Proposta do Projeto Final - Sistemas Distribuídos

1. Proposta Geral da Aplicação (Cenário de Uso)

O projeto propõe o desenvolvimento de uma **plataforma distribuída de processamento assíncrono de pedidos**, inspirada em sistemas amplamente utilizados no mercado, como plataformas de delivery e marketplaces digitais (imagine um iFood, só que bem básico, para fins meramente didáticos).

O sistema tem como objetivo principal demonstrar, de forma prática, a aplicação dos conceitos fundamentais de **Sistemas Distribuídos** abordados em sala, tais como desacoplamento, comunicação assíncrona, escalabilidade e tolerância a falhas. Além disso, o sistema também demonstrará as tecnologias que foram abordadas individualmente por cada aluno em sala (descritas ao decorrer do documento), mostrando um projeto real com essas tecnologias trabalhando juntas.

O sistema disponibiliza uma **API REST** por meio da qual clientes autenticados podem realizar operações de login e criação de pedidos. Cada pedido representa uma solicitação de serviço que demanda múltiplas etapas de processamento, não sendo adequada a execução síncrona durante a requisição HTTP.

Após a criação do pedido, a API publica um evento em um sistema de mensageria baseado em **Apache Kafka**, que atua como intermediário entre a camada de entrada (API) e os componentes responsáveis pelo processamento. Os pedidos são então consumidos por serviços distribuídos que executam tarefas assíncronas por meio do **Celery**, tais como validação de dados, simulação de disponibilidade, processamento de pagamento e atualização do status do pedido.

O usuário pode consultar o estado de seu pedido posteriormente, garantindo uma experiência responsiva e eficiente, mesmo sob alta carga de requisições.

2. Tecnologias Adotadas

As tecnologias selecionadas para o desenvolvimento do projeto foram escolhidas com base em sua ampla utilização no mercado e adequação ao contexto de sistemas distribuídos.

Linguagens de Programação

- **Python** (linguagem principal do sistema)
- **SQL** (consultas e definição do banco de dados)
- **YAML** (configuração de infraestrutura)
- **Shell Script (Bash)** (automação de execução)

Frameworks e Bibliotecas

- **Flask**: desenvolvimento da API REST
- **Celery**: processamento assíncrono de tarefas
- **PyJWT**: geração e validação de tokens JWT
- **SQLAlchemy**: acesso ao banco de dados
- **Kafka-Python**: integração com Apache Kafka

Infraestrutura e Ferramentas

- **Apache Kafka**: mensageria e comunicação baseada em eventos
  Redis ou RabbitMQ: broker de mensagens do Celery
- **PostgreSQL**: banco de dados relacional
- **Docker e Docker Compose**: containerização e orquestração local
- **Git e GitHub**: versionamento e colaboração

Protocolos

- **HTTPS**
- **TCP/IP**

3. Justificativa e Relação com a Disciplina de Sistemas Distribuídos

O projeto proposto apresenta forte aderência aos conteúdos abordados na disciplina de **Sistemas Distribuídos**, permitindo a aplicação prática de conceitos teóricos fundamentais.

Entre os principais tópicos contemplados, destacam-se:

- **Comunicação assíncrona entre processos distribuídos**, por meio de mensageria (Kafka);
- **Arquitetura orientada a eventos**, promovendo baixo acoplamento entre componentes;
- **Processamento concorrente e paralelo**, utilizando workers distribuídos com Celery;
- **Escalabilidade horizontal**, permitindo a adição de múltiplos consumidores e workers;
- **Tolerância a falhas**, uma vez que falhas isoladas não interrompem o funcionamento global do sistema;
- **Segurança em ambientes distribuídos**, com autenticação e autorização via JWT;
- **Separação de responsabilidades**, com serviços independentes e bem definidos.

Dessa forma, o projeto proporciona uma visão prática dos desafios e soluções adotados em sistemas distribuídos modernos.

4. Equipe e Responsabilidades Técnicas Individualizadas

A equipe será composta por **cinco integrantes**, com responsabilidades técnicas bem definidas, visando organização, paralelismo no desenvolvimento e melhor integração entre os componentes do sistema.

Clebson Santos e Wallan - Arquitetura e Coordenação Técnica

- Definição da arquitetura geral do sistema
- Padronização do código e boas práticas
- Integração entre os serviços
- Apoio técnico aos demais integrantes

João Marcos - API REST (Endpoints e Lógica de Negócio)

- Implementação da API REST
- Criação e manutenção dos endpoints relacionados a pedidos
- Validação de dados recebidos nas requisições
- Integração da API com o sistema de mensageria (publicação de eventos no Kafka)
- Padronização das respostas HTTP e códigos de status
- Documentação da API (ex.: Swagger/OpenAPI)

Clebson Santos - Autenticação e Autorização (JWT)

- Implementação do mecanismo de autenticação baseado em JWT
- Criação do fluxo de login e geração de tokens
- Validação de tokens em endpoints protegidos
- Definição de permissões e regras de acesso
- Implementação de middlewares/decorators de segurança
- Apoio à segurança geral da API

Natan e Wallan - Mensageria e Eventos (Kafka)

- Configuração do Apache Kafka
- Criação e gerenciamento dos tópicos
- Implementação dos produtores de eventos
- Implementação dos consumidores de eventos

Rafael Lima - Processamento Assíncrono (Celery)

- Configuração do Celery e do broker de mensagens
- Implementação das tarefas assíncronas
- Integração entre Kafka e Celery
- Monitoramento básico das tarefas

Clebson Santos e Wallan - Infraestrutura, Banco de Dados e Testes

- Modelagem e configuração do banco de dados
- Criação de scripts SQL
- Configuração do Docker e Docker Compose
- Testes de integração e documentação técnica

5. Repositório do Projeto

Todo o código-fonte, documentação, diagramas de arquitetura e artefatos produzidos ao longo do desenvolvimento do projeto serão versionados e disponibilizados em um repositório público no GitHub.

Link do repositório: https://github.com/Natan-ls/processamento-assincrono-pedidos

Observação:

A integração entre os componentes do sistema será realizada por meio de containers Docker, orquestrados com Docker Compose, permitindo a execução distribuída e isolada da API REST, do Apache Kafka, dos workers Celery e dos serviços de apoio, como banco de dados e broker de mensagens.

Levando em conta que essa detalhação está sendo feita com mais de um mês de antecedência em relação à entrega efetiva do projeto final, existe a possibilidade de que alguns pequenos detalhes sejam alterados ao decorrer do desenvolvimento, conforme a equipe julgar necessário. Porém, a ideia central, assim como a equipe desenvolvedora e as principais tecnologias, será mantida.