



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

	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?
<b>Desenvolvedores</b>	Computador , teclado e mouse	Home office ou escritório	Atuando no desenvolvimento das demandas da empresa	Desenvolvimento das funcionalidades e correção de bugs
<b>Pessoas de produto</b>	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.
<b>Pessoas de projeto</b>	Computador , teclado e mouse	Home office ou escritório	Atuando na organização dos processos da empresa	Organização de pessoas e processos.
<b>Alta gerencia</b>	Computador , teclado e mouse	Home office ou escritório	Negociando novos contratos com novos clientes	Proporcionando novos negócios para as empresas.

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



### 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?

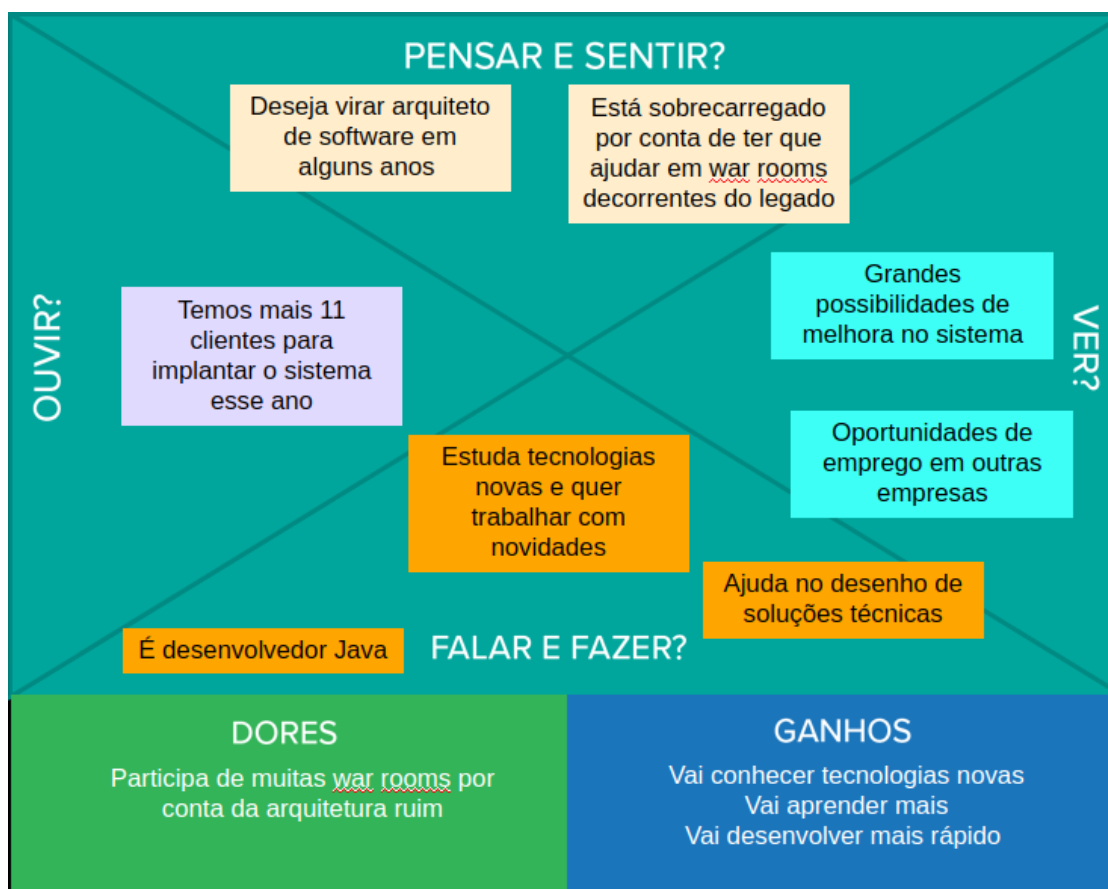
- Ajuda no desenho de soluções técnicas
- É desenvolvedor Java
- Estuda tecnologias novas e quer trabalhar com novidades

### 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

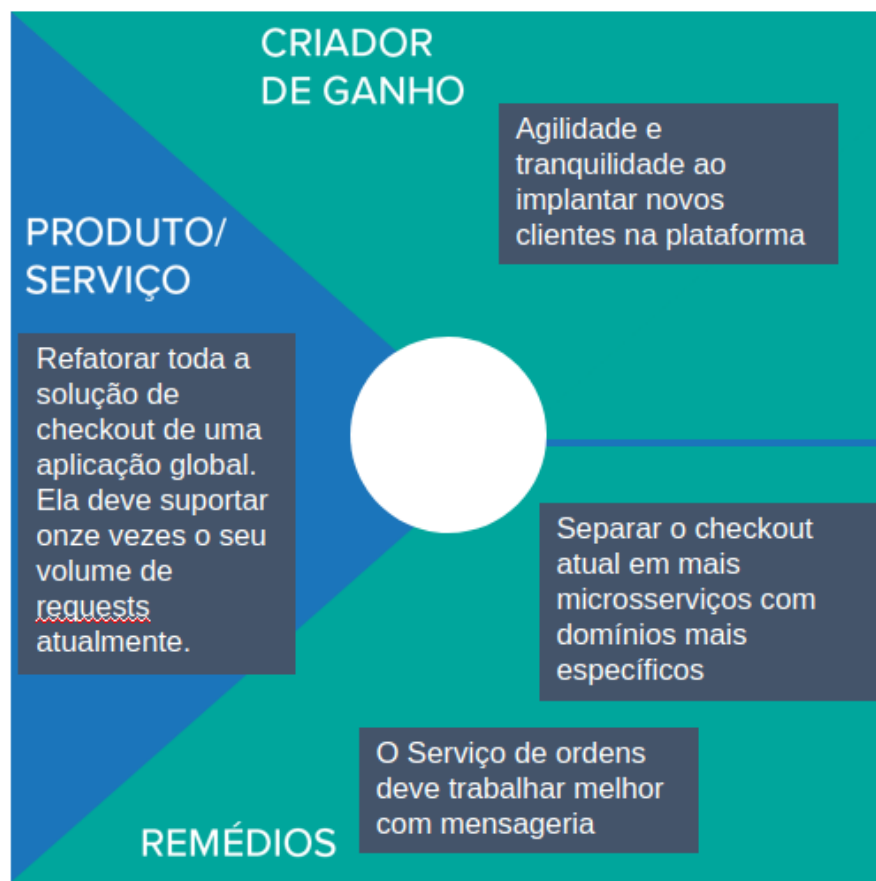


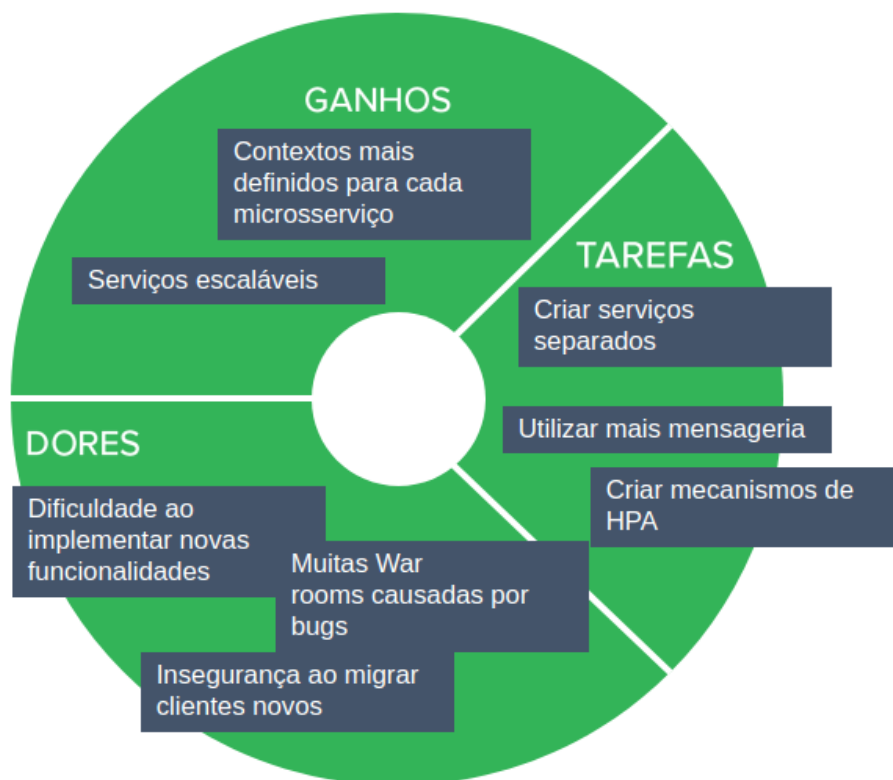
### 1.1.3 Benefícios e Justificativas

Itens	Checkout Service	Order Service	Order Notification	Post Order Service
<b>Objetivos</b>	Verificar dados da compra e finalizar pedido	Armazenar os dados da ordem	Notificar cliente sobre sua compra	Consumir estoque e dar pontos.
<b>Atividades</b>	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

<b>Questões</b>	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?
<b>Barreiras</b>	Manter o legado funcionando	Manter o legado funcionando	Manter o legado funcionando	Manter o legado funcionando
<b>Ações do cliente</b>	Fechar um pedido	N/A - Ação automática do sistema	Receber um e-mail ou notificação	N/A - Ação automática do sistema
<b>Funcionalidades</b>	Finalizar Compras	Salvar as ordens	Notificar o usuário	Executar chamadas assíncronas do checkout
<b>Interação</b>	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
<b>Mensagem</b>	Compra efetuada com sucesso	200 - OK	N/A	N/A
<b>Onde ocorre</b>	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
<b>Tarefas aparentes</b>	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

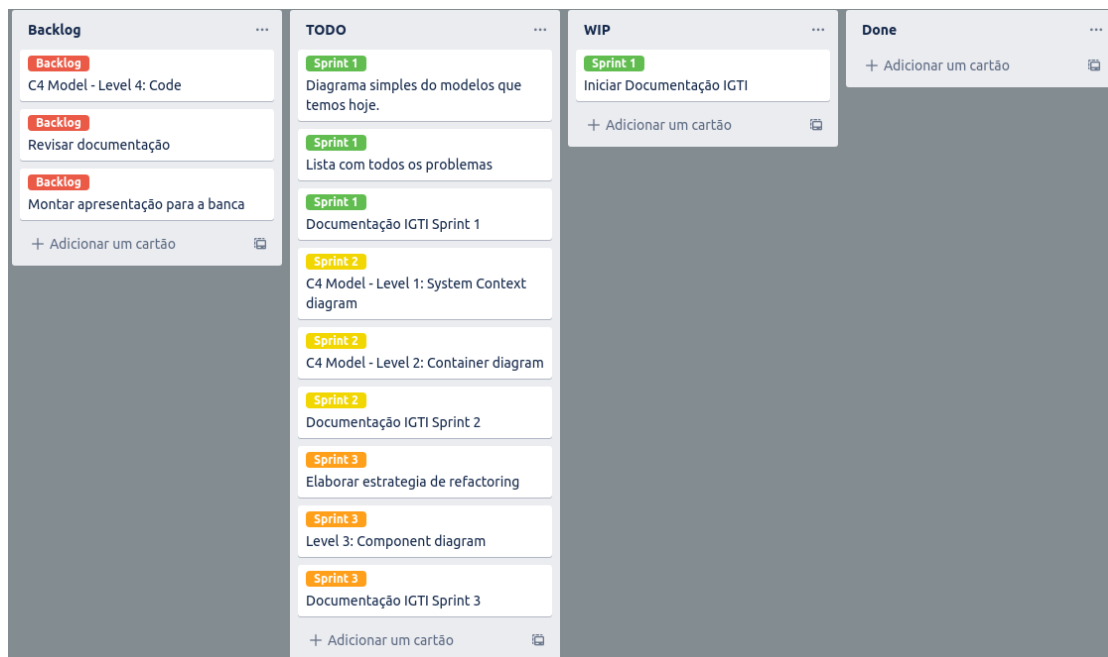
- Esse projeto apenas contemplará a quarta parte do C4 Model se todas os processos foram feitos previamente.
- Os contratos com os Front end e ERP's externos devem ser mantidos
- Os sistemas legados deverão permanecer rodando enquanto a refatoração acontece
- Os sistemas deverão respeitar minimamente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

