

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**

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

**Geydson Batista dos Santos**

**SISTEMA PARA VENDAS DE SUPLEMENTOS BASEADOS EM  
MICROSSERVIÇOS**

São Paulo  
2023

**Geydson Batista dos Santos**

**SISTEMA PARA VENDAS DE SUPLEMENTOS BASEADOS EM  
MICROSSERVIÇOS**

Trabalho de Conclusão de Curso de Especialização em  
Arquitetura de Software Distribuído como requisito parcial  
à obtenção do título de especialista.

Orientador(a): Samuel Martins da Silva

São Paulo  
2023

## **RESUMO**

Este projeto apresenta uma proposta arquitetural orientada a micros serviços para o segmento de suplementos. A empresa Integralmédica possui a necessidade em atualizar sua aplicação legada, visando o crescimento da empresa e pela crescente demanda nas vendas de suplementos nutricionais. Este novo sistema não deverá impactar nas atividades atuais da empresa e deve manter uma integração com as aplicações já implantadas e em funcionamento na mesma. Com base no levantamento de requisitos a arquitetura será modelada para que possa ser criado esse projeto arquitetural, em que seja capaz de melhorar a usabilidade e desempenho nas vendas.

**Palavras-chave:** arquitetura de software, projeto de software, microserviços.

## Sumário

1. <i>Objetivos do trabalho</i> .....	6
2. <i>Especificação Arquitetural da solução</i> .....	7
2.1 Restrições Arquiteturais .....	7
2.2 Requisitos Funcionais .....	7
2.3 Requisitos Não-funcionais .....	8
2.4 Mecanismos Arquiteturais.....	9
3. Modelagem Arquitetural .....	10
3.1 Diagrama de Contexto.....	10
3.2 Diagrama de Container.....	11
3.3 Diagrama de Componentes .....	12
4. Avaliação da Arquitetura ATM.....	15
4.1 Análise de abordagens arquiteturais.....	15
4.1.2 Segurança .....	15
4.1.3 Usabilidade.....	15
4.1.4 Manutenibilidade.....	16
4.1.5 Disponibilidade .....	16
4.1.6 Interoperabilidade.....	16
4.1.7 Desempenho .....	16
4.2 Cenários.....	17
4.2.1 Cenário 1 .....	17
4.2.2 Cenário 2 .....	17
4.2.3 Cenário 3 .....	17
4.2.4 Cenário 4 .....	17
4.2.5 Cenário 5 .....	17
4.2.6 Cenário 6 .....	18
4.2.7 Cenário 7 .....	18
4.2.8 Cenário 8 .....	18
4.3 Evidências da Avaliação de Cenários .....	18
4.3.1 Cenário 1 .....	18
4.3.2 Evidência do Cenário 1 .....	20
4.4 Cenário 2 .....	22
4.4.1 Evidência do Cenário 2 .....	22
4.5 Cenário 3 .....	24

4.5.1 Evidência do Cenário 3 .....	25
4.6 Cenário 4 .....	26
4.6.1 Evidência do Cenário 4 .....	27
4.7 Cenário 5 .....	27
4.7.1 Evidência do Cenário 5 .....	28
4.8 Cenário 6 .....	29
4.8.1 Evidência Cenário 6 .....	31
4.9 Cenário 7 .....	32
4.9.1 Evidência de Cenário 7 .....	33
5. Cenário 8 .....	35
5.1 Evidências de Cenário 8.....	35
6. Avaliação Crítica dos Resultados.....	37
7. Conclusão .....	38
<i>REFERÊNCIAS</i> .....	39
<i>APÊNDICES</i> .....	40

## 1. Objetivos do trabalho

A empresa Integralmédica é uma empresa de grande porte em que atua no segmento de suplementos alimentares em todo o território nacional, realizando a venda dos seus produtos para diversas empresas através dos seus próprios vendedores. Com a alta demanda pelos produtos de suplementos alimentares, houve um grande crescimento na disputa neste mercado.

O objetivo geral deste projeto é apresentar uma proposta arquitetural baseada em microsseviços para a Integralmédica, com o intuito de modernizar sua plataforma buscando mais segurança e robustez, sendo assim proporcionando uma melhor experiência para os usuários.

Os objetivos específicos propostos são:

- ☐ Criar um sistema modular e multiplataforma, em que seja possível realizar o acesso via web em computadores e em dispositivos móveis.
- ☐ Implementar módulo de autenticação de usuários com níveis de acesso, em que cada tipo de acesso poderá acessar ou não os módulos, para que haja um controle tanto de acesso a plataforma ou dos dados contidos na plataforma.
- ☐ Possibilitar a gestão e venda dos suplementos alimentares, através dos controles de estoque, inserção de pedidos, acompanhamento dos pedidos, cadastro de clientes e acompanhamento das vendas.

## 2. Especificação Arquitetural da solução

Esta seção apresenta a especificação básica da solução arquitetural destinada a aplicação “Venda de Suplementos Alimentares”.

### 2.1 Restrições Arquiteturais

- ☐ R1: O software deve ser desenvolvido em NodeJS.
- ☐ R2: Será implementado o conceito de Application Programming Interfaces (API) na criação dos serviços.
- ☐ R3: O sistema deve ser desenvolvido em módulos para facilitar a implantação.
- ☐ R4: O sistema deve ser hospedado on-premise.
- ☐ R5: O sistema deve ter sua arquitetura orientada a serviços.
- ☐ R6: O sistema deve possuir integrações com sistemas externos.
- ☐ R7: O sistema deve possuir seu layout responsivo, para que possa ser utilizado tanto em computadores como dispositivos moveis.
- ☐ R8: O sistema deve ter seu build feito através de integração continua.
- ☐ R9: O sistema deve ter pipelines de teste em sua integração continua.
- ☐ R10: O sistema deve possuir seus serviços acessados apenas por instancias que rodem o api gateway.

### 2.2 Requisitos Funcionais

#### Acesso

- ☐ RF01– O sistema deve permitir o cadastro de novos usuários, com tipos de acesso.
  - Tipo de acesso administrador, gerente e vendedor.
- ☐ RF02 – O sistema deve permitir acesso as funcionalidades da plataforma somente após a realização de login.
- ☐ RF03 – O sistema deve permitir que o usuário realize logoff.

#### Dashboard

- RF04 – O sistema deve permitir visualizar as vendas realizadas, para acompanhamento do próprio usuário vendedor que realizou a venda, quanto no geral para o acesso da gerência.
  - Identificar o total de vendas.
  - Identificar o valor das vendas dia a dia.
  - Identificar o total de novos clientes no mês.

### **Módulo de Clientes**

- RF05 – O sistema deve permitir o cadastro de clientes.
  - Este atrelado a um vendedor.
- RF06 – O sistema deve comunicar-se com a receita para obter os dados do cliente.

### **Módulo de Produtos**

- RF07 – O sistema deve permitir o cadastro de produtos (Privado), devendo permitir e manter as informações.
- RF08 – O sistema deve permitir a atualização dos dados do produto, com integração com ERP.

### **Módulo de Pedidos**

- RF09 – O sistema deve permitir a inserção de pedidos.
- RF10 – O sistema deve permitir a consulta de pedidos.
- RF11 – O sistema deve permitir a exclusão de pedidos.
- RF12 – O sistema deve permitir a integração com ERP, para fins de captação de novos pedidos para faturamento e atualização de status dos pedidos.

## **2.3 Requisitos Não-funcionais**

- RNF01 – Segurança – O sistema não deve permitir acesso sem estar logado.



- ☐ RNF02 – Usabilidade – O sistema deve possuir interface responsiva, adequando o layout ao dispositivo utilizado no acesso.
- ☐ RNF03 – Usabilidade – O sistema deve ser simples de usar.
- ☐ RNF04 – Manutenibilidade – O sistema deve ser simples em sua manutenção.
- ☐ RNF05 – Disponibilidade – O sistema deve se manter ativo mesmo que um dos seus nós fique fora.
- ☐ RNF06 – Interoperabilidade – O sistema deve comunicar-se com sistemas externos através do uso de APIs.
- ☐ RNF07 – Interoperabilidade – O sistema deve permitir que os módulos se comuniquem quando necessário.
- ☐ RNF08 – Desempenho – O sistema deve ser rápido no acesso aos dados.

