Repositórios de Dados e NoSQL

ShopHero: uma plataforma online de shopping para parceiros

Grupo 2

Aline Bini
Ana Lívia Franco
Ana Pris
João Squinelato
Marcelo Pena
Thais Carvalho

Sumário:

Lista de Figuras:

Proposta

O objetivo deste projeto é implementar uma plataforma online de shopping para parceiros, com o intuito de facilitar a venda de eletrodomésticos e eletrônicos. Nesse contexto, empresas parceiras que atuam em tais segmentos poderão promover suas ofertas através da plataforma **ShopHero**. Assim, os clientes terão acesso à plataforma, onde poderão consultar produtos de diversos parceiros, pesquisar sobre as melhores ofertas e realizar suas compras de forma prática.

Para isso, a plataforma possuirá comunicação com os parceiros em *near-real-time*, compartilhando informações do cliente e do pedido sempre que uma compra for efetuada. Além disso, o parceiro deverá garantir o fornecimento do valor do frete e do tempo estimado de entrega, além de também ser responsável pela logística de entrega.

Nesse contexto, a plataforma **ShopHero** deverá ser altamente escalável, disponível e de baixa latência, a fim de garantir a correta integração com os parceiros e uma experiência de usuário eficiente. Para atingir tais requisitos, o armazenamento e gerenciamento dos dados deverá ser realizado pelo recurso Amazon DynamoDB como banco de dados NoSQL.

Repositório do Projeto

Com o intuito de agrupar integralmente os conteúdos e códigos utilizados para a realização deste projeto, foi criado o repositório GitHub Squinelato / eEDB-016-2024-2, o qual pode ser acessado de forma pública.

Requisitos funcionais

Para maior clareza sobre o escopo abrangido pela plataforma, a seguir são apresentadas as principais funcionalidades que devem ser garantidas e suas respectivas regras:

1. O cliente poderá ter mais de um endereço associado à sua conta;
2. As empresas parceiras devem ser cadastradas na plataforma **ShopHero** pelos administradores da plataforma;
3. Os produtos serão agrupados em categorias e subcategorias para melhor visibilidade ao usuário;
4. O estoque dos produtos deverá ser gerenciado pelo o parceiro, a plataforma **ShopHero** apenas apresenta a informação de disponibilidade;
5. O parceiro é responsável por fornecer o valor do frete, tempo estimado de entrega e realizar a logística de envio do pedido. Além disso, o parceiro deverá informar os status de entrega dos pedidos à plataforma ShopHero, para que o status seja alterado e disponibilizado para que o cliente consulte;
6. Ao efetuar uma compra, após a confirmação de recebimento do pedido, o cliente poderá adicionar comentários e avaliações sobre cada produto adquirido. Essas informações serão visíveis por meio da expansão de “mais detalhes” na página do produto;
7. O cliente poderá criar listas de desejo para melhor visualizar seus produtos favoritos. Nesse sentido, caso um produto da lista de desejos tornar-se indisponível o mesmo deve ser mantido na lista, porém com sinalização sobre a falta de estoque do produto;
8. O cliente possuirá uma página de recomendações com sugestões de produtos que tenham afinidade com os itens de sua lista de desejo e de seus pedidos realizados.

Requisitos não funcionais

A partir das informações apresentadas anteriormente, para a plataforma **ShopHero**, foram identificados os requisitos não funcionais listados abaixo:

1. Fluidez para lidar com milhões de consultas por segundo, garantindo escalabilidade, baixa latência e consistência dos dados.

Wireframe

Com o intuito de garantir a experiência do usuário de forma satisfatória, foram idealizados diversos protótipos, em versão mobile, sobre as páginas que a plataforma deverá manter. Tais protótipos, apresentados a seguir, além de evidenciar as funcionalidades essenciais do projeto, também permitem a identificação das necessidades de usabilidade.

Nesse sentido, visando uma boa experiência ao usuário sobre a criação, acesso e manutenção de sua conta na plataforma **ShopHero**, foram desenvolvidos os protótipos abaixo: login do cliente, criação da conta do cliente, atualização da conta do cliente e remoção da conta do cliente.

Figura 1: Protótipos de “Login”, “Criar conta”, “Atualizar Conta” e “Remover Conta”.

Além disso, ainda sobre a experiência do cliente sobre suas informações, foi desenvolvido o protótipo da tela “Minha Conta”, com o intuito de agrupar as informações sobre o mesmo e seus respectivos endereços vinculados, conforme apresentado pela figura a seguir.

Figura 2: Protótipo “Minha Conta”.

Em sequência, também foram idealizadas as telas necessárias para criação e manutenção dos perfis de parceiros da plataforma **ShopHero**. Nesse sentido, foram desenvolvidos quatro protótipos referentes às páginas de cadastro, consulta, alteração e remoção de parceiros, apresentados respectivamente pelas imagens abaixo. Tais páginas devem estar disponíveis apenas para uso de administradores da plataforma, os quais atuaram sobre elas conforme as informações concedidas pelos parceiros.

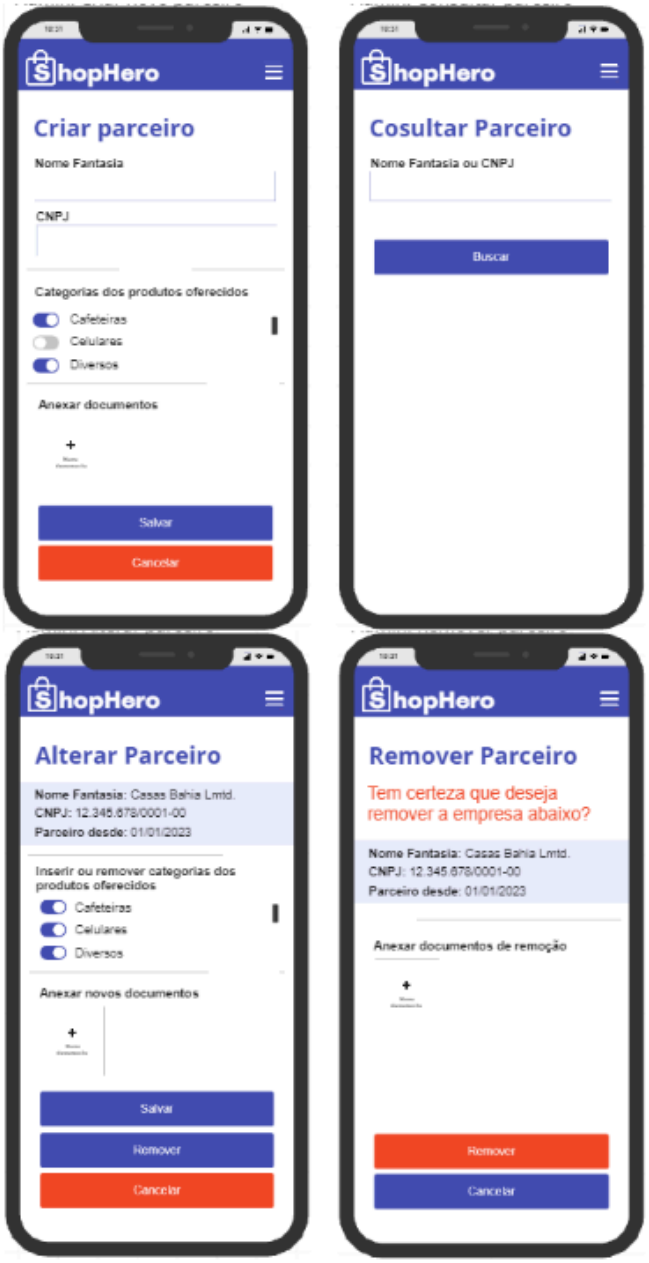


Figura 3: Protótipos “Criar parceiro”, “Consultar parceiro”, “Alterar parceiro” e “Remover parceiro”.

Com intuito de garantir uma navegação consistente dos clientes entres produtos, também foi prototipada a página “Início”. Nesta página serão apresentados os principais produtos em promoção e também seções com os produtos divididos por categorias. Além disso, a página também permite a busca por produtos específicos, acesso ao carrinho de compras e às demais páginas desenvolvidas, conforme apresentado pela Figura 4.

Figura 4: Protótipo “Início”.

Para contemplar a busca de produtos, foram desenvolvidas três telas de protótipo, conforme apresentado pela Figura 5, as quais têm como objetivo garantir a busca via barra de pesquisas, mas também a busca de produtos de um parceiro específico e por categoria. Além disso, os resultados retornados pelas buscas podem ser filtrados e devidamente ordenados, de acordo com as preferências do cliente.

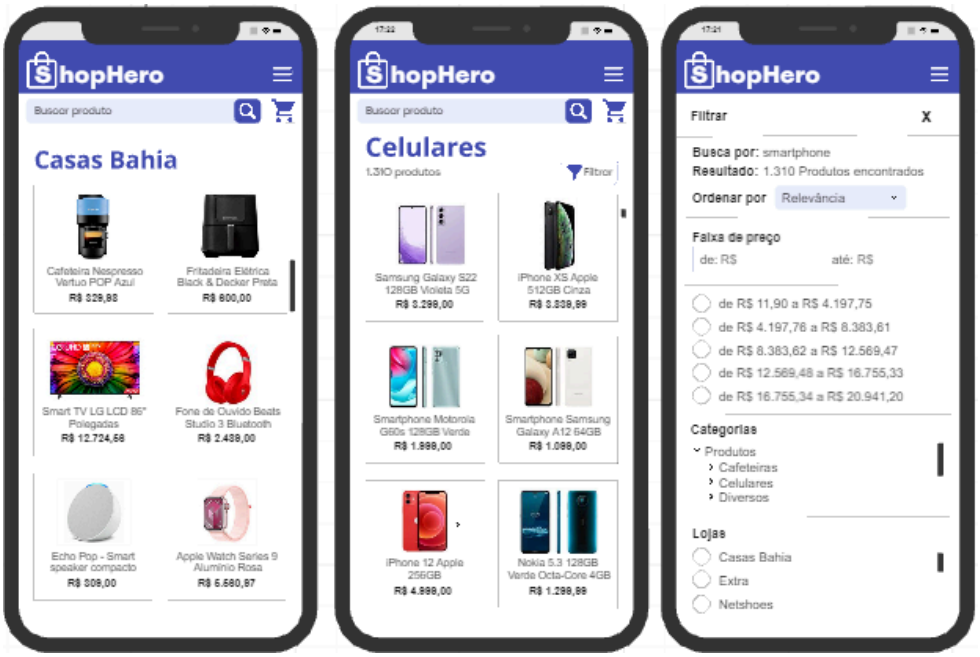


Figura 5: Protótipos “Busca por parceiro”, “Busca por categoria”, “Filtros sobre resultados”.

A fim de apresentar produtos relacionados às compras realizadas pelos clientes, foi prototipada a tela “Recomendações”. Assim, o usuário poderá visualizar com maior facilidade produtos que tenham semelhança com suas preferências e ter maior tendência para novas compras, conforme apresentado pela Figura 6.

Figura 6: Protótipo “Recomendações”.

Consequentemente, ao selecionar um produto, o cliente deve ser direcionado a tela referente ao protótipo “Produto”, apresentado na Figura 7. Essa página garante a visualização de informações sobre o fornecedor, especificações sobre o produto e também permite a visualização das avaliações deixadas por outros clientes, adição do produto à uma lista de desejo e também ao carrinho de compras.

Figura 7: Protótipo “Produto”.

