**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

**Pós-graduação *Lato Sensu* em Arquitetura de *Software* Distribuído**

Projeto Integrado

Relatório Técnico

Delivery Store

Frederick Fernando Frigieri

São Paulo

08/2022

# Projeto Integrado – Arquitetura de Software Distribuído

**Sumário**

[Projeto Integrado – Arquitetura de Software Distribuído 2](#_Toc110952330)

[1. Introdução 3](#_Toc110952331)

[2. Cronograma do Trabalho 5](#_Toc110952332)

[3. Especificação Arquitetural da solução 6](#_Toc110952333)

[3.1 Restrições Arquiteturais 6](#_Toc110952334)

[3.2 Requisitos Funcionais 6](#_Toc110952335)

[3.3 Requisitos Não-funcionais 7](#_Toc110952336)

[3.4 Mecanismos Arquiteturais 7](#_Toc110952337)

[4. Modelagem Arquitetural 8](#_Toc110952338)

[4.1 Diagrama de Contexto 8](#_Toc110952339)

[5. Link do Vídeo de Apresentação da Etapa 1 9](#_Toc110952340)

[Referências 10](#_Toc110952341)

## Introdução

Com crescimento do mercado online, empresas, comércios de pequeno, médio e grande porte, e até o setor industrial têm se deparado com a demanda de um consumidor cada vez mais exigente: prazo de entrega. Quanto menor o prazo, maior valor agregado ao produto.

Muitas coisas mudaram em pouco tempo, tendo a pandemia como uma das principais aceleradoras de diversas dessas modificações no mercado. Uma delas foi a explosão do *e-commerce* e o setor logístico, que precisou se adaptar, mudando em alguns meses o que levaria anos. Hoje, a logística é novo marketing: com promessas de entregas cada vez mais rápidas. A experiência de comprar sem sair de casa e receber em pouco tempo é surreal e os clientes valorizam isso cada vez mais. Essa é uma realidade que veio para ficar e quem não entrar neste movimento vai acabar perdendo o jogo. (Alonso, M. Diretor Sênior Ifood, Intermodal Digital, Set. 2021)

Diante deste cenário, a Delivery Store, uma *start up*[[1]](#footnote-1) no mercado de logística e *dark store*[[2]](#footnote-2), desenvolveu uma solução completa para atender seus parceiros, desde o armazenamento seguro, com baixo custo, sistema para gestão de estoque, abastecimento e coleta, e também a inteligência de malha logística para que o consumidor final[[3]](#footnote-3) seja atendido no menor prazo possível.

Dentro da necessidade da Delivery Store, o projeto desse sistema busca solucionar, através de uma integração via API, a comunicação entre “loja”, *dark store*, entregador e consumidor.

API é um tipo de software que funciona como um mediador entre duas plataformas diferentes, possibilitando uma comunicação padronizada ainda que os sistemas tenham sido desenvolvidos com linguagens e tecnologias distintas. (Redação Impacta, 2020)

O objetivo é permitir que o cliente final consiga receber seu produto de forma rápida e segura, no menor prazo disponível, com possibilidade de atender entregas para o mesmo dia do *input*[[4]](#footnote-4) do pedido. Para isso, a Delivery Store disponibiliza aos “lojistas” armazéns em localidades centrais, e o sistema localiza o produto mais próximo do cliente, bem como, o entregador que esteja disponível para atender a demanda, no prazo mais curto.

O sistema integrador tem como principal motivação atender o mercado de lojas online (eCommerce), com necessidade de ampliar armazém, garantir segurança e entregar no mesmo dia. Esta API vai permitir que a comunicação aconteça de forma segura e online, garantindo o cumprimento de prazo e satisfação do consumidor final.

Na integração a comunicação vai possibilitar que o sistema de venda do parceiro lojista, o estoque dele disponível na *dark store* e o parceiro de entrega se consultem automaticamente, permitindo que o pedido do consumidor final seja atendimento pelo estoque mais próximo de seu endereço, com a transportadora disponível mais próxima.

