**Configuración Servidor ExpressJS – MongoDB**

**Paso 1**:

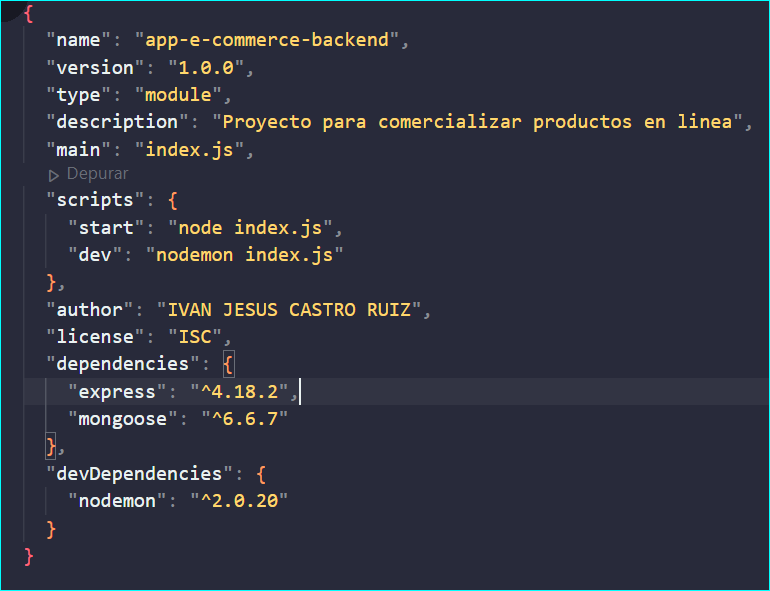
Inicializamos el proyecto:

**Comando**: npm i

**Paso 2**:

Instalamos ExpressJS

Configuramos el package.json para trabajar con import



**Comando**: npm i express

Iniciamos la configuración del servidor, e instalamos todas las dependencias necesarias tanto de desarrollo y producción.

**Comando**: npm i - - save-dev nodemon (dependencia de desarrollo)

**Comando**: npm i mongoose (dependencia de producción)

Configuramos las variables de entorno

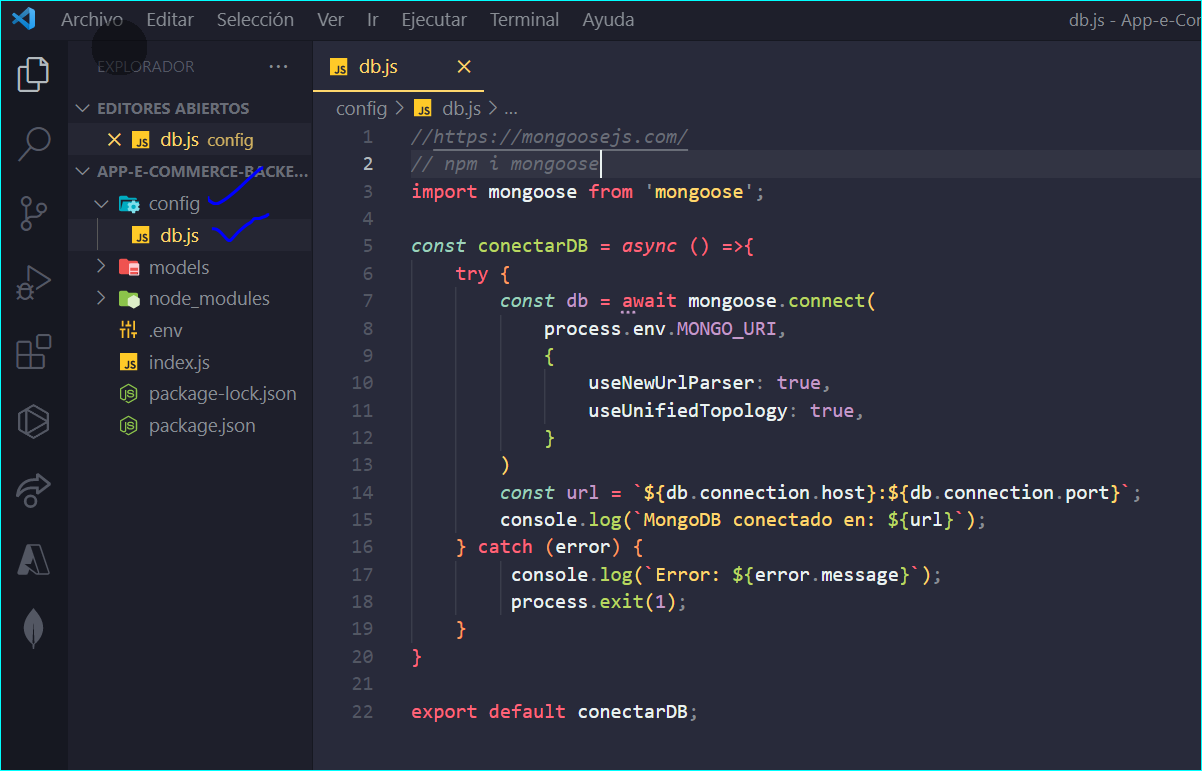
**Comando**: npm i dotenv

Creamos un archivo **.env**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Archivo **db.js** (Conexión a la BD)



