eduardo Sampaio Viana

Belo Horizonte

Março 2023

# Projeto Integrado – Arquitetura de Software Distribuído

**Sumário**

[Projeto Integrado – Arquitetura de Software Distribuído 2](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844011)

[1. Introdução 3](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844012)

[2. Especificação Arquitetural da solução 5](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844013)

[2.1 Restrições Arquiteturais 5](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844014)

[2.2 Requisitos Funcionais 5](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844015)

[2.3 Requisitos Não-funcionais 6](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844016)

[2.4 Mecanismos Arquiteturais 7](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844017)

[3. Modelagem Arquitetural 7](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844018)

[3.1 Diagrama de Contexto 8](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844019)

[3.2 Diagrama de Container 9](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844020)

[3.3 Diagrama de Componentes 11](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844021)

[4. Avaliação da Arquitetura (ATAM) 13](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844022)

[4.1 Análise das abordagens arquiteturais 13](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844023)

[4.2 Cenários 13](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844024)

[4.3 Evidências da Avaliação 14](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844025)

[5. Avaliação Crítica dos Resultados 16](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844026)

[6. Conclusão 17](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844027)

[Referências 18](file:///C:\projetos\Projeto%20TCC\docs\RelatorioTecnico%20Entrega%20A2%20Eduardo%20Sampaio%20Viana.docx#_Toc116844028)

## Introdução

No Brasil os ativos de renda variável são negociados pela B3 que é uma bolsa de valores fundada em São Paulo em 1890. Atualmente o número de investidores na bolsa de valores tem aumentado cada vez mais. As pessoas estão migrando da poupança para investimento de renda variável e renda fixa. Em um cenário atual com uma taxa Selic 13,75% ao ano, a renda fixa tornou-se uma excelente oportunidade de ganhos para o curto prazo. A taxa Selic é a taxa básica de juros da economia brasileira e influência as demais taxas de juros do Brasil. Em contrapartida, os investimentos de renda variável apresentam também grandes oportunidades devido à taxa Selic elevada, fazendo com que as ações e fundos imobiliários fiquem bastantes descontados, pois, se encontram em um ciclo de baixa da bolsa de valores, porém mesmo descontados apresentam um maior risco e devem ser analisados e gerenciados com detalhes.

O mercado financeiro possui uma variedade grande de ativos e a escolha de um ativo de renda variável necessita de muitas informações, tornando uma tarefa difícil mesmo para profissionais experientes no mercado financeiro. Um bom investimento depende de boas decisões, errar na escolha de um determinado ativo pode levar o investidor a perder dinheiro. Temos hoje aproximadamente 400 empresas no qual podemos ser sócios por meio de ações com segmentos variados e 150 fundos imobiliários listados na bolsa de valores, tornando a escolha um pouco difícil. O investidor está buscando comprar ações e fundos imobiliários por um bom preço que paguem bons dividendos. Dividendos são proventos pagos aos acionistas por possuir um determinado ativo. Cada empresa paga seus dividendos em datas diferentes e com frequências diferentes, podendo ser de forma anual, semestral, quadrimestral, trimestral ou mensal, essas informações são importantes para o investidor ter uma previsibilidade de seus ganhos. Outro problema é a visualização de ativos escolhidos que muitos investidores recorrem a ferramentas como Excel ou outros sites extremamente complexos dificultando gerenciamento desses ativos, através dessas ferramentas o investidor tenta visualizar a taxa de rentabilidade dos investimentos e sua evolução patrimonial.

O objetivo deste trabalho é permitir que os investidores de renda variável possam ter informações sobre ações e fundos imobiliários que ajudem na sua escolha para montagem e gerenciamento de sua carteira de investimentos e a visualização de pagamentos de proventos, rentabilidade da carteira de investimento e evolução patrimonial. Ações são títulos de renda variável que representam uma fração do capital social de uma empresa, quando compramos uma ação nos tornamos sócios desta empresa e também dos seus resultados. Os fundos imobiliários são títulos de renda variável que reúnem seus recursos para serem aplicados em conjunto no mercado imobiliário. Uma carteira de investimentos agrupa todos os ativos de um investidor.

A principal motivação deste trabalho é uma tentativa de facilitar e ajudar de maneira social na educação financeira dos brasileiros, pois ao contrário dos Estados Unidos ainda é uma parcela muito pequena da população que conseguem ter conhecimento e dinheiro para realizar investimentos.

O objetivo deste trabalho é apresentar uma descrição do projeto arquitetural sistema gerenciador de investimentos de renda variável.

