Para criar uma API que permita o envio de dados de um formulário de um front-end feito com React, JavaScript e CSS responsivo para um e-mail, você precisará de uma abordagem que inclui backend para processar o envio do e-mail e a integração com o front-end. Aqui está um guia completo com dicas e exemplos de código:

**Passos para Implementação**

**1. Configurar o Front-End com React**

* Crie um formulário em React.
* Use useState para gerenciar os dados do formulário.
* Utilize fetch ou axios para enviar os dados ao backend.

Exemplo de componente React:

import React, { useState } from 'react';

const ContactForm = () => {

const [formData, setFormData] = useState({

name: '',

email: '',

message: '',

});

const handleChange = (e) => {

setFormData({ ...formData, [e.target.name]: e.target.value });

};

const handleSubmit = async (e) => {

e.preventDefault();

try {

const response = await fetch('http://localhost:5000/send-email', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(formData),

});

const data = await response.json();

if (data.success) {

alert('E-mail enviado com sucesso!');

} else {

alert('Erro ao enviar e-mail.');

}

} catch (error) {

console.error(error);

alert('Erro de conexão com o servidor.');

}

};

return (

<form onSubmit={handleSubmit} style={{ maxWidth: '500px', margin: 'auto' }}>

<label>

Nome:

<input type="text" name="name" value={formData.name} onChange={handleChange} required />

</label>

<label>

E-mail:

<input type="email" name="email" value={formData.email} onChange={handleChange} required />

</label>

<label>

Mensagem:

<textarea name="message" value={formData.message} onChange={handleChange} required />

</label>

<button type="submit">Enviar</button>

</form>

);

};

export default ContactForm;

**2. Criar o Backend**

Você pode usar Node.js com o pacote nodemailer ou Python com Flask para processar o envio do e-mail.

**Backend com Node.js**

1. Instale as dependências:

npm install express cors nodemailer body-parser

1. Crie o arquivo server.js:

const express = require('express');

const nodemailer = require('nodemailer');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

app.use(cors());

app.use(bodyParser.json());

const transporter = nodemailer.createTransport({

service: 'Gmail',

auth: {

user: 'seu-email@gmail.com',

pass: 'sua-senha', // Use variáveis de ambiente para segurança

},

});

app.post('/send-email', (req, res) => {

const { name, email, message } = req.body;

const mailOptions = {

from: email,

to: 'seu-email@gmail.com',

subject: `Mensagem de ${name}`,

text: message,

};

transporter.sendMail(mailOptions, (error, info) => {

if (error) {

console.error(error);

res.status(500).send({ success: false, error });

} else {

res.send({ success: true, info });

}

});

});

app.listen(5000, () => console.log('Server running on port 5000'));

**Backend com Flask**

1. Instale as dependências:

pip install flask flask-cors flask-mail

1. Crie o arquivo app.py:

from flask import Flask, request, jsonify

from flask\_mail import Mail, Message

from flask\_cors import CORS

app = Flask(\_\_name\_\_)

CORS(app)

app.config['MAIL\_SERVER'] = 'smtp.gmail.com'

app.config['MAIL\_PORT'] = 587

app.config['MAIL\_USE\_TLS'] = True

app.config['MAIL\_USERNAME'] = 'seu-email@gmail.com'

app.config['MAIL\_PASSWORD'] = 'sua-senha' # Use variáveis de ambiente

mail = Mail(app)

@app.route('/send-email', methods=['POST'])

def send\_email():

data = request.get\_json()

name = data['name']

email = data['email']

message = data['message']

msg = Message(f'Mensagem de {name}',

sender=email,

recipients=['seu-email@gmail.com'])

msg.body = message

try:

mail.send(msg)

return jsonify({'success': True}), 200

except Exception as e:

return jsonify({'success': False, 'error': str(e)}), 500

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True, port=5000)

1. **Testar a Integração**

* Inicie o backend (node server.js ou python app.py).
* Execute o front-end (npm start).
* Preencha o formulário e envie os dados.

1. **Melhorias**

* **Segurança:** Nunca exponha senhas diretamente no código. Use variáveis de ambiente.
* **Validação:** Valide os campos no front-end e no back-end.
* **Erros:** Trate erros e exiba mensagens claras aos usuários.
* **Responsividade:** Ajuste o CSS para telas menores e otimize a experiência do usuário.