Avaliação de Arquitetura de Soluções – Comércio

# Objetivo

Criar Arquitetura de Solução de Sistema para o Comércio, de acordo com os requisitos recebidos no Arquivo “desafio-arquiteto-solucao-ago2024”.

# Considerações iniciais

A princípio, a primeira avaliação dos requisitos leva o Arquiteto a considerar a implantação de um ERP de mercado, porém, como o intuito do teste é avaliar o conhecimento de Arquitetura e o relacionamento com áreas técnicas e de negócios. Para tanto a solução aplicada foi de um sistema rodando em Cloud, com arquitetura corporativa baseada em domínio (Financeiro) e Arquitetura de Solução baseada em Eventos, seguindo as premissas abaixo listadas:

- O desenho da solução termina no nível Componentes do C4: a justificativa é que, por se tratar de uma avaliação, e para não dispender tempo com soluções relativas à Arquitetura de Software/Engenharia e Desenvolvimento, as quais não são necessariamente relevantes para a avaliação de um Arquiteto de Soluções, que tem a visão global. Portanto, detalhes de sequencias, DTO’s, Swaggers, MVC’s, SOLID e outros itens relacionados à Arquitetura de Software, não foram desenhados.

- A parte relativa ao transacional foi feita utilizando uma visão mais simplista de uma transação que envolve Banco e outras entidades autenticadoras, porém, para uma simulação real, seria necessário mais um controlador de eventos e possíveis status para os pagamentos e outras validações. Esta parte foi abstraída, para não aumentar a complexidade da solução. Porém, fica anotado que para o funcionamento, em contexto real desta aplicação, este “módulo” de eventos seria necessário.

-Por falta de definição estratégica e direcionamentos, foi desconsiderada a preexistência de módulos iniciais, sejam eles “Cloud” ou “on Premisses”, além de outros componentes, servidores, etc. Portanto, foi criada uma aplicação Cloud nova, considerando um Portal para acesso e toda essa aplicação está baseada na AWS (poderia ser em outra cloud, enfim), utilizando recursos nativos de tal ambiente.

- Por fim, também foi considerada somente, em contexto de infra, os recursos mínimos para tal e a questão relativa à redundância, deve ser tratada com espelhamento em regiões distintas, bem como um a infra enxuta para um disaster recovery.

- O desenho da solução foi elaborado, também por questões de simplificação, que o ator Gestor realiza todas as transações, seja ela débito, crédito ou relatórios. Essa suposição se faz necessária, pois entende-se, pelo requisito, que é ele quem faz o “Controle”, ou seja, além de leitura é ele quem pode realizar os lançamentos, mas não foi considerado, por simplificação a utilização de um AD, ou outro componente de autenticação de usuário. Portanto, foi considerada entrada em tela, porém, a API de lançamentos está preparada para receber também chamadas externas, atendendo ao requisito funcional proposto.

- Para a identificação e ilustração, tanto do “suposto” domínio funcional e capacidades de negócio (Arquitetura de Negócio), quando o desenho da Arquitetura de solução “Macro”, estão ilustrados na apresentação ArquiteturaComercio.ppt. Os demais desenhos, C4, além de ilustrados aqui, neste documento, estão disponíveis no arquivo “C4-Financeiro.drawio”.

- Poderia ser considerada a utilização de ferramentas especializadas para o BI (ex: Power BI) para a visualização do relatório. Porém, aqui consideramos uma solução sistêmica.

- O cálculo do custo não foi realizado, pois falta informações, além das “50 requisições por segundo no consolidado e picos de 5% de perda”, informações estas, que vão desde o tamanho das mensagens até a quantidade dos restantes das transações. Porém, este orçamento pode ser realizado no portal da AWS ou de outras clouds, em conjunto com o pessoal responsável pela infra, para definir os critérios de clusters, redundância, ambiente recovery e entender o direcionamento da empresa no que tange investimentos e aplicação de tecnologias, seja de Bancos de Dados seja de sistemas operacionais e ferramentas. Deve também ser considerado, para efeito de orçamento, o tipo e condições de contrato que a empresa possui com o servidor, pois os valores podem ser bem diferentes, de acordo com o tipo de contrato.

- Para este contexto do teste, a estrutura criada no github tem somente o intuito de compartilhar os insumos criados, portanto não foi criada com o intuito de demonstrar as divisões e compartilhamentos considerando governança e futuras derivações para desenvolvimento do sistema.