Os objetivos específicos propostos são:

* Visualizar detalhes ações e fundos imobiliários
* Gerenciar carteira de investimentos
* Visualizar a rentabilidade da carteira e de seus ativos
* Visualizar próximos proventos
* Visualizar evolução patrimonial

## Especificação Arquitetural da solução

Esta seção apresenta a especificação básica da arquitetura da solução a ser desenvolvida, incluindo diagramas, restrições e requisitos definidos pelo autor, tal que permitam visualizar a macro arquitetura da solução.

## Restrições Arquiteturais

A abaixo as restrições arquiteturais relativas à aplicação a ser desenvolvida:

R1: Deve utilizar a linguagem Python.

R2: Deve ser utilizado o framework FastApi para fornecer uma API REST

R3: Deve ser uma aplicação web utilizando framework NextJs

R4: Deve permitir usuário se autenticar através do google ou por um novo cadastro

R5: Deve permitir autorização através token JWT

## Requisitos Funcionais

Os requisitos funcionais descrevem o como sistema deve ser comportar para atender a expectativas e necessidades do usuário.

Listagem de requisitos funcionais do sistema:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Descrição Resumida** | **Dificuldade (B/M/A) \*** | **Prioridade**  **(B/M/A) \*** |
| RF01 | Buscar ativos pelo código | B | A |
| RF02 | Visualizar detalhes de ações | A | A |
| RF03 | Visualizar detalhes de fundos imobiliários | A | A |
| RF04 | Visualizar próximos pagamentos de proventos | A | M |
| RF05 | Visualizar patrimônio | M | A |
| RF06 | Cadastrar uma nova transação de compra de ativos | M | A |
| RF07 | Cadastrar uma nova transação de venda de ativos | M | A |
| RF08 | Atualizar uma transação de compra ou venda de ativos | B | A |
| RF09 | Listar transação realizadas | B | A |
| RF10 | Cadastrar uma nova data de proventos para um ativo | B | B |
| RF11 | Atualizar data de proventos para um ativo | B | B |
| RF12 | Deletar ativos da carteira de investimento | B | B |

\*B=Baixa, M=Média, A=Alta.

## Requisitos Não-funcionais

Os requisitos funcionais são requisitos que estão relacionados ao uso da aplicação que como restrições e aspectos de qualidade.

Listagem de requisitos não funcionais do sistema:

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrição** | **Prioridade**  **B/M/A** |
| RNF01 | Validar código do ativo existente | A |
| RNF02 | Validar quantidade mínima de caracteres | B |
| RNF03 | Validar confirmação de senha de usuário | B |
| RNF04 | Resposta API REST em JSON | B |
| RNF05 | Validar e-mail de cadastro do usuário | A |

## Mecanismos Arquiteturais

Esta seção deve apresentar uma visão geral dos mecanismos que compõem a arquitetura do software, baseando-se em três estados: (1) análise, (2) design e (3) implementação. Em termos de Análise devem ser listados os aspectos gerais que compõem a arquitetura do software, como: persistência, integração com sistemas legados, geração de logs do sistema, ambiente de front end, tratamento de exceções, formato dos testes, formato de distribuição/implantação (deploy), dentre outros. Em Design deve-se identificar o padrão tecnológico a seguir para cada mecanismo identificado na análise. Em Implementação deve-se identificar o produto a ser utilizado na solução, caso ela fosse implementada.

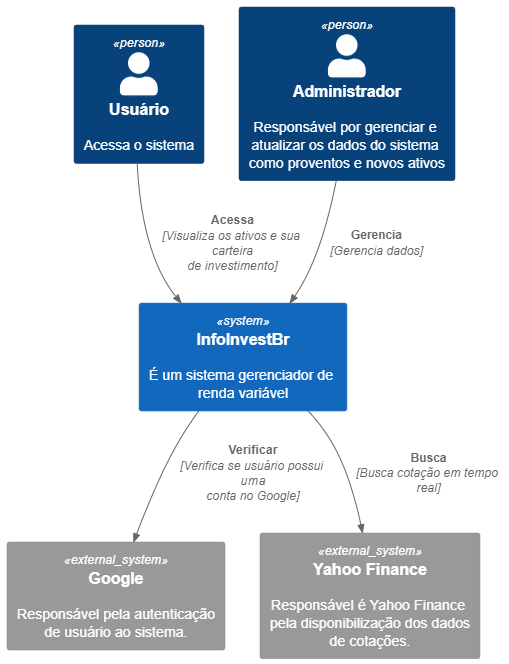
|  |  |  |
| --- | --- | --- |
| **Análise** | ***Design*** | **Implementação** |
| Persistência | ORM | SqlAlchemy |
| Persistência | Banco de dados Relacional | PostgreSQL |
| Cache | Banco de dados chave e valor | Redis |
| Front end | Interface de comunicação com usuário do sistema | Next JS |
| Back end | Linguagem | Python |
| API | REST | FastApi |
| Log do sistema | Implementação dos recursos de log | Logoru |
| Autenticação e Autorização | Protocolo de autenticação e autorização | OpenID |
| Segurança Criptografia | Hash | Passlib bcrypt |
| Versionamento | Versionamento de código | Git |
| Deploy | Utilização de container | Docker |