O objetivo deste trabalho é apresentar a descrição do projeto arquitetural de uma aplicação WEB.

Os objetivos específicos propostos são:

* Realizar um estudo de mercado sobre a área de negócio da aplicação proposta;
* Descrever os requisitos da aplicação de forma resumida, clara e objetiva;
* Apresentar uma solução para integrar lojas, armazéns, transportadores e cliente final com comunicação fluida
* Garantir para seu cliente a entrega do pedido para o consumidor final em até 1 dia útil após a compra

## Cronograma do Trabalho

A seguir é apresentado o cronograma proposto para as etapas deste trabalho.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** |
| **De** | **Até** |
| 01/05/2022 | 08/08/2022 | 1. Cronograma | Definição do cronograma |
| 01/05/2022 | 08/08/2022 | 1. Introdução / Contextualização | Elaboração da introdução e contextualização |
| 01/05/2022 | 08/08/2022 | 1. Definição dos requisitos arquiteturais | Lista dos requisitos arquiteturais identificados |
| 01/05/2022 | 08/08/2022 | 1. Definição dos requisitos funcionais | Listar dos requisitos funcionais identificados |
| 01/05/2022 | 08/08/2022 | 1. Definição dos requisitos não funcionais | Listar dos requisitos não funcionais levantados |
| 01/05/2022 | 08/08/2022 | 1. Definição dos mecanismos arquiteturais | Listar os mecanismos arquiteturas identificados |
| 01/08/2022 | 08/08/2022 | 1. Construção dos diagramas de contextos – Modelo C4 | Diagrama de contexto criado |
| 05/08/2022 | 08/08/2022 | 1. Revisão do documento | Documento Revisado |
| 08/08/2022 | 08/08/2022 | 1. Construção do vídeo de apresentação I | Vídeo criado |
| 15/08/2022 | 15/10/2022 | 1. Construção do diagrama de Contêiner – Modelo C4 | Diagrama criado e vinculado ao documento do projeto |
| 15/08/2022 | 15/10/2022 | 1. Construir diagrama de componentes | Diagrama criado e vinculado ao documento |
| 15/08/2022 | 15/10/2022 | 1. Construção do Wireframe | Protótipos de telas |
| 15/08/2022 | 15/10/2022 | 1. Código da aplicação | Implementação de três requisitos |
| 15/10/2022 | 15/12/2022 | 1. Análise das abordagens arquiteturais | Análise documentado na seção solicitada |
| 15/10/2022 | 15/12/2022 | 1. Cenários | Cenários documentados na seção solicitada |
| 15/10/2022 | 15/12/2022 | 1. Evidências da avaliação | Evidências documentadas na seção solicitada |
| 15/10/2022 | 15/12/2022 | 1. Resultados obtidos | Resultados documentados na seção solicitada |
| 15/10/2022 | 15/12/2022 | 1. Avaliação críticas dos resultados | Críticas documentadas na seção solicitada |
| 15/10/2022 | 15/12/2022 | 1. Conclusão | Conclusão documentada na seção solicitada |
| 15/10/2022 | 15/12/2022 | 1. Vídeo da apresentação final | Vídeo da etapa 3 criado |

## Especificação Arquitetural da solução

Esta seção apresenta a especificação básica da arquitetura da solução a ser desenvolvida, incluindo diagramas, restrições e requisitos definidos pelo autor, tal que permitem visualizar a macroarquitetura da solução.

