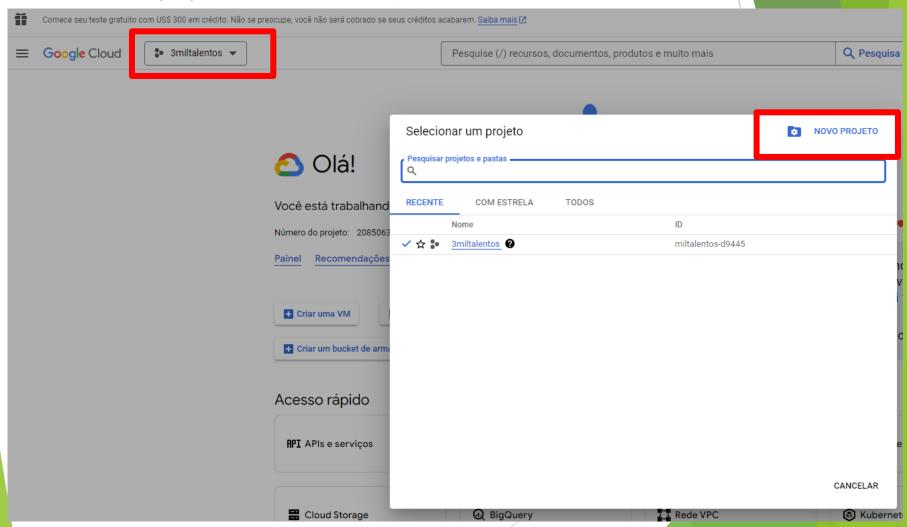
PRÁTICAS AVANÇADAS EM DESENVOLVIMENTO WEB

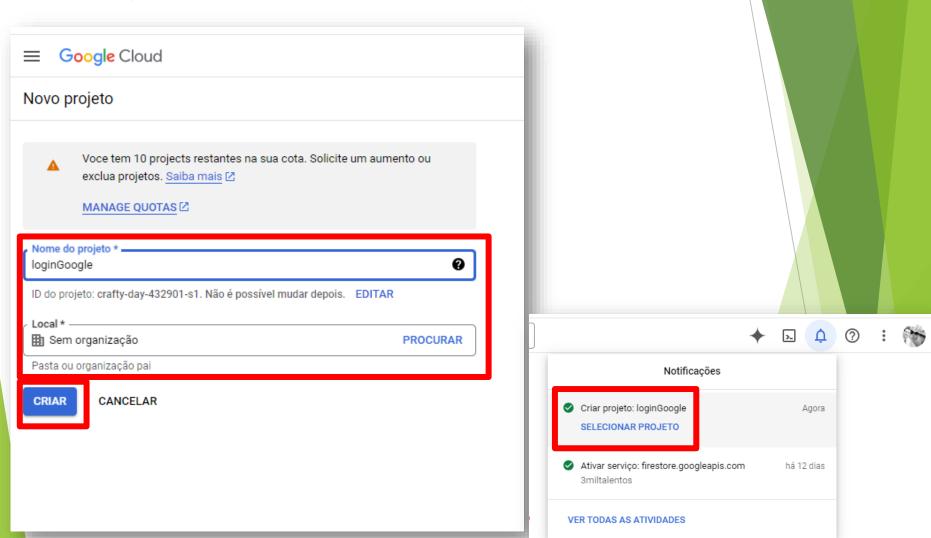
Agenda

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

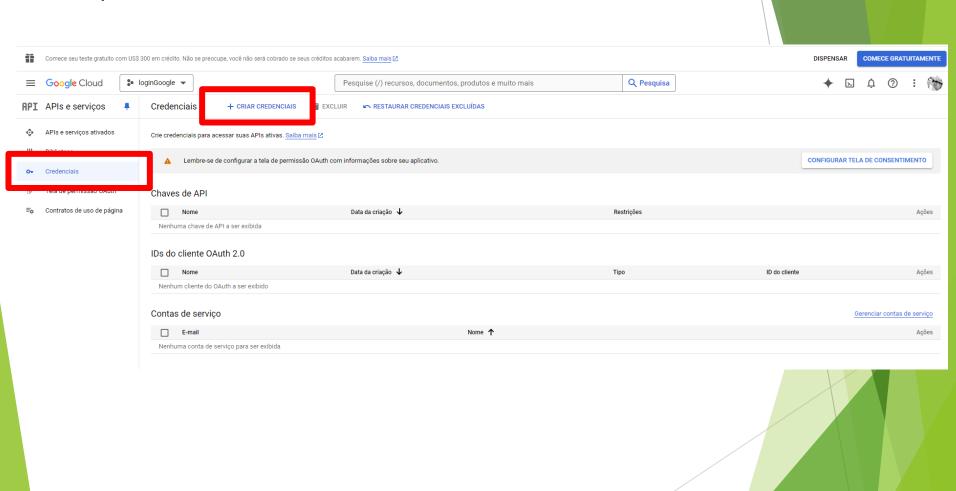
- Criar novo projeto no Console do Google, acessar https://console.cloud.google.com/
- No topo da página, à esquerda, clique no seletor de projeto (um menu suspenso ao lado do nome do projeto atual, se houver).



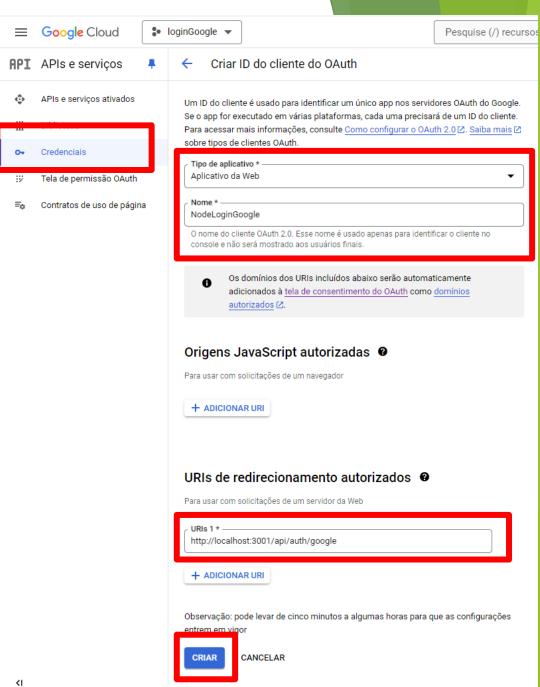