Archivo **index.js**

Texto

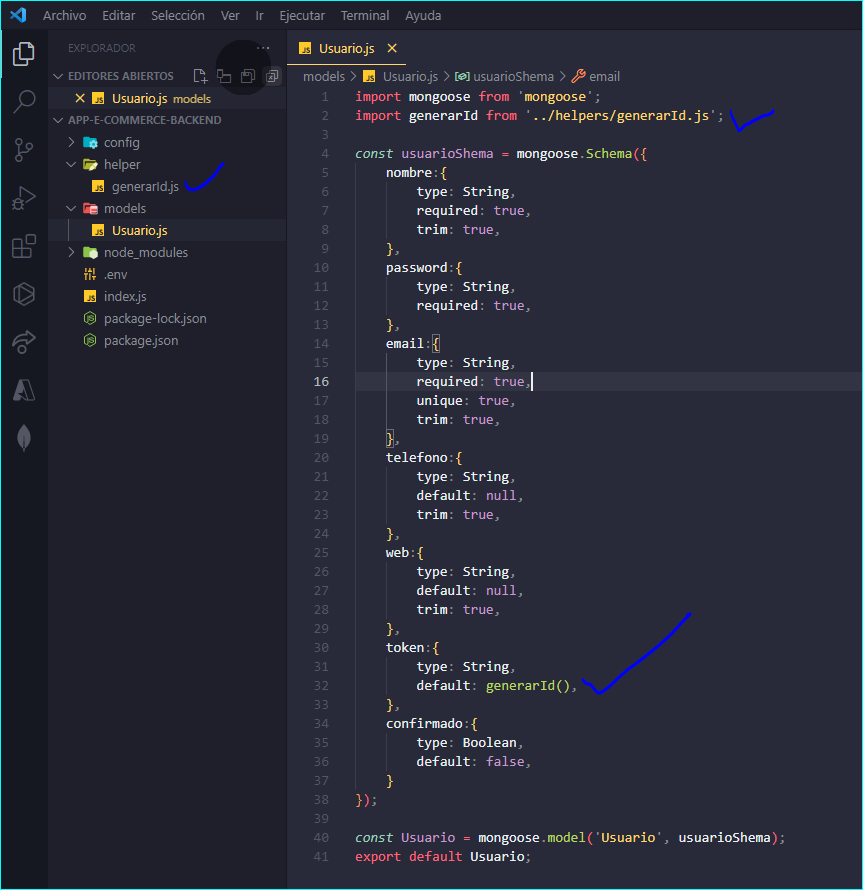
Descripción generada automáticamente

Con esto ya debe estar inicializado el servidos y conectado en BD

Texto

Descripción generada automáticamente

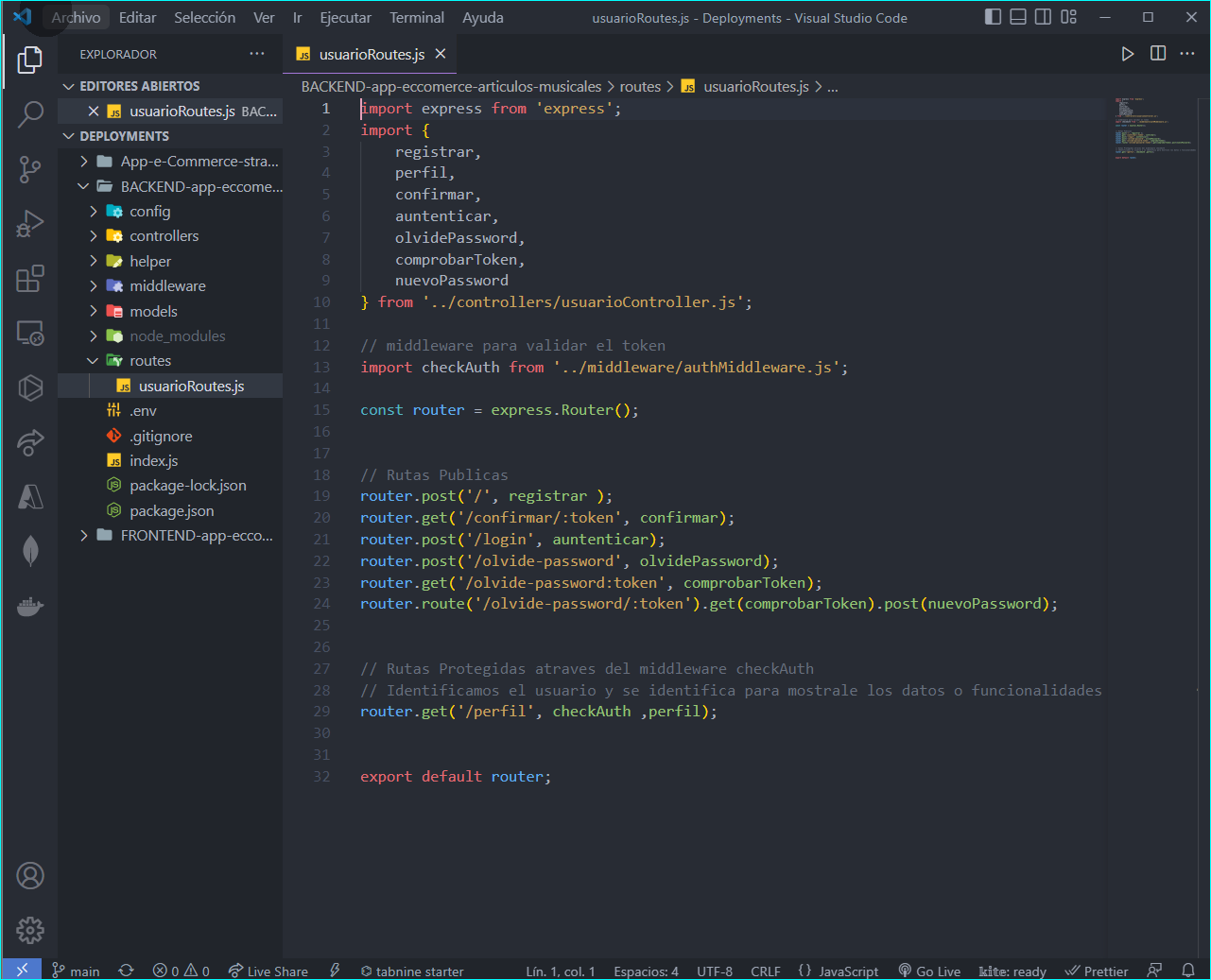
Ahora definimos el modelo



Captura de pantalla de computadora

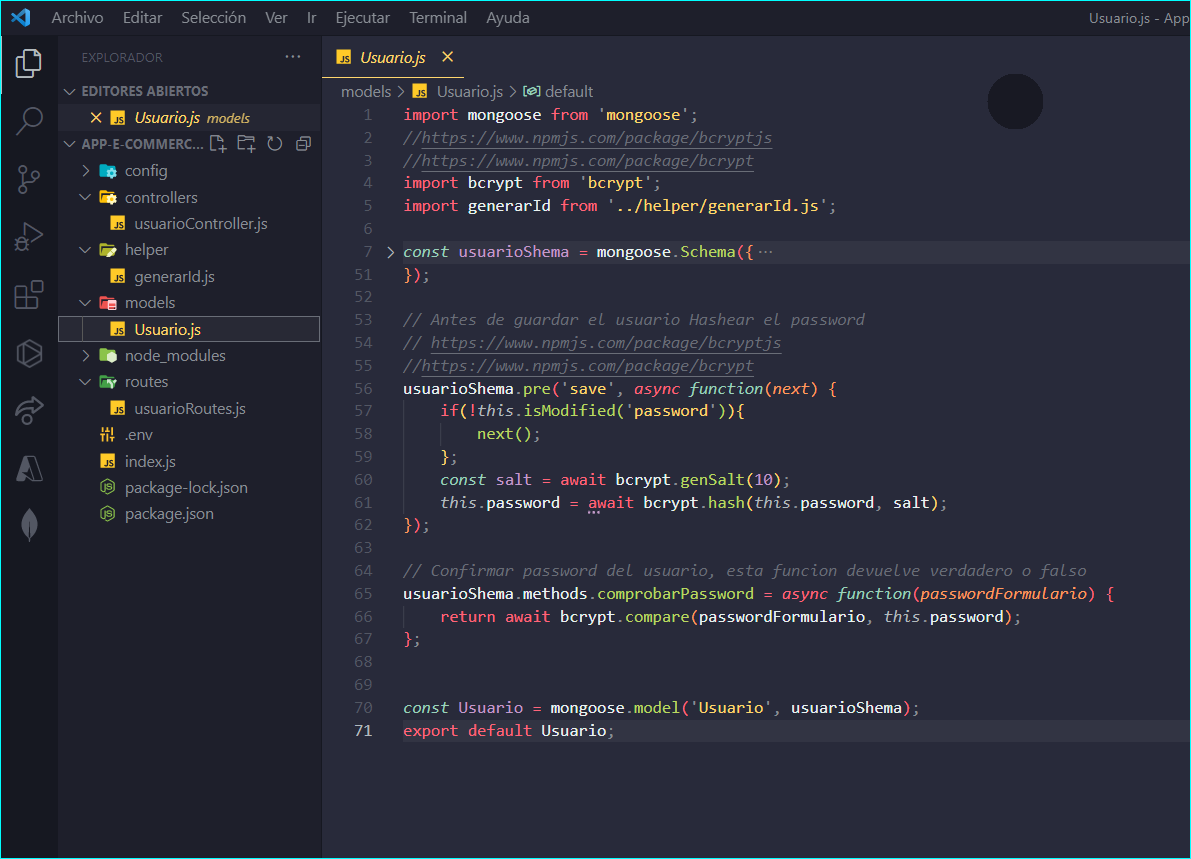
Descripción generada automáticamente

**Inicializamos las rutas**

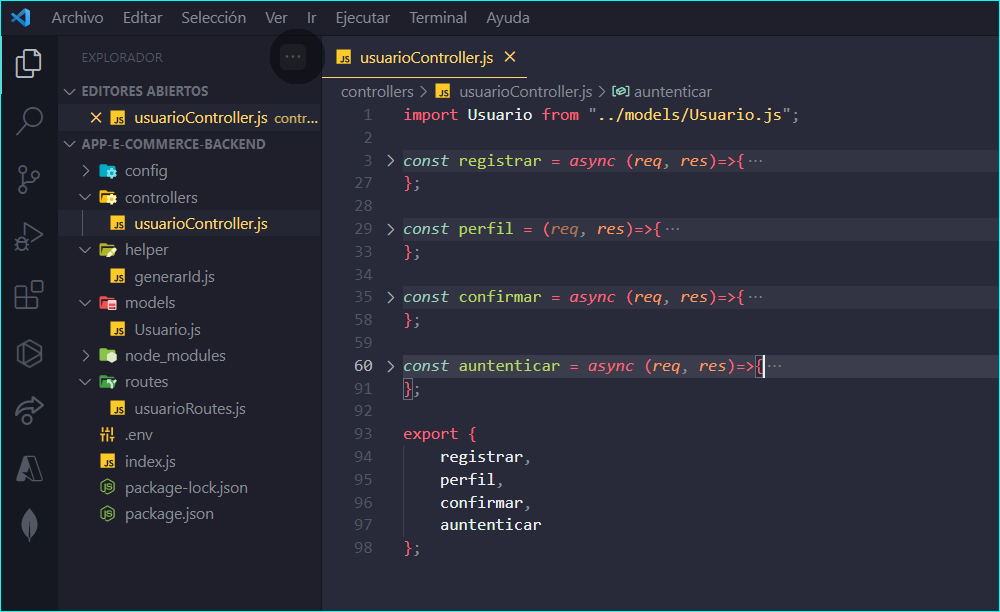


**Hashear Passwords**

**Comando**: npm i bcrypt

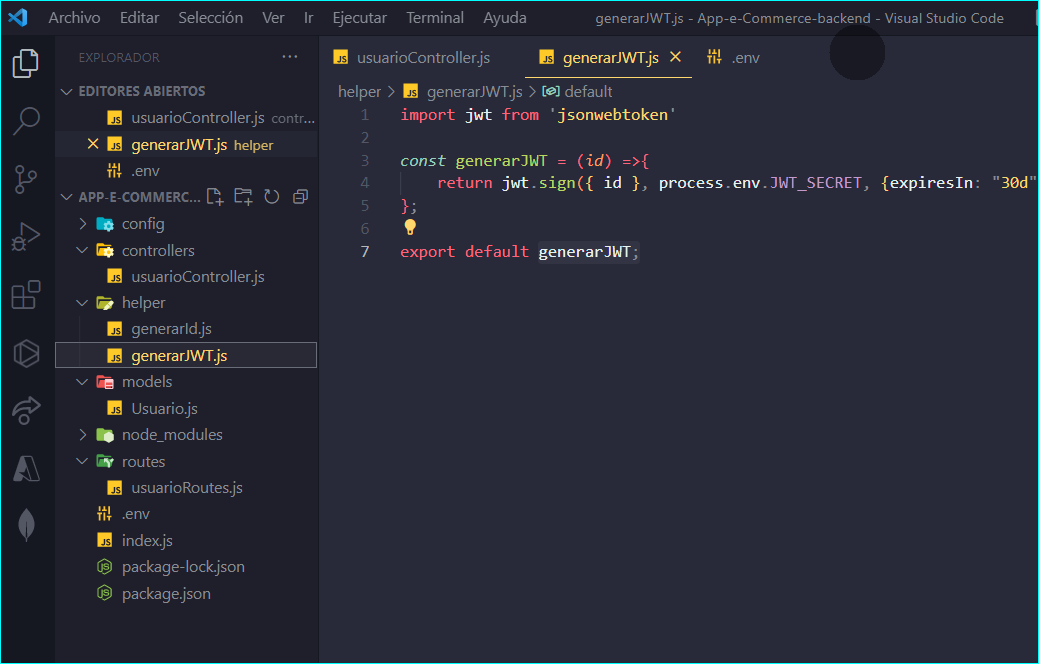


Validar también los metodos en el controlador (confirmar)



<https://jwt.io/>

**Comando**: npm i jsonwebtoken



Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Creación de un **middleware** para proteger las rutas

**usuarioRoutes.js**

import express from 'express';

import {

    registrar,

    perfil,

    confirmar,

    auntenticar,

    olvidePassword,

    comprobarToken,

    nuevoPassword

} from '../controllers/usuarioController.js';

*// middleware para validar el token*

import checkAuth from '../middleware/authMiddleware.js';

*const* router = express.Router();

*// Rutas Publicas*

router.post('/', registrar );

router.get('/confirmar/:token', confirmar);

router.post('/login', auntenticar);

router.post('/olvide-password', olvidePassword);

router.get('/olvide-password:token', comprobarToken);

router.route('/olvide-password/:token').get(comprobarToken).post(nuevoPassword);

*// Rutas Protegidas atraves del middleware checkAuth*

*// Identificamos el usuario y se identifica para mostrale los datos o funcionalidades que le corresponden.*

router.get('/perfil', checkAuth ,perfil);

export default router;

**usuarioController.js**

import Usuario from "../models/Usuario.js";

import generarJWT from "../helper/generarJWT.js";

import generarId from "../helper/generarId.js";

*//import emailOlvidePassword from "../helper/emailOlvidePassword.js";*

*const* registrar = *async* (*req*, *res*)=>{

*const* { nombre, email, password, telefono, direccion, web } = req.body;

