Pitch

PROJETO APLICADO

MODELO ARQUITETURAL PARA REFATORAÇÃO DA CAMADA DE BACK-END DO PROCESSO DE CHECKOUT EM UMA APLICAÇÃO GLOBAL

Apresentação

Quem sou eu?

- Luiz Victor Tinini
- 26 Anos
- Formado em Sistema de Informação pela PUC Campinas
- Arquiteto de Software
- Campinas – SP
- LinkedIn: https://www.linkedin.com/in/luizvictortinini/

Relevância do projeto para mim:

Este projeto me deu a oportunidade de redesenhar uma arquitetura legada e trazer conceitos mais modernos ao sistema em questão. Durante o projeto pude abordar conceitos de design patterns, SOLID, refactoring, entre outros.

Contexto

Breve história do sistema

- Iniciado com um CMS conceituado no mercado;

- Se provou viável rapidamente;

- Teve sua implantação em vários países em menos de 2 anos de projeto;

- A necessidade de remoção do CMS se tornou necessária;

- Novo mercado, novos conceitos, necessidades ainda não conhecidas...

- Menos de 4 anos de projeto, 14 países live e crescendo cada dia mais.

Problema

Qual é a minha dor?

- Expansão muito rápida causou a geração de muitos débitos técnicos;

- Manter o sistema funcionando hoje exige um gasto muito grande com hardware;

- O tempo de desenvolvimento é muito grande;

- O tempo de teste é muito grande;

- Microsserviços misturam contextos;

- Escalar os serviços é complexo;

- Problemas de integração são recorrentes por conta de instabilidades;

- Insatisfação do time;

Solução

O que eu proponho?

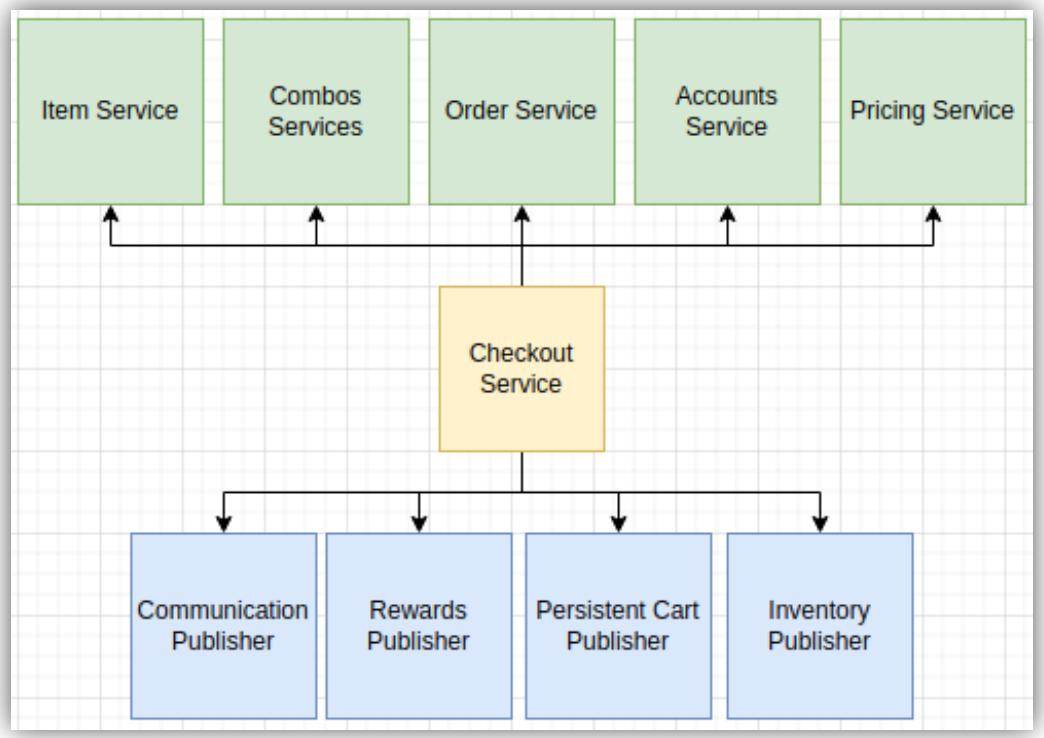
- Refatorar todo o back - end do processo checkout da aplicação;

- Segregar mais os microsserviços;