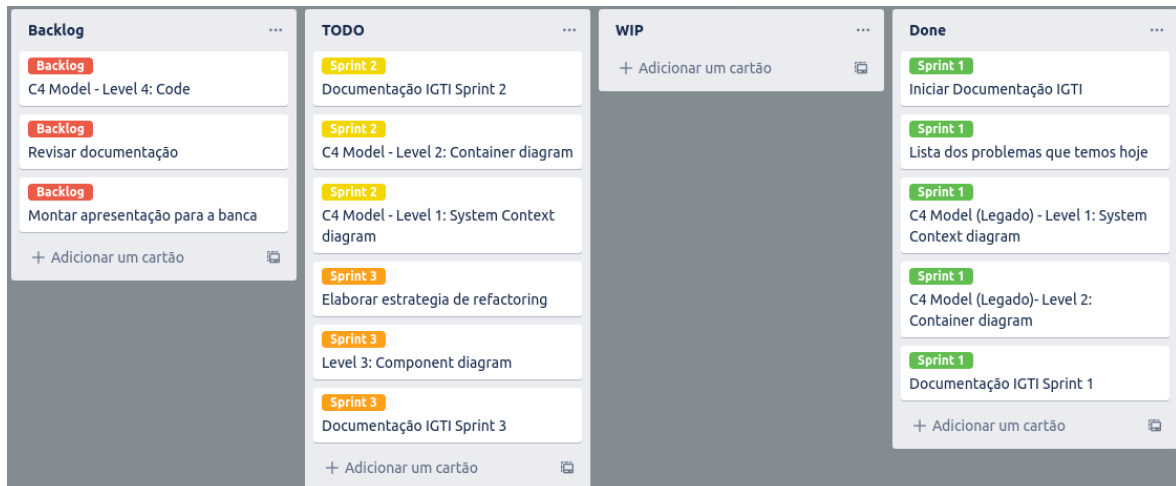


## 2. Área de Experimentação

### 2.1 Sprint 1

#### 2.1.1 Solução

- Evidência do planejamento:

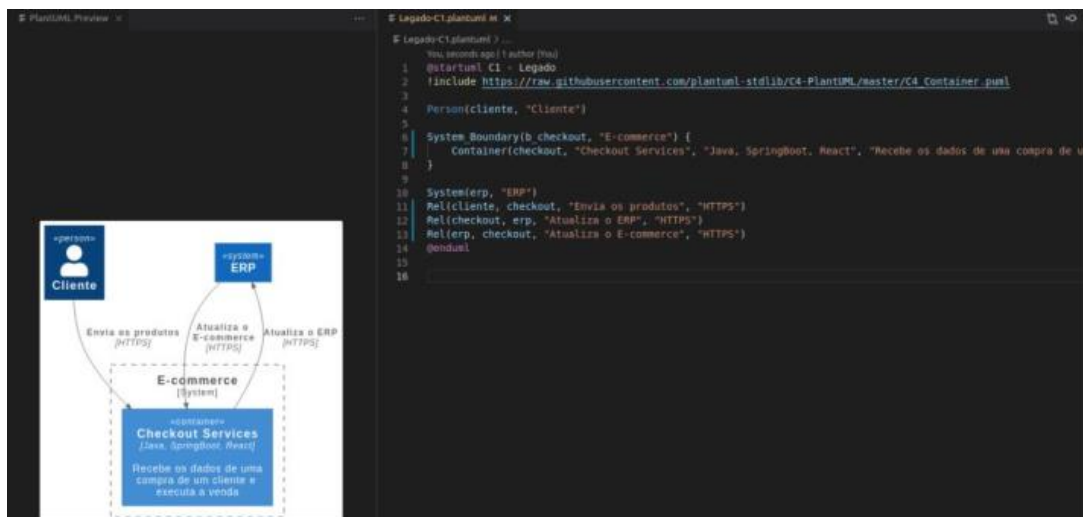


- Evidência da execução de cada requisito:

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

Desenvolvido em visual studio code com o plugin do plantUML

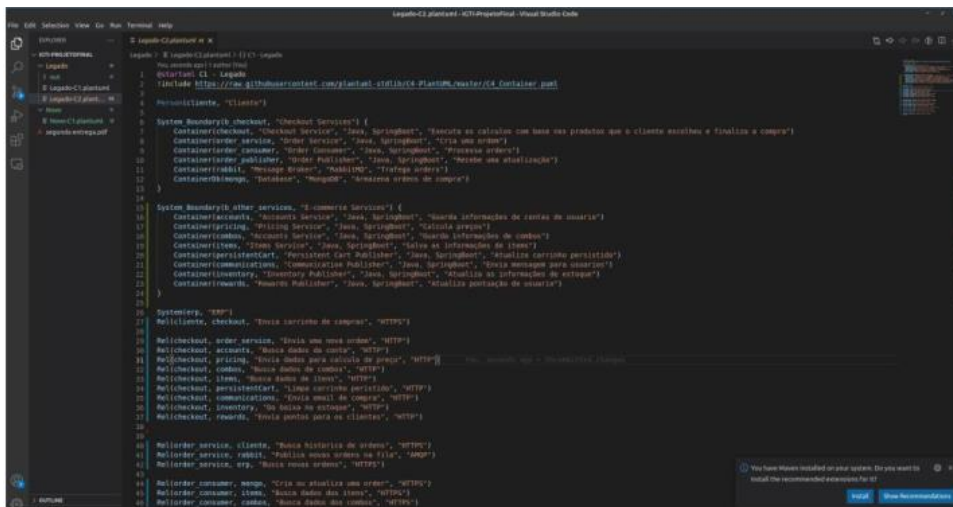
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Legado/Legado-C1.plantuml>



#### 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>



```

1 legado C2.plantuml
2
3 title Legado - C2
4
5 include https://raw.githubusercontent.com/lvictor05/IGTI-ProjetoFinal/master/Legado-C2.plantuml
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

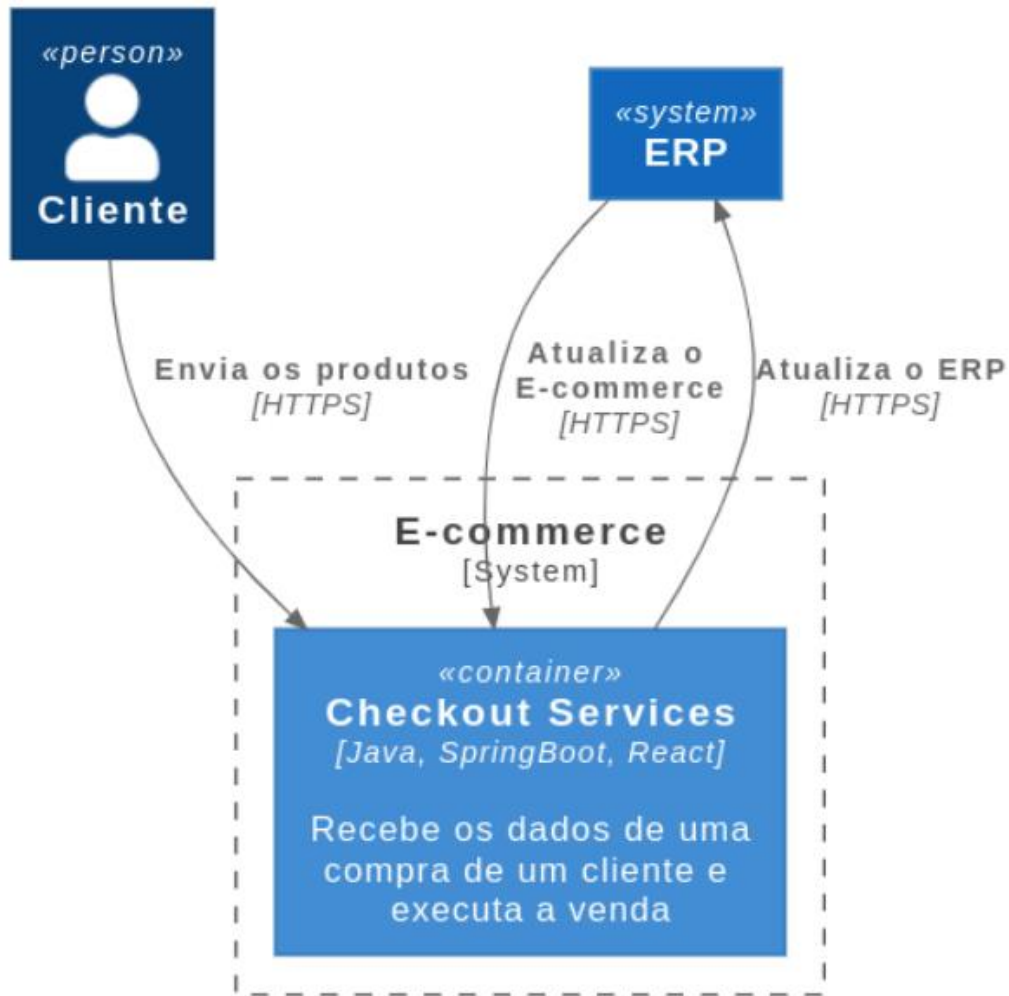
```

- Evidência dos resultados:

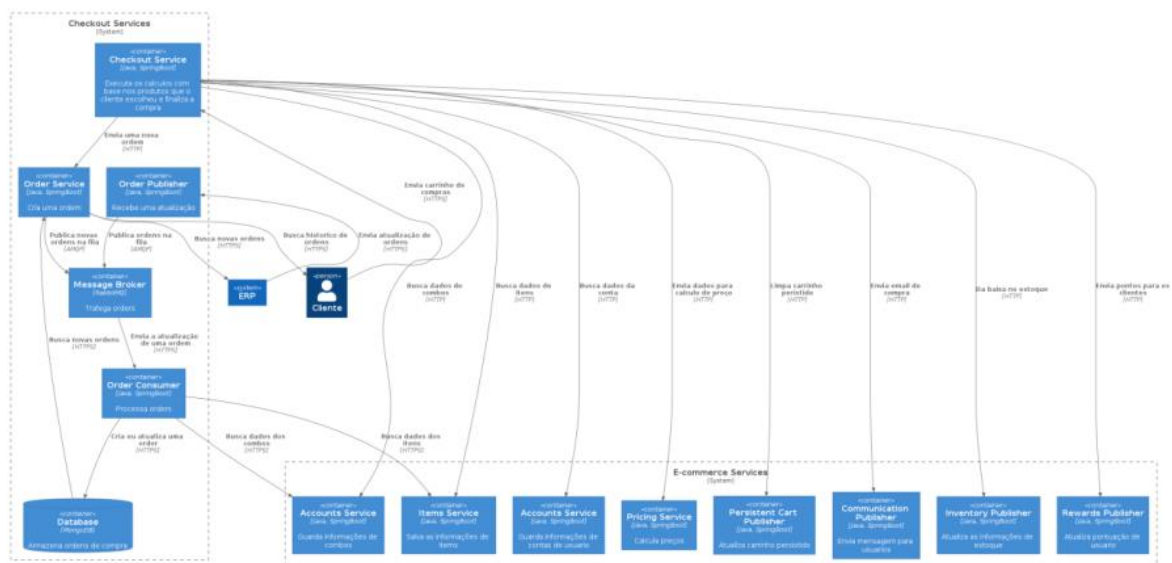
Obs: Para facilitar a visualização, segue abaixo os diagramas completos e o diagrama C2 completo e separados em duas partes com zoom.

