Especificação de proposta de arquitetura

Autor: Mauricio Walther Souza Guzzi – 22/10/2025

Destinatário: Act (https://actdigital.com/pt/) – *Conforme solicitado por meio de Evelin Brito*

Objetivo: Proposta de Arquitetura para Sistema de Integração e Orquestração (desafio) – versão *Beta*

Contexto

A missão recebida é propor uma arquitetura para um sistema que atua como hub de integração entre sistemas terceiros, realizando:

- Recebimento de requisições externas (via APIs ou mensagens);
- Processamento interno com regras de negócio;
- Envio de dados para outros sistemas;
- Gerenciamento de estados e workflow das requisições.

Este sistema será responsável por garantir resiliência, rastreabilidade, observabilidade e escalabilidade.

Objetivo do desafio

Você deve propor uma arquitetura que atenda aos requisitos funcionais e não funcionais descritos acima, considerando boas práticas modernas de desenvolvimento e operação de sistemas distribuídos.

A proposta deve conter:

- Diagrama de arquitetura (alto nível);
- Justificativas técnicas para as escolhas feitas;
- Estratégias para garantir estabilidade, rastreabilidade e segurança;
- Pontos de atenção e riscos identificados.

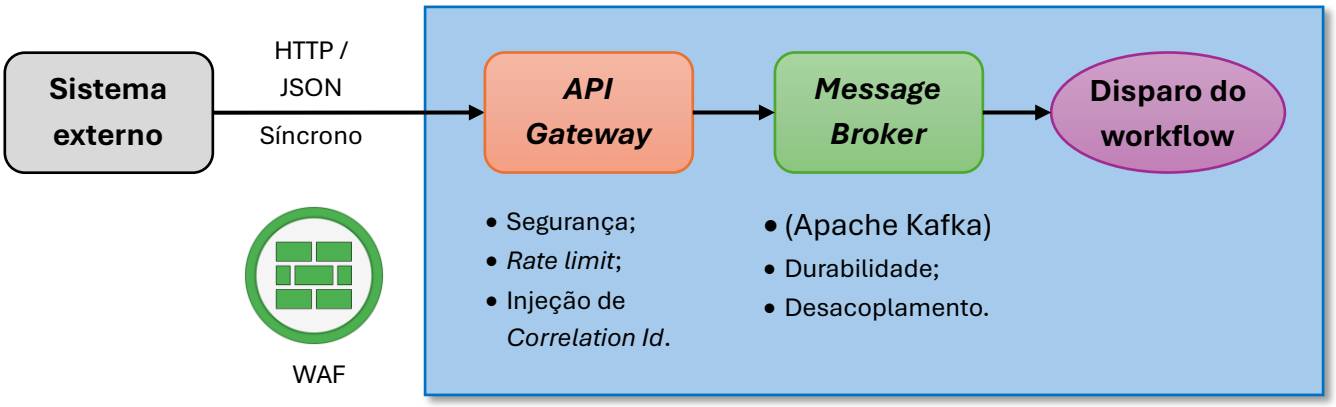
Requisitos não funcionais esperados

- Alta disponibilidade e tolerância a falhas;
- Observabilidade com logs estruturados e métricas;
- Escalabilidade horizontal;
- Segurança na comunicação entre sistemas;
- Facilidade de manutenção e evolução.

