Engenharia de software: Plataforma de vendas online

Grupo: Gustavo Pereira de Carvalho
Matheus Bezerra Dantas Saraiva
Arthur Gaag Duarte David
Leonardo Moniz Sodre Lopes Teixeira

O projeto foca na criação de uma plataforma de vendas online, ainda sem nome definido, para que vendedores e clientes possam realizar transações de forma simples. Dessa forma, são tidos como objetivos principais o cadastro dos usuários, clientes e vendedores, bem como de produtos por parte dos vendedores, além da remoção de produtos, consulta de produtos e usuário, implementação de formas de pagamento e demais funções essenciais.

As primeiras funcionalidades escolhidas para serem implementadas estão dispostas na forma de histórias de usuários listadas abaixo:

1 - Como vendedor, desejo que seja possível cadastrar produtos, para adicionar mais itens ao meu catálogo de vendas.

Critério de aceitação: o vendedor deve ter a possibilidade de adicionar novos produtos no banco de dados.

2 - Como vendedor, desejo que seja possível remover produtos, para remover produtos já vendidos do meu catálogo de vendas.

Critério de aceitação: o vendedor deve ter a possibilidade de remover produtos no banco de dados.

3 - Como usuário, desejo que seja possível me cadastrar na plataforma, para salvar meus dados e mantê-los seguros.

Critério de aceitação: O usuário deve ter a possibilidade de se cadastrar como cliente ou vendedor para proteger sua conta.

4 - Como usuário, desejo que seja possível consultar os produtos, clientes e vendedores existentes, para buscar por produtos e outros usuários.

Critério de aceitação: O usuário cadastrado deve poder consultar tanto produtos, quanto outros usuários no banco de dados.

5 - Como usuário, desejo poder realizar o login na plataforma, para acessar minha informações salvas e permissões do perfil.

Critério de aceitação: Todo usuário cadastrado deve poder realizar login, utilizando seu login e senha, além de receber um token referente ao seu nível de autorização.

O funcionamento da plataforma foi ilustrado através de diagramas UML. Os modelos escolhidos foram o diagrama de casos de uso, para uma visualização mais simples das funcionalidades e permissões da plataforma, e o diagrama de sequência, para demonstrar de maneira mais detalhada o funcionamento do programa.

Segue abaixo o diagrama de casos de uso e o diagrama de sequência, respectivamente:

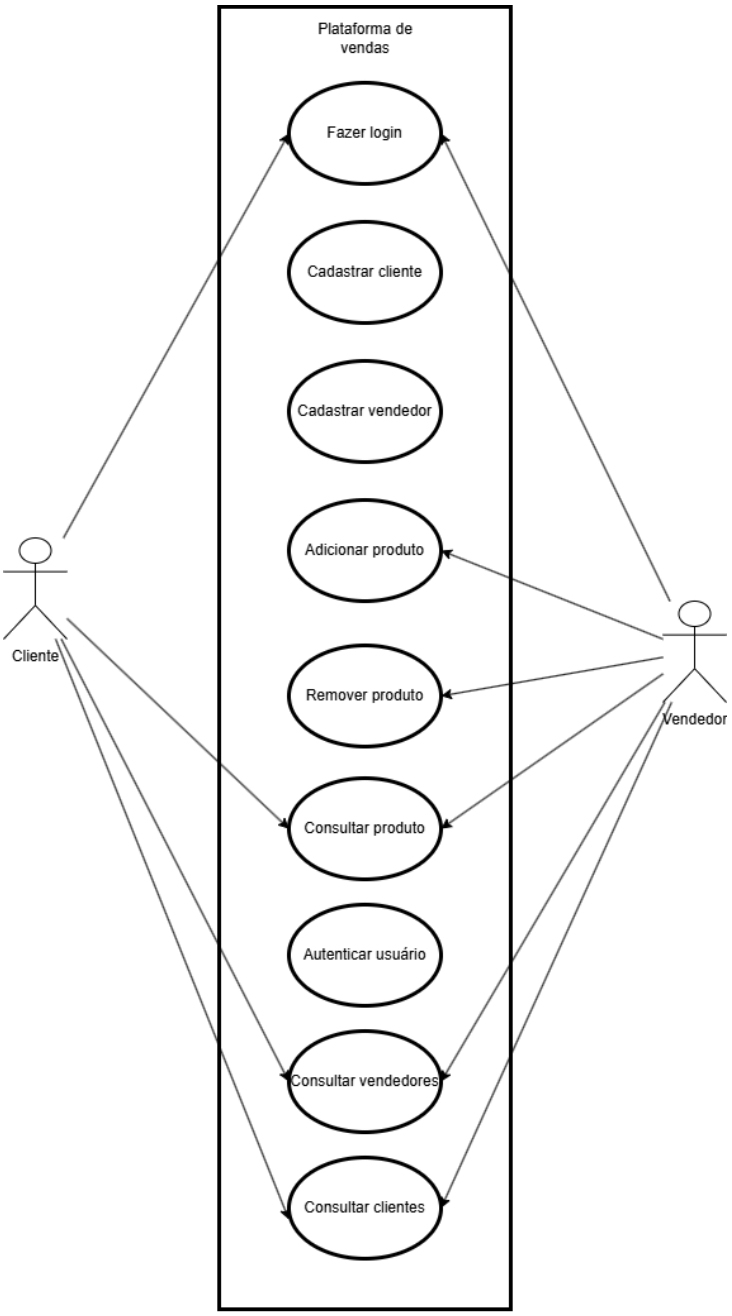

Figura 1 - Diagrama de casos de uso.

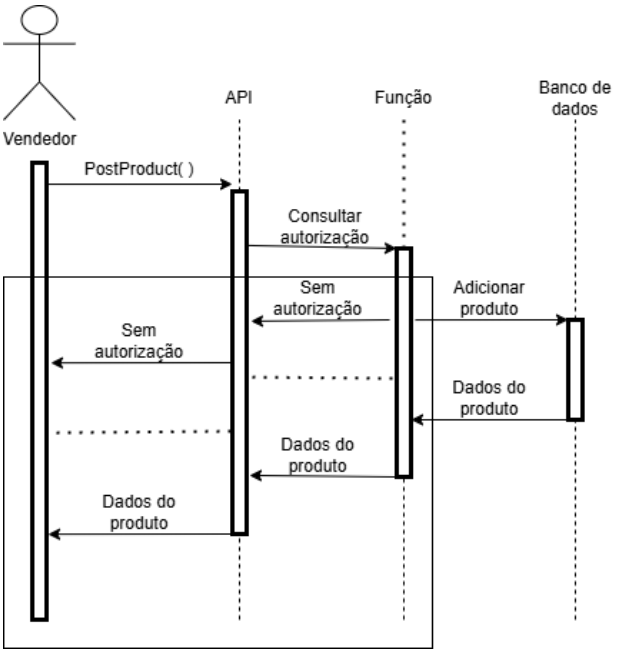

Figura 2 - Diagrama de sequência (Adição de produto).

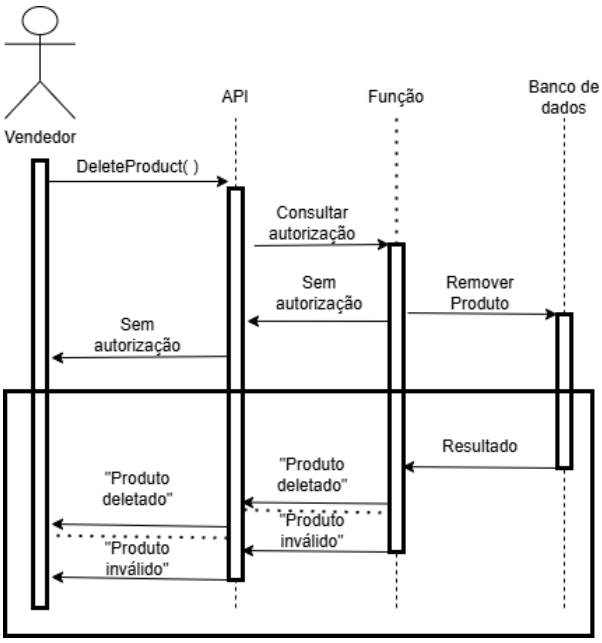

Figura 3 - Diagrama de sequência (Remoção de produto).