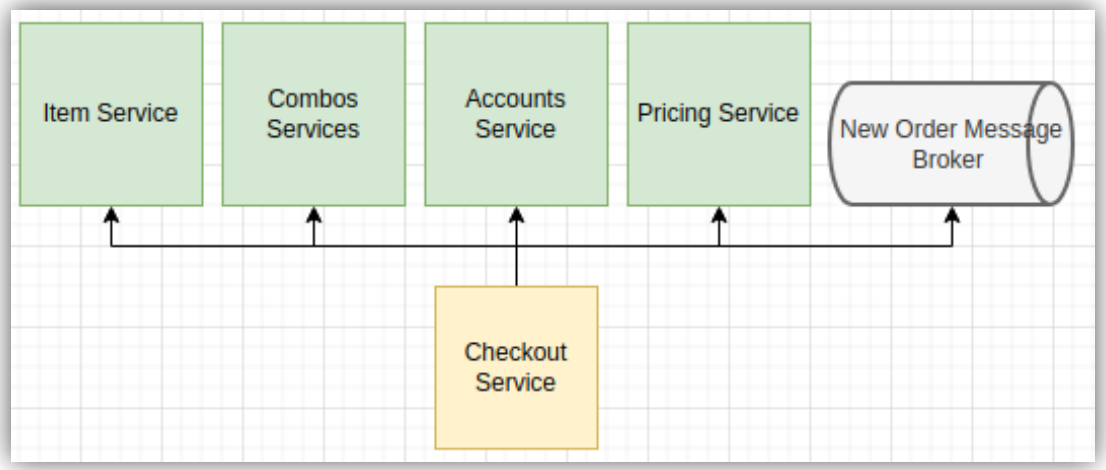
- Utilizar de mais de processamentos assíncronos;

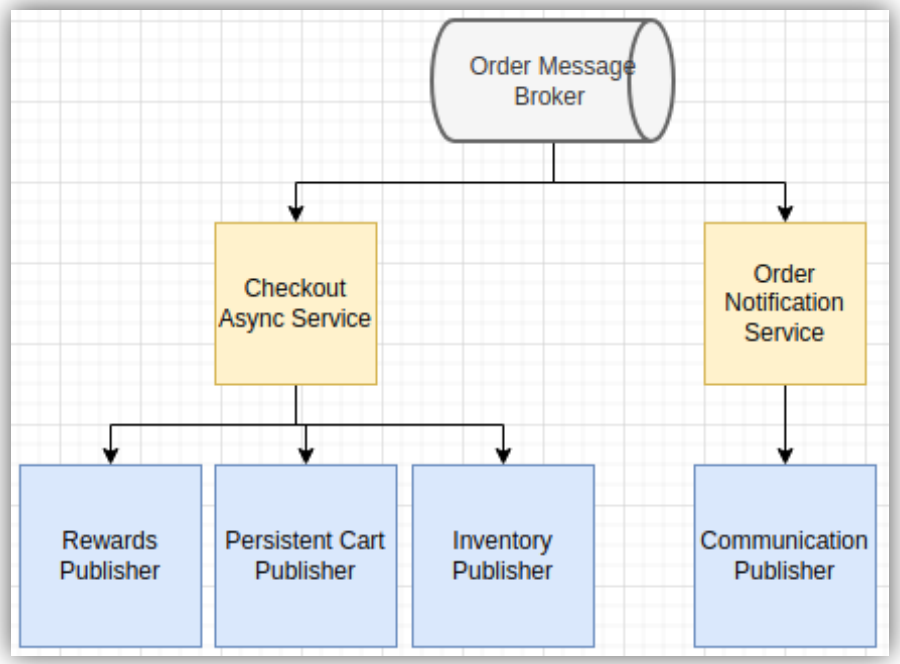
Exemplos nos próximos slides:

Legado



Arquitetura Proposta

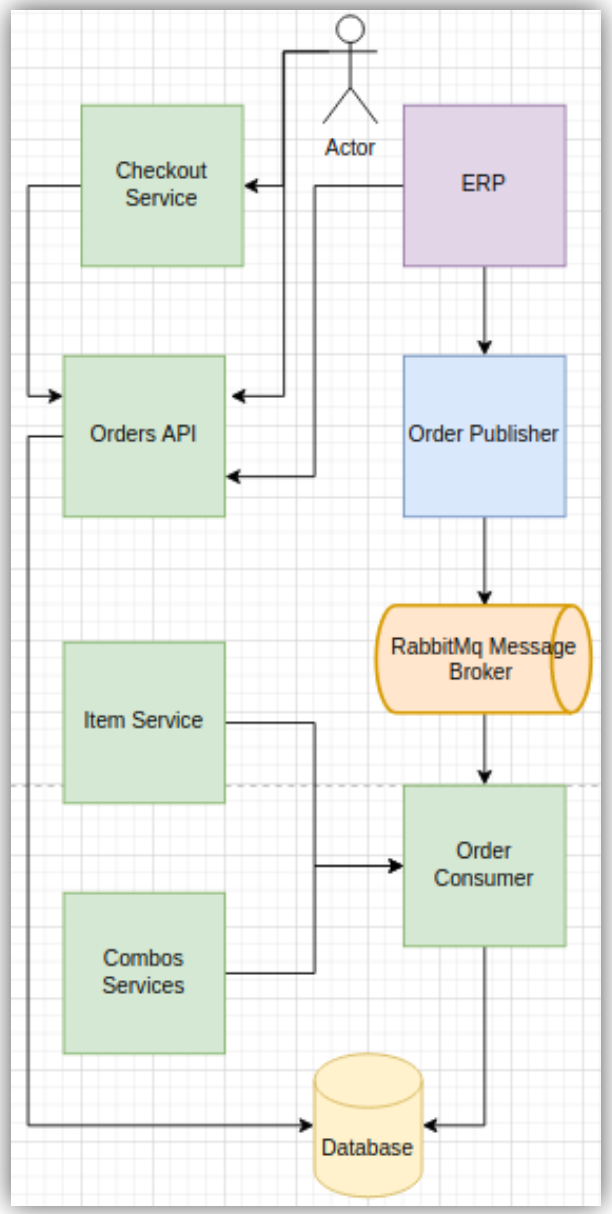




Benefícios

- Menos integrações no Serviço de Checkout;
- Chamadas que podem ser assíncronas foram repassadas para um serviço dedicado;
- Novas ordens ganharam um tópico Kafka dedicado;
- Ordens já processadas estarão disponíveis em um tópico dedicado;

Legado



Benefícios

- Processamento de ordens é feito de forma assíncrona;

- Enriquecimento de ordens é feito em um microsserviço dedicado;

- Order Consumer tem apenas a responsabilidade de salvar ordens;

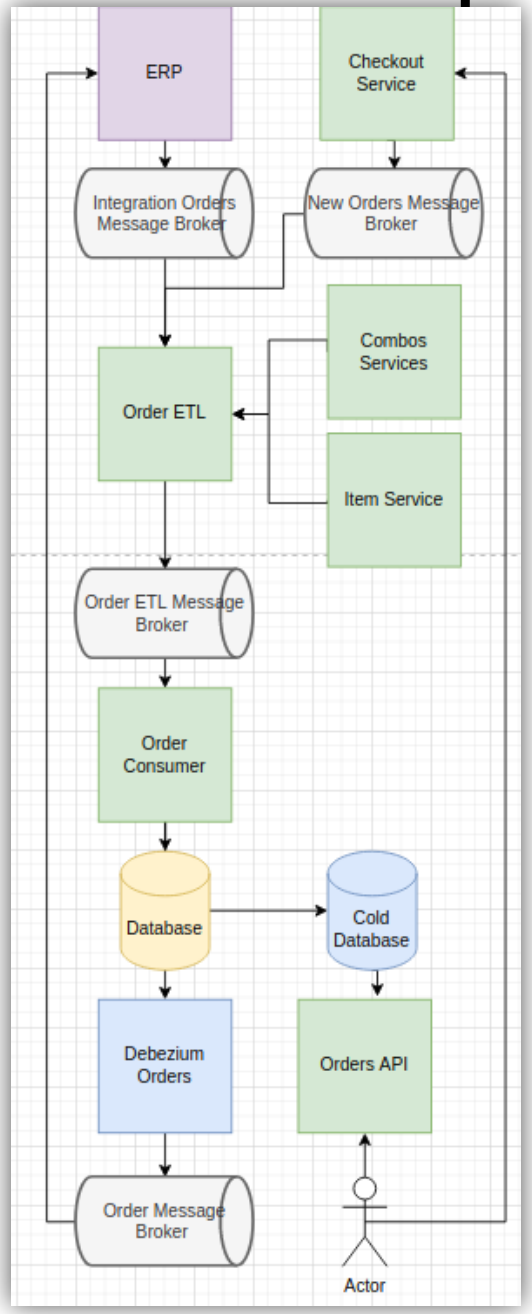
- Debezium Ordens vai publicar todas as atualizações do DB de ordens em um tópico;

- ERP vai postar e receber atualizações via Kafka

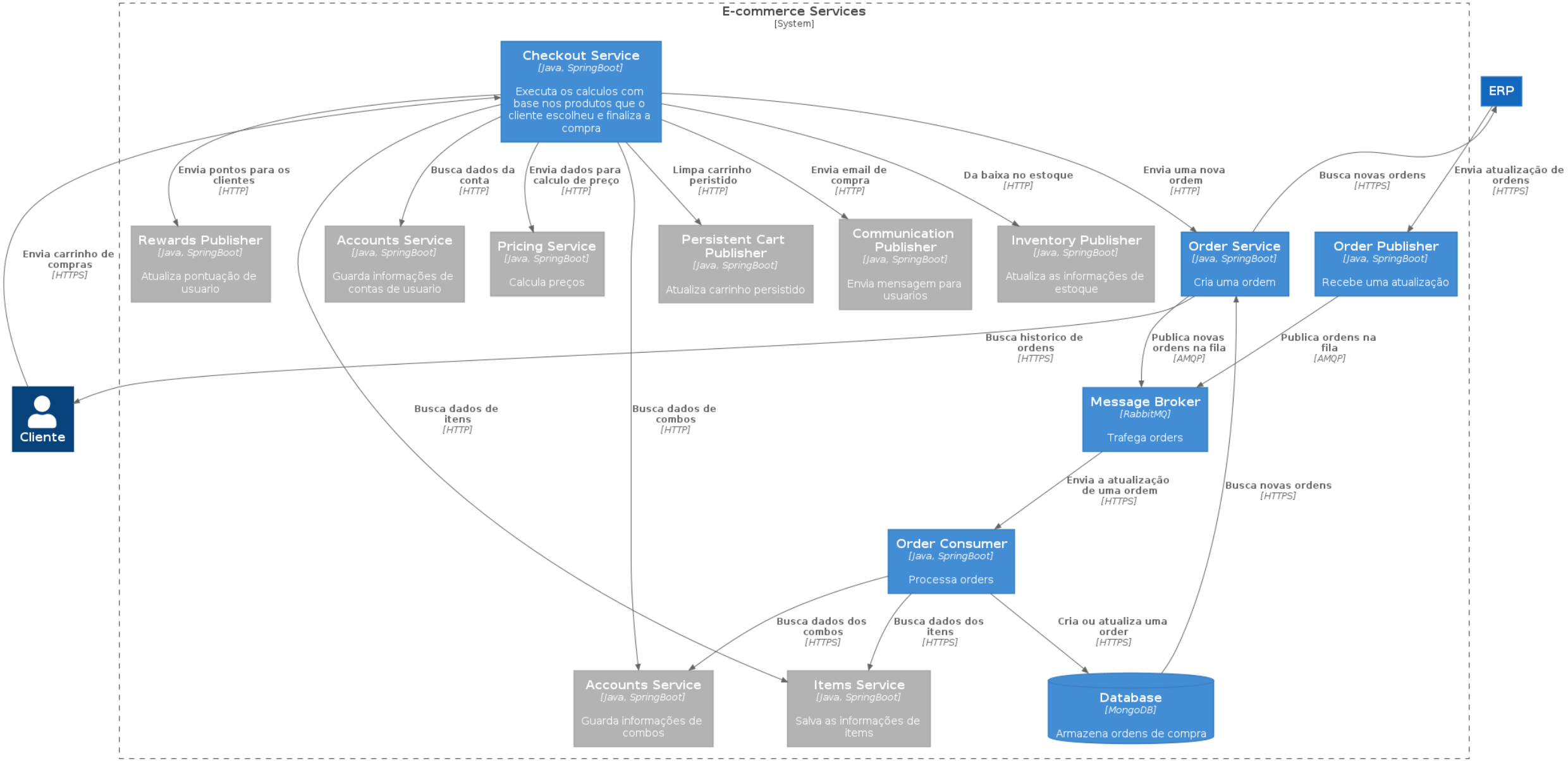
- Checkout publica novas ordens em um tópico Kafka;

- Ordens API está conectado em um nó analítico da base de dados;

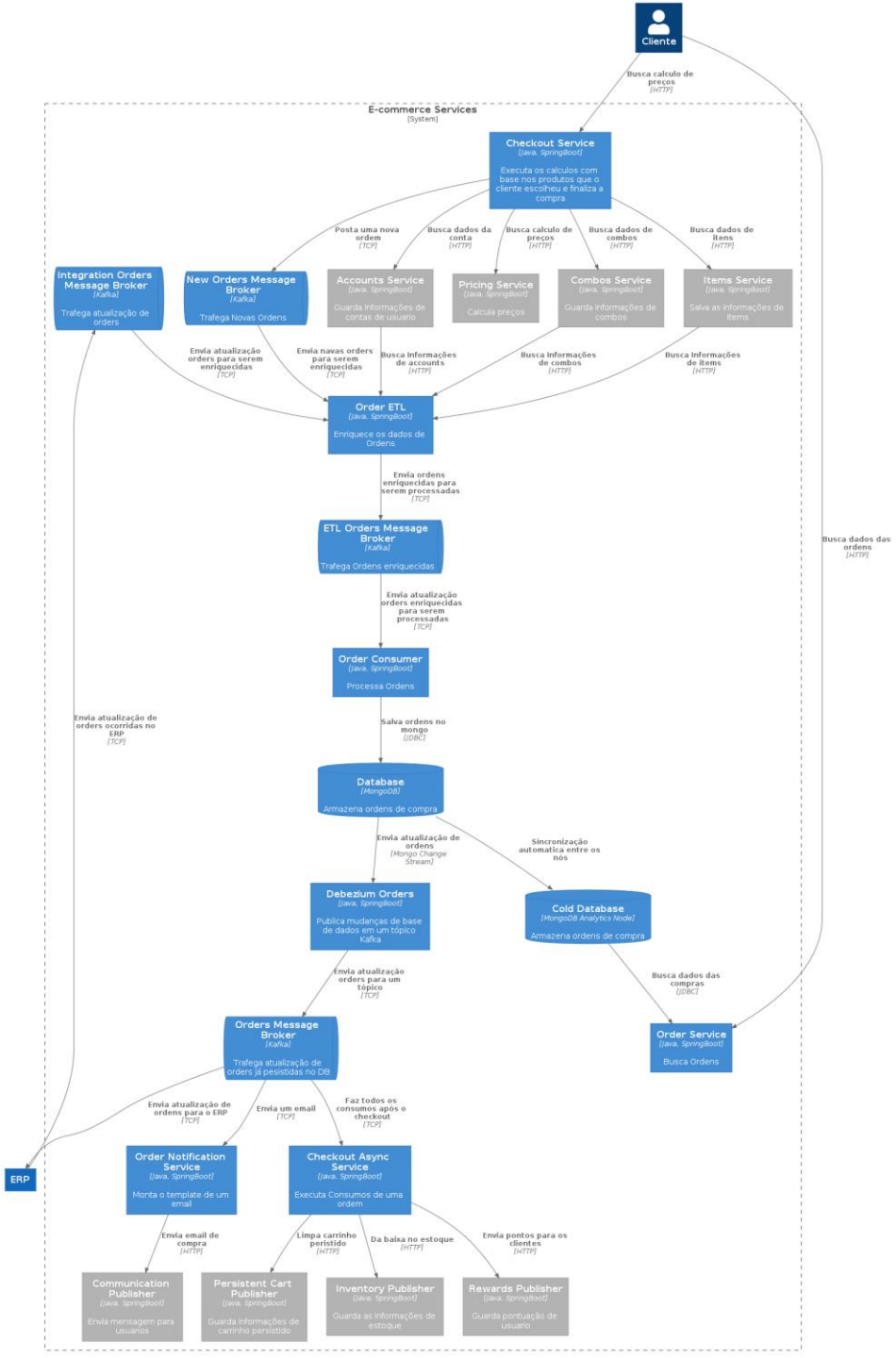
Arquitetura Proposta



Legado – C4 Model Level 2



Arquitetura Proposta – C4 Model Level 2



Diferencial

O que a sua solução tem de especial?

- Manter a solução legada enquanto a nova está sendo construída;

- Manter a linguagem de programação conhecida pelo time;

- Mudar os frameworks de mensageria para trazer performance e segurança;

- Criar serviços simples e de fácil manutenção;

- Deixar cada serviço mais focado em uma pequena parte do todo;

- Usar muito de processamentos assíncronos;

- Trazer inovação com frameworks já consolidados no mercado. Inovar mas sem colocar em risco a estabilidade da plataforma;

Impacto

Qual é o impacto do seu produto?

- Aumento da confiabilidade;

- Facilidade com novas integrações;

- Aumento da satisfação do time;

- Redução do tempo de desenvolvimento;

- Redução do tempo de teste;

- Redução de custos;

- Bom exemplo para ser seguido em outras partes do sistema;

Próximos passos

Qual é o impacto do seu produto?

- Criar o nível 4 do C4 model para alguns containers;
- Criar todos os épicos e user stories do Scrum;
- Elaborar uma estrategia de priorização;
- Começar o processo de refactoring;
- Mostrar os ganhos para outras equipes da empresa;

Obrigado!