**INICIANDO EL PROYECTO BACKEND**

* **Yarn init y enter**

LENOVO@Raychu MINGW64 /d/Blog\_CSC/BACKEND

$ yarn init

yarn init v1.22.19

question name (BACKEND): Blog-CSC

question version (1.0.0):

question description: Blog diseñado para la CSC

question entry point (index.js):

question repository url:

question author: Raychu

question license (MIT):

question private:

success Saved package.json

Done in 107.81s.

Me creo el **package.json**

{

  "name": "Blog-CSC",

  "version": "1.0.0",

  "description": "Blog diseñado para la CSC",

  "main": "index.js",

  "author": "Raychu",

  "license": "MIT"

}

Vamos a crear la estructura de nuestro ´proyecto creamos las siguientes carpetas

**controlador:** guardamos las funciones que se ejecutan a una petición htpps

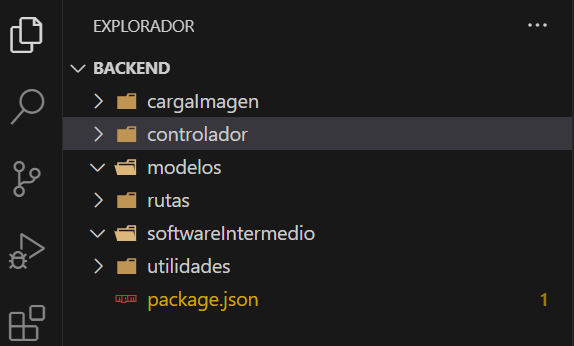
**sofwareIntermedio**: peticiones intermedias que se ejecutan en el proyecto

**modelos:** donde van todas las tablas

**rutas:** donde van a estar todas las rutas

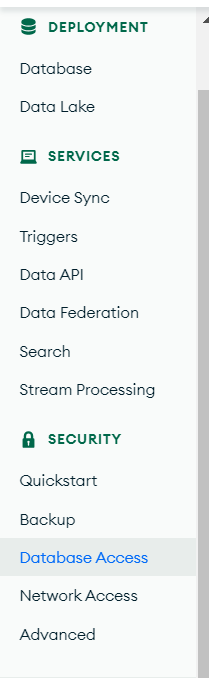
c**argaImagen**: donde se van a cargar todas las imágenes

**utilidades:** se crean utilidades que se vayan necesitando



1. Creando la base de datos en MongooseDB

<https://www.mongodb.com/es>

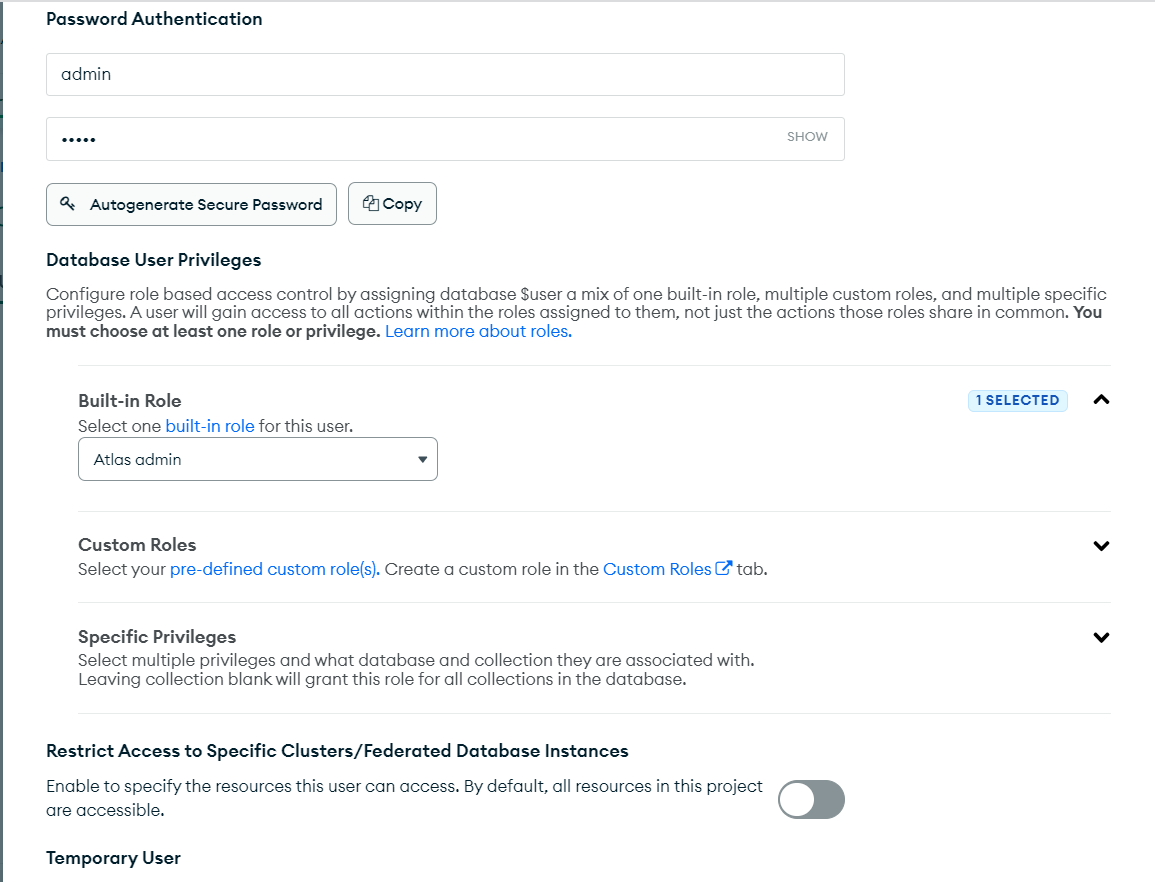
* Creamos el cluster o base de datos
* Creamos el usuario de acceso a la BD dando clic en Database acces
* 