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

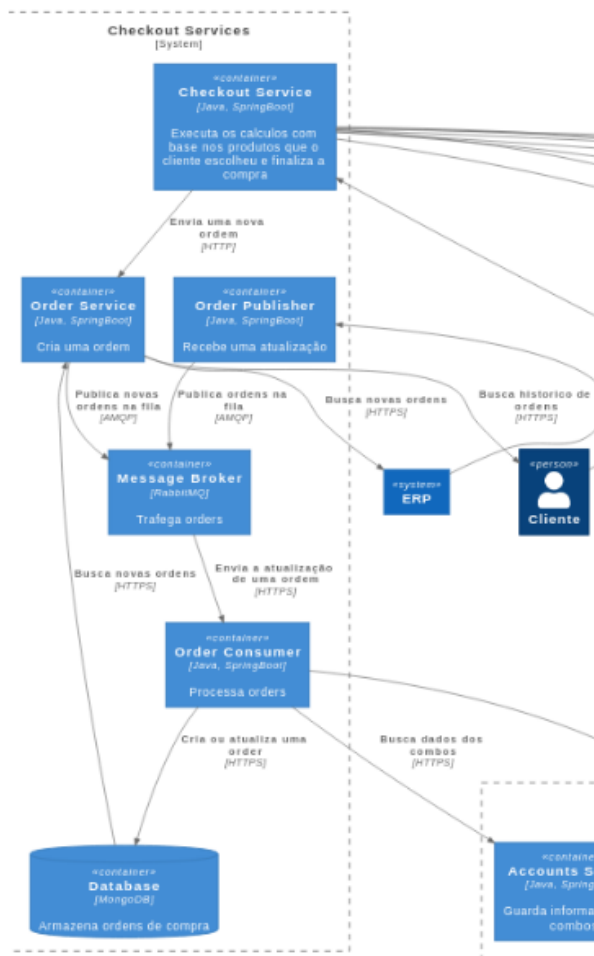
Arquitetura atual - C4 Model (Legado) - Level 1: System Context diagram



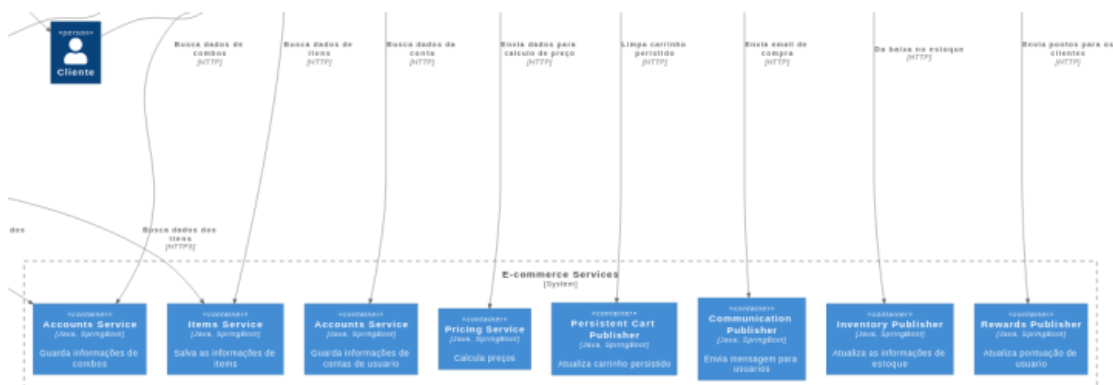
### C4 Model (Legado) - C4 Model (Legado)- Level 2: Container diagram



### 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

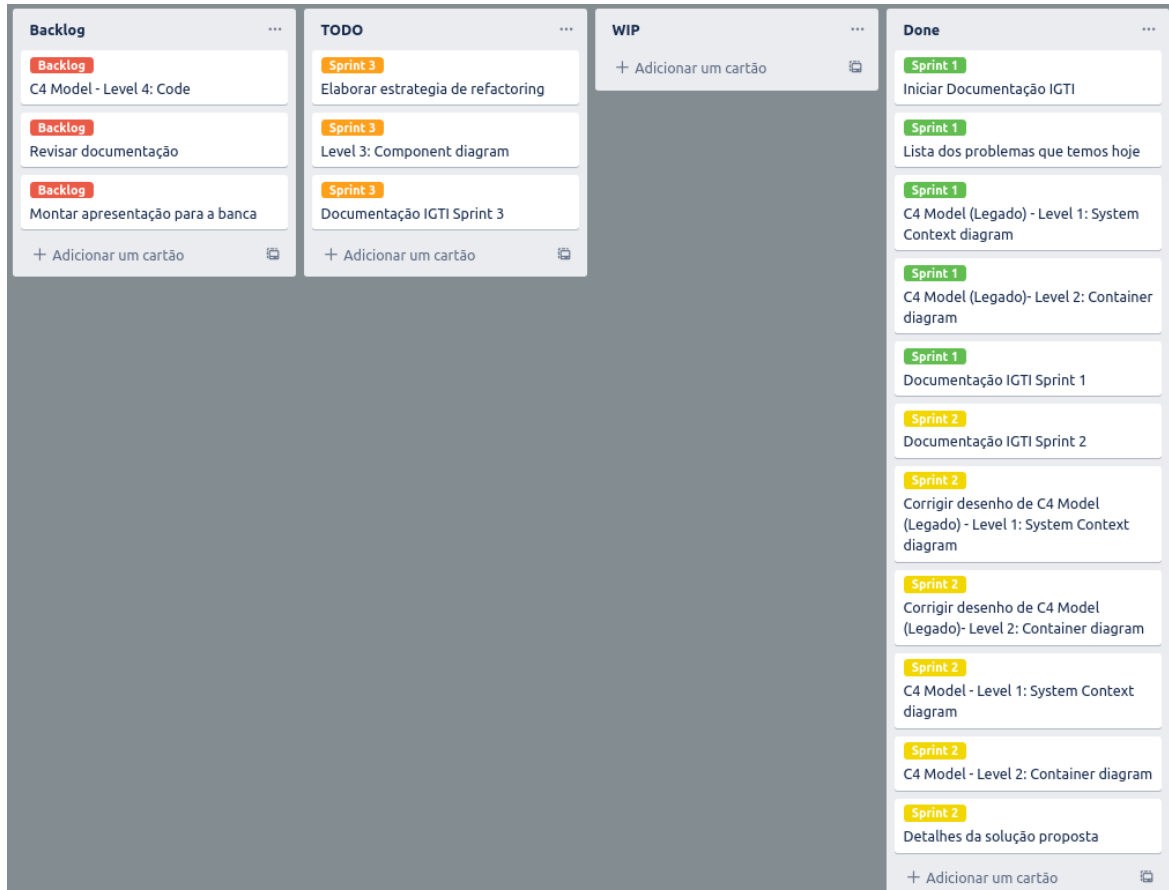
- O C4 model é um diagrama muito simples e completo. Demorei um tempo para entender como ele funciona, porém desenvolver o projeto é mais rápido do que eu havia planejado. Talvez eu consiga desenvolver mais diagramas do que esperava até o final do projeto.
- O próprio criador do C4 model sugere que os 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:



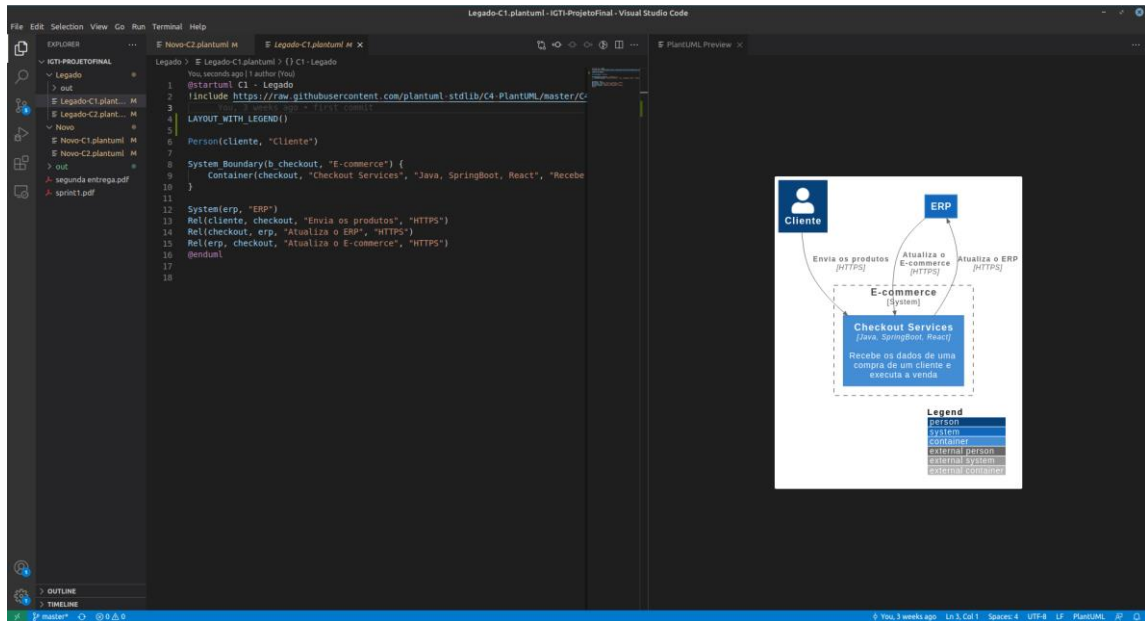
- Evidência da execução de cada requisito:

### 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

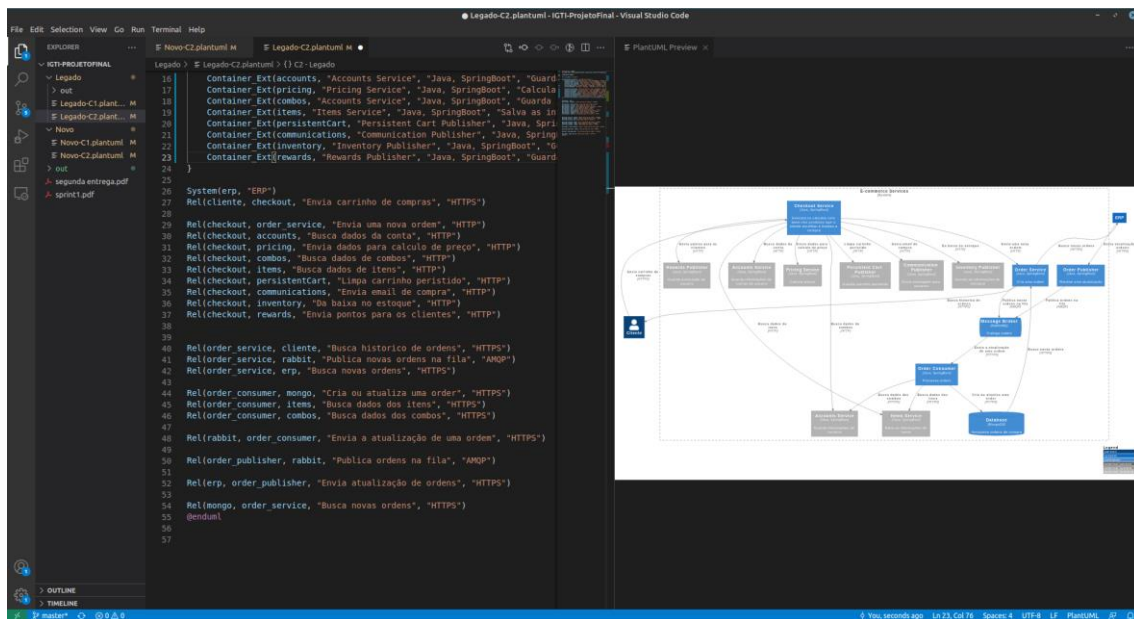
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Legado/Legado-C1.plantuml>



