API Terraform

API em Flask com Terraform

Flask

CORS

HashiCorp Terraform

Journey goal

Função

A Função é definida pela plataform e por Criar_Azur /Criar_AWS (Funções Principais)

Ao chamar as funções principais São executados as SubFunções

SubFunções: Ex: criar_vpc criar_grupo_recursos

Ao Chamar as SubFunções ele executa conforme a plataforma

Plataformas: AWS AZURE

CORS

A Integração do CORS com Flask inicia com o CORS(app)

Configuração por padrão permite a solicitação de qualuqer origem para permitir solicitações apenas para as origens específicas usamos CORS, como origins

Tratamento de solicitações CORS varifica se a solicitação é uma origem permitida (Access-Control-Allow-Origin)

Solicitações de Prefligth ultiliza certos tipos de solicitações como POTS, o navegador pode enviar uma solicitação como OPTIONS

A integração do Flask-CORS é feita através da chamada CORS(app) onde habilita o CORS para todas as rotas
O CORS lida com a adição de cabeçalhos CORS as respostas do seu aplicativo Flaks
Exemplo de CORS:
CORS(app, origins=["http://example.com"])

Endpoint

Na API temos 2 formas de usar endpoints, um para cada função(boa prática) e um para tudo
(na API temos um pra cada plataforma e um pra cada função

Método POST: este método é usado para criar recursos
Método OPTIONS: este método é usado para solicitações de preflinght

Endpoint para destruir é usado o método POST onde o corpo da solicitação deve ser um JSON contendo um campo: platform

Endpoint para menu não é comum já que só vamos consumir a API Como estmos fazendo tudo pelo Postman criamos um para facilitar nas requisições

Os Endpoints Funcionam da seguinte forma:
Solicitação para Criar Recursos
Solicitação para Destruir Recursos
Exibição do Menu

Frontend

Consumidno a API para o Front

- A Configuração do CORS é uma política de segurança implementada pelos navegadores para restingir como os recurosos de uma página porem ser acessados por outras paginas
  na API ultilizamos :
  CORS(app, resources={
  r"/azure/*": {"origins": "*"},
  r"/aws/*": {"origins": "*"}
  })
  Para definir as rotas das plataformas

- O Frontend consome a API do Back-end usando a o metodo Fetch, que é uma interface moderna para fazer solicitações HTTP:

```
async function criarRecursosAzure(recurso) {
try {
const response = await
fetch(`http://localhost:5000/azure/${recurso}`,
{
method: 'POST',
headers: {
 'Content-Type': 'application/json'
},
body: JSON.stringify({})
});
```

O Fluxo de Dados tem 4 partes

Solicitação do Front-end:
O front-end faz uma solicitação HTTP (por exemplo, POST) para um endpoint específico na API do back-end. A solicitação inclui os dados necessários para a operação desejada, como o nome do recurso a ser criado.

Processamento do Back-end:
O back-end recebe a solicitação, processa os dados conforme necessário (por exemplo, executando comandos Terraform para criar recursos na Azure ou AWS), e retorna uma resposta.

Resposta do Back-end: A resposta do back-end é enviada de volta para o front-end. Esta resposta pode incluir mensagens de sucesso ou erro, que são então usadas para atualizar a interface do usuário.

Ao realizar os 3 proecessor vai ser Autalizado a Interface do Usuário onde o O front-end usa a resposta da API para atualizar a interface do usuário, como exibir mensagens de sucesso ou erro