*// Validar usuario duplicado*

*// findOne busca por los diferentes atributos de la coleccion*

*const* existeUsuario = await Usuario.findOne({email});

    if(existeUsuario){

*const* error = new Error("Usuario ya registrado");

        return res.status(400).json({msg: error.message});

    };

    try {

*const* usuario = new Usuario(req.body);

*const* usuarioGuardado = await  usuario.save();

        res.json(usuarioGuardado);

    } catch (error) {

        console.error(error.message);

    };

};

*const* perfil = (*req*, *res*)=>{

*//Extraemos los datos del usuario almacenado en el servidor de nodejs*

*//console.log(req.usuario);*

*const* { usuario } = req;

    try{

        res.status(200).json({

            usuario

        });

    } catch (error) {

        return res.status(404).json({

            status: 'error',

            error: error.message

        });

    }

};

*const* confirmar = *async* (*req*, *res*)=>{

*// req.params para leer datos de la URL, en este caso token por que asi lo definimos en la ruta*

*const* { token } = req.params;

*const* usuarioConfirmar = await Usuario.findOne({token});

    if(!usuarioConfirmar){

*const* error = new Error("Token no valido");

        return res.status(404).json({msg: error.message});

    };

    try {

        usuarioConfirmar.token = null;

        usuarioConfirmar.confirmado = true;

        await usuarioConfirmar.save();

        res.json({

            msg: "Usuario confirmado correctamente"

        });

    } catch (error) {

        console.error(error.message);

    }

};

*const* auntenticar = *async* (*req*, *res*)=>{

*const* { email, password } = req.body;

*const* usuario = await Usuario.findOne({email});

    if(!usuario){

*const* error = new Error("Usuario no existe");

        return res.status(403).json({msg: error.message});

    };

*// Comprobar si el usuario esta confirmado o no*

    if(!usuario.confirmado){

*const* error = new Error("Tu cuenta no ha sido confirmada");

        return res.status(403).json({msg: error.message});