## Restrições Arquiteturais

R1: Deve utilizar a linguagem C# para desenvolvimento do backend

R2: Deve utilizar a linguagem Angular para desenvolvimento do Frontend

R3: Deve dividir o backend em três microservices (Pedido, Estoque, Transportadora)

R4: Cada microservice deve ter seu próprio repositório

R5: Comunicação entre microservice deve acontecer através de fila de mensagens

## Requisitos Funcionais

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Descrição Resumida** | **Dificuldade (B/M/A) \*** | **Prioridade**  **(B/M/A) \*** |
| RF01 | O sistema deve permitir acesso através de login e senha | B | A |
| RF02 | O sistema deve integrar com a plataforma Bling para importar os pedidos e os produtos dos seus clientes | A | M |
| RF03 | O sistema deve permitir cadastrar parceiros | M | A |
| RF04 | O sistema deve permitir cadastrar armazéns e posições | M | A |
| RF05 | O sistema deve disponibilizar um relatório do tipo Timeline para acompanhar os pedidos | M | A |
| RF06 | O sistema deve disponibilizar detalhes do pedido | B | A |
| RF07 | O sistema deve disponibilizar relatório analítico exibir dos pedidos | B | A |
| RF08 | O sistema deve permitir recebimento de produtos dos seus parceiros | M | A |
| RF09 | O sistema deve permitir impressão de etiqueta para colar no produto depois de embalado | M |  |
| RF10 | O sistema deve permitir impressão da nota fiscal que acompanhará o pedido | M | A |
| RF11 | O sistema deve gerenciar o estoque | A | A |
| RF12 | Sistema deve disponibilizar um arquivo no formato PDF com a rota do dia | B | A |
| RF13 | O sistema deve ter no mínimo as fases de importação, estoque, roteirização e despachado | M | A |
| RF14 | Sistema deve roteirizar um número x de pedidos todo dia as 7 horas da manhã | M | A |
| RF15 | Sistema deve permitir que a transportadora envie uma mensagem com o status final da entrega | B | A |
| RF16 | Sistema deve permitir alteração da senha do usuário | B | M |
| RF17 | Sistema deve permitir cadastro simples utilizando o e-mail | B | A |
| RF18 | Sistema deve encontrar o armazém com estoque mais próximo do endereço de destino e vincular ao pedido | A | A |
| RF19 | Sistema deve disponibilizar um relatório com os produtos por parceiro informando a quantidade de produto disponível | B | A |
| RF20 | Sistema deve disponibilizar um relatório do estoque por posição | B | M |
| RF21 | Sistema deve disponibilizar uma forma de encerrar a sessão | B | B |
| RF22 | Sistema deve ter a inteligência de movimentar o estoque quando o produto for recebido e quando ele sair para entrega | A | A |

\*B=Baixa, M=Média, A=Alta.

## Requisitos Não-funcionais

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrição** | **Prioridade**  **B/M/A** |
| RNF01 | A integração com o parceiro para buscar os pedidos deve respeitar o prazo de funcionamento do armazém sendo as 07h até 22h | A |
| RNF02 | O sistema deve armazenar as requisições como log | A |
| RFN03 | Os microservices devem se comunicar apenas por mensagens | A |
| RFN04 | O sistema deve ser responsivo | A |
| RFN05 | O sistema deve gerar alerta amigável ao usuário em caso de falha | M |
| RFN06 | O sistema deve permitir escalonar os microservices | M |

## Mecanismos Arquiteturais

|  |  |  |
| --- | --- | --- |
| **Análise** | ***Design*** | **Implementação** |
| Persistência | ORM | Entity Framework / MS SQL SERVER |
| Frontend | Single Page Application | Angular |
| Backend | WebApi Rest | Aspnet Core / C# |
| Comunicação Microservice | Mensageria | RabbitMq / SQS / Service Bus |
| Log do sistema | Requisições, Respostas e Falhas | Elastic Search / Serilog |
| Consulta de dados | ORM | Dapper / MS SQL SERVER |
| Deploy | CI / CD | Azure DevOps / AWS |
| Tratamento de exceções | Middleware | Aspnet Core |
| Gerenciador de Jobs | Framework | HangFire |
| Autenticação | Token | JWT |

## Modelagem Arquitetural

Esta seção apresenta a modelagem arquitetural da solução proposta, de forma a permitir seu completo entendimento visando à implementação da Prova de Conceito (PoC) da plataforma 4Pet na seção 5.

Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para documentação de arquitetura de software. Mais informações a respeito podem ser encontradas aqui: <https://c4model.com/> e aqui: <https://www.infoq.com/br/articles/C4-architecture-model/>. Dos quatro níveis que compõem o modelo C4 três serão apresentados aqui e somente o Código será apresentado na próxima seção (5).

## 4.1 Diagrama de Contexto

A figura 1 mostra a especificação o diagrama geral da solução proposta, com todos seus principais sistemas e pessoas envolvidas nos processos.

***Interface gráfica do usuário

Descrição gerada automaticamente com confiança média***

Figura 1 – Diagrama de Contexto

## 4.2 Diagrama de Container

Diagrama

Descrição gerada automaticamente  
Figura 2 – Diagrama de container

A Figura 2 tem como objetivo dar mais detalhes dos sistemas apresentados no diagrama de contexto. Aqui são apresentados os containers que compõem o sistema da DeliveryStore e como eles se relacionam.

Existem 3 tipos de perfis que acessam o sistema.

O consumidor final acessará o sistema através de uma url enviada por e-mail com objetivo de acompanhar o status do seu pedido

O parceiro/cliente acessará o sistema depois de realizar o cadastro no próprio site. Este acesso é feito através de login e senha informados no momento do cadastro. O parceiro tem como objetivo acompanhar os pedidos dos seus consumidores assim como gerenciar seus produtos e estoques.

O operador logístico, acessa o sistema com login e senha criado por um login master. O operador que tem como objetivo gerenciar todo o fluxo de entrada de pedido e produtos até o envio do pedido a transportadora.

## 4.3 Diagrama de Componentes

Diagrama

Descrição gerada automaticamente

Figura 3 – Diagrama de Componente Geral

A figura 3 mostra o diagrama de componentes da solução, seguindo o padrão UML. Através dele é mostrada com mais detalhes a base tecnológica da aplicação.

Os componentes da solução:

**Angular**: É uma aplicação Web desenvolvida em angular, utilizada como interface pelos usuários para acessar o sistema.

**Autenticação**: É uma api desenvolvida em .Net Core 3 que valida os dados de acesso do usuário e retorna para a aplicação um token de acesso.

**Gateway**: É uma api de gateway desenvolvida em .Net Core 3 utilizada para centralizar as chamadas da aplicação Web com os Microservice.

**MS-OMS**: Microservice responsável pelos dados de Pedido e Parceiro.

**MS-WMS**: Microservice responsável pelos dados de Produtos, Estoques e Armazém.

**MS-TMS**: Microservice responsável pelos dados da Transportadora e Entrega de Pedido.

Os Microservices foram desenvolvidos em .Net Core 3. Eles são compostos pelos seguintes componentes:

**Api**: Componente que tem como objetivo receber e retornar informações para o componente Gateway.

**Application**: Componente desenvolvido em .Net Core responsável por transformar os dados recebidos pela api em dados do negócio (domínio) e armazenar em um repositório.

**Domain**: Componente responsável pela regra de negócio do sistema DeliveryStore. Desenvolvido em .Net Core 3.

**Infrastructure**: Componente responsável pela implementação de serviços de terceiros, como serviços para acesso a repositório, mensagerias, e-mail, entre outros. Desenvolvido em .Net Core 3.

**SharedMessages**: Componente responsável pelos objetos que são armazenados na mensageria. Desenvolvido em .Net Core 3.

**Worker**: Componente responsável por consumir mensagens enviadas para a mensageria. Desenvolvido em .Net Core 3

**HangFire**: Componente web responsável por executar e gerenciar tarefas no sistema. Desenvolvido em .Net Core junto do pacote HangFire.

