

PRÁTICAS AVANÇADAS EM DESENVOLVIMENTO WEB

Davi Schneid - davi.Schneid@gmail.com

18/08/2024

Agenda

- ▶ Criar login com Google
- ▶ Dependências
 - ▶ `npm install passport`
 - ▶ `npm install passport-google-oauth20`
 - ▶ `npm install google-auth-library`
 - ▶ `npm install express-session`

Google

- ▶ Criar novo projeto no Console do Google, acessar <https://console.cloud.google.com/>
- ▶ No topo da página, à esquerda, clique no seletor de projeto (um menu suspenso ao lado do nome do projeto atual, se houver).

The screenshot shows the Google Cloud Console interface. At the top, there is a banner for a free trial. Below it, the Google Cloud logo is on the left, and a search bar is on the right. In the center, there is a dropdown menu for selecting a project, currently showing '3miltalentos'. A red box highlights this dropdown. Below the dropdown, the main content area shows a greeting 'Olá!' and a list of recent projects. A modal window titled 'Selecionar um projeto' is open, showing a search bar and a table of projects. A red box highlights the 'NOVO PROJETO' button in the top right corner of the modal. The table lists the project '3miltalentos' with ID 'miltalentos-d9445'.

Comece seu teste gratuito com US\$ 300 em crédito. Não se preocupe, você não será cobrado se seus créditos acabarem. [Saiba mais](#)

Google Cloud 3miltalentos

Pesquise (/) recursos, documentos, produtos e muito mais Pesquisa

Selecionar um projeto

NOVO PROJETO

Pesquisar projetos e pastas

RECENTE COM ESTRELA TODOS


Nome	ID
✓ ☆ 3miltalentos ?	miltalentos-d9445

CANCELAR


Cloud Storage BigQuery Rede VPC Kubernetes

Google


- ▶ **Nome do Projeto:** Insira um nome para o seu projeto
- ▶ **Organização (opcional):** Se você não tiver uma organização, pode deixar essa opção como está.
- ▶ **Localização:** Preencher somente se tiver uma organização.




Novo projeto

 Você tem 10 projects restantes na sua cota. Solicite um aumento ou exclua projetos. [Saiba mais](#)

[MANAGE QUOTAS](#)


Nome do projeto *
loginGoogle 

ID do projeto: crafty-day-432901-s1. Não é possível mudar depois. [EDITAR](#)


Local *
 Sem organização [PROCURAR](#)


Pasta ou organização pai

[CRIAR](#) [CANCELAR](#)



Notificações

 Criar projeto: loginGoogle Agora
[SELECIONAR PROJETO](#)

 Ativar serviço: firestore.googleapis.com há 12 dias
3mltalentos

[VER TODAS AS ATIVIDADES](#)



- ▶ Clicar em credenciais no menu a esquerda
- ▶ Depois em Criar credenciais

ComECE seu teste gratuito com US\$ 300 em crédito. Não se preocupe, você não será cobrado se seus créditos acabarem. [Saiba mais](#)

Google Cloud loginGoogle Pesquise (/) recursos, documentos, produtos e muito mais Pesquisa

API APIs e serviços Credenciais + CRIAR CREDENCIAIS EXCLUIR RESTAURAR CREDENCIAIS EXCLUÍDAS

APIs e serviços ativados

Crie credenciais para acessar suas APIs ativas. [Saiba mais](#)

Lembre-se de configurar a tela de permissão OAuth com informações sobre seu aplicativo. CONFIGURAR TELA DE CONSENTIMENTO

Chaves de API

<input type="checkbox"/>	Nome	Data da criação ↓	Restrições	Ações
Nenhuma chave de API a ser exibida				

IDs do cliente OAuth 2.0

<input type="checkbox"/>	Nome	Data da criação ↓	Tipo	ID do cliente	Ações
Nenhum cliente do OAuth a ser exibido					

Contas de serviço

<input type="checkbox"/>	E-mail	Nome ↑	Ações
Nenhuma conta de serviço para ser exibida			