    }

*// Aumtemticar el usuario*

*// Revisar el password si es correcto*

    if( await usuario.comprobarPassword(password)){

*// Auntenticar JWT*

*// https://jwt.io/*

        res.json({

            token: generarJWT(usuario.\_id),

            msg: "Usuario auntenticado"

        });

    }else{

*const* error = new Error("el password es incorrecto");

        return res.status(403).json({msg: error.message});

    }

};

*const* olvidePassword = *async* (*req*, *res*) =>{

*const* { email } = req.body;

*const* existeUsuario = await Usuario.findOne({email});

    if(!existeUsuario){

*const* error = new Error('El usuario no existe');

        return res.status(400).json({

            status: 'error',

            msg: error.message

        });

    };

    try {

        existeUsuario.token = generarId()

        await existeUsuario.save();

*// Enviar email con Instrucciones*

*// emailOlvidePassword({*

*//     email,*

*//     nombre: existeUsuario.nombre,*

*//     token: existeUsuario.token*

*// });*

        res.json({msg: "Hemos enviado un email con las instrucciones"});

    } catch (error) {

        return res.status(404).json({

            status: 'error',

            error: error.message

        });

    }

};

*const* comprobarToken = *async* (*req*, *res*) =>{

*const* { token } = req.params;

*const* tokenValido = await Usuario.findOne({token});

    if(tokenValido){

        res.json({msg: "Usuario valido"});

    }else{

*const* error = new Error('Token no valido');

        return res.status(400).json({

            status: 'error',

            msg: error.message

        });

    }

};

*const* nuevoPassword = *async* (*req*, *res*) =>{

*const* { token } = req.params;

*const* { password } = req.body;

*const* usuario = await Usuario.findOne({token});

    if(!usuario){

*const* error = new Error('Hubo un error');

        return res.status(400).json({

            status: 'error',

            msg: error.message

        });

    };

    try {

        usuario.token = null;

        usuario.password = password;

        await usuario.save();

        res.json({msg: "Password modificado correctamente"});

    } catch (error) {

        console.log("error: ", error.message);

        return res.status(404).json({

            status: 'error',

            error: error.message

        });

    };

};

export {

    registrar,

    perfil,

    confirmar,

    auntenticar,

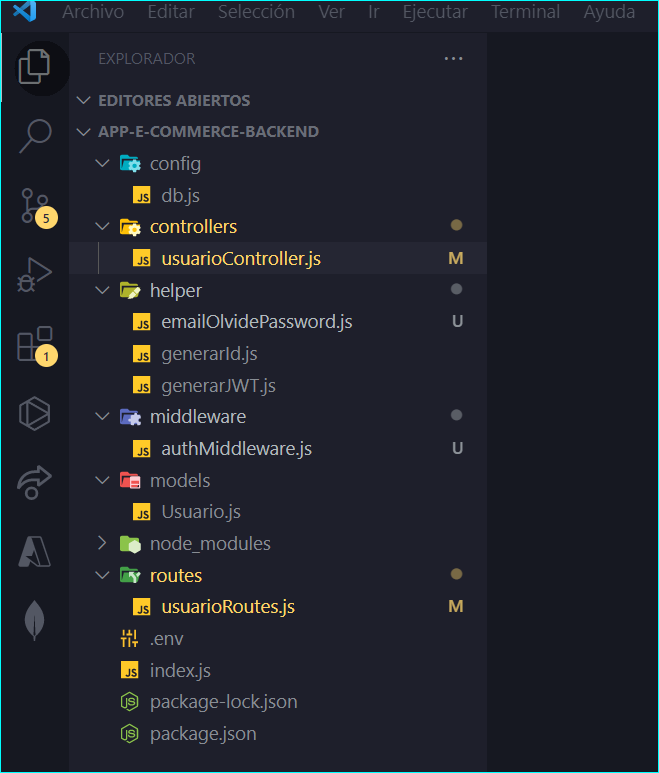
    olvidePassword,

    comprobarToken,

    nuevoPassword

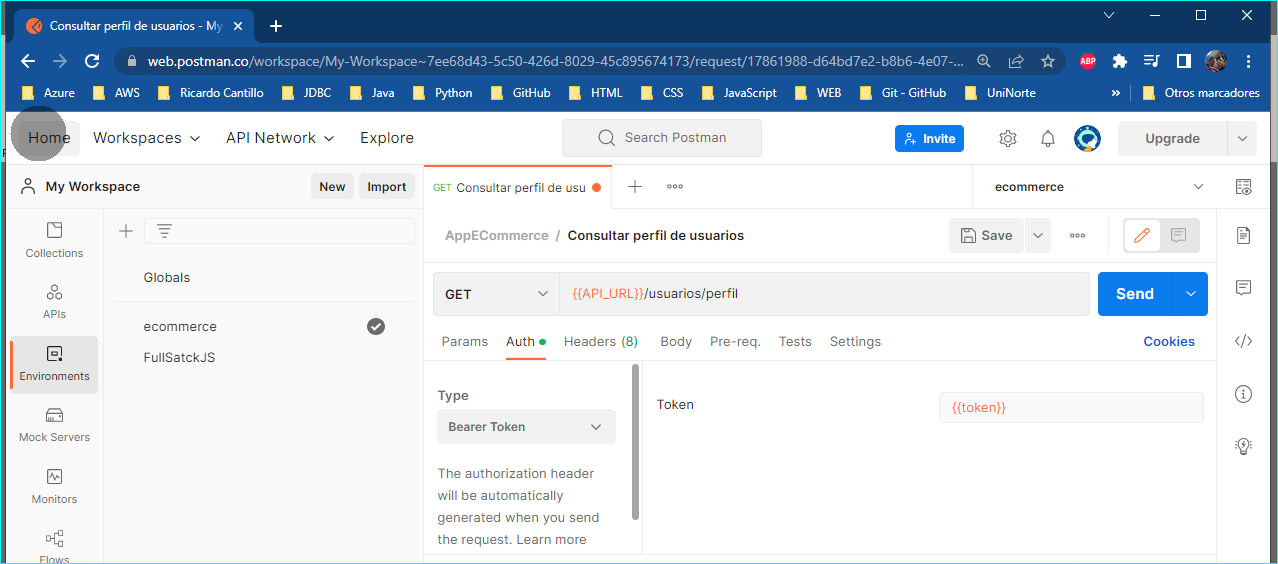
};