## Modelagem Arquitetural

Esta seção apresenta a proposta da modelagem arquitetural do sistema de gerenciamento de renda variável, a solução proposta permite ter uma visão geral do projeto. Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para documentação de arquitetura de software. Dos quatro níveis que compõem o modelo C4 três serão apresentados aqui que são diagrama de contexto, diagrama de container e o diagrama de componente. Para desenvolvimento dos diagramas foi utilizado a ferramenta PlantUML que permite criar diagramas por meio de uma linguagem de texto simples.

## 3.1 Diagrama de Contexto

O nível 1, o diagrama de contexto, apresenta uma visão geral da macro arquitetura do projeto proposto com nível de abstração bem elevado, mostrando o sistema de software e suas responsabilidades e seus principais usuários e suas dependências externas. O usuário e o administrador acessam o sistema através sua conta pessoal. O sistema possui um mecanismo de autorização de acesso através do Google e utiliza a API do Yahoo Finance para obter cotações dos ativos. A aplicação permite o cadastro de transações de ativos para construir uma carteira de investimentos e também permite análises de ações e fundos imobiliários e visualização de patrimônio cadastrados na plataforma. O papel do administrador e ter um acesso para poder cadastrar novos ativos na plataforma e incluir informações sobre novos pagamentos de proventos. Abaixo a figura do diagrama de contexto:

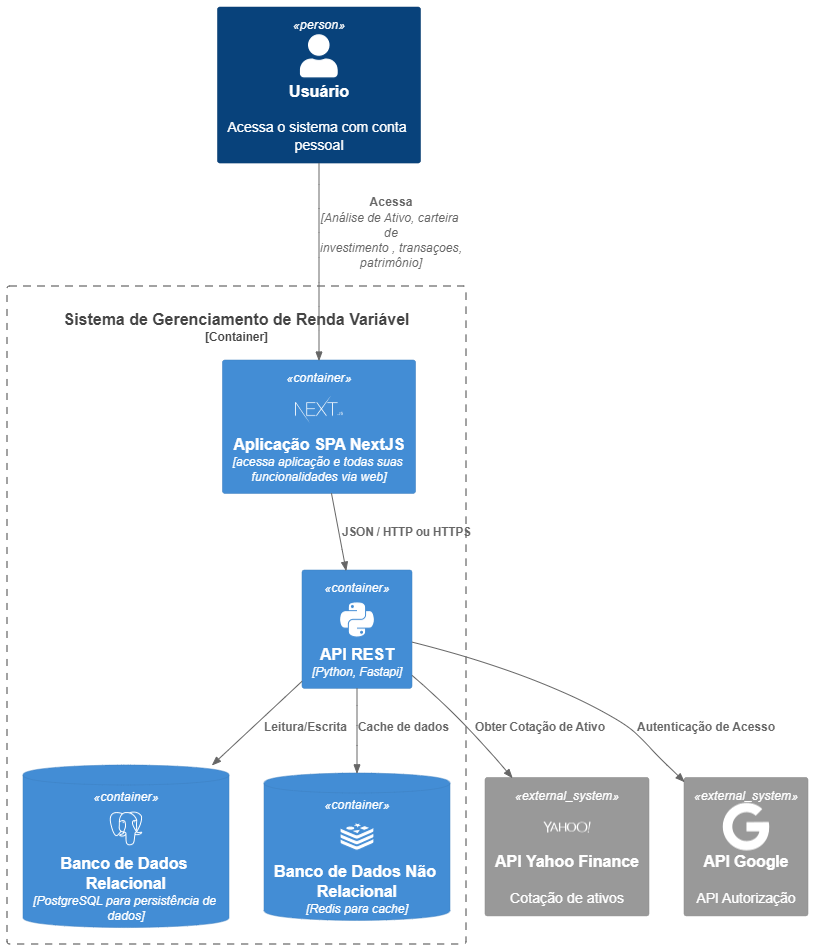


**Figura 1 – Diagrama de Contexto da aplicação**

**Link:** [**Link para imagem**](https://imgur.com/a/vxRz4T9)

## 3.2 Diagrama de Container

O nível 2, o diagrama de Container amplia o sistema de software e mostra os containers da aplicação como os componentes (aplicativos, armazenamentos de dados, microservices, etc.) que compõem esse sistema de software estão distribuídos e organizados. As decisões de tecnologia que você tomou devem ser contempladas nesse diagrama. O diagrama abaixo apresenta as soluções tecnológicas utilizadas. A proposta do projeto é uma aplicação web, uma Single Page Application construída com o framework NextJs. NextJs é um framework de código aberto criado em 2016 pela Vercel que permite funcionalidades como renderização do lado do servidor e geração de sites estáticos para aplicativos da web baseados em React. O backend possui uma API REST construída com framework FastApi. Framework FastApi é focado no desenvolvimento de API com Python e tem como principais características ser rápido, simples e moderno. Para persistência dos dados foi utilizado SGBD PostgreSQL devido sua alta performance e escalabilidade e também por ser de código aberto gerando um custo bem menor em relação a outras soluções de banco de dados.



**Figura 2 – Diagrama de container.**

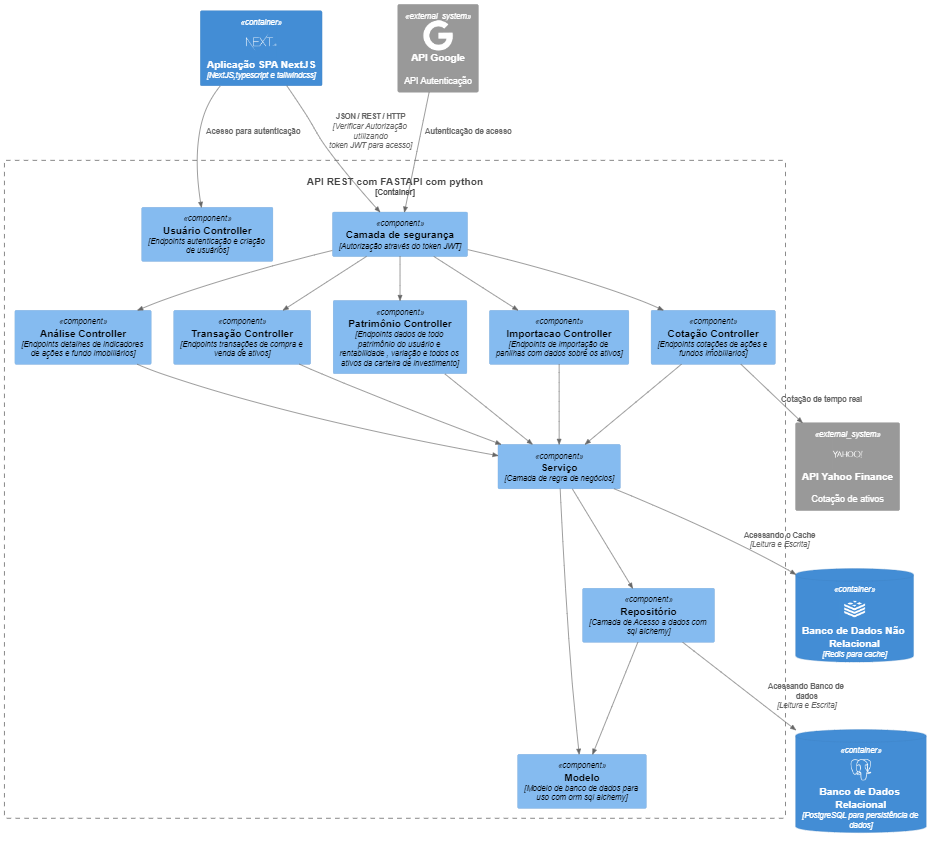
[Link para imagem](https://imgur.com/fg4EFiC)

## 3.3 Diagrama de Componentes

O diagrama componente mostra o nível 3 de detalhes para mostrar de maneira individual os detalhes da aplicação. O diagrama abaixo exibe cada funcionalidade da aplicação fornecidas por meio de uma API REST e seus endpoints.