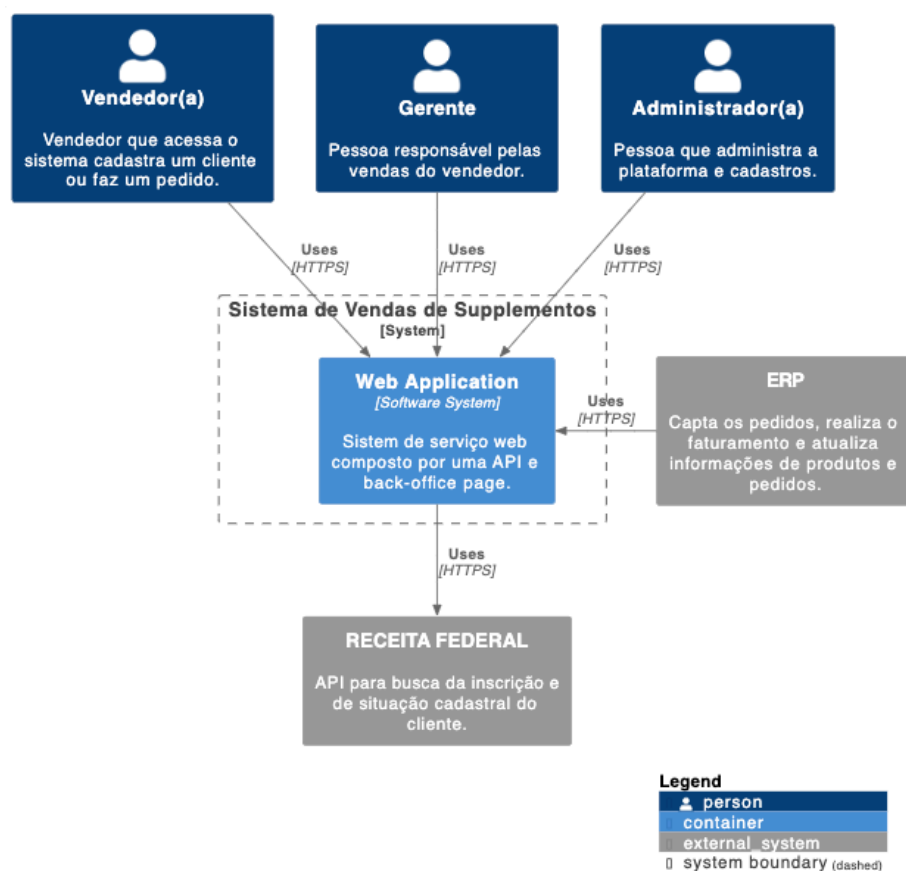
## 2.4 Mecanismos Arquiteturais

Análise	Design	Implementação
Persistência	Banco de dados NoSQL	MongoDB
Contêiner	Serviço de containerização de software	Docker
Front-end	Interface entre o usuário e a aplicação.	React.js
Back-end	API webservice	NodeJS
Comunicação entre módulos e/ou sistemas	Interfaces de comunicação entre os módulos utilizando JSON	APIs REST
Build	Ferramenta de build das imagens Docker	Azure Devops
Segurança	Autenticação e Autorização	JWT
Versionamento	Versionamento de Código	GIT
Cache	Ferramenta de cache	Redis
Alta disponibilidade	Balanceamento de carga das requisições para os serviços	Konga, Grafana, Kubernetes
ApiGateway	Camada intermediária entre o front-end e os serviços da aplicação	Kong

### 3. Modelagem Arquitetural

Nesta seção apresenta-se a modelagem arquitetural da solução proposta, de forma a permitir seu completo entendimento, com base nos diagramas apresentados que permitem entender a arquitetura da aplicação, para possibilitar sua implementação.

#### 3.1 Diagrama de Contexto



**Figura 1 - Visão Geral da Solução.**

A figura 1 mostra especificação do diagrama geral da solução proposta, com os principais módulos e suas interfaces da aplicação de vendas de suplementos. Nesta figura é mostrado todas as pessoas e sistemas que fazem parte do processo.

Vendedor(a) é a pessoa que realiza o cadastro de um cliente novo e gera um pedido para um cliente através da aplicação web de vendas de suplementos.

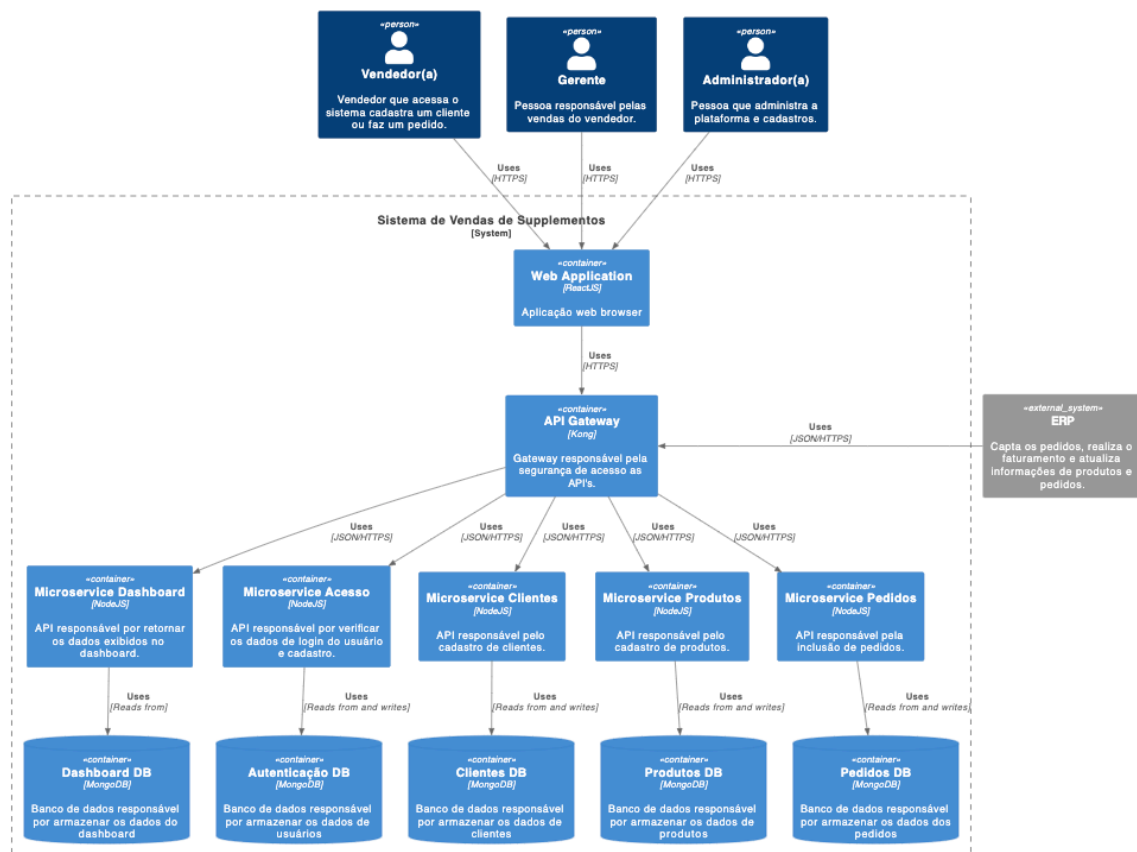
Administrador(a) é a pessoa responsável pela plataforma e pelo cadastro de usuários e cadastro de produtos para que um vendedor(a) possa gerar um pedido para um cliente.

Gerente é a pessoa responsável pela gerência dos vendedores, para atingir a vendas necessária para empresa, observando as informações das vendas no dashboard através da aplicação web de vendas de suplementos.

ERP é o sistema externo em que consome os pedidos inseridos na aplicação e realiza a atualização das informações dos produtos e pedidos através de API's.

RECEITA FEDERAL é o sistema externo que a aplicação consome através de API, para busca dos dados do cliente que será cadastrado na aplicação web de vendas de suplementos.