- Nome do Projeto: Insira um nome para o seu projeto
- Organização (opcional): Se você não tiver uma organização, pode deixar essa opção como está.
- Localização: Preencher somente se tiver uma organização.



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



Criar o ID do cliente OAuth



Cliente OAuth criado

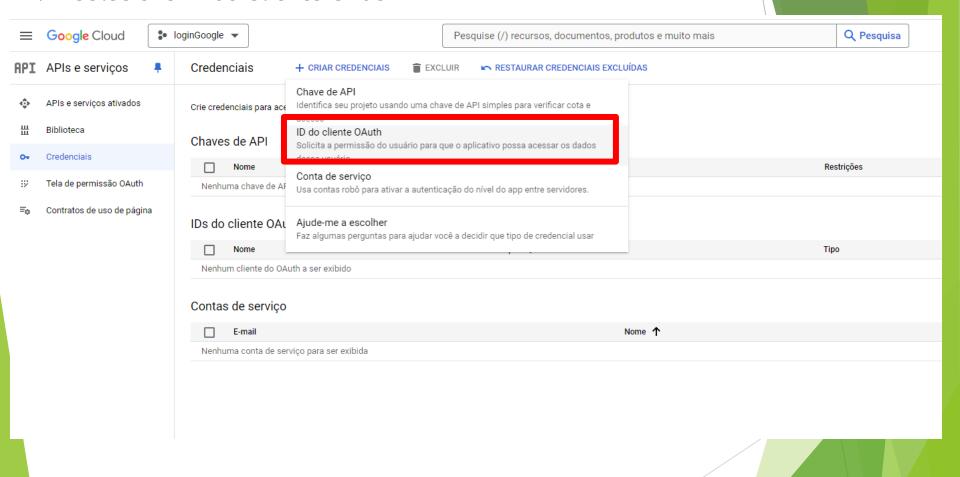
A ID do cliente e o secret estão sempre disponíveis em "Credenciais" na página "APIs e serviços".



