FIAP

Atividade
Microsserviços

Grupo:
Matheus Lemes Tassara
Vitória Barbosa da Silva

São Paulo, 15 de fevereiro de 2023

Etapa 1:

Implementar algumas telas usando o framework AngularJS. Avaliar se vale a pena iniciar a montagem ou uso de um container.

As atividades foram baseadas num aplicativo como autosserviço de compra de bilhetes de embarque de trens do metrô.

Embora os membros do time não estejam acostumados com AngularJS, um frontend simples pode ser encontrado aqui: https://github.com/mathtass/atv4-fiap

A página inicial se resume em uma lista de usuários, pensando em um controle gerencial de todos os usuários da plataforma. Além disto, para implementar o uso de botões, foi implementado os botões "Comprar Bilhete" e "Usar Bilhete", de modo que "Usar Bilhete" só aparece para aqueles que possuem um saldo disponível.

Abaixo, temos a visão do usuário *Vitória* com informações de acordo com o seu perfil, como Detalhes do usuário com login, saldo e número de bilhetes disponíveis:

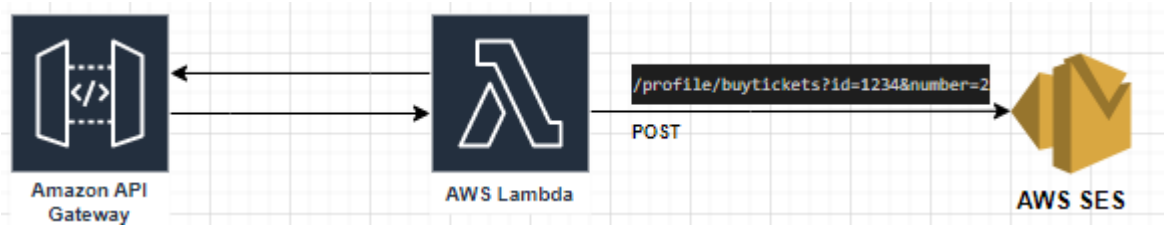
Além disto, podemos ver outra tela com um formulário para compra de bilhetes e função para o carrinho de compras:

Etapa 2:

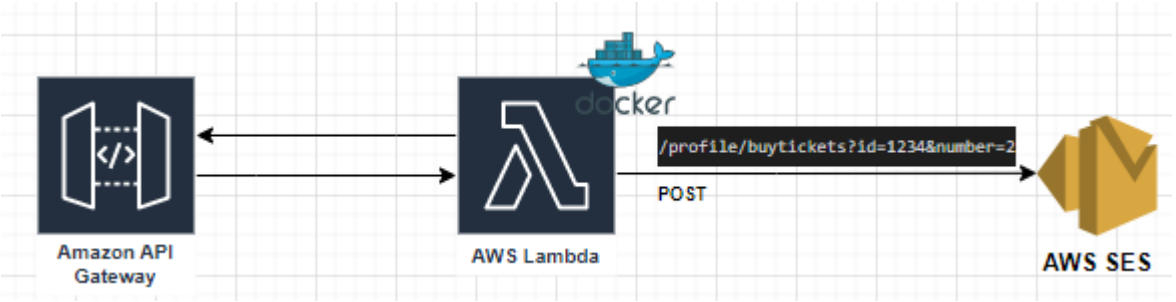
Antes de implementar, discuta com o grupo a real necessidade, considerando as vantagens e pontos de atenção apresentados no curso!

De acordo com o cenário, o uso de microsserviços não é aplicado a este projeto, em específico, pois sua arquitetura tem sido voltada para serverless com a integração AWS API Gateway x AWS Lambda, na qual fora desenvolvida em Python tendo em vista a familiaridade entre os membros do grupo.

Para que possa ser visualizado, o desenho da arquitetura idealizada está abaixo:



No entanto, para desenvolvimento da atividade, adaptamos a Lambda para uma Lambda Container com uso de Docker.



Etapa 3:

A terceira etapa consiste em implementar o Container.