Corroborar estos archivos hasta este momento

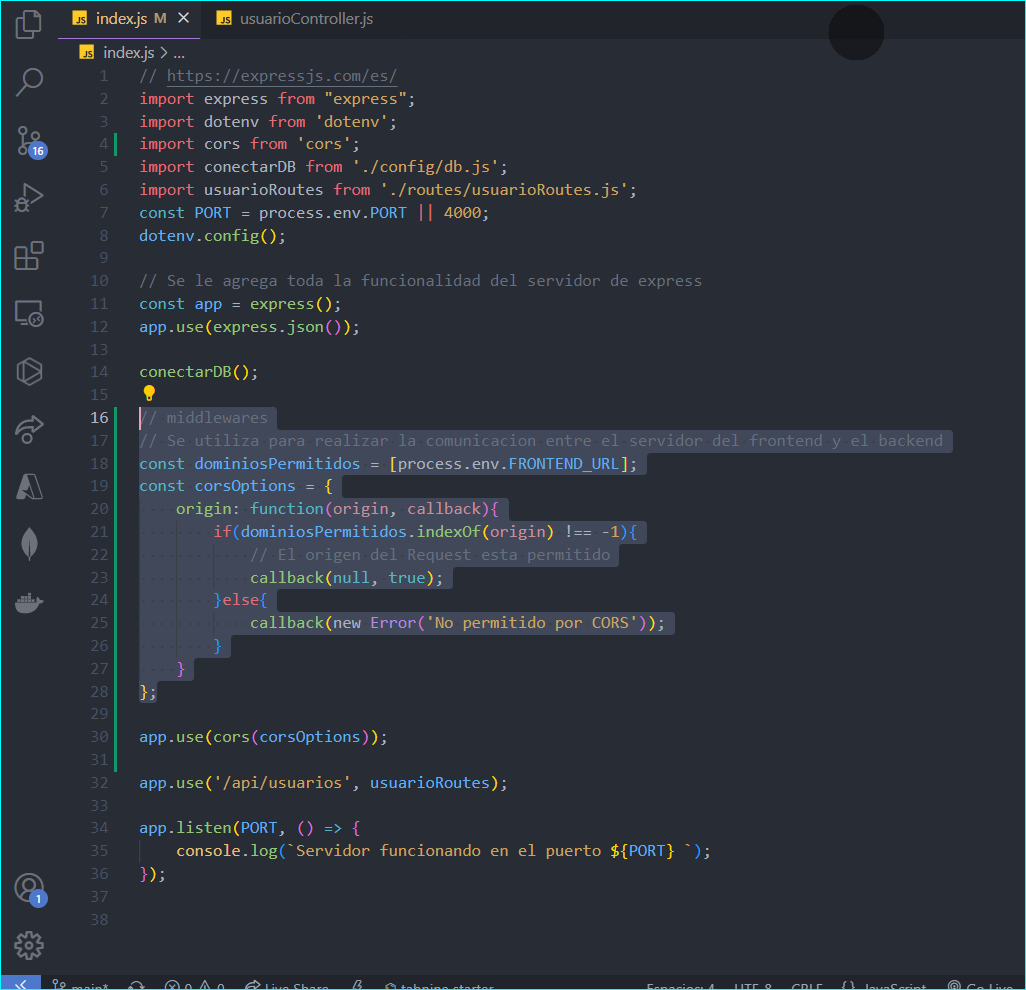


Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

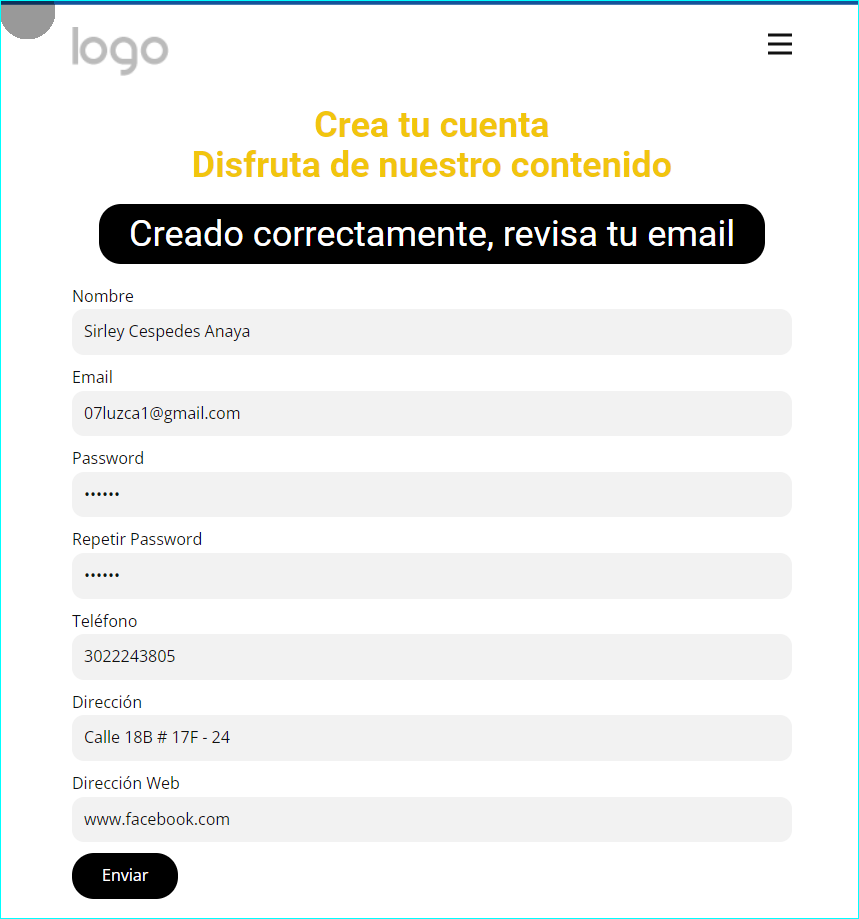


**Comando**: npm i cors



Texto

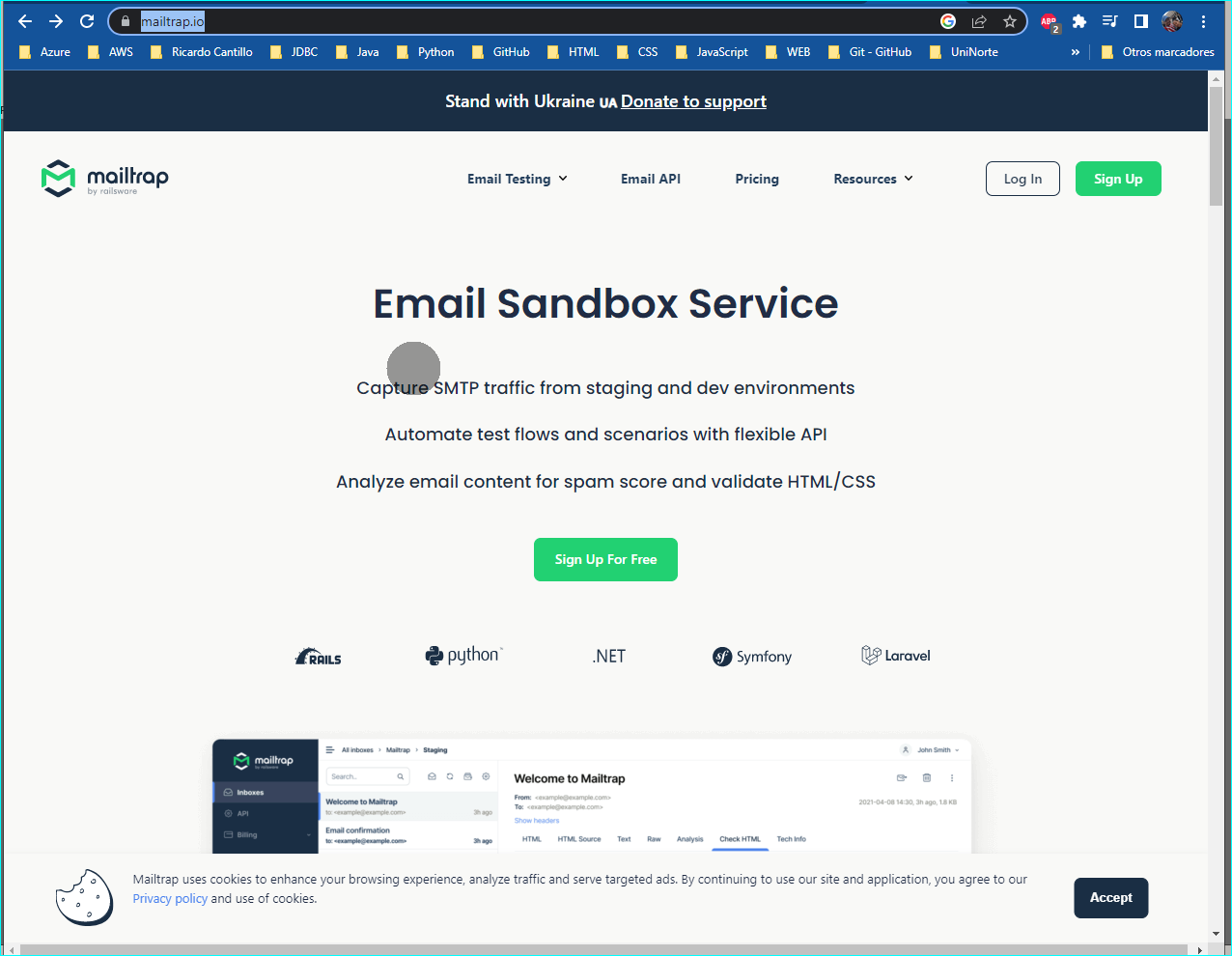
Descripción generada automáticamente