Creamos el usuario admin

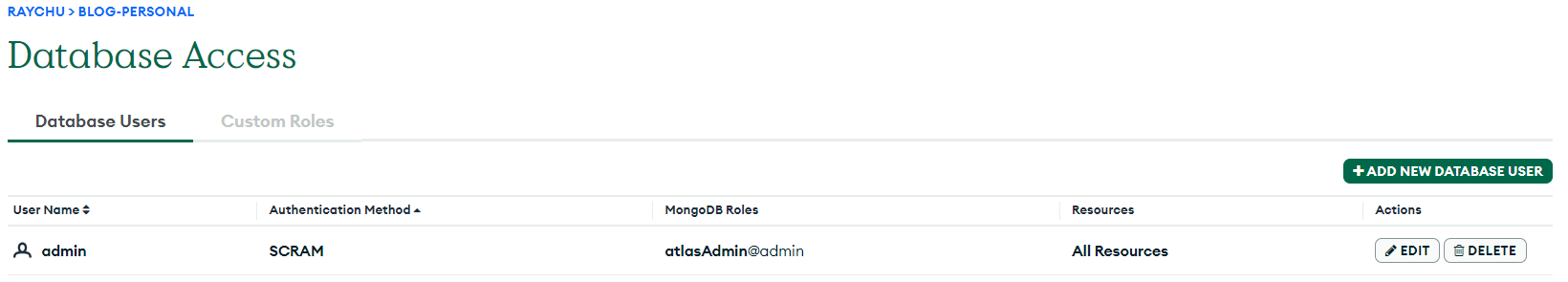
Contraseña 12345

Seleccionamos el role: admin

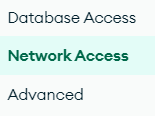
Y crear usuario



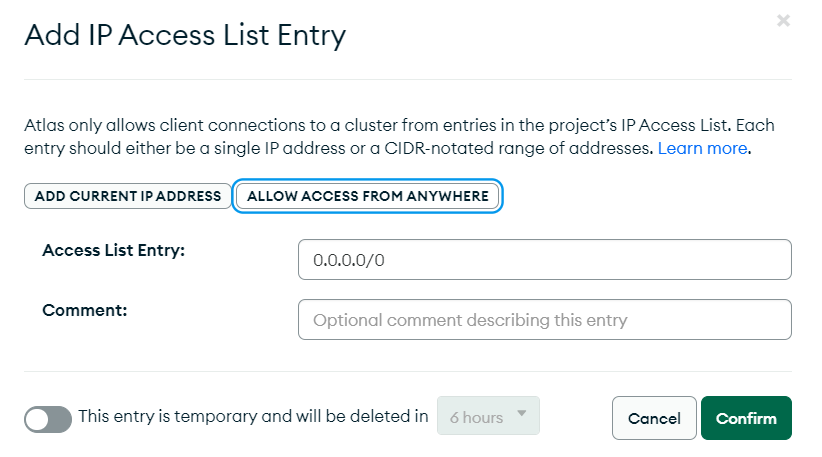
Y nos aparece lo siguiente



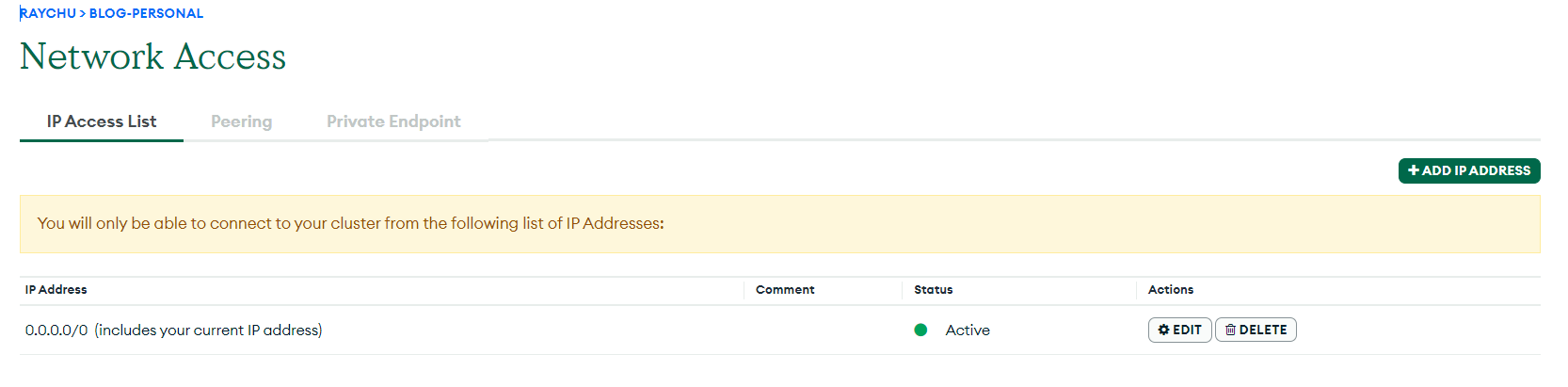
Ahora vamos a darle acceso a una ip a la base de datos le damos clic a Network Acces



Le damos clic que todo el mundo pueda acceder



Y confirmar



1. Mongoose conectando a la APP con la base de datos

Instalamos mongoose:

Yarn add mongoose la ultima versión

Yarn add [mongoose@6.6.1](mailto:mongoose@6.6.1) una versión en especial

Vamos acrear un fichero que se **llama constantes.js**

const USUARIO\_BD = 'admin';

const CONTRASENA\_BD = '12345';

const IP\_BD = 'blogpersonal.p5uujtw.mongodb.net';

const API\_VERSION = 'v1';

const IP\_SERVER = 'localhost';

module.exports = {

  USUARIO\_BD,

  CONTRASENA\_BD,

  IP\_BD,

  API\_VERSION,

  IP\_SERVER,

};

Ahora vamos a crear el fichero **index.js donde vamos hacer la conexión a la base de datos**

//Conectando a la base de datos de mongoose

const mongoose = require('mongoose');

//const app = require('./app');

const {

  USUARIO\_BD,

  CONTRASENA\_BD,

  API\_VERSION,

  IP\_BD,

  IP\_SERVER,

} = require('./constantes');

