



igti

RELATÓRIO

PROJETO APLICADO

Instituto de Gestão e Tecnologia da Informação
Relatório do Projeto Aplicado

Modelo arquitetural para refatoração da camada de back-end do processo de checkout em uma aplicação global

Luiz Victor Stefani Tinini

Orientador: Professor Ricardo Brito Alves

Março de 2022





LUIZ VICTOR STEFANI TININI

INSTITUTO DE GESTÃO E TECNOLOGIA DA INFORMAÇÃO

RELATÓRIO DO PROJETO APLICADO

Modelo arquitetural para refatoração da camada de back-end do processo de checkout em uma aplicação global

Relatório de Projeto Aplicado
desenvolvido para fins de conclusão do
curso MBA em Arquitetura de software e
soluções.

Orientador: Professor Ricardo Brito
Alves

CAMPINAS

Março de 2022

Sumário

1. CANVAS do Projeto Aplicado	5
1.1 Desafio	6
1.1.1 Análise de Contexto	6
1.1.2 Personas	7
1.1.3 Benefícios e Justificativas	8
1.1.4 Hipóteses	9
1.2 Solução	10
1.2.1 Objetivo SMART	10
1.2.2 Premissas e Restrições	11
1.2.3 Backlog de Produto	12
2. Área de Experimentação	13
2.1 Sprint 1	15
2.1.1 Solução	15
• Evidência do planejamento:	15
• Evidência da execução de cada requisito:	15
• 20	
2.1.2 Lições aprendidas	15
2.2 Sprint 2	16
2.2.1 Solução	16
• Evidência do planejamento:	16
• Evidência da execução de cada requisito:	16
• Evidência dos resultados:	16
2.2.2 Lições aprendidas	16
2.3 Sprint 3	17
2.3.1 Solução	17
• Evidência do planejamento:	17
• Evidência da execução de cada requisito:	17
• Evidência dos resultados:	17
2.3.2 Lições aprendidas	17
3. Considerações Finais	18

3.1 Resultados Finais	18
3.2 Contribuições	18
3.3 Próximos passos	18

1. CANVAS do Projeto Aplicado



1.1 Desafio

1.1.1 Análise de Contexto

Uma empresa possui um produto global que escalou de um para mais de catorze países em um tempo muito curto. Apesar de ainda estar funcionando, o processo de checkout é um gargalo enorme no processo de uma venda. Ocorrem lentidões e falhas com uma certa frequência. A forma de mitigar esse cenário até agora foi investindo em hardware, porem hoje uma refatoração do back-end se faz necessária.

Esse projeto irá focar no modelo arquitetural base para que essa refatoração ocorra.

Matriz CSD

	Certezas	Suposições	Duvidas
Atores	<ul style="list-style-type: none"> O time de produto sabe que a arquitetura atual é ultrapassada O time de engenharia tem muita vontade de focar na refatoração 		
Cenários	<ul style="list-style-type: none"> Novos clientes têm que ser integrados sem colocar a saúde da plataforma em risco 		
Regras	<ul style="list-style-type: none"> É preciso criar um plano de refatoração É preciso criar um 	<ul style="list-style-type: none"> A refatoração não pode causar impacto negativos 	

	desenho da nova arquitetura	nos clientes atuais	
--	-----------------------------	---------------------	--

POEMS

Pessoas	Objetos	Ambiente	Mensagem	Serviços
Quem está envolvido no contexto em análise?	Que objetos fazem parte do ambiente?	Quais são as características do ambiente?	Que mensagens são comunicadas?	Quais serviços são oferecidos?
Desenvolvedores	Computador , teclado e mouse	Home office ou escritório	Atuando no desenvolvimento das demandas da empresa	Desenvolvimento das funcionalidades e correção de bugs
Pessoas de produto	Computador , teclado e mouse	Home office ou escritório	Atuando no desenho das novas funcionalidades do sistema	Desenho de funcionalidades e protótipos. Validar as funcionalidades desenvolvidas.
Pessoas de projeto	Computador , teclado e mouse	Home office ou escritório	Atuando na organização dos processos da empresa	Organização de pessoas e processos.
Alta gerencia	Computador , teclado e mouse	Home office ou escritório	Negociando novos contratos com novos clientes	Proporcionando novos negócios para a empresa.

Registros	Insights
As informações iniciais foram obtidas através de entrevista com desenvolvedores, Product owner e diretores	<ul style="list-style-type: none"> - Envolver bastante os desenvolvedores e arquitetos no desenho da solução - Envolver o produto, projeto engenharia e gerencia para que todos saibam a necessidade do projeto.

1.1.2 Personas



João Silva

- Arquiteto de Sistemas
- 38 anos
- 5 anos de empresa

O que ele pensa e sente?

- Fica feliz por conta da expansão constante da empresa.
- Frustração por conta da quantidade de débitos técnicos.
- Tem medo que o sistema de checkout pare inesperadamente por conta de sobrecarga.

O que ele escuta?

- O sistema está lento para finalizar novas ordens de compra
- Nosso processo de checkout não é escalável e não vai suportar a expansão

O que fala e faz?

- Orienta os arquitetos de software para os rumos que a empresa está indo
- Negocia com o time de produto tempo para melhorias técnicas

O que ele vê?

- Grandes possibilidades de melhora no sistema

Quais são suas necessidades?

- Precisa de apoio do time de negócio para poder corrigir problemas da arquitetura legada.



José Carlos

- Desenvolvedor
- 26 anos
- 2 anos de empresa

O que ele pensa e sente?

- Deseja virar arquiteto de software em alguns anos
- Está sobrecarregado por conta de ter que ajudar em war rooms decorrentes do legado

O que ele escuta?

- Temos mais 11 clientes para implantar o sistema esse ano

O que fala e faz?

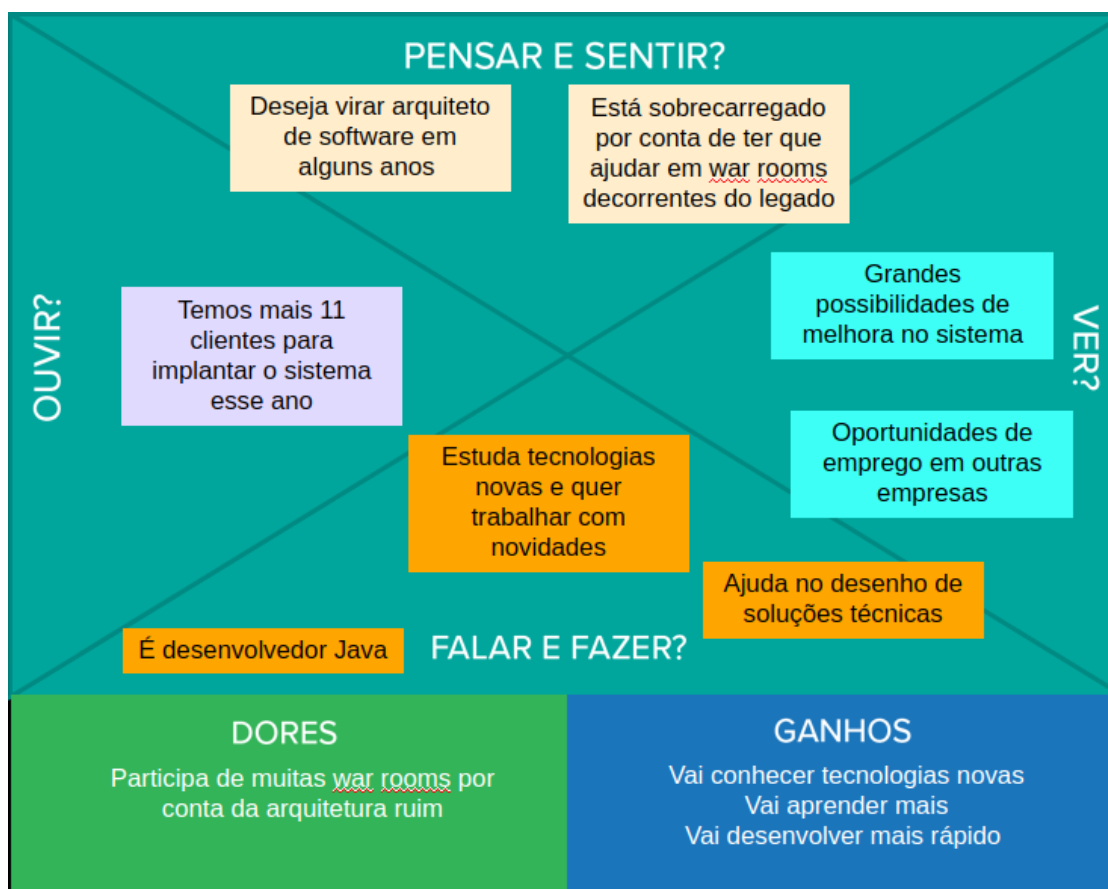
- Ajuda no desenho de soluções técnicas
- É desenvolvedor Java
- Estuda tecnologias novas e quer trabalhar com novidades

O que ele vê?

- Grandes possibilidades de melhora no sistema
- Oportunidades de emprego em outras empresas

Quais são suas necessidades?

- Precisa trabalhar com novas tecnologias para poder acelerar o processo de sua mudança de carreira.



Ana Maria

- Product Manager
- 29 anos
- 3 anos de empresa

O que ela pensa e sente?

- Vê grandes possibilidades de expansão do negocio

O que ela escuta?

- Que a empresa tem que integrar mais 11 clientes na plataforma

O que fala e faz?

- Entende as necessidades dos clientes
- Ajuda os PO's a fazerem os epicos
- Negocia com o time de engenharia prioridades

O que ela vê?

- Grandes possibilidades para a plataforma no futuro

Quais são suas necessidades?