* O componente usuário controller fornece endpoints para manipulação de usuário como a criação de um novo usuário e verificação de autenticação para usuários já cadastrados e esse único endpoint no qual não necessário passar pela camada de segurança da aplicação para ter acesso.
* A Camada de segurança fornece mecanismos para controle de acesso aos endpoints da aplicação através token JWT. Para acesso o usuário envia suas credenciais para aplicação, ao validar essas informações é retornado um token de acesso que autoriza o usuário a poder utilizar a aplicação.
* O controller de análise visa fornecer endpoints com informação em detalhes sobre ação e fundos imobiliários.
* O controller de cotação visa fornece cotações extraídas em tempo real da API do Yahoo Finance.
* O controller de transação visa cadastrar operações de compra e venda formando uma espécie de histórico do usuário, podendo ser consultado a qualquer momento.
* O controller de patrimônio visa consolidar as transações em uma carteira de investimentos, mostrando todos os ativos possuídos de uma maneira agrupada e realizando cálculos que irão ajudar o investidor em suas decisões.
* O controller de importação somente pode ser acesso pelo papel de administrador do sistema, onde ele pode importar planilhas com os de renda variável em lote.
* Componente serviço visa concentrar as regras de negócio da aplicação.
* Componente repositório visa acessar o banco de dados e realizar operações.
* Componente camada de modelo visa através do ORM SqlAlchemy gerar as tabelas no banco de dados através mapeamento feito no modelo.

Abaixo o diagrama de componente:



**Figura 3 – Diagrama de Componentes.**

[Link para imagem](https://imgur.com/orEWW6t)

## Avaliação da Arquitetura (ATAM)

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção visando avaliar se ela atende ao que foi solicitado pelo cliente, segundo o método ATAM.

## Análise das abordagens arquiteturais

Apresente aqui um breve resumo das principais características da proposta arquitetural. Para isto, utilize o método Architecture Tradeoff Analysis Method (ATAM), no qual são utilizados cenários para fazer essa análise.

Exemplo:

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributos de Qualidade** | **Cenários** | **Importância** | **Complexidade** |
| Confiabilidade | Cenário 1 Validar quantidade mínima de caracteres | A | A |
| Confiabilidade | Cenário 2 Validar confirmação de senha | A | A |
| Confiabilidade | Cenário 3 Valida código de ativo existente | A | B |
| Interoperabilidade | Cenário 4 Resposta API REST em JSON | A | B |
| Confiabilidade | Cenário 5 o usuário precisa de um e-mail válido | A | B |

## 

## 4.2 Cenários

Mostre os cenários utilizados na realização dos testes da sua aplicação. Escolha cenários de testes que demonstrem os requisitos não funcionais (atributos de qualidade) sendo satisfeitos. Priorize os cenários para a avaliação segundo critérios quantitativos ou qualitativos.

*:*

*Cenário 1 - Confiabilidade: o sistema valida a quantidade mínima de caracteres no nome de usuário.*

*Cenário 2 - Confiabilidade: o* *sistema valida se senha e igual a confirmação de senha de usuário.*

*Cenário 3 - Confiabilidade: O sistema valida o código do ativo afim de evitar transações e busca com códigos inválidos.*

*Cenário 4 - Interoperabilidade: O sistema se comunica através de uma API REST trafegando os dados via JSON facilitando a integração com diversas tecnologias.*

*Cenário 5 - Confiabilidade:* *O sistema valida se o e-mail tem formato correto e se ele não existe no sistema já cadastrado.*

## 4.3 Evidências da Avaliação

Apresente as medidas registradas na coleta de dados. Para o que não for possível quantificar apresente uma justificativa baseada em evidências qualitativas que suportem o atendimento ao requisito não-funcional.

Evidências retirada da documentação da API REST do sistema:

O sistema exige uma quantidade mínima de caracteres para o campo nome no cadastro de usuário.