Também foi desenvolvido um protótipo para a tela “Minhas Listas”, responsável por apresentar ao usuário todas as listas de desejo que ele possui e possibilitar a criação de novas listas. Ao clicar sobre alguma das listas, o cliente é direcionado para a tela “Lista de desejo”, para que possa visualizar todos os produtos favoritos e, caso almeje, deletar produtos ou até mesmo deletar a própria lista. Nesse contexto, ambos os protótipos citados são representados pelas figuras abaixo.

Figura 8: Protótipo “Minhas listas” e “Lista de desejo”.

A fim de garantir a experiência do cliente sobre o ato da compra, foi prototipada também a tela para o carrinho de compras. Nesta página, o cliente poderá visualizar os produtos que pretende adquirir, alterar a quantidade de itens de um mesmo produto e excluir itens do carrinho. Além disso, também será possível visualizar o valor da compra, o valor total e solicitar a finalização da compra, para que a solicitação do pedido seja enviada ao parceiro.

Figura 9: Protótipo “Carrinho”.

Além disso, quando uma compra é finalizada, um ou mais pedidos são gerados com os itens do carrinho de compras relativos aos produtos ofertados pelos parceiros. Nesse sentido, a fim de garantir a visualização do cliente sobre os seus pedidos realizados, bem como status e informações sobre os produtos adquiridos, foi prototipada a página “Pedidos”, vide figura abaixo.

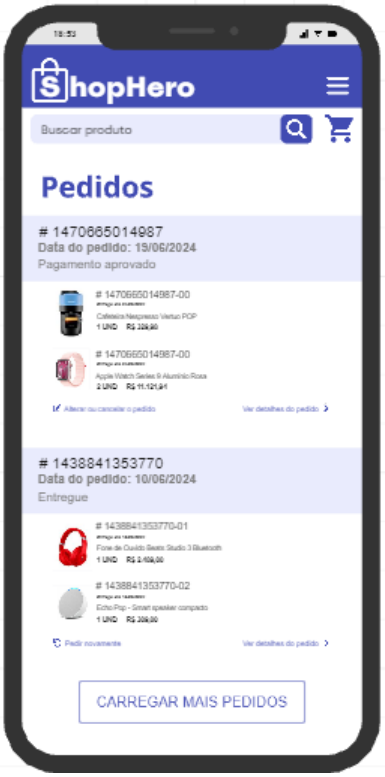


Figura 10: Protótipo “Pedidos”.

Por fim, todo cliente que possuir pedidos marcados com o status “entregue” poderá adicionar uma avaliação sobre todos os produtos adquiridos no pedido em questão, a fim de compartilhar suas experiências sobre a entrega, qualidade e funcionalidade do produto com outros clientes. Para isso, foi prototipada a tela “Avaliações”, acessível através do protótipo “Produto”, a qual é responsável por exibir todas as avaliações que um determinado produto possui, conforme ilustrado pela figura abaixo.

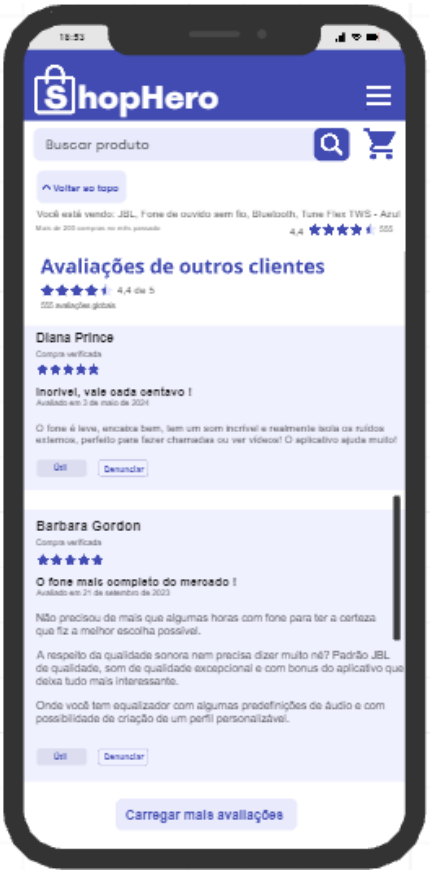


Figura 11: Protótipo “Avaliações”.

Modelo Entidade Relacionamento

A partir das funcionalidades da plataforma descritas nos tópicos anteriores, foram mapeadas as seguintes entidades:

1. **Cliente:** Possui informações relevantes sobre o cliente;
2. **Endereço:** Possui os endereços atrelado à um cliente;
3. **Parceiro:** Informações sobre as empresas parceiras que disponibilizam seus produtos na plataforma **ShopHero**;
4. **Carrinho:** Informações sobre o carrinho de compras atual de cada cliente;
5. **Pedido:** Armazena informações sobre os pedidos de cada cliente;
6. **Produto:** Representa cada produto ofertado pelos parceiros;
7. **Item:** Expressa os itens de um carrinho ou pedido referente aos produtos. Assim, um carrinho pode possuir N itens sobre um mesmo produto, garantindo a identificação de quantidade;
8. **Lista de Desejos:** Agrupa os produtos de desejo de cada cliente;
9. **Avaliação:** Armazena as avaliações cedidas pelos clientes sobre os produtos por ele adquiridos;
10. **Recomendação:** Informações sobre produtos que têm afinidade com o perfil de compra de cada cliente;

Além disso, também foram mapeados os relacionamentos entre as entidades citadas, os quais estão descrito a seguir:

- Cada **Cliente** pode ter 0 à N **Endereços;**
- Cada **Cliente** possui 0 à N **Listas de Desejo;**
- Cada **Cliente** recebe 0 à N **Recomendação**;
- Cada **Cliente** escreve 0 à N **Avaliação**;
- Cada **Cliente** possui 0 à N **Carrinho**;
- Cada **Cliente** possui 0 à N **Pedido**;
- Cada **Produto** pode estar em 0 até N **Recomendação**;
- Cada **Produto** pode estar em 0 até N **Listas de Desejo**;
- Cada **Produto** pode ter 0 à N **Avaliação**;
- Cada **Produto** representa de 0 à N **Item**;
- Cada **Produto** deve ser ofertado por 1 **Parceiro**;
- Cada **Carrinho** converte-se em 0 à N **Pedido**;
- Cada **Carrinho** contém 1 à N **Item**;

- Cada **Lista de Desejo** contém 0 à N **Produto**;
- Cada **Recomendação** contém 0 à N **Produto**;
- Cada **Pedido** deve estar atrelado a 2 **Endereço**: um endereço para **cobrança** e outro para a **entrega**;
- Cada **Endereço** pode estar atrelado de 0 à N **Pedido**;
- Cada **Item** deve estar atrelado a 1 **Carrinho** ou a 1 **Pedido**;
- Cada **Parceiro** pode ofertar de 0 à N **Produto**;

Nesse contexto, o diagrama entidade relacionamento foi modelado, conforme expressado na figura abaixo:

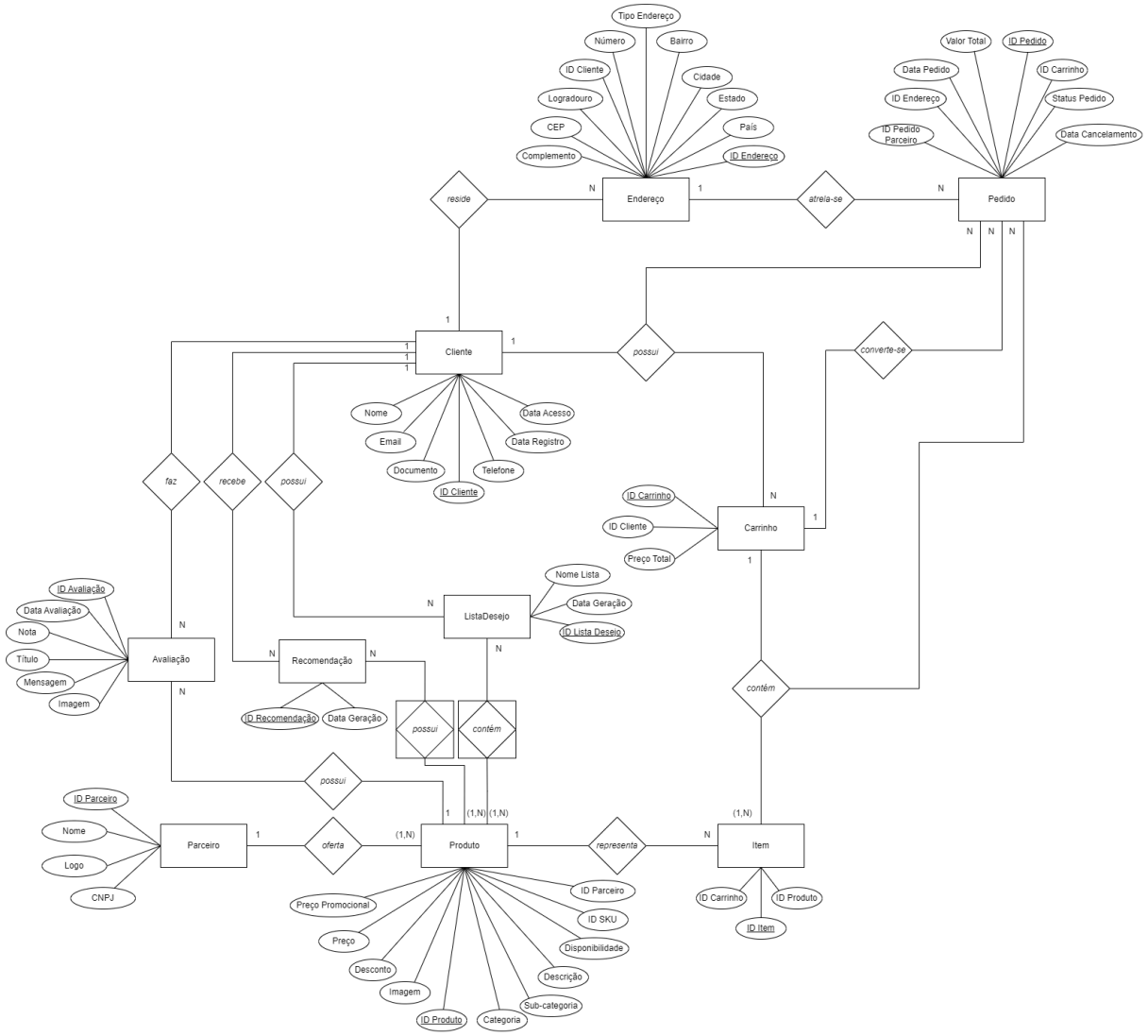


Figura 12: Diagrama Entidade Relacionamento.

Modelo de Agregação

A partir das entidades mapeadas pelo modelo entidade relacionamento e pelo protótipo das telas da plataforma, foram identificadas seis visões ideais para a definição das agregações, sendo estas:

- **Visão Cliente:** Garante que o cliente possa consultar suas informações, bem como seus endereços cadastrados, conforme ilustrado no protótipo da tela "Minha Conta";
- **Visão Lista de Desejos:** Permite consultar as listas de desejo e os produtos atrelados a elas para cada cliente, conforme ilustrado no protótipo da tela "Minhas Listas";
- **Visão Recomendação:** Garante a consulta sobre quais produtos devem ser recomendados a cada cliente, enquanto o mesmo está ativo na plataforma. Nesse sentido, a tela referente a utilização do cliente sobre esse serviço pode ser observada conforme o protótipo "Recomendações";
- **Visão Carrinho:** Garante que o cliente confira seu carrinho de compras atual, bem como os itens nele contidos, conforme apresentado pelo protótipo "Carrinho";
- **Visão Pedido:** Permite que os pedidos sejam consultados, assim como os itens adquiridos e os respectivos endereços de entrega, conforme descrito no protótipo "Pedidos";
- **Visão Produto:** Garante a consulta de detalhes sobre os produtos ofertados, bem como seus respectivos parceiros e a avaliação dos clientes sobre eles, conforme ilustrado pelo protótipo "Produto".