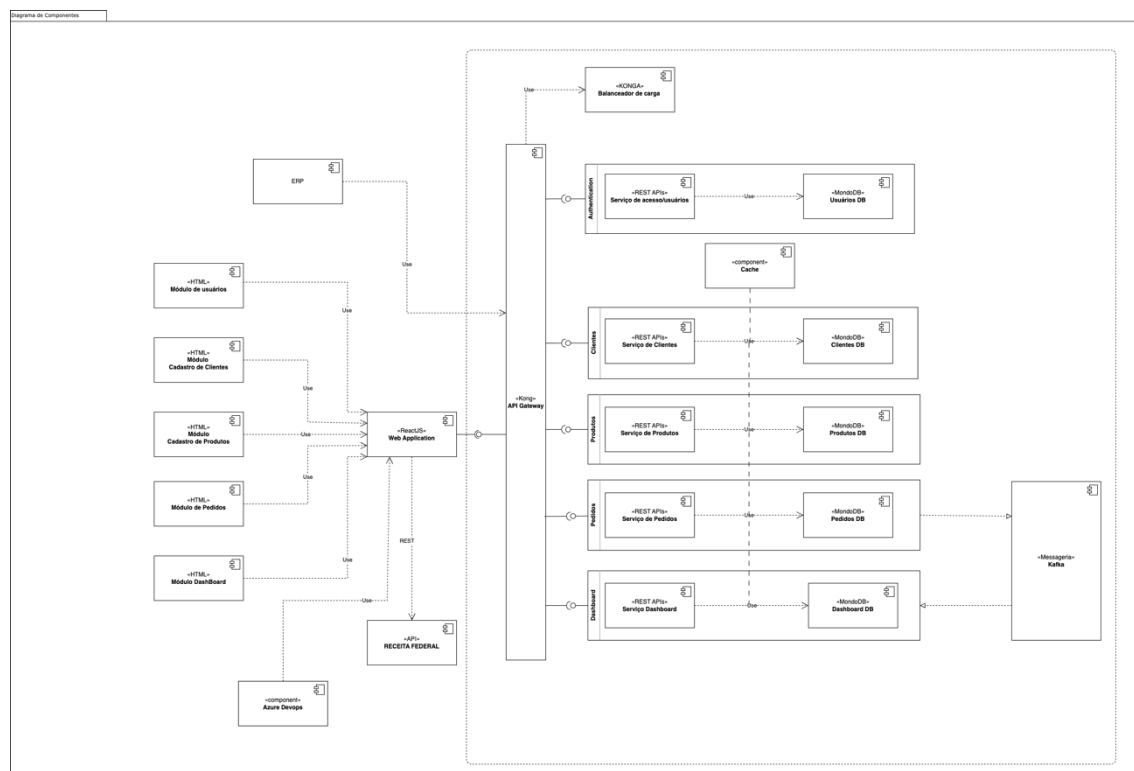
### 3.2 Diagrama de Container



**Figura 2 – Diagrama de container.**

A figura 2 apresenta os containers da aplicação web de venda de produtos.

### 3.3 Diagrama de Componentes



**Figura 3 – Diagrama de Componentes.**

Conforme diagrama apresentado na figura 3, está sendo apresentado os componentes do sistema e como são distribuídos.

- **Componente Web** – Para melhor manutenibilidade do sistema, foi desenvolvido separado do back-end (camada de serviços), com sua comunicação seja através de chamadas a APIs ao back-end. Este componente também possui seus módulos como clientes, usuários, produtos, pedidos e dashboard.
- **Componente Azure Devops** – O sistema desenhado para rodar em ambiente on premise, sendo cada componente representado na figura acima um container Docker. Sendo para a integração contínua do projeto, utilizado os serviços do Azure Devops, facilitando a realização do teste unitários e de integração após algum módulo do sistema sofrer alguma alteração, criando uma imagem Docker do módulo alterado, garantindo os requisitos não funcionais de testes e manutenibilidade.
- **Componente ERP** – Sistema interno que realiza o uso das API's de captação de novos pedidos e também responsável pela atualização dos produtos e pedidos.

- **Componente RECEITA FEDERAL** – API utilizada para busca de alguns dados dos clientes na receita federal, para realização do cadastro do cliente.
- **Componente KONG API Gateway** – Camada que fica entre o front-end e o back-end, este que possui seus módulos privados, mascarando os serviços(endpoints) reais da aplicação. Ainda em relação a API do sistema, estão disponíveis apenas com uma solicitação que possua um token válido no cabeçalho, sendo este token obtido através de uma requisição ao serviço de acesso(authenticação) ou ao front-end, em que o usuário acessa a tela de login informando usuário e senha sendo ele um acesso válido, retornando um token que deverá ser enviado nas requisições, ele não conseguirá acessar uma tela ou um recurso sem o token ou com um token inválido. Garantindo o requisito não funcional de segurança.
- **Componente KONGA** – Módulo utilizado pelo KONG API Gateway, para realizar o load balance, distribuindo a carga de utilização dos módulos/serviços, sendo assim garantindo o requisito não funcional de escalabilidade.
- **Componente Cache** – Com a finalidade de melhorar a performance do acesso aos dados do sistema foi adicionado a ferramenta Redis para uso do cache, configurado para realizar o cache das informações solicitadas, diminuindo o tráfego na rede desnecessariamente, com isso a navegação seja mais performática, atendendo ao requisito funcional de desempenho.
- **Componente Authentication** – Módulo responsável pelo cadastro de usuários e pela autenticação deles e mantém os dados armazenados em banco de dados NoSql MongoDB.
- **Componente Clientes** – Módulo responsável pela gerência dos dados dos clientes e mantém os dados armazenados em banco de dados NoSql MongoDB, que serão utilizados para realização de um pedido para ele.
- **Componente Produtos** – Módulo responsável pela gerência dos dados dos produtos e mantém os dados armazenados em banco de dados NoSql MongoDB, poderá receber atualização dos produtos. E serão utilizados para realização de um pedido para um cliente.
- **Componente Pedidos** – Módulo responsável pela gerência dos dados dos pedidos e mantém os dados armazenados em banco de dados NoSql MongoDB, que serão

inseridos na plataforma e realizando o uso da mensageria a fim de atualizar os dados de vendas no dashboard.

- **Componente Dashboard** – Módulo responsável pela gerência dos dados do dashboard e mantém os dados armazenados em banco de dados NoSql MongoDB, em que são mostrados os dados atuais das vendas, realizando o uso da mensageria para consumir os dados enviados pelo componente de pedidos.
- **Componente Mensageria Kafka** – Módulo para transmissão de dados, que permite serviços independente se comuniquem entre si.

## **4. Avaliação da Arquitetura ATM**

A Avaliação arquitetural ATM, é um processo de mitigação de riscos usado no início do ciclo de vida do desenvolvimento de software.

ATM foi desenvolvido pelo Software Engineering Institute da Carnegie Mellon University. Seu objetivo é ajudar a escolher uma arquitetura adequada para um sistema de software, encontrando compensações e pontos sensíveis.[1]

### **4.1 Análise de abordagens arquiteturais**

Abaixo foi listado as abordagens arquiteturais, com seus cenários, importância e suas complexidades (sendo definidas em B – baixa M – média e A – alta).

#### **4.1.2 Segurança**

Cenário 1: A arquitetura deve ser pensada em tornar todas as trocas/envio de informações de forma protegida, assim garantindo acesso seguro a plataforma. Um usuário sem autorização não deveria conseguir realizar o acesso as informações confidenciais do sistema. Desta adotando criptografia nas senhas utilizadas, controle de autenticação e autorização na plataforma, validando suas credenciais a todo momento de utilização do sistema, devendo redirecionar o usuário para a página de login.

**Importância: A - Complexidade: A**

#### **4.1.3 Usabilidade**

Cenário 2: A arquitetura deve considerar possuir uma boa usabilidade, facilitando a utilização da plataforma pelos usuários. Para que isso seja atendido sendo necessário, possui responsividade entre os dispositivos sejam móveis ou desktop, e possuindo interfaces intuitivas e de fácil integração de novos usuários na plataforma.

**Importância: A - Complexidade: M**

#### **4.1.4 Manutenibilidade**

Cenário 3: A arquitetura deve considerar a facilidade de manutenção da plataforma, conforme ao logo da utilização buscando ser criada uma estrutura que se possa permitir uma evolução dele, sendo utilizado padrões de projeto, sendo separados as responsabilidades dentro da arquitetura.

**Importância: A - Complexidade: A**

#### **4.1.5 Disponibilidade**

Cenário 4: A arquitetura deve considerar a capacidade de quando um módulo ficar indisponível ou fora as outras partes dos sistemas não sejam completamente afetadas.

**Importância: A - Complexidade: A**

#### **4.1.6 Interoperabilidade**

Cenário 4: A arquitetura deve considerar a capacidade de o sistema em se comunicar com outros sistemas, sejam eles por padrão de comunicação como RESTFUL e com formatos de dados em JSON, e a arquitetura de considerar a capacidade dos módulos internos da plataforma possam se comunicar entre si.

**Importância: A - Complexidade: A**

#### **4.1.7 Desempenho**

Cenário 5: A arquitetura deve considerar a capacidade em se manter estável e atingir o requisito de tempo de resposta, mesmo com picos de cargas de trabalho, com acessos rápidos aos dados no banco de dados ou em cache, sendo possível a sua escalabilidade.

**Importância: M - Complexidade: A**



## **4.2 Cenários**

Abaixo foi listado os cenários para serem efetuados na aplicação, sendo possível satisfazer os atributos de qualidade de software (requisitos não funcionas).

### **4.2.1 Cenário 1**

Segurança: O sistema deverá criptografar os dados sensíveis do software, como a senha do usuário sendo trafegado por cada requisição um token JWT, caso o usuário tente acessar um recurso da plataforma sem estar logado ele será impedido, devido a plataforma identificar que o usuário não está autenticado, redirecionando o mesmo para a tela de login. Sendo Assim este cenário atende o RNF01- Segurança.

### **4.2.2 Cenário 2**

Usabilidade: Ao acessar a plataforma deverá se adaptar ao dispositivo em questão seja ele móvel ou desktop, mantando sua semelhança independente do dispositivo. Sendo Assim este cenário atende o RN02 – Usabilidade.