## 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>

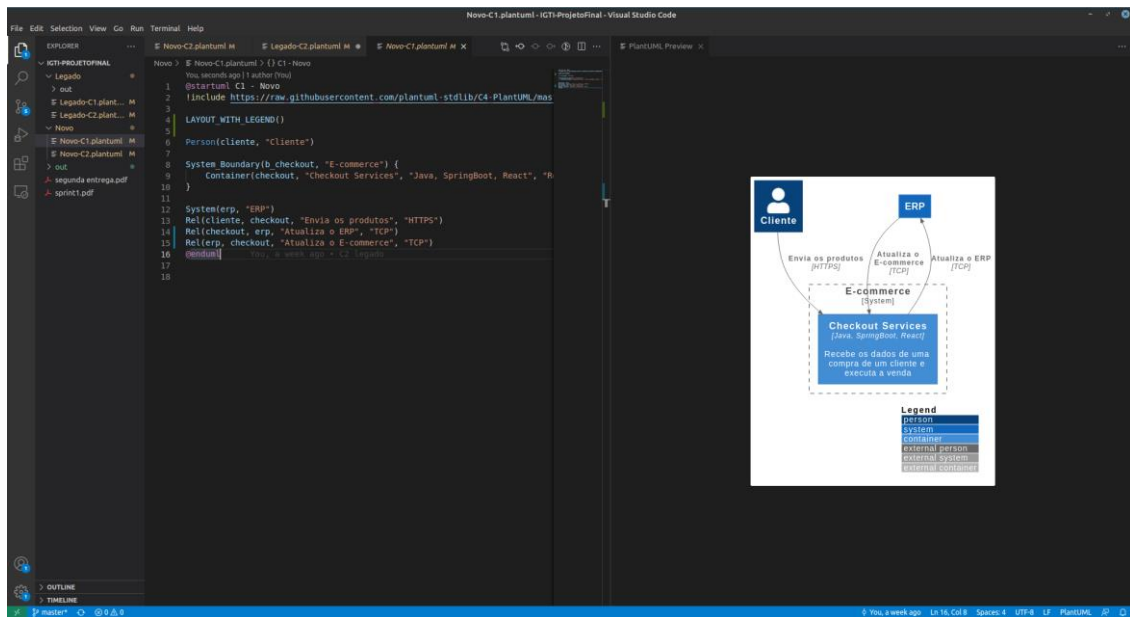


## Diagramas da solução proposta

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

Desenvolvido em Visual Studio Code com o plugin do plantUML

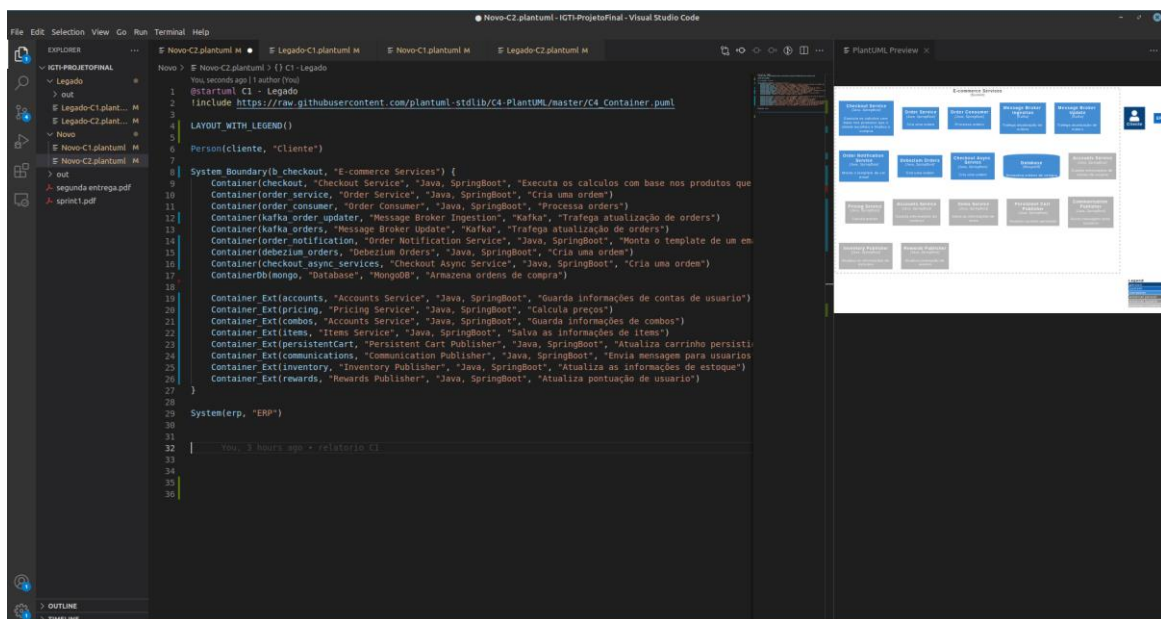
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C1.plantuml>



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

Desenvolvido em Visual Studio Code com o plugin do plantUML

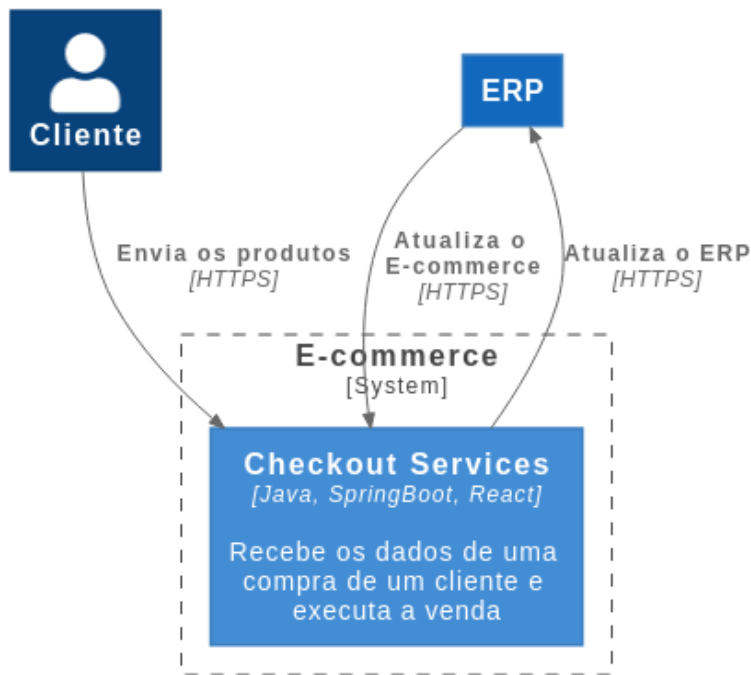
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C2.plantuml>



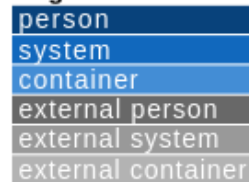
- Evidência dos resultados:

## Diagramas do legado (Correções da Sprint 1)

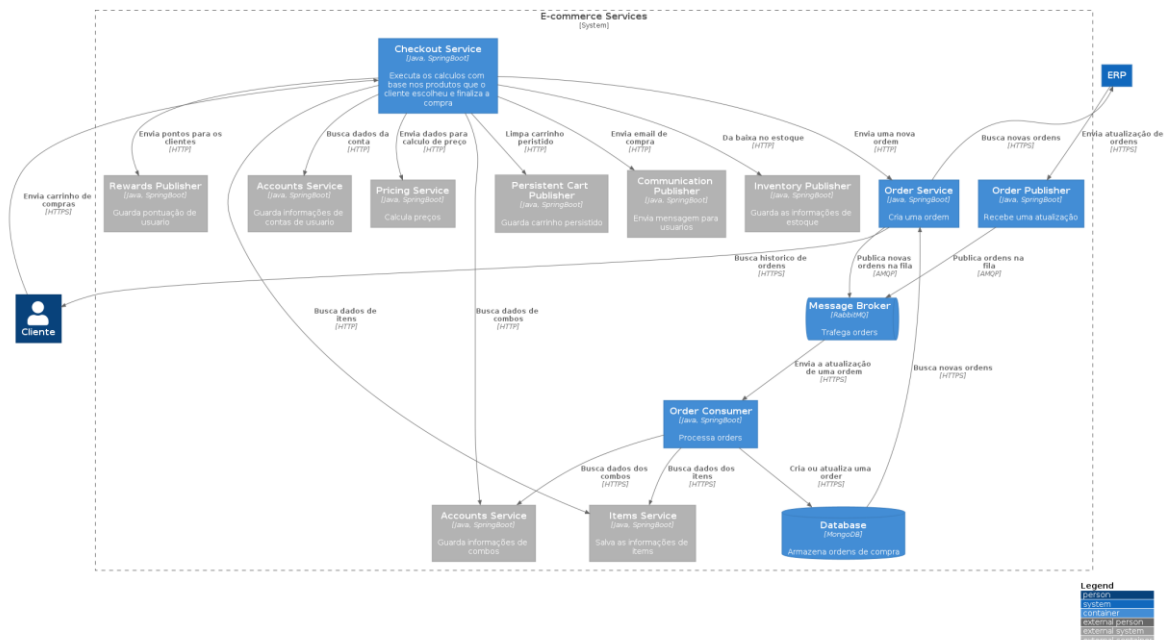
Correção C4 Model (Legado) - Level 1: System Context diagram



