

# PRÁTICAS AVANÇADAS EM DESENVOLVIMENTO WEB

Davi Schneid - [davi.Schneid@gmail.com](mailto:davi.Schneid@gmail.com)

09/08/2024

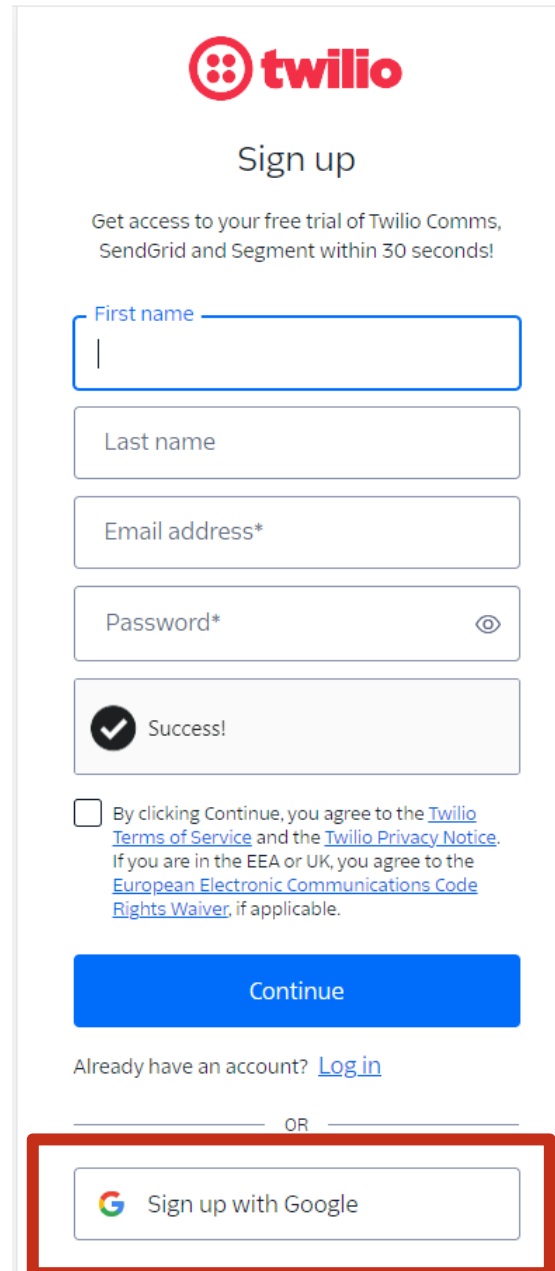
# Agenda

- ▶ Enviando SMS com Twilio
  - ▶ Configurar a conta
  - ▶ Instalar o pacote Twilio
  - ▶ Preparar Backend para enviar SMS
- ▶ Enviar e-mail com SMTP Google
- ▶ Dependências:
  - ▶ `npm install twilio`
  - ▶ `npm install nodemailer`




# Twilio

- ▶ Criar uma conta
- ▶ Acessar com a conta do Google



The image shows a screenshot of the Twilio sign-up page. At the top is the Twilio logo. Below it is the heading "Sign up" and a subtext: "Get access to your free trial of Twilio Comms, SendGrid and Segment within 30 seconds!". The form contains several input fields: "First name" (with a cursor), "Last name", "Email address\*", and "Password\*" (with an eye icon). Below these is a "Success!" message with a checkmark icon. A checkbox is followed by a paragraph of terms and conditions. A blue "Continue" button is below that. At the bottom, it says "Already have an account? [Log in](#)". Below a horizontal line with "OR" in the center is a red-bordered box containing a "Sign up with Google" button with the Google logo.




## Sign up


Get access to your free trial of Twilio Comms, SendGrid and Segment within 30 seconds!

First name

Last name

Email address\*

Password\*  


 Success!

☐ By clicking Continue, you agree to the [Twilio Terms of Service](#) and the [Twilio Privacy Notice](#). If you are in the EEA or UK, you agree to the [European Electronic Communications Code Rights Waiver](#), if applicable.

[Continue](#)

Already have an account? [Log in](#)


OR


 Sign up with Google

## ► Verificar o seu telefone

### We'll also need to verify your phone number



 So you can hit the ground running and start using our **Twilio services**.

 To verify you during log in through **two-factor authentication (2FA)**.

 To help us **mitigate fraud and abuse**.

Country

BR (+55) Brazil



Phone Number

519 [REDACTED]

[Send code via SMS](#)

[Send code via voice call](#)

### Check your phone for a verification code

Twilio Verify has sent the code to:

+55519 [REDACTED]



Enter verification code

087601

[Verify](#)

[Resend code via SMS](#)

[Send code via voice call](#)

[Verify with another phone number](#)

## ► Copiar o código de recuperação



### You're all verified!

If you lose your phone, or don't have access to your verification device, this code is your failsafe to access your account.

