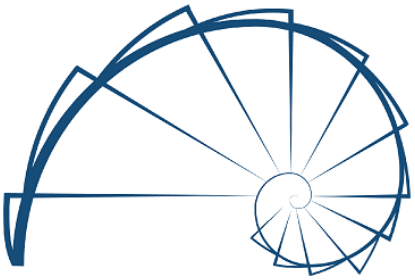

Underlying

PROJETO 2022.01 - UNDERLYING
DOCUMENTO DE ARQUITETURA

Versão 0.3

Equipe de Projeto Underlyng:

Bruno Brandão Borges - 2018014331

Ivan Leoni Vilas Boas - 2018009073

Leonardo Rodrigo de Sousa - 2018015965

Lucas Tiense Blazzi - 2018003310

Thiago Marcelo Passos - 2018002850

Wesley Alexandre de Almeida Gomes - 2018005806

IMC - Instituto de Matemática e Computação

Av. BPS, 1303 - Caixa postal 50 - 37500-903

Itajubá - MG - Brasil Telefone: 35-3629-1135

E-mail:imc@unifei.edu.br

Revisões do Documento

Revisões são melhoramentos na estrutura do documento e também no seu conteúdo. O objetivo primário desta tabela é a fácil identificação da versão do documento. Toda modificação no documento deve constar nesta tabela.

Data	Versão	Descrição	Autor
13/05/2022	0.0	Elaboração da DA	Ivan
15/05/2022	0.0	Definição dos Elementos de sistema	Lucas
17/05/2022	0.0	Análise RNF e Definição do RAS	Ivan
18/05/2022	0.1	Persona Pedro para RNF e Atributos de qualidade	Ivan
19/05/2022	0.2	Visão geral da arquiteta	Lucas
19/05/2022	0.2	Visão de modulo	Lucas
23/05/2022	0.3	Visão de deployment	Lucas
23/05/2022	0.3	Visão logica	Leonardo

Auditorias do Documento

Auditorias são inspeções conduzidas o SEPG – Software Engineer Process Group (Grupo de Engenharia de Processo de Software), e tem por objetivo garantir uma qualidade mínima dos artefatos gerados durante o processo de desenvolvimento. Essa tabela pode ser utilizada também pelo GN – Gerente da Área de Negócio com o objetivo de documentar a viabilidade do mesmo.

Data	Versão	Descrição	Autor
19/05/2022	0.0	Persona e analise de RNF e RAS	Lucas
19/05/2022	0.0	Elementos do sistema	Ivan
19/05/2022	0.1	Persona	Leonardo
23/05/2022	0.2	Visão Geral	Ivan
24/05/2022	0.3	Visão modulo, deployment e láogica	Wesley / Bruno

Sumário de Ilustrações

Figura 1 - Visão geral da arquitetura do sistema com as 6 camadas 24
Figura 2 - Módulo principal 25
Figura 3 - Módulo de api das opções 27
Figura 4- Módulo de data-sources de opções 28
Figura 5 - Módulo de api das ações 29
Figura 6 - Módulo de data-sources de ações 30
Figura 7 - Módulo de api das estratégias.....30
Figura 8 - Visão de deployment 33
Figura 9 - Visão de lógica dos dados 35

Sumário da Tabelas

Tabela 1 - serviços de arquitetura do Sistema UNDERLYING 11
Tabela 2 - Componentes de arquitetura do Sistema UNDERLYING 14
Tabela 3 - Recursos de arquitetura do Sistema UNDERLYING 15
Tabela 4 - Atributos de qualidade e RNF 18
Tabela 5 - Definições de RAS e das ferramentas de implementação 22
Tabela 6 - Atributos de qualidade aos RAS 22

Sumário

1. INTRODUÇÃO 7

1.1 Convenções, termos e abreviações 7

1.2 Identificação dos Requisitos 7

1.3 Descrição do cliente 8

1.4 Descrição dos usuários 8

2. PERSONA 8

2.1 Persona Única do Sistema 8

2.2 Principais Objetivos da Persona Pedro: 9

3. DESCRIÇÃO DOS ELEMENTOS 10

3.1 Elementos do sistema 10

3.2 Serviços do sistema Uderlyng: 10

3.3 Componentes do sistema Underlyng: 12

3.4 Recursos do sistema: 14

4. ANÁLISE DE RNF 17

4.1 Identificação dos RNF: 17

4.2 Definição dos RAS: 18

5. REPRESENTAÇÃO DAS VISÕES 23

5.1 Visão geral da arquitetura: 23

5.2 Visão de módulos: 24

5.3 Visão de deployment: 31

5.4 Visão lógica: 34

6. ANEXOS: 36

1. INTRODUÇÃO

Este documento especifica toda a arquitetura do sistema UNDERLYING fornecendo aos desenvolvedores as informações necessárias para a execução de seu projeto e implementação, assim como para a realização dos testes e homologação.

Esta introdução fornece as informações necessárias para fazer um bom uso deste documento, explicitando seus objetivos e as convenções que foram adotadas no texto. As demais seções apresentam a especificação do UNDERLYING e estão organizadas como descrito abaixo:

Seção 2 - **Persona Pedro**: apresenta a Persona do Pedro um do mercado de Opções e as histórias de usuário para a geração dos atributos de qualidade e dos requisitos não funcionais

Seção 3 **- Descrição dos elementos da arquitetura:** Identifica cada um dos elementos da arquitetura com a sua responsabilidade em prol do sistema e referencias dos elementos módulos, componentes e serviços.

Seção 4 - **Analise de RNF da arquitetura**: é realizada a descrição de como serão abordados na arquitetura os RNF através da análise e identificação dos padrões arquiteturais usados nos frameworks ou plataformas de desenvolvimento utilizados.

Seção 5 - **Representação da arquitetura:** São apresentadas as seguintes visões arquiteturais: geral, de módulos, de deployment e a lógica.

Seção 6 - **Referências e Anexos:** são citados os documentos em anexos que colaboram para o desenvolvimento deste

1.1 Convenções, termos e abreviações

A correta interpretação deste documento exige o conhecimento de algumas convenções e termos específicos, que são descritos a seguir.

1.2 Identificação dos Requisitos

Por convenção, a referência aos requisitos será feita através do identificador, de acordo com o esquema abaixo:

[identificador de tipo + identificador Numérico]

O identificador de tipo de requisito pode ser:

- RF – Requisito funcional
- RFS – Requisito funcional de sistema

- RNF – Requisito não-funcional
- RAS - Requisitos Arquiteturalmente Significativos

Identificador é um número, criado sequencialmente, que determina que aquele requisito/história é único para um determinado tipo.

Exemplos: RF01, RF02, RFS01, RFS02, RNF01, RNF02, RAS01, RAS02

Add o vocabulário de domínio.

1.3 Descrição do cliente

Visto a necessidade do mercado financeiro e de seus usuários, e, a fim de auxiliar os investidores no processo de tomada de decisão através de dados apurados e gráficos o projeto está sendo desenvolvido para qualquer usuário investidor, seja um aprendiz que queira adentar ao mundo dos investimentos ou um experiente investidor que queira melhorar sua experiência como usuário da plataforma, a fim de tomarem as melhores decisões de mercado. O sistema é indicado principalmente para aqueles investidores que que desejam obter informações de produtos do mercado de opções e testar estruturas de operações.

1.4 Descrição dos usuários

A aplicação possuirá apenas um tipo de usuário. Este por sua vez será capaz de realizar todas as operações que o sistema ofereça, desde que seja realizado o cadastro e login na plataforma.

2. PERSONA

Nesta sessão ser apresentado a Persona Pedro, um investido, que representará o principal usuário do sistema e logo a seguir as histórias de Usuário.

2.1 Persona e cenários de atributos de qualidade

Pedro Costa Damasceno é um investidor do mercado de Opções. Ele opera mercado de opções a 3 anos. Se formou em Administração da universidade de santa Catarina em 2021. Possui 26 anos e utiliza a diariamente seu notebook para realizar pesquisas na B3 e demais sites para se informar melhor sobre o mercado de Opções. Atualmente ele utiliza um sistema pago para obter as informações que precisa para tomar as melhores decisões acerca do mercado de Opções, porém não somente o design do sistema o incomoda como os gráficos não lhe proporcionam uma boa **experiencia de usuário**, falta a disponibilização de alguns dados que são vistos como importantes para análises mais complexas, o processo ainda é muito manual paro o acesso aos dados e para a montagem de cenários fictícios e, ainda apresenta uma **limitação visua**l na geração dos cenários, e, somado a isso, o custo de acesso a tais plataformas atuais são caras. Pedro esperava obter um

sistema web que seja **intuitivo, com recursos visuais (gráficos)** e que estes sejam **fáceis de serem interpretados** com a **utilização de legendas** para assim analisar e realizar suas conclusões rapidamente.

O sistema para Pedro deve ser capaz de **validar os dados inseridos** para garantir **proteção contra erros** e sempre **abrir os links em uma nova página** e a página para maior usabilidade deve ser **responsiva** independentemente do tamanho da tela, ser **multiplataforma** (independente de sistema operacional) e **ser compatível** com a maioria dos **navegadores** modernos uma vez que Pedro quer utilizar o seu celular para fazer acesso ao sistema além do seu notebook.

É imprescindível para Pedro que o sistema **apresenta dados reais**, **atuais e consistentes** oriundos **diretamente da B3**. Uma vez que ele deve ser um espelho confiável das opções de mercado em tempo real.

Pedro gostaria de o sistema apresentasse uma **página de glossário** para pesquisar algum termo, abreviatura ou palavras que desconheça do mercado de opções e que facilite ele apresentar o sistema a novos possíveis usuários que não tem muita familiaridade com o mercado de opções

Pedro gostaria que o sistema permita apenas **a utilização de senha fortes** para garantir sua **segurança**, sendo capaz de **criptografar** seus dados e de **garantir sua autenticidade**, evitando assim o roubo de sua conta e o **acesso indevido** a suas informações e ao sistema.

Pedro quer que o sistema seja capaz tanto de fazer quanto de desfazer algumas ações do sistema como por exemplo de **recuperar alguma ação de exclusão ou desfazer o compartilhamento antes permitido.**

Como Pedro viaja bastante gostaria de ter um **sistema tolerante a falhas** em caso de houver oscilações na rede e que possa **acessar a qualquer hora** do dia e noite para sua utilização, principalmente nos horários de pico, assim para Pedro o sistema deverá apresentar o máximo de **disponibilidade** possível de acesso independentemente da quantidade de usuários que venham realizar o acesso.

Para Pedro, portanto, a qualidade estaria além das funções e tarefas que o sistema venha realizar, mas na capacidade do serviço como um todo que o sistema seria capaz de fornecer, assim para Pedro a qualidade do sistema estaria também na sua **usabilidade, segurança, confiabilidade, desempenho, compatibilidade e portabilidade.**

2.2 Principais Objetivos da Persona Pedro:

- ✓ Sistema padronizado e amigável
- ✓ Gráficos com legendas
- ✓ Validar dados de entrada
- ✓ Abrir nova página para links externos
- ✓ Sistema responsivo
- ✓ Multiplataforma
- ✓ Compatibilidade com vários navegadores
- ✓ Glossário

- ✓ Informações Reais, Atuais e consistentes a B3
- ✓ Criptografia
- ✓ Autenticidade
- ✓ Senhas fortes
- ✓ Fazer e desfazer exclusão e compartilhamento
- ✓ Tolerante a falha
- ✓ Alta disponibilidade
- ✓ Escalabilidade

3. DESCRIÇÃO DOS ELEMENTOS

Nesta sessão serão identificados e apresentados os elementos da arquitetura com suas responsabilidades e as referências.

3.1 Elementos do sistema

Os requisitos identificados na etapa de análise do sistema precisam ser estruturados em grupos ou elementos de software como por exemplo, módulos, componentes, serviços, recursos ou até mesmo outros sistemas. Nesta sessão será apresentado os elementos da arquitetura do sistema e seu propósito (responsabilidade) que são formados por elementos de estrutura (módulos, componentes, serviços, BD), elementos de comunicação (barramentos, brokers, proxys) e elementos externos.

O sistema UNDERLYING será estruturado em serviços, componentes e recursos, que juntos formaram a arquitetura do sistema. Para estes, portanto, serão detalhados quais os requisitos funcionais, requisitos funcionais de sistema e requisitos não funcionais de sistema em que cada elemento irá será responsável, referenciando assim os IDs para cada RF, RFS e RNF conforme foi especificado no documento de requisitos.

3.2 Serviços do sistema Uderlyng:

A seguir serão apresentados os serviços de arquitetura do Sistema com uma breve descrição e suas respectivas responsabilidades, assim como pode ser observado na tabela1 a seguir:

Nome dos Serviços	Descrição geral	Lista de responsabilidades (IDs RF e RFS)
Serviço de Usuários	Gerenciamento de informações de	RF03

	usuários do sistema, manutenção do sistema de autenticação	RFS06, RFS07, RFS08, RFS09
Serviço de Opções	Gerenciamento de dados e feed dos dados de ações do sistema, deverá realizar cálculos de métricas de mercado referente as opções, cálculo de greeks e payoff	RF01, RF02 RFS01, RFS02, RFS03, RFS04, RFS05, RFS10-RFS17, RFS25
Serviço de Ações	Gerenciamento de dados e feed dos dados de ações do sistema, deverá realizar cálculos de métricas de mercado referente as ações	RF08, RF09 RFS26, RFS27, RFS28
Serviço de Estratégias	Gerenciamento e manutenção de dados das estratégias dos usuários, elaboração do payoff das estratégias	RF05, RF06, RF07, RF10 RFS18, RFS19, RFS20, RFS21, RFS22, RFS23, RFS24, RFS25, RFS29, RFS30, RFS31
Serviço Externo Provedor de dados: B3	Serviço externo de onde serão consumidas informações de registro e séries de opções e ações	RF01, RF08 RFS03

Tabela 1 - serviços de arquitetura do Sistema UNDERLYING

Como a o desenvolvimento da arquitetura é baseado na **composição de microserviços** o sistema foi dividido em quatro serviços internos, e um serviço externo.

Os serviços externos podem ser divididos em: (1) Serviço de usuários, (2) Serviço de opções, (3) Serviço de ações e (4) Serviço de estratégias. A seguir será apresentado uma melhor descrição de cada um:

1- **Serviço de usuários**: realizará o controle dos dados dos usuários registrados na plataforma, além de ser responsável pelo processo de autenticação. Para isso o serviço contará com os **componentes de registro, login, atualização, remoção e recuperação de senha de usuários,** componentes esses que estarão distribuídos entre o front e o backend. Para essas tarefas os **recursos de Banco de Dados** (DynamoDB), para manutenção dos dados dos usuários, e o **Recurso de gerenciamento de usuários** (AWS Cognito), para registro, login, atualização e troca de senhas, serão utilizados.

2- **Serviço de opções**: realizará a manutenção de informações referentes as opções, fazendo a coleta de dados no Serviço Externo – B3, para feed da base de dados, sendo mantidas informações de registro e séries históricas de opções. O serviço deverá disponibilizar uma API para consumo das informações de opções pelo frontend, além de manter componentes auxiliares para o cálculo de algumas informações auxiliares relacionadas as opções. Os componentes associados a esse serviço são os **componentes de feed de opções, cálculo de métricas e greeks de opções, consulta de opções, cálculo de payoff.** Para a implementação o serviço de opções utilizará dos recursos: **Recurso de Storage (S3)** para armazenamento da

série histórica de opções, **Recurso de API (API Gateway)** para desenvolvimento da api de opções, **Recurso de funções serverless** (Lambda) para executar o código responsável por cada tarefa do serviço.

3- **Serviço de ações:** realizará a manutenção de informações referentes as ações, fazendo a coleta de dados no Serviço Externo – B3, para feed da base de dados, sendo mantidas informações de registro e séries históricas de ações. O serviço deverá disponibilizar uma API para consumo das informações de ações pelo frontend. Os componentes associados a esse serviço são os **componentes de feed de ações, consulta de ações, consulta de grade de opções da ação**. Para a implementação o serviço de opções utilizará dos recursos: **Recurso de Storage** (S3) para armazenamento da série histórica de ações, **Recurso de API** (API Gateway) para desenvolvimento da api de ações, **Recurso de funções serverless** (Lambda) para executar o código responsável por cada tarefa do serviço.

4- **Serviço de estratégias:** realizará o controle de dados referentes as estratégias criadas pelos usuários da plataforma, para isso, contarão com o desenvolvimento dos **componentes de criação, edição, compartilhamento, consulta e remoção das estratégias, além de auxiliares gráficos** desenvolvidos pelo frontend. Para o desenvolvimento arquitetural o serviço contará com os seguintes recursos: **Recurso de API (AWS API Gateway)** para desenvolvimento da api utilizada pelo frontend, **Recurso de Banco de Dados** (AWS DynamoDB) para armazenamento das informações referentes as estratégias, **Recurso de funções serverless** (AWS Lambda) para desenvolvimento do código responsável por desempenhar cada função do serviço.

Além dos serviços listados, o sistema contará com um **serviço externo (B3)** de onde serão obtidas as informações referentes e ações e opções registradas na plataforma, para isso, o sistema utilizará de um endereço do serviço externo que disponibiliza um arquivo “cru” no formato txt com as informações históricas desses objetos de investimento.

3.3 Componentes do sistema Underlyng:

A seguir serão apresentados os Componentes de arquitetura do Sistema com uma breve descrição e suas respectivas responsabilidades, assim como pode ser observado na tabela2 a seguir:

Nome dos Componentes	Descrição geral	Lista de responsabilidades (IDs RF e RFS)
Componente de login	Validação de usuário para disponibilizar token de autenticidade e permitir acesso ao sistema	RF03
Componente de registro de usuário	Registro de usuário a partir do fornecimento de dados cadastrais	RF03 RFS06

Componente de atualização de usuário	Alteração de dados cadastrais do usuário	RF03 RFS08
Componente de remoção de usuário	Remoção lógica de perfil de usuário impossibilitando login no sistema	RF03 RFS09
Componente de recuperação de senha	Recuperação de senha por email em caso de esquecimento	RF03
Componente de feed de opções	Feed da base de dados de opções a partir de scripts e normalização de dados disponibilizados em arquivo de texto pela B3	RF01 RFS03
Componente de cálculo de métricas e greeks de opções	Cálculo de métricas de mercado, como valor intrínseco e hedge ratio, além de greeks, como theta, gamma, delta	RF04 RFS10-RFS17
Componente de cáculo de métricas e greeks de opções	Cálculo de métricas de mercado, como valor intrínseco e hedge ratio, além de greeks, como theta, gamma, delta	RF04 RFS10-RFS17
Componente de consulta de opção	Disponibilização de dados completos de uma opção a partir de seu identificador	RF01 RFS01
Componente de cálculo de payoff	Cálculo da série de payoff de uma opção ou estratégia a partir dos dados base	RF07 RFS24
Componente de feed de ações	Feed da base de dados de opções a partir de scripts e normalização de dados disponibilizados em arquivo de texto pela B3	RF08
Componente de consulta de ação	Disponibilização de dados completos de uma ação a partir de seu identificador	RF08 RFS26
Componente de cálculo de métricas de ações	Cálculo de métricas de mercado de ações, como retorno e volatilidade	RF08
Componente de consulta de grade de opções	Disponibilização da grade de opções de uma ação a partir da ação alvo informada	RF09 RFS28
Componente de criação de estratégia	Registro de estratégia pelo usuário composta por um conjunto de opções reais e/ou fictícias	RF05, RF10 RFS18, RFS29