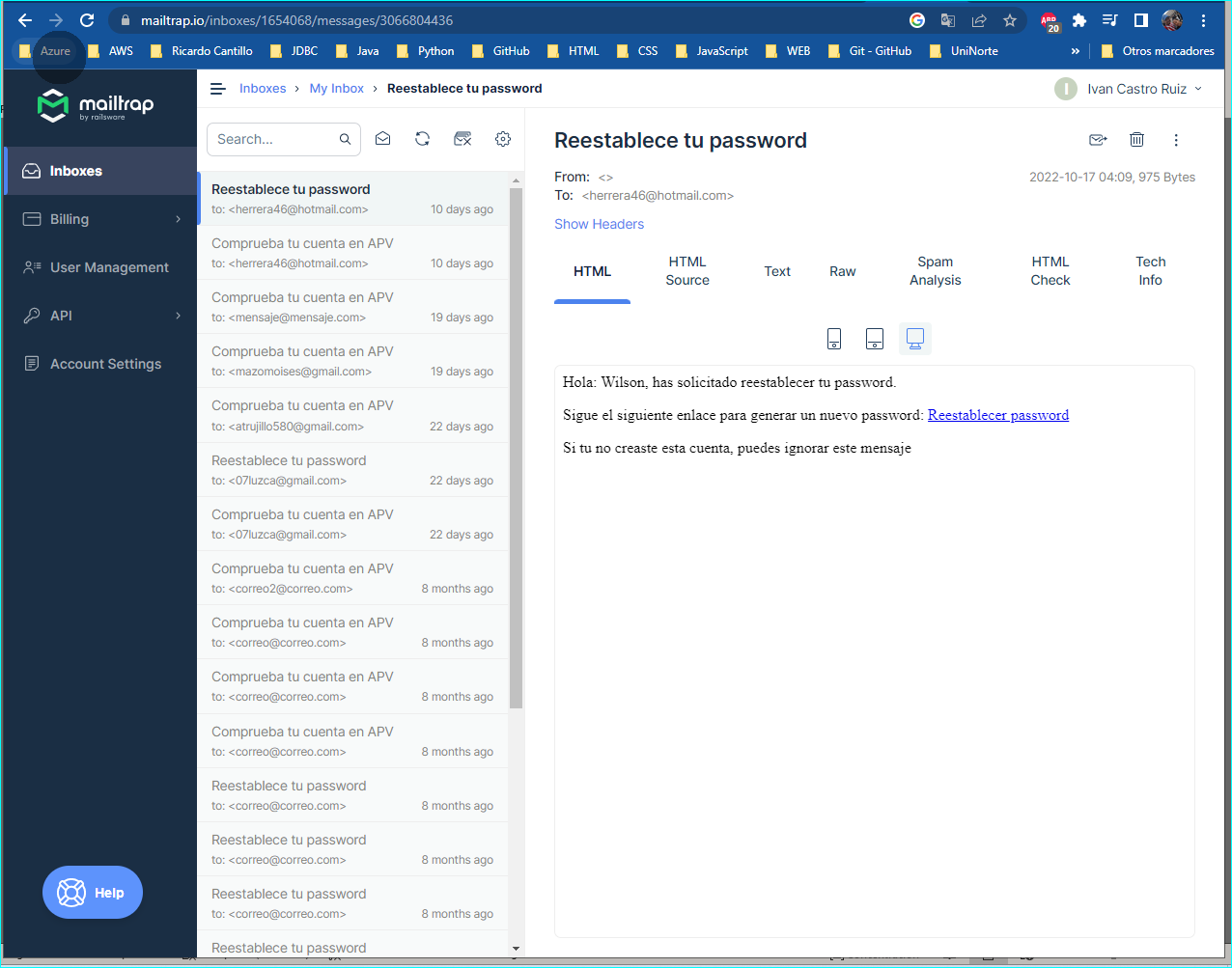


Texto

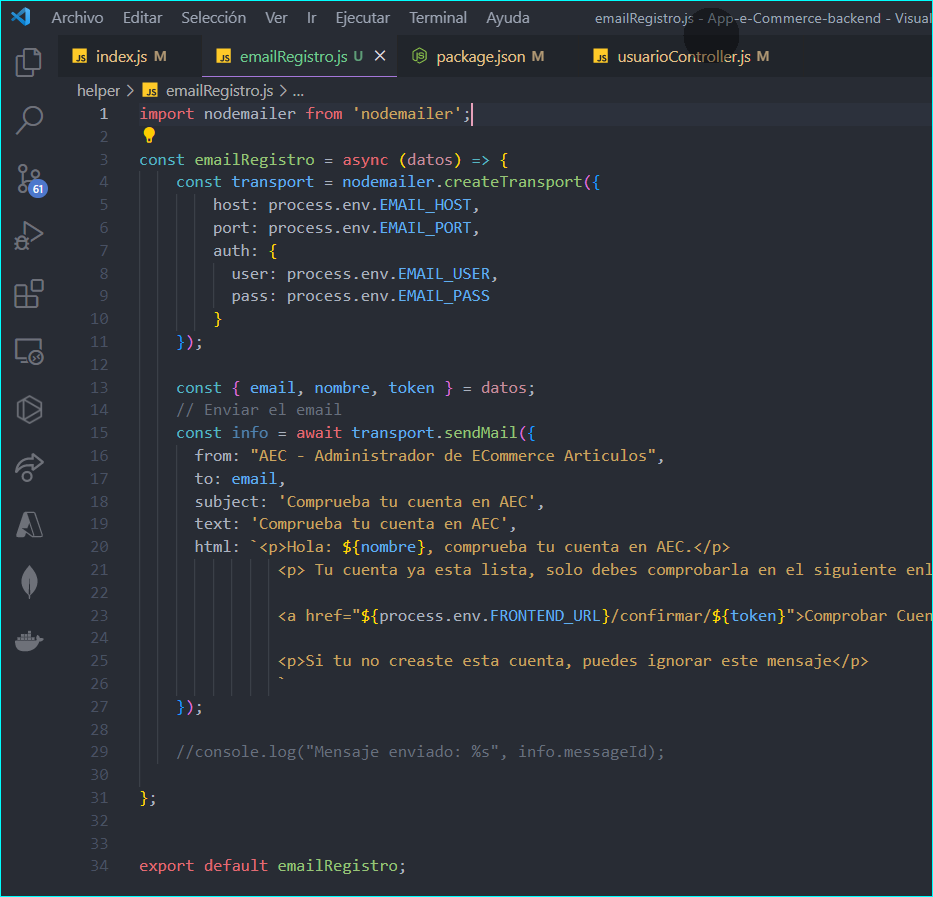
Descripción generada automáticamente

<https://mailtrap.io/>





**Comando**: npm i nodemailer



Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Una captura de pantalla de una computadora