//mongodb+srv://admin:<password>@blogpersonal.p5uujtw.mongodb.net/?retryWrites=true&w=majority

mongoose.connect(

  `mongodb+srv://${USUARIO\_BD}:${CONTRASENA\_BD}@${IP\_BD}/`,

  (error) => {

    if (error) throw error;

      console.log('######################');

      console.log('###### API REST - CONEXIÓN ARRIBA - CSC  ######');

  }

);

**Vamos a probar si funciona entonces nos vamos al package.json y creamos un scripts**

****

Creando el servidor HTTP con express instalamos

Yarn add express

Ahora creamos el fichero **app.js**

const express = require('express');

const { API\_VERSION } = require('./constantes');

const app = express();

// Import routings

// Configure Body Parser //para mandar cobtenido json del body, es lo que nos llega del cliente al servidor

// Configure static folder, para poder ver las imagenes, en caso que se necesiten mas carpetass se adicionan

//Configure Header HTTP - CORS, cuando hagamos peticiones http no las bloquee sino las deje pasar

// Configure rutas

module.exports = app;

ahora que temenos el app.js vamos a exportarlo en el index.js

//Conectando a la base de datos de mongoose

const mongoose = require('mongoose');

const app = require('./app');

const {

  USUARIO\_BD,

  CONTRASENA\_BD,

  API\_VERSION,

  IP\_BD,

  IP\_SERVER,

} = require('./constantes');

const PORT = process.env.POST || 4001;