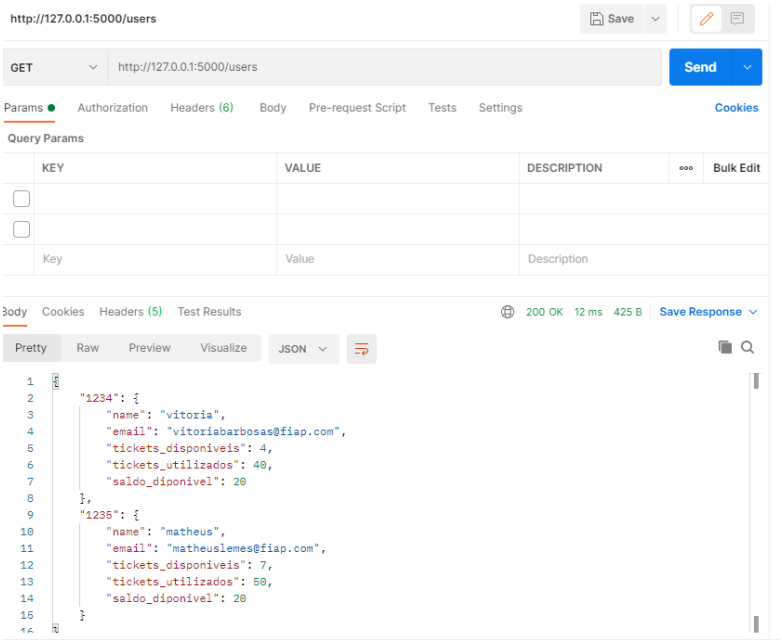
Esta atividade foi realizada em Python com Flask, e pode ser encontrado no repositório https://github.com/vitoriabarbosas/pos/tree/main/Microsservicos

Seguindo a mesma ideia de um autosserviço que possa realizar a compra de tickets, foi dado o nome de RecargaTOP para esta aplicação.

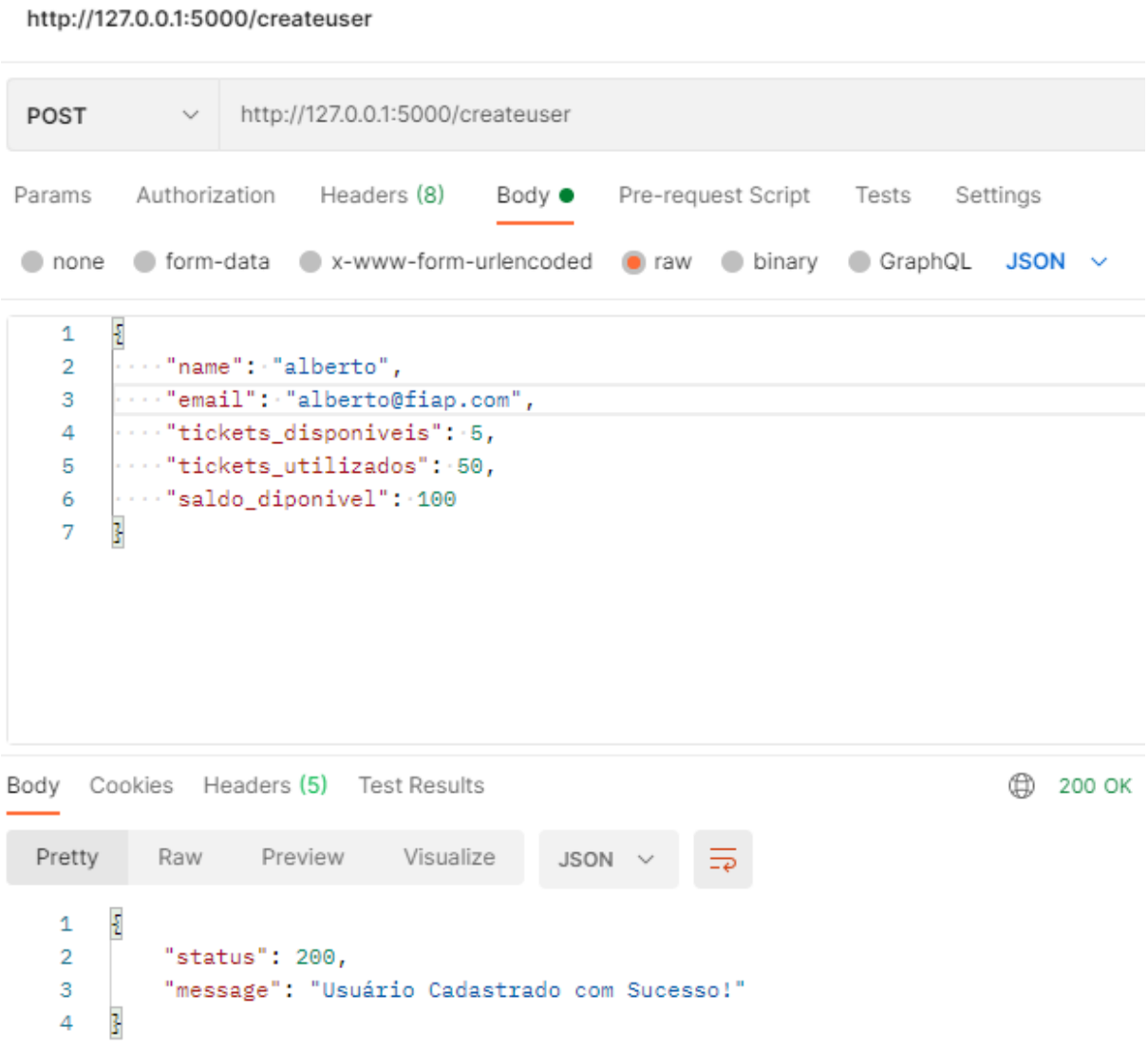
Basicamente a API possui 4 rotas principais:

Rota	Método	Descrição
/users	GET	Acesso a todos os usuários
/profile	GET	Acesso a um perfil específico
/createuser	POST	Criação de novos usuários
/buytickets	POST	Compra de novos tickets

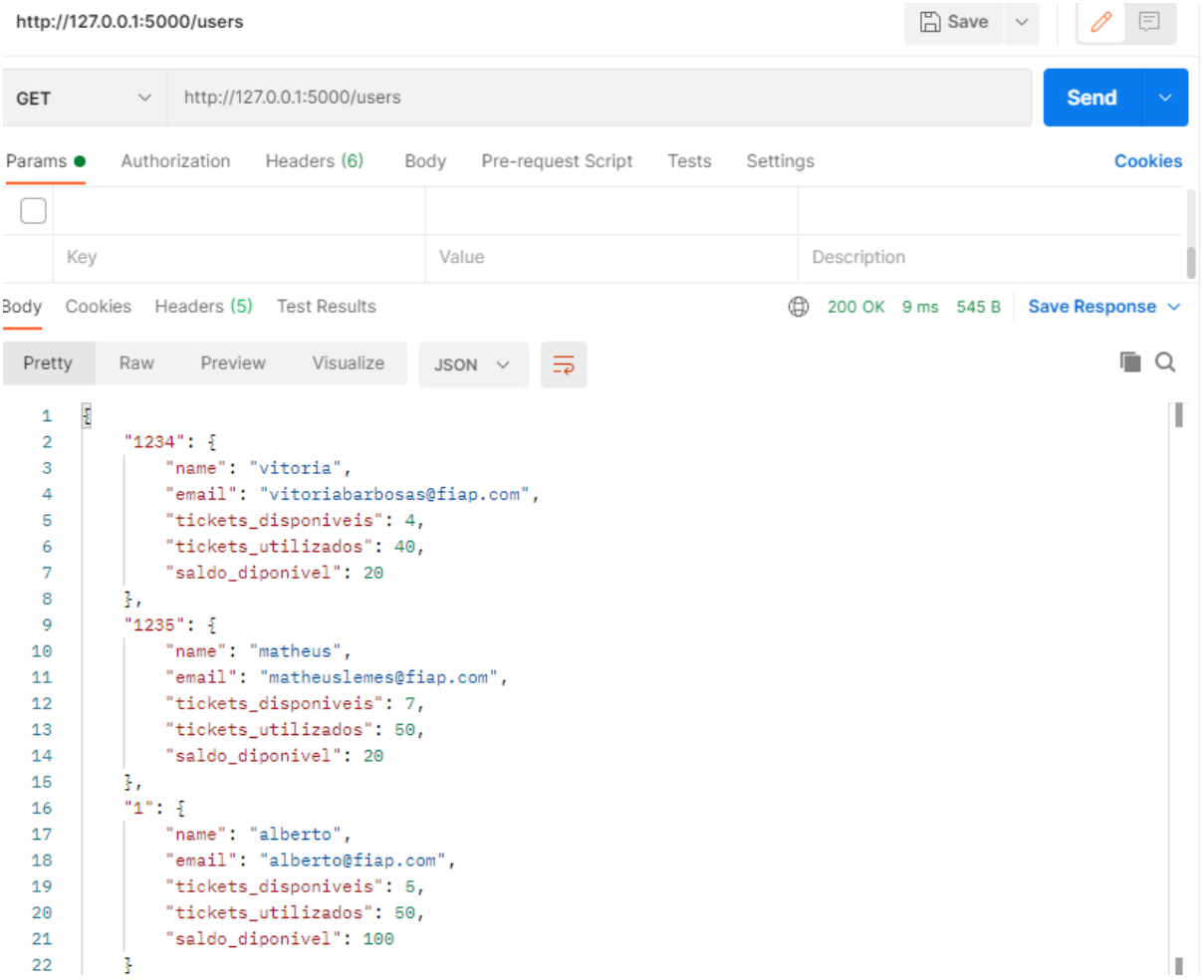