Assim, com base nas visões definidas, é apresentada abaixo a figura com a modelagem de agregação desenvolvida:

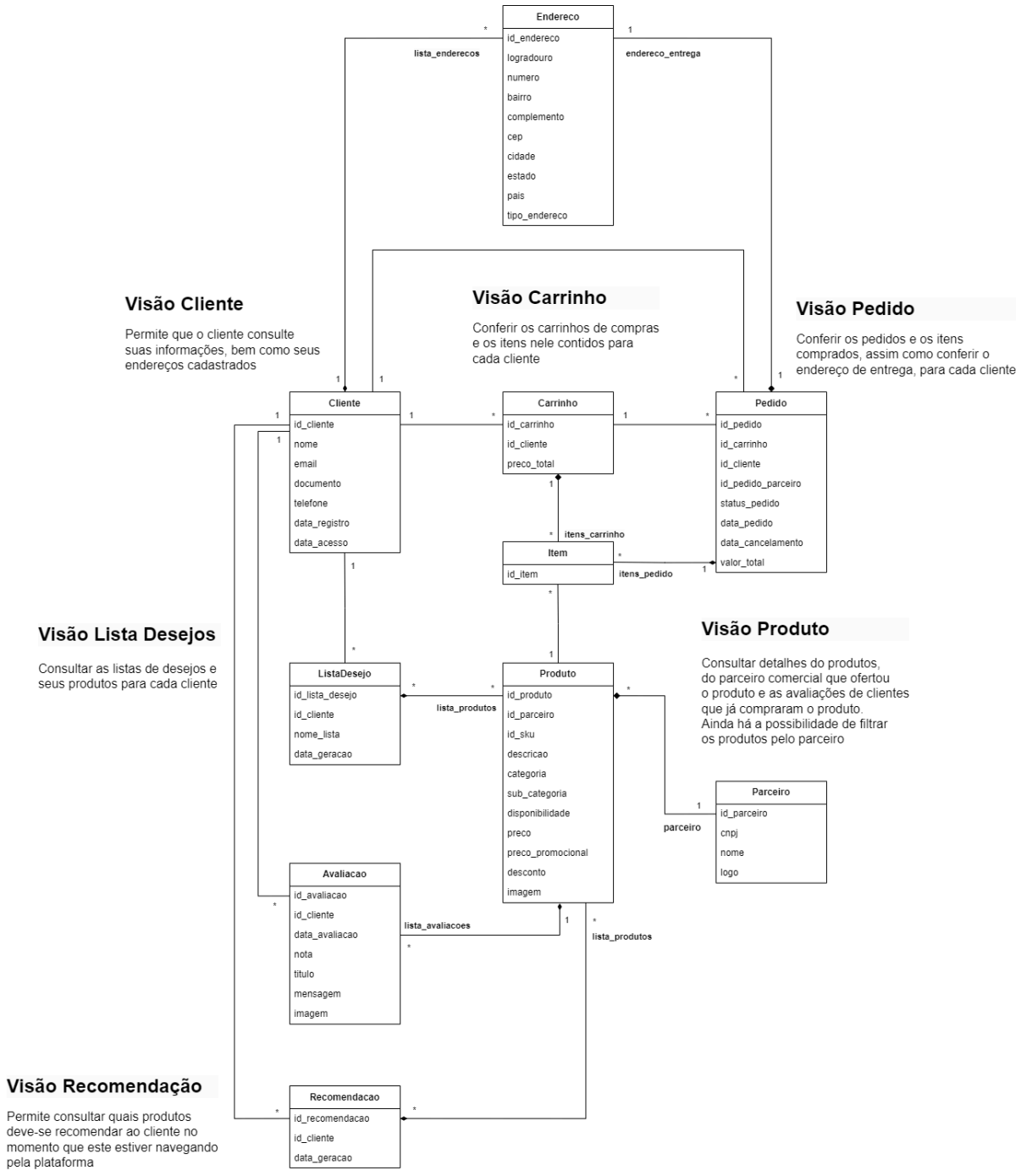


Figura 13: Modelo de Agregação.

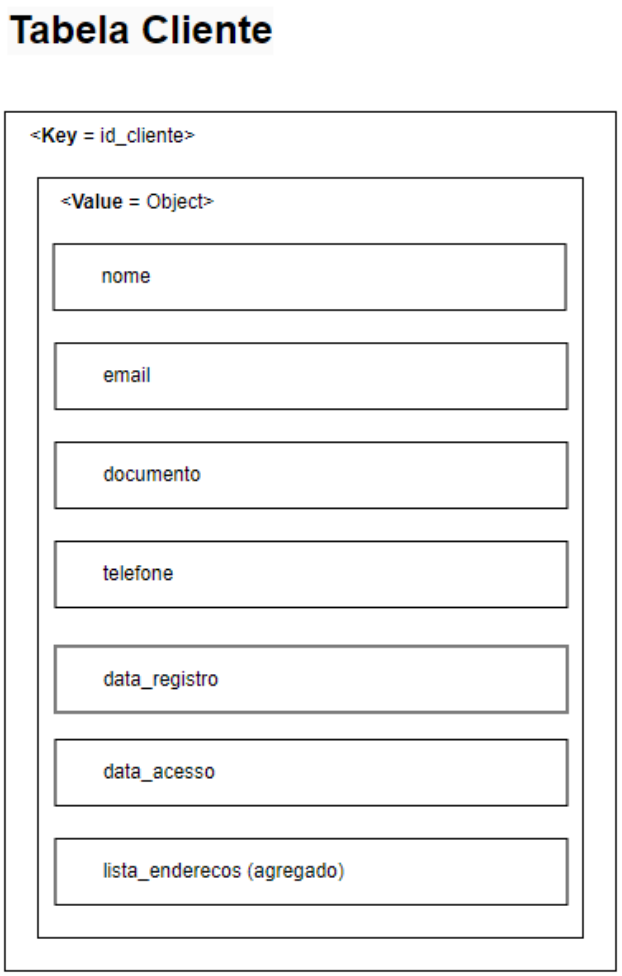
Criação do modelo lógico

Por meio do modelo lógico, descreve-se como as informações serão armazenadas, organizadas e acessadas, assegurando que todas as entidades e seus relacionamentos atendam aos requisitos funcionais e de desempenho da aplicação. Nesse contexto, abaixo serão detalhadas as modelagens das principais tabelas e seus atributos, bem como as estruturas JSON correspondentes, que ilustram a implementação lógica dos dados.

Cliente

A tabela Cliente representa a estrutura de um cliente, contendo informações detalhadas tanto do cliente quanto de seus endereços. Sua chave primária é representada pelo atributo *id_cliente*.

Nesse contexto, o diagrama da tabela e JSON correspondente são apresentados abaixo:



```
{
    "id_cliente": "",
    "nome": "",
    "email": "",
    "documento": "",
    "telefone": "",
    "data_registro": "",
    "data_acesso": "",
    "lista_enderecos": [
        {
            "id_endereco": "",
            "logradouro": "",
            "numero": "",
            "bairro": "",
            "complemento": "",
            "cep": "",
            "cidade": "",
            "estado": "",
            "pais": "",
            "tipo_endereco": ""
        }
    ]
}
```

Pedido

A tabela Pedido representa a estrutura de um pedido, detalhando informações tanto do pedido em si quanto dos itens incluídos, avaliações de clientes, e o endereço de entrega. Sua chave primária é o atributo *id_pedido*, porém a fim de possibilitar consultas rápidas para recuperar todos os pedidos de um cliente específico ou ainda de um carrinho específico, foram estabelecidos os atributos *id_cliente* e *id_carrinho* como sendo índices secundários globais (GSI). Ainda, o atributo *data_pedido* foi utilizado como chave de ordenação para o índice *id_cliente*.

Nesse contexto, o diagrama da tabela e JSON correspondente são apresentados abaixo:

Tabela Pedido


```
{
    "id_pedido": "",
    "id_carrinho": "",
    "id_cliente": "",
    "id_pedido_parceiro": "",
    "status_pedido": "",
    "data_pedido": "",
    "data_cancelamento": "",
    "valor_total": 0,
    "itens_pedido": [
```

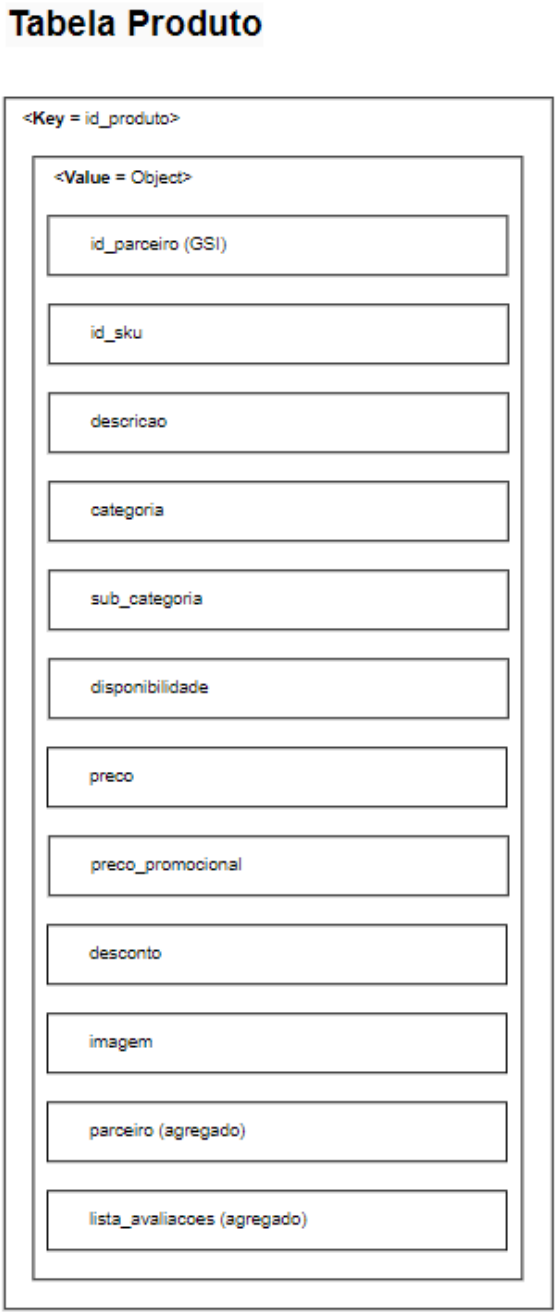
```
        {
            "id_item": "",
            "id_produto": "",
            "id_parceiro": "",
            "id_sku": "",
            "descricao": "",
            "categoria": "",
            "sub_categoria": "",
            "disponibilidade": "",
            "preco": "",
            "preco_promocional": "",
            "desconto": "",
            "imagem": "",
            "parceiro": {
                "id_parceiro": "",
                "cnpj": "",
                "nome": "",
                "logo": ""
            },
            "lista_avaliacoes": [
                {
                    "id_avaliacao": "",
                    "id_cliente": "",
                    "data_avaliacao": "",
                    "nota": "",
                    "titulo": "",
                    "mensagem": "",
                    "imagem": ""
                }
            ]
        }
    ],
    "endereco_entrega": {
        "id_endereco": "",
        "logradouro": "",
        "numero": "",
        "bairro": "",
        "complemento": "",
        "cep": "",
        "cidade": "",
        "estado": "",
        "pais": "",
        "tipo_endereco": ""
    }
}
```

Produto

A tabela Produto representa a estrutura de um produto, detalhando informações tanto do produto quanto do parceiro que o fornece e das avaliações

feitas pelos clientes. O atributo *id_produto* é sua chave primária; contudo, para realizar consultas rápidas a fim de recuperar todos os produtos fornecidos por um parceiro específico, foi estabelecido como GSI o atributo *id_parceiro*, em que o atributo *preco* foi utilizado como chave de ordenação.