# Considerações de Arquitetura

Atendendo aos requisitos de negócio e requisitos não funcionais, seguem as considerações de:

## Software

- A aplicação será desenvolvida com fronte-end em React, BFF em Node.js, Microsserviços em Java, API’s Rest com exposição em Gateway Aws e mensageria trafegando JSON através do Kafka;

- Serão utilizadas consultas em BDCache e transações persistidas em BD relacional, além de utilização de ingestão de JSON em Datalake para a geração de relatórios e histórico;

- Os componentes serão conteinerizados utilizando Docker, sendo que os MS-Debito Caixa e MS-Credito Caixa, apesar de virtualmente estarem no mesmo EC2, estarão empacotados em 2 conteineres distintos, gerenciados através do ECS com Fargate;

- Será utilizada arquitetura orientada à eventos (EDA) com gestor de eventos para orquestrar o transacional dos lançamentos aumentando a escalabilidade, bem como a utilização de cache de Dados, microsserviços que facilitam o desacoplamento da aplicação, facilitando a migração de tecnologia e reutilização de componentes em caso de necessidades futuras aumentando a escalabilidade da aplicação;

- Todos os serviços e microsserviços se comunicam via https, não é permitido tráfego via http, mesmo em comunicação interna, aumentando a segurança da aplicação;

- A segurança de acesso aos componentes e à aplicação será tratada através de autenticações tokenizadas JWT para as API’s e mTLS para as mensagens Kafka, além da definição das políticas de acesso (IAM);

- É obrigatória, a utilização do cache de banco, no caso da consulta do MS-consolidado. No caso de evoluções e necessidade de o gestor solicitar por manipulação de visões, a solução do relatório estático deve ser revista, sendo, nesse caso substituída por ferramentas de BI (ex: Quicksight);

- Foi determinada a utilização de um Lake com ferramentas de transformação e exibição para o não transacional (cópia do registro de lançamentos), com eventuais transformações e exibições do consolidado, garantindo a centralização dos Dados para futuras ingestões de informações de outras áreas e domínios, resultando em um local “único” para obtenções de informações relevantes para o negócio para a Diretoria da empresa.

## Infra:

- Cloud AWS, com utilização de servidores EC2 para as aplicações, RDS para o BD transacional, WAF (AWS Web Firewall), Docker (com NGINX, React e node.js) com AWS ECR (contêiner register) e AWS ECS (contêiner service) para o frontend e BFF;

- AWS API Gateways (Serverless), para API’s;

- AWS Elasticache for REDIS, para cache do banco, podendo ser expandido para REDIS + Elastic Cache, em caso de necessidade da empresa;

- Apache Kafka para mensageria, instanciado em cluster;