### Legend

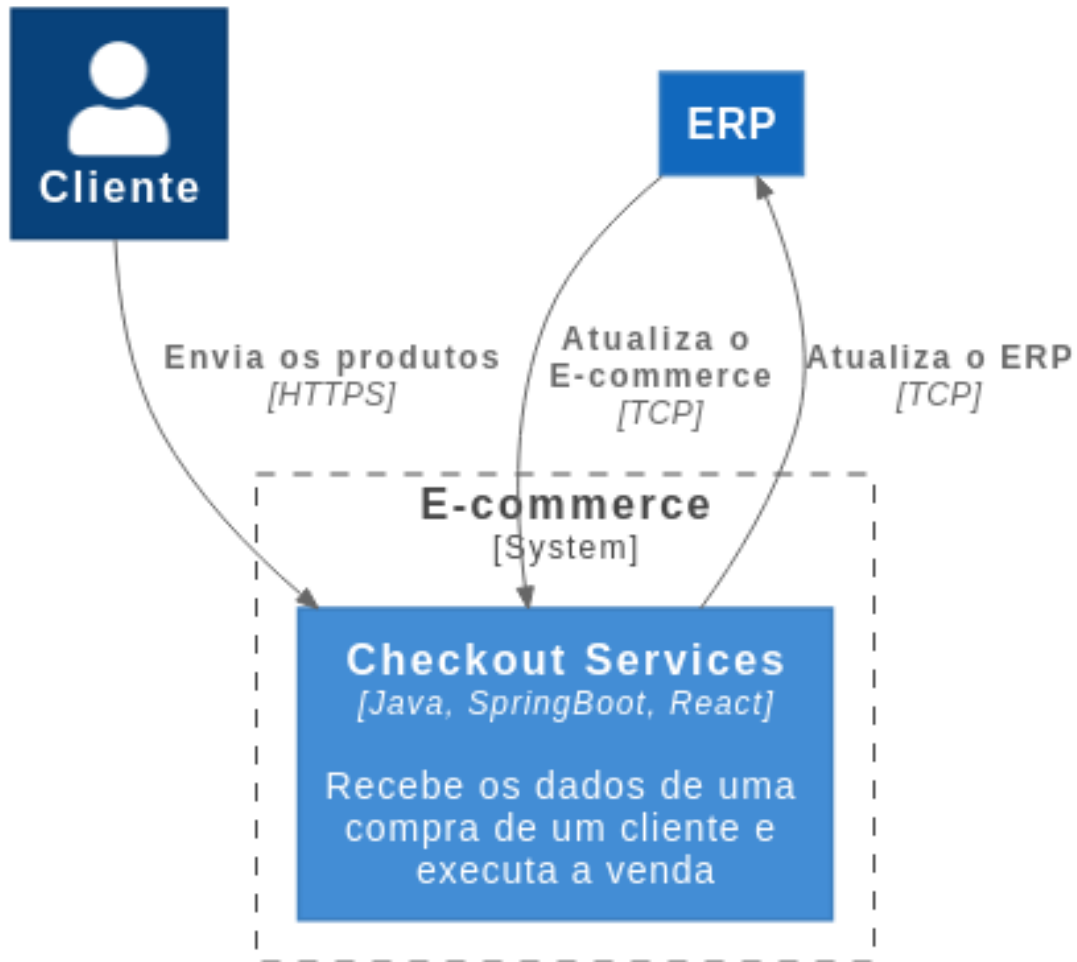


## Correção C4 Model (Legado) - Level 2: Container diagram



## Diagramas da solução proposta

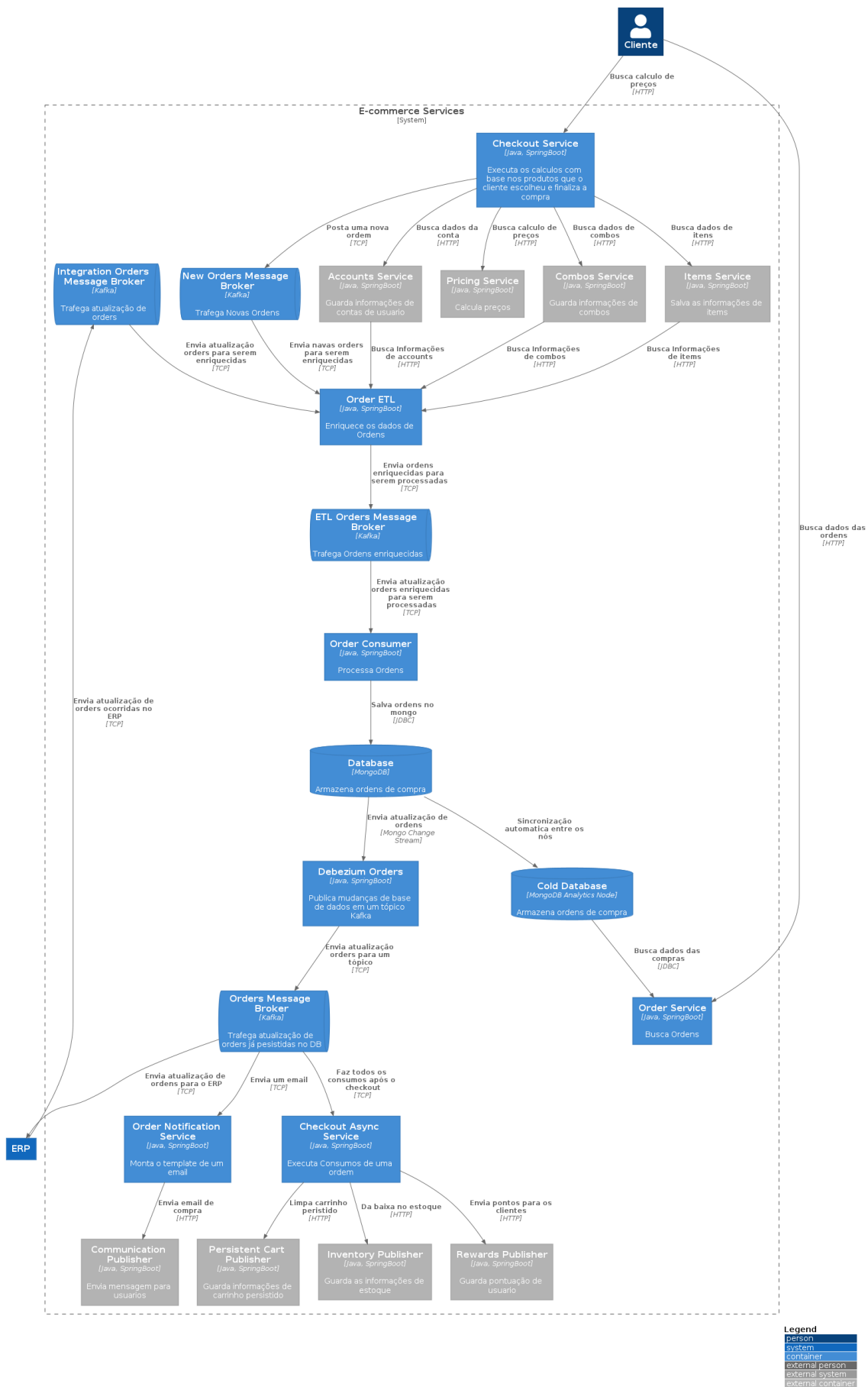
### C4 Model - Level 1: System Context diagram



### 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.

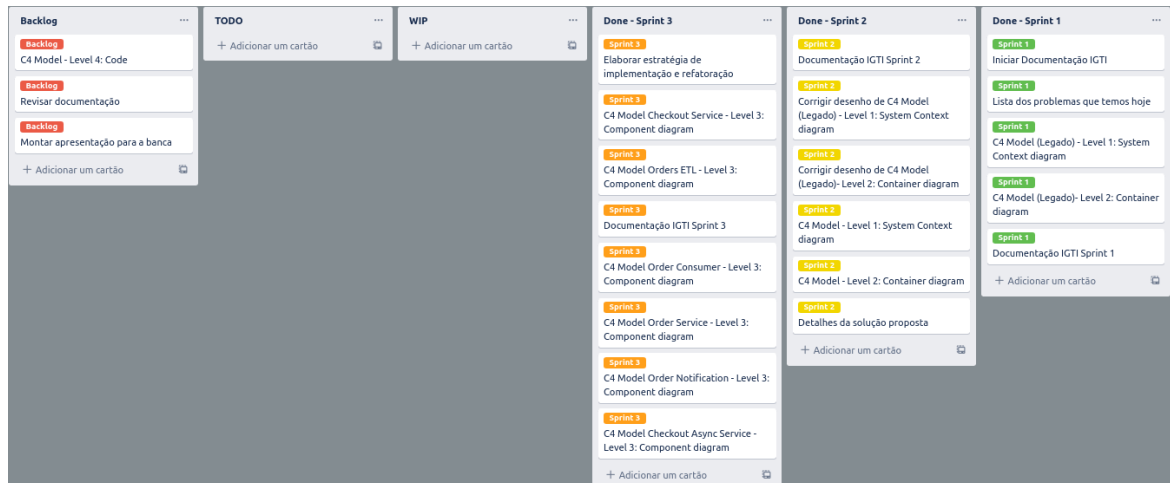
- Como o projeto é focado em melhorar uma arquitetura atual, pude aproveitar muito dos diagramas do legado nos diagramas da nova arquitetura proposta.



## 2.3 Sprint 3

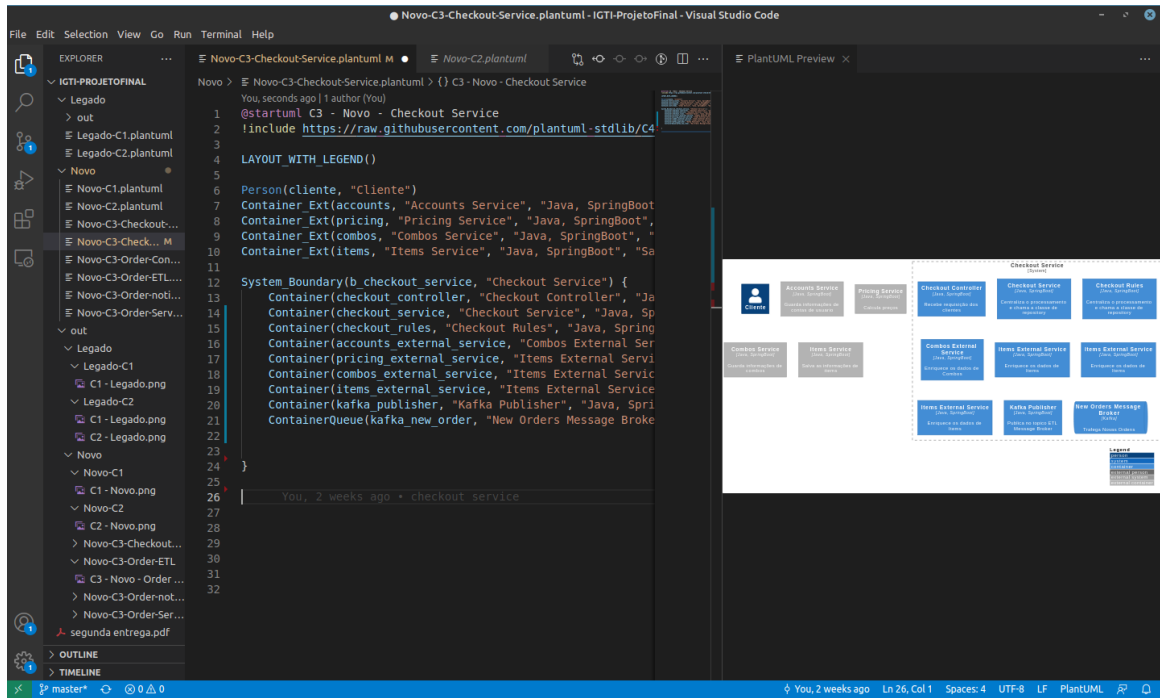
### 2.3.1 Solução

- Evidência do planejamento:



- Evidência da execução de cada requisito:

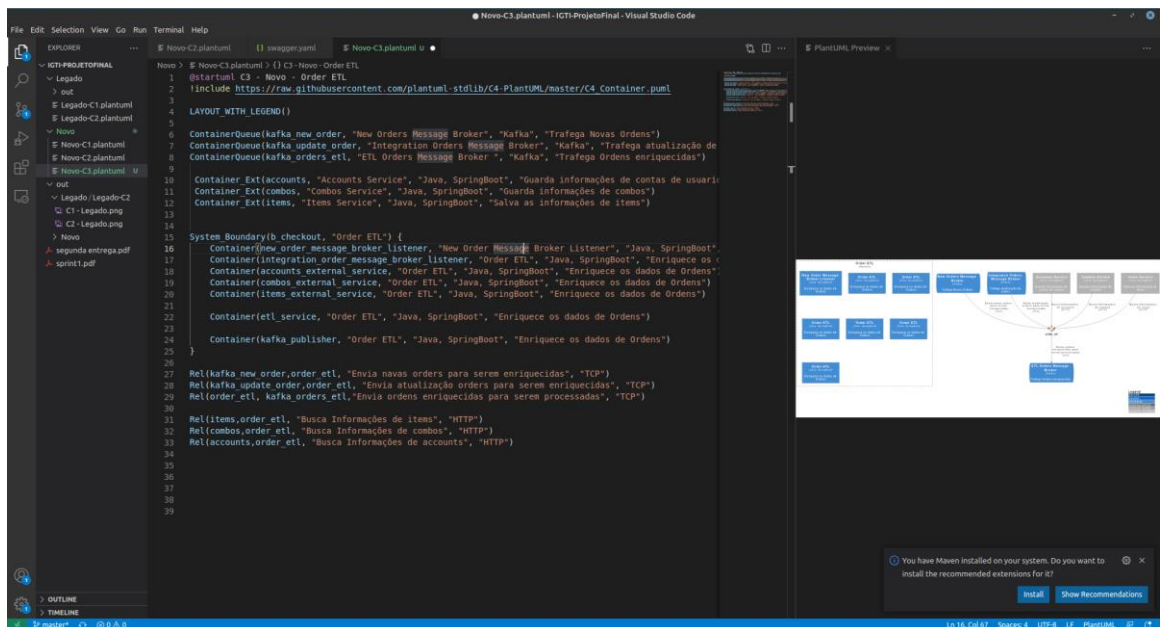
**Evidencia do desenvolvimento da elaborar estratégia de implementação e refatoração:** A estratégia de refatoração é apenas um guia simples do passo a passo que deve ser tomado para que o processo de refatoração seja concluído. Ele teve como ideia principal sempre criar o novo e logo após remover o processo legado.