Nesse contexto, o diagrama da tabela e JSON correspondente são apresentados abaixo:



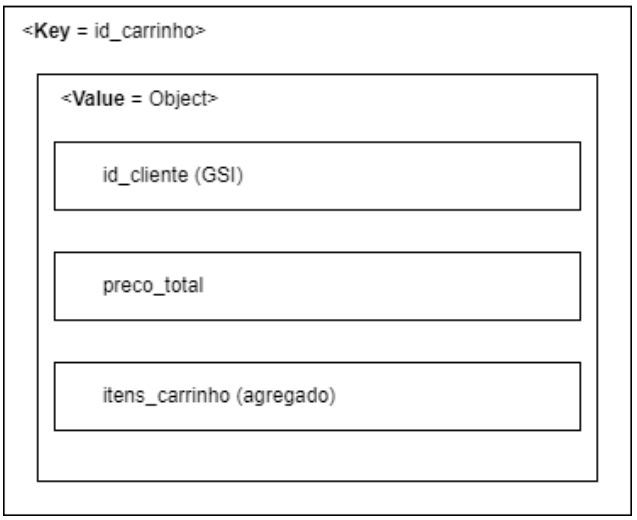
```
{
    "id_produto": "",
    "id_parceiro": "",
    "id_sku": "",
    "descricao": "",
    "categoria": "",
    "sub_categoria": "",
    "disponibilidade": "",
    "preco": 0,
    "preco_promocional": 0,
    "desconto": 0,
    "imagem": "",
    "parceiro": {
        "id_parceiro": "",
        "cnpj": "",
        "nome": "",
        "logo": ""
    },
    "lista_avaliacoes": [
        {
            "id_avaliacao": "",
            "id_cliente": "",
            "data_avaliacao": "",
            "nota": "",
            "titulo": "",
            "mensagem": "",
            "imagem": ""
        }
    ]
}
```


Carrinho

A tabela Carrinho representa a estrutura de um carrinho de compras, incluindo informações sobre o carrinho, o cliente associado e os itens contidos no carrinho. Sua chave primária é o atributo *id_carrinho*, e para consultar rapidamente todos os carrinhos de compras associados a um cliente específico, foi estabelecido o atributo *id_cliente* como GSI.

Nesse contexto, o diagrama da tabela e JSON correspondente são apresentados abaixo:

Tabela Carrinho




```
{
    "id_carrinho": "",
    "id_cliente": "",
    "preco_total": 0,
    "itens_carrinho": [
        {
            "id_item": "",
            "produto": {
                "id_produto": "",
                "id_parceiro": "",
                "id_sku": "",
                "descricao": "",
                "categoria": "",
                "sub_categoria": "",
                "disponibilidade": "",
                "preco": 0,
                "preco_promocional": 0,
                "desconto": 0,
                "imagem": "",
                "parceiro": {
                    "id_parceiro": "",
                    "cnpj": "",
                    "nome": "",
                    "logo": ""
                },
                "lista_avaliacoes": [
                    {
                        "id_avaliacao": "",
                        "id_cliente": "",
                        "data_avaliacao": "",
                        "nota": "",
                        "titulo": "",
```

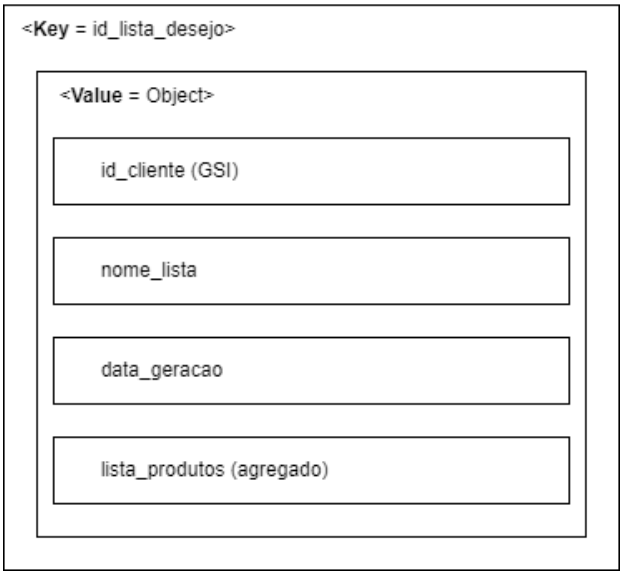
```
                        "mensagem": "",
                        "imagem": ""
                    }
                ]
            }
        }
    ]
}
```

ListaDesejo

A tabela ListaDesejo representa a estrutura de uma lista de desejos criada pelo cliente. São incluídas informações sobre a lista de desejos, o cliente associado e os produtos contidos na lista. O atributo *id_lista_desejo* é a sua chave primária, e para recuperar rapidamente todas as listas de desejo associadas a um cliente específico, foi estabelecido o atributo *id_cliente* como GSI, em que o atributo *data_geracao* foi atribuído como chave de ordenação.

Nesse contexto, o diagrama da tabela e JSON correspondente são apresentados abaixo:

Tabela ListaDesejo



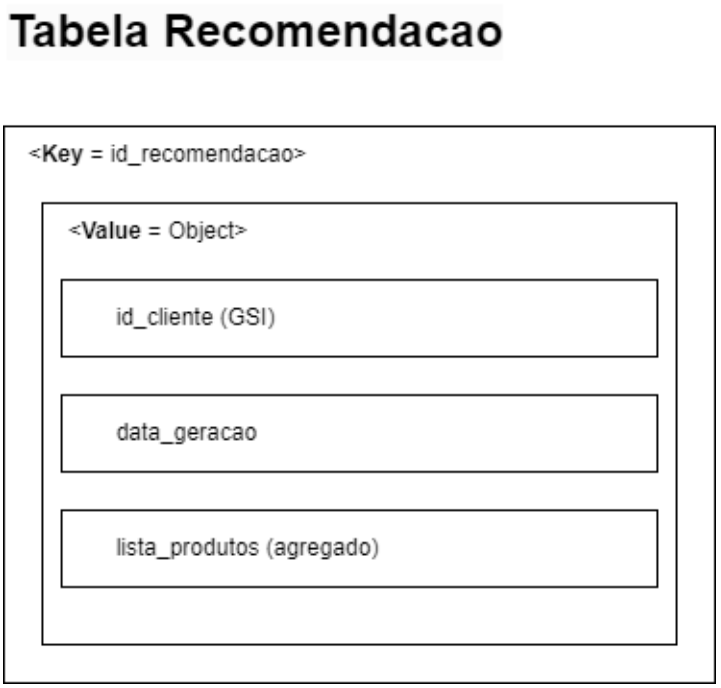
```
{
    "id_lista_desejo": "",
    "id_cliente": "",
    "nome_lista": "",
    "data_geracao": "",
    "lista_produtos": [
        {
            "id_produto": "",
            "id_parceiro": "",
            "id_sku": "",
            "descricao": "",
            "categoria": "",
            "sub_categoria": "",
            "disponibilidade": "",
            "preco": 0,
            "preco_promocional": 0,
            "desconto": 0,
            "imagem": "",
            "parceiro": {
                "id_parceiro": "",
                "cnpj": "",
                "nome": "",
                "logo": ""
            },
            "lista_avaliacoes": [
                {
                    "id_avaliacao": "",
                    "id_cliente": "",
                    "data_avaliacao": "",
                    "nota": "",
                    "titulo": "",
                    "mensagem": "",
                    "imagem": ""
                }
            ]
        }
    ]
}
```


Recomendação

A tabela Recomendação representa as recomendações personalizadas, armazenando recomendações específicas para cada cliente com base em seu histórico de compras, preferências e comportamento de navegação. Como chave primária foi estabelecido o atributo *id_recomendacao*, e a fim de recuperar

rapidamente todas as recomendações associadas a um cliente específico, foi estabelecido o atributo *id_cliente* como GSI, em que o atributo *data_geracao* foi atribuído como chave de ordenação.

Nesse contexto, o diagrama da tabela e JSON correspondente são apresentados abaixo:




```
{
    "id_recomendacao": "",
    "id_cliente": "",
    "data_geracao": "",
    "lista_produtos": [
        {
            "id_produto": "",
            "id_parceiro": "",
            "id_sku": "",
            "descricao": "",
            "categoria": "",
            "sub_categoria": "",
            "disponibilidade": "",
            "preco": 0,
            "preco_promocional": 0,
            "desconto": 0,
            "imagem": "",
            "parceiro": {
                "id_parceiro": "",
                "cnpj": "",
                "nome": "",
```

```
                "logo": ""
            },
            "lista_avaliacoes": [
                {
                    "id_avaliacao": "",
                    "id_cliente": "",
                    "data_avaliacao": "",
                    "nota": "",
                    "titulo": "",
                    "mensagem": "",
                    "imagem": ""
                }
            ]
        }
    ]
}
```

Criação do modelo físico

Em busca de uma melhor organização do trabalho, adotou-se um repositório no GitHub para versionar todos os códigos referentes à criação de recursos AWS, assim como a integração com estes recursos via linguagem de programação Python. Como dito anteriormente, o conteúdo dos códigos na íntegra estão disponibilizados no repositório GitHub Squinelato / eEDB-016-2024-2.

Para trazer à materialidade o modelo lógico em um modelo físico o qual será capaz de armazenar dados fisicamente utilizou-se o recurso de CloudFormation da AWS. Este recurso permite com que crie-se um conjunto de recursos, indexando-os em um mesmo grupo (*stack*) de recursos, o que facilita sua criação, replicação, atualização, exclusão e até migração para outras contas AWS.

Para então utilizar o CloudFormation, é necessário que utilizemos um arquivo de extensão JSON ou YAML para definir quais recursos gostaríamos de agrupar e criar na AWS. Esta forma de criação de recursos é conhecida como Infraestrutura como código (IaaC).

Logo, para materializar nossas seis tabelas lógicas supracitadas, criamos um arquivo YAML definindo as tabelas DynamoDB, vide definição da tabela *ListaDesejo*:

```yaml
TabelaListaDesejo:
  Type: AWS::DynamoDB::Table
  Properties:
    TableName: ListaDesejo
    AttributeDefinitions:
      - AttributeName: "id_lista_desejo"
        AttributeType: "S"
      - AttributeName: "id_cliente"
        AttributeType: "S"
      - AttributeName: "data_geracao"
        AttributeType: "S"
    KeySchema:
      - AttributeName: "id_lista_desejo"
        KeyType: "HASH"
    ProvisionedThroughput:
      ReadCapacityUnits: 5
      WriteCapacityUnits: 5
    GlobalSecondaryIndexes:
```

```
      - IndexName: "gsi_cliente"
        KeySchema:
          - AttributeName: "id_cliente"
            KeyType: "HASH"
          - AttributeName: "data_geracao"
            KeyType: "RANGE"
        Projection:
          ProjectionType: "ALL"
        ProvisionedThroughput:
          ReadCapacityUnits: 5
          WriteCapacityUnits: 5
    Tags:
      - Key: "Projeto"
        Value: "Repositorio de Dados e NoSQL"
```

Com o arquivo YAML é possível atribuir um nome à tabela, definir quais serão os índices, além de definir quais destes índices serão chave de partição ou GSI. Ademais, é possível escolher o tipo de chave *HASH* ou *RANGE* (ordenação).