Recovery code

4HYYSVPBM2ZG85ZHS1836K86



Save this code somewhere safe and accessible

Continue

## ► Preencher o formulário

## Ahoy Davi Schneid, welcome to Twilio!

Tell us a bit about yourself so we can personalize your experience. You will have access to all Twilio products.

• What do you plan to build with Twilio?

Identity & Verification (OTP & 2FA)

• Which Twilio product are you here to use?

Verify

• Which best describes you/your organization?

☐ Business

☐ Nonprofit or government entity

☐ Sole proprietor / Self-employed

☒ Hobbyist or Student

• How do you want to build with Twilio?

☒ With code

Customize exactly what you want

☐ With minimal code

Build on top of our code samples

☐ With no code at all

Launch a starter app with no code

• What is your preferred coding language?

JavaScript

• Would you like Twilio to host your code?

Host your Twilio app on our secure servers

☐ Yes, host my code on Twilio

☒ No, I want to use my own hosting service

Your billing country is Brazil. [\(Change\)](#)

Get Started with Twilio

## ► Preencher o formulário

Twilio Home

My first Twilio account ▾

Trial: \$15.50 [Upgrade](#)

Q Jump to...

Account Dashboard

Develop Monitor

> # Phone Numbers

> ✓ Verify

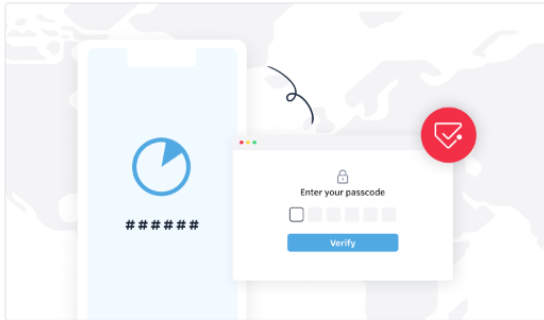
> Marketplace Beta

> Messaging

> Video

Explore products +

## Ahoy Davi, welcome to Twilio!



### Multi-channel verification with Verify

SMS WhatsApp Voice Email Push TOTP SNA

Verify is a fully managed API for multichannel user verification. And it now includes guaranteed protection from SMS pumping fraud with Fraud Guard.

Start building → Learn more ↗

One-time password (OTP) Two-factor authentication (2FA) [Show more](#)

▼ Account Info

Account SID

AC310dc45008593c2a1f3c8cf4680d00b9 [Copy](#)

Auth Token

..... [Copy](#) [Show](#)

⚠ Always store your token securely to protect your account. [Learn more](#)

You are in a trial account and can only send messages and make calls to [verified phone numbers](#). [Learn more about your trial account](#)

▼ Verify builder resources

Start building solutions with selected Verify-specific resources:

- Quick start guides & integrations
- Best practices
- Technical documentation

Explore Verify Info Hub →



# ApiRouter

- Instalar o modulo Twilio: `npm install twilio`

```
● PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiRouter> npm install twilio  
  
added 6 packages, and audited 186 packages in 15s  
  
17 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities
```

# ApiRouter

- ▶ Adicionar no arquivo .env as credencias do Twilio.

```
10  
11  #Usado para enviar SMS  
12  #Substitua pelo seu Account SID  
13  ACCOUNT_SID=AC310dc45008593c2a1f3c8cf4680d00b9  
14  
15  #Substitua pelo seu Auth Token  
16  AUTH_TOKEN=efcbdd4db096e752a195435d2177b153
```

# ApiRouter

- ▶ Criar uma pasta chamada service na raiz da aplicação:
- ▶ Criar o arquivo enviarSMSService.js

```
service > JS enviarSMSService.js > enviarSMSService > from
1 // Carrega a biblioteca Twilio
2 const twilio = require('twilio');
3
4 // Suas credenciais do Twilio
5 const accountSid = process.env.ACCOUNT_SID; // Substitua pelo seu Account SID
6 const authToken = process.env.AUTH_TOKEN; // Substitua pelo seu Auth Token
7
8 // Cria uma instância do cliente Twilio
9 //const client = new twilio(process.env.ACCOUNT_SID, process.env.AUTH_TOKEN);
10
11 const client = new twilio(accountSid, authToken);
12
13 // Função para enviar um SMS
14 function enviarSMSService(para, mensagem) {
15   console.log('ENVIAR SMS');
16   client.messages.create({
17     body: mensagem, // O conteúdo da mensagem
18     from: '??????', // Substitua pelo seu número do Twilio
19     to: para // Número para o qual a mensagem será enviada
20   })
21   .then((message) => console.log(`Mensagem enviada com ID: ${message.sid}`))
22   .catch((error) => console.error(error));
23 }
24
25 module.exports = { enviarSMSService };
26
```