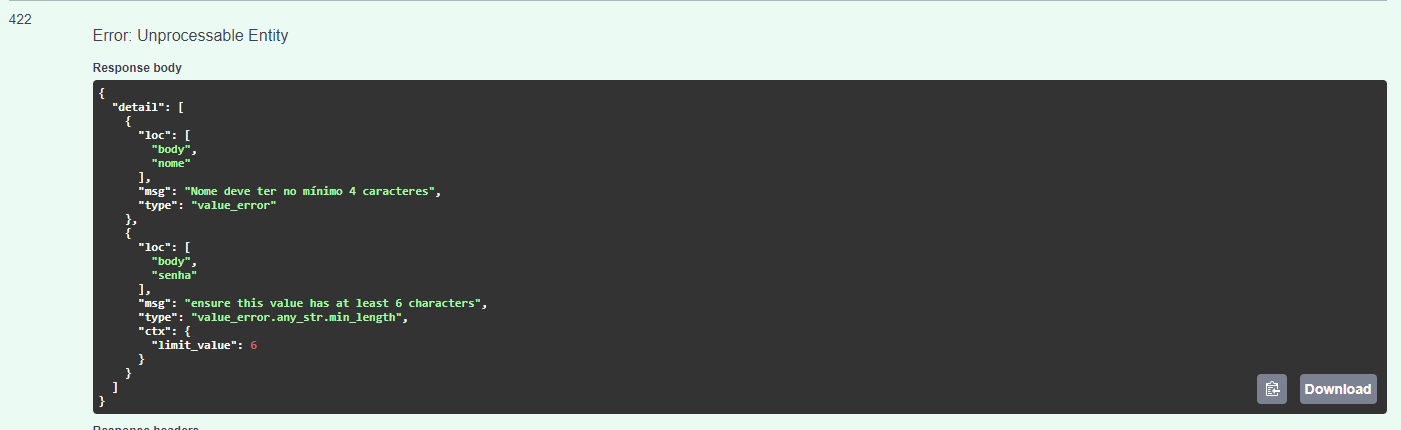


Figura 4: cenário *1*

Ao cadastrar um novo usuário os campos de senha e confirmar senha devem ser iguais.

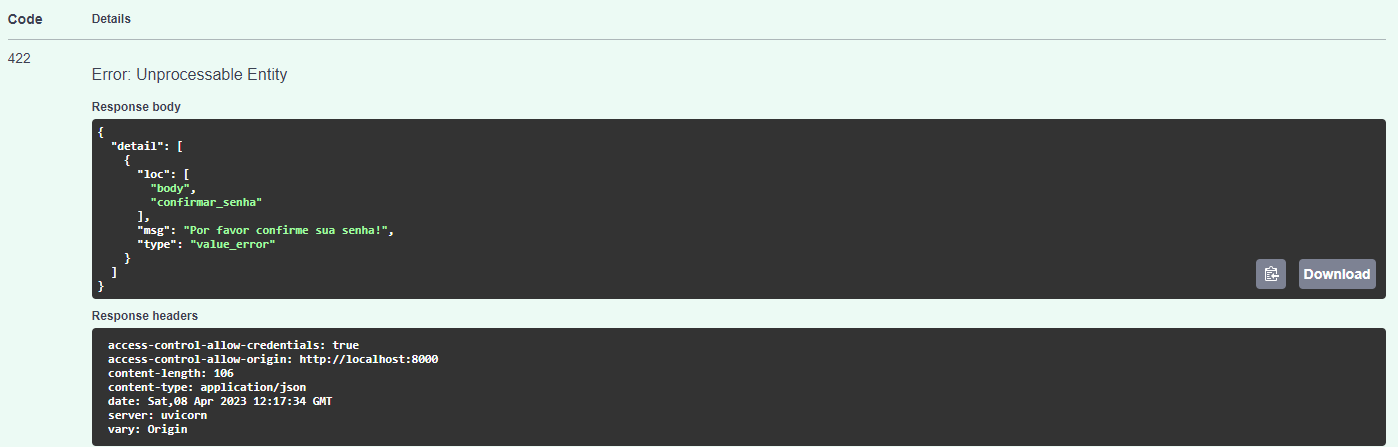


Figura 5: cenário 2

Ao buscar um determinado ativo o sistema valida se código buscado existe.



**Figura 6**: Cenário 3

O retorno da API REST tem sua resposta em JSON facilitando a integração com diversas tecnologias.