Descripción generada automáticamente

Creando un Context de Autenticación

Frontend

**Paso 1 App.jsx**

import {

  BrowserRouter,

  Routes,

  Route

} from 'react-router-dom';

import { useState } from 'react';

// Context

import { AuthProvider } from './context/AuthProvider';

import { UsuariosProvider } from './context/UsuariosProvider';

// components Layout

import LayoutPublic from './Layout/LayoutPublic';

import RutaProtegida from './Layout/RutaProtegida';

// pages

import Home from './pages/Home';

import Vite from './pages/Vite';

import Main from './pages/Main';

// Login

import Registrar from './pages/Registrar';

import OlvidePassword from './pages/OlvidePassword';

import ConfirmarCuenta from './pages/ConfirmarCuenta';

import NuevoPassword from './pages/NuevoPassword';

// Fin Login

import ListaProductos from './pages/ListaProductos';

import DetalleArticulo from './pages/DetalleArticulo';

import AdministrarUsuarios from './pages/AdministrarUsuarios';

import EditarPerfil from './pages/EditarPerfil';

import CambiarPassword from './pages/CambiarPassword';

// Styles

//import './assets/css/App.css';

//import './assets/css/index.css';

import './assets/css/normalize.css';

import './assets/css/styles.css';

import reactLogo from './assets/react.svg';

function App() {

  // Hooks - vite

  //,estado,setEstado

  const [count, setCount] = useState(0);

  const [estado,setEstado] = useState(false);

  return (

    <BrowserRouter>

      <AuthProvider>

        <UsuariosProvider>

          {/\* Rutas Publicas \*/}

          <Routes>

            <Route exact path="/" element={<LayoutPublic

                                            reactLogo={reactLogo}

                                          />}>

              <Route path="/" element={<Home/>}/>

              <Route path="registrar" element={<Registrar/>} />

              <Route path="olvide-password" element={<OlvidePassword/>} />

              <Route path="olvide-password/:token" element={<NuevoPassword/>} />

              <Route path="confirmar/:token" element={<ConfirmarCuenta/>} />

              <Route path="vite" element={<Vite

                                            count={count}

                                            setCount={setCount}

                                            estado={estado}

                                            setEstado={setEstado}

                                        />}/>

              <Route path="contacto" element={<Main/>}/>

            </Route>

            {/\* Rutas Protegidas \*/}

            <Route path="/admin" element={<RutaProtegida/>}>

              <Route index element={<AdministrarUsuarios/>} />

              <Route path="perfil" element={<EditarPerfil/>} />

              <Route path="lista-articulo" element={<ListaProductos/>}/>

              <Route path="detalle-articulo/:id" element={<DetalleArticulo/>}/>

              <Route path="cambiar-password" element={<CambiarPassword/>} />

            </Route>

          </Routes>

        </UsuariosProvider>

      </AuthProvider>

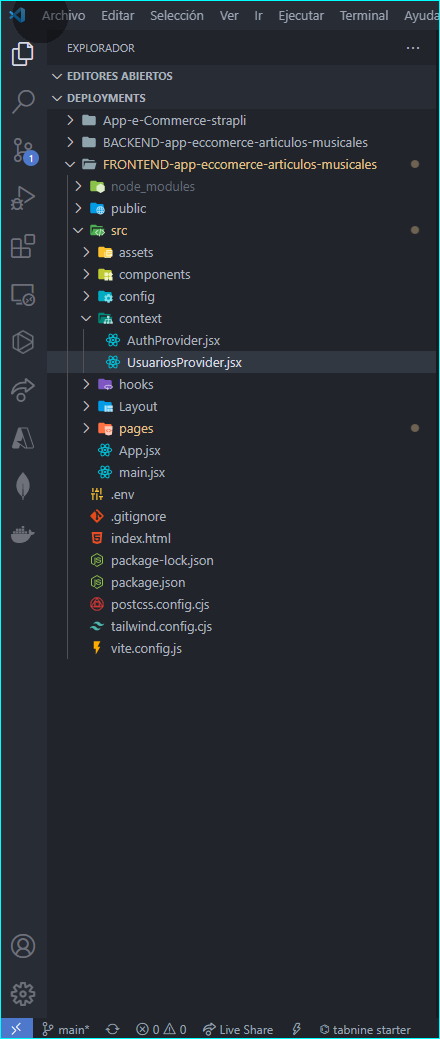
    </BrowserRouter>

  )

}

export default App

**Paso 2 – Creación de los context**



**AuthProvider.jsx**

import { useState, useEffect, createContext } from 'react';

import clienteAxios from '../config/axios'

const AuthContext = createContext();

const AuthProvider = ({children}) =>{

    const [ cargando, setCargando ] = useState(true);

    const [ auth, setAuth ] = useState({});

    useEffect(()=>{

        const autenticarUsuario = async () =>{

            const token = localStorage.getItem('token');

            if(!token){

                setCargando(false);

                return;

            }

            const config = {

                headers: {

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

                    Authorization: `Bearer ${token}`

                }

            };

            try {

                const { data } = await clienteAxios('/usuarios/perfil', config);

                setAuth(data);

                //console.log(data);

                //setCargando(false);

            } catch (error) {

                console.log(error.response.data.msg);

                setAuth({});

            }

            setCargando(false);

        };

        autenticarUsuario();

    }, []);

    const cerrarSesion = () => {

        localStorage.removeItem('token');

        setAuth({});

    };

    const actualizarPerfil = async datos => {

        const token = localStorage.getItem('token')

        if(!token) {

            setCargando(false)

            return

        }

        const config = {

            headers: {

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

                Authorization: `Bearer ${token}`

            }

        }

        try {

            const url = `/usuarios/perfil/${datos.\_id}`

            await clienteAxios.put(url, datos, config)

            return {

                msg: 'Almacenado Correctamente'

            }

        } catch (error) {

            return {

                msg: error.response.data.msg,

                error: true

            }

        }

    }

    const guardarPassword = async (datos) => {

        const token = localStorage.getItem('token')

        if(!token) {

            setCargando(false)

            return

        }

        const config = {

            headers: {

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

                Authorization: `Bearer ${token}`

            }

        }

        try {

            const url = '/usuarios/actualizar-password'

            const { data } = await clienteAxios.put(url, datos, config)

            console.log(data)

            return {

                msg: data.msg

            }

        } catch (error) {

            return {

                msg: error.response.data.msg,

                error: true

            }

        }

    }

    return(

        <AuthContext.Provider

            value={{

                auth,

                setAuth,

                cargando,

                cerrarSesion,

                actualizarPerfil,

                guardarPassword

            }}

        >

            {children}

        </AuthContext.Provider>

    )

};

export {

    AuthProvider

};

export default AuthContext;

**UsuariosProvider.jsx**

import { createContext, useState, useEffect } from 'react';

import clienteAxios from '../config/axios';

const UsuariosContext = createContext();

const UsuariosProvider = ({children}) =>{

    const [ usuarios, setUsuarios ] =  useState([]);

    const [ usuario, setUsuario ] =  useState({});

    useEffect(()=>{

        const obtenerUsuarios = async () =>{

            try{

                const token = localStorage.getItem('token');

                if(!token) return;

                const config = {

                    headers: {

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

                        Authorization: `Bearer ${token}`

                    }

                };

                const { data } = await clienteAxios('/usuarios', config);

                setUsuarios(data);

                console.log(data);

            } catch (error){

                console.log(error);

            }

        }

        obtenerUsuarios();