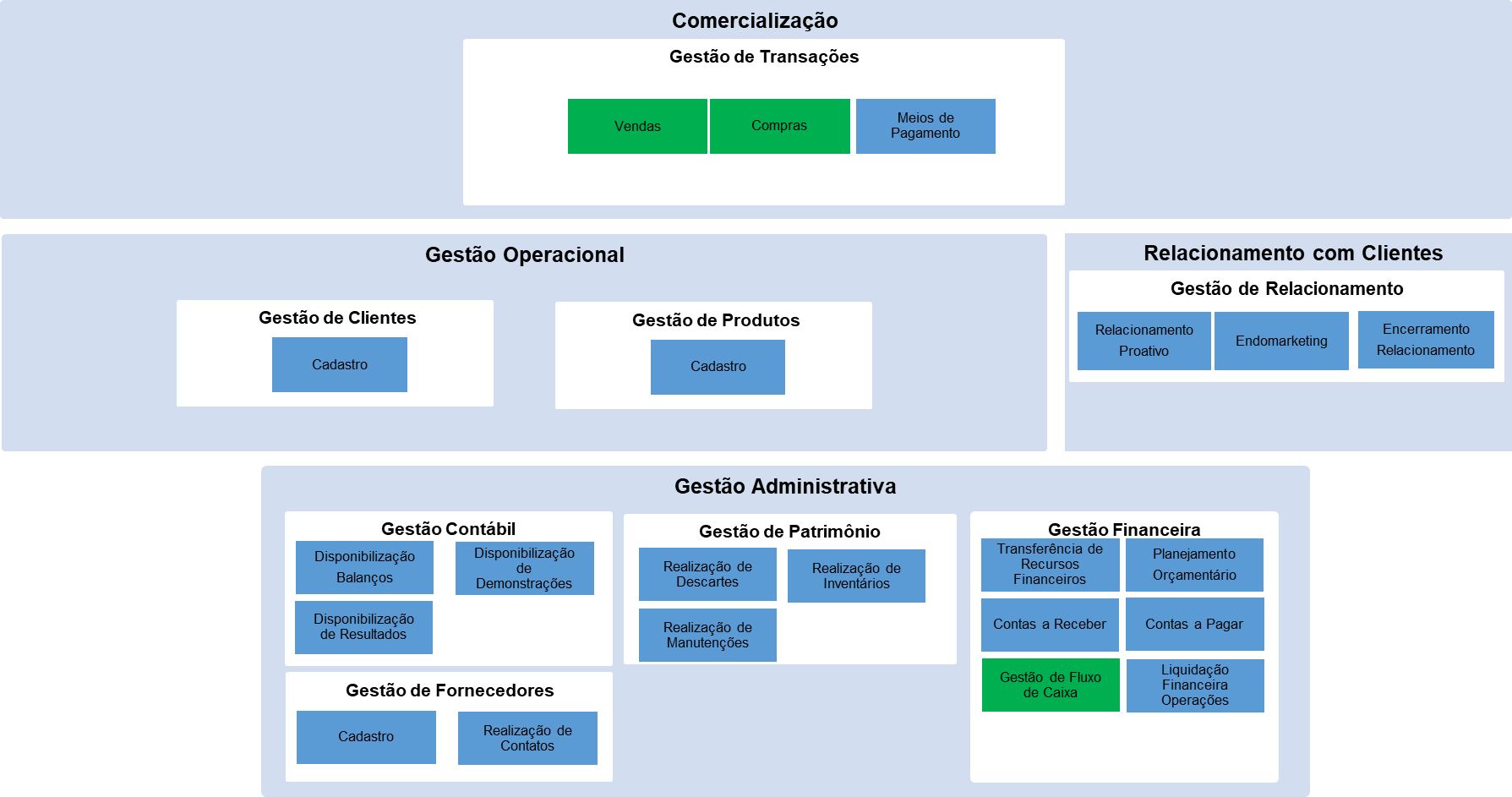
- Amazon S3 para o Datalake, com AWS Glue para ingestão e transformação;

- AWS CloudWatch, para monitoramento das instâncias e serviços e execução dos log traces;