### **4.2.3 Cenário 3**

Usabilidade: Ao acessar a plataforma a mesma deverá ser de fácil utilização, sem possuir um grau de dificuldade na navegação do sistema. Sendo Assim este cenário atende o RN03 – Usabilidade.

### **4.2.4 Cenário 4**

Manutenibilidade: Para o desenvolvimento de uma boa plataforma, que possa se adaptar com mais agilidade quando houver uma alteração/atualização na regra de negócio, sendo de grande importância possuir suas camadas separadas, permitindo uma maior facilidade na manutenção da plataforma. Sendo assim este cenário atende o RN04 – Manutenibilidade.

### **4.2.5 Cenário 5**

Disponibilidade: O sistema deverá ficar disponível mesmo que um de seis nós fique fora/inoperante, assim não afetando a plataforma como um todo e apenas uma parte dela. Sendo assim este cenário atende o RN05 – Disponibilidade.

### 4.2.6 Cenário 6

Interoperabilidade: O sistema se comunicar com sistemas externos, sejam no consumo de algum recurso externo como consultas no site da receita para consulta dos dados de clientes junto a Receita Federal, ou mesmo como provedor para envio de informações para ERP, sendo necessário neste último caso a comunicação através de autenticação.

Este cenário atende o RN6 – Interoperabilidade.

### 4.2.7 Cenário 7

Interoperabilidade: O sistema deve se comunicar com outros nós, para propagação dos dados em todos os nós que são necessários tais informações, garantindo a consistência dos dados. Sendo assim este cenário atende o RN7 – Interoperabilidade.

### 4.2.8 Cenário 8

Desempenho: Ao efetuar a autenticação na plataforma o sistema deverá validar e permitir ou não seu acesso a plataforma em até 5 segundos, assim permitindo uma navegação sem gargalos e com maior fluidez. Sendo assim este cenário atende o RN08 – Desempenho.

## 4.3 Evidências da Avaliação de Cenários

### 4.3.1 Cenário 1

Atributo de Qualidade:	Segurança
Requisito de Qualidade:	O sistema deverá criptografar os dados sensíveis como senha de acesso do usuário e suas requisições trafegarem com autenticação JWT.
Preocupação:	
Proteger os dados sensíveis contido na plataforma, para evitar o acesso não autorizado aos dados.	
Cenário(s):	
Cenário 1	
Ambiente:	
Sistema em operação normal	

Estímulo:	
Tentativa não autorizada de acesso a plataforma e a senha do usuário.	
Mecanismo:	
Utilização de algoritmos de criptografia para proteção da senha do usuário.	
Medida de resposta:	
Dificuldade em descobrir a senha do usuário.	
Considerações sobre a arquitetura:	
Riscos:	Mesmo a criptografia da senha do usuário poderá não ser suficiente, pois poderão tentar encontrar senhas com criptografia já conhecidas ou mesmo sites que hoje já são possíveis utilizar para de criptografar, dependendo da forma utilizada na criptografia. Assim tornando necessário por incluindo na criptografia algum método que dificulte mais ainda como o salt, combinando caracteres para gerar uma senha criptografada mais robusta.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

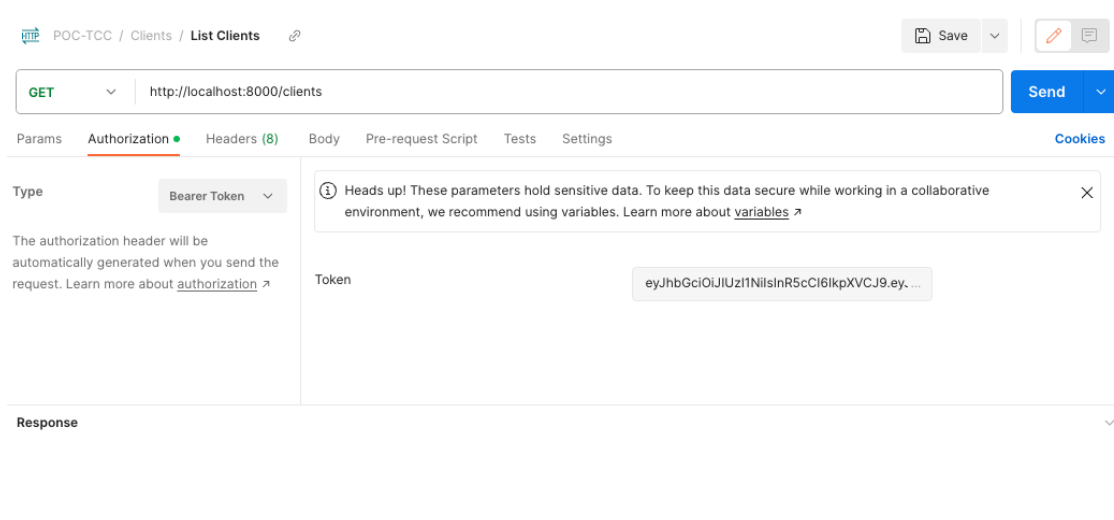
### 4.3.2 Evidência do Cenário 1

Relacionado a segurança, foi identificado a necessidade de criptografar os dados sensíveis como a senha, por se tratar de uma informação sensível.



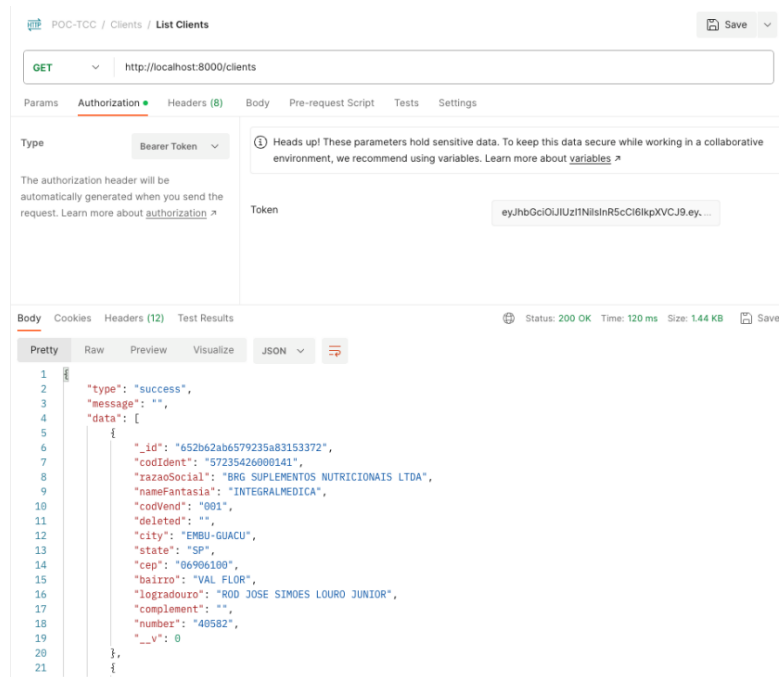
**Figura 4 – Criptografia de senha.**

As requisições e navegações na plataforma são através de um token JWT, foi identificado essa necessidade para não ser necessário trafegar a senha criptografada do usuário, tendo uma maior segurança.



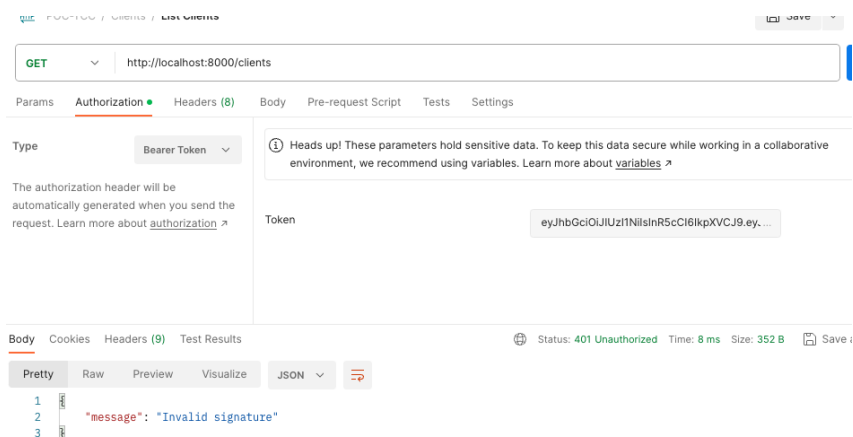
**Figura 5 – Tráfego das requisições com JWT.**

Usuário com um token JWT válido, obtendo os dados requisitados corretamente.



**Figura 6 – Tráfego das requisições com JWT válido.**

Usuário utilizando um token inválido é retornado “acesso não autorizado”, caso não tenha um token valido que foi gerado e estando dentro da validade, o usuário não conseguirá acessar a informações desejadas.