- Precisa que os débitos técnicos sempre sejam desenhados no backlog para que possamos encaixar com o tempo nas sprints

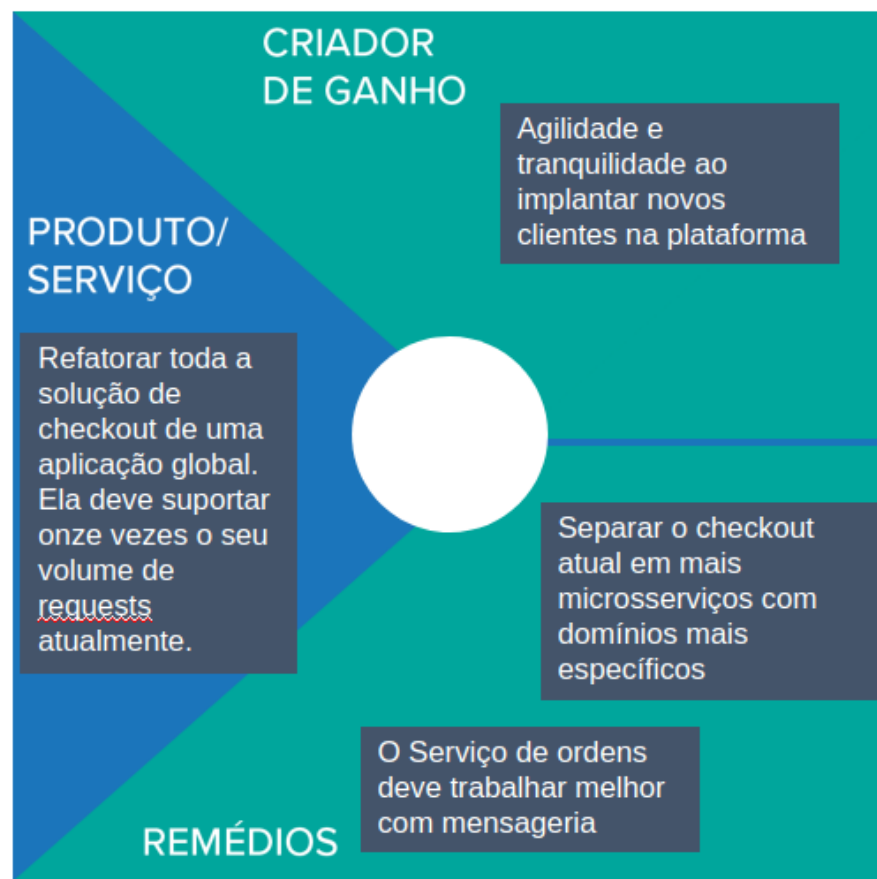


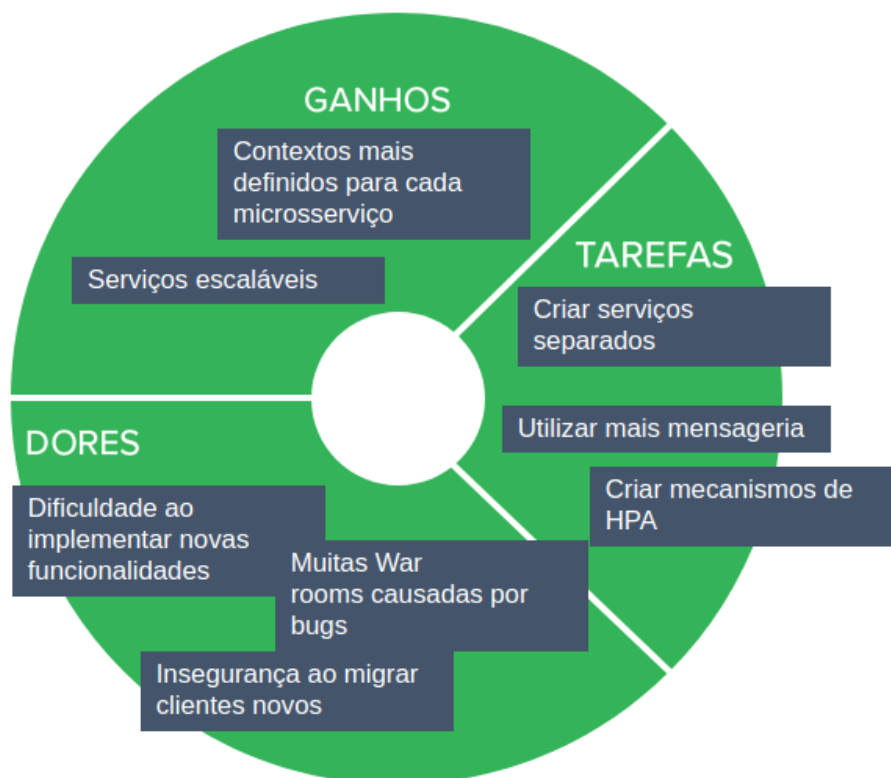
1.1.3 Benefícios e Justificativas

Itens	Checkout Service	Order Service	Order Notification	Post Order Service
Objetivos	Verificar dados da compra e finalizar pedido	Armazenar os dados da ordem	Notificar cliente sobre sua compra	Consumir estoque e dar pontos.
Atividades	Remover chamadas assíncronas Remover e-mail Criar uma integração via fila	Salvar uma ordem	Mandar email de atualização de uma ordem	Consumir estoques e bonificar os clientes

Questões	Quais os domínios que esse serviço deve ter?	Quais os domínios que esse serviço deve ter?	Quais os domínios que esse serviço deve ter?	Quais os domínios que esse serviço deve ter?
Barreiras	Manter o legado funcionando	Manter o legado funcionando	Manter o legado funcionando	Manter o legado funcionando
Ações do cliente	Fechar um pedido	N/A - Ação automática do sistema	Receber um e-mail ou notificação	N/A - Ação automática do sistema
Funcionalidades	Finalizar Compras	Salvar as ordens	Notificar o usuário	Executar chamadas assíncronas do checkout
Interação	Front end manda uma request para o checkout	Lê uma fila e processa a informação	Lê uma fila e manda e-mail de acordo com o status	Lê uma fila e executa chamadas de acordo com o status
Mensagem	Compra efetuada com sucesso	200 - OK	N/A	N/A
Onde ocorre	Após a seleção de produtos no carrinho de compra	Após os cálculos do checkout	Após a criação ou atualização de uma ordem de compra	Após a criação de uma ordem de compra
Tarefas aparentes	Remover grande parte do código e repassar para os outros microserviços	Remover grande parte do código e repassar para os outros microserviços	Criar microserviço	Criar microserviço

Tarefas escondidas	N/A	N/A	Criar os templates de e-mail	N/A
Processos de suporte	Dashboards, logs e telemetria dos serviços	Dashboards, logs e telemetria dos serviços	Dashboards, logs e telemetria dos serviços	Dashboards, logs e telemetria dos serviços
Saída desejável	Compra finalizada	Ordem integrada	Mandar um e-mail	Executar chamadas assíncronas





1.1.4 Hipóteses

Matriz de observações para hipóteses

Observação	Hipótese
Devemos remover componentes assíncronos do checkout service	Podemos criar serviços menores para cada processo assíncrono.
Devemos usar mais o padrão de mensageria nas integrações entre os microserviços	Integrações entre o serviço de ordens e checkout deveria ser feito por mensageria, para garantir que instabilidades no microserviço
Devemos manter a versão legada funcionando enquanto a nova está em desenvolvimento	Podemos criar um fork dos microserviços e trabalhar no fork durante a refatoração
Devemos usar tecnologias que garantam 99.99% de uptime	Podemos usar dois clusters em regiões diferentes para garantir que ele sempre esteja online. Os serviços de mensageria podem ser contratados por vendedores externos para não ter a necessidade de cuidar da infra de um cluster.

Ideias	B	A	S	I	C	O	Somatório	Priorização
--------	---	---	---	---	---	---	-----------	-------------

Integração entre serviço de checkout e de ordens feita usando Kafka	5	4	5	5	3	3	25	2
Integração entre os ERP's e o consumer de ordens feitas usando Kafka	5	4	4	4	2	2	21	5
Centralizar todos os emails de ordens em um único microserviço	4	3	5	4	3	3	22	3
Criar um serviço com todas as baixas necessárias nos outro microserviços	4	3	4	3	4	3	21	
Criar um tópico Kafka com todas as ordens criadas e atualizadas	5	5	5	3	5	2	25	1
Usar o mongoDB online archive no cluster de orders	3	3	3	5	3	5	22	4
Implementar HPA nos serviços de ordens	3	3	4	5	2	3	20	6

Escala	B - Benefícios	A - Abrangência	S - Satisfação	I - Investimentos	C - Cliente	O - Operacionalidade
5	De vital importância	Total (de 70% a 100%)	Muito grande	Pouquíssimo investimento	Nenhum impacto	Muito fácil
4	Significativo	Muito grande (de 40% a 70%)	Grande	Algum investimento	Pequeno impacto	Fácil
3	Razoável	Razoável (de 20% a 40%)	Média	Médio investimento	Médio impacto	Média facilidade
2	Poucos benefícios	Pequena (de 5% a 20%)	Pequena	Alto investimento	Grande impacto	Difícil
1	Algum benefício	Muito pequena	Quase não é notada	Altíssimo investimento	Impacto muito grande no cliente	Muito difícil

1.2 Solução

1.2.1 Objetivo SMART

Criar um projeto arquitetural utilizando o C4 model visando a refatoração do back-end do processo de checkout de um sistema global de e-commerce que hoje não é escalável devido a quantidade de débitos técnicos acumulados. Esse projeto deve ser executado em dois meses e tem como objetivo final o redesenho do sistema para que ele se adeque melhor nos conceitos de microserviços, fique mais modularizado, escalável, utilizando conceitos como: arquitetura SAGA e CQRS.