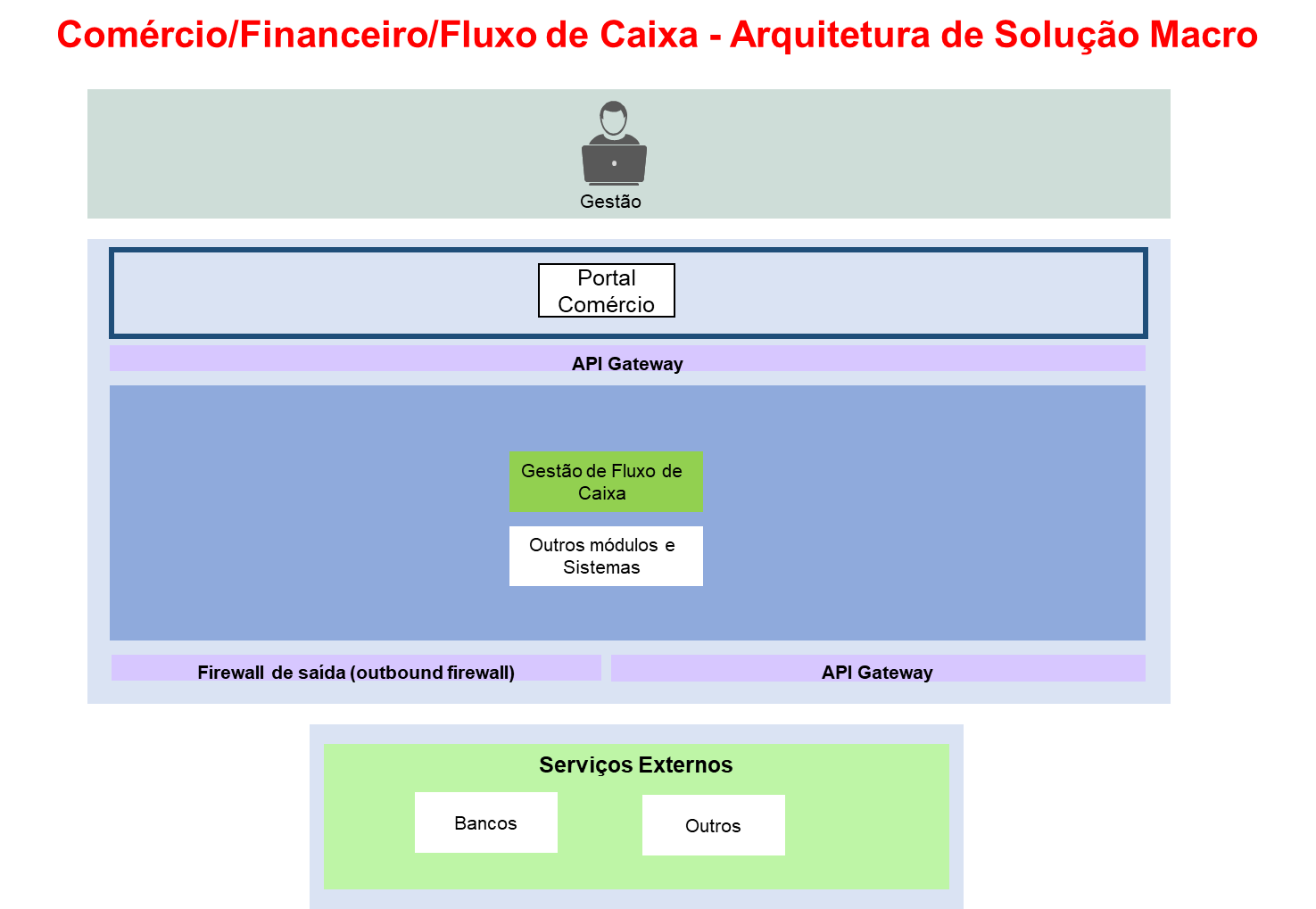
- Para a esteira CI/CD, configuração do GitHub (branches), GitLab, Jenkins (Build e testes) ECR (registro do Docker) e Fargate (implantação);