Na imagem abaixo, é possível ter um controle gerencial de todos os usuários da plataforma a partir da rota /users:



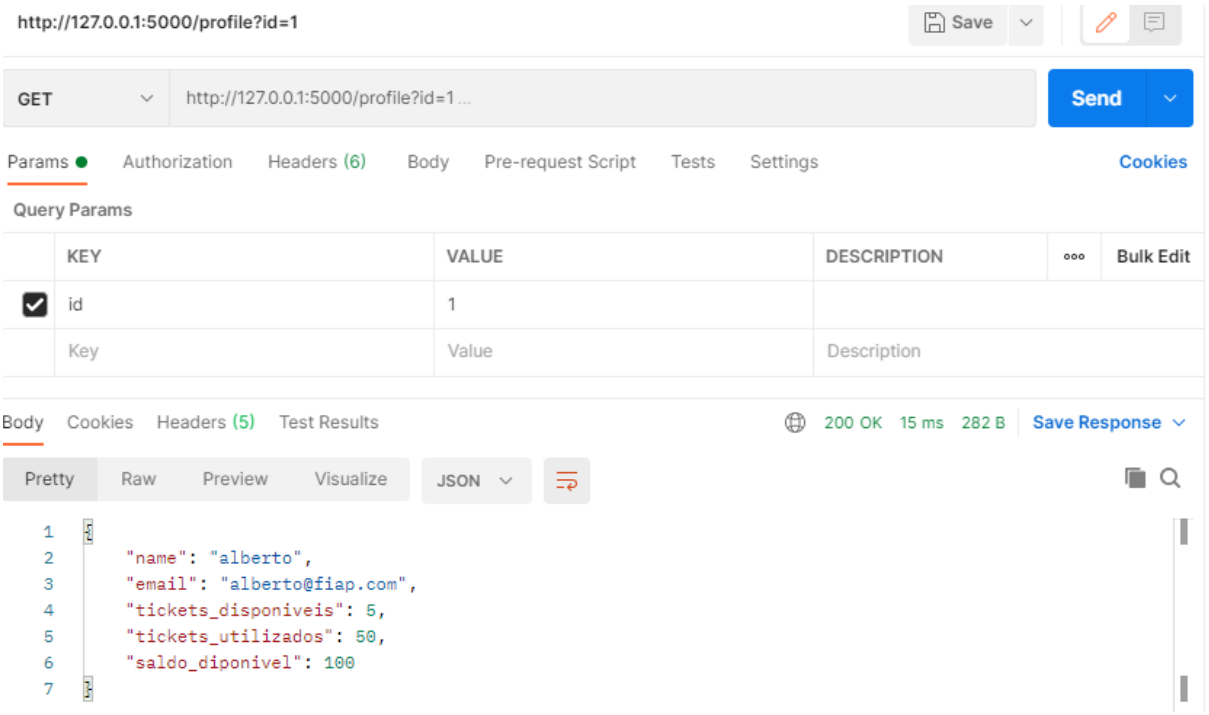
Pensando em criação de novos usuários, foi criada a rota /createuser, onde requisita os dados do novo usuário a ser criado:



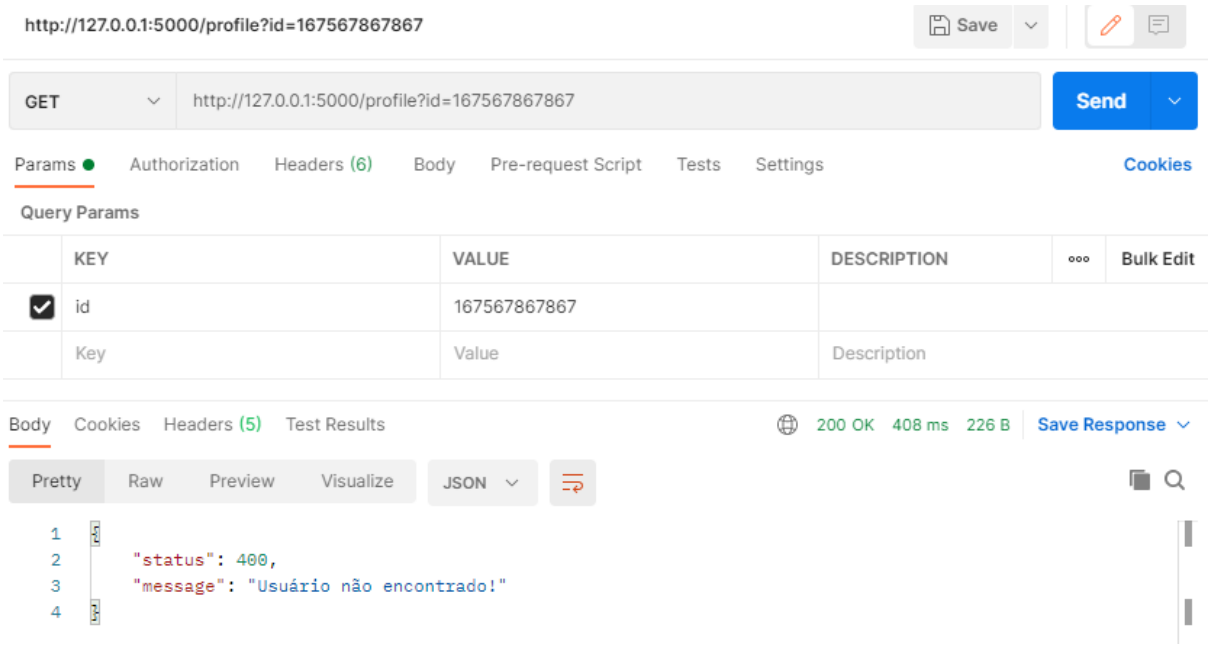
Para este novo usuário, será gerado um ID randômico para funcionamento do sistema.