1.2.2 Premissas e Restrições

Premissas:

- O time de produto irá priorizar as demandas dessa refatoração
- Processo de refatoração vai acontecer de forma faseada.
- Todos os microsserviços serão analisados e modificados se caso necessário

Restrições

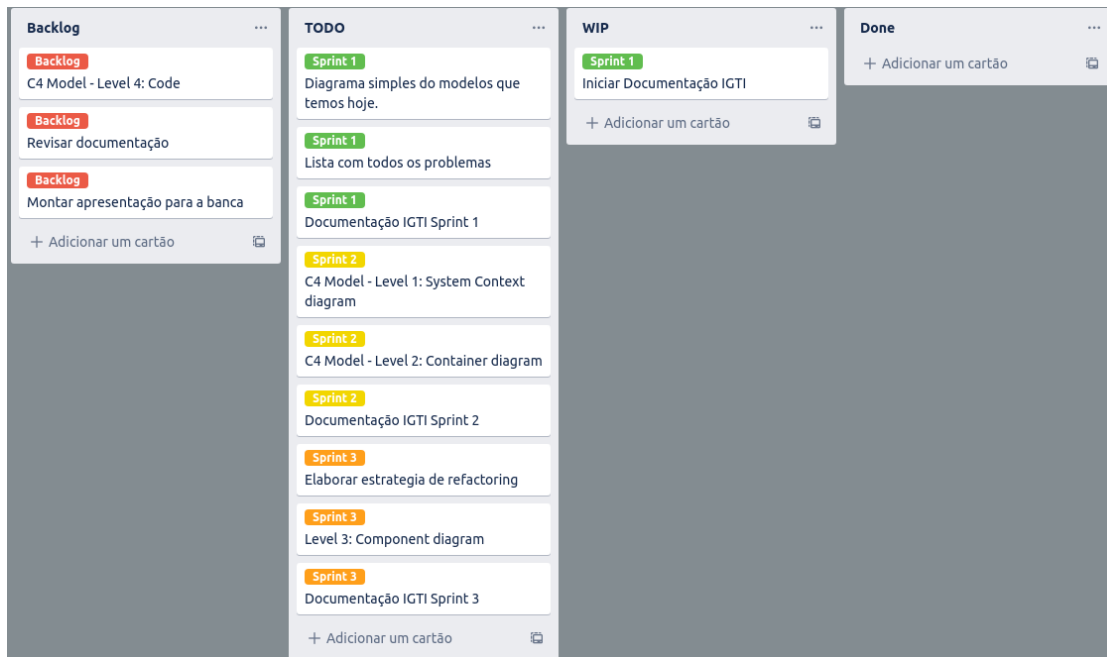
- Esse projeto apenas contemplará a quarta parte do C4 Model se todas os processos foram feitos previamente.
- Os contratos com os Front end e ERP's externos devem ser mantidos
- Os sistemas legados deverão permanecer rodando enquanto a refatoração acontece
- Os sistemas deverão respeitar minimamente as tecnologias da empresa

Riscos do projeto:

- **R001** - Ser necessário a mudança de contrato com o front end.
- **R002** - Ser necessário a mudança de contrato com os ERP's
- **R003** - Novas demandas chegarem e despriorizarem o projeto
- **R004** - Manter o legado e o novo ao mesmo tempo até ser possível a migração.

Probabilidade	Alta	Media	Alta R001 R002	Alta R003
	Media	Baixa	Media R004	Alta
	Baixa	Baixa	Baixa	Media
		Insignificante	Moderado	Catastrófico
	Impacto			

1.2.3 Backlog de Produto

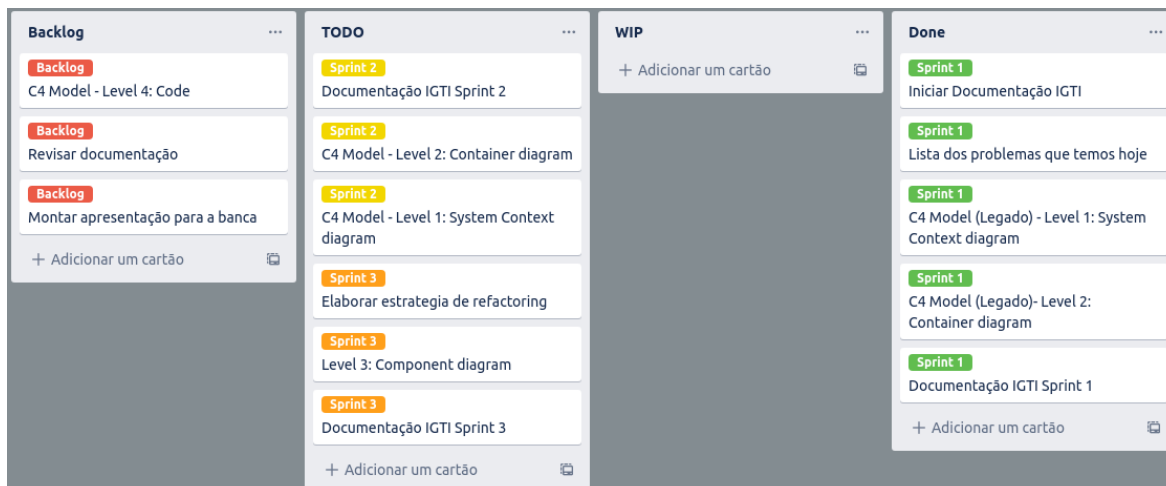


2. Área de Experimentação

2.1 Sprint 1

2.1.1 Solução

- Evidência do planejamento:

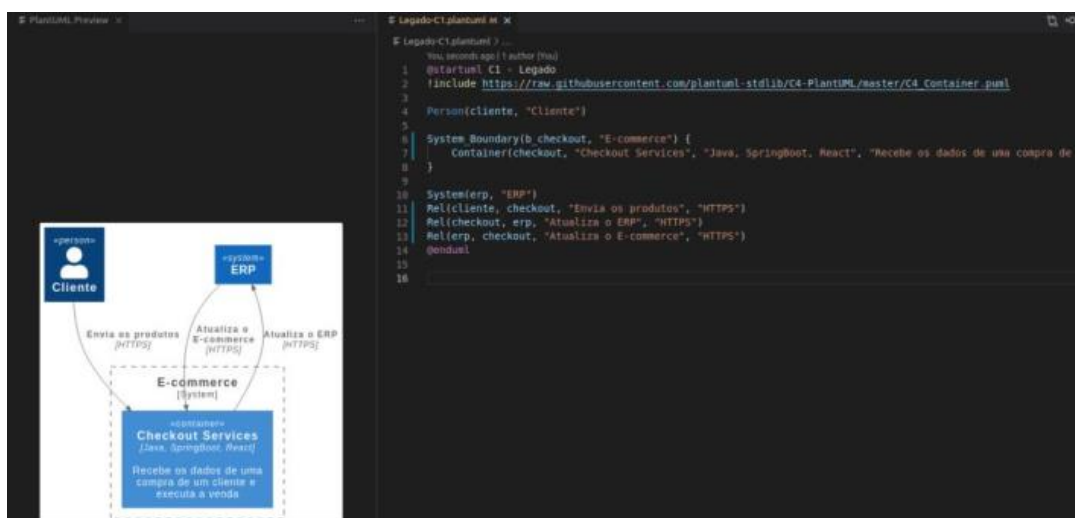


- Evidência da execução de cada requisito:

Evidencia da codificação do C4 Model (Legado) - Level 1: System Context diagram

Desenvolvido em visual studio code com o plugin do plantUML

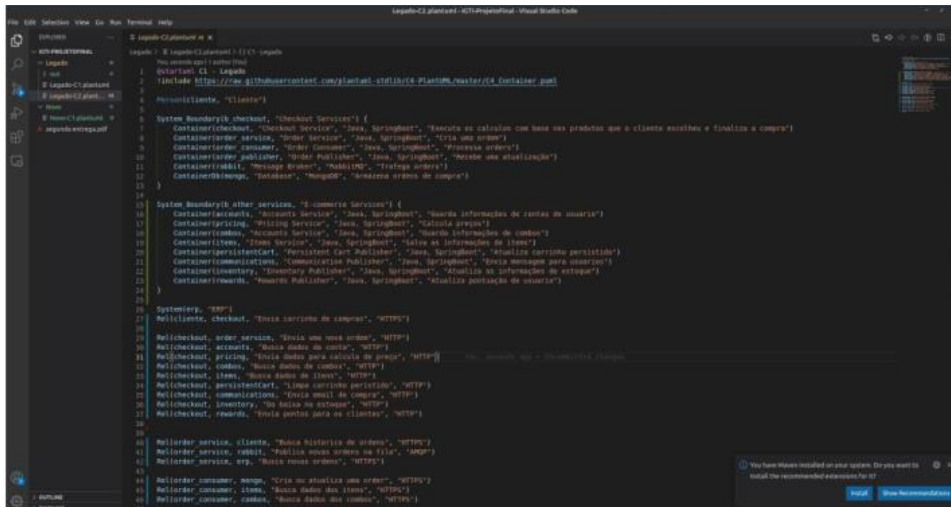
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Legado/Legado-C1.plantuml>



Evidencia da codificação do C4 Model (Legado) - C4 Model (Legado)- Level 2: Container diagram