[Gerenciar contas de serviço](#)

► Criar o ID do cliente OAuth

Google Cloud

loginGoogle

Pesquise (/) recursos

API APIs e serviços

APIs e serviços ativados

Credenciais

Tela de permissão OAuth

Contratos de uso de página

Um ID do cliente é usado para identificar um único app nos servidores OAuth do Google. Se o app for executado em várias plataformas, cada uma precisará de um ID do cliente. Para acessar mais informações, consulte [Como configurar o OAuth 2.0](#). [Saiba mais](#) sobre tipos de clientes OAuth.

Tipo de aplicativo *

Aplicativo da Web

Nome *

NodeLoginGoogle

O nome do cliente OAuth 2.0. Esse nome é usado apenas para identificar o cliente no console e não será mostrado aos usuários finais.

Os domínios dos URIs incluídos abaixo serão automaticamente adicionados à [tela de consentimento do OAuth](#) como [domínios autorizados](#).

Origens JavaScript autorizadas

Para usar com solicitações de um navegador

+ ADICIONAR URI

URIs de redirecionamento autorizados

Para usar com solicitações de um servidor da Web

URIs 1 *

http://localhost:3001/api/auth/google

+ ADICIONAR URI

Observação: pode levar de cinco minutos a algumas horas para que as configurações entrem em vigor

CRIAR




CANCELAR

Cliente OAuth criado

A ID do cliente e o secret estão sempre disponíveis em "Credenciais" na página "APIs e serviços".



O acesso OAuth é restrito aos [usuários de teste](#) listados na [tela de consentimento do OAuth](#).

ID do cliente	48666121220-7np8qijbotsmfqjq4kbjh8nurd97lpe0.apps.googleusercontent.com	
Chave secreta do cliente	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	
Data da criação	17 de agosto de 2024 23:03:23 GMT-3	
Status	 Ativada	

 BAIXAR O JSON

OK

API Web

► Selecione ID do Cliente OAuth

The screenshot shows the Google Cloud 'APIs e serviços' console. The left sidebar contains a menu with 'APIs e serviços', 'APIs e serviços ativados', 'Biblioteca', 'Credenciais' (highlighted), 'Tela de permissão OAuth', and 'Contratos de uso de página'. The main area is titled 'Credenciais' and includes buttons for '+ CRIAR CREDENCIAIS', 'EXCLUIR', and 'RESTAURAR CREDENCIAIS EXCLUÍDAS'. A dropdown menu is open from the '+ CRIAR CREDENCIAIS' button, showing three options: 'Chave de API' (Identifica seu projeto usando uma chave de API simples para verificar cota e acesso), 'ID do cliente OAuth' (Solicita a permissão do usuário para que o aplicativo possa acessar os dados desse usuário), and 'Conta de serviço' (Usa contas robô para ativar a autenticação do nível do app entre servidores). The 'ID do cliente OAuth' option is highlighted with a red box. Below the dropdown, there are sections for 'Chaves de API', 'IDs do cliente OAuth', and 'Contas de serviço', each with a table header and a 'Nenhuma' (None) entry.

Google Cloud loginGoogle Pesquise (/) recursos, documentos, produtos e muito mais Pesquisa

APIs e serviços

- APIs e serviços ativados
- Biblioteca
- Credenciais**
- Tela de permissão OAuth
- Contratos de uso de página

Credenciais + CRIAR CREDENCIAIS EXCLUIR RESTAURAR CREDENCIAIS EXCLUÍDAS

Crie credenciais para acessar APIs

Chaves de API

- ☐ Nome
- Nenhuma chave de API para ser exibida

IDs do cliente OAuth

- ☐ Nome
- Nenhum cliente do OAuth a ser exibido

Contas de serviço

- ☐ E-mail
- Nenhuma conta de serviço para ser exibida

Nome ↑

Restrições

Tipo

- ▶ Configurar OAuth Consent Screen
- ▶ Vá para "APIs e serviços" > "Tela de consentimento OAuth".
- ▶ Preencha os campos obrigatórios, como o nome da aplicação e as informações de contato.