**Database**: Componente responsável pelo armazenamento de dados, utiliza o MS Sql Server junto do componente EF da Microsoft para armazenamento e o componente Dapper para recuperar as informações.

**ElasticSearch**: Componente utilizado para armazenar todas as requisições feita na Api. Utiliza o componente Serilog para fazer o armazenamento.

## Prova de Conceito (PoC)

Essa sessão tem como objetivo detalhar a prova de conceito para que o objetivo deste trabalho fosse atendido.

5.1. Integração entre componentes

O sistema proposto tem como objetivo receber pedidos e produtos dos seus parceiros de um sistema externo conhecido no mercado como Bling.

Como envolveria um custo para manter esse sistema externo e como o objetivo do curso é a arquitetura por trás e não o sistema externo em si, para contornar esse imprevisto foi desenvolvido endpoints e interfaces para entrada de pedido e produto.

Para apresentação da PoC foi desenvolvido uma wireframe navegável através da ferramenta Figma que está disponível no link abaixo

<https://www.figma.com/proto/kSrKocDyxdBKgQSZ4ghyut/TCC-PUC?node-id=1%3A3&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=1%3A3&show-proto-sidebar=1>

**5.1.1. RF03 - O sistema deve permitir cadastrar parceiros**

Para atender este requisito foi implementado um botão para que o parceiro realize o cadastro na tela de login. Basta acessar o link do wireframe e na primeira tela clicar no botão “Criar Nova Conta”

**5.1.2. RF01 - O sistema deve permitir acesso através de login e senha**

Para atender a este requisito foi criado um componente com duas entradas de dados, uma para o e-mail e outra para a senha e um botão “Entrar”.

**5.1.3. RF08 - O sistema deve permitir recebimento de produtos dos seus parceiros**

Para atender a este requisito foi criado um componente para a entrada dos dados do produto quando o parceiro estiver logado. Para visualizar esse componente no wireframe na tela de login clique no botão Entrar e depois no menu esquerdo clique no link Produtos, feito isso carregará a tela com a lista de produto e mais abaixo o botão “Novo Produto”, clique e veja o componente.

**5.1.4. RF05- O sistema deve disponibilizar um relatório do tipo Timeline para acompanhar os pedidos**

Para atender este requisito foi criado um componente para exibir os pedidos do parceiro na visão fase x horas. Para acessar esse componente no wireframe na tela de login clique em Entrar e depois no menu ao lado esquerdo dentro do menu OMS clique no link Timeline.

## Código da Aplicação

Nesta seção irei abordar um pouco sobre o código da aplicação utilizando diagramas dos principais fluxo e também link para o repositório onde contém todo código da aplicação.

Diagrama

Descrição gerada automaticamente

Figura 4 – Cadastro de Parceiro

Diagrama

Descrição gerada automaticamente

Figura 5 – Cadastro de Pedido

Diagrama, Esquemático

Descrição gerada automaticamente

Figura 6 – Cadastro de Produto

Diagrama

Descrição gerada automaticamente

Figura 7 – Cadastro de Estoque

As figuras 4, 5, 6 e 7 apresentam os componentes e seus relacionamentos dos principais fluxos do sistema.

Observe que os fluxos são bem parecidos, o dado entra por um endpoint no componente de API e passa pelo componente Application através de Command e CommandHandler, um objeto de negócio é criado passando pelo componente de Domain e sendo persistido para um repositório implementado no componente Infrastructure.

Nas figuras 4, 5 e 6 além da persistência ao repositório, o componente Domain dispara um evento indicando que algo aconteceu naquele objeto, esse evento é interceptado por um Notification no componente de Application e persistido para um repositório dentro de uma tabela chamada OutboxMessages com o objetivo postar essa mensagem para outros sistemas ou Microservices através de mensageria.

Esse processo garante que a mensagem só será enviada quando a alteração for salva no repositório com sucesso.