Desenvolvido em visual studio code com o plugin do plantUML

Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Legado/Legado-C2.plantuml>



```

graph TD
    subgraph "Legado-C2.plantuml"
        direction TB
        checkout[checkout]
        order[order]
        payment[payment]
        shipping[shipping]
        checkout --> order
        checkout --> payment
        checkout --> shipping
    end

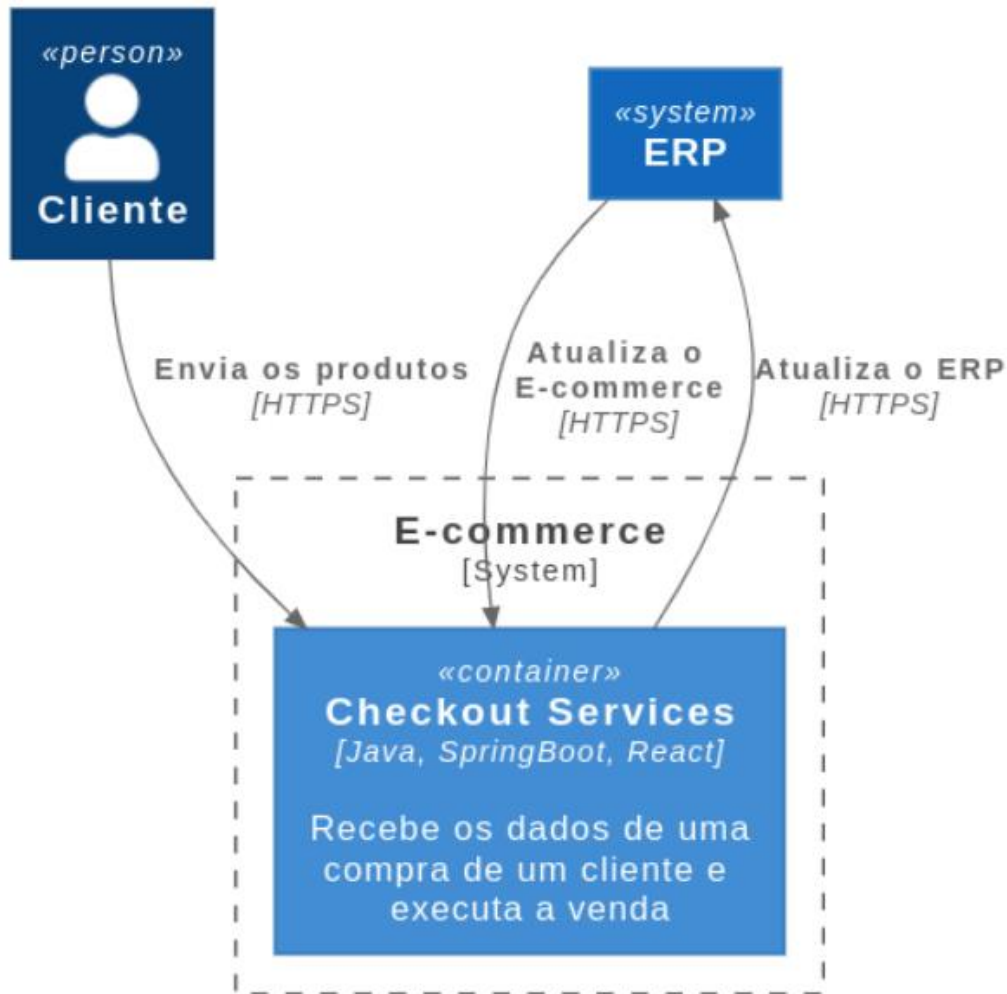
```

- Evidência dos resultados:

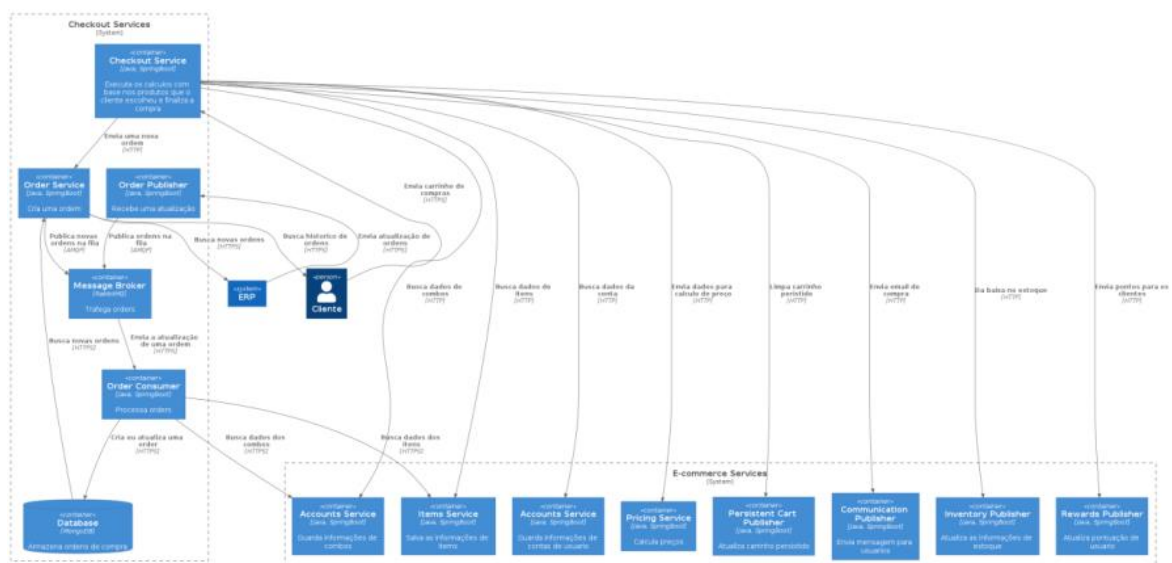
Obs: Para facilitar a visualização, segue abaixo os diagramas completos e o diagrama C2 completo e separados em duas partes com zoom.

Visto que nessa parte do projeto iremos apenas demonstrar os problemas atuais, o diagrama C3 não se tornou um requisito.

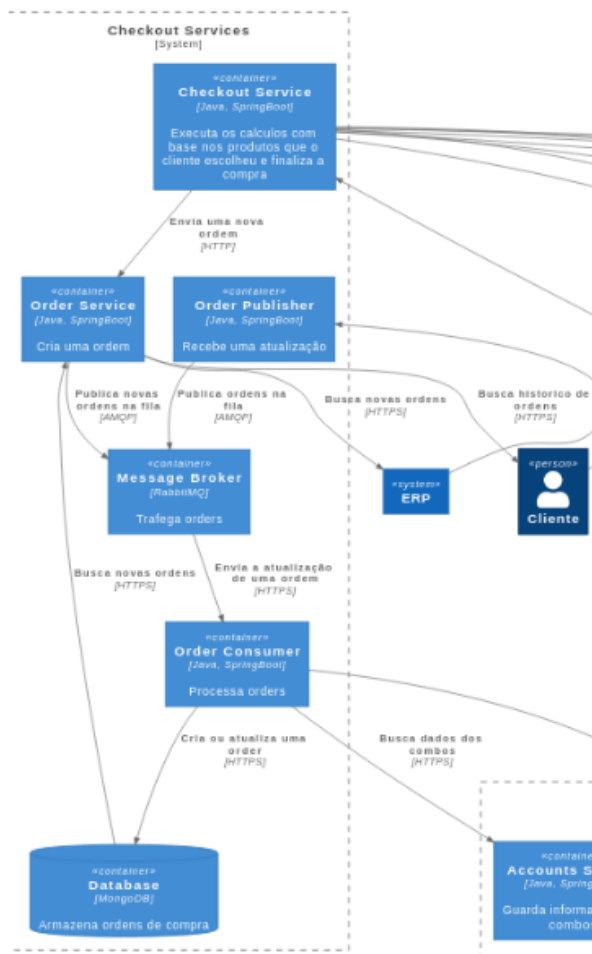
Arquitetura atual - C4 Model (Legado) - Level 1: System Context diagram



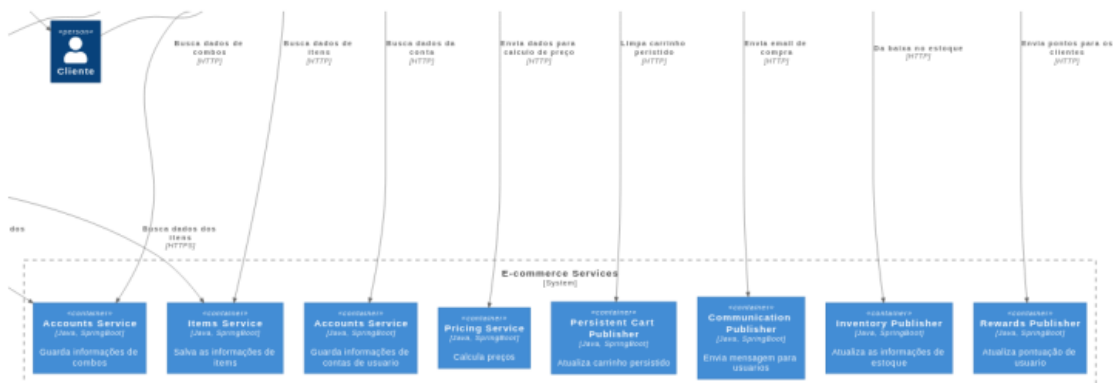
C4 Model (Legado) - C4 Model (Legado)- Level 2: Container diagram



Zoom Checkout Services:



Zoom E-commerce Services



Lista de problemas na arquitetura atual:

- **Checkout Service**
 - Mover chamadas assíncronas para outro microserviço
 - Enviar novas ordens diretamente para o order consumer
- **Order Service**
 - Remover criação de ordens
 - Apontar para um nó de banco de dados analítico (read only)

- **Order Consumer**
 - Remover consumo do RabbitMQ
 - Criar Consumer de Kafka
- **Order Publisher**
 - Descontinuar microserviço
- **ERP**
 - Remover integração via rest, fazer atualizações via kafka
 - Pegar novas ordens via Kafka