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

Desenvolvido em Visual Studio Code com o plugin do plantUML

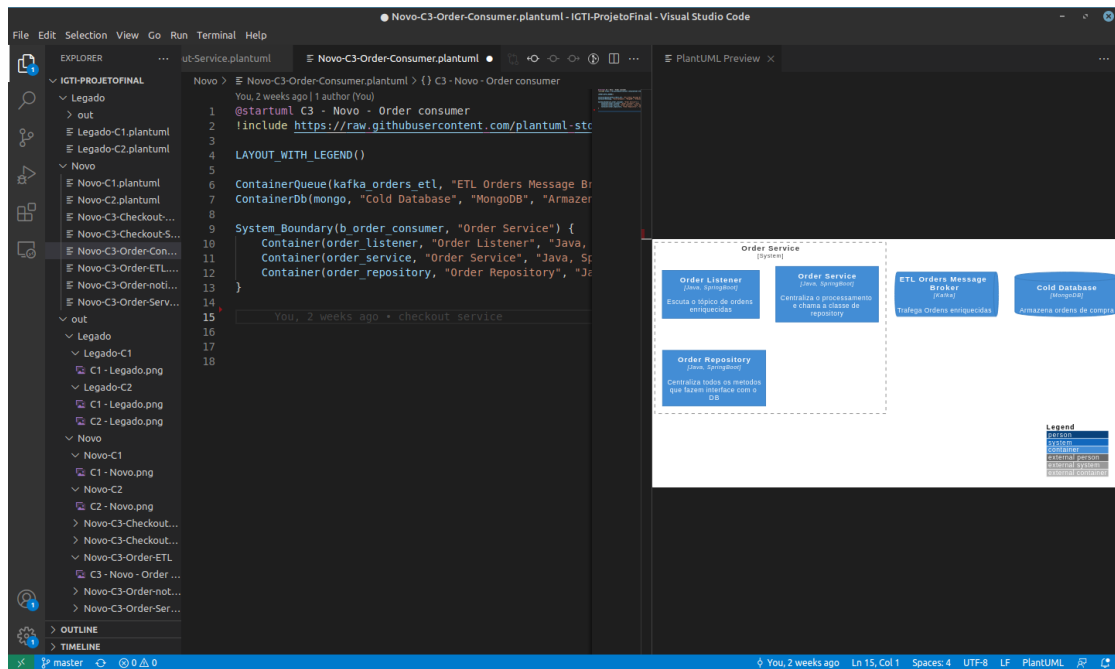
Código disponível em: [GitHub: https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C3-Checkout-Service.plantuml](https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C3-Checkout-Service.plantuml)



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

Desenvolvido em Visual Studio Code com o plugin do plantUML

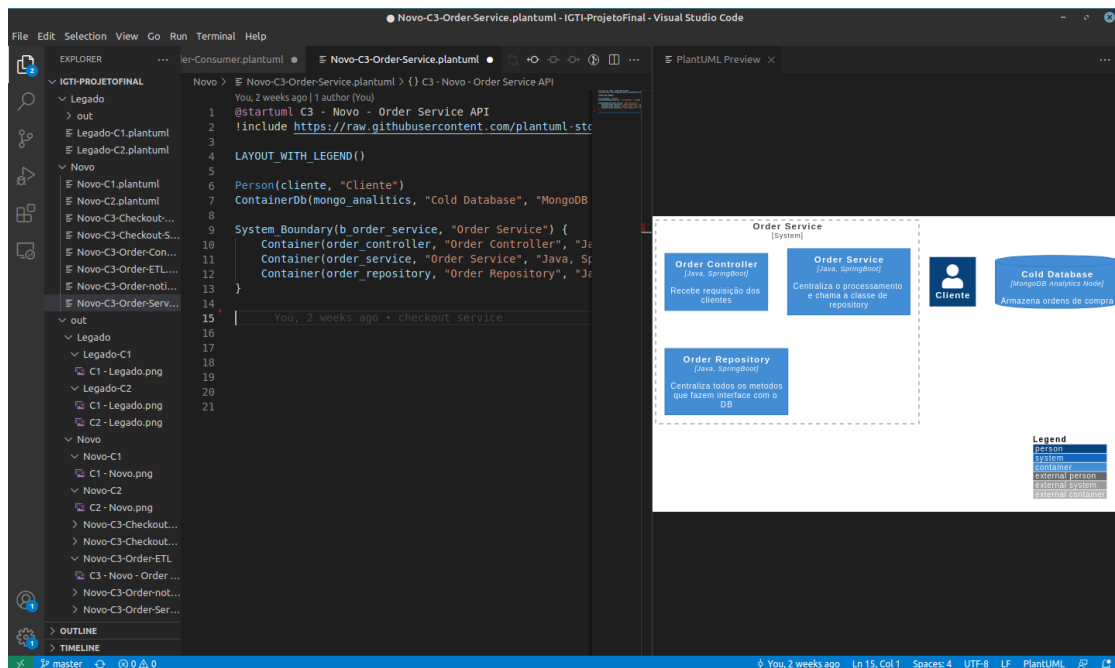
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C3-Order-ETL.plantuml>



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

Desenvolvido em Visual Studio Code com o plugin do plantUML

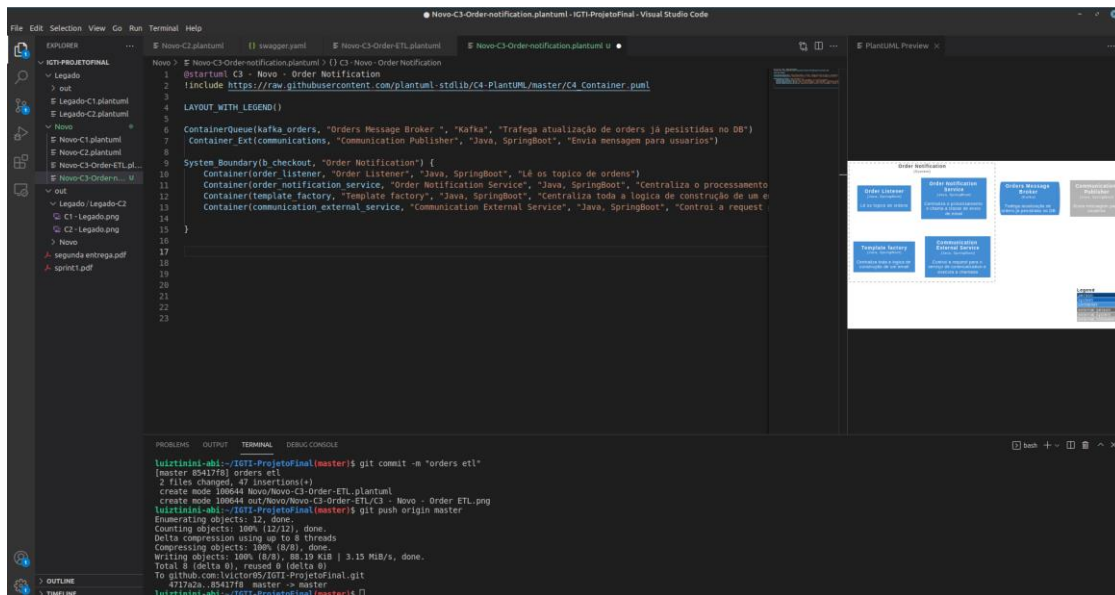
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C3-Order-Consumer.plantuml>



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

Desenvolvido em Visual Studio Code com o plugin do plantUML

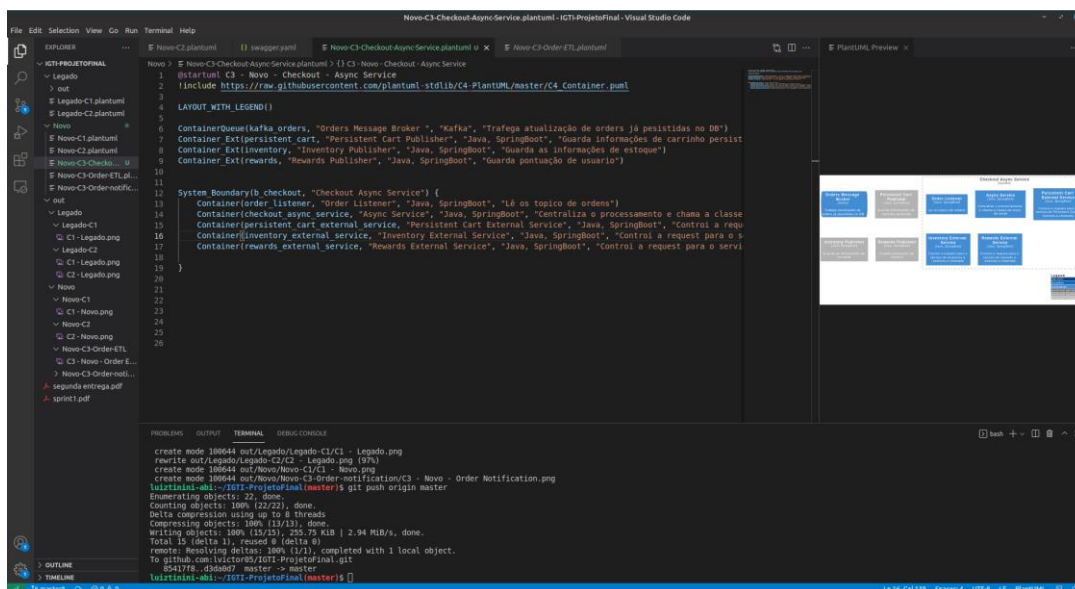
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C3-Order-Service.plantuml>