Para publicar essa mensagem na mensageria é utilizado um componente chamado HangFire que será exibido na Figura 8.

Diagrama

Descrição gerada automaticamente

Figura 8 – HangFire publicando Mensagem

Note na figura 8 que o componente do HangFire irá buscar no banco através da interface ISqlConnectionFactory mensagens representadas por Commands, esses Commands são interceptados pelos seus Handlers que tem como objetivo criar e publicar a mensagem na mensageria.

Diagrama

Descrição gerada automaticamente

Figura 9 – Consumo de mensagens através do Worker

Na figura 9 é apresentado os componentes e seus relacionamentos responsáveis por consumir uma mensagem da mensageria.

O componente Infrastructure tem a implementação e configuração com a mensageria e o componente Application é responsável por consumir a mensagem e transformá-la em um Command e persistir para o repositório na tabela InternalCommands.

Note que para consumir a mensagem é utilizado a interface IConsumer do componente do tipo package chamado MassTransit e para armazenar na InternalCommands é utilizada a interface ICommandScheduler.

Diagrama

Descrição gerada automaticamente

Figura 10 – HangFire processando Commands da InternalCommands

A figura 10 apresenta os componentes e seus relacionamentos responsáveis por processar um command. Ele é bem parecido com o processamento do OutboxMessages apresentado na figura 8.

A diferença entre eles é conceitual, enquanto o OutboxMessages é utilizado para enviar mensagens para a mensageria, o processo do InternalCommands tem como responsabilidade consumir as mensagens da mensageria e processá-las internamente dentro de um sistema ou microservice.

Nesta seção abordei os principais componentes e seus relacionamentos com base nas RFs escolhidas.

Todo o conteúdo está disponível no link do repositório abaixo.

Repositório: <https://github.com/frederickfrigieri/puc-projeto-integrado>

## Link do Vídeo de Apresentação da Etapa 1

<https://vimeo.com/737784518>

## Referências

Intermodal Digital (<https://digital.intermodal.com.br/especialistas/demanda-por-entregas-cada-vez-mais-rapidas-e-uma-realidade-que-veio-para-ficar>)

Sebrae

<https://www.sebrae.com.br/sites/PortalSebrae/artigos/o-que-e-uma-startup,6979b2a178c83410VgnVCM1000003b74010aRCRD>

Escola de Ecommerce

<https://www.escoladeecommerce.com/artigos/dark-store/amp/?gclid=CjwKCAjw6MKXBhA5EiwANWLODHP0IynYZ37vrCXPtTkIngs8CM4LxwD-uzZmlcitEe9s3o5tT2ejsRoC998QAvD_BwE>

Impacta Blog

<https://www.impacta.com.br/blog/o-que-integracao-via-api-como-funciona-pratica/>

Significados.com.br

<https://www.significados.com.br/input/>

1. Uma empresa que nasce em torno de uma ideia diferente, escalável e em condições de extrema incerteza. (Sebrae, 2022) [↑](#footnote-ref-1)
2. A Dark Store funciona como um centro de distribuição e de logística para empresas, em sua maioria grandes varejistas, atuarem com um ponto físico. (Escola de Ecommerce) [↑](#footnote-ref-2)
3. O Consumidor Final, pode ser uma Pessoa Física ou Pessoa Jurídica, apresentada como Destinatária da NF-e ou NFC-e, que está adquirindo um produto / mercadoria para seu uso, normalmente, produtos em seu estado final de comercialização, como um celular, um notebook, uma mesa, cadeira, televisão etc. [↑](#footnote-ref-3)
4. INPUT é uma expressão da língua inglesa que significa entrada. O termo é muito utilizado na área da Tecnologia da Informação (TI), como também em diversas outras áreas da atividade humana, como eletricidade, hidráulica etc. (Significados.com.br) [↑](#footnote-ref-4)