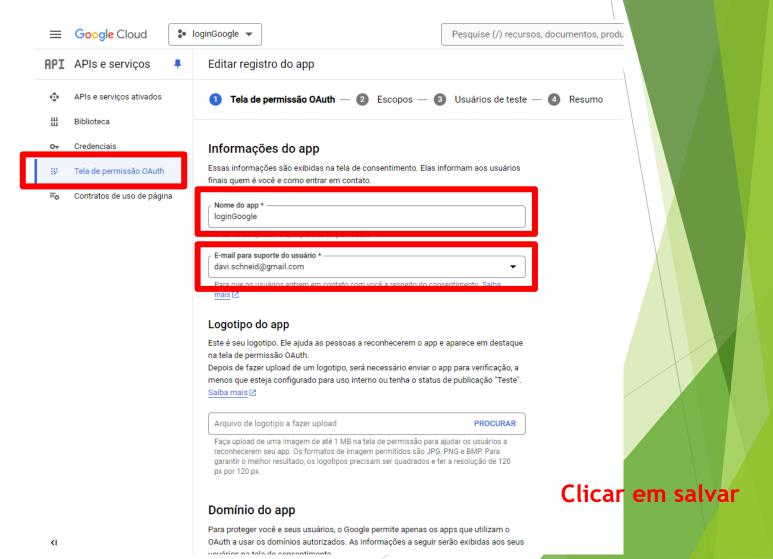
O acesso OAuth é restrito aos $\underline{usu\'{arios}}$ de teste ${}^{\c C}$ listados na \underline{tela} de consentimento do OAuth.

ID do cliente	48666121220- 7np8qijbotsmfqjq4kbjh8nurd97lpe0.apps. googleusercontent.com
Chave secreta do cliente	GOCSPX- rH8z7dOv95Yx6VzzS_lyqK9bVTSL
Data da criação	17 de agosto de 2024 23:03:23 GMT-3
Status	Ativada
★ BAIXAR O JSON	

Selecione ID do Cliente OAuth



- 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.



- 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>
      const passport = require('passport');
      const GoogleStrategy = require('passport-google-oauth20').Strategy;
      passport.use(new GoogleStrategy({
          clientID: process.env.GOOGLE CLIENT ID,
          clientSecret: process.env.GOOGLE CLIENT SECRET,
          callbackURL: "http://localhost:3001/api/auth/google/callback"
        function(accessToken, refreshToken, profile, done) {
  9
          return done(null, profile,{ accessToken });
      ));
      passport.serializeUser((user, done) => {
        done(null, user);
      });
      passport.deserializeUser((user, done) => {
        done(null, user);
      });
```

Adicionar o cliente e secretId do google no .env



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

```
controllers > JS googleController.js > ...
      const passport = require('passport');
      // Autentica o usuario no google
      exports.autenticarGoogle = (req, res, next) => {
           console.log('Iniciando autenticação com Google');
           passport.authenticate('google', { scope: ['profile', 'email'] })(req, res, next);
      // Efetua o callback da autenticacao do usuario no google
      exports.callBackAutenticacaoGoogle = (req, res, next) => {
          passport.authenticate('google', { failureRedirect: '/login' }, (err, user, info) => {
               if (err) {
                   return next(err);
               if (!user) {
                   return res.redirect('/login');
              // O token estará disponível em info.accessToken
               const token = info.accessToken;
               console.log('Token gerado no Google:', token);
              // Loga o usuário
               req.logIn(user, (err) => {
                   if (err) {
                       return next(err);
                   // Redireciona para o frontend com o token
                   res.redirect(`http://localhost:3000/auth/google/callback?token=${token}`);
          })(req, res, next);
      // Efetuar o logout do usuario
      exports.logoutGoogle = async (req, res) => {
           req.logout((err) => {
               if (err) { return next(err); }
               res.redirect('/');
```

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

```
rotas > JS googleRotas.js > ...
       //Importa o modulo Express
       const express = require('express');
       const router = express.Router();
       //Importa o modulo usuarioController
       const googleController = require('../controllers/googleController');
              summary: Autenticação com Google
                  description: Redireciona para o Google para autenticação.
       router.get('/auth/google', googleController.autenticarGoogle);
        * @swagger
                  description: Autenticação bem-sucedida. Redireciona para a página principal.
                  description: Falha na autenticação.
                  description: Erro no servidor.
       router.get('/auth/google/callback', googleController.callBackAutenticacaoGoogle);
```

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