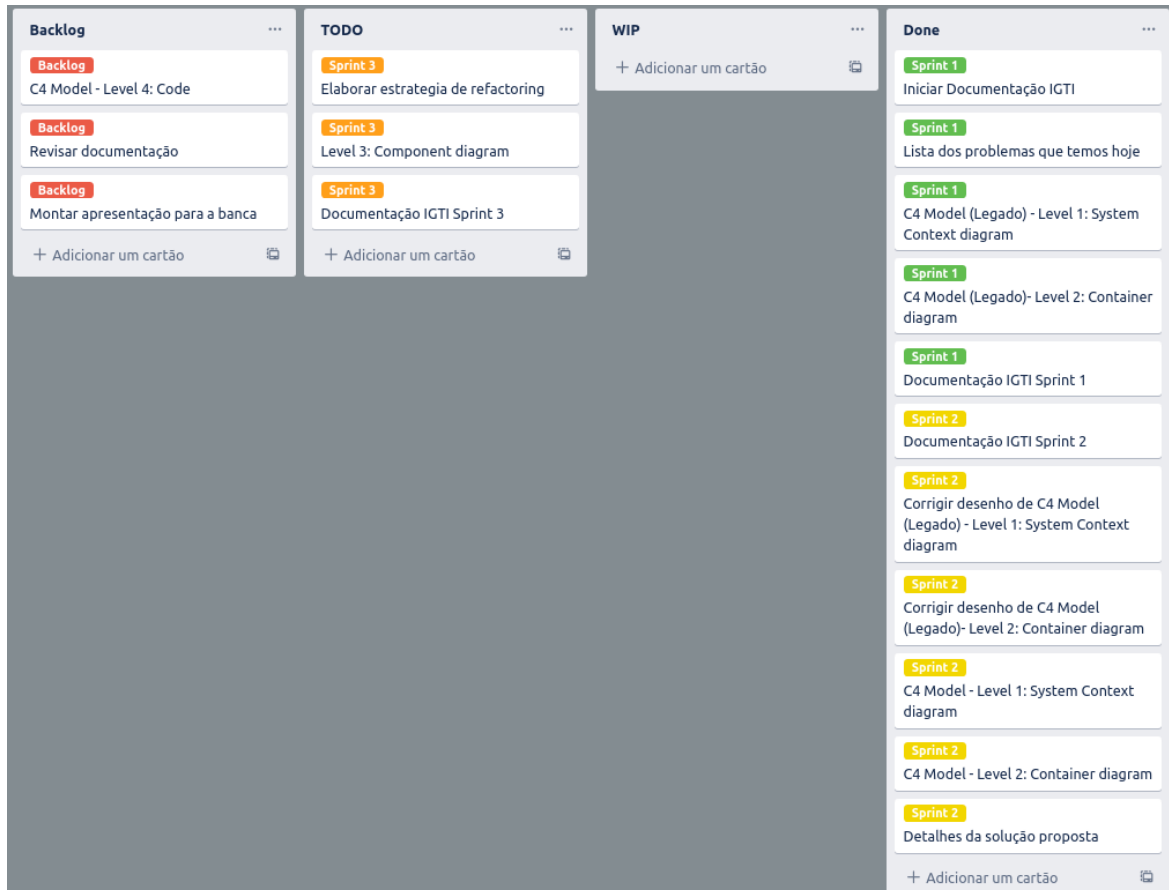
2.1.2 Lições aprendidas

- O C4 model é um diagrama muito simples e completo. Demorei um tempo para entender como ele funciona, porém desenvolver o projeto é mais rápido do que eu havia planejado. Talvez eu consiga desenvolver mais diagramas do que esperava até o final do projeto.
- O próprio criador do C4 model sugere que os diagramas sejam feitos por scripts em ferramentas que geram automaticamente o desenho e que não seja usado ferramentas gráficas como o Luccid Charts. Porém, a organização dos diagramas não fica com uma fácil visualização.

2.2 Sprint 2

2.2.1 Solução

- Evidência do planejamento:



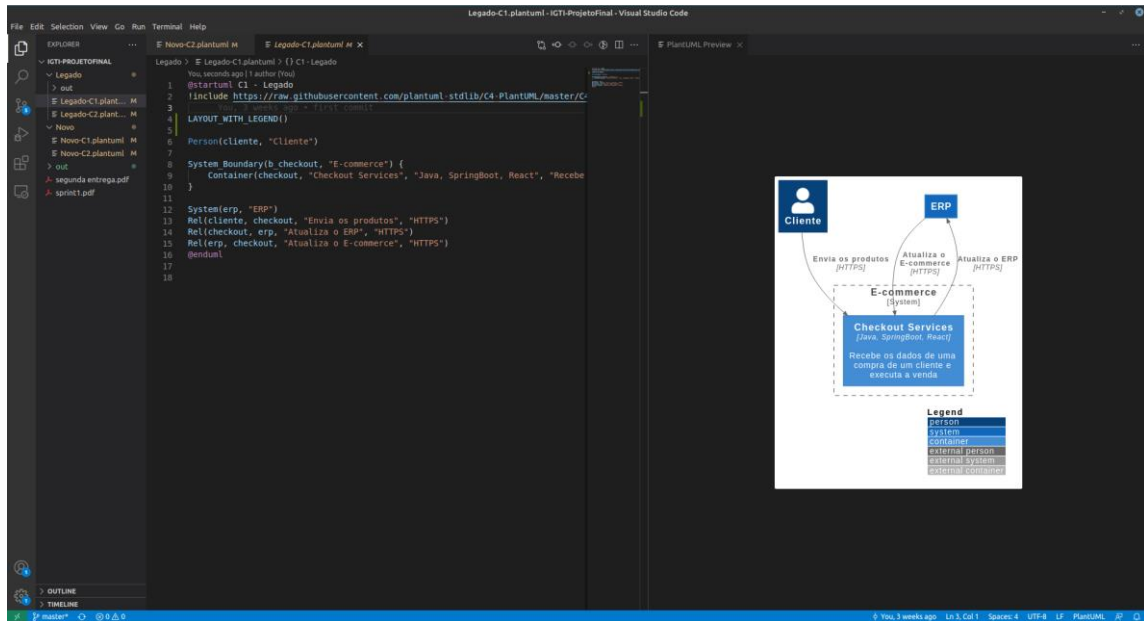
- Evidência da execução de cada requisito:

Diagramas do legado (Correções da Sprint 1)

Correção - Evidencia da codificação do C4 Model (Legado) - Level 1: System Context diagram

Desenvolvido em Visual Studio Code com o plugin do plantUML

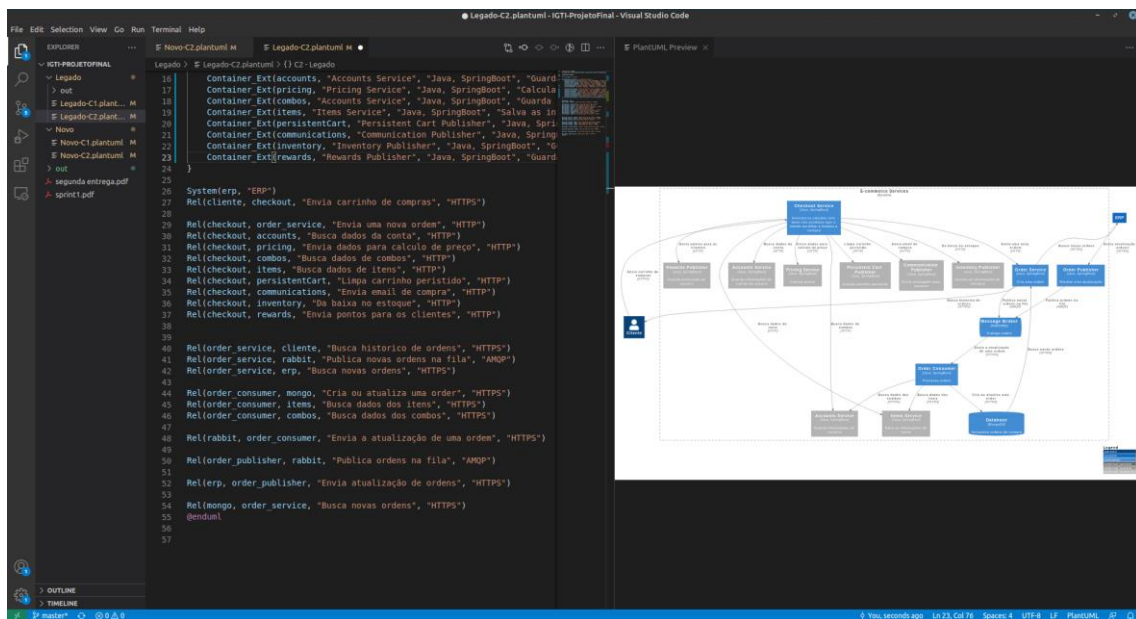
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Legado/Legado-C1.plantuml>



Correção - Evidência da codificação do C4 Model (Legado) - Level 2: Container diagram