Outra propriedades que podemos definir são a capacidade de leitura e a capacidade de escrita de nossas tabelas DynamoDB, tanto para a tabela principal quanto para a gerada em virtude dos índices secundários globais.

Ainda quanto à tabela gerada via GSI, podemos especificar quais atributos desejos que sejam copiados. Em questão, escolhemos *ALL* para termos todos os campos em nossas consultas. Por fim, como boa prática, criamos *tags* para rastrear os custos decorridos da criação e utilização destas tabelas DynamoDB.

Assim, acessando o console AWS e indo até a aba de CloudFormation na opção de criar nova *stack*, importarmos nosso arquivo YAML e demos um nome para esta *stack*, criando assim nossas seis tabelas DynamoDB: *Carrinho*, *Cliente*, *ListaDesejo*, *Pedido*, *Produto* e *Recomendacao*.

Nesse contexto, a stack de recursos criados no CloudFormation, pode ser visualizada pela figura a seguir.

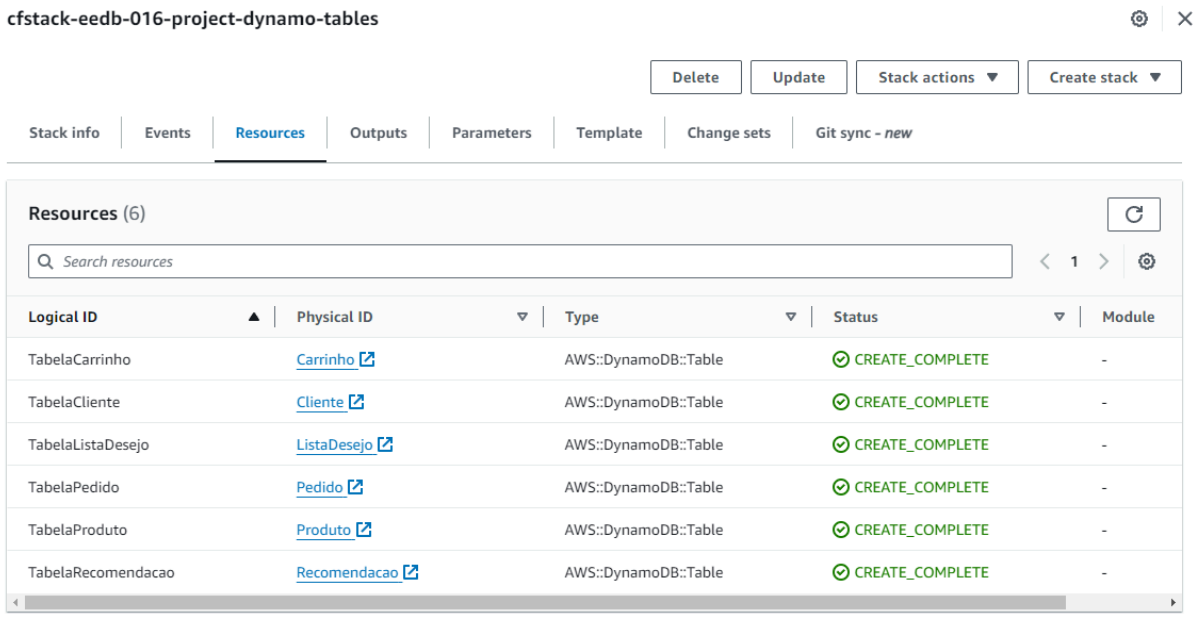


Figura 14: *Stack* cfstack-eedb-o16-project-dynamo-tables.

Além disso, a criação das tabelas por meio da *stack* de recursos também pode ser validada verificando a existência das tabelas na aba do recurso DynamoDB no console AWS, conforme elucidado pela imagem abaixo.

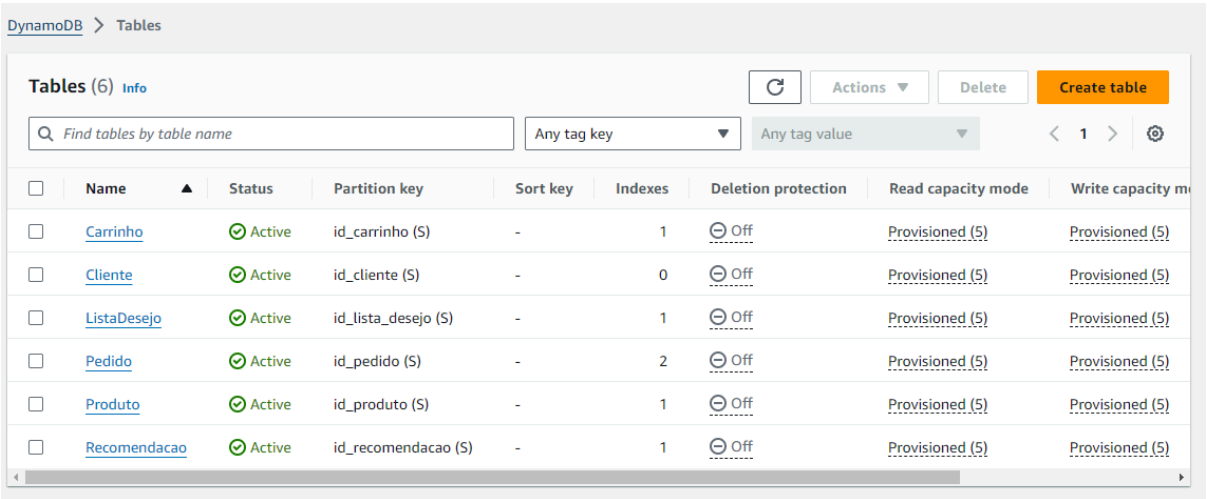


Figura 15: Tabelas DynamoDB criadas.

Como consequência, as tabelas DynamoDB podem ser acessadas individualmente, possibilitando a visualização de suas respectivas definições, conforme o exemplo abaixo sobre a tabela *ListaDesejo*.

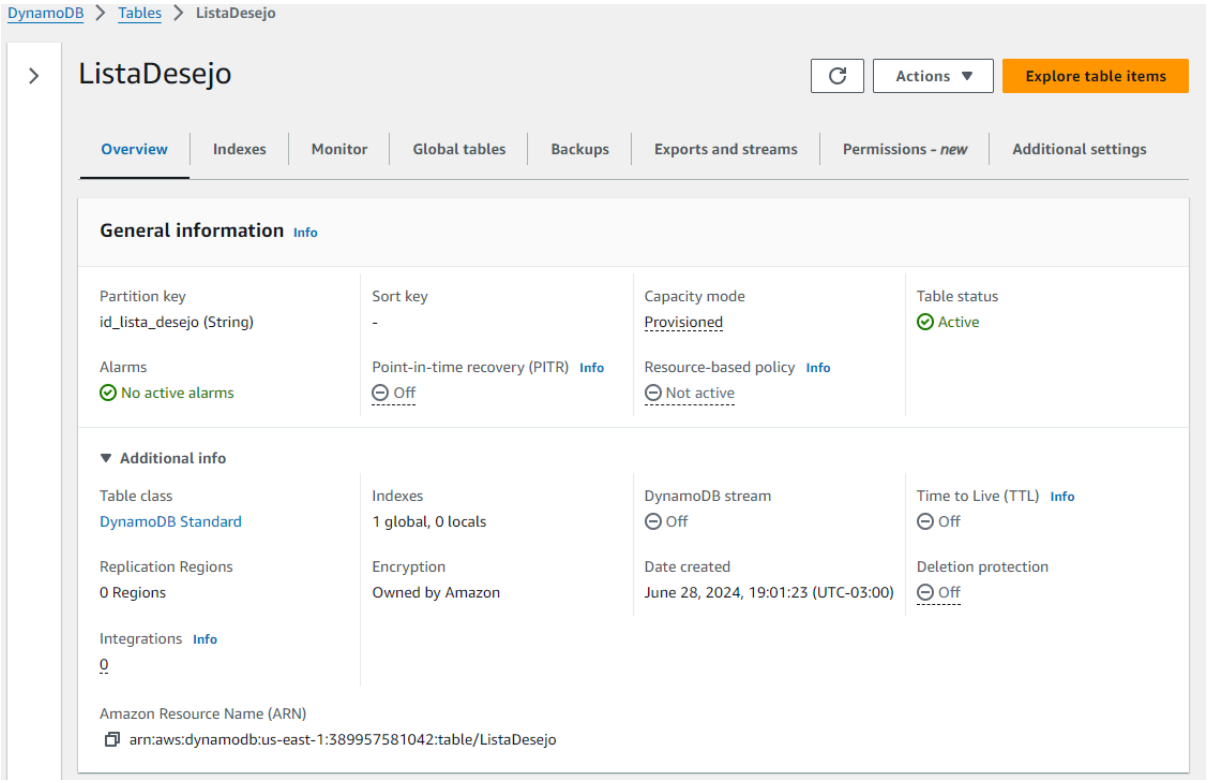


Figura 16: Tabela ListaDesejo.

Por fim, ainda sobre a tabela *ListaDesejo*, na coluna "*Indexes*" é possível validar a correta criação do índice secundário global, conforme apresentado pela Figura 17.

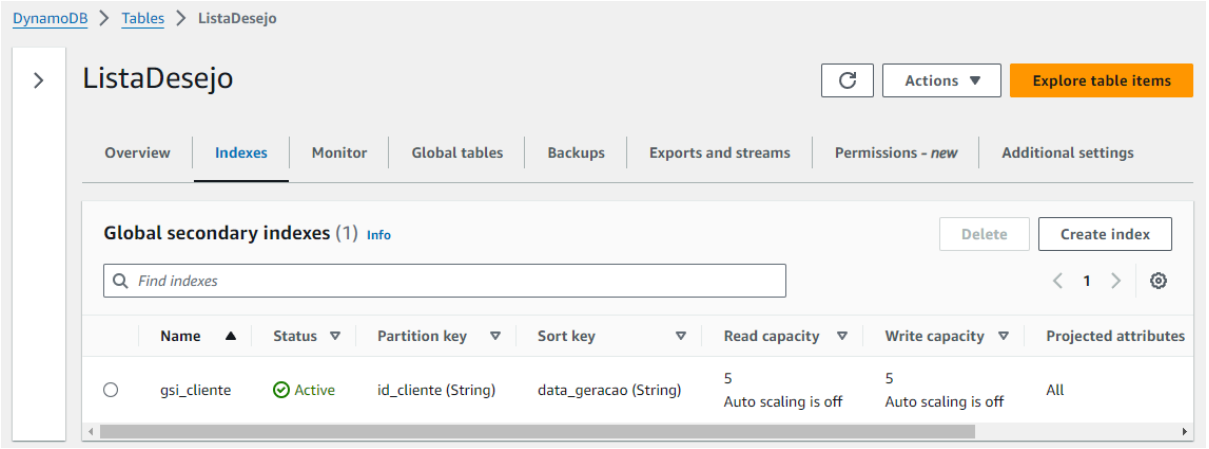


Figura 17: Validação de índice secundário global.