//mongodb+srv://admin:<password>@blogpersonal.p5uujtw.mongodb.net/?retryWrites=true&w=majority

mongoose.connect(

  `mongodb+srv://${USUARIO\_BD}:${CONTRASENA\_BD}@${IP\_BD}/`,

  (error) => {

    if (error) throw error;

    app.listen(PORT, () => {

      console.log('######################');

      console.log('###### API REST - CONEXIÓN ARRIBA - CSC  ######');

      console.log('######################');

      console.log(`http://${IP\_SERVER}:${PORT}/api/${API\_VERSION}`);

    });

  }

);

1. **Recargando el servidor automáticamente**

Instalamos el paquete nodemon

Yarn add nodemon

Ahora vamos al package.json

  "scripts": {

    "start": "node index.js",

    "dev": "nodemon index.js" este es el código adicional

  },

1. **Configurando los body parser**

Instalamos el paquete body-parser

Yarn add body-parser

Nos **vamos al app.js y** vamos a configurar

const express = require('express');

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

const { API\_VERSION } = require('./constantes');

const app = express();

// Import routings

// Configure Body Parser //para mandar cobtenido json del body, es lo que nos llega del cliente al servidor

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

// Configure static folder, para poder ver las imagenes, en caso que se necesiten mas carpetass se adicionan

//Configure Header HTTP - CORS, cuando hagamos peticiones http no las bloquee sino las deje pasar

// Configure rutas

module.exports = app;

1. **Configurando carpeta de estaticos**

Vamos al archive app.js

// Configure static folder (cargaImagen), para poder ver las imagenes, en caso que se necesiten mas carpetass se adicionan

app.use(express.static('cargaImagen'));

1. **Configurando los CORS**

Instalamos CORS

Yarn add cors

Vamos app.js

const cors = require('cors');

//Configure Header HTTP - CORS, cuando hagamos peticiones http no las bloquee sino las deje pasar

app.use(cors());

**SISTEMA DE AUTENTICACIÓN BACKEND**

1. Crear nuestro 1er endpoint para observar cómo se hace y así solo mande un mensaje

Creamos un fichero en la carpeta **controlador llamado autenticación.js**

function registro(req, res) {

  console.log('Se ha ejecutado el registro');

  res.status(200).send({ msg: 'TODO OKI' });

}

module.exports = {

  registro,

};

Ahora nos vamos a la carpeta **rutas y creamos el fichero autenRuta.js**

const express = require('express');

const AutenControler = require('../controlador/autenticacion');

//vamos a crear las rutas

const api = express.Router();

//Vamos a definir nuestra primera ruta

api.post('/autenticacion/registro', AutenControler.registro);

// y lo exportamos

module.exports = api;

**ahora vamos api.js**

const express = require('express');

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

const { API\_VERSION } = require('./constantes');

const cors = require('cors');

const app = express();

// Import routings

const rutaAutentica = require('./rutas/autenRuta');

// Configure Body Parser //para mandar cobtenido json del body, es lo que nos llega del cliente al servidor

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

// Configure static folder (cargaImagen), para poder ver las imagenes, en caso que se necesiten mas carpetass se adicionan

app.use(express.static('cargaImagen'));

//Configure Header HTTP - CORS, cuando hagamos peticiones http no las bloquee sino las deje pasar

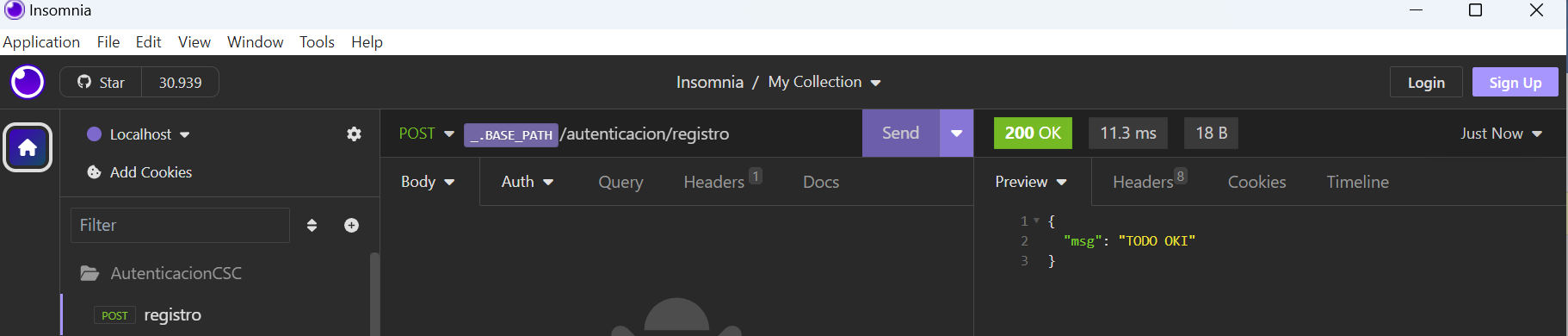
app.use(cors());