Google Cloud loginGoogle Pesquise (/) recursos, documentos, produ

API APIs e serviços

APIs e serviços ativados

Biblioteca

Credenciais

Tela de permissão OAuth

Contratos de uso de página

Editar registro do app

1 Tela de permissão OAuth — 2 Escopos — 3 Usuários de teste — 4 Resumo

Informações do app

Essas informações são exibidas na tela de consentimento. Elas informam aos usuários finais quem é você e como entrar em contato.

Nome do app *
loginGoogle

E-mail para suporte do usuário *
davi.schneid@gmail.com

Para que os usuários entrem em contato com você a respeito do consentimento. [Saiba mais](#)

Logotipo do app

Este é seu logotipo. Ele ajuda as pessoas a reconhecerem o app e aparece em destaque na tela de permissão OAuth.

Depois de fazer upload de um logotipo, será necessário enviar o app para verificação, a menos que esteja configurado para uso interno ou tenha o status de publicação "Teste". [Saiba mais](#)

Arquivo de logotipo a fazer upload [PROCURAR](#)

Faça upload de uma imagem de até 1 MB na tela de permissão para ajudar os usuários a reconhecerem seu app. Os formatos de imagem permitidos são JPG, PNG e BMP. Para garantir o melhor resultado, os logotipos precisam ser quadrados e ter a resolução de 120 px por 120 px.

Domínio do app

Para proteger você e seus usuários, o Google permite apenas os apps que utilizam o OAuth a usar os domínios autorizados. As informações a seguir serão exibidas aos seus usuários na tela de consentimento.

Clicar em salvar

APIRouter

- ▶ Efetuar as instalações das dependências no projeto.
- ▶ Criar uma pasta google na raiz do projeto, dentro desta pasta criar o arquivo `passaporteGoogle.js` escrever o código abaixo.

```
google > JS passaporteGoogle.js > <function>
1  const passport = require('passport');
2  const GoogleStrategy = require('passport-google-oauth20').Strategy;
3
4  passport.use(new GoogleStrategy({
5    clientID: process.env.GOOGLE_CLIENT_ID,
6    clientSecret: process.env.GOOGLE_CLIENT_SECRET,
7    callbackURL: "http://localhost:3001/api/auth/google/callback"
8  },
9    function(accessToken, refreshToken, profile, done) {
10     return done(null, profile, { accessToken });
11   }
12 ));
13
14 passport.serializeUser((user, done) => {
15   done(null, user);
16 });
17
18 passport.deserializeUser((user, done) => {
19   done(null, user);
20 });
```


APIRouter

- ▶ Adicionar o cliente e secretId do google no .env

```
GOOGLE_CLIENT_ID=4866[REDACTED]apps.googleusercontent.com  
GOOGLE_CLIENT_SECRET=[REDACTED]
```

APIRouter

- Criar o arquivo googleController.js na pasta controllers, escrever o código:

```
controllers > JS googleController.js > ...
1  
2  const passport = require('passport');
3
4  // Autentica o usuario no google
5  exports.autenticarGoogle = (req, res, next) => {
6    console.log('Iniciando autenticação com Google');
7    passport.authenticate('google', { scope: ['profile', 'email'] })(req, res, next);
8  };
9
10
11 // Efetua o callback da autenticação do usuario no google
12 exports.callBackAutenticacaoGoogle = (req, res, next) => {
13
14   passport.authenticate('google', { failureRedirect: '/login' }, (err, user, info) => {
15     if (err) {
16       return next(err);
17     }
18     if (!user) {
19       return res.redirect('/login');
20     }
21     // O token estará disponível em info.accessToken
22     const token = info.accessToken;
23     console.log('Token gerado no Google:', token);
24
25     // Loga o usuário
26     req.login(user, (err) => {
27       if (err) {
28         return next(err);
29       }
30       // Redireciona para o frontend com o token
31       res.redirect(`http://localhost:3000/auth/google/callback?token=${token}`);
32     });
33   })(req, res, next);
34 };
35
36
37 // Efetuar o logout do usuario
38 exports.logoutGoogle = async (req, res) => {
39   req.logout((err) => {
40     if (err) { return next(err); }
41     res.redirect('/');
42   });
43 };
44
45
```