# ApiRouter

- ▶ Na pasta controller, criar o arquivo enviarMensagemController.js
- ▶ Escrever o código

```
3  const { enviarSMSService } = require('../service/enviarSMSService');
4
5
6  exports.enviarSMS = async (req, res) => {
7    const { telefone , mensagem } = req.body;
8    console.log('Telefone:',telefone);
9    console.log('Mensagem:',mensagem);
10   try{
11     if (!telefone) {
12       console.log('Retorna http 400');
13       res.status(400).json('Informar telefone');
14     }else{
15       enviarSMSService(telefone,mensagem);
16     }
17   }catch(err){
18     console.log('Erro ao enviar SMS',err);
19     res.status(500).json({ error: 'Erro ao enviar SMS' });
20   }
21 };
22
```

# ApiRouter

- ▶ Na pasta rotas, criar o arquivo enviarMensagemRotas.js, e criar a rota enviarSMS
- ▶ Escrever o código

```
rotas > JS enviarMensagemRotas.js > ...
1
2 //Importa o modulo Express
3 const express = require('express');
4
5 //Cria o objeto rotas
6 const router = express.Router();
7
8
9 //Importa o modulo tokenController
10 const enviarMensagemController = require('../controllers/enviarMensagemController');
11 |
12 //cria a rota de enviar SMS
13 /**
14  * @swagger
15  * /enviarsms:
16  *   post:
17  *     summary: Enviar mensagem via SMS
18  *     tags: [Envio de mensagens]
19  *     requestBody:
20  *       required: true
21  *       content:
22  *         application/json:
23  *           schema:
24  *             type: object
25  *             properties:
26  *               telefone:
27  *                 type: string
28  *                 description: Telefone para enviar a mensagem
29  *               mensagem:
30  *                 type: string
31  *                 description: Mensagem a ser enviada
32  *     responses:
33  *       200:
34  *         description: Mensagem enviada com sucesso
35  *         content:
36  *           application/json:
37  *             schema:
38  *               type: object
39  *               properties:
40  *                 retorno:
41  *                   type: string
42  *       400:
43  *         description: Telefone não encontrado ou inválido
44  *       500:
45  *         description: Erro ao enviar a mensagem
46  */
47 router.post('/enviarsms', enviarMensagemController.enviarSMS);
48
49
```

# ApiRouter

- No arquivo server.js, adicionar a nova rota.

```
JS server.js > [PORT]
1
2 const express = require('express');
3 const sequelize = require('./data_base/db');
4 const usuariosRotas = require('./rotas/usuarioRotas');
5 const uploadArquivoRotas = require('./rotas/uploadArquivoRotas');
6 const validarToken = require('./rotas/tokenRotas');
7 const enviarMensagem = require('./rotas/enviarMensagemRotas');
8
9 //Importar o modulo Swagger
10 const setupSwagger = require('./swagger');
11
12 //importar o modulo cors para receber requisicoes de diferente origem
13 const cors = require('cors');
14
15
16 const app = express();
17 const PORT = process.env.PORT;
18
19
20 app.use(require("cors")());
21
22 //restringir chamadas somente da origem conhecida
23 const corsOptions = {
24   origin: 'http://localhost:3000', // Permitir apenas essa origem
25   methods: 'GET,HEAD,PUT,PATCH,POST,DELETE', // Métodos permitidos
26   credentials: true, // Permitir envio de cookies
27   optionsSuccessStatus: 204 // Status para requisições preflight
28 };
29
30 app.use(cors(corsOptions));
31
32 // Configurar Swagger
33 setupSwagger(app);
34
35 app.use(express.json());
36 app.use('/api', usuariosRotas);
37 app.use('/api', uploadArquivoRotas);
38 app.use('/api', validarToken);
39 app.use('/api', enviarMensagem);
40
41
42 sequelize.sync().then(() => {
43   app.listen(PORT, () => {
44     console.log(`Servidor rodando na porta ${PORT}`);
45   });
46 });
```

# ApiRouter

- ▶ Testar o envio de SMS, pelo Swagger.

## Envio de mensagens

POST

/enviarsms

Enviar mensagem via SMS

Parameters

No parameters

Request body required

application/json

```
{  "telefone": "51999999999",  "mensagem": "Teste SMS"}
```

Execute

# ApiRouter

- ▶ Enviar e-mail com SMTP Google
- ▶ Instalar o modulo Nodemailer: `npm install nodemailer`

```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiRouter> npm install nodemailer

added 1 package, and audited 187 packages in 3s

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



# ApiRouter

- Adicionar no arquivo .env as credencias para enviar E-mail.