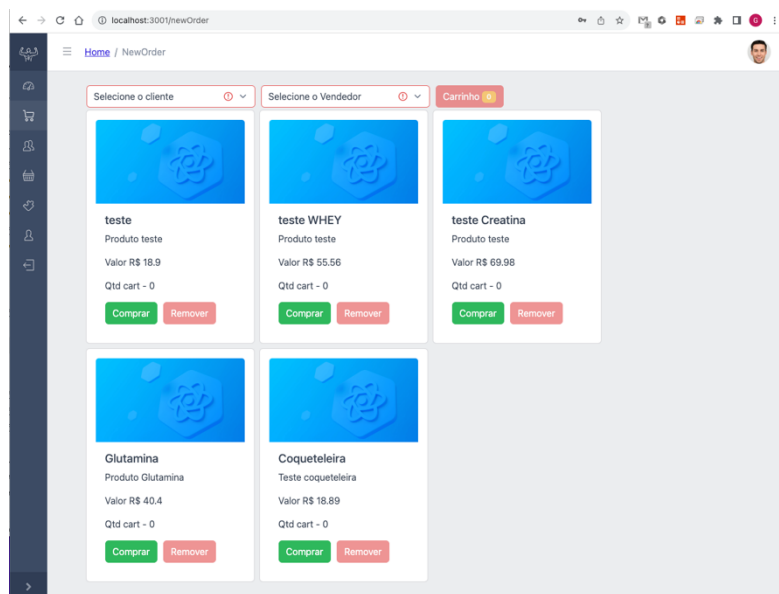
**Figura 7 – Tráfego das requisições com JWT inválido.**

## 4.4 Cenário 2

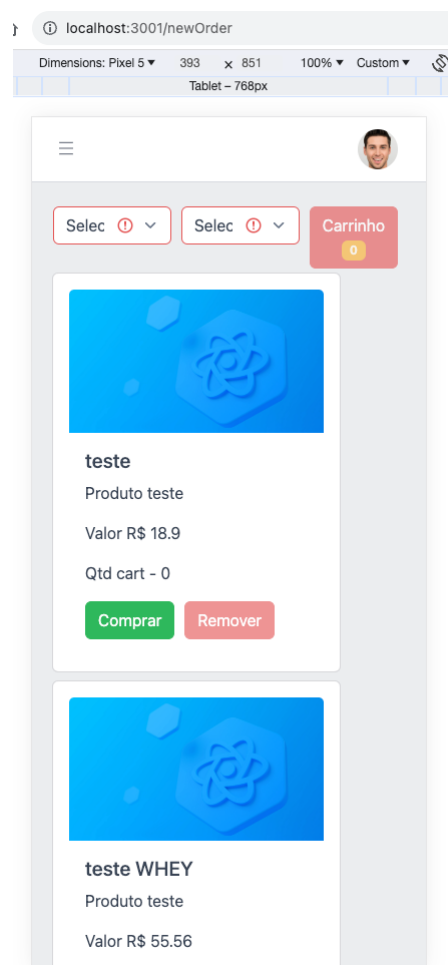
Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	A plataforma deve possuir adaptabilidade a diferentes dispositivos como desktop e móveis.
Preocupação:	
Ao realizar o acesso de diferentes dispositivos o padrão dever ser mantido, adaptando a sua resolução.	
Cenário(s):	
Cenário 2	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Ao acessar telas de diferentes tamanhos, deverá se adaptar a diferentes resoluções.	
Mecanismo:	
Utilização de template que permite a padronização das telas mantendo uma boa usabilidade entre diferentes resoluções de tela.	
Medida de resposta:	
Aprendizado na utilização do sistema.	
Considerações sobre a arquitetura:	
Riscos:	Em versões mais antigas de navegadores, poderá apresentar alguma falha em sua estrutura, devido a dispositivos mais antigos.
Pontos de Sensibilidade:	Dispositivos mais antigos.
Tradeoff:	A utilização do template se tem um esforço maior desenvolvimento, mas que isso é compensado em suas facilidades que agregarão e seguirão um padrão.

### 4.4.1 Evidência do Cenário 2

Sobre a usabilidade optou-se por utilizar no front-end o ReactJs Core UI, que é baseado no Bootstrap fornecendo componentes prontos para diversas necessidades.



**Figura 8 – Utilização do React.Js Core UI – Visualização Navegador.**



**Figura 9 – Utilização do React.Js Core UI – Visualização celular.**

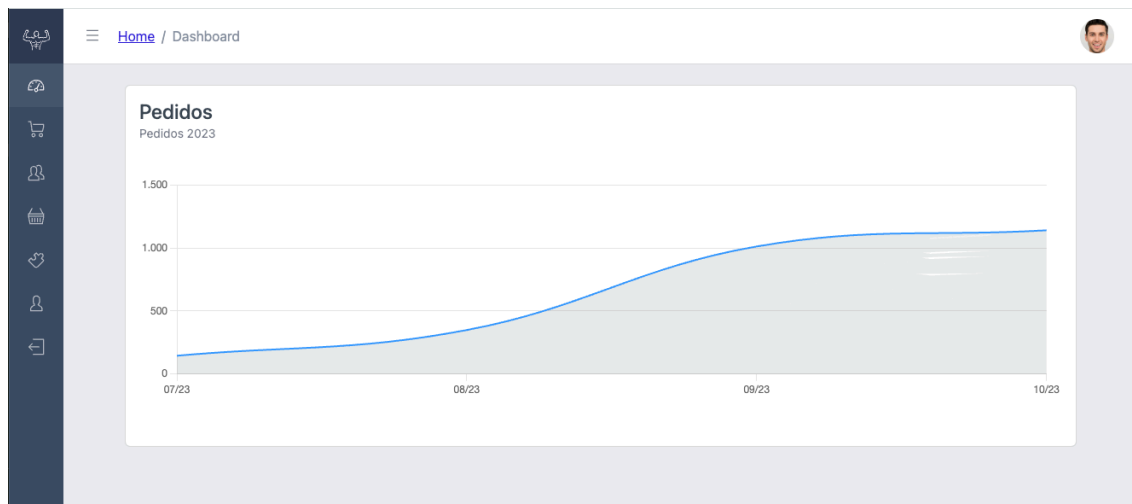
## 4.5 Cenário 3

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	A plataforma deve ser de fácil utilização, sem dificultar a navegação na mesma.
Preocupação:	
O usuário possuir uma boa usabilidade no sistema, sem complexidade e facilidade de encontrar as funções.	
Cenário(s):	
Cenário 3	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Ao acessar o sistema e realizar a navegação dentro da plataforma.	
Mecanismo:	
Utilização de template e experiência de usuário, para facilitar o uso da plataforma.	
Medida de resposta:	
Experiência de usuário.	
Considerações sobre a arquitetura:	
Riscos:	Não há
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

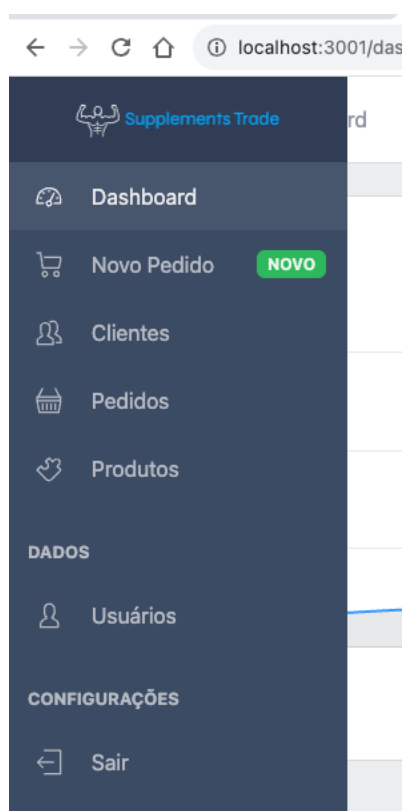


### 4.5.1 Evidência do Cenário 3

Sobre a facilidade em utilizar a plataforma, com opções de fácil acesso foi pensando na experiência do usuário, para entregar algo simples e prático na utilização.



**Figura 10 – Dashboard visualização de vendas.**



**Figura 11 – Menu de fácil acesso a todas as opções da plataforma.**

## 4.6 Cenário 4

Atributo de Qualidade:	Manutenibilidade
Requisito de Qualidade:	Arquitetura deve considerar a facilidade ao longo do tempo, buscando uma plataforma em que permita sua evolução.
Preocupação:	
O sistema deve possuir suas camadas desacopladas, permitindo uma facilidade em sua manutenibilidade.	
Cenário(s):	
Cenário 4	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Mudanças de requisitos e inclusão de novos campos.	
Mecanismo:	
Utilização de classes para rotas, controllers, models e middlerares, aplicando boas práticas cada um possuir suas responsabilidades.	
Medida de resposta:	
Tempo e a praticidade nas mudanças.	
Considerações sobre a arquitetura:	
Riscos:	Ao não está acostumado com um padrão, pode ser um pouco difícil inicialmente, que no decorrer irá se adequando, lembrando que há a necessidade em manter o padrão.
Pontos de Sensibilidade:	Caso possua necessidade em realizar alterações com frequência, poderá ser necessário uma revisão, para que atenda as tais mudanças com frequência.
Tradeoff:	Pode ocorre um aumento de complexidade no decorrer, mas que futuramente agrará na manutenibilidade a longo prazo.