APIRouter

- Criar o arquivo googleRotas.js na pasta rotas, escrever o código:

```
rotas > JS googleRotas.js > ...
 1 //Importa o modulo Express
 2 const express = require('express');
 3
 4 //Cria o objeto rotas
 5 const router = express.Router();
 6
 7 //Importa o modulo usuarioController
 8 const googleController = require('../controllers/googleController');
 9
10 /**
11  * @swagger
12  * /auth/google:
13  *   get:
14  *     tags:
15  *       - Autenticação
16  *     summary: Autenticação com Google
17  *     description: Inicia o processo de autenticação com o Google.
18  *     responses:
19  *       302:
20  *         description: Redireciona para o Google para autenticação.
21  *       500:
22  *         description: Erro no servidor.
23  */
24 router.get('/auth/google', googleController.autenticarGoogle);
25
26
27 /**
28  * @swagger
29  * /auth/google/callback:
30  *   get:
31  *     tags:
32  *       - Autenticação
33  *     summary: Callback da Autenticação com Google
34  *     description: Rota de callback para tratar o retorno da autenticação com o Google.
35  *     responses:
36  *       200:
37  *         description: Autenticação bem-sucedida. Redireciona para a página principal.
38  *       401:
39  *         description: Falha na autenticação.
40  *       500:
41  *         description: Erro no servidor.
42  */
43 router.get('/auth/google/callback', googleController.callBackAutenticacaoGoogle);
44
```

APIRouter

- Continuação do arquivo googleRotas.js na pasta rotas, escrever o código:

```
45
46  /**
47   * @swagger
48   * /logout:
49   *   get:
50   *     tags:
51   *       - Autenticação
52   *     summary: Logout do Google
53   *     description: Realiza o logout do usuário autenticado via Google.
54   *     responses:
55   *       200:
56   *         description: Logout bem-sucedido. Redireciona para a página inicial.
57   *       500:
58   *         description: Erro no servidor.
59   */
60  router.get('/logout', googleController.logoutGoogle);
61
62  /**
63   * @swagger
64   * /protegerrotas:
65   *   get:
66   *     tags:
67   *       - Autenticação
68   *     summary: Protege rotas
69   *     description: Verifica se o usuário está autenticado para acessar rotas protegidas.
70   *     responses:
71   *       200:
72   *         description: Acesso permitido. O usuário está autenticado.
73   *       401:
74   *         description: Acesso negado. O usuário não está autenticado.
75   *       500:
76   *         description: Erro no servidor.
77   *
78  router.get('/protegerrotas', googleController.protegerRotas);
79
80
81  //exporta as rotas criadas
82  module.exports = router;
```

APIRouter

- Configurar o `passaporteGoogle.js` no arquivo `main` da aplicação

```
12 require('./google/passaporteGoogle'); // Importa a configuracao do Passport do google
13
14 //Importar o modulo Swagger
15 const setupSwagger = require('./swagger');
16
17 //importar o modulo cors para receber requisicoes de diferente origem
18 const cors = require('cors');
19
20
21 const app = express();
22 const PORT = process.env.PORT;
23
24
25 app.use(require("cors")());
26
27 app.use(session({
28   secret: process.env.SECRET_ID_APP,
29   resave: false,
30   saveUninitialized: true
31 }));
32
33 app.use(passport.initialize());
34 app.use(passport.session());
35
```