Componente de edição de estratégia	Alteração de dados de uma estrutura de operação já composta	RF05, RF10 RFS20, RFS30, RFS31
Componente de compartilhamento de estratégia	Compartilhamento de uma estrutura de operação, já criada, possibilitando a visualização por todos os usuários do sistema	RF06 RFS22
Componente de remoção de estratégia	Remoção lógica da estratégia, impossibilitando sua visualização	RF05, RF10 RFS21
Componente de consulta de estratégia	Consulta de uma estratégia já criada para visualização	RF05 RFS23
Componente de tabela de greeks	Visualização da composição das greeks de uma opção	RF01 RFS02
Componente de gráfico de série histórica	Visualização de gráficos de série, como preço de fechamento e série de métricas	RF01, RF08 RFS05
Componente de gráfico de payoff	Visualização do gráfico de payoff de uma opção ou estratégia	RF01, RF10 RFS25
Componente de visualização de opção	Visualização de dados cadastrais de uma opção	RF01 RFS02
Componente de visualização de ação	Visualização de dados cadastrais de uma ação	RF08 RFS27
Componente de grade de opção	Visualização da grade de opções completa de uma ação	RF09 RFS28

Tabela 2 - Componentes de arquitetura do Sistema UNDERLYING

3.4 Recursos do sistema:

A seguir serão apresentados os Recursos de arquitetura do Sistema com uma breve descrição e suas respectivas responsabilidades, assim como pode ser observado na tabela3 a seguir:

Nome dos Recursos	Descrição geral	Lista de responsabilidades (IDs RF, RFS, RNF)
Recurso de Storage (AWS S3)	Armazenamento de dados de séries históricas (ações e opções)	RNF26

		RNF17 RNF29
Recurso de API (AWS API Gateway)	Gerenciamento de rotas e integrações da API com as funções do sistema	RNF19 RNF20 RNF26
Recurso de Banco de Dados (AWS DynamoDB)	Armazenamento de dados dos usuários e das estratégias	RNF17 RNF26 RNF29
Recurso de funções serverless (AWS Lambda)	Execução serverless das funções python/node desenvolvidas no sistema	RNF26
Recurso de gerenciamento de usuários (AWS Cognito)	Gerenciamento e controle de usuários e autenticação (criptografia, recuperação de senhas, login)	RNF08 ao RNF12 RNF26
Recurso de interface de queries SQL (AWS Athena)	Interface para execução de queries SQL no recurso de storage, para melhor aplicação de filtros	RF01 RFS01 RNF26
Gerenciador de recursos AWS (AWS Cloudformation)	Organização e gestão de recursos sob a forma de código (templates yaml) para melhor pipeline de CI/CD	RNF16 RNF19 RNF20 RNF21 RNF26
Gerenciador de workflow de deploy (GitHub Actions)	Deploy automatizado dos recursos desenvolvidos para a AWS (CI / CD)	RNF16 RNF21 RNF26

Tabela 3 - Recursos de arquitetura do Sistema UNDERLYING

A seguir será apresentado uma melhor descrição de cada um dos **8 recursos** definidos para garantir a qualidade do sistema:

1- **O Recurso de Storage (AWS S3)** foi selecionado para realizar o armazenamento de dados de séries históricas de ações e opções, o motivo dessa seleção está associado ao volume de dados armazenado e ao custo. Com o S3 é possível ter um baixo custo de armazenamento mesmo para um alto volume de dados, o que o torna viável para o projeto.

2- **O Recurso de API (AWS API Gateway)** foi selecionado para o desenvolvimento das apis do sistema, o motivo da seleção é o custo, a operação serverless e a fácil integração com os outros serviços da AWS. Com o API Gateway é possível desenvolver API's que são cobradas por volumes de requisição, não necessitando de um servidor disponível 24/7 para o consumo da api, além disso, esse recurso possui fácil integração com o sistema de autenticação que será utilizado (AWS Cognito) facilitando o desenvolvimento do produto.

3- **O Recurso de Banco de Dados (AWS DynamoDB)** foi selecionado para o armazenamento de dados das estratégias dos usuários, o motivo dessa seleção é a velocidade de resposta do banco de dados. Por um banco de dados não relacional atender as demandas do projeto, ele foi selecionado dada sua alta velocidade de operação, permitindo uma melhor experiência ao usuário.

4- **Recurso de funções serverless (AWS Lambda)** fará a execução serverless das funções em python/node desenvolvidas no sistema, realizando a integração com as apis para a geração da resposta, ou sendo invocadas por outras lambdas que necessitarem da função específica.

5- **O Recurso de gerenciamento de usuários (AWS Cognito)** foi selecionado para a realização de todas as tarefas relacionadas ao usuário, o motivo dessa seleção é a abstração e a implementação de aspectos de seguranças inerentes ao recurso. Com o AWS Cognito temos a implementação de requisitos de segurança como a criptografia de senhas e funções de recuperação de senha por email já embutidas no serviço o que facilitará o desenvolvimento do sistema e a integração com os outros recursos.

6- **O Recurso de interface de queries SQL (AWS Athena)** foi selecionado por fornecer uma interface de execução de queries SQL para consultas no S3. Como foi selecionado o S3 para armazenamento de informações históricas, o Athena permitirá a realização de consultas nessa base de dados, facilitando o acesso a esses dados.

7- **O Gerenciador de recursos AWS (AWS Cloudformation)** será utilizado para controle dos recursos utilizados na AWS, assim, cada serviço possuirá um template o qual serão listados todos os recursos utilizados, facilitando a manutenção e a atualização deles, melhorando o processo de CI e CD.

8- **O Gerenciador de workflow de deploy (GitHub Actions)** garantirá o deploy de cada serviço separadamente, trabalhando em auxilio com o cloud formation, todos os recursos e as funções serão lançadas para produção de modo automático, melhorando o processo de entrega.

4. ANÁLISE DE RNF

Nesta sessão será apresentado como serão abordados na arquitetura os RNF e para isso serão identificados os RNF, definidos os RAS e as ferramentas utilizadas: os frameworks, plataformas, bibliotecas e serviços externos utilizados no desenvolvimento do sistema para atender aos RNF.

Para o sucesso do sistema e atender as necessidades da persona Pedro foram definidos os principais atributos de qualidades requisitados. São eles: **Usabilidade, segurança, confiabilidade, desempenho e portabilidade**. A partir destes atributos e da entrevista foi possível então obter os RNF do sistema, que serão abordados a seguir.

4.1 Identificação dos RNF:

Nesta sessão serão apresentados os RNF cruciais para a execução do projeto. Todos os RNF possuem a classificação **Essencial, Importante** ou **desejável** da documentação (DRE e no RNF de produto) foram identificados para a sua devida análise e implementação como pode ser observado na tabela 4 de atributos de qualidade e RNF a seguir:

Atributo de Qualidade	Identificação	Requisito	Prioridade
Usabilidade	RNF01	Padronizar Interfaces	Essencial
	RNF02	Legendar gráficos	Importante
	RNF03	Validar dados de entrada	Importante
	RNF04	Abrir novas abas para links externos	Importante
	RNF05	Utilizar responsividade	Importante
	RNF06	Incluir Opções de Alteração visual	Desejável
	RNF07	Incluir opção de alteração de língua	Desejável
	RNF34	Incluir Glossário	Essencial
Segurança	RNF08	Criptografar Senhas	Essencial
	RNF09	Verificar Autenticidade	Essencial
	RNF10	Realizar Logout	Desejável
	RNF11	Permitir Apenas Senhas Fortes	Essencial
	RNF12	Autenticar Usuário	Essencial
Confiabilidade	RNF13	Desfazer ação de exclusão	Essencial
	RNF14	Desfazer ação de	Importante

		compartilhamento	
	RNF15	Interromper solicitação	Essencial
	RNF16	Apresentar Disponibilidade (99%)	Essencial
	RNF17	Apresentar Informações Reais, Atuais e Consistentes com o mercado de Opções	Essencial
	RNF18	Recuperar de Falhas	Essencial
Desempenho Eficiência	RNF19	Apresentar Tempo de Resposta Baixo (2S)	Desejável
	RNF20	Apresentar Tempo de Throughput Baixo (1S)	Desejável
	RNF21	Realizar Escalabilidade	Essencial
Compatibilidade	RNF22	Acessar em Diversos Navegadores	Essencial
Portabilidade	RNF23	Acessar por diferentes plataformas	Essencial
RNF-Organizacional	RNF29	Armazenamento de Dados	Essencial
RNF - Externo	RNF31	LGPD	Essencial
	RNF32	Integração com Sistemas Externos	Essencial

Tabela 4 - Atributos de qualidade e RNF

Obs: Os demais RNF aqui não retratos e que estão da DRE referem-se aos RNF organizacionais e aos RNF externos que não estão diretamente ligados a arquitetura do sistema.

4.2 Definição dos RAS:

Se faz necessário determinar COMO as partes do sistema (ou elementos de software) devem ser organizadas e quais são os níveis de qualidade que devem ser garantidos durante a execução dos requisitos funcionais, por isso nesta sessão serão definidos os RAS para a implementação da arquitetura.

Os RAS se diferenciam de outros requisitos no sentido que estes têm efeitos profundos na arquitetura de software de um sistema, ou seja, a arquitetura poderia ser dramaticamente diferente na ausência de tais requisitos. Nem todos os requisitos são relevantes para a arquitetura de um sistema. Para definir os RAS determinou-se anteriormente os principais requisitos de atributos de qualidade como usabilidade, segurança, confiabilidade,

desempenho, compatibilidade portabilidade, além de outros requisitos que possuem as maiores chances de afetar significativamente a arquitetura do sistema. Para determinar os RAS optou-se em escolher os RNF **conforme o nível de importância/relevância para o usuário final dos sistemas,** ou seja, **foram considerados os RNF classificados como essenciais e importantes** na DRE, **porém os RNF desejáveis não foram utilizados como critérios para definição do RAS.** Assim apresentamos a tabela com os RNF essenciais e importantes para a execução do projeto com as respectivas definições de RAS e identificadas as ferramentas que serão utilizadas para a implantação de cada RAS, conforme pode ser observado na tabela 5 seguir:

Identificação Do RNF	Requisito	Identificação do RAS	RAS	Implementação (Frameworks, Bibliotecas, Plataformas, Serviços)
RNF01	Padronizar Interfaces	RAS02	Todas as telas devem apresentam visual único (padronizado) e agradável.	Via código JavaScript
RNF02	legendar gráficos	RAS02	Todos os gráficos devem apresentam legendar de modo que facilite o entendimento do usuário	react-apexcharts
RNF03	Validar dados de entrada	RAS03	Os campos de input de usuário devem ser validados	AWS Cognito (Modo Usuário)
RNF04	Abrir novas abas para links externos	RAS04	Os links devem ser abertos em novas abas	nativo
RNF05	Utilizar responsividade	RAS05	A visualização dos dados deve ser responsável para que o usuário visualize o sistema independentemente do tamanho da tela	Reactstrap Nativo react-super-responsive-table
RNF34	Incluir Glossário	RAS06	Apresentar explicação de termos, palavras, expressões e conceitos mais utilizados no	Via código JavaScript

			mercado de opções	
RNF08	Criptografar Senhas	RAS07	Devera ser realizada a criptografia sobre as senhas de usuários	AWS Cognito
RNF09	Verificar Autenticidade	RAS08	Deverá ser solicitado a identificação da fonte para realizar qualquer alteração	AWS Cognito para criação do token e as validação das requisições da API será feita através deste token
RNF11	Permitir Apenas Senhas Fortes	RAS09	Realizar avaliaçao das senhas do usuário impedindo o cadastro de senhas fracas	AWS Cognito
RNF12	Autenticar Usuário	RAS10	Realizar o gerenciamento de usuário no sistema	AWS Cognito
RNF13	Desfazer ação de exclusão	RAS11	Deverá ser fornecida a opção para desfazer uma exclusão realizada por usuário	react-toastr
RNF14	Desfazer ação de compartilhamento	RAS12	Deverá ser fornecida a opção de desfazer um compartilhamento realizado por usuário	react-toastr
RNF15	Interromper solicitação	RAS13	Quando houver persistência na ausência de conexão (internet) ou houver queda imediata de energia o sistema não continuará com as ações solicitadas pelo usuário até que o mesmo acesse novamente online,	react-toastr

			faça login e requisite novamente	
RNF16	Apresentar Disponibilidade (99%)	RAS14	Possuir disponibilidade de 99,99% (não deve exceder 52,56 minutos indisponibilidade em 365 dias.	AWS Acordo de Nível de Serviço (SLA)
RNF17	Apresentar Informações Reais, Atuais e Consistentes com o mercado de Opções	RAS15	As informações devem ser consistentes e reais ao do mercado de opções	Requests e serviços da B3
RNF18	Recuperar de Falhas	RAS16	Deverá mediante a oscilação de rede continuar operante e realiza o processamento das ações solicitadas pelo usuário.	AWS S3
RNF21	Realizar Escalabilidade	RAS17	Se auto escalar ao receber milhares de solicitações de usuários manter a sua disponibilidade dos serviços.	AWS S3
RNF22	Acessar em diversos navegadores	RAS18	O usuário poderá realizar o acesso por diversos navegadores	nativo
RNF23	Acessar por diferentes plataformas	RAS19	Permitir o acesso de diferentes tipos sistemas operacionais.	Via código JavaScript
RNF29	Armazenamento de Dados	RAS20	Deverá possuir espaço em disco o suficiente para armazenar dados diversas vezes ao dia e permitir a	AWS DynamoDB

			pronta leitura dos dados.	
RNF31	LGPD	RAS21	Deverá atender a Lei 13.709 para garantir disponibilidade e privacidade dos dados	AWS Cognito
RNF32	Integração com Sistemas Externos	RAS22	De forma automática deverá busca por registros de opções em plataformas de cadastro da B3	Requests e serviços da B3
RNF33	Ético	RAS23	Não deverá apresentar dados privativos aos demais usuários.	AWS Cognito

Tabela 5 - Definições de RAS e das ferramentas de implementação

Os campos da tabela em azul claro acima serão de responsabilidades do frontend, enquanto os demais serão implementados no backend.

A relação dos Atributos de qualidade aos RAS ficou da seguinte forma como pode ser observado na tabela 6 a seguir:

Atributo de Qualidade	ID do RAS
Usabilidade	RAS01, RAS02, RAS03, RAS04 e RAS06
Segurança	RAS07, RAS08, RAS09 e RAS10
Confiabilidade	RAS11, RAS12, RAS13, RAS14, RAS15 e RAS16
Desempenho	RAS17
Compatibilidade	RAS18
Portabilidade	RAS19
Organizacional	RAS20
Externo	RAS21, RAS22, RAS23

Tabela 6 - Atributos de qualidade aos RAS

5. REPRESENTAÇÃO DAS VISÕES

Nesta sessão será apresentado e explicado as visões: arquitetural geral, de módulos, de *deployment* e a lógica.

5.1 Visão geral da arquitetura

Na divisão da arquitetura em camadas o sistema foi dividido em **6 camadas principais**. Elas são: (1) Consumer Layer, (2) camada externa, (3) deployment, (4) serviços de negócio, (5) serviços do sistema e (6) repositórios

1- **Consumer Layer**: Nesta camada teremos a disponibilização da plataforma que poderá ser acessada pelos usuários via web browser através dos recursos disponibilizados pela camada de negócio da aplicação.

2- **Externa** (B3 – Brasil Bolsa Balcão): Nesta camada teremos a divisão em dois componentes externos do sistema, de onde serão consultados os dados de opções (Derivatives Data) e ações (Stocks Data) esse serviço é mantido pela B3.

3- **Deployment** (Deployment Layer): Nesta camada teremos o gerenciamento do workflow que será responsável por realizar o deployment individual de cada serviço da camada de serviços de modo independente, implementando um pipeline no GitHub Actions que lançará os recursos da AWS para produção.

4- **Serviços de negócio** (Business services layer): Nesta camada teremos os métodos de disponibilização de recursos para os usuários, que poderão acessar a parte visual fornecida por uma aplicação web (Client Web Application) via web browser e a API caso queira consultas os dados brutos fornecidos pelos sistemas (HTTP API).

5- **Serviços do sistema**: Nesta camada teremos a divisão entre os microserviços do sistema, sendo cada um isolado na aplicação. A descrição desses serviços pode ser vista de modo detalhado no item 3.1.1 Serviços do Sistema, cada um desses serviços possuirá uma stack própria que manterá todos os recursos associados a ele.

6- **Repositórios** (Repositories Layer): Nesta camada teremos as bases de dados que serão utilizadas pelo sistema, sendo eles, dois componentes de storage (S3 de séries de opções e séries de ações), dois índices de Search engine (ElasticSearch com índices de registro de ações e opções para busca rápida e otimizada) e duas

tabelas de banco de dados não relacional (DynamoDB que manterá registros de usuários e estratégias)

Como pode ser observado na figura 1 a seguir são apresentados graficamente os elementos da arquitetura em camadas:

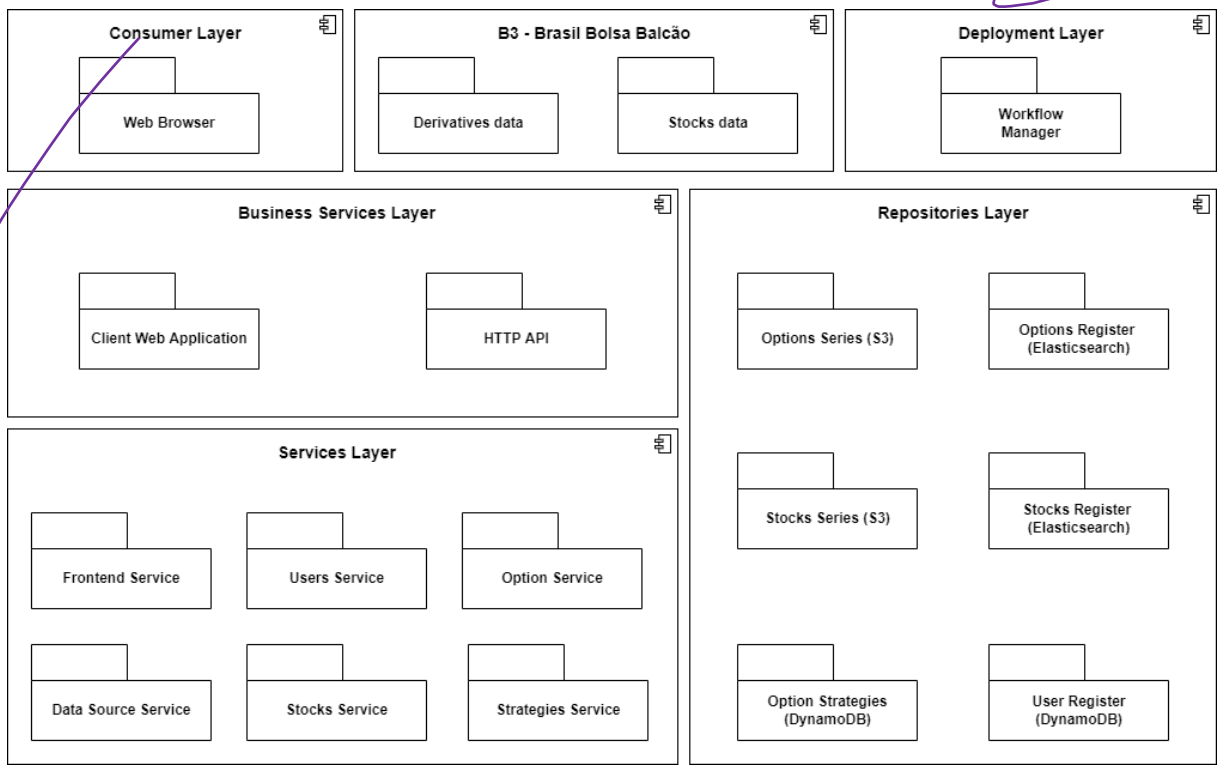

Figura 1 - Visão geral da arquitetura do sistema com as 6 camadas

5.2 Visão de módulos

A seguir teremos a decomposição de um sistema em partes menores denominadas módulos. As visões de módulos serão detalhadas a seguir, sendo cada uma acompanhada de uma descrição referente.

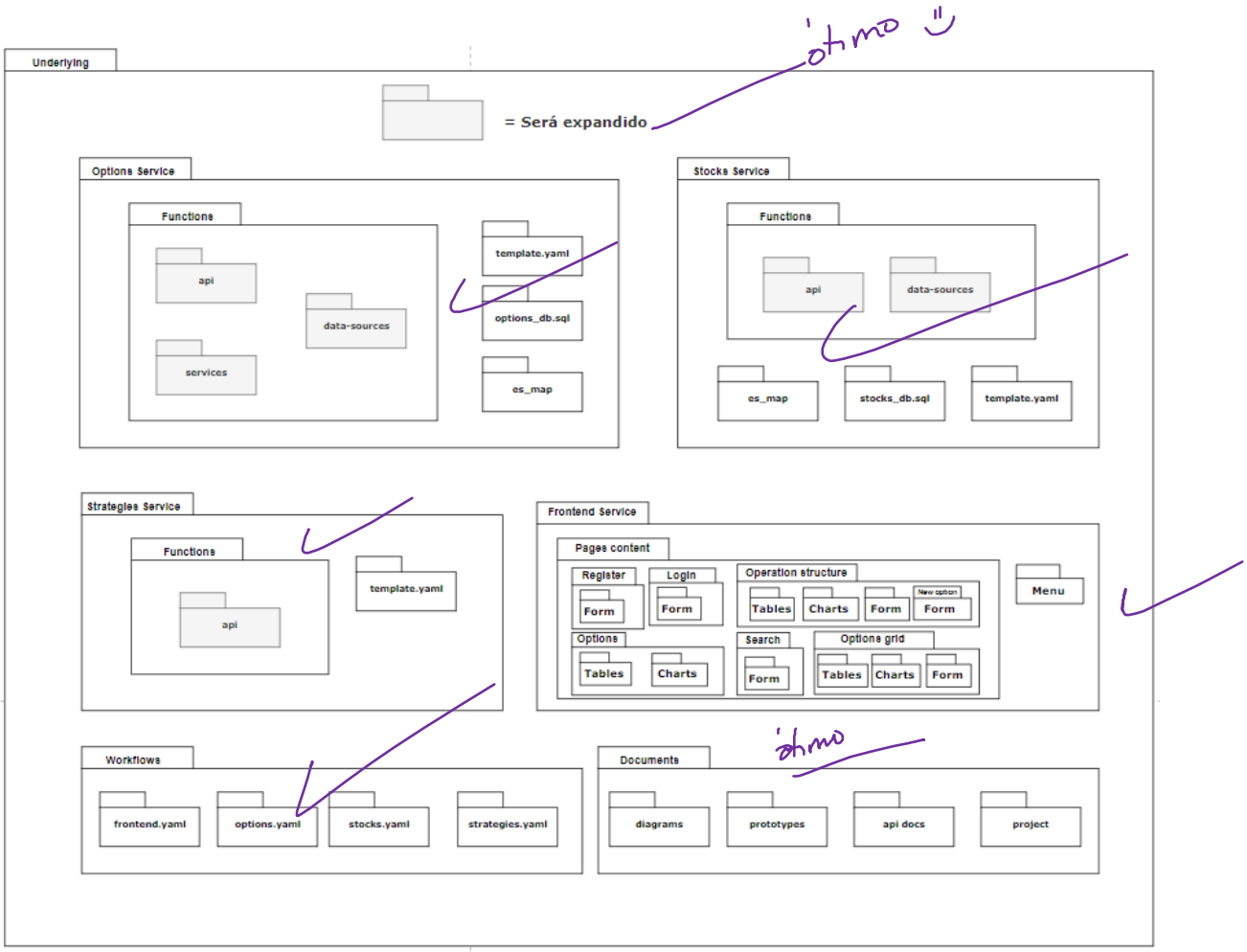

Figura 2 - Módulo principal

No **módulo principal,** como pode ser observado na figura 2, teremos a divisão dos serviços que irão compor nosso repositório resultando no sistema. O sistema será organizado em **seis componentes,** o de (1) opções, (2) ações, (3) estratégias, (4) frontend, (5) worflows e (6) documentos. A seguir será detalhado melhor os componentes:

1- O **serviço de opções** manterá no **diretório de funções**, os códigos referentes aos **três pilares principais** desse serviço, **a api** (funções e endpoints), o **data-source** (crawler e etl para processamento de dados da B3) **e o services** (funções auxiliares para cálculos relacionados a opções). Além disso, o serviço possuirá um arquivo de **template** que armazenará as os recursos utilizados para deploy na AWS via CloudFormation, **um arquivo SQL** com o mapeamento da base de dados para o Athena (que criará a tabela para consultas) e um **arquivo de mapeamento** que manterá informações de criação do índice utilizado no ElasticSearch.

2- **O serviço de ações** seguirá o mesmo padrão do serviço de opções com a exceção de que ele **não possuirá um módulo de services**, pois não serão calculadas informações extras referentes a ações.

3- No serviço de **strategies** teremos a disponibilização de uma **api** seguindo o mesmo padrão das opções, assim como a manutenção do **template** para deployment dos recursos na AWS, esse serviço **não possui os arquivos de mapeamento** já que ele não possuirá nenhuma Search Engine e nem banco de dados relacional associados.

4- O **frontend** conta com um menu compartilhado pelas páginas. As páginas de Registro, Login e Pesquisa são compostas por um formulário que enviará as informações para algum dos endpoints das APIs. A página de Opções é composta por tabelas e gráficos referentes a Opção desejada. A página de Grade de Opções é composta por tabela e gráfico de Ação, a tabela de grade de Opções e um formulário para inserção de Ação. Para finalizar os componentes do serviço de frontend temos a página de Estrutura de Operações, esta é composta por gráficos e tabelas das Opções selecionadas e gráfico de payoff, pelo formulário de busca de Opções para compor a Estrutura e a página/modal com um formulário para adição de opção fictícia.

5- No **workflows** teremos os arquivos referentes ao **deployment** de cada serviço do sistema, esses arquivos seguirão o padrão de script do GitHub Actions o que possibilitará o deployment individual de cada serviço, assim teremos os deployments **frontend, options, stocks e strategies**.

6- Por fim teremos um diretório para **manutenção da documentação** do sistema, que possuirá **diagramas arquiteturais desenvolvidos, protótipos implementados no figma, documentação das apis desenvolvidas e a documentação geral do projeto**.