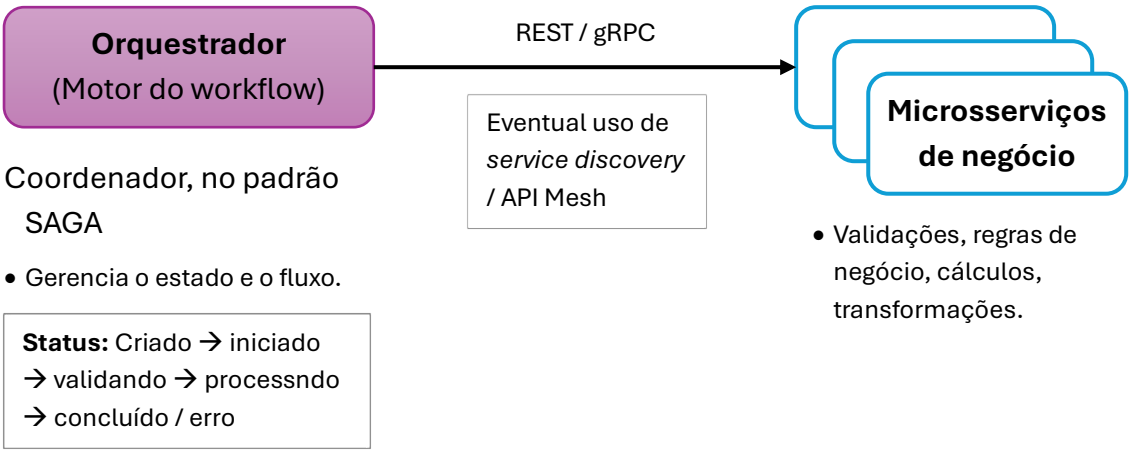
Proposta de arquitetura

Diagrama

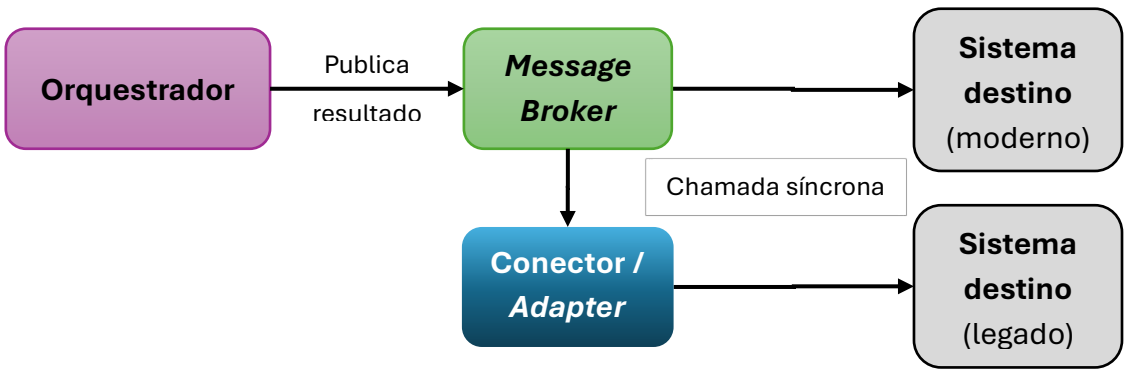
1. Camada de entrada (fontes de dados)



2. Camada de controle e orquestração



3. Saída para outros sistemas



Descrição

1. Visão geral dos conceitos arquiteturais

Nesta arquitetura, usamos 3 conceitos principais: o **padrão de microsserviços**, a **orientação a eventos** (*event-driven*) e uma **orquestração no padrão SAGA**. A tabela abaixo mostra a justificativa:

Padrão (*pattern*)	**Função**	**Benefício**
Arquitetura orientada a eventos	Prover desacoplamento	Para que a falha em um sistema/componente não quebre a cadeia de valor.
Microsserviços	Processar regras	Regras de negócio podem ser desenvolvidos e implantados de forma independente.
Orquestração SAGA	Gerenciar o workflow	A lógica do fluxo de trabalho, mesmo um complexo, fica centralizada; e o fluxo se mantém consistente por ser tratado de forma transacional.

2. Descrição dos componentes chave

A. Camada de entrada

Componente	**Função**	**Padrões usados (*patterns*)**
API Gateway	Atuar como ponto de entrada único para todas as requisições externas assíncronas (APIs). Neste componente também efetuamos a autenticação do sistema cliente que faz a requisição, *rate limiting* (evitando DOS e sobrecargas), e transformações básicas.	*API Gateway pattern*, *Façade*
Message Broker	Receber e armazenar os eventos de forma durável, blindando contra perda dos dados.	*Message Queue pattern*, *publish/subscribe*

B. Camada de controle e orquestração de processamento

Componente	**Função**	**Observação**
Orquestrador *(motor do workflow)*	Recebe as requisições (vindas do API Gateway ou da Fila), gerencia seu *status*, e executa cada etapa do workflow. O padrão SAGA garante a consistência do fluxo por comandar uma transação distribuída, atada por um *Correlation Id* único inserido na camada de entrada.	Fluxo: 1. Recebe a requisição (início do fluxo) → 2. Envia aos microsserviços; → 3. Aguarda processamento e colhe o status; → 4. Atualiza status.
Microsserviços de negócio	Cada microsserviço é uma unidade de processamento leve e independente, encapsulando regras de negócio específicas (p. ex.: validação de dados, cálculos, transformações etc.). Eles são acionados pelo Orquestrador e retornam o status do seu processamento.	Recomenda-se a separação dos "microsserviços" de modo que cada um cuide de uma responsabilidade específica. (Princípio de *single responsability*.)