- Adicionar a rota do google no arquivo `main` da aplicação

```
48
49 app.use(express.json());
50 app.use('/api', usuariosRotas);
51 app.use('/api', uploadArquivoRotas);
52 app.use('/api', validarToken);
53 app.use('/api', enviarMensagem);
54 app.use('/api', google);
55
```

APIRouter

- Adicionar a validação do token do Google no tokenController.js

controllers > JS tokenController.js > validaToken > validaToken

```
1  const jwt = require('jsonwebtoken');
2  const axios = require('axios');
3
4  exports.validaToken = async (req, res) => {
5    const { token } = req.body;
6
7    try {
8      if (!token) {
9        return res.status(400).json({ valid: false });
10      }
11
12      // Aguarda a verificação do token Google
13      const httpstatus = await verificarTokenGoogle(token);
14
15      //se o token do google for valido nao valida o token da aplicacao
16      if (httpstatus === 200) {
17        return res.status(200).json({ valid: true });
18      } else {
19        // Verifica o token usando JWT se o token do Google não for válido
20        jwt.verify(token, process.env.JWT_KEY, (err, decoded) => {
21          if (err) {
22            return res.status(401).json({ valid: false });
23          } else {
24            return res.status(200).json({ valid: true });
25          }
26        });
27      }
28    } catch (err) {
29      return res.status(500).json({ valid: false, error: 'Erro interno do servidor' });
30    }
31  };
32
33  async function verificarTokenGoogle(token) {
34    try {
35      const response = await axios.get(`https://www.googleapis.com/oauth2/v1/tokeninfo?access_token=${token}`);
36      return 200; // Se o token for válido, retorna 200
37    } catch (error) {
38      console.error('Erro ao verificar o token:', error.response?.data || error.message);
39      return 401; // Se o token for inválido, retorna 401
40    }
41  }
42
```


APIWeb

- ▶ Editar a página login.js adicionando o botão do Login com o Google

```
// Função para redirecionar para o Google login
const loginWithGoogle = () => {
  window.location.href = 'http://localhost:3001/api/auth/google';
  const urlParams = new URLSearchParams(window.location.search);
  console.log('urlParams', urlParams);
};
```

```
<div className="button-container">
  <button onClick={loginWithGoogle} className="button google-button">
    <img src={googleIcon} style={{ width: '32px', height: '32px', marginRight: '8px' }} />
    <span>Login com Google</span>
  </button>
</div>
<br />
```

APIWeb

- ▶ Editar a página login.js adicionando o botão do Login com o Google
- ▶ Adiciona o css para o botão do login.

```
// Função para redirecionar para o Google login
const loginWithGoogle = () => {
  window.location.href = 'http://localhost:3001/api/auth/google';
  const urlParams = new URLSearchParams(window.location.search);
  console.log('urlParams', urlParams);
};
```


```
<div className="button-container">
  <button onClick={loginWithGoogle} className="button google-button">
    <img src={googleIcon} style={{ width: '32px', height: '32px', marginRight: '8px' }} />
    <span>Login com Google</span>
  </button>
</div>
<br />
```


```
111 .google-button {
112   display: flex;
113   align-items: center;
114   background-color: white;
115   color: #4285F4;
116   border: 1px solid #4285F4;
117   border-radius: 4px;
118   padding: 10px 20px;
119   cursor: pointer;
120   font-size: 16px;
121   justify-content: center;
122 }
123 .google-button:hover {
124   background-color: #f8f9fa;
125 }
126
127 .google-button svg {
128   margin-right: 8px;
129 }
```

APIWeb

- ▶ A página do login ficará assim.