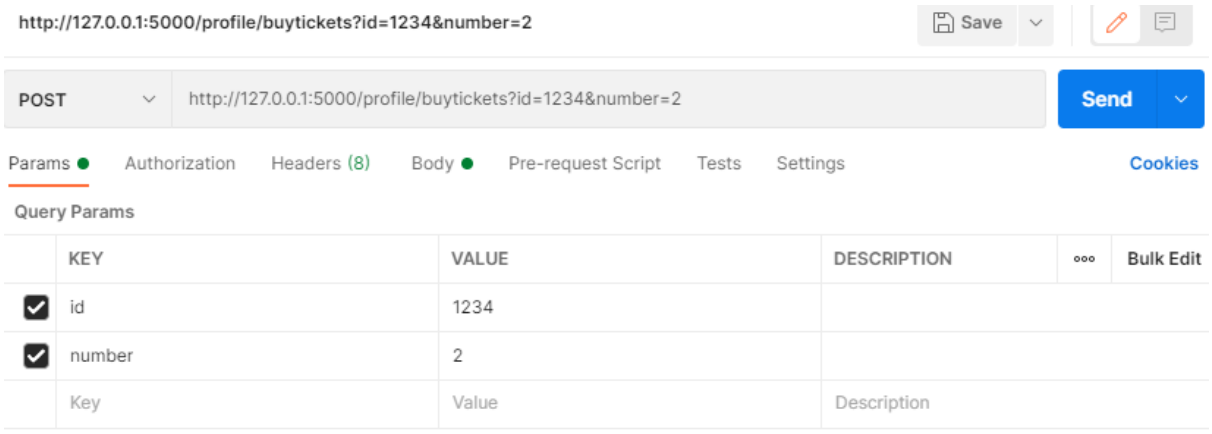
Para obter informações do perfil do usuário é possível utilizar a rota /profile, onde é requisitado o query parameter de acordo com o ID deste usuário. Em uma aplicação real, ao realizar o login na plataforma, este ID é passado por default entre as rotas.

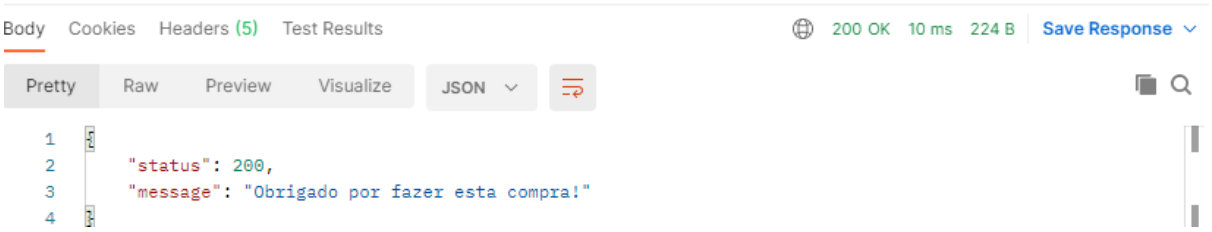


Caso o usuário não exista:



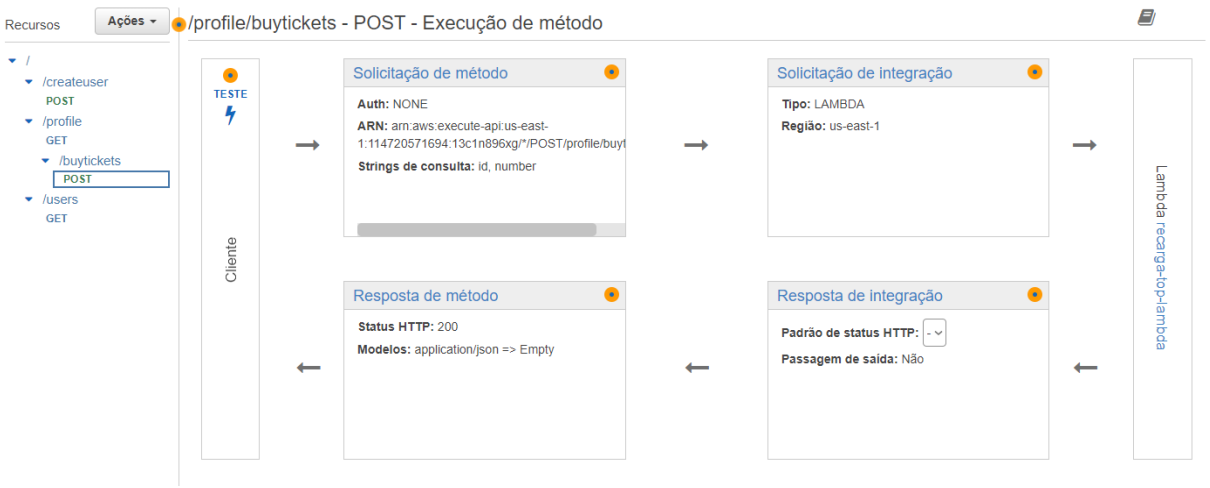
Além disto, temos a rota /buytickets para compra de bilhetes.