// Configure rutas

app.use(`/api/${API\_VERSION}`, rutaAutentica);

module.exports = app;

**vamos a probarlo con insomia**

****

**Nota esa es la forma de crear nuestros endpoint**

1. **Crear el modelo o tabla de usuario:**

Creamos un fichero dentro de la carpeta modelos llamada usuario.js

const mongoose = require('mongoose');

const UsuarioEsquema = mongoose.Schema({

  nombre: String,

  apellido: String,

  email: {

    type: String, //tipo string y unico no se repite el email

    unique: true,

  },

  password: String,

  rol: String,

  active: Boolean,

});

module.exports = mongoose.model('Usuario', UsuarioEsquema);

1. **Registrando Usuario**

**Vamos al fichero autenticación.js que se encuentra en la carpeta controlador**

Instalamos el fichero bcryptjs para encriptar la contraseña

**Yarn add** bcryptjs

const User = require('../modelos/Usuario');

const bcrypt = require('bcryptjs');

function registro(req, res) {

  const { nombre, apellido, email, password } = req.body;

  if (!email) res.status(400).send({ msg: 'El email es obligatorio' });

  if (!password) res.status(400).send({ msg: 'La contraseña es obligatoria' });

  const usua = new User({

    nombre,

    apellido,

    email: email.toLowerCase(), // tolower siempre se registra en miniscula

    rol: 'Usuario',

    active: false,

  });

  // Encriptación de Contraseña

  const salt = bcrypt.genSaltSync(10);

  const hashPassword = bcrypt.hashSync(password, salt);

  usua.password = hashPassword;

  //Guardar el usuario en la base de datos

  usua.save((error, userStorage) => {

    if (error) {

      res.status(400).send({ msg: 'Error al crear el usuario' });

    } else {

      res.status(200).send(userStorage);

    }

  });

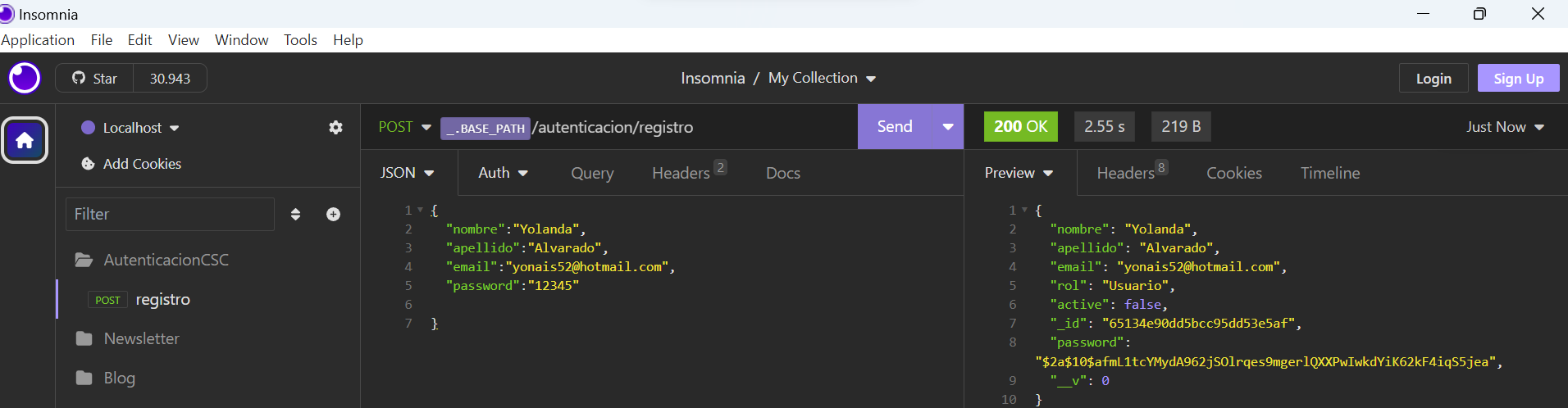
}

module.exports = {

  registro,

};