```
summary: Logout do Google
           description: Logout bem-sucedido. Redireciona para a página inicial.
           description: Erro no servidor.
router.get('/logout', googleController.logoutGoogle);
       summary: Protege rotas
      description: Verifica se o usuário está autenticado para acessar rotas protegidas.
           description: Acesso permitido. O usuário está autenticado.
           description: Acesso negado. O usuário não está autenticado.
           description: Erro no servidor.
router.get('/protegerrotas', googleController.protegerRotas);
//exporta as rotas criadas
module.exports = router;
```

Configurar o passaporteGoogle.js no arquivo main da aplicação

```
require('./google/passaporteGoogle'); // Importa a configuracao do Passport do google
//Importar o modulo Swagger
const setupSwagger = require('./swagger');
//importar o modulo cors para receber requisicoes de diferente origem
const cors = require('cors');
const app = express();
const PORT = process.env.PORT;
app.use(require("cors")());
app.use(session({
  secret: process.env.SECRET_ID_APP,
  resave: false,
  saveUninitialized: true
}));
app.use(passport.initialize());
app.use(passport.session());
```

Adicionar a rota do google no arquivo main da aplicação

```
app.use(express.json());
app.use('/api', usuariosRotas);
app.use('/api', uploadArquivoRotas);
app.use('/api', validarToken);
app.use('/api', enviarMensagem);
app.use('/api', google);
```

Adicionar a validação do token do Google no tokenController.js

```
controllers > JS tokenController.js > ♥ validaToken > ♥ validaToken
      const jwt = require('jsonwebtoken');
       const axios = require('axios');
      exports.validaToken = async (req, res) => {
           const { token } = req.body;
          try {
               if (!token) {
                   return res.status(400).json({ valid: false });
               // Aguarda a verificação do token Google
               const httpstatus = await verificarTokenGoogle(token);
               //se o token do google for valido nao valida o token da aplicacao
               if (httpstatus === 200) {
                   return res.status(200).json({ valid: true });
               } else {
                   // Verifica o token usando JWT se o token do Google não for válido
                   jwt.verify(token, process.env.JWT KEY, (err, decoded) => {
                       if (err) {
                           return res.status(401).json({ valid: false });
                           return res.status(200).json({ valid: true });
 28
             catch (err) {
               return res.status(500).json({ valid: false, error: 'Erro interno do servidor' });
      async function verificarTokenGoogle(token) {
          try {
               const response = await axios.get(`https://www.googleapis.com/oauth2/v1/tokeninfo?access_token=${token}`);
               return 200; // Se o token for válido, retorna 200
           } catch (error) {
              console.error('Erro ao verificar o token:', error.response?.data || error.message);
               return 401; // Se o token for inválido, retorna 401
```

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

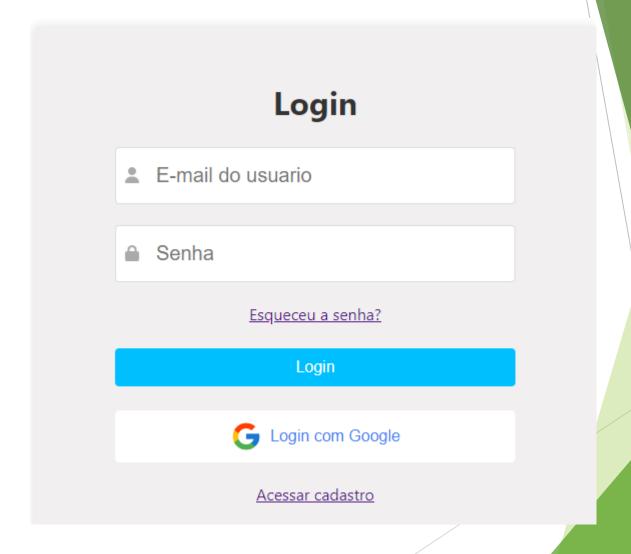
```
// Função para redirecionar para o Google login
const loginWithGoogle = () => {
    window.location.href = 'http://localhost:3001/api/auth/google';
    const urlParams = new URLSearchParams(window.location.search);
    console.log('urlParams', urlParams);
};
```

- 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);
};
```

```
.google-button {
          display: flex;
          align-items: center;
          background-color: white;
          color: #4285F4;
          border: 1px solid ■#4285F4;
          border-radius: 4px;
          padding: 10px 20px;
          cursor: pointer;
          font-size: 16px;
          justify-content: center;
      .google-button:hover {
          background-color: ■#f8f9fa;
127
      .google-button svg {
          margin-right: 8px;
```

A página do login ficará assim.