### 4.6.1 Evidência do Cenário 4

O desenvolvimento foi pensado na utilização de rotas para centralização das entradas, models onde é localizada a informações das tabelas, middlewares que podem ser utilizados em mais de um local, facilitando a recriação de alguma função e temos os controllers responsáveis por buscar as informações das bases tratando-as e devolvendo-as para o usuário.

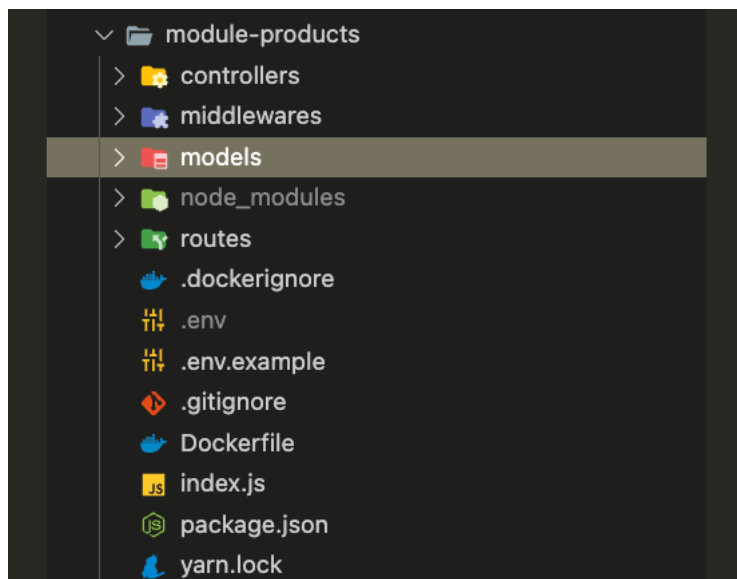


Figura 12 – Organização dos arquivos fontes.

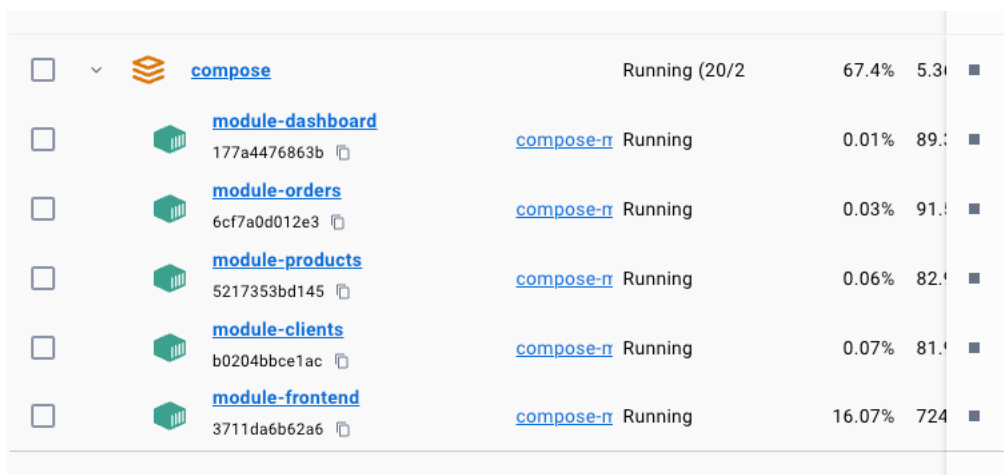
## 4.7 Cenário 5












Atributo de Qualidade:	Disponibilidade
Requisito de Qualidade:	O sistema ficará disponível mesmo que um de seus nós fique fora.
Preocupação:	
Não deixar o sistema todo fora, caso a cause seja em apenas um do nós, assim não afetando o restante do sistema.	
Cenário(s):	
Cenário 5	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Algum dos nós caia por algum problema em específico, ou mesmo uma atualização que afetou a disponibilidade do nó.	

Mecanismo:	
Foi utilizado microsserviços para construção dos nós e que pudessem ser independentes.	
Medida de resposta:	
Utilização do sistema mesmo com algum nó fora.	
Considerações sobre a arquitetura:	
Riscos:	Mesmo que seja utilizado microsserviços, caso seja algum problema que possa afetar todos os microsserviços, poderá a plataforma ficando fora do ar.
Pontos de Sensibilidade:	Complexidade na criação dos microsseviços e a segurança entre eles.
Tradeoff:	O desenvolvimento de microsserviços pode exigir um esforço adicional.

#### 4.7.1 Evidência do Cenário 5

O desenvolvimento foi pensando na utilização de microsserviços, em que cada um dos nós seria um container, operando de forma independentes sem um afetar diretamente o outro.



<input type="checkbox"/>		<a href="#">compose</a>	Running (20/2)	67.4%	5.30	■
<input type="checkbox"/>		<a href="#">module-dashboard</a> 177a4476863b 	<a href="#">compose-n</a> Running	0.01%	89.5	■
<input type="checkbox"/>		<a href="#">module-orders</a> 6cf7a0d012e3 	<a href="#">compose-n</a> Running	0.03%	91.1	■
<input type="checkbox"/>		<a href="#">module-products</a> 5217353bd145 	<a href="#">compose-n</a> Running	0.06%	82.9	■
<input type="checkbox"/>		<a href="#">module-clients</a> b0204bbce1ac 	<a href="#">compose-n</a> Running	0.07%	81.9	■
<input type="checkbox"/>		<a href="#">module-frontend</a> 3711da6b62a6 	<a href="#">compose-n</a> Running	16.07%	724	■

**Figura 13 – Containers dos serviços de Dashboard, Pedidos, Produtos, Clientes.**

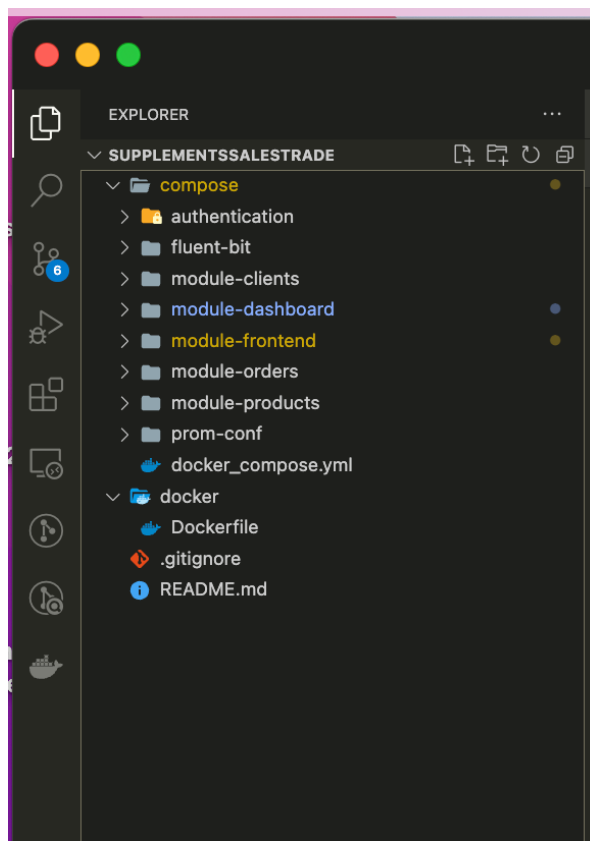


Figura 14 – Pasta com os arquivos para criar os containers.

## 4.8 Cenário 6

Atributo de Qualidade:	Interoperabilidade
Requisito de Qualidade:	Arquitetura devera possuir a capacidade de se comunicar com outros sistemas de software em outras tecnologias de padrão de comunicação aceitos, como RESTFul, utilizando formatos de dados como JSON.
Preocupação:	
	Garantir que o sistema possa integrar-se com outros sistemas de softwares, com padrões amplamente adotados no mercado.
Cenário(s):	
	Cenário 6
Ambiente:	
	Sistema em operação normal
Estímulo:	

Necessidade em buscar os dados cadastrais de clientes junto a Receita Federal.	
Mecanismo:	
Utilização de protocolos de comunicação amplamente aceitos, para integrar com a consulta dos dados cadastrais externos.	
Medida de resposta:	
Sucesso na realização da consulta e integração pelos meios de comunicação aceitos.	
Considerações sobre a arquitetura:	
Riscos:	O sistema pode estar vulnerável a ameaças de segurança levando o comprometimento de dados e roubo de informações sensíveis.
Pontos de Sensibilidade:	Armazenamento dos dados cadastrais que são buscados junto a receita. Sendo necessário mascarar dos dados, assim necessário um ajuste na arquitetura.
Tradeoff:	Sempre é bom pensar em segurança, por se tratar de dados de clientes, que acabam por ser dados sensíveis, podendo trazer algum impacto para ele. Sendo necessário garantir a segurança conforme mencionado anteriormente, além de estar alinhado com a infraestrutura, para evitar possíveis ataques que possam acabar em vazamentos de dados.