População de dados

Para a geração de dados artificiais, baseamo-nos nos modelos JSON criados na sessão de Criação do modelo lógico. Além disso, a ferramenta de Inteligência Artificial Generativa ChatGPT da empresa OpenIA foi utilizada para auxiliar no processo de geração de dados fictícios. Todavia, uma curadoria humana foi necessária para que os dados gerados fossem verossímeis com a proposta deste projeto.

Isto posto, os objetos com dados fictícios foram agrupados em uma lista e armazenados dentro de um arquivo do tipo JSON. Para cada tabela DynamoDB, um arquivo JSON homônimo foi criado e armazenado na pasta dados/ em nosso repositório GitHub.

Já para popular as tabelas DynamoDB, utilizamos a linguagem de programação Python e definimos o código *insert_dynamodb.py* para realizar a inserção em *batch* de itens nas seis tabelas criadas na sessão anterior.

```python
from typing import NoReturn
from decimal import Decimal
from more_itertools import batched

import argparse
import boto3
import json

def insert(table_name: str, file_path: str) -> NoReturn:
    """
    Insert a list of JSON items into a DynamoDB table using the batch
    write operator in chunks of 25 items each.

    Keyword arguments:
    table_name -- The name of the DynamoDB table
    file_path -- the path to the file containing a JSON list of items
    """
    print(f"Creating DynamoDB client")
    dynamodb = boto3.resource("dynamodb", region_name="us-east-1")
    table = dynamodb.Table(table_name)

    print(f"Opening file '{file_path}'")
    with open(file_path, 'r', encoding='utf-8') as file:
```

```
        json_list = json.load(file, parse_float=Decimal)

        batches = batched(json_list, 25)
        print("Splitting items into batches of 25 items")

        for index, batch in enumerate(batches):

            print(f"Inserting {index+1}º batch with {len(batch)} items")

            with table.batch_writer() as batch_writer:

                print(f"Inserting items into '{table_name}' table: ", end="")
                for item in batch:
                    print(".", end="")
                    batch_writer.put_item(Item=item)

        print(f"\nItems inserted successfully!")

if __name__ == "__main__":

    parser = argparse.ArgumentParser()
    parser.add_argument('-t', '--table', required=True)
    parser.add_argument('-f', '--file', required=True)
    args = parser.parse_args()

    table_name = args.table
    file_path = args.file

    insert(table_name, file_path)
```

Inicialmente, o código se encarrega de criar um cliente com a API do DynamoDB na AWS para que possamos carregar nossos dados nas tabelas. Em seguida, o código acessa o arquivo JSON contendo uma lista de objetos a serem inseridos em uma determinada tabela. Após carregada em memória, essa lista é subdivida em listas (*batches*) menores de até 25 itens, os quais são salvos na tabela DynamoDB especificada. Essa limitação de 25 itens deve ser respeitada para que não se ultrapasse o limite de escrita do DynamoDB.

Dito isso, para utilizar este código basta abrir um terminal, com a linguagem Python devidamente instalada em seu ambiente, e então executar o comando abaixo especificando a tabela que deseja inserir os dados, assim como o caminho para o arquivo JSON que contém os dados a serem inseridos:

```
$ python insert_dynamodb.py --table Cliente --file dados/Cliente.json
```

Logo, para popular todas as seis tabelas invocou-se o código de inserção para cada uma delas, especificando os respectivos arquivos JSON.

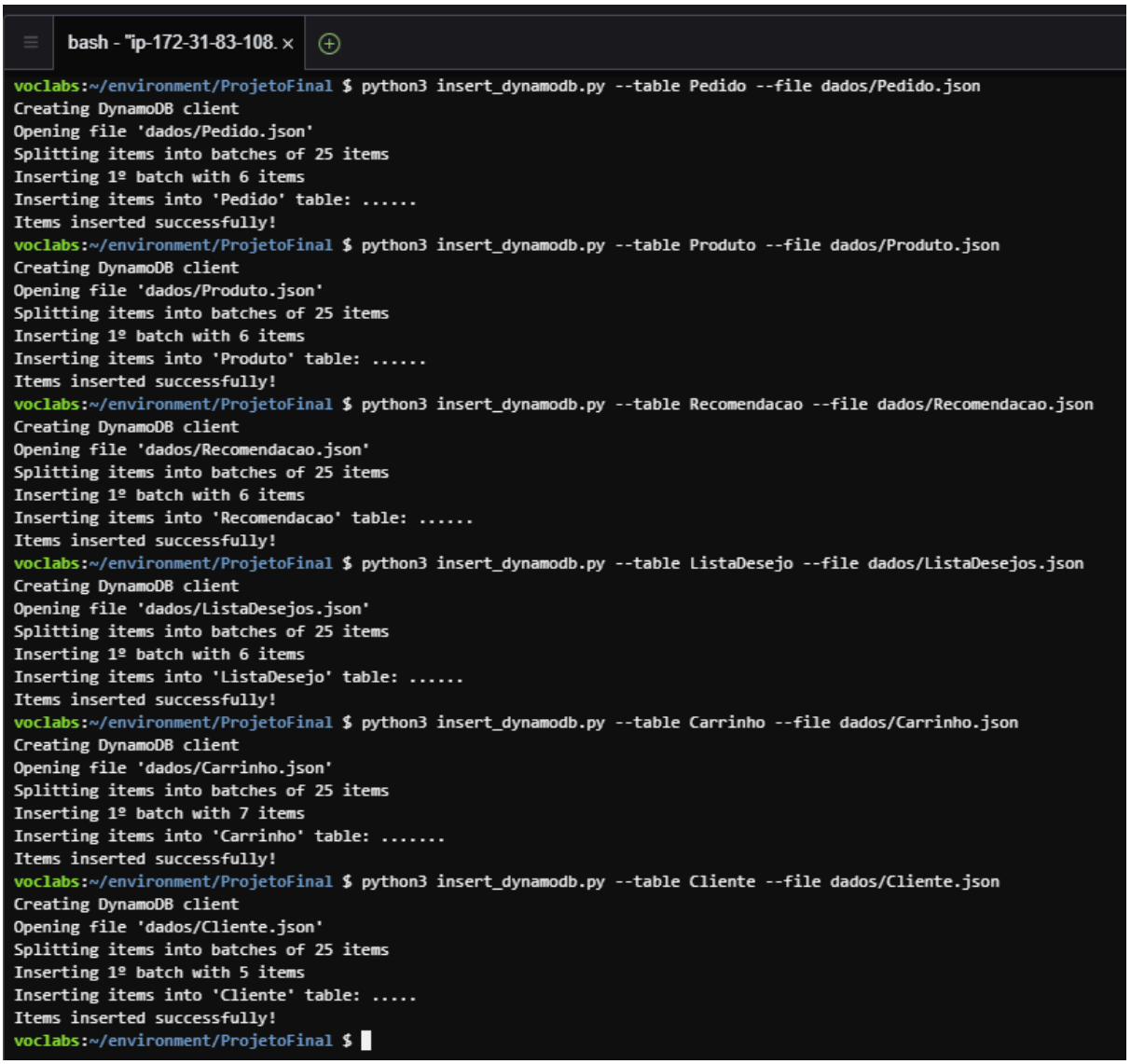


Figura 18: Inserção de dados nas tabelas DynamoDB.

Para validar se os itens foram devidamente salvos nas tabelas DynamoDB, basta utilizarmos o console AWS e escanear as tabelas, vide imagem abaixo:

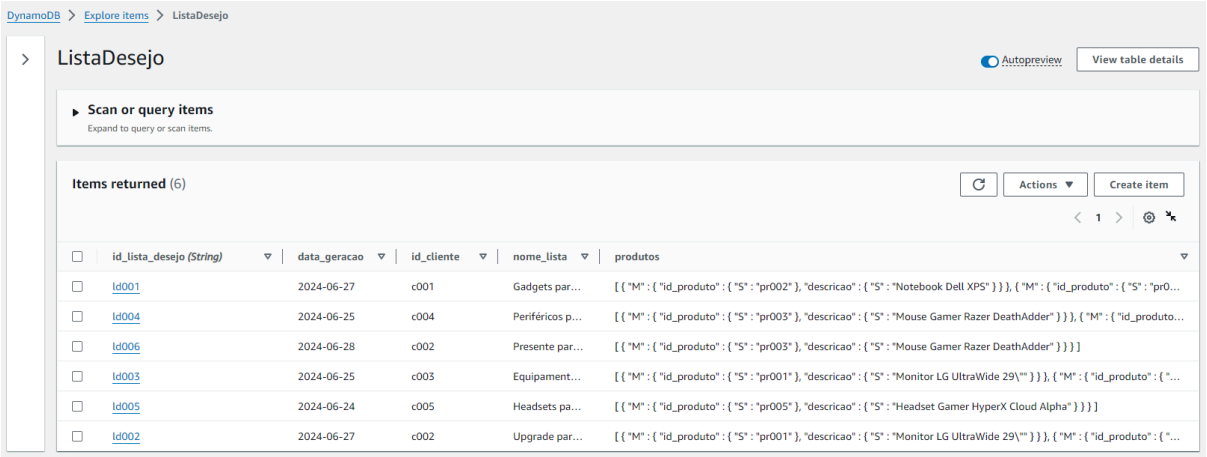


Figura 19: Itens salvos na tabela *ListaDesejo*.

Consultas

A plataforma ShopHero foi projetada para oferecer uma experiência de usuário eficiente. A seguir, são apresentados exemplos de consultas que demonstram como os dados podem ser acessados de forma rápida e organizada. Essas consultas cobrem diversos cenários de uso comuns na plataforma.

A primeira consulta realizada recupera todas as listas de desejos dos últimos três meses de um cliente específico, ordenando-as por suas respectivas datas de geração:

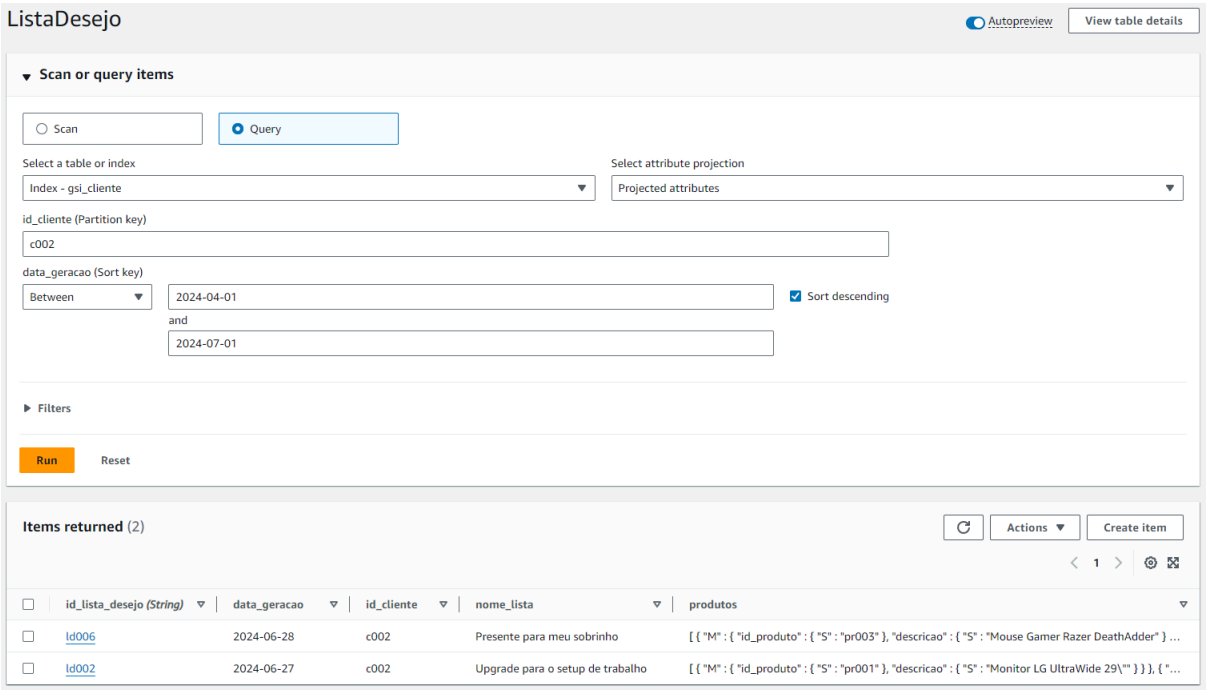


Figura 20: Consulta sobre a lista de desejos.