Criar o componente GoogleCallback.js na pasta componentes.

```
src > componentes > JS GoogleCallback.js > [∅] default
      import React, { useEffect, useContext, useRef } from 'react';
      import { useNavigate } from 'react-router-dom';
      import { AuthContext } from '../autenticacao/autenticacao';
      const GoogleCallback = () => {
          const { setAuthToken } = useContext(AuthContext); // useContext para setar o token no contexto
          const navigate = useNavigate();
          // Ref para garantir que o useEffect só execute uma vez
          const hasExecuted = useRef(false);
          useEffect(() => {
              if (hasExecuted.current) return; // Se já executou, não faz nada
              hasExecuted.current = true;
              // Captura o token da URL
              const urlParams = new URLSearchParams(window.location.search);
              const token = urlParams.get('token');
              if (token) {
                  setAuthToken(token); // Seta o token no AuthContext
                  localStorage.setItem('token', token); // Salva o token no localStorage
                  navigate('/home'); // Redireciona para a página inicial
              } else {
                  navigate('/login'); // Redirectiona para o login se algo deu errado
          }, [setAuthToken, navigate]); // Remove authToken da lista de dependências
          return <div>Carregando...</div>; // Renderiza algo enquanto o efeito está sendo processado
      export default GoogleCallback;
 33
```

Adiciona a rota de GoogleBallback no arquivo de Rotas.js

```
24 v function Rotas() {
         return (
             <AuthProvider>
             <BrowserRouter>
                 <Routes>
                     <Route path="/cadastro" element={<Cadastro />} />
                     <Route path="/login" element={<Login />} />
                     <Route path="/auth/google/callback" element={<GoogleCallback />} />
                     <koute path= /esqueci-minna-senna element={<esqueciminnasenna />} />
                     <Route path="/resetar-senha/:token" element={<ResetarSenha />} />
                     <Route path="/homematerials" element={<HomeMaterials />} />
37 V
                     <Route path="/home" element={<PrivateRoute />}>
                         <Route path="/home" element={<Home />} />
                     </Route>
                     <Route path="/lista" element={<PrivateRoute />}>
                         <Route path="/lista" element={<ListaRegistros />} />
                     </Route>
                     <Route path="/editar/:id" element={<PrivateRoute />}>
                         <Route path="/editar/:id" element={<EditarRegistro />} />
                     </Route>
                     <Route path="/upload" element={<PrivateRoute />}>
                         <Route path="/upload" element={<Upload />} />
                     </Route>
                 </Routes>
             </BrowserRouter>
         </AuthProvider>
     export default Rotas;
```

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

```
src > autenticacao > 🧦 rotasPrivadas.js > ខ PrivateRoute > 😚 useEffect() callback
      import React, { useContext, useEffect, useState } from 'react';
      import { Navigate, Outlet } from 'react-router-dom';
      import { AuthContext } from './autenticacao';
      import axios from 'axios';
      const PrivateRoute = () => {
          const [isValid, setIsValid] = useState(null);
                  const response = await axios.post('http://localhost:3001/api/validarToken', { token: authToken });
                  setIsValid(response.data.valid);
                   if(isValid){
                       localStorage.setItem('token', authToken);
                  localStorage.removeItem('token');
                   setIsValid(false);
          verifyToken();
          useEffect(() => {
               if (authToken) {
                   console.log('Token valido !!!! ');
                else {
 32
                   localStorage.removeItem('token');
                  setIsValid(false);
          }, [authToken]);
          if (isValid === null) {
               return <div>Loading...</div>;
          return isValid ? <Outlet /> : <Navigate to="/login" />;
      export default PrivateRoute;
```

Executar a aplicação e testar o login