**Y efectivamente ya me esta creando el usuario en la base de datos**

****

1. **AccesToken y RefreshToken**

**Instalamos el paquete yarn add jsonwebtoken para el accenToken y refreshToken**

**En la carpeta utilidades creamos el fichero jwt.js**

const jwt = require('jsonwebtoken');

const { JWT\_SECRET\_KEY } = require('../constantes');

function createAccessToken(user) {

  const expToken = new Date();

  expToken.setHours(expToken.getHours() + 3);

  const payload = {

    token\_type: 'access',

    user\_id: user.\_id,

    iat: Date.now(),

    exp: expToken.getTime(),

  };

  return jwt.sign(payload, JWT\_SECRET\_KEY);

}

function createRefreshToken(user) {

  const expToken = new Date();

  expToken.getMonth(expToken.getMonth() + 1);

  const payload = {

    token\_type: 'refresh',

    user\_id: user.\_id,

    iat: Date.now(),

    exp: expToken.getTime(),

  };

  return jwt.sign(payload, JWT\_SECRET\_KEY);

}

function decoded(token) {

  return jwt.decode(token, JWT\_SECRET\_KEY, true);

}

module.exports = {

  createAccessToken,

  createRefreshToken,

  decoded,

};

**Pero en el fichero constantes.js**

const JWT\_SECRET\_KEY = 'gR7cH9Svfj8JLe4c186Ghs48hheb3902nh5DsA';

1. **Login del Usuario**

Nos vamos a la carpeta controlador y al **fichero autenticacion.js**

const jwt = require('../utilidades/jwt');

function login(req, res) {

  const { email, password } = req.body;

  if (!email) res.status(400).send({ msg: 'El email es obligatorio' });

  if (!password) res.status(400).send({ msg: 'La contraseña es obligatoria' });

  const emailLowerCase = email.toLowerCase();

  // buscar solo un usuario

  User.findOne({ email: emailLowerCase }, (error, userStore) => {

    if (error) {

      res.status(500).send({ msg: 'Error del servidor' });

    } else {

      bcrypt.compare(password, userStore.password, (bcryptError, check) => {

        if (bcryptError) {

          res.status(500).send({ msg: 'Error del servidor' });

        } else if (!check) {

          res.status(400).send({ msg: 'Contraseña incorrecta' });

        } else if (!userStore.active) {

          res.status(401).send({ msg: 'Usuario no autorizado o no activo' });

        } else {

          res.status(200).send({

            access: jwt.createAccessToken(userStore),

            refresh: jwt.createRefreshToken(userStore),

          });

        }

      });

    }

  });

}

function refreshAccessToken(req, res) {

  const { token } = req.body;

  if (!token) res.status(400).send({ msg: 'Token requerido' });

  const { user\_id } = jwt.decoded(token);

  User.findOne({ \_id: user\_id }, (error, userStorage) => {

    if (error) {

      res.status(500).send({ msg: 'Error del servidor' });

    } else {

      res.status(200).send({

        accessToken: jwt.createAccessToken(userStorage),

      });

    }

  });

}

module.exports = {

  registro,

  login,

  refreshAccessToken,

};

Vamos a rutas – autenRuta.js

const express = require('express');

const AutenControler = require('../controlador/autenticacion');

//vamos a crear las rutas

const api = express.Router();

//Vamos a definir nuestra primera ruta

api.post('/autenticacion/registro', AutenControler.registro);

api.post('/autenticacion/login', AutenControler.login);