C. Camada de saída

O envio de dados para outros sistemas, por ser síncrono, vai garantir a resiliência do Hub.

Sequência de processamento:

1. O Orquestrador finaliza o processamento e publica um evento no *Message Broker*.
2. Os sistemas destino se inscrevem nos tópicos da fila que sejam relevantes para eles.
3. Sistemas de destino que trabalhem com APIs (que estamos chamando de "legados") são acionados por um *Adapter*, o qual retira eventos da Fila e os converte para chamada de API.

3. Como os requisitos não funcionais (NFRs) estão sendo atendidos:

Requisito NF	Solução arquitetural	Detalhes
Resiliência	Desacoplamento assíncrono e Padrão SAGA	O uso de um *Message Broker* evita o acoplamento temporal da invocação. O SAGA com compensação reverte os passos de uma transação em caso de falha, garantindo sua consistência.
Alta disponibilidade	*Failover* / *Load Balancing* / *Circuit Braker* / *Auto-scaling* / Replicação (talvez geográfica)	As escolhas arquiteturais precisam levar em conta o ambiente, o SLA e os custos.
Escalabilidade	Microsserviços + orquestração	Cada microsserviço pode ser escalado horizontalmente de modo independente de acordo com sua carga (eventualmente de modo elástico). O uso de um orquestrador (ex.: Kubernetes) automatiza esta mecânica.
Rastreabilidade	*Correlation Id* + observabilidade	A criação e propagação de um Id único (UId) permite uma ligação rastreável em toda a cadeia.
Observabilidade	Centralização de Logs e Métricas / Instrumentação / Telemetria	A monitoração de status, fluxo de dados, desempenho e saúde do ambiente pode ser delegada a uma *stack* dedicada (p.ex.: Prometheus / Grafana / Dynatrace / ELK etc.).

Tecnologias sugeridas para o Hub de Integração

As sugestões abaixo procuram priorizar soluções open-source, mas que tenham reputação como sendo altamente escaláveis e como tendo forte apoio da comunidade.

Componente	Objetivo	Tecnologia sugerida	Observações
API Gateway	Ponto de entrada, segurança e *rate limiting*.	Kong (open-source), ou o disponibilizado pela nuvem.	A opção *cloud-native* será escolhida se a infra estiver na nuvem (ex.: Azure APIM).
Messageria e *Broker*	Desacoplamento e resiliência assíncrona.	Apache Kafka	O Kafka é ideal para alto volume. Cenários mais simples suportariam um RabbitMQ.
Orquestração de fluxos	Gerenciamento de estado, fluxos e SAGA.	Temporal.io ou Camunda	Simplificam a escrita de workflows e garantem a durabilidade do estado.
Microsserviços	Onde implementa-se a lógica de negócio.	Java (Spring Boot) ou C# (.NET)	A escolha do stack leva em conta a experiência do time.

Orquestração (microsserviços)	Gerenciamento de *containers* / *autoscaling*.	Docker + Kubernetes	Padrão da indústria; disponíveis em ambientes on-premises ou em nuvem.
Banco de dados (orquestrador)	Armazenamento do status do workflow.	PostgreSQL (nuvem: CosmosDB / DynamoDB / etc.)	Produtos robustos e que suportam NoSQL.
Observabilidade: logs	Rastreabilidade e *troubleshooting*.	ELK *stack*: Elasticsearch + Logstash + Kibana	Coleta, indexação e visualização de logs. Rastreabilidade através de *Correlation Id*.
Observabilidade: métricas	Monitoramento de performance e saúde.	Prometheus + Grafana	Coleta de métricas em *real time*, e visualização em *dashboards* configuráveis.