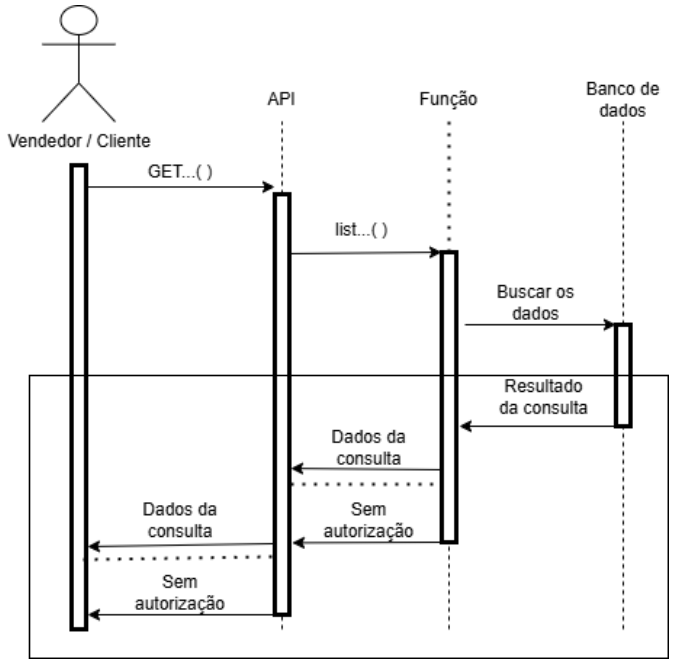

Figura 4 - Diagrama de sequência (Consulta).

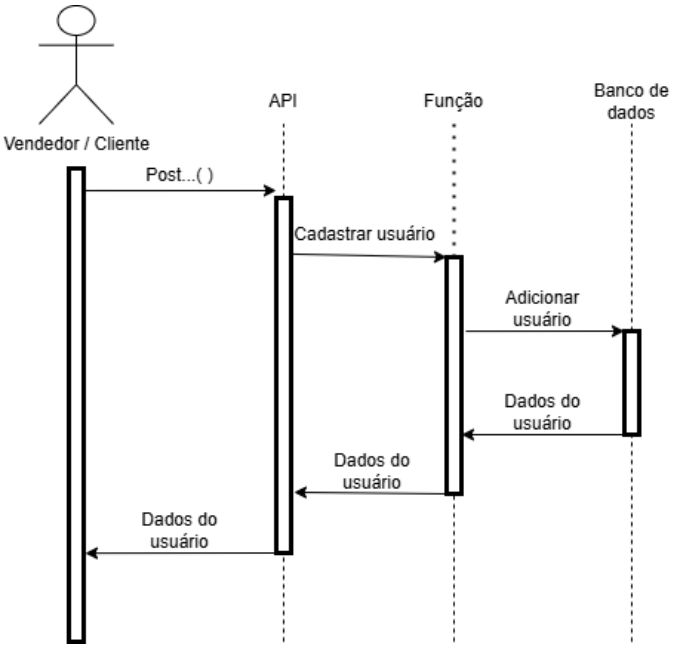

Figura 5 - Diagrama de sequência (Cadastro).

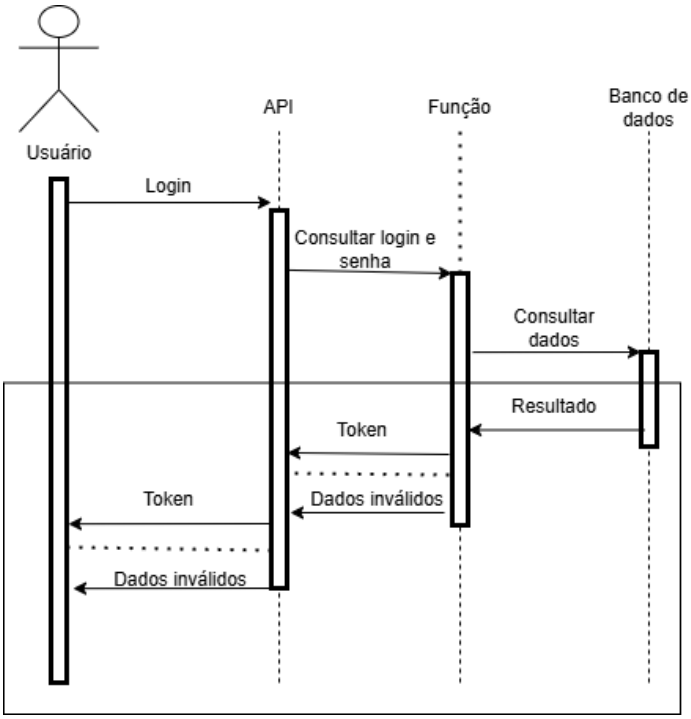

Figura 6 - Diagrama de sequência (Login).