api.post(

  '/autenticacion/refresh\_access\_token',

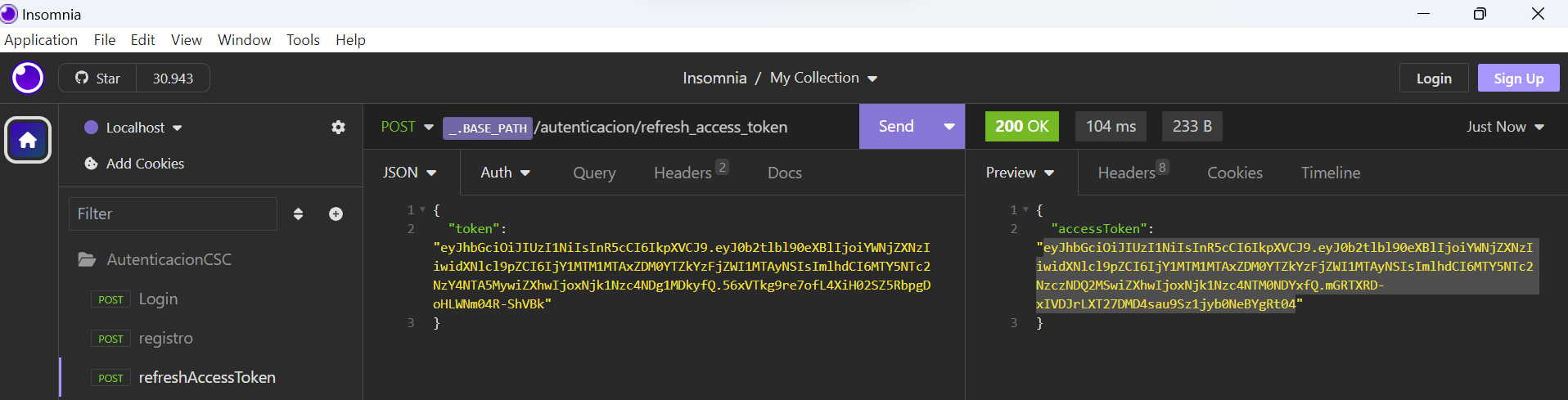
  AutenControler.refreshAccessToken

);

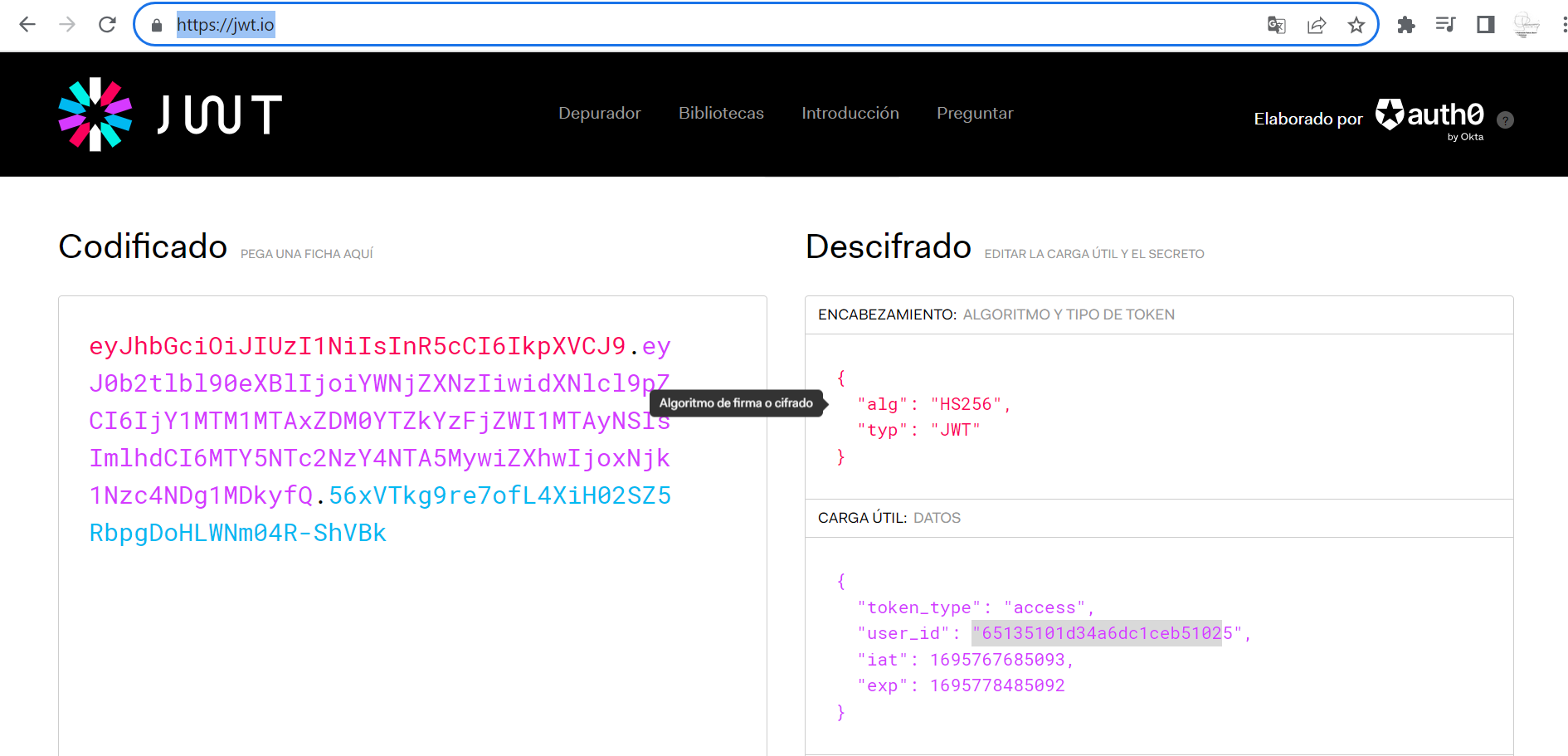
// y lo exportamos

module.exports = api;

y obtenemos en insomia lo siguiente:



En la página JWT podemos decodificar el Token <https://jwt.io/>



**SISTEMA DE USUARIO BACKEND**

1. **Estructura API USUARIO**

Vamos a controlador y creamos un fichero llamado usuario.js

Creamos las funciones de obtener usuarios, crear usuario, actualizar usuario y eliminar

const bcrypt = require('bcryptjs');

const User = require('../modelos/Usuario');

//función de los datos del usuario logueado

async function consigueme(req, res) {

  const { user\_id } = req.user;

  const response = await User.findById(user\_id);

  if (!response) {

    res.status(400).send({ msg: 'No se ha encontrado usuario' });

  } else {

    res.status(200).send(response);

  }

}

async function ObtenerUsuarios(req, res) {

  const { active } = req.query;

  let response = null;

  if (active === undefined) {

    response = await User.find();

  } else {

    response = await User.find({ active });

  }

  res.status(200).send(response);

}

async function crearUsuario(req, res) {

  //console.log(req.body); //puebas antes de empezar a programar y guadar el registro de usuario

  //res.status(200).send({ msg: 'creación de usuario desde la administración' }); // instrucción para verificar el endpoint

  const { password } = req.body; //Mostramos la contraseñala contraseña llega sin encriptar y hay que encriptarla

  const creauser = new User({ ...req.body, active: false }); // nuevo usuario

  // encriptamos la contraseña

  const salt = bcrypt.genSaltSync(10);

  const hasPassword = bcrypt.hashSync(password, salt);

  creauser.password = hasPassword;

  creauser.save((error, userStored) => {

    if (error) {

      res.status(400).send({ msg: 'Error al crear el usuario' });

    } else {

      res.status(201).send(userStored);

    }

  });

}

async function actualizar(req, res) {

  const { id } = req.params;

  const userData = req.body; // datos del usuario

  //password

  if (userData.password) {

    const salt = bcrypt.genSaltSync(10);

    const hashPassword = bcrypt.hashSync(userData.password, salt);

    userData.password = hashPassword;

  } else {

    delete userData.password;

  }

  User.findByIdAndUpdate({ \_id: id }, userData, (error) => {

    if (error) {

      res.status(400).send({ msg: 'Error al actualizar el usuario' });

    } else {

      res.status(200).send({ msg: 'Actualizacion correcta' });

    }

  });

}

async function eliminar(req, res) {

  const { id } = req.params;

  User.findByIdAndDelete(id, (error) => {

    if (error) {

      res.status(400).send({ msg: 'Error al eliminar el usuario' });

    } else {

      res.status(200).send({ msg: 'Usuario eliminado' });

    }

  });

}

module.exports = {

  consigueme,

  ObtenerUsuarios,

  crearUsuario,

  actualizar,

  eliminar,

};

Ahora vamos a la ruta y creamos un fichero usuaRuta.js

const express = require('express');

const UserControler = require('../controlador/usuario');

const md\_auth = require('../softwareIntermedio/autenticacion');

const api = express.Router();

api.get(

  '/usuarios/me',

  [md\_auth.asegurarAutenticacion],

  UserControler.consigueme

);

api.get(

  '/allusers',

  [md\_auth.asegurarAutenticacion],

  UserControler.ObtenerUsuarios

);

api.post(

  '/crearUsuario',

  [md\_auth.asegurarAutenticacion],

  UserControler.crearUsuario

);

api.patch(

  '/usuario/:id',

  [md\_auth.asegurarAutenticacion],

  UserControler.actualizar

);

api.delete(

  '/usuario/:id',

  [md\_auth.asegurarAutenticacion],

  UserControler.eliminar

);

module.exports = api;

Vamos a app.js

const express = require('express');

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

const { API\_VERSION } = require('./constantes');

const cors = require('cors');

const app = express();

// Import routings

const rutaAutentica = require('./rutas/autenRuta');

const rutaUsuario = require('./rutas/usuaRuta');

// Configure Body Parser //para mandar cobtenido json del body, es lo que nos llega del cliente al servidor

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

// Configure static folder (cargaImagen), para poder ver las imagenes, en caso que se necesiten mas carpetass se adicionan

app.use(express.static('cargaImagen'));

//Configure Header HTTP - CORS, cuando hagamos peticiones http no las bloquee sino las deje pasar

app.use(cors());

// Configure rutas

app.use(`/api/${API\_VERSION}`, rutaAutentica);

app.use(`/api/${API\_VERSION}`, rutaUsuario);

module.exports = app;