    },[]);

    const guardarUsuario = async (usuario) =>{

        console.log(usuarios)

        const token = localStorage.getItem('token');

        const config = {

            headers: {

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

                Authorization: `Bearer ${token}`

            }

        }

       if(usuario.id) {

            try{

                const { data } = await clienteAxios.put(`/usuarios/${usuario.id}`, usuario, config);

                console.log(data);

                const usuariosActualizados = usuarios.map( usuarioState => usuarioState.\_id === data.\_id ? data : usuarioState);

                setUsuarios(usuariosActualizados);

            } catch (error) {

                console.log(error);

            }

       }else{

            try{

                const { data } = await clienteAxios.post('/usuarios', usuario, config)

                console.log(data)

                //const { ...usuarioAlmacenado } = data

                setUsuarios([data, ...usuarios])

                //console.log(usuarioAlmacenado)

                console.log(usuarios)

            } catch (error) {

                console.log(error.response.data.msg);

            }

       }

    };

    const setEdicion = (usuario) =>{

        setUsuario(usuario);

    };

    const eliminarUsuario = async (id) =>{

        const confirmar = confirm('¿Confirmas que deseas eliminar?');

        if(confirmar){

            try{

                const token = localStorage.getItem('token');

                const config = {

                    headers: {

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

                        Authorization: `Bearer ${token}`

                    }

                }

                const { data } = await clienteAxios.delete(`/usuarios/${id}`, config);

                const usuariosActualizados = usuarios.filter( usuarioState => usuarioState.\_id !== id);

                setUsuarios(usuariosActualizados);

            } catch (error){

                console.log(error);

            }

        };

    };

    return (

        <UsuariosContext.Provider

            value={{

                usuarios,

                guardarUsuario,

                setEdicion,

                usuario,

                eliminarUsuario

            }}

        >

            {children}

        </UsuariosContext.Provider>

    );

}

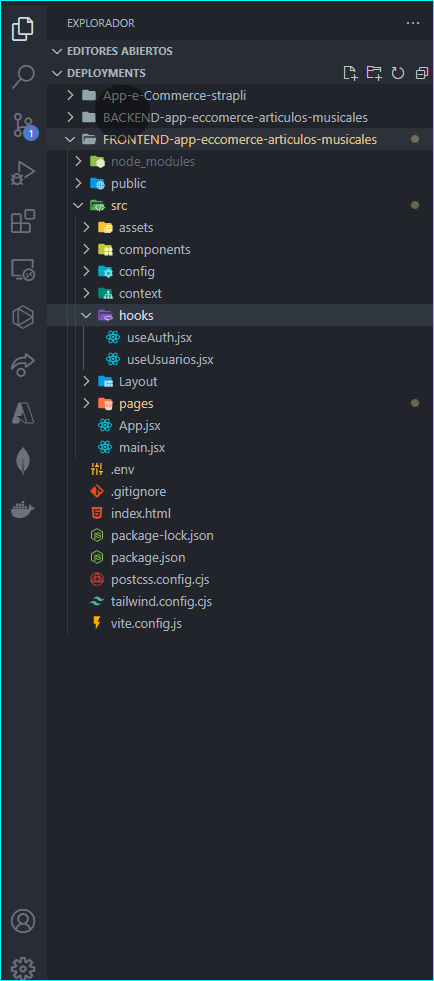
export {

    UsuariosProvider

}

export default UsuariosContext;

**Creación de los hooks – propios para gestionar los context y que la información se maneje de manera global la del usuario registrado.**



**useAuth.jsx**

import { useContext } from 'react'; // Para extraer los datos

import AuthContext from '../context/AuthProvider';

const useAuth = () =>{

    return useContext(AuthContext);

};

export default useAuth

**useUsuarios.jsx**

import { useContext } from 'react';

import UsuariosContext from '../context/UsuariosProvider';

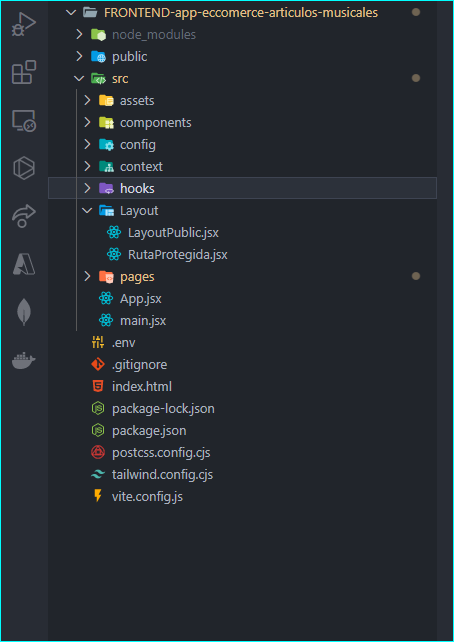
const useUsuarios = () =>{

    return useContext(UsuariosContext);

};

export default useUsuarios

**Creación de Layout Públicos y Protegidos**



**LayoutPublic.jsx**

import { Outlet  } from 'react-router-dom';

// Componentes

import Header from '../components/Header';

import Navegacion from '../components/Navegacion';

import Footer from '../components/Footer';

import Sidebar from '../components/Sidebar';

const LayoutPublic = ({reactLogo}) => {

  return (

    <>

      <Header/>

      <Sidebar/>

      <Navegacion/>

      <main className="container mx-auto md:grid md:grid-cols-2 mt-12 gap-10 p-5 items-center">

        <Outlet/>

      </main>

      <Footer

        reactLogo={reactLogo}

      />

    </>

  )

}

export default LayoutPublic

**RutaProtegida.jsx**

import  { Outlet, Navigate } from 'react-router-dom';

import useAuth from '../hooks/useAuth';

import Sidebar from '../components/rutasProtegidas/Sidebar';

import Header from '../components/rutasProtegidas/Header';

import Footer from '../components/rutasProtegidas/Footer';

const RutaProtegida = () => {

  const { auth, cargando } = useAuth();

  const { usuario } = auth;

  if(cargando) return 'cargando...'

  return (

    <>

      <Header/>

      <Sidebar />

      {usuario?.\_id ? (

          <main className='container mx-auto mt-10'>

              <Outlet />

          </main>

      ): <Navigate to="/" /> }

      <Footer/>

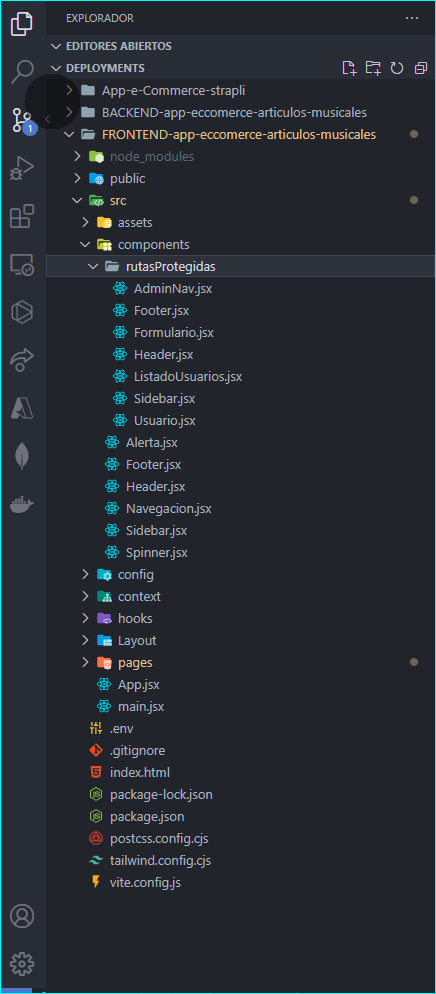
    </>

  )

}

export default RutaProtegida

**Componentes Protegidos Componentes Públicos**

 Texto

Descripción generada automáticamente

**Componentes Paginas**