```
18  #Usando para o envio de email
19  USER_EMAIL=turmamiltalentos@gmail.com
20  PWD_EMAIL_GMAIL=turm@3Milt@lentos
21  PWD_EMAIL_GMAIL_APP=xjzsvikpbgencbuj
22  |
23  CLIENT_ID_GMAIL=341121125456-34m79g8efsqh9ao0plcv0qu8qoihk97a.apps.googleusercontent.com
24  SECRET_ID_GMAIL=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
```

# ApiRouter

- ▶ Criar na pasta service o arquivo enviarEmailService.js
- ▶ Escrever o código

```
service > JS enviarEmailService.js > [?] transporter > [?] auth > [?] user
1
2
3  const nodemailer = require('nodemailer');
4
5  // Configura o transporte de email
6  const transporter = nodemailer.createTransport({
7    host: 'smtp.gmail.com',
8    port: 465,
9    secure: true, // true para usar SSL/TLS
10   service: 'gmail', // ou outro serviço como Yahoo, Outlook, etc.
11   auth: {
12     user: process.env.USER_EMAIL, // Seu email
13     pass: process.env.PWD_EMAIL_GMAIL_APP // Sua senha de email ou senha de app
14   }
15 });
16
17 // Função para enviar email
18 function enviarEmailService(destinatario, assunto, mensagem) {
19   console.log('EnviarEmailService.destinatario',destinatario);
20   const mailOptions = {
21     from: process.env.USER_EMAIL, // Remetente
22     to: destinatario, // Destinatário
23     subject: assunto, // Assunto do email
24     text: mensagem // Conteúdo do email
25   };
26
27   transporter.sendMail(mailOptions, (error, info) => {
28     if (error) {
29       console.log('Erro ao enviar email:', error);
30     } else {
31       console.log('Email enviado com sucesso:', info.response);
32     }
33   });
34 }
35
36 module.exports = { enviarEmailService };
```

# ApiRouter

- ▶ No arquivo enviarMensagemController.js, criar a função enviarEmail:
  - ▶ Importar o serviço enviarEmailService.js

```
4  const { enviarEmailService } = require('../service/enviarEmailService');  
5
```

```
22  
23  exports.enviarEmail = async (req, res) => {  
24    const { destinatario, assunto, mensagem } = req.body;  
25    try {  
26      enviarEmailService(destinatario, assunto, mensagem);  
27      res.status(200).json({ success: true, message: 'Email enviado com sucesso' });  
28    } catch (err) {  
29      console.log('Erro ao enviar E-mail', err);  
30      res.status(500).json({ error: 'Erro ao enviar E-mail' });  
31    }  
32  }  
33  
34
```

# ApiRouter

- ▶ Na pasta rotas, adicionar a rota enviaremail
- ▶ Escrever o código

```
51
52 //cria a rota de enviar Email
53 /**
54  * @swagger
55  * /enviaremail:
56  *   post:
57  *     summary: Enviar mensagem via E-mail
58  *     tags: [Envio de mensagens]
59  *     requestBody:
60  *       required: true
61  *       content:
62  *         application/json:
63  *           schema:
64  *             type: object
65  *             properties:
66  *               destinatario:
67  *                 type: string
68  *                 description: Destinatario que recebera o e-mail
69  *               assunto:
70  *                 type: string
71  *                 description: Assunto do e-mail
72  *               mensagem:
73  *                 type: string
74  *                 description: Mensagem do e-mail
75  *     responses:
76  *       200:
77  *         description: E-mail enviada com sucesso
78  *         content:
79  *           application/json:
80  *             schema:
81  *               type: object
82  *               properties:
83  *                 retorno:
84  *                   type: string
85  *       400:
86  *         description: Erro ao enviar o e-mail ou e-mail inválido
87  *       500:
88  *         description: Erro ao enviar a e-mail
89  */
90 router.post('/enviaremail', enviarMensagemController.enviarEmail);
91
```

# ApiRouter

- ▶ Testar o envio de Email, pelo Swagger.

The image shows a Swagger UI interface for testing an API endpoint. The interface is divided into several sections:

- Method and Path:** A green bar at the top left shows the method **POST** and the path **/enviaremail** with the description "Enviar mensagem via E-mail".
- Parameters:** A section below the method and path, labeled "Parameters", showing "No parameters".
- Request body:** A section labeled "Request body" with a red "required" tag. It includes a dropdown menu set to "application/json".
- Request body content:** A large text area containing a JSON object: 

```
{  "destinatario": "davi.schneid@gmail.com",  "assunto": "Teste Davi",  "mensagem": "E-mail enviado com sucesso"}

```
- Buttons:** At the bottom, there are two buttons: a blue "Execute" button and a "Clear" button.

# ApiRouter

## ► Resultado do teste:

