Documentação referente ao Projeto de Software

Identificação de data Palíndroma

Desenvolvedor Responsável: Fernando Ferreira da Cruz

Descrição da Atividade:

Desenvolver uma API para identificação da próxima data Palíndroma a partir de uma data inicial informada por parâmetro.

Ferramentas Utilizadas:

AWS Lambda

AWS API gateway

NodeJs14

Postman

Data última atualização da Documentação: 20/12/2023

Data de Finalização do Projeto: 20/12/2023

Objetivo:

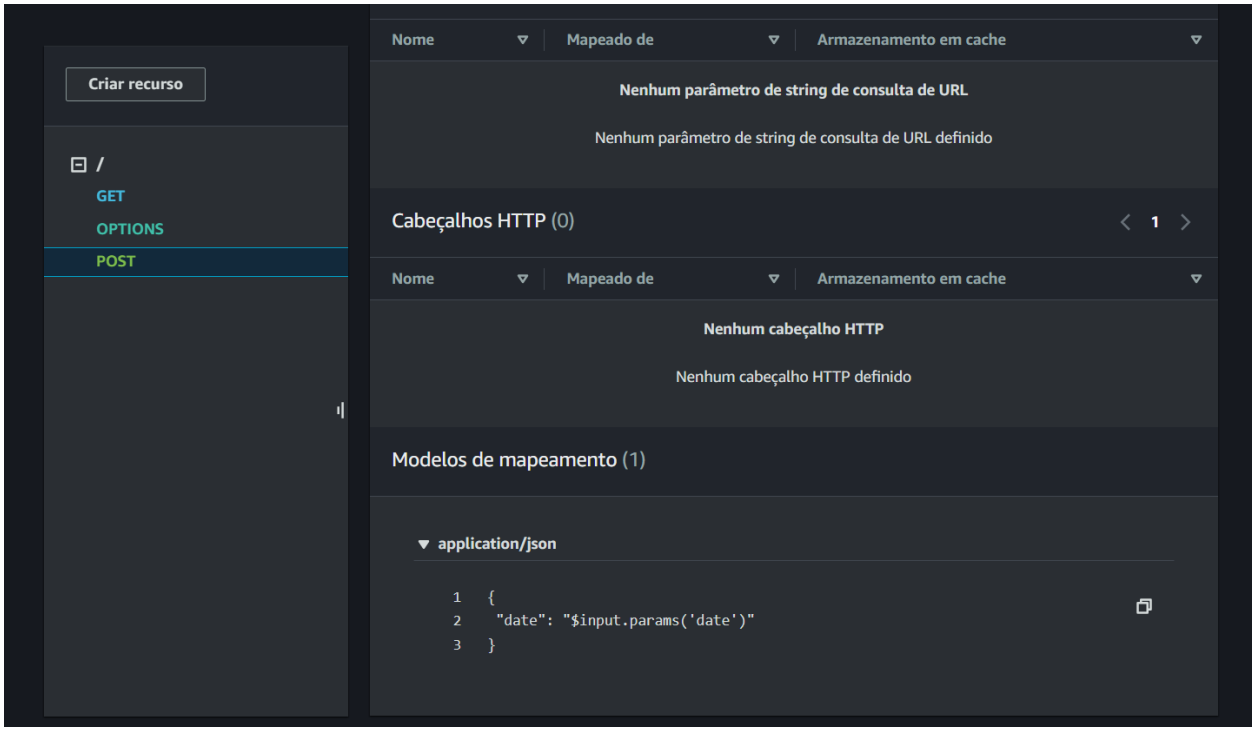
A API tem como objetivo receber um parâmetro em formato de data por requisição no endpoint e retornar em formato JSON uma mensagem contendo a próxima data a partir da informada que se encaixa como palindroma, ou seja, que mesmo quando invertida mantem a mesma sequencia numérica.

Para tal foram utilizadas as ferramentas **AWS API gateway** e **AWS Lambda**.
Para testes foi utilizado a ferramenta **PostMan**
Para desenvolvimento de Lógica foi Utilizado **NodeJs**

Modelos de Código e exemplo de Requisição Postman

AWS API gateway:

Configuração do APi gateway para recebimento do parâmetro ‘date’



AWS Lambda:

Configurações do Serverless criado no lambda, fazendo o processo automática de criação de configuração do Lambda, mediante é claro a alteração nos primeiros campos de ‘org’,’app’ e ‘service’ para conexão correta a Conta AWS responsável em Hospedar o API.