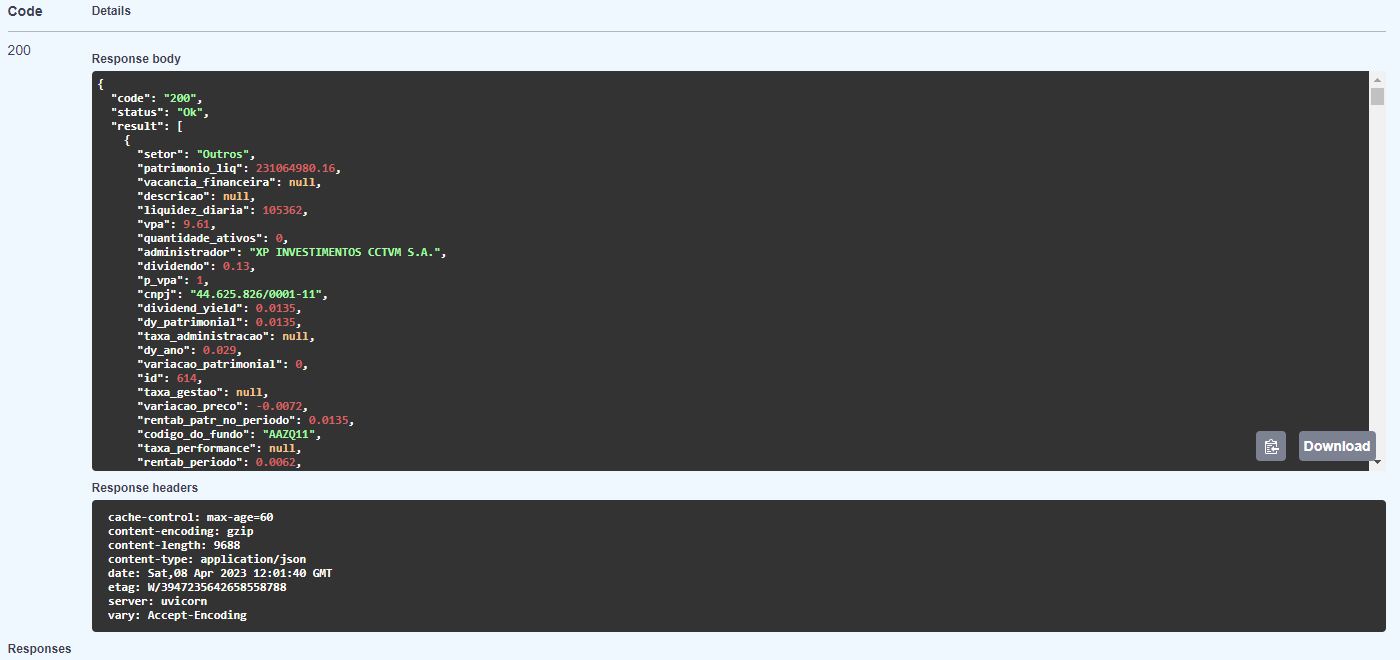


Figura 7: cenário 4

Ao Cadastrar um novo usuário o sistema valida se o e-mail é válido através de uma expressão regular e se o e-mail já foi cadastro no banco.

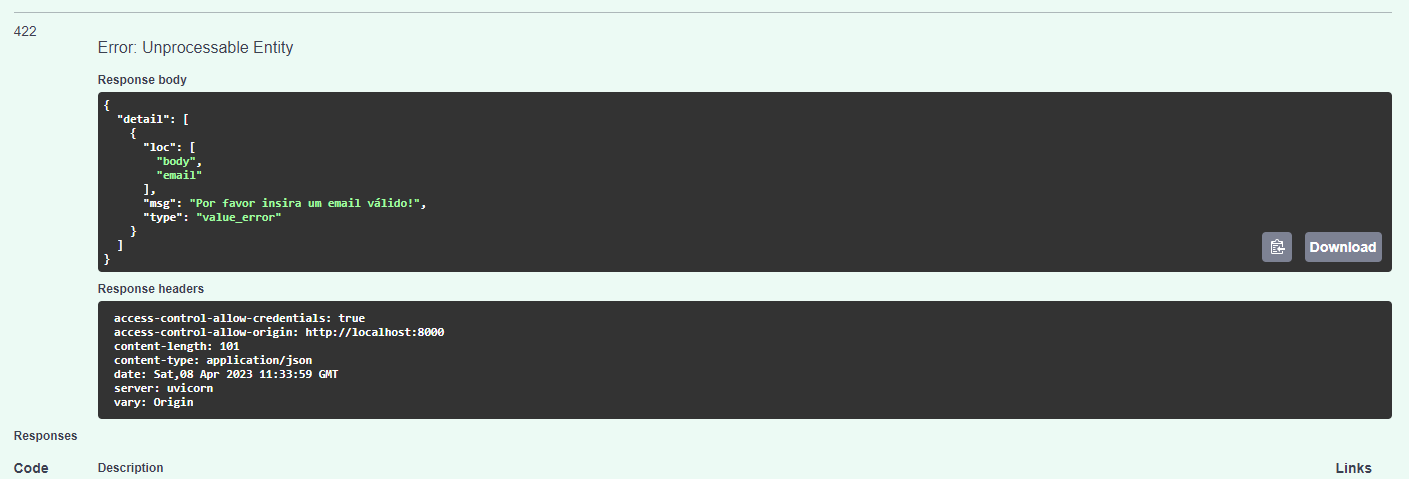


Figura 8: cenário 5

## Avaliação Crítica dos Resultados

Apresente aqui, de forma resumida, os principais pontos positivos e negativos da arquitetura proposta. Adote uma postura crítica que permita entender as limitações arquiturais, incluindo os prós e contras das tecnologias. Você pode utilizar o formato textual ou produzir um quadro resumo.