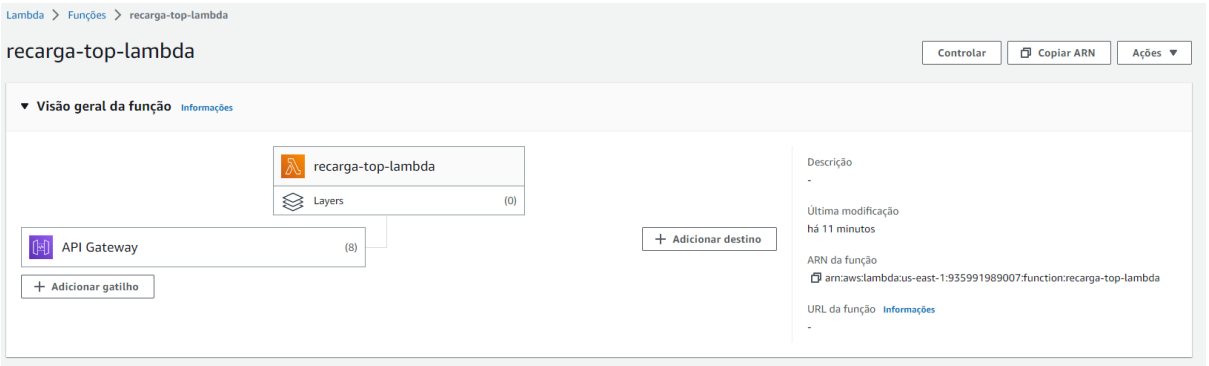


Componentes de infraestrutura na AWS:

API Gateway



Lambda

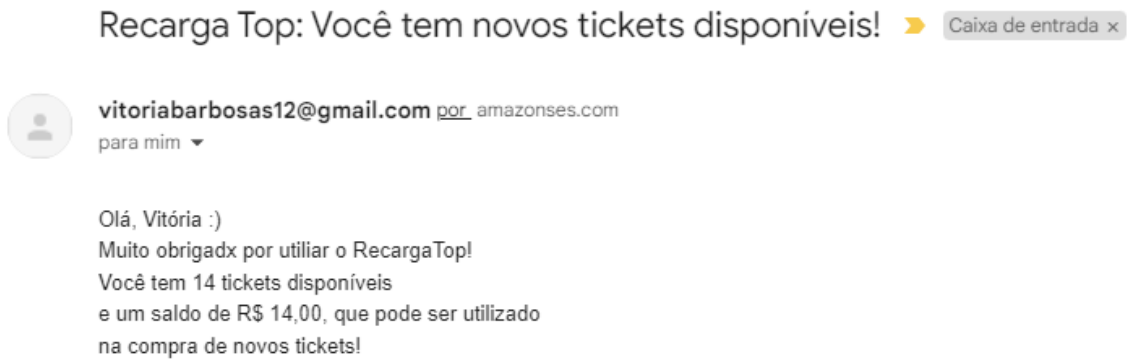


Etapa 4:

Implementação de um serviço de mensageria para que seja realizado o envio de mensagens como SMS e e-mail.