Com base nos diagramas, o código do projeto foi implementado. A linguagem utilizada foi GO, juntamente com o framework Gin. Além disso, o JWT foi utilizado para criação de tokens e autenticação.

Em resumo, o funcionamento do código se dá com o Gin gerenciando as rotas e middleware, enquanto o JWT serve para autenticar e autorizar os usuários.

Além disso, o código possui um Dockerfile implementado, que permite a criação de um container e uso do docker. No entanto, apesar de ser possível criar a imagem do docker e executá-lo, não foi possível testar o funcionamento, uma vez que não houve êxito, ao tentar acessar a plataforma pelo navegador.

Segue abaixo uma sequência de comandos que podem ser usados para testar as funcionalidades do programa.

Vale lembrar que em todas as funcionalidades que exigem autenticação, deve ser feito o login, seguido do uso do token retornado. O token deve ser inserido no lugar de “TOKEN_JWT”.

Teste 1: Cadastro de vendedor:

curl -X POST http://localhost:8080/sellers -H "Content-Type: application/json" -d '{"name": "Nova Loja", "cnpj": "12345678000102", "phone": "11988880001", "email": "nova_loja@example.com", "password": "novaSenha456", "address": "Novo Endereço"}'

Resultado esperado: {"id":"D","name":"Nova Loja","cnpj":"12345678000102","phone":"11988880001","email":"nova_loja@example.com","password":"novaSenha456","address":"Novo Endereço"}

Teste 2: Cadastro de cliente:

curl -X POST http://localhost:8080/customers -H "Content-Type: application/json" -d '{"name": "Novo Cliente", "email": "novo_cliente@example.com", "password": "senha123", "cpf": "12345678900", "phone": "11999990000", "address": "Endereço Novo"}'

Resultado esperado: {"id":"D","name":"Novo Cliente","email":"novo_cliente@example.com","password":"senha123","cpf":"12345678900", "phone":"11999990000","address":"Endereço Novo"}

Teste 3: Login:

Login como cliente: curl -X POST http://localhost:8080/login \
-H "Content-Type: application/json" \
-d '{"email": "joao@example.com", "password": "12345"}'

Login como vendedor: curl -X POST http://localhost:8080/login \
-H "Content-Type: application/json" \
-d '{"email": "loja_a@example.com", "password": "password123"}'

Resultado esperado: {
  "token": "TOKEN_JWT"
}

Teste 4: Adição de produto:

curl -X POST http://localhost:8080/products \
-H "Content-Type: application/json" \
-H "Authorization: TOKEN_JWT" \
-d '{"name": "Teclado", "type": "Eletrônico", "quantity": 50}'

Resultado esperado: {
  "id": "F",
  "name": "Teclado",
  "type": "Eletrônico",
  "quantity": 50
}

Teste 5: Remoção de produto:

curl -X DELETE http://localhost:8080/products/F \
-H "Authorization: TOKEN_JWT"

Resultado esperado: {
"message": "Product deleted"
}

Teste 6: Consultas:

Listar produtos: curl -X GET http://localhost:8080/products

Listar vendedores: curl -X GET http://localhost:8080/sellers \
-H "Authorization: TOKEN_JWT"

Listar clientes: curl -X GET http://localhost:8080/customers \
-H "Authorization: TOKEN_JWT"

Por fim, para uso do docker, foi criado o arquivo Dockerfile. Os comandos necessários para execução do docker, estão listados abaixo e consideram que o docker já está instalado na máquina que irá executar o código.

Buildar o docker:
docker build -t prototipo-go .

Rodar o docker:
docker run -p 8080:8080 prototipo-go

Parar o docker:
docker stop <container_id>

Listar dockers:
docker ps

Log do contêiner:
docker logs <container_id>

Uma vez que o docker está em execução basta acessá-lo pelo navegador no endereço localhost:8080.