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

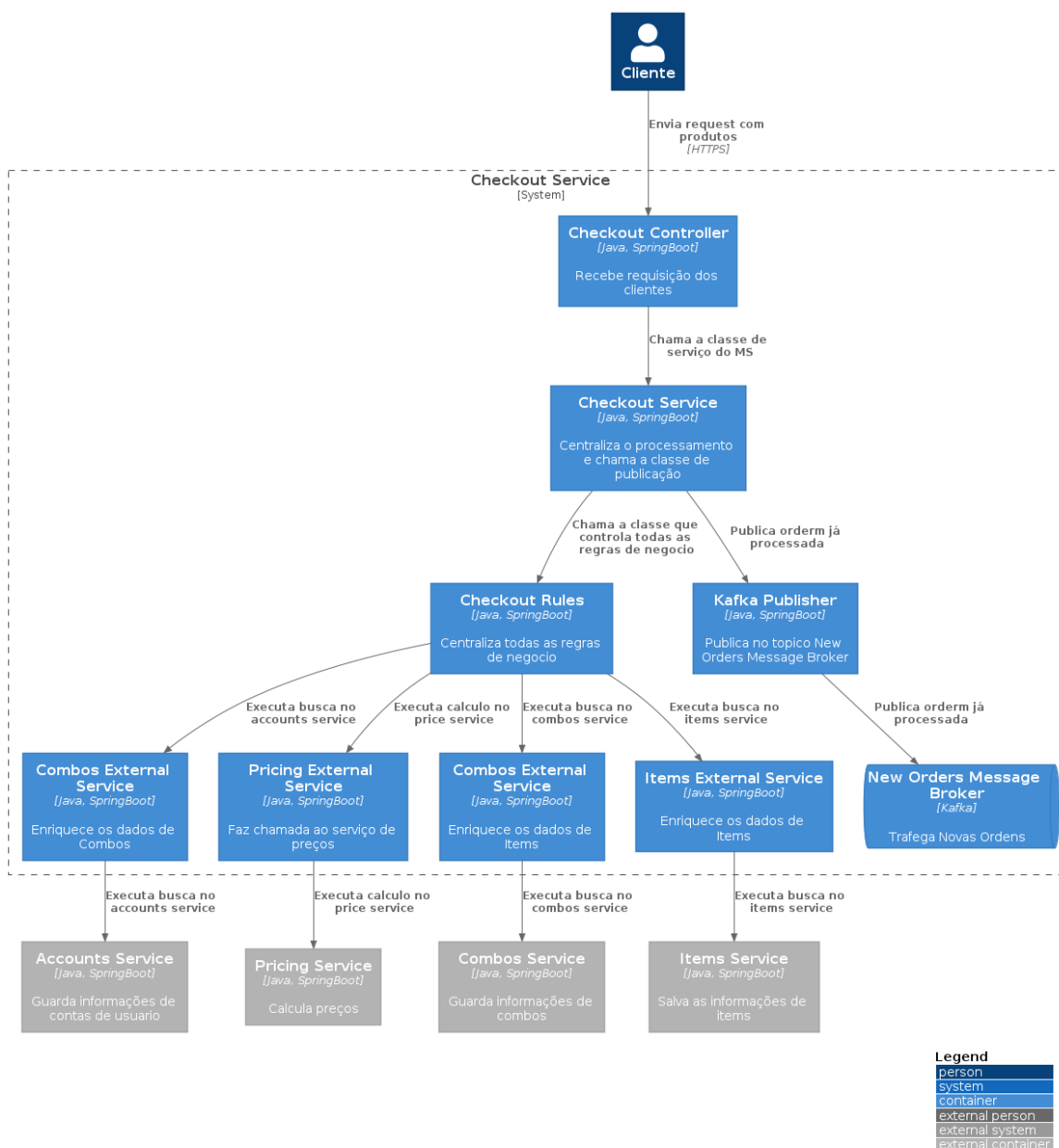
Código disponível em: **GitHub:** <https://github.com/lvictor05/IGTI-ProjetoFinal/blob/master/Novo/Novo-C3-Checkout-Async-Service.plantuml>

- Evidência dos resultados:

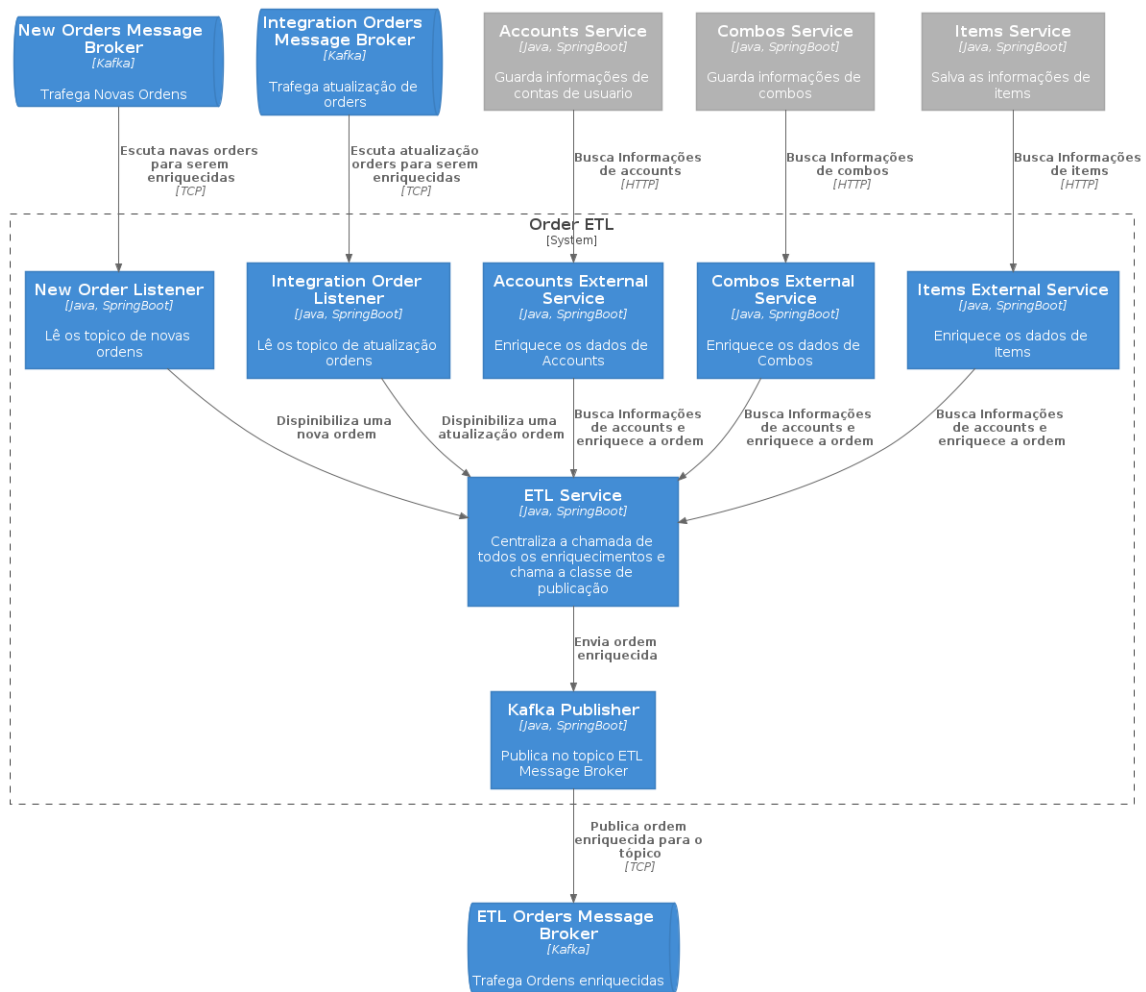
### Diagramas:

A arquitetura interna dos microsserviços abaixo 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 Checkout Service - Level 3: Component diagram



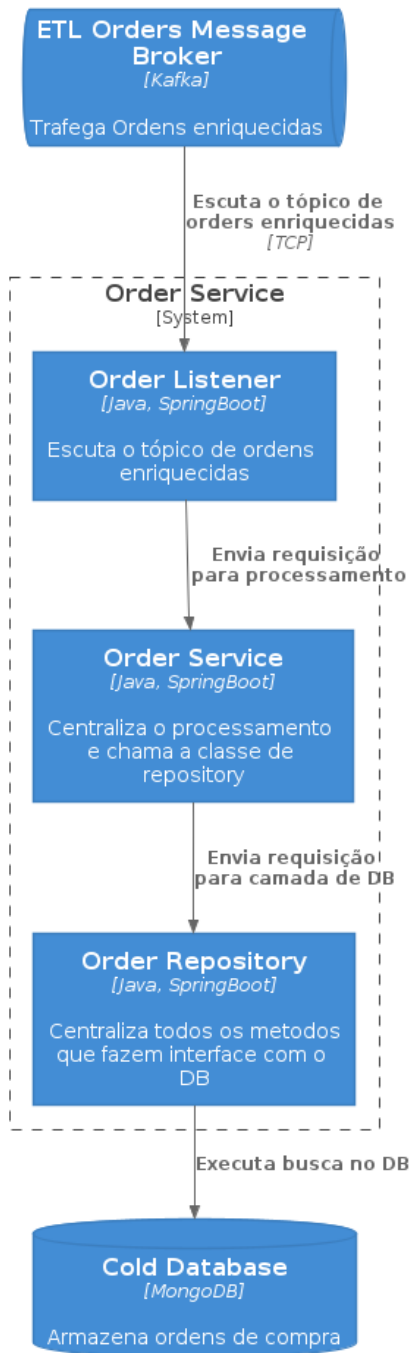
### C4 Model Orders ETL - Level 3: Component diagram



**Legend**

person
system
container
external person
external system
external container

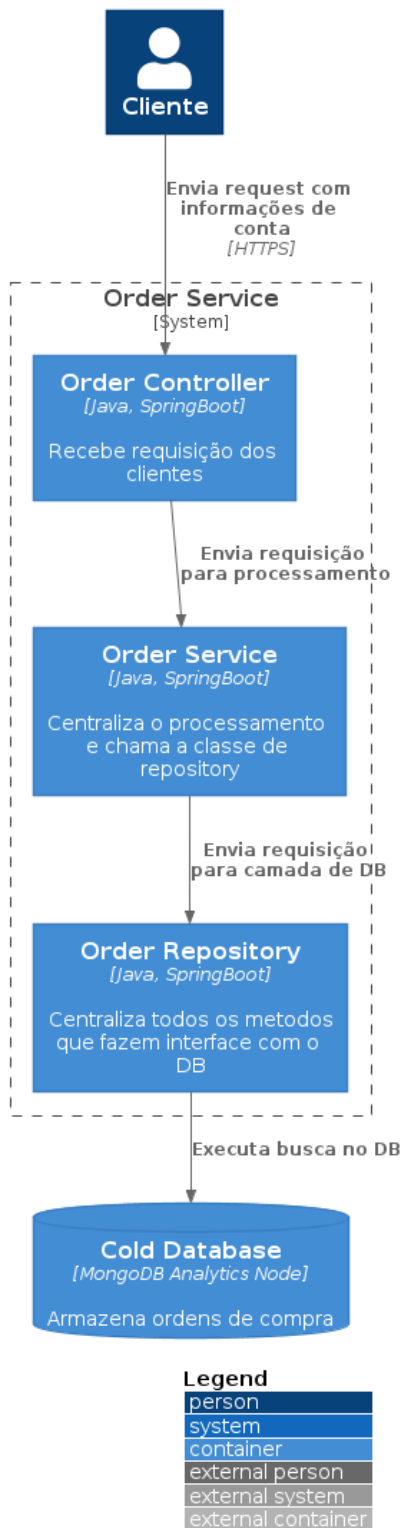
## C4 Model Order Consumer - Level 3: Component