A segunda consulta recupera todas as recomendações de produtos geradas para um cliente específico, ordenando-as por suas respectivas datas de geração. Apenas a recomendação mais recente será exibida para o cliente:

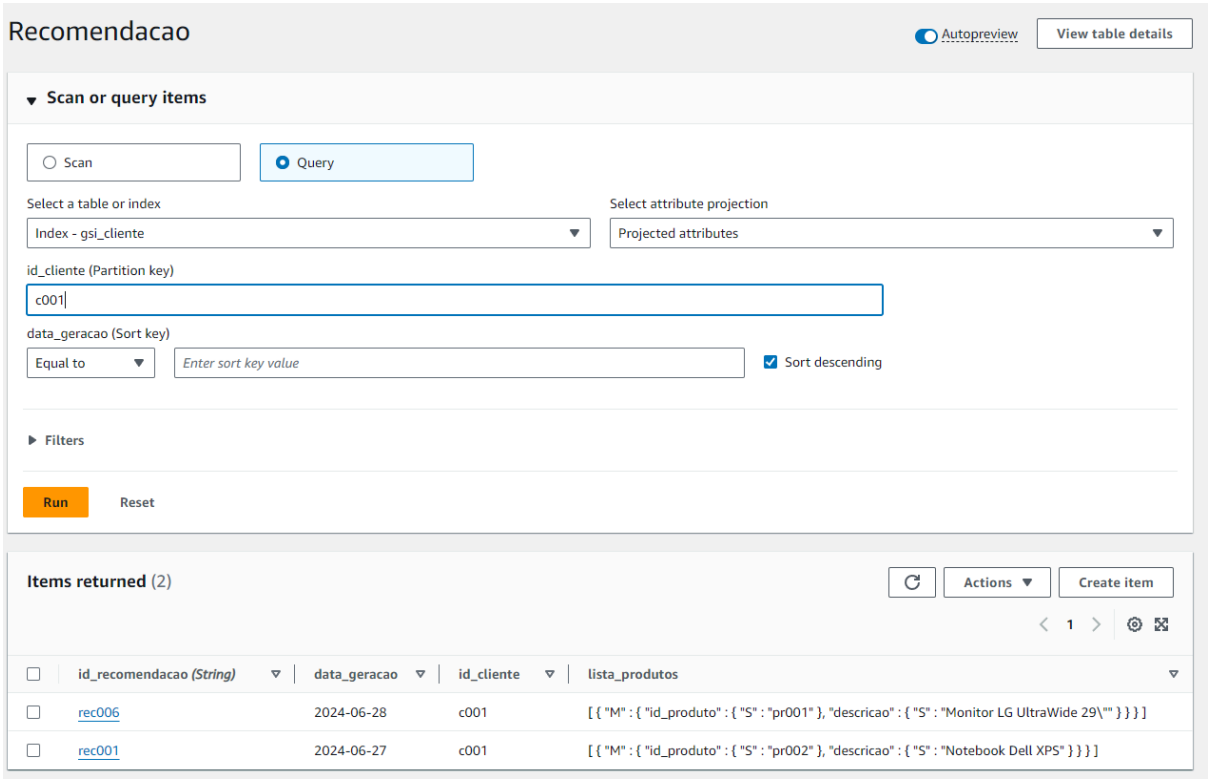


Figura 21: Consulta sobre recomendações.

A terceira consulta permite que um cliente visualize o conteúdo do seu carrinho de compras atual:

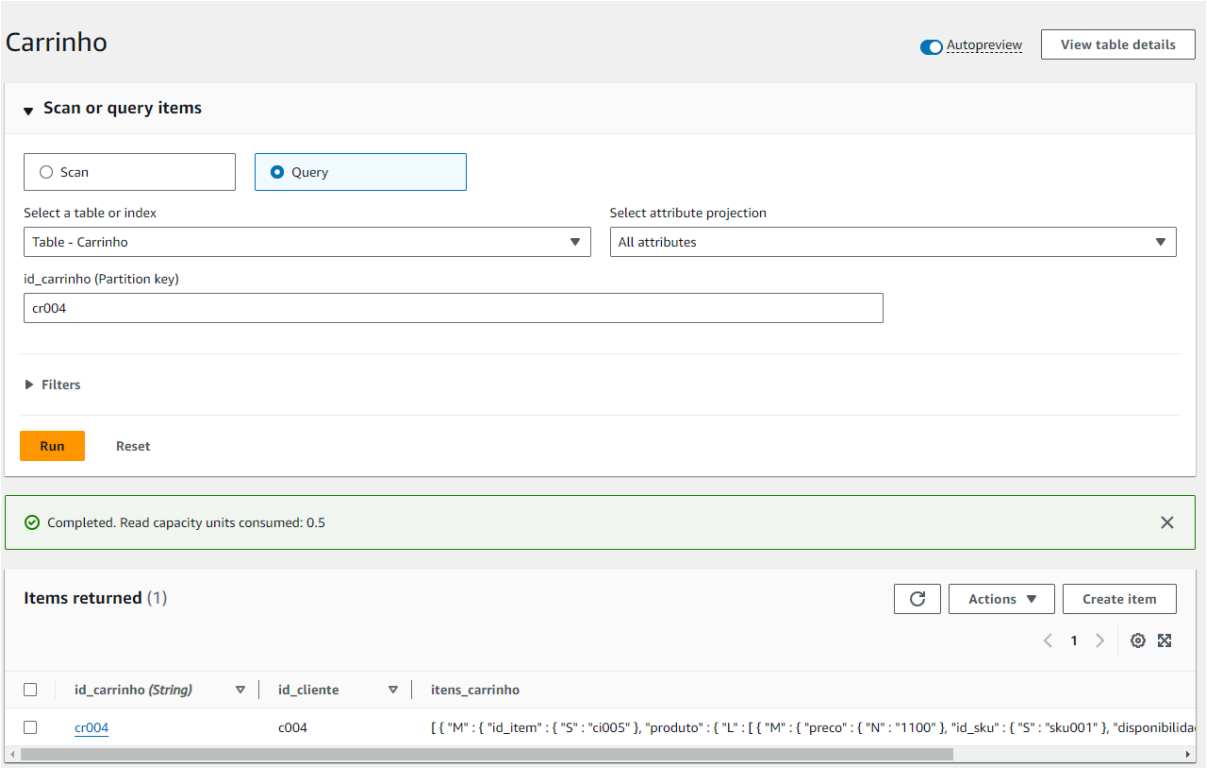


Figura 22: Consulta sobre carrinho de compras.

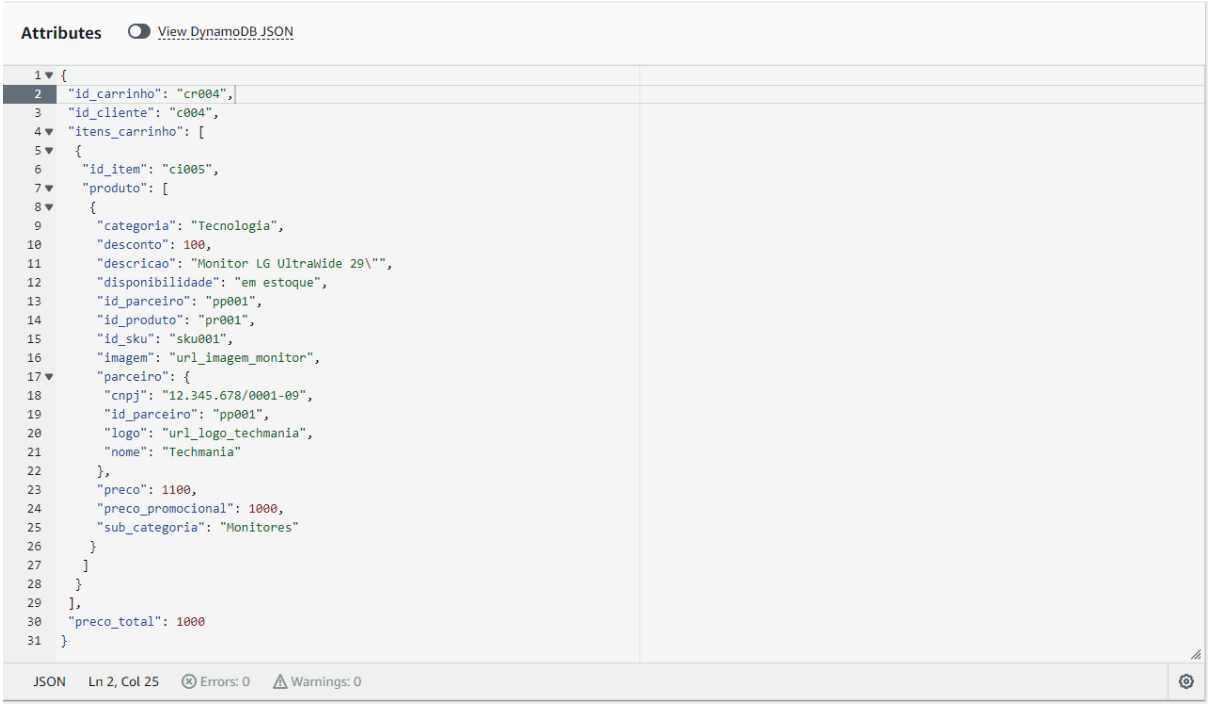


Figura 23: Atributos sobre o carrinho de compras retornado.