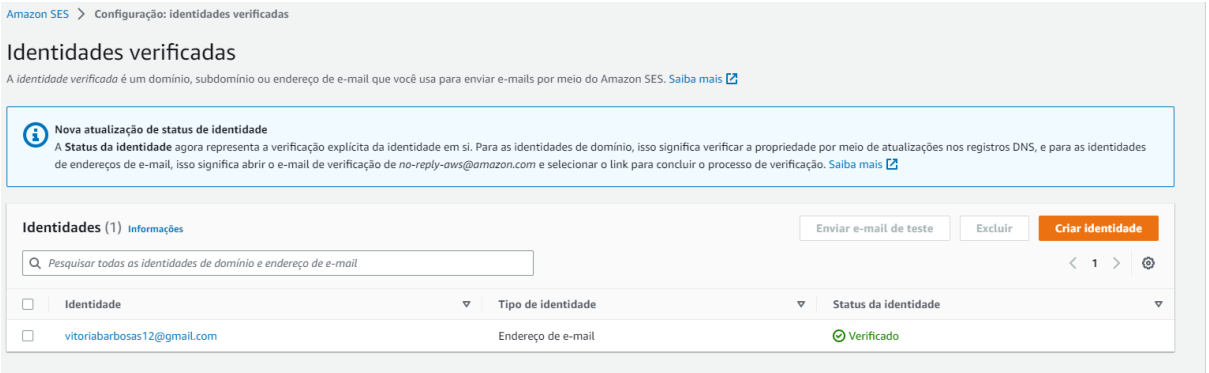
Por estarmos trabalhando na AWS podemos utilizar o SES (Simple Email Service), um serviço dedicado para envio de e-mails.

Para isto, quando uma requisição é feita na rota de compra de tickets, um email é enviado ao usuário solicitante:

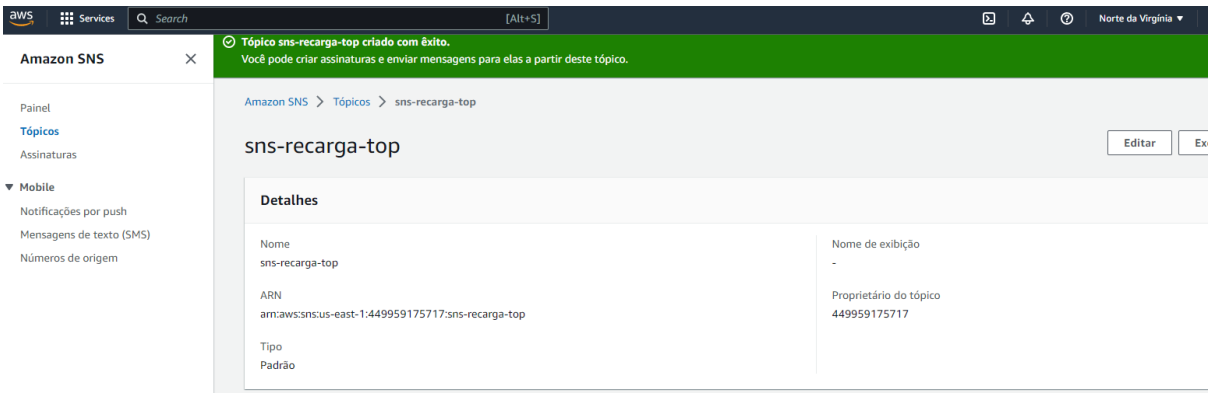


Na aplicação, foi adotado um email noreply, no entanto, este email precisa ser criado e confirmado no SES para que o envio seja possível. Para execução dos testes, um email já existente foi adotado.

Implementação da Infraestrutura na AWS
SES:



Por outro lado, também poderia ser utilizado o AWS SNS, no entanto, como a ideia do grupo foi realizar um email customizado, o AWS SES foi adotado.



Referências:

[1]AWS Lambda Container - https://docs.aws.amazon.com/lambda/latest/dg/images-create.html