# Ilustrações

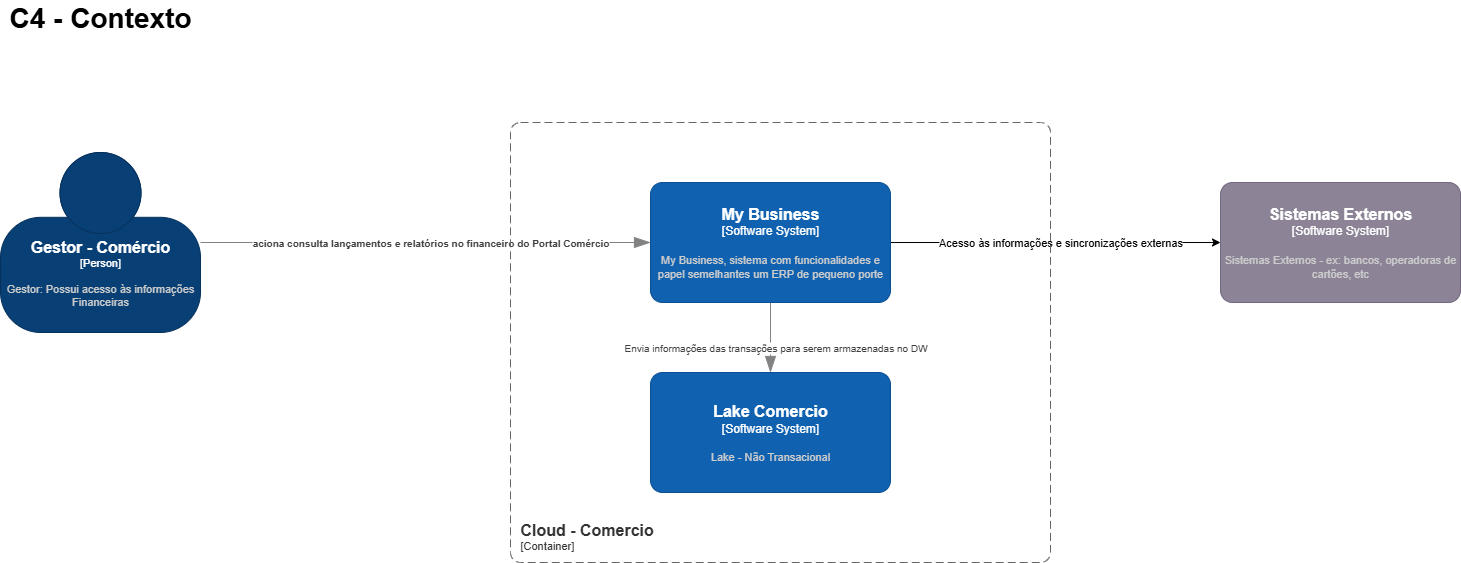
## Arquitetura de negócio



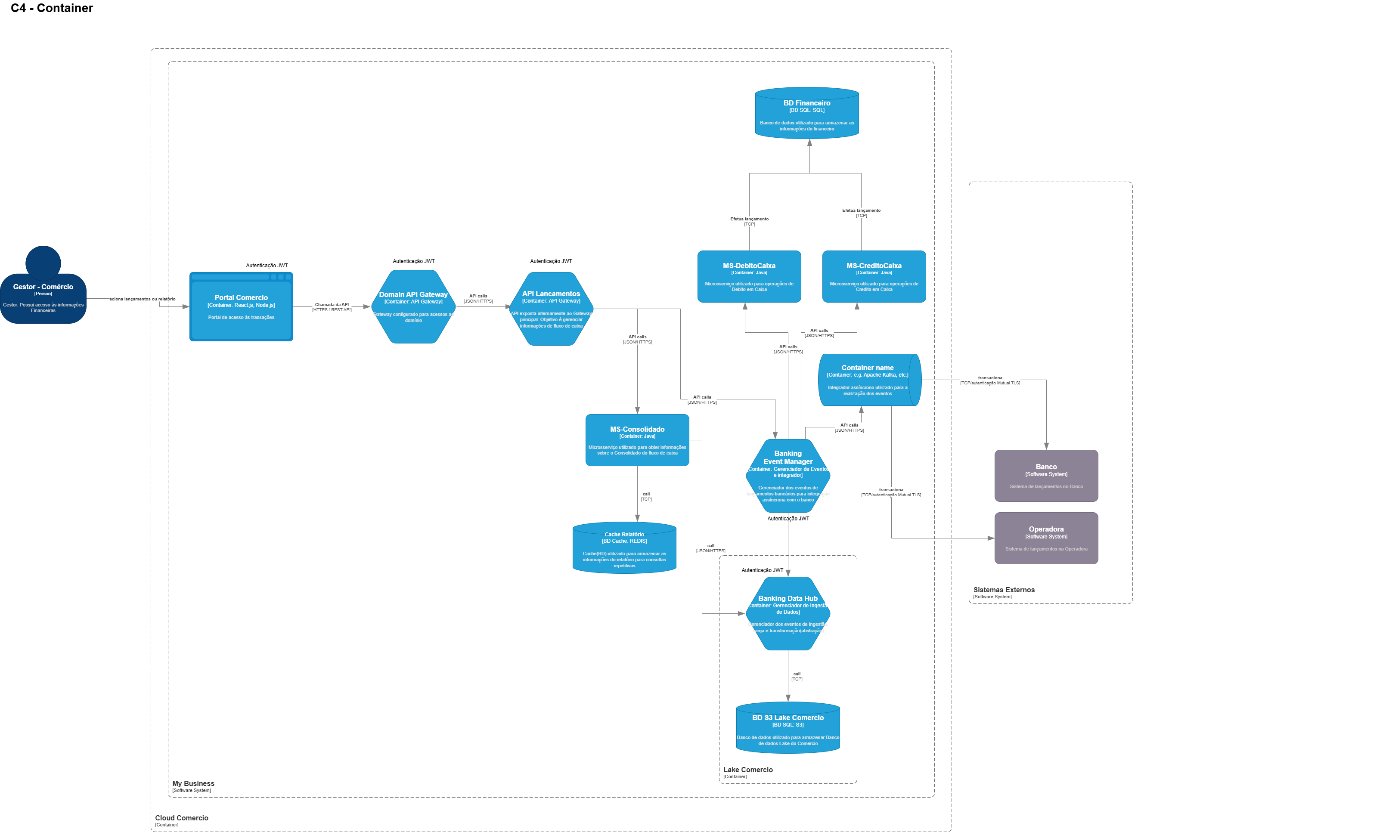
## Comércio/Financeiro/Fluxo de Caixa - Arquitetura de Solução Macro



## C4 – Contexto



## C4 – Contêiner



## C4 – Contêiner + Infra