### 4.8.1 Evidência Cenário 6

Integração realiza com a receita para busca dos dados cadastrados dos clientes, para utilização em nossa plataforma para preenchimento e utilização em assuntos fiscais.

```

You, 1 second ago | 1 author (You)
import dotenv from "dotenv-safe";
import jwt from "jsonwebtoken";
import Clients from "../models/Clients.js";

dotenv.config();

export const loadCheckClient = async (request, response) => {
  const codIdentification = request.params.codIdentification;
  let data = {
    type: "error",
    message: "",
    data: null,
  };

  if (codIdentification) {
    let hasClient = await Clients.findOne({ codIdent: codIdentification });
    if (!hasClient) {
      let req = await fetch(
        "https://www.receitaws.com.br/v1/cnpj/" + codIdentification,
        {
          method: "get",
          headers: {
            "Content-Type": "application/json",
            "Authorization": "Bearer " + process.env.SECRET,
          },
        },
      );

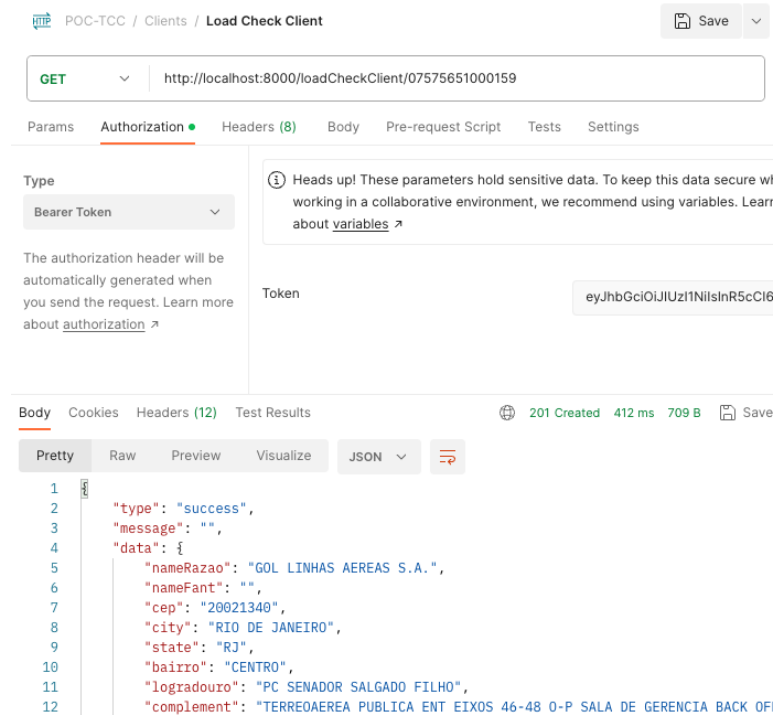
      const result = await req.json();

      if (result && result?.status.toLowerCase() !== "error") {
        const newCep = result?.cep
          ? result.cep.replace(/[^\d]/g, "")
          : result.cep;

        data.type = "success";
        data.data = {
          nameRazao: result.nome,
          nameFant: result.fantasia,
          cep: newCep,
          city: result.municipio,
          state: result.uf,
          bairro: result.bairro,
          logradouro: result.logradouro,
          complement: result.complemento,
          number: result.numero,
        };
      } else if (result && result?.status.toLowerCase() === "error") {
        data.message = result?.message ? result.message : "Ocorreu um erro!";
      } else {
        data.type = "warning";
        data.message = "Não foi possível obter os dados, preencha manualmente!";
      }
    } else {
      data.message = "Cliente já possui cadastro!";
    }
  }
};

```

**Figura 15** – Controller responsável pela comunicação.



**Figura 16 – Teste de integração obtendo o conteúdo em JSON.**

## 4.9 Cenário 7

Atributo de Qualidade:	Interoperabilidade
Requisito de Qualidade:	A Arquitetura devera se conectar com outros nós através de eventos, para propagação da informação.
Preocupação:	
Garantir a integridade das informações em todos os nós, pois como são independentes a informação só chegará no outro, por meio de uma integração entre os sistemas.	
Cenário(s):	
Cenário 7	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Integridade dos dados em todos os nós.	
Mecanismo:	
Utilização de eventos com Kafka, para envio de eventos cos os dados de um nó para ou outro.	



Medida de resposta:	
Sucesso na realização da integração do nós por meios de eventos, vide as evidências.	
Considerações sobre a arquitetura:	
Riscos:	Sistema de mensageria caso venha a falhar, será necessário ajusta a arquitetura para que mesmo que o Kafka fique fora, ele guarde os eventos para assim que estiver disponível realizar uma nova tentativa.
Pontos de Sensibilidade:	Sistema de mensageria ficar fora, conforme mencionado sem a devida arquitetura eventos/dados, não serão propagados entre os microserviços.
Tradeoff:	Por se tratar eventos, é importante pensar na propagação dos dados, para manter a integridade dos dados em todos os pontos do sistema.

### 4.9.1 Evidência de Cenário 7

**module-orders**  
[compose-module-orders](#)  
6cf7a0d012e3   
[3338:3338](#)

**STATUS**  
Running (6 hours ago)

**Logs**
Inspect
Bind mounts
Exec
Files
Stats

```

2023-10-15 20:16:18 Resposta {"type":"success","message":"Dados registrados com sucesso!","data":{"clientId":"652b62ab6579235a83153372","codVend":"001","totalPrice":203.73000000000002,"totalQuant":5,"dateCreated":"2023-10-15T23:16:18.068Z","deleted":"","_id":"652c72c25a4ff102b569c34d","__v":0}}
2023-10-15 20:16:33 Resposta {"type":"success","message":"Dados registrados com sucesso!","data":{"clientId":"652b69259de657d9583fb1a6","codVend":"002","totalPrice":203.73,"totalQuant":5,"dateCreated":"2023-10-15T23:16:33.503Z","deleted":"","_id":"652c72d15a4ff102b569c34f","__v":0}}
2023-10-15 20:16:47 Resposta {"type":"success","message":"Dados registrados com sucesso!","data":{"clientId":"652b69eb1e55a163df039b1a","codVend":"001","totalPrice":203.73000000000002,"totalQuant":5,"dateCreated":"2023-10-15T23:16:47.469Z","deleted":"","_id":"652c72df5a4ff102b569c351","__v":0}}

```

**Figura 17 – Log do evento enviado para o container Dashboard.**

module-dashboard

compose-module-dashboard

177a4476863b

3339:3339

STATUS

Running (6 hours ago)

Logs

Inspect

Bind mounts

Exec

Files

Stats

```

2023-10-15 14:10:37 MongoDB is connected
2023-10-15 20:16:18 Resposta {"state":"SP","codIdent":"652b62ab6579235a83153372","razaoSocial":"BRG SUPLEMENTOS NUTRICIONA
IS LTDA","nameFantasia":"INTEGRALMEDICA","quantItens":5,"total":203.73000000000002,"dateCreated":"2023-10-15T23:16:17.968Z
","codVend":"001","deleted":"","_id":"652c72c1b3ae8b1622a85b9e","__v":0}
2023-10-15 20:16:33 Resposta {"state":"SP","codIdent":"652b69259de657d9583fb1a6","razaoSocial":"AMAZON SERVICOS DE VAREJO
DO BRASIL LTDA.","nameFantasia":"AMAZON.COM.BR","quantItens":5,"total":203.73,"dateCreated":"2023-10-15T23:16:33.477Z","co
dVend":"002","deleted":"","_id":"652c72d1b3ae8b1622a85bab","__v":0}
2023-10-15 20:16:47 Resposta {"state":"RJ","codIdent":"652b69eb1e55a163df039b1a","razaoSocial":"AMERICANAS S.A - EM RECUPE
RACAO JUDICIAL","nameFantasia":"AMERICANAS S.A","quantItens":5,"total":203.73000000000002,"dateCreated":"2023-10-15T23:16:
47.455Z","codVend":"001","deleted":"","_id":"652c72dfb3ae8b1622a85bb8","__v":0}

```

Figura 18 – Log do evento recebido pelo container Dashboard.

## All topics

☒ Hide internal topics

Topics	Partitions
Topic name	Total partitions
<a href="#">data-dashboard</a>	1
<a href="#">orders-new-response</a>	1

Figura 19 – Topics criados no Kafka.