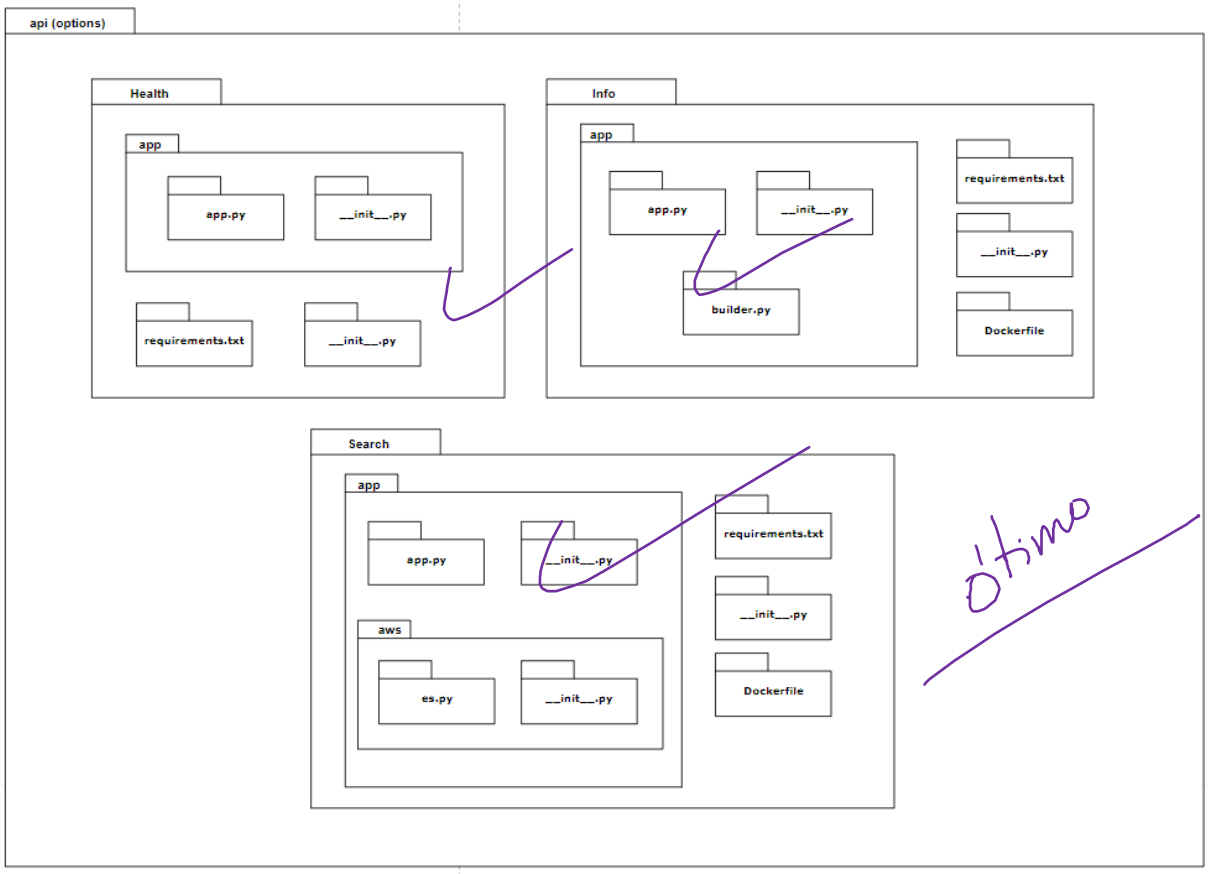

Figura 3 - Módulo de api das opções

No **módulo de api das opções**, como pode ser observado na figura 3, teremos **uma pasta para cada endpoint** sendo eles: **health** (para validação da disponibilidade da api), **info** (para consulta de dados de uma opção) e **Search** (para busca de opções por uma query). A estrutura se baseia em um **arquivo app.py** que possuirá a coordenação das funções e a resposta da api, os módulos auxiliares associados a AWS serão organizados em um **diretório aws** (exemplo es.py para utilização do ElasticSearch), todos os níveis de diretório devem possuir o arquivo **__init__.py** para identificação dos imports, além disso, toda camada mais externa deve possuir o arquivo **requirements.txt** com informações das bibliotecas versionadas utilizadas. O arquivo **Dockerfile** se faz necessário quando o conteúdo das bibliotecas ultrapassa o limite de memória possível por lambdas no s3, necessitando o encapsulamento desses em uma imagem no processo de deployment. O endpoint info contará com um arquivo **builder.py** que fará a montagem da resposta para a API já que esse endpoint trará diferentes tipos de informação.

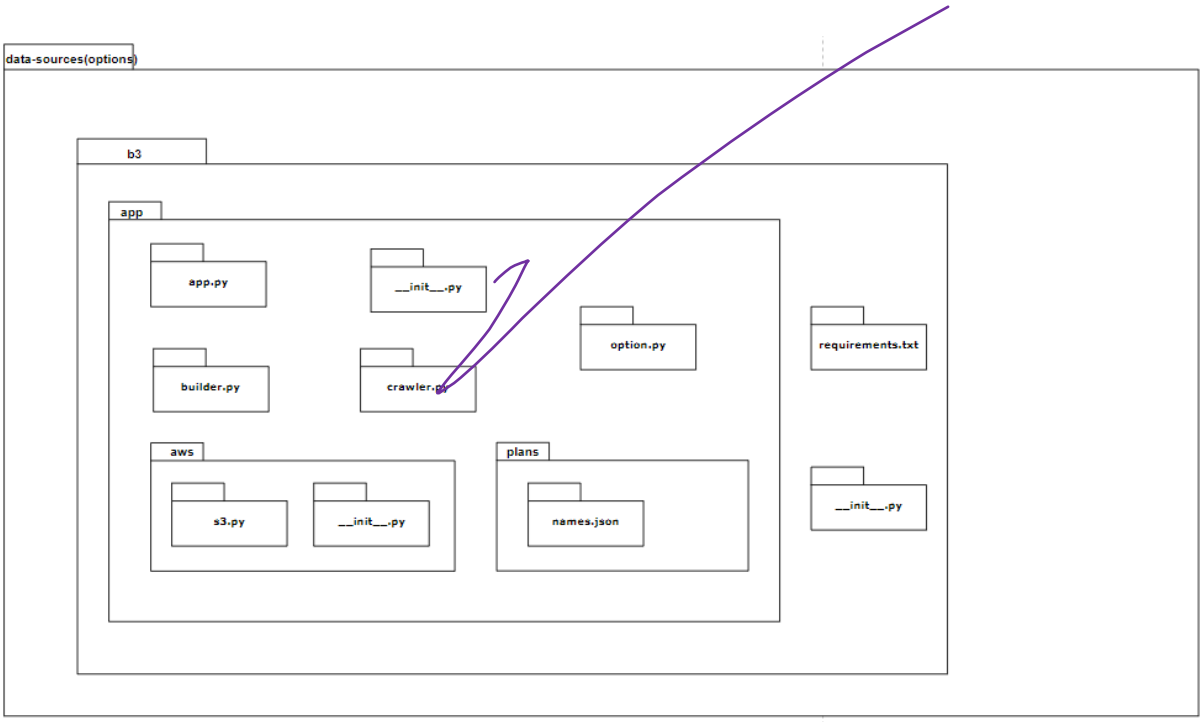

Figura 4- Módulo de data-sources de opções

No **módulo de data-sources de opções,** como pode ser observado na figura 4, teremos **o crawler** e o **tratamento dos dados coletados no serviço externo (B3**). O **app.py** fará a orquestração das funções. O **crawler.py** realizará a busca do txt no endpoint publico disponibilizado pela B3 e repassará essa informação para o **builder.py**. o builder realizará o tratamento dos dados e a organização deles em opções únicas, utilizando do **option.py** para enriquecimento das informações. O option.py irá **realizar** cálculos e gerar informações de greeks e métricas de mercado para serem salvas; no **diretório plans** teremos um mapeamento dos nomes das colunas disponibilizadas pelo **txt**, o builder utilizará esse arquivo para o cruzamento das informações brutas e seleção dos dados; por fim no diretório **aws** teremos a interface de comunicação com o s3 onde serão escritos os dados processados.

