Documentação da API para Cálculo de Números Primos

1. Introdução

Este projeto consiste em uma API desenvolvida em Flask que calcula os números primos dentro de um intervalo especificado. Um cliente em Node.js é usado para consumir esta API e exibir os resultados no console.

---

2. API Flask - server.py

2.1 Descrição

A API fornece um endpoint HTTP para calcular números primos em um intervalo definido e tem como dependência o Flask.

2.2 Estruturação do código

- **Inicialização aplicação Flask e suas bibliotecas:**

```python
from flask import Flask, request, jsonify

app = Flask(__name__)
```

- **Função de cálculo de primos**:

```python
# Função utilizada para calcular números primos em um intervalo
def calcular_primos(limite_inferior, limite_superior):
    primos = []                                          # Lista para armazenar os números primos
    for num in range(limite_inferior, limite_superior + 1):   # Itera números no intervalo
        if num > 1:                                      # Se Números primos são maiores que 1
            for i in range(2, int(num ** 0.5) + 1):      # Verifica divisibilidade até a raiz quadrada
                if num % i == 0:                         # verifica se i divide num
                    break
            else:
                primos.append(num)                       # Adiciona à lista se for primo
    return primos
```

- **Endpoint /primos:** O método GET recebe dois parâmetros obrigatórios: limite_inferior e limite_superior, que definem o intervalo para a operação. Tendo como respostas as possíveis mensagens:
    - **200**: Lista de números primos.
    - **400**: Erros de validação ou Bad Request (ex.: limites inválidos).

- **500**: Erro genérico não tratado ou erro no servidor.

```python
@app.route('/primos', methods=['GET'])
def func_primos():
    try:
        limite_inferior = int(request.args.get('limite_inferior'))
        limite_superior = int(request.args.get('limite_superior'))

        # Verifica se os limites são válidos
        if limite_inferior > limite_superior:
            return jsonify({"erro": "Limite inferior maior que o superior"}), 400  # Resposta de erro HTTP 400

        primos = calcular_primos(limite_inferior, limite_superior)
        return jsonify({"primos": primos})  # Retorna a lista de primos como resposta JSON

    except ValueError:
        return jsonify({"erro": "Parâmetros inválidos"}), 400 # Erro de parametros inválidos, Bad Request
    except Exception as e:
        return jsonify({"erro": str(e)}), 500 # Erro genérico não tratado
```

- **Execução:**

```python
# Parte Main da aplicação
if __name__ == '__main__':
    # Executa a aplicação Flask no modo de depuração
    app.run(debug=True)
```

Ao executar a aplicação a API estará disponível em: http://127.0.0.1:5000/primos

3. Cliente Node - client.js

3.1 Descrição

O cliente realiza uma requisição à API Flask para calcular os números primos em um intervalo e exibe o resultado no console e tendo dependências como node.js e axios (Biblioteca para realizar requisições HTTP).

3.2 Estruturação do código

- **Inicialização do axios:**

```javascript
// Importa a biblioteca axios para realizar requisições HTTP
const axios = require('axios');
```

- **Função “primos”**:

```javascript
async function primos(limiteInferior, limiteSuperior) {
    try {
        // Faz uma requisição GET para a API Flask
        const response = await axios.get('http://127.0.0.1:5000/primos', {
            params: {
                limite_inferior: limiteInferior, // Envia o limite inferior como parâmetro
                limite_superior: limiteSuperior  // Envia o limite superior como parâmetro
            }
        });
        // Verifica se a resposta HTTP é bem-sucedida (código 200)
        if (response.status === 200) {
            // Exibe os números primos retornados pela API
            console.log("Números primos no intervalo:", response.data.primos);
        } else {
            // Caso a resposta tenha outro código, exibe a mensagem de erro
            console.error("Erro:", response.data.erro);
        }
    } catch (error) {
        // Captura erros na execução da requisição
        if (error.response) {
            // Caso o erro seja uma resposta da API (ex.: erro 400 ou 500)
            console.error("Erro na resposta:", error.response.data.erro);
        } else {
            // Caso o erro seja um problema de rede ou configuração
            console.error("Erro na requisição:", error.message);
        }
    }
}
```

- **Execução:**

```javascript
// Exemplo de teste da função
// Faz uma consulta à API para encontrar os números primos no intervalo de 1 a 100
primos(1, 100);
```

4. Observação: Para a execução se deve ter todas as dependências comentadas, como também o server.py deve estar rodando antes da execução do client.js via terminal usando o comando “node client.js”.

assim tendo o seguinte resultado:

```
Números primos no intervalo: [
   2,  3,  5,  7, 11, 13, 17, 19,
  23, 29, 31, 37, 41, 43, 47, 53,
  59, 61, 67, 71, 73, 79, 83, 89,
  97
]
```