## All consumer groups

Consumer group ID	Messages behind	Number of consumers	Number of topics
<a href="#">orders-new-group-receiver</a>	0	1	1
<a href="#">_confluent-controlcenter-6-0-0-1-command</a>	0	1	1
<a href="#">_confluent-controlcenter-6-0-0-1</a>	96	8	6
<a href="#">orders-dashboard-group-receiver</a>	0	1	1

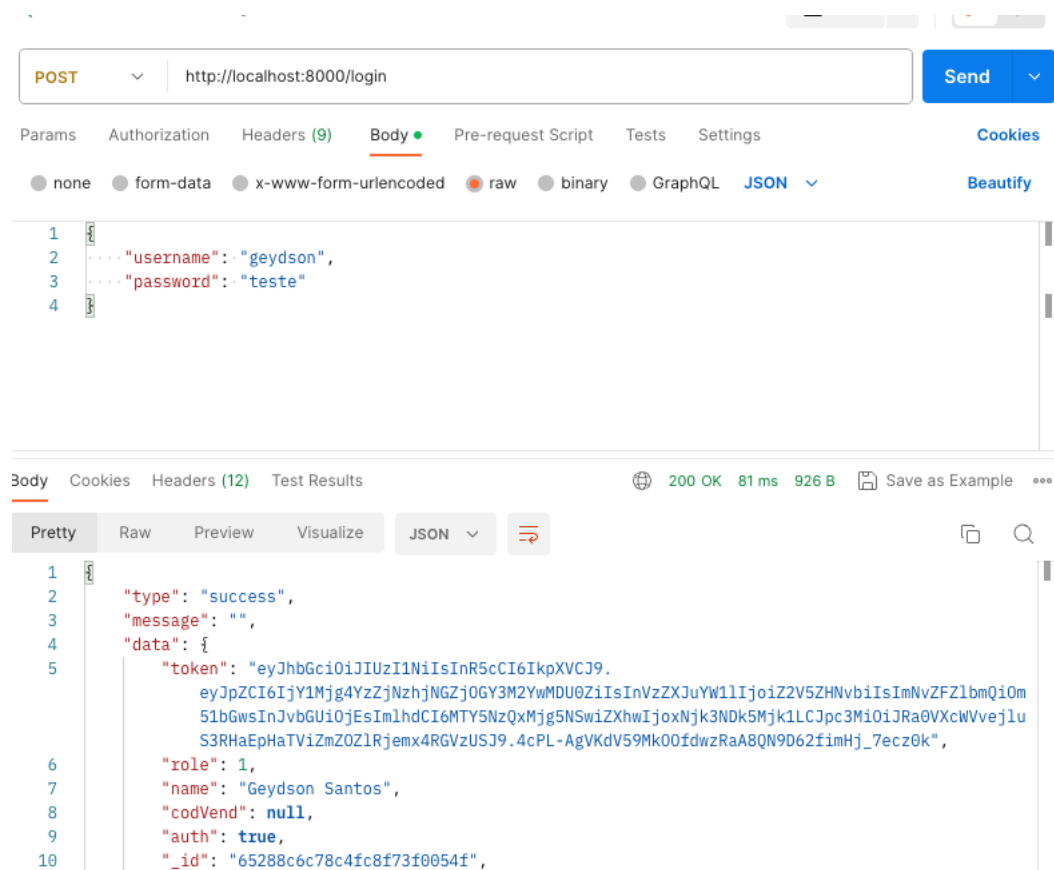
Figura 20 – Grupo de consumer no Kafka.

## 5. Cenário 8

Atributo de Qualidade:	Desempenho
Requisito de Qualidade:	O sistema deverá apresentar para o usuário a tela principal, após inserir os dados de acesso em menos de 5 segundos
Preocupação:	
Garantir uma melhor experiência de fluidez para o usuário, demonstrando o tempo de resposta rápido.	
Cenário(s):	
Cenário 8	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Login no sistema.	
Mecanismo:	
Implementação de autenticação com usuário e senha. Utilização do JWT para geração de token e posteriormente ser validado em todas as rotas do sistema.	
Medida de resposta:	
Tempo de resposta para geração do token de acesso as telas do sistema.	
Considerações sobre a arquitetura:	
Riscos:	Sobrecarga caso ocorra muitos acessos simultâneos.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

### 5.1 Evidências de Cenário 8

Realizado acesso ao sistema utilizando a ferramenta POSTMAN, em que foi obtido o tempo de resposta de 81ms. Para um ambiente de desenvolvimento é um tempo excelente, mas no uso do dia a dia e pela quantidade de acessos simultâneos possa subir, mas aumentando a infraestrutura conforme a demanda se mantará aceitável.



**Figura 21 – Tempo de resposta do acesso ao sistema em 81ms.**

## 6. Avaliação Crítica dos Resultados

A proposta do projeto ao que foi concluído, validamos que os requisitos não funcionais foram atendidos e validados com sucesso, como nem tudo sai perfeito vemos que alguns pontos podem ser melhorados.

Ponto avaliado	Descrição
Arquitetura Microserviços	A utilização de microserviços permite uma melhor escalabilidade dos serviços, porém pode ter mais custos ou não, depende de mais pessoas desenvolvendo e inicialmente pode se complexa.
Usabilidade	Com atualização do React.JS e do template Core UI, podemos seguir um padrão na criação das interfaces, com a compatibilidade em diversos dispositivos, sem a necessidade e muito esforço pois fornece diversos componentes que podem ser utilizados. Para equipes maiores e que mantenham o padrão seria bem útil, inicialmente poderia ter um pouco de dificuldade em relação a adaptabilidade, que com o tempo se resolveria.
Segurança	Em relação a segurança foi realizando a máxima proteção possível dentro tempo hábil, com criptografia de senha e navegação com JWT, porém a ainda pontos a melhorar, inclusive a segurança entre os microserviços.
Desempenho	Ao compreender o requisito de desempenho, conseguimos compreender, como uma boa experiência para o usuário e com fluidez, conseguimos saber o quanto importante é. O sistema sendo acompanhado dia a dia, poderá vermos a necessidade de um crescimento na infraestrutura, visando a escalabilidade em momentos de muitos acessos.

## 7. Conclusão

Com a finalização deste trabalho conseguimos perceber ou ter mais clareza, de quão importante é a definição da arquitetura de um sistema, envolve muitos pontos, possui muitas formas em realizar o que deseja cada uma contendo características melhores que as outras em cada uma de suas especialidades. Realizando uma avaliação como está, podemos ver sem um planejamento e definição do que será feito e como será feito, acabamos que podemos deixar a desejar em alguns aspectos técnicos, definição e criação em si seja do problema que visa solucionar ou mesmo uma nova solução, que devemos pensar nos detalhes para chegar naquilo que você almeja criar.

A arquitetura do projeto que foi adota, é baseada em microsserviços já que o meu proposito seria criar uma arquitetura escalável, modular e independente(nós), foi um grande desafio criar tanto o back-end quanto o front-end, e poder ver funcionando como você imagina, caso tenha opte também por microsservicos e aconselhável se aprofundar no tema, pois reque um pouco de conhecimento, já em relação ao front-end foi utilizado React.js juntamente com o template Core UI, que ajuda bastante no desenvolvimento por possuir bastante componentes prontos, por ser um template pronto caso queira ele te prende um pouco ao template, mas ele também permite um customização, que poderá chegar em algo que te agrada ou que tecnicamente fique bem mais amigável a sua utilização.

Para concluir, a definição de forma cuidadosa da arquitetura que pretende utilizar, juntamente a uma avaliação crítica, obtêm-se uma arquitetura robusta e eficiente, com uma constante busca por uma arquitetura equilibrada é um processo contínuo, visto as melhorias, necessidades e requisitos e possível evoluir de forma contínua e saudável.

## REFERÊNCIAS

- Architecture tradeoff analysis method – Wikipedia.  
[https://en.wikipedia.org/wiki/Architecture\\_tradeoff\\_analysis\\_method](https://en.wikipedia.org/wiki/Architecture_tradeoff_analysis_method), 2023.
- C4 PlantUML. Documentação do C4 PlantUML. <<https://C4-PlantUML>>, 2023
- Core UI. Documentação do Core UI. <<https://pt-br.legacy.reactjs.org/>>, 2023
- Docker. Documentação do Docker. <<https://docs.docker.com/>>, 2023
- Kafka. Documentação do Kafka. <<https://kafka.apache.org/>>, 2023
- KONG API Gateway. Documentação do KONG.  
<https://konghq.com/solutions/istio-gateway>, 2023
- NodeJS. Documentação do NodeJS. <<https://nodejs.org/en>>, 2023
- React JS. Documentação do React JS. <<https://pt-br.legacy.reactjs.org/>>, 2023
- Redis. Documentação do Redis. <<https://redis.io>>, 2023

## APÊNDICES

### **URL do vídeo e apresentação 1:**

<https://drive.google.com/drive/folders/1YfsMOlnmm-a2j6QEt-76Xv75sJ1Lit0P?usp=sharing>

### **URL do vídeo e apresentação 2:**

<https://drive.google.com/drive/folders/1WsifnG4yNO9I4DYtD14qiStt1aVkWVjv?usp=sharing>