Desenvolvido em Visual Studio Code com o plugin do plantUML

Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Legado/Legado-C2.plantuml>

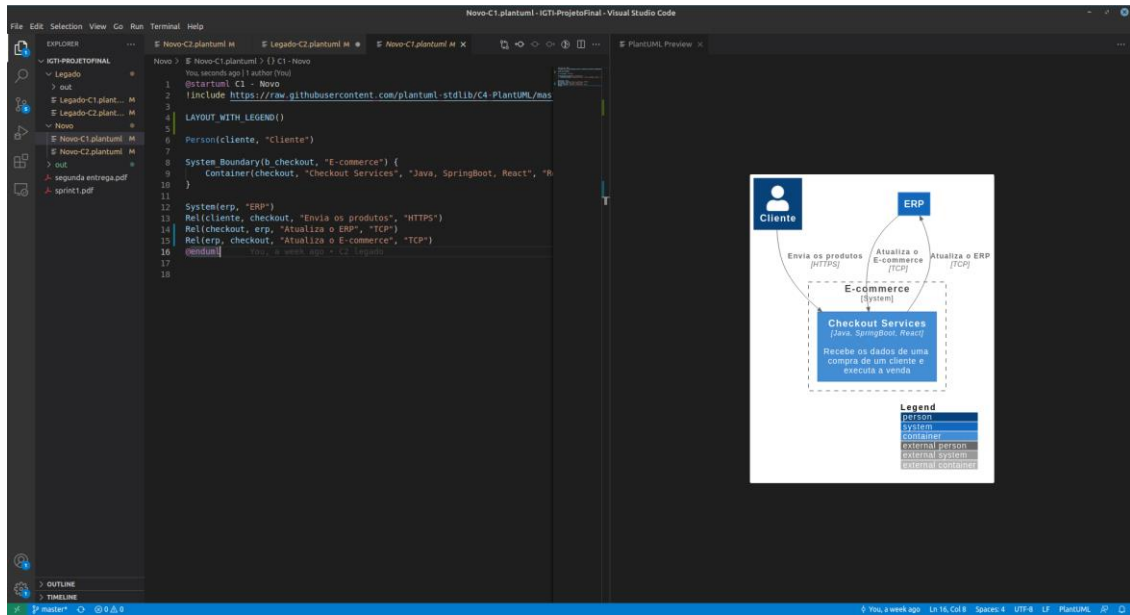


Diagramas da solução proposta

Evidência da codificação do C4 Model - Level 1: System Context diagram

Desenvolvido em Visual Studio Code com o plugin do plantUML

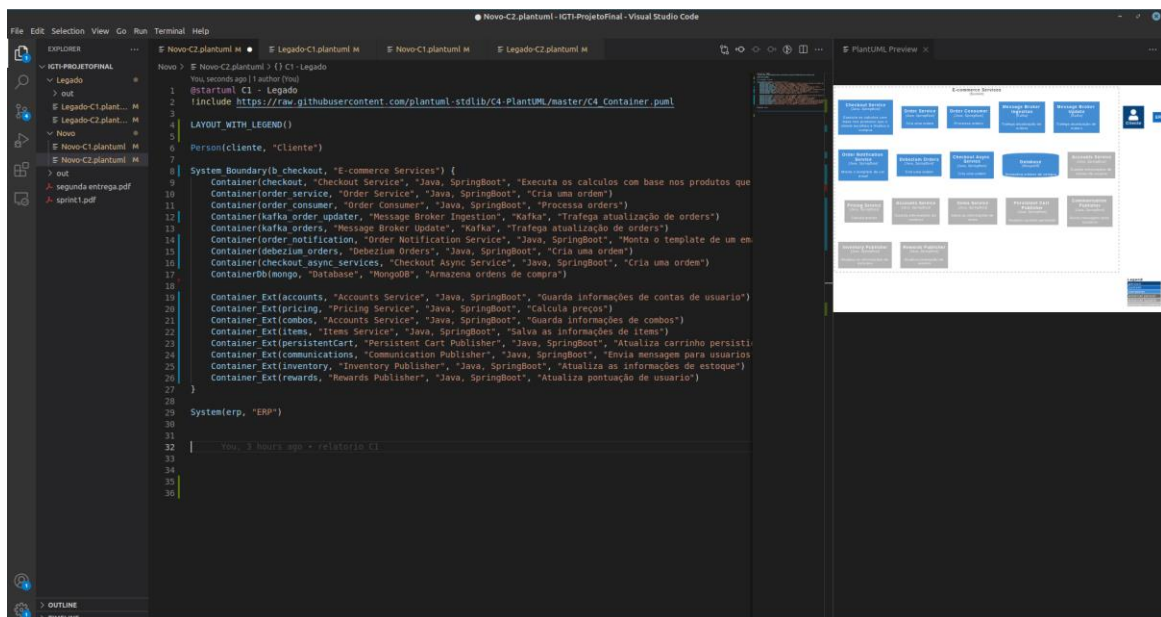
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C1.plantuml>



Evidencia da codificação do C4 Model - Level 2: Container diagram

Desenvolvido em Visual Studio Code com o plugin do plantUML

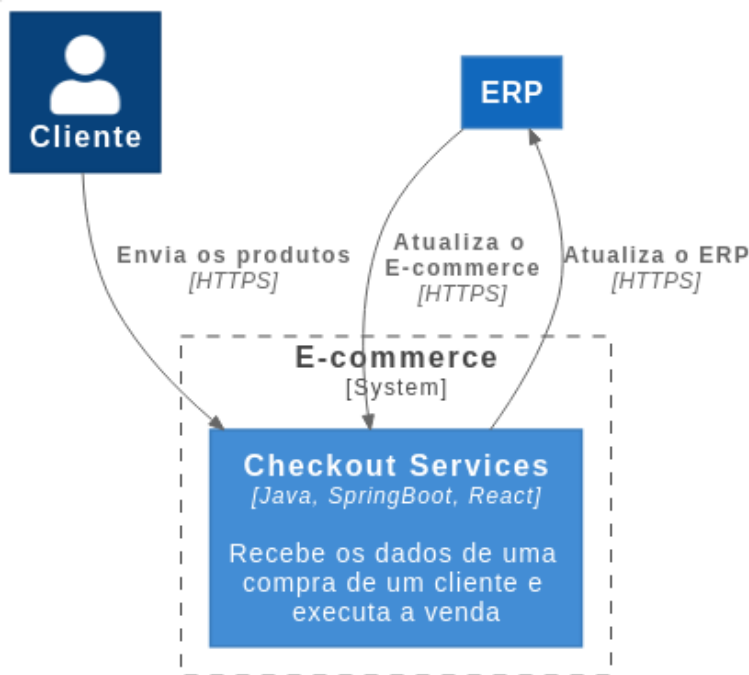
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C2.plantuml>



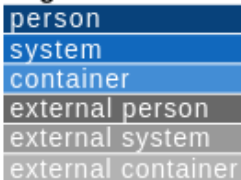
- Evidência dos resultados:

Diagramas do legado (Correções da Sprint 1)

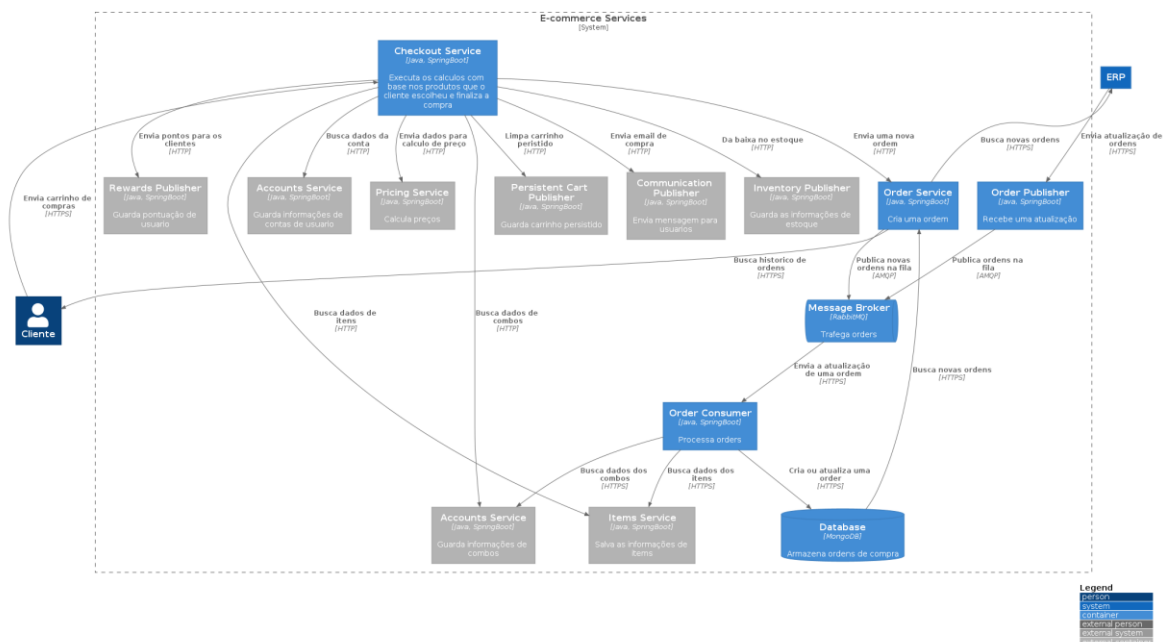
Correção C4 Model (Legado) - Level 1: System Context diagram