Ex. de quadro resumo:

|  |  |
| --- | --- |
| **Ponto avaliado** | **Descrição** |
| Custo | Ponto positivo devidos as tecnologias escolhidas e o uso de container Docker projeto tem custo muito baixo de sua implantação. |
| Escalabilidade | Ponto positivo facilmente escalável através de tecnologias como kubernetes. |
| Interoperabilidade | Ponto negativo o sistema ainda não se comunica através de WebSockets podendo tornar a interface mais rica ao receber informações em tempo real sem necessidade de diversas chamadas ao backend. |
| Usabilidade | Ponto negativo o sistema ainda não fornece mecanismos de acessibilidade. |
| Segurança | Ponto negativo o sistema não possui autenticação em dois fatores. |
| Segurança | Ponto positivo o sistema possui autenticação através do google facilitando a entrada do usuário que não precisa criar uma nova conta no sistema. |

## Conclusão

Este trabalho apresentou um projeto arquitetural de um sistema de gerenciamento de renda variável. Esse projeto visa ajudar o investidor a escolher e gerenciar sua carteira de investimento, trazendo informações de forma organizada para facilitar a tomada de decisão do investidor de renda variável. A arquitetura proposta possui grandes desafios e tecnologias recentes e isso gerou diversos aprendizados, a grande dificuldade foi conseguir reunir os dados reais e organizá-los de uma maneira que seja de fácil acesso posteriormente. Os dados do sistema foram extraídos de API como Yahoo Finance e mediante importação de planilhas de dados extraídos do site Fundamentus e organizados em um banco relacional, facilitando obtenção desses de dados posteriormente. A escolha da linguagem de programação Python foi devida sua simplicidade e extensa documentação e número de bibliotecas disponíveis no mercado. O Python em conjunto com seu framework FastApi trouxeram muita simplicidade ao desenvolvimento e isso ajudou na construção da API REST. O desafio de construir um sistema que tenha um baixo custo de implantação e um bom desempenho foram os dos principais pesos na escolha da arquitetura.

Lições aprendidas (ex.):

1. O uso do swagger como documentação de API.
2. O uso adequado de API REST através verbos http e seus código de status.
3. Produção de diagramas no modelo C4 através da ferramenta Plantuml.
4. Uso de cache distribuído através do Redis.
5. Utilização de container Docker.
6. Utilização de ORM SqlAlchemy.
7. Criação de aplicações com NextJs.
8. Desenvolvimento de estilos de interface utilizando o tailwindcss.

Desenvolvimentos Futuros:

* Utilização de WebSockets possibilitando cotação ser atualizada de maneira automática.
* A Expansão em mais ativos como renda fixa, ETF e tesouro direto
* A possibilidade de trabalhar com Stocks, Reits que são parte do mercado financeiro americanos.
* Incluir cadastro de ativos de criptomoeda na carteira de investimento.

## Referências

ADESHINA, A. A. **Building Python Web APIs with Fast API a fast -paced guide to building high-performance, robust web APIs with very little boilerplate code**. [S.l.]: PACKT PUBLISHING LIMITED, 2022.

‌ANTONIO, M. **Python e mercado financeiro**. [s.l.] Editora Blucher, 2021.

KUTLUCA, H. **C4 Model with PlantUML**. Disponível em: <https://medium.com/software-architecture-foundations/software-architecture-modeling-with-c4-model-e9e61d952121>. Acesso em: 8 abr. 2023.

**C4-PlantUML**. Disponível em: <https://github.com/plantuml-stdlib/C4-PlantUML>. Acesso em: 8 abr. 2023.

‌**Explicitando os Componentes de um Contêiner usando C4 Model**. Disponível em: <https://eximia.co/explicitando-os-componentes-de-um-conteiner-usando-c4-model>. Acesso em: 8 abr. 2023.

RIVA, M. **Real-World Next.js**. [s.l.] Packt Publishing Ltd, 2022.

**Getting Started | Next.js**. Disponível em: <https://nextjs.org/docs>.

‌ FASTAPI. **FastAPI**. Disponível em: <https://fastapi.tiangolo.com/>.

‌