A quarta consulta permite que o cliente visualize suas informações pessoais, incluindo detalhes de contato e endereços associados:

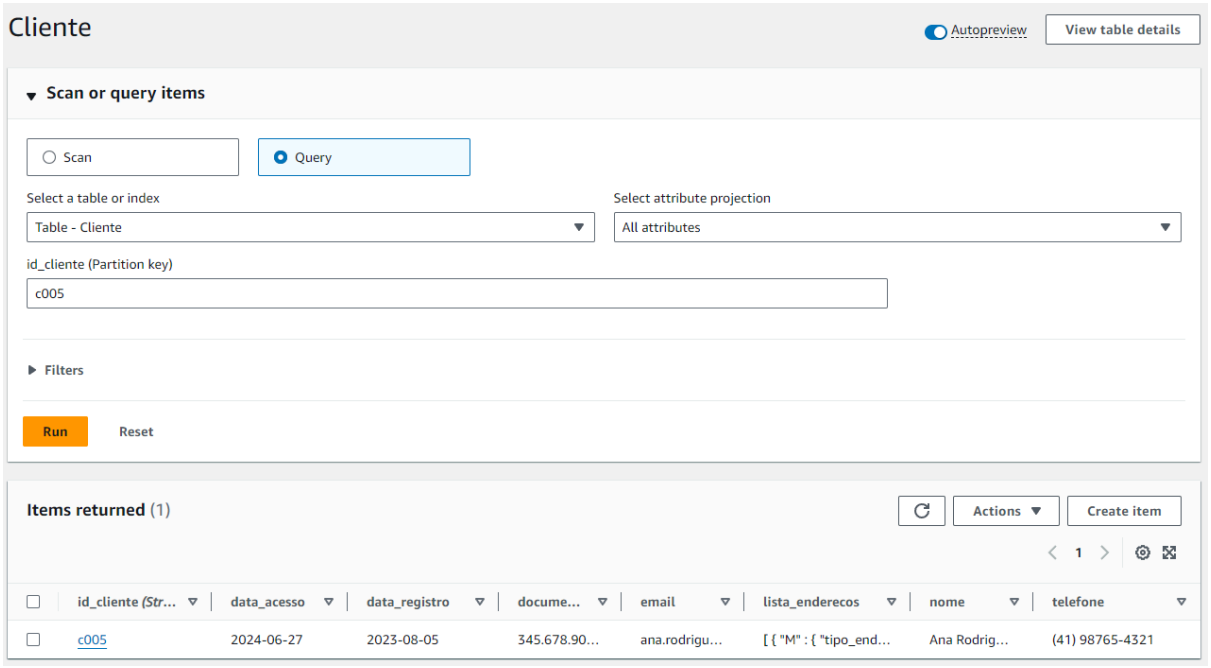


Figura 24: Consulta sobre informações do cliente.

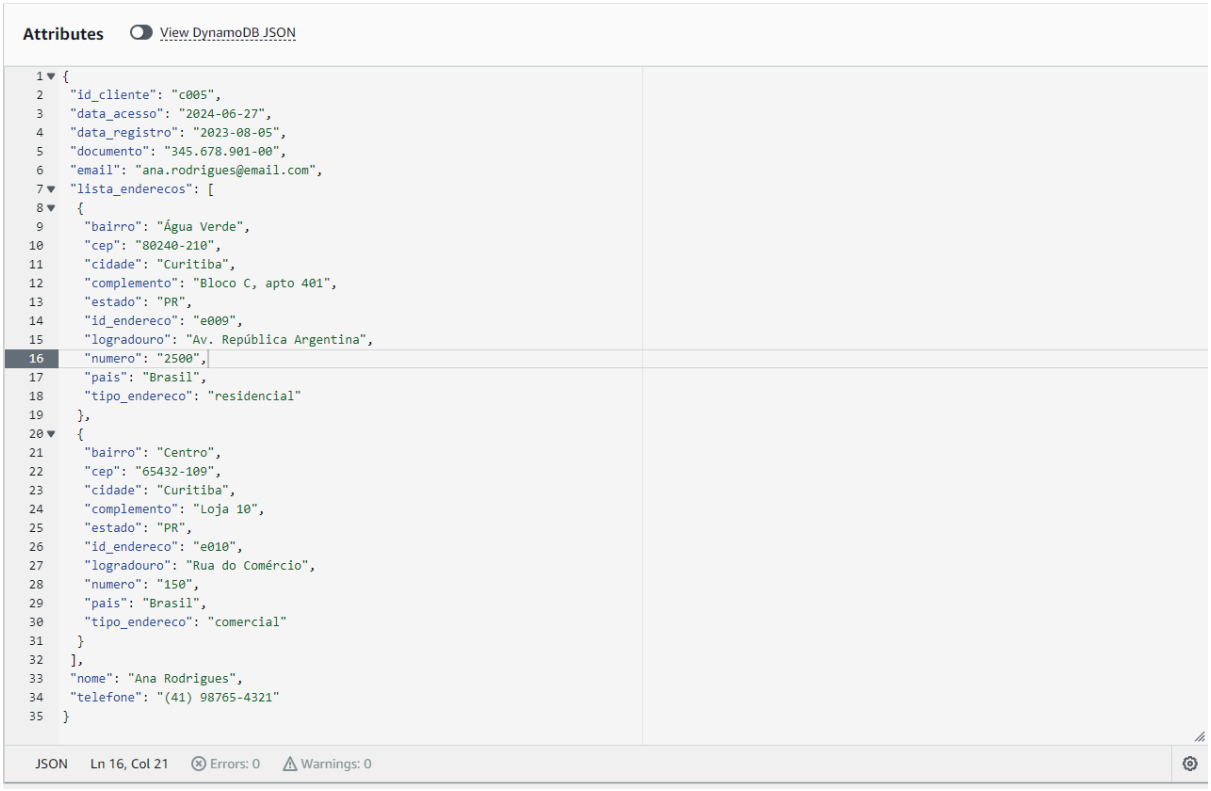


Figura 25: Atributos sobre o cliente retornado.

A quinta consulta recupera todos os produtos oferecidos por um parceiro específico, ordenados pelo preço do produto mais barato para o mais caro:

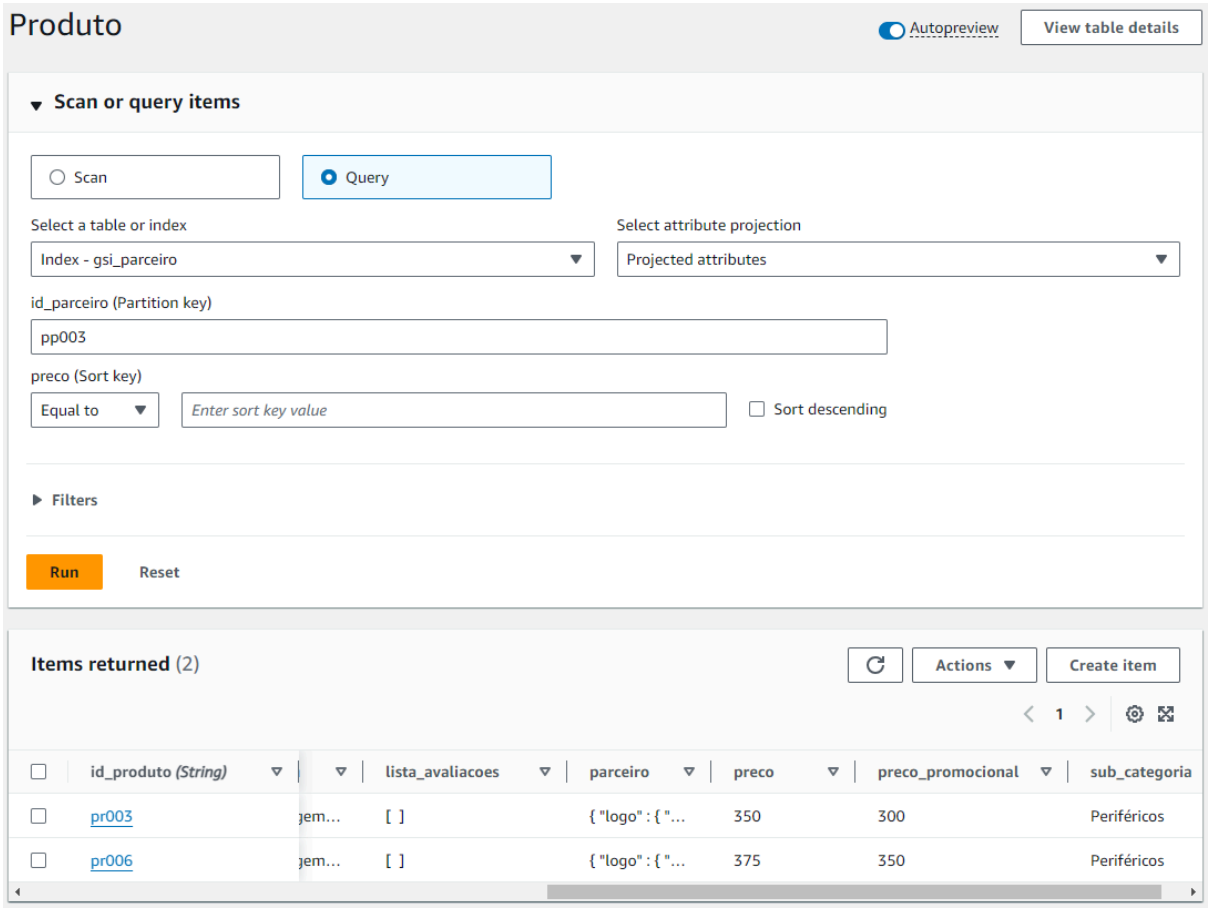


Figura 26: Consulta sobre produtos de um mesmo parceiro.

Por fim, a sexta consulta recupera todos os pedidos gerados a partir do fechamento de um carrinho específico, onde os produtos comprados podem ser de diferentes parceiros.

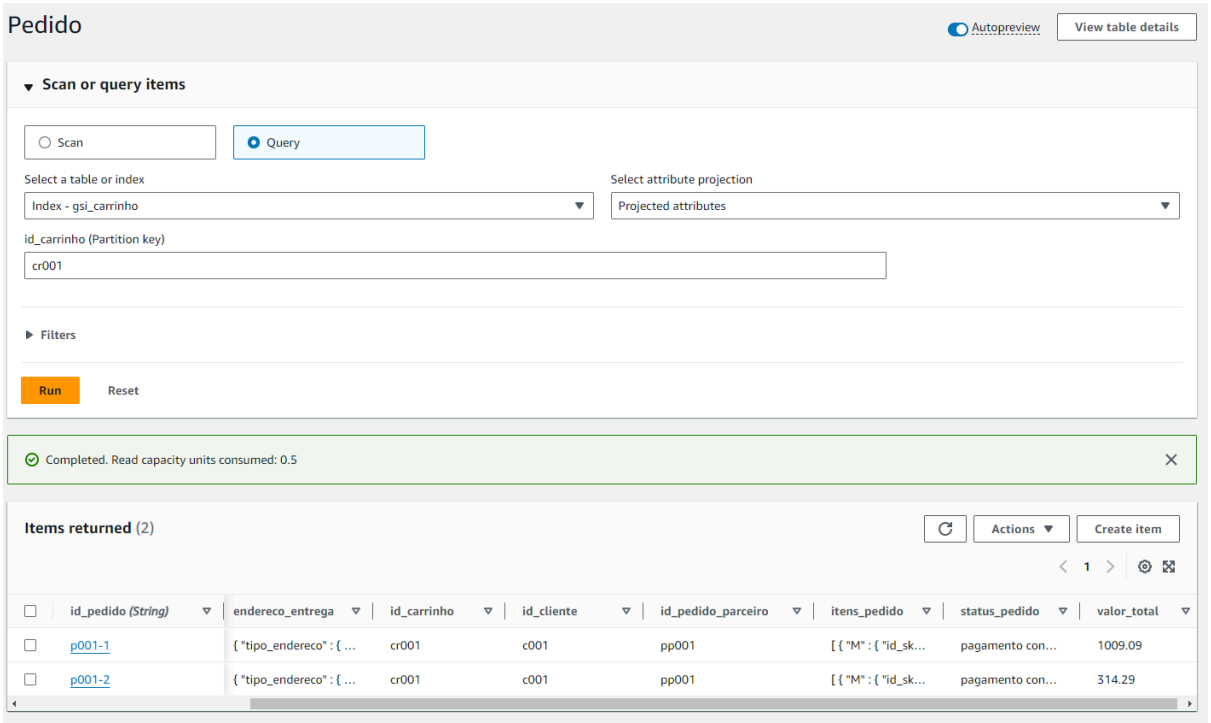


Figura 27: Sobre pedidos gerados a partir de um carrinho.