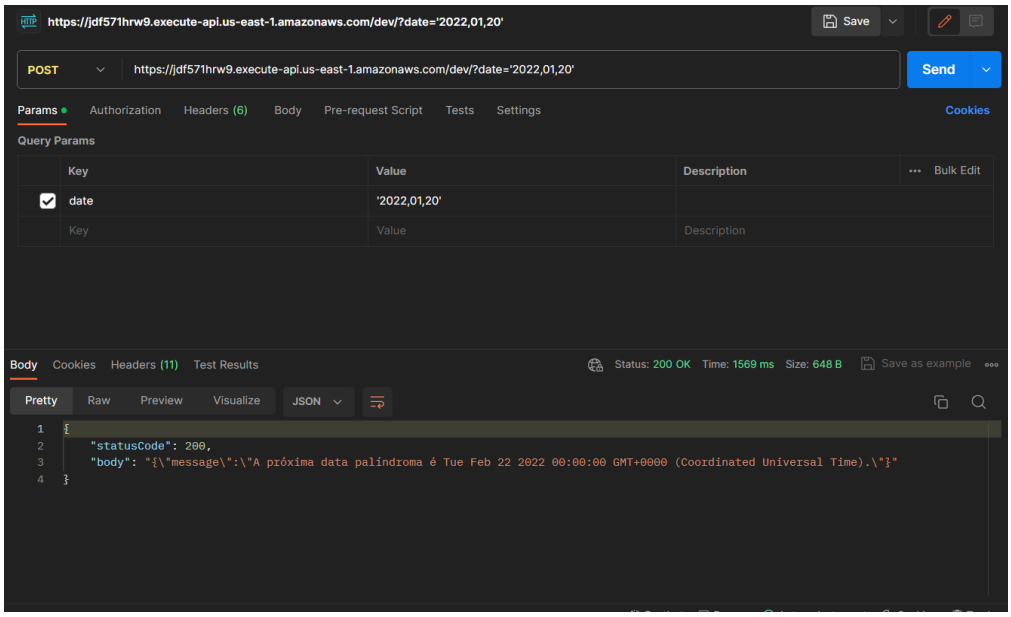
serverless.yml

```yaml
org: cruzfernando
app: sls-palindromo
service: sls-palindromo

provider:
  name: aws
  runtime: nodejs14.x
  stage: ${opt:stage, 'dev'}
  region: ${opt:region, 'us-east-1'}
  memorySize: 128
  timeout:

functions:
  index:
    handler: index.handler
    description : Return next Palidrome
    events:
     - http:
         path: /
         method: get
         cors: true

```

PostMan:

Exemplo de Request e Response da API mediante a ferramenta Postman.

Utilizado o URL : https://jdf571hrw9.execute-api.us-east-1.amazonaws.com/dev/.



NodeJs

Lógica de programação do API, os detalhes de funcionamento estão junto ao Código.

```
exports.handler = async function  handler(event,context) {
    const ParmPost = event.date;

    //Verifica se o parametro date foi preenchido
    if (ParmPost === '' || ParmPost === 'undefined'){
        return{
            statusCode: 200,
            body: JSON.stringify({message: `Favor informar um valor`})
        };
    }

    const dataAtual = new Date(ParmPost); //Gera a data a ser utilziada a partir do parametro

    //Identifica se a data gerada é valida, caso não, informa e exemplifica como preencher o campo
    if (isNaN(dataAtual)){
        return{
            statusCode: 200,
            body: JSON.stringify({message: `Favor inserir um valor válido de data. Ex: '2022,01,20'`})
        };
    }

    const proximaDataPalindroma = encontrarProximaDataPalindroma(dataAtual); //Chama a função principal que faz o processo de encontrar o próximo palindromo

    //Retorna o palindromo identificado
    return{
        statusCode: 200,
        body: JSON.stringify({message: `A próxima data palíndroma é ${proximaDataPalindroma}.`})
    };
};
    //Funções secundárias de filtragem de objeto data
    function verificarDataPalindroma(data) { //Recebe o campo date, e inverte para depois conferir se é palindromo
    const dataFormatada = data.toLocaleDateString('pt-BR', { timeZone: 'UTC' }); //Formata o campo na Timezone Brasileira
    const dataInvertida = dataFormatada.split('').reverse().join(''); // Qubra o campo Date informado para inverter e junta novamente. Exemplo 2023/01/12 -> 21/10/3202
    const DataFinal     = dataInvertida.toString().replace('/', '').replace('/', ''); // Remove os campos '/' da String para poder posteriormente comparar
        return DataFinal;
    }

  function FiltraDataNormal(data) { //Recebe o campo date para formatar e comparar posteriormente com a versão espelhada
    const dataFormatada = data.toLocaleDateString('pt-BR', { timeZone: 'UTC' }); //Formata o campo na Timezone Brasileira
    const dataString = dataFormatada.toString;//split('').join('');
    const DataFinal     = dataString.toString().replace('/', '').replace('/', ''); // Remove os campos '/' da String para poder posteriormente comparar
        return DataFinal;
  }
```

```javascript
// Confere se a data é de fato palindroma e busca dias após dia até achar a próxima a partir da data informada
function encontrarProximaDataPalindroma(data) {
    let Data = new Date(data);
    let DataInvertida = '';
    let DataAux = '';

    while(true){

        DataAux      = FiltraDataNormal(Data); //Formata data para comparação retornando campo String
        DataInvertida = verificarDataPalindroma(Data); //Formata e inverte data para comparação retornando campo String

        //Faz o teste lógico para identificar se é String, caso não seja incrementa mais um dia a data e tenta novamente
        if (DataInvertida === DataAux){
            break;
        } else {
            Data.setDate(Data.getDate() + 1);
        }
    }

    return Data; // Ao final retorna a Primeira data Palindroma encontrada a partir da data informada
}
```