**Legend**

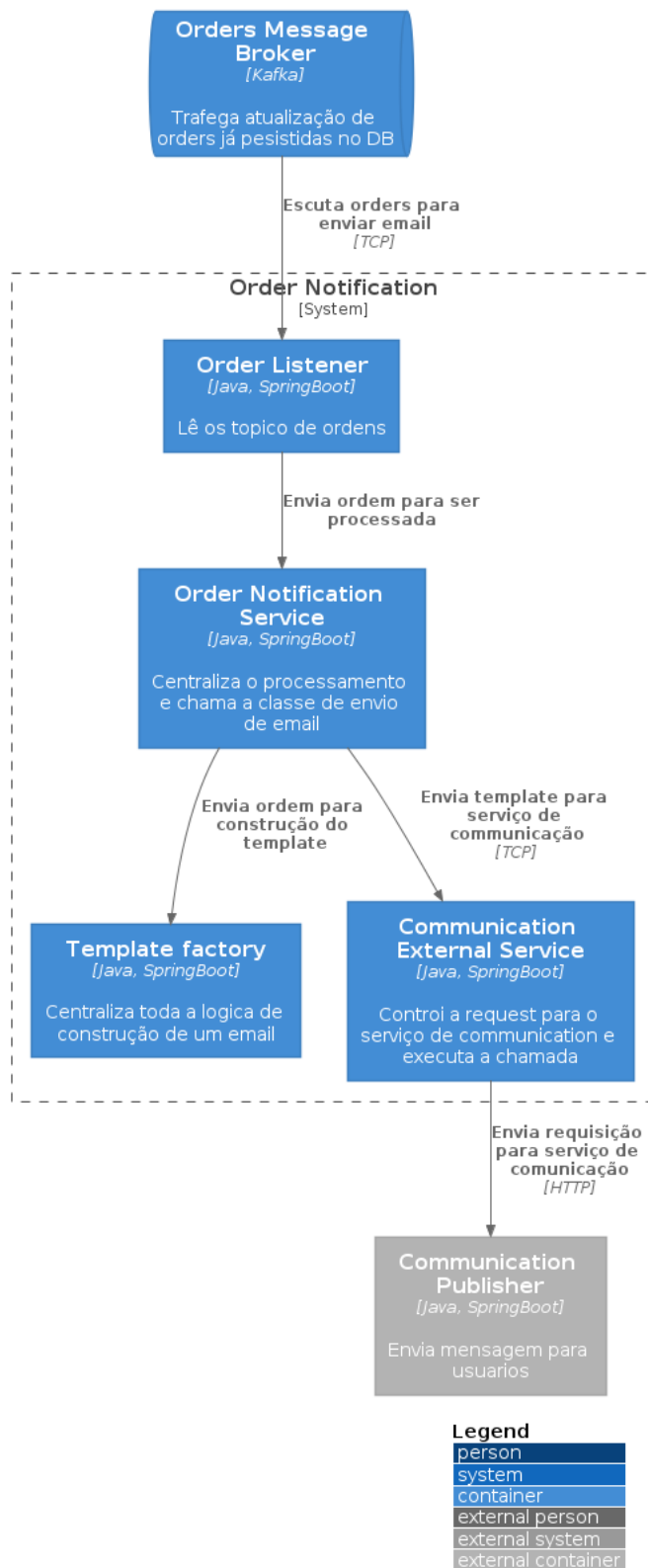
person
system
container
external person
external system
external container

#### C4 Model Order Service - Level 3: Component

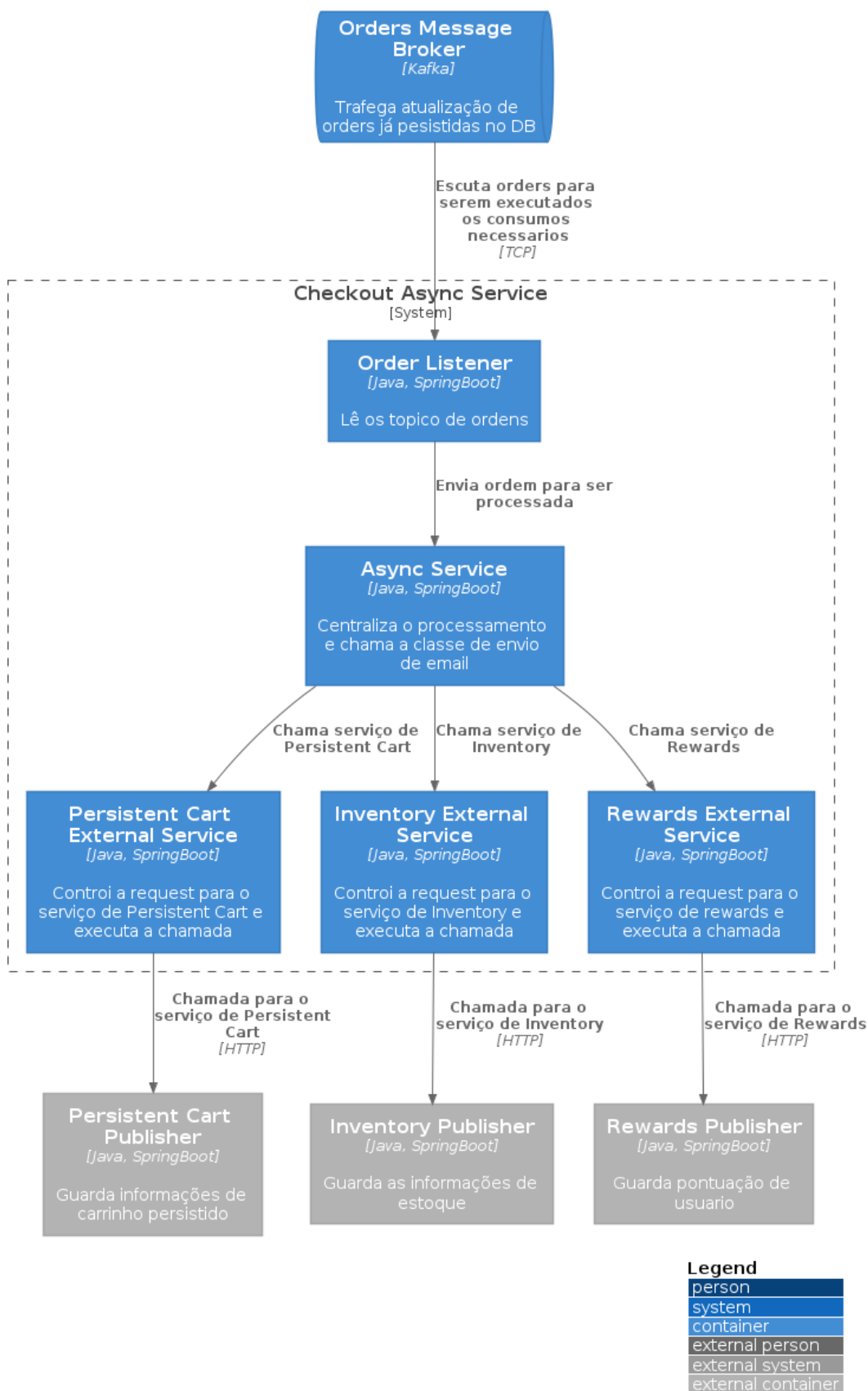


#### C4 Model Order Notification - Level 3: Component





#### C4 Model Checkout Async Service - Level 3: Component



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 topico 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

### 2.3.2 Lições aprendidas

Acredito que ao final dessa sprint pude concluir o muito bem esse projeto. O C4 Model se mostrou realmente muito simples de ser implementado. Com a pratica pude entender melhor o framework e utiliza-lo da forma correta. Preferi manter as cores e layouts padrão do framework, apesar de ser possível modifica-lo.

Acredito que a única camada que seria bom chegar ao nível 4 do framework seria o Checkout Service - Rules. Essa camada é responsável por manter cada uma das regras de negócio do serviço de checkout e por esse motivo é muito importante ser bem detalhada.

## 3. Considerações Finais

### 3.1 Resultados

Considero esse projeto concluído com sucesso. Pude abordar todos os pontos que havia planejado anteriormente. Restringi o escopo focando apenas na refatoração do sistema de checkout para justamente para que eu pudesse abordar com um pouco mais de detalhes todos os passos do processo.

A maior lição aprendida durante esse projeto foram todos os benefícios que um projeto bem organizado e dividido pode trazer. Pude dividir muito bem cada parte do trabalho em uma sprint diferente. A Sprint 1 foi mais focada em apresentar a arquitetura atual e suas falhas. Na Sprint 2 comecei a desenhar a nova arquitetura, utilizando o primeiro e segundo nível do C4 model. Por fim na Sprint 3 fui mais afundo na nova arquitetura e desenvolvi o terceiro nível do C4 model para a arquitetura proposta.

Não houveram grandes dificuldades durante todo o processo de desenvolvimento desse projeto. O mais trabalhoso foi aprender um padrão de documentação que apenas de muito conhecido no mercado, era uma novidade para mim. O C4 model tem uma documentação muito bem descrita e com exemplos, portanto com um pouco de esforço é possível entender todo o framework.

Esse trabalho proporcionou uma serie de ganhos para minha carreira profissional. Durante sua construção pude utilizar conceitos de design thinking, design patterns, SOLID, C4 Model, entre outros. Com certeza usei muitos conceitos abordados nos cursos de Arquitetura de Software e Arquitetura de Solução.

A única desvantagem que posso levantar agora no momento é o tempo curto do projeto, gostaria de ao menos ter tido a oportunidade de documentar o início da refatoração na pratica. Depois da implementação certamente vamos conseguir um sistema mais organizado, de simples manutenção, escalável e até mesmo mais barato de se manter.

### 3.2 Contribuições

O projeto propunha uma completa reformulação do sistema de checkout de um sistema já existente. Era necessário criar o novo, mantendo o existente ainda funcionando. Com o passar do tempo os microsserviços legados foram ganhando cada dia mais funcionalidades e acabaram fugindo do seu papel inicial. Esse projeto em buscou desacoplar componentes, restringir e centralizar responsabilidades para que cada microsserviço fosse capaz de fazer uma pequena parte do todo. Em resumo a nova arquitetura se divide em:

Checkout Service: Validar produtos e submeter uma ordem.

Order ETL: Fazer transformações nas ordens atualizadas.

Order Consumer: Salvar ordens no banco de dados.

Order Service: Prover ordens para os clientes.

Checkout Async Service: Executar pequenos Jobs após uma ordem ser submetida.

Order Notification: Avisar o cliente sobre atualizações de ordens.

Para garantir estabilidade e segurança, toda a comunicação entre os microsserviços foi migrada para o Kafka, uma ferramenta de mensageria mais robusta e até mesmo mais segura que a usada anteriormente. Usamos muito das vantagens da comunicação assíncrona e com frameworks e ferramentas que já tem grandes cases de sucesso em projetos que demandam até mais performance que esse.

Como se trata de um sistema crítico, quanto mais simples a implementação melhor. Já que a grande maioria dos colaboradores possui experiência em Java, inicialmente todos os microsserviços ainda se manterão em Java e só serão reescritos caso haja necessidade. A empresa e os times sempre estão em constantes mudanças, portanto o padrão MVC será usado na arquitetura interna dos microsserviços.

Considero esse projeto muito inovador sim. Certamente poderia ter sido abordado tecnologias que estão na “hipe” do momento, porém estamos falando de um sistema que suporta vários países e transaciona uma quantia financeira considerável. A inovação nesse caso se deu na forma de usar tecnologias mais consolidadas no mercado, uma inovação considerando riscos e trazendo confiabilidade.

### 3.3 Próximos passos

Todos os diagramas e ideias geradas nesse projeto serão utilizados para de fato começar o processo de refatoração do processo de checkout no qual esse trabalho foi baseado. Porém antes terei que desenvolver mais algumas etapas.

Acredito que terei que ir para o nível quatro do C4 model nos seguintes containers:

C4 Model Checkout Service - Level 3: Component diagram container Checkout Rules e

C4 Model Order Notification - Level 3: Component Container Template Factory.

Depois criar todos os épicos e user stories do Scrum para que essa demanda possa ser devidamente priorizada e iniciada pelo time de desenvolvimento.