Login

 E-mail do usuario

 Senha

[Esqueceu a senha?](#)

Login

 Login com Google

[Acessar cadastro](#)

APIWeb

- Criar o componente GoogleCallback.js na pasta componentes.

```
src > componentes > JS GoogleCallback.js > [🔗] default
1  import React, { useEffect, useContext, useRef } from 'react';
2  import { useNavigate } from 'react-router-dom';
3  import { AuthContext } from '../autenticacao/autenticacao';
4
5  const GoogleCallback = () => {
6
7      const { setAuthToken } = useContext(AuthContext); // useContext para setar o token no contexto
8      const navigate = useNavigate();
9
10     // Ref para garantir que o useEffect só execute uma vez
11     const hasExecuted = useRef(false);
12
13     useEffect(() => {
14         if (hasExecuted.current) return; // Se já executou, não faz nada
15         hasExecuted.current = true;
16
17         // Captura o token da URL
18         const urlParams = new URLSearchParams(window.location.search);
19         const token = urlParams.get('token');
20
21         if (token) {
22             setAuthToken(token); // Seta o token no AuthContext
23             localStorage.setItem('token', token); // Salva o token no localStorage
24             navigate('/home'); // Redireciona para a página inicial
25         } else {
26             navigate('/login'); // Redireciona para o login se algo deu errado
27         }
28     }, [setAuthToken, navigate]); // Remove authToken da lista de dependências
29
30     return <div>Carregando...</div>; // Renderiza algo enquanto o efeito está sendo processado
31 };
32
33 export default GoogleCallback;
```

APIWeb

- Adiciona a rota de GoogleBallback no arquivo de Rotas.js

```
24  function Rotas() {
25    return (
26      <AuthProvider>
27      <BrowserRouter>
28      <Routes>
29        <Route path="/cadastro" element={<Cadastro />} />
30        <Route path="/login" element={<Login />} />
31        <Route path="/" element={<Login />} />
32        <Route path="/auth/google/callback" element={<GoogleCallback />} />
33        <Route path="/esqueci-minha-senha" element={<EsqueciMinhaSenha />} />
34        <Route path="/resetar-senha/:token" element={<ResetarSenha />} />
35        <Route path="/homematerials" element={<HomeMaterials />} />
36
37        <Route path="/home" element={<PrivateRoute />>
38          <Route path="/home" element={<Home />} />
39        </Route>
40
41        <Route path="/lista" element={<PrivateRoute />>
42          <Route path="/lista" element={<ListaRegistros />} />
43        </Route>
44
45        <Route path="/editar/:id" element={<PrivateRoute />>
46          <Route path="/editar/:id" element={<EditarRegistro />} />
47        </Route>
48
49        <Route path="/upload" element={<PrivateRoute />>
50          <Route path="/upload" element={<Upload />} />
51        </Route>
52      </Routes>
53    </BrowserRouter>
54  </AuthProvider>
55  )
56 }
57 export default Rotas;
```

APIWeb

- Ajuste no controle do token no arquivo rotasPrivadas.js, se o token for valido

```
src > autenticacao > JS rotasPrivadas.js > PrivateRoute > useEffect() callback
1
2 import React, { useContext, useEffect, useState } from 'react';
3 import { Navigate, Outlet } from 'react-router-dom';
4 import { AuthContext } from '../autenticacao';
5 import axios from 'axios';
6
7 const PrivateRoute = () => {
8
9     const { authToken } = useContext(AuthContext);
10     const [isValid, setIsValid] = useState(null);
11
12     const verifyToken = async () => {
13
14         try {
15             const response = await axios.post('http://localhost:3001/api/validarToken', { token: authToken });
16             setIsValid(response.data.valid);
17             if(isValid){
18                 localStorage.setItem('token', authToken);
19             }
20         } catch {
21             localStorage.removeItem('token');
22             setIsValid(false);
23         }
24     };
25
26     verifyToken();
27
28     useEffect(() => {
29         if (authToken) {
30             console.log('Token valido !!!! ');
31         } else {
32             localStorage.removeItem('token');
33             setIsValid(false);
34         }
35     }, [authToken]);
36
37     if (isValid === null) {
38         return <div>Loading...</div>;
39     }
40
41     return isValid ? <Outlet /> : <Navigate to="/login" />;
42 };
43
44 export default PrivateRoute;
```

APIWeb

- ▶ Executar a aplicação e testar o login