Legend

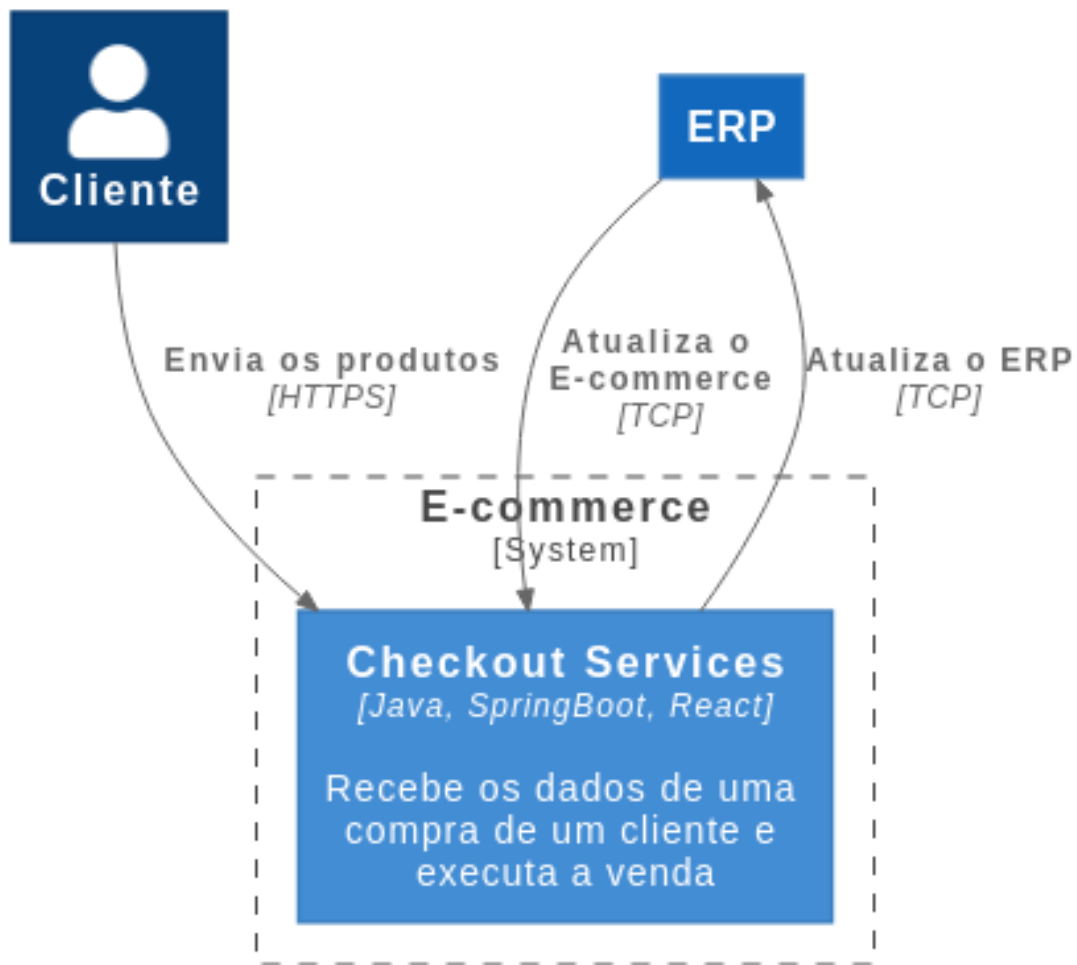


Correção C4 Model (Legado) - Level 2: Container diagram



Diagramas da solução proposta

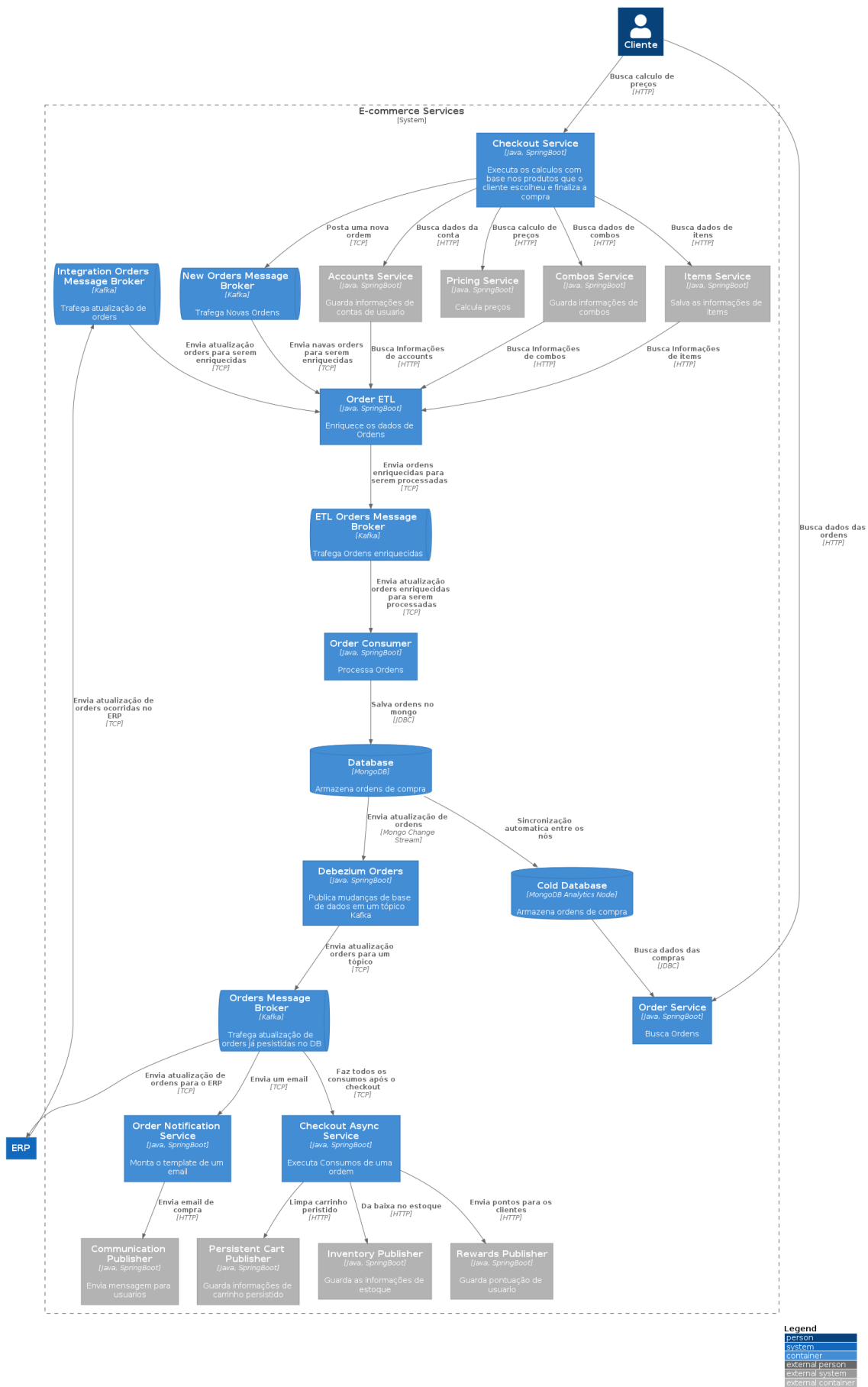
C4 Model - Level 1: System Context diagram



Legend

person
system
container
external person
external system
external container

C4 Model - Level 2: Container diagram



Detalhes da solução proposta

As mudanças propostas na nova arquitetura buscaram reduzir o escopo dos microsserviços existentes e trabalhando melhor com processos assíncronos. O Kafka foi muito usado pois é capaz de gerenciar e trafegar grandes volumes de dados de forma constante e segura.

Order ETL e ETL Orders Message Broker: Na arquitetura legada tínhamos a necessidade de chamar outros microsserviços no order-consumer. Na arquitetura atual segregamos essa responsabilidade para um microsserviço e um tópico kafka dedicados.

Debezium Orders e Orders Message Broker: Uma necessidade constante da plataforma é alertar outros sistemas da criação e atualizações de uma ordem. Usando o Debezium todas as alterações ocorridas na base de dados são publicadas em um tópico Kafka.

Order Notification Service: todos os e-mails que eram enviados pelo microsserviço de checkout foram repassados para esse novo serviço. Ele consome as atualizações do Order Message Broker, gera layout e repassa para o serviço de comunicação da plataforma. Foi necessário um microsserviço só para este fim, pois o time de produto já vê a necessidade de layouts mais modulares e customizáveis por país e status de uma ordem.

Checkout Async Service: todas as validações, consumos e atualizações que aconteciam de forma assíncrona no microsserviço de checkout foram repassados para esse novo serviço. Ele consome as atualizações do Order Message Broker e executa as ações.

New Orders Message Broker: Foi substituído o processo de geração de uma ordem, agora ele é completamente assíncrono e possui um tópico Kafka que traz uma segurança maior na integração entre os sistemas, já que é possível consumir novamente as mensagens em caso de alguma instabilidade.

Integration Orders Message Broker: Foi substituído o processo integração de uma ordem, agora ele é completamente assíncrono e possui um tópico Kafka que traz uma segurança maior na integração entre os sistemas, já que é possível consumir novamente as mensagens em caso de alguma instabilidade.

Cold Database: O MongoDB possibilita criar nós de somente leitura chamado nós analíticos. Seguindo os conceitos de CQRS (Command Query Responsibility Segregation) podemos usar a base principal apenas para criação e atualização de orders e o nó analítico (cold database) para leitura.

2.2.2 Lições aprendidas

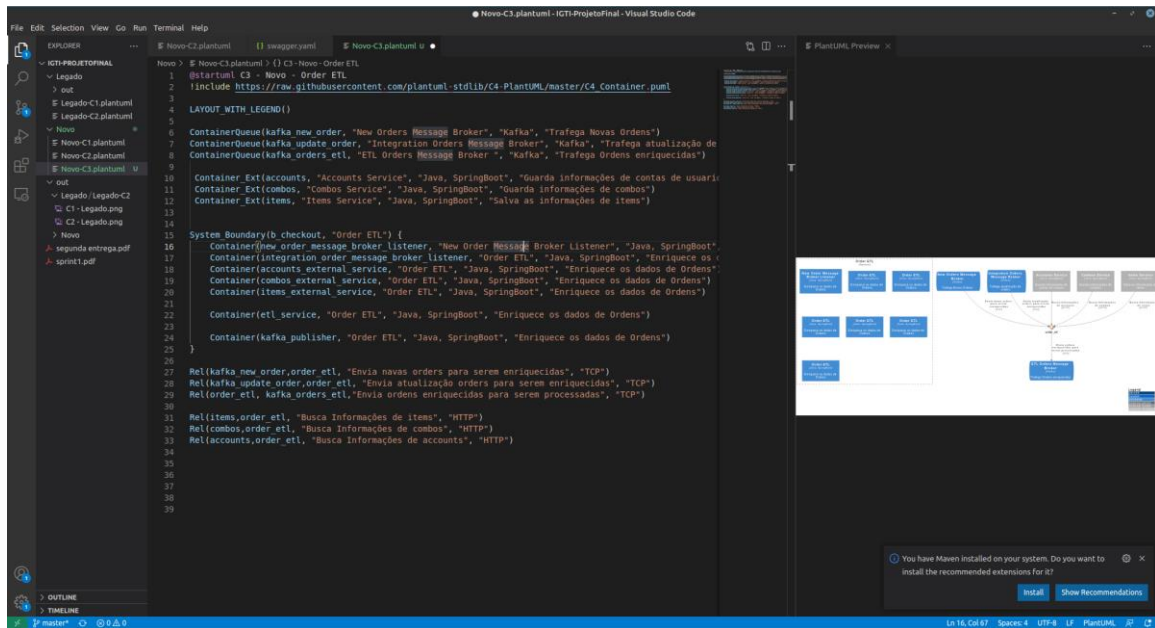
- Estudando um pouco mais os conceitos do C4 model percebi que usei de forma incorreta as separações de sistemas, descobri novos tipos de containers (Externos e filas) e a possibilidade de geração de legenda. Após essas descobertas resolvi corrigir os diagramas já feitos na sprint 1 e também consegui cumprir as tasks antes planejadas para essa sprint.
- C4 model tem se mostrado cada dia melhor e mais simples. Seus containers padrão já abrangem muitos cenários e ainda existe a possibilidade de customiza-los se caso necessário.

- Como o projeto é focado em melhorar uma arquitetura atual, pude aproveitar muito dos diagramas do legado nos diagramas da nova arquitetura proposta.

2.3 Sprint 3

2.3.1 Solução

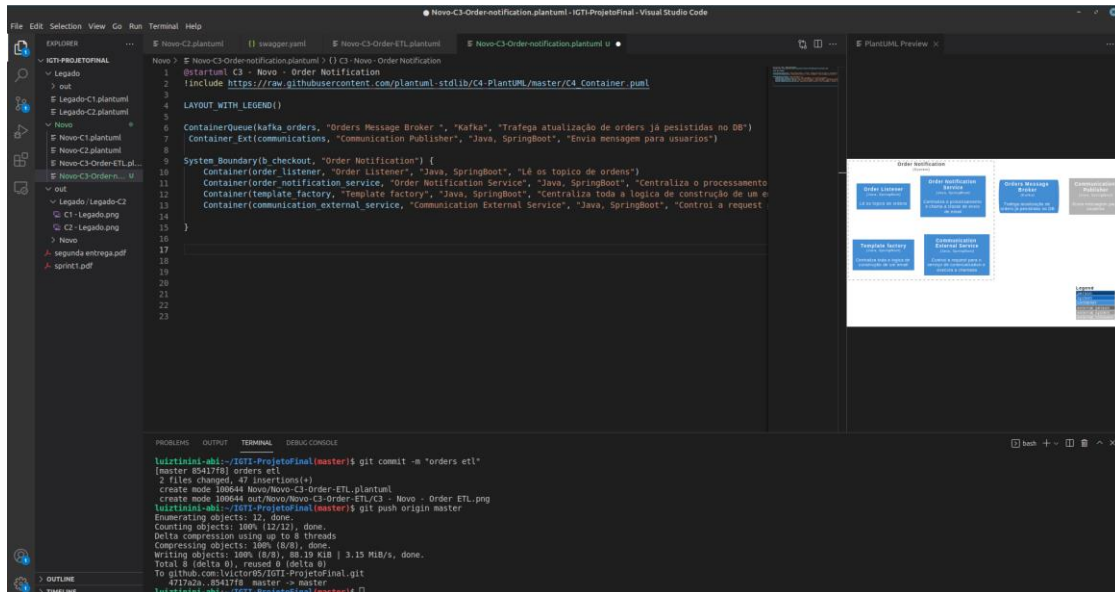
- Evidência do planejamento:
- Evidência da execução de cada requisito:



Evidencia da codificação do C4 Model Orders ETL - Level 3: Component diagram

Desenvolvido em Visual Studio Code com o plugin do plantUML

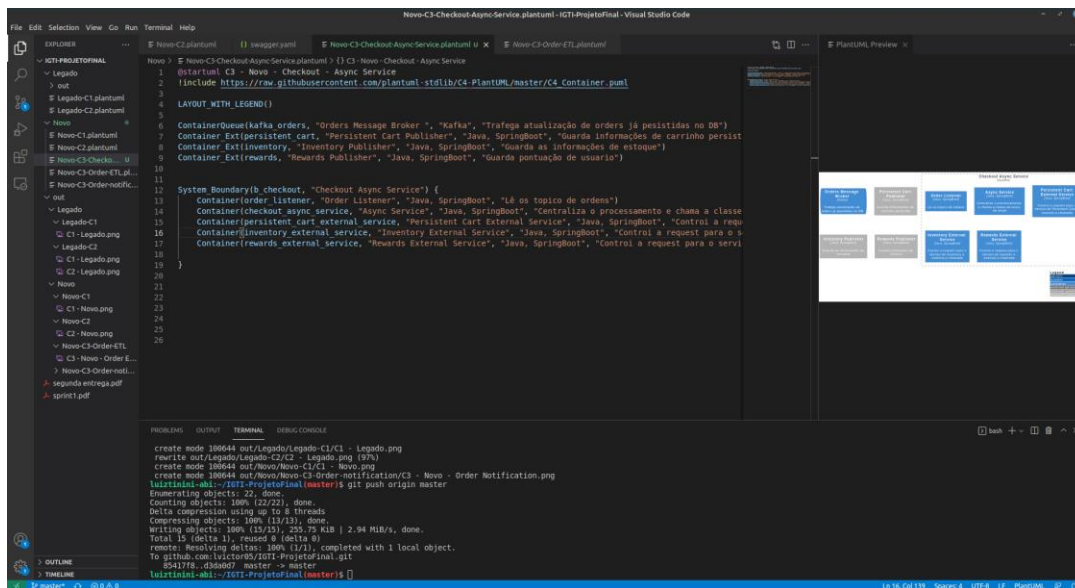
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C3-Order-ETL.plantuml>



Evidencia da codificação do C4 Model Order Notification - Level 3: Component diagram

Desenvolvido em Visual Studio Code com o plugin do plantUML

Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C3-Order-notification.plantuml>



Evidencia da codificação do C4 Model Checkout Async Service - Level 3: Component diagram

Desenvolvido em Visual Studio Code com o plugin do plantUML

Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C3-Checkout-Async-Service.plantuml>

- Evidência dos resultados:

Elaborar estratégia de implementação e refatoração

O processo de implementação e refatoração vai seguir a seguinte estratégia. Primeiro vamos criar os serviços e integrações novas e depois começar a modificar os serviços já existentes. Essa lista é apenas uma sequência lógica high level da melhor forma de começar o processo de refatoração.

1. Implementar Debezium Orders e Orders Message Broker.
2. Criar integração entre ERP e Order Message Broker.
3. Criar serviço de Order Notification.
4. Remover envio de e-mail do Checkout Service.
5. Criar serviço Checkout Async Service.
6. Remover serviços assíncronos do Checkout Service.
7. Criar Integration Orders Message Broker e plugar ERP ao tópico.
8. Criar Serviços de Order ETL e ETL Order Message Broker
9. Plugar ETL Order Message Broker ao Order Consumer
10. Criar Integração entre checkout e criar tópico New Order Message Broker
11. Criar integração entre New Order Message Broker e Orders ETL
12. Criar nó analítico para a base de ordens.
13. Remover criação de ordens do serviço order-service
14. Conectar Order-Service ao nó analítico do banco de dados
15. Remover implementações do RabbitMQ do order-consumer
16. Remover enriquecimentos do order-consumer
17. Desativar microsserviço order-publisher

C4 Model Orders ETL - Level 3: Component diagram

external container

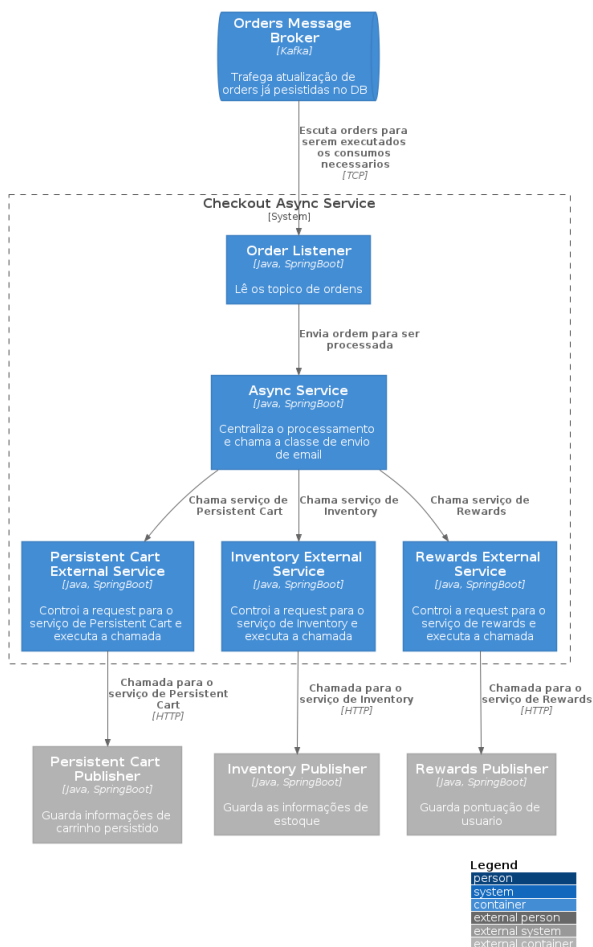
A arquitetura interna do microsserviço tem como premissa ser simples e de fácil manutenção. Foi utilizado um pattern muito comum em projetos, o MVC (Model view controller). As camadas entrada, processamento e saída são bem definidas seguindo os conceitos de SOLID e trazendo uma fácil manutenção até mesmo para desenvolvedores mais inexperientes.

C4 Model Order Notification - Level 3: Component

external container

Ainda mantendo os conceitos de MVC e SOLID, o serviço de order notification vai centralizar todas as suas principais regras de negócio na classe de template factory. Usando essa abordagem é possível criar vários layouts para diversas necessidades de negócio diferentes. A estrutura do template factory pode ser explorada melhor no quarto level do C4 model, porém não será abordado por esse trabalho.

C4 Model Checkout Async Service - Level 3: Component



O Serviço de Checkout Async service foi desenhado também no conceito de MVC e seguindo os conceitos de SOLID. O Serviço Async Service fará todas as chamadas de forma assíncrona e não bloqueante. Seguindo essa arquitetura é fácil e rápido implementar novos consumos se caso necessário.

2.3.2 Lições aprendidas

3. Considerações Finais

3.1 Resultados

Por meio de um texto detalhado, apresente os principais resultados alcançados pelo seu Projeto Aplicado.

Cite os pontos positivos e negativos, as dificuldades enfrentadas e as lições aprendidas durante todo o processo.

3.2 Contribuições

Apresente quais foram as contribuições que o seu Projeto Aplicado trouxe para que o Desafio proposto fosse solucionado.

Cite, por exemplo, as inovações, as vantagens sobre os similares, as melhorias alcançadas, entre outros.

3.3 Próximos passos

Descreva quais são os próximos passos que poderão contribuir com o aprimoramento da solução apresentada pelo seu Projeto Aplicado.