PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA
UNIDADE EDUCACIONAL PRAÇA DA LIBERDADE
Bacharelado em Engenharia de Software

Gabriel Santana

Douglas Soares da Cunha

Rafael Araújo Badaró

Trabalho de RabbitMQ

Belo Horizonte

2019

Introdução

Este projeto é uma implementação em Python de uma aplicação de publish-subscribe com RabbitMQ.

Classes e operações

- broker.py
    - Representa os brokers/corretoras que podem realizar as compras, vendas, info e assinar os tópicos de ações que desejam escutar.
    - Operações:
        - realiza_operacao(operacao)
            - Realiza as operações de compra e venda
            - Publica em um fila chamada 'BROKER' uma string 'mensagem' que contém a rota (operacao.acao), quantidade, valor e corretora.
        - receber_notificacoes()
            - Assina em tópico as ações para escutá-las
            - Declara uma exchange chamada BOLSADEVALORES
            - Cria uma fila, realiza um bind para cada binding_key (que segue um padrao: *.acao) e consome dessa fila.
        - montar_conteudo()
            - Recebe um vetor de string e monta mensagens a serem trocadas

- bolsa.py
    - Representa a bolsa de valores que envia notificações aos brokers sobre alguma ação e recebe operações dos mesmos.
    - Operações:
        - enviar_notificacoes(topicos)
            - Monta uma rota(operacao.acao) e envia uma mensagem com os dados da operação.
        - receber_operacoes()

            - Declara uma fila chamada ‘BROKER’ e consome essa fila.
        - montar_conteudo(topicos: str[])
            - Concatena a quantidade, valor e nome do broker em uma string
        - salvar_operacao(operaca_acao: str, oferta: str)
            - Concatena a operacao_acao com a oferta e chama o método registrar_oferta()
        - realiza_info(operaca_acao: str, data: str)
            - Concatena a operacao_acao com a da realiza a consulta no livro de ofertas.

- livro_de_ofertas.py
    - Armazena as operações de compra e venda e realiza as transações.
    - Operações:
        - carrega_ativos()
            - Carrega na aplicação uma lista de objetos do tipo Ativo, sobre os quais as operações serão feitas
        - ver_ofertas_de_compra(oferta: String)
            - Acessa a lista de ofertas de compra e procura por todas que batem com uma oferta de venda
        - ler_ofertas_de_venda()
            - Acessa a lista de ofertas de venda e procura por todas que batem com uma oferta de compra
        - registrar_oferta(oferta: String)
            - Registra oferta de compra/venda no livro de ofertas
        - atualiza_oferta(oferta: String)
            - Atualiza um oferta de compra/venda
        - remover_oferta(oferta: String)
            - remove uma oferta de compra/venda do livro de ofertas
        - realizar_transacao(oferta: String)

            - Recebe uma determinada oferta de compra/venda e caso ela bata com outra oferta realiza a transação das ofertas.
        - consultar_transacao(transacao: String)
            - Consulta por uma determinada transação

- transacoes.py
    - Armazena as transações que foram realizadas e as envia para os brokers
    - Operações:
        - enviar_transacao(transacao: str)
            - Envia a transação realizada para todos os brokers que escutam ela
        - gravar_transacao(transacao: str)
            - Armazena a transação na lista de transações
        - ler_transacao(data : str, acao: str)
            - Lê a lista de transações procurando a data e a ação referentes

- main.py
    - Responsável por rodar a aplicação
    - Responsável por ficar ouvindo as operações que o usuário quer fazer (compra, venda etc.)