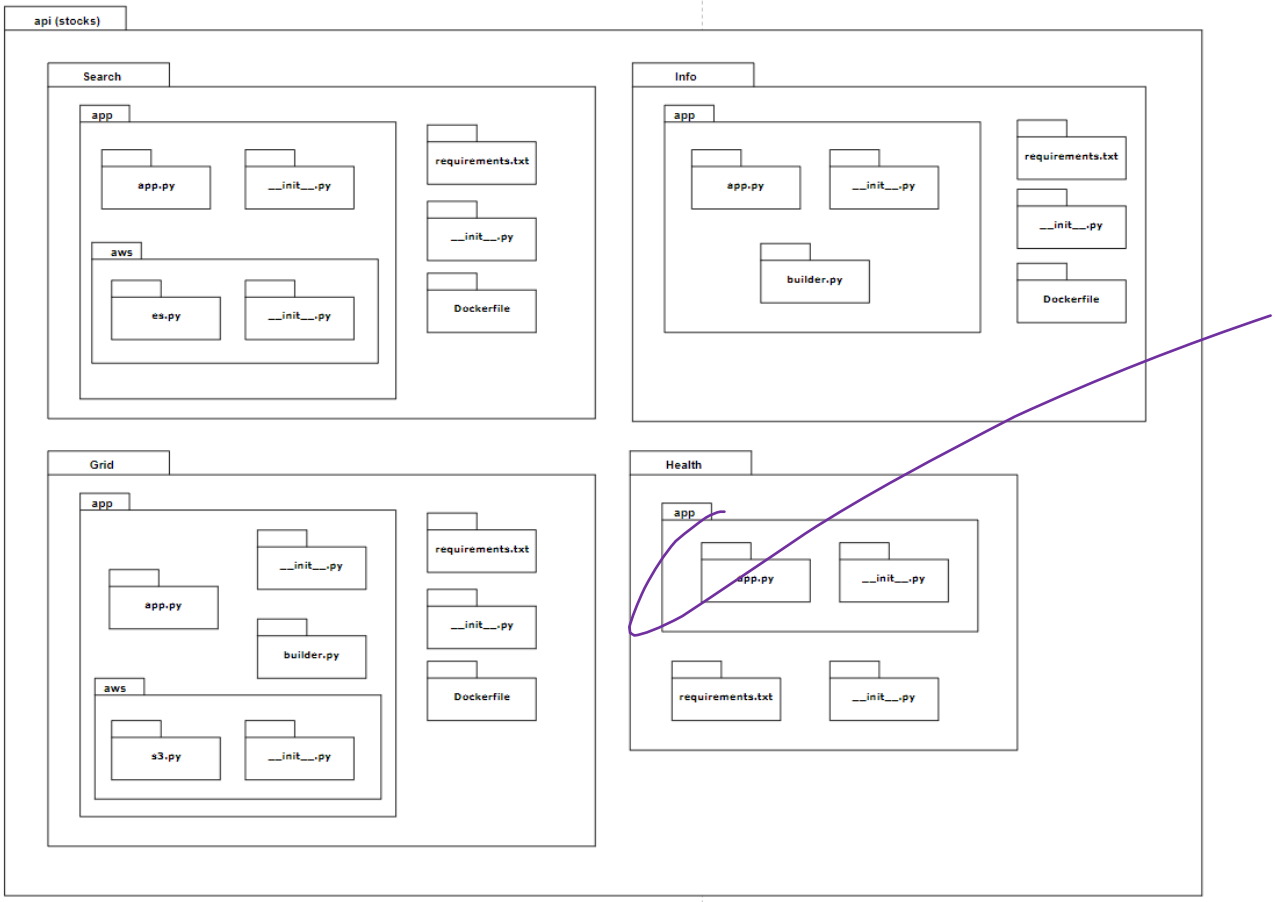

Figura 5 - Módulo de api das ações

Para o **módulo de api das ações,** como pode ser observado na figura 5, teremos a mesma organização do módulo de api das opções, com excesão da adição de um diretório referente ao **endpoint de grade de opções (grid)** que fará a consulta no s3 pela interface **s3.py** e organizará o retorno através do **builder.py** seguindo um padrão semelhante ao endpoint info de opções.

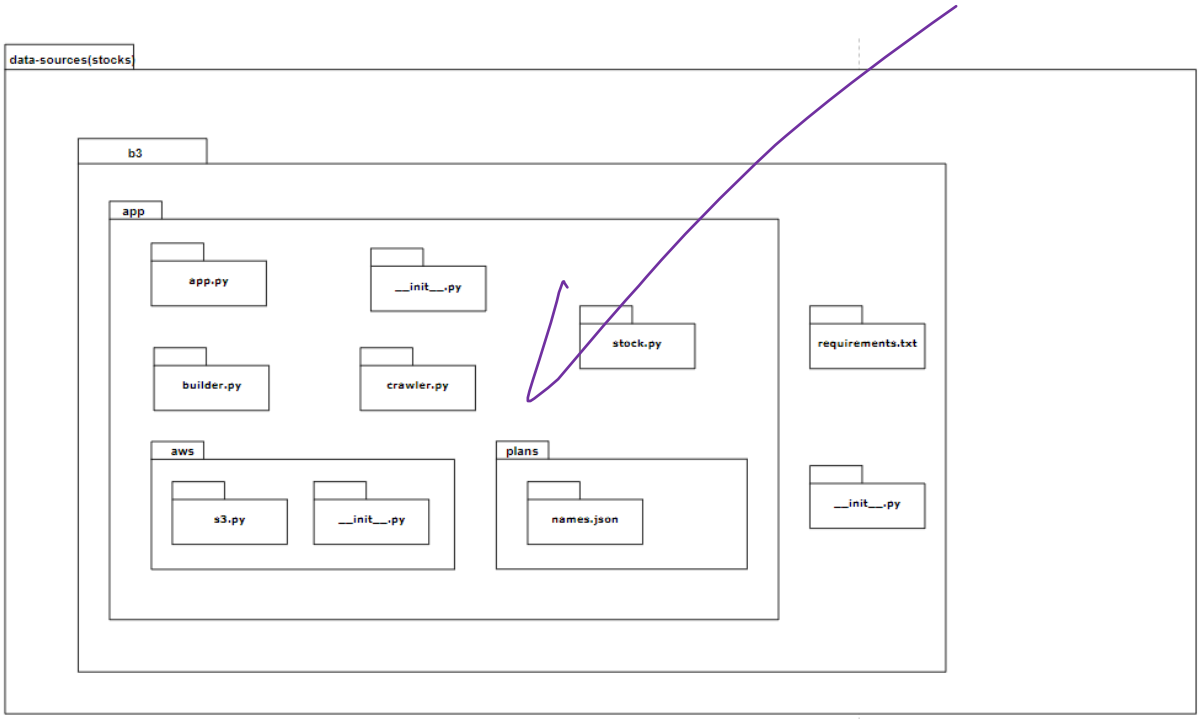

Figura 6 - Módulo de data-sources de ações

O **módulo de data-sources de ações,** como pode ser observado na figura 6, será uma réplica do modulo de opções com a diferença da **filtragem dos dados e da consulta no serviço exteno**, atributos esses **controlados pelo builder.py e pelo crawler.py**

Figura 7 - Módulo de api das estratégias

Por fim, temos o **módulo de api das estratégias,** como pode ser observado na figura 7, observamos que a organização entre os componentes desse módulo é semelhante, já que eles consultarão a mesma base de dados, apenas alternando a função executada no banco de dados, essas funções são identificadas no arquivo **app.py** e são executadas pela interface desenvolvida no **dynamo.py** do diretório aws.

5.3 Visão de deployment

Na visão de deployment temos a **demonstração de física dos recursos** que o sistema utilizará para seu desenvolvimento. Como foi proposta uma arquitetura **serverless na cloud**, todos os recursos utilizados compõem o sistema da AWS e estão divididos de acordo com o serviço.

No **serviço de opções** temos a divisão nos três módulos anteriormente detalhados, em nível físicos no módulo de api teremos um API Gateway para gerenciamento da api, sendo que cada endpoint possuirá um sistema de roteamento que direcionará a chamada para uma Lambda, responsável pelo retorno da chamada, a validação de autenticidade das chamadas será realizada por um Cognito Authorizer, sendo que a Lambda poderá fazer consultas as bases de dados, como o ElasticSearch no caso da rota /Search e ao Athena – S3 no caso da rota /info.

O serviço Externo (Provedor de dados: B3) utilizará o módulo de datasource de opções que apresentará um lambda para consumir uma fonte externa (B3) e salvando essas informações em duas bases de dados, o S3 para séries históricas e o ElasticSearch para registro, após a carga de dados uma tabela Athena é criada a partir de um Crawler Glue.

No módulo de **serviços de opções** temos apenas um lambda responsável pelo cálculo de payoff das opções, essa lambda será invocada por outros serviços, como o de estratégias (para cálculo de payoff de estratégia) e o próprio de opções.

Para o **serviço de estratégias** teremos apenas o módulo de API, que seguirá os mesmos padrões estabelecidos pela api de opções, a diferença nesse caso é que as consultas, remoções, edições e inserções serão realizadas em uma base de dados NoSQL da AWS, o DynamoDB.

Para o **serviço de ações** temos uma arquitetura semelhante ao serviço de opções com a diferença do endpoint adicional /grid que responderá com a grade de opções de uma ação, e para isso necessitará da realização de consultas em dois S3, o de opção de o de ação para cruzamento dos dados.

Para os **serviços do usuário** teremos o armazenamento do código fonte no S3, esse serviço realizará a comunicação com todas as APIs do sistema e fará o controle dos usuários através da comunicação com o serviço de usuários da AWS, o Cognito, o código será hospedado a partir do serviço de hospedagem embutido no S3, que faz o controle do tráfego de usuários.

Vale salientar que o **workflow de deployment** utilizará do **GitHub Action** para CI/CD realizando o deployment individual a partir de um arquivo yaml com as instruções necessárias, no caso da AWS, esse arquivo irá fazer o build do template **contido em cada**

serviço que resultará na subida dos recursos para o ambiente de produção, criando uma stack para cada serviço a partir do template transformado em recurso pelo cloudformation.

A seguir são apresentados graficamente a visão de deployment como pode ser observado na figura 8:

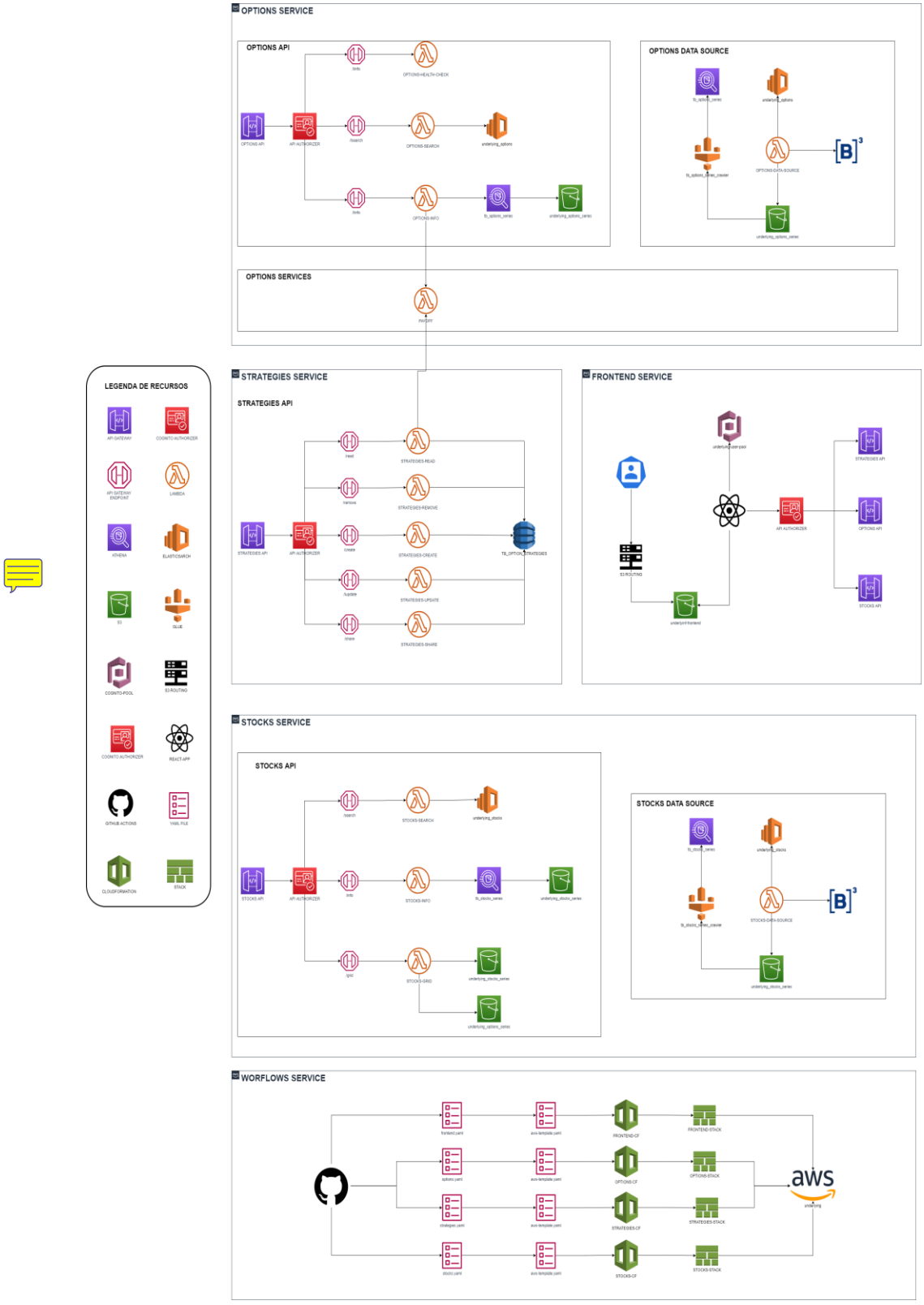

Figura 8 - Visão de deployment

5.4 Visão lógica

Nesta seção será apresentada as entidades de dados do sistema. Os acessos aos dados são fundamentais para a qualidade do sistema, uma vez que, se concentram os métodos de acesso, como consulta direta, extrações e serviços de dados e também a proteção de dados que incluem precauções de segurança de perímetro, métodos de criptografia e controles de acesso baseados em funções e atributos. A infraestrutura de dados estabelece além da definição dos dados as ferramentas a serem utilizadas para os serviços de gerenciamento de dados e serviços de armazenamento.

Assim para a conformidade dos RNF, RF e RA a fim de atender o **gerenciamento e uso de dados do sistema** foram definidas a utilização da arquitetura de **ferramentas da AWS** para atender aos principais atributos de qualidade que garantem as especificações do sistema e foram estabelecidas **7 entidades** para o sistema. Temos as seguintes entidades: (1) User, (2) Strategy, (3) StrategyOption, (4) Option, (5) OptionSerie, (6) Stock, e (7) StockSerie. Estas entidades serão melhor apresentadas a seguir com os respectivos recursos (ferramentas) de armazenamento da AWS que garantem principalmente a disponibilidades, segurança, e atendem as propriedades ACID (**A**tomicidade, **C**onsistência, **I**solamento e **D**urabilidade):

1- **User**: dados dos usuários armazenados pelo **cognito.**

2- **Strategy:** informações de estratégias desenvolvidas pelos usuários e armazenadas no **dynamoDB.**

3- **StrategyOption**: objeto de opção que fará parte da composição de uma estratégia e será armazenado no mesmo objeto do **DynamoDB.**

4- **Option**: informação de registro de opção armazenada no **ElasticSearch** para busca de opções.

5- **OptionSerie**: dados diários históricos de opções acompanhados pelas greeks e métricas de mercado previamente calculados e salvos no **S3**.

6- **Stock**: informação de registro de ação armazenada no **ElasticSearch** para busca de ações.

7- **StockSerie**: dados históricos de ações com foco nas transações de mercado realizadas diariamente e serão armazenadas no **ElasticSearch.**

Em termos de relacionamento temos apenas a entidade strategy se relacionando com outras classes do sistema, essa entidade poderá possuir n entidades de StrategyOption que irão compor uma estratégia, sendo que cada estratégia possuirá um usuário único referente, o que justifica o relacionamento com a entidade User.

A seguir são apresentados graficamente a visão logica dos dados, como pode ser observado na figura 9:

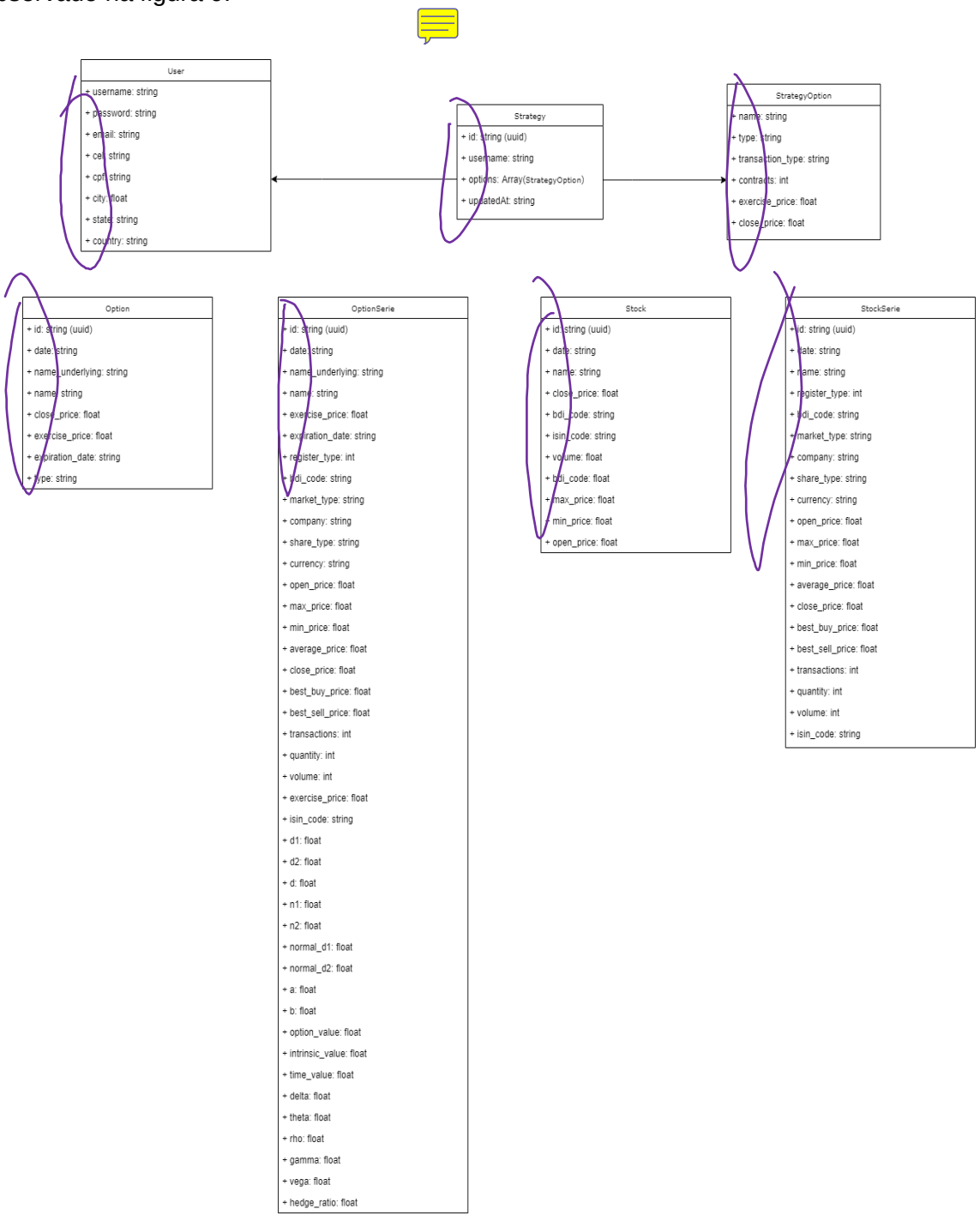

Figura 9 - Visão de lógica dos dados

6. REFERÊNCIAS:

A seguir são apresentados os seguintes artefatos que foram de grande importância para o desenvolvimento deste projeto.

I. **Underlying - DRE**
II. **Underlying - RNF de Produto – Atributos de qualidade**
III. **https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/47/210/431-1?inline=1**
IV. **https://aws.amazon.com/pt/architecture/**
V. **https://aws.amazon.com/pt/architecture/well-architected/?achp_wa1&&wa-lens-whitepapers.sort-by=item.additionalFields.sortDate&wa-lens-whitepapers.sort-order=desc**
VI. **https://aws.amazon.com/pt/well-architected-tool/?achp_wa2&whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc**
VII. **https://linamgr.github.io/SIGEAN/**