**Conexión a la base de datos**

Dentro de api.js se realiza la conexión con la base de datos

Después de importar el framwork

*//Creacion de una constante mongoose*

*//importar la dependencia mongoose*

const mongoose=require('mongoose');

luego después de asignar el puerto

*//conexion a la base de datos*

*//se llama a la constante*

*//se llama al método connect*

*//se pega la url sacada de la web de mongodb*

mongoose.connect('mongodb+srv://miltonXD:Atlas1234@cluster0.venxvyj.mongodb.net/miApp?retryWrites=true&w=majority');

conexión lista

**Modelo de base de datos**

Para realizar el modelo de base de dato se crea un nuevo archivo



Dentro del archivo

*//importar mongoose*

const mongoose= require('mongoose')

*//modelo de base dedatos*

*//un modelo tiene que tener un nombre 'User'*

*//objeto js con la forma de los documentos que se guardaran dentro de la coleccion User*

const Users=mongoose.model('User',{

*//esquema*

*//agaregar las propiedades*

*//type:String = es el tipo de elemento que va a recivir*

*//require:true = si el atributo es obligatorio su ingreso o no*

*//minLength:3 = tramaño minimo del el string*

    nombre:{type:String, require:true, minLength:3},

    apellido:{type:String, require:true, minLength:3},

})

*//exportar el modulo*

module.exports=Users;

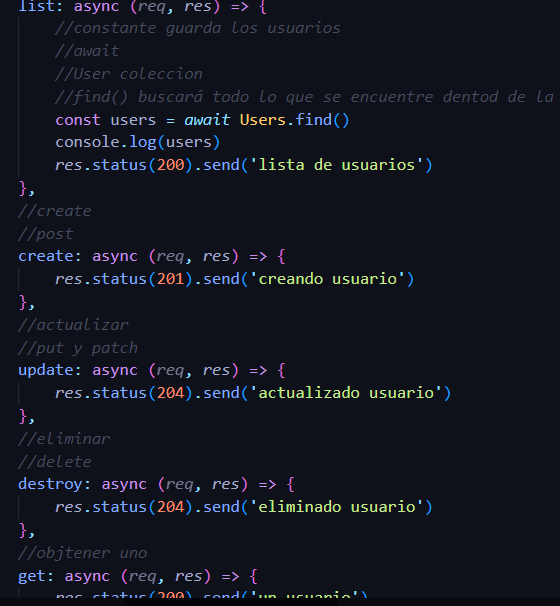
**En el archivo de user.controller**

*//importamos la dependecia User*

const Users = require('./User')

**modificación de los endpoints**

Todas las funciones se volverán asíncronas

****

*//get*

    list: async (*req*, *res*) => {

*//constante guarda los usuarios*

*//await*

*//User coleccion*

*//find() buscará todo lo que se encuentre dentod de la colección User*

        const users = *await* Users.find()

        console.log(users)

*res*.status(200).send(users)

    },

*//create*

*//post*

    create: async (*req*, *res*) => {

*//datos de las peticiones post viende desde body*

        console.log(*req*.body)

*res*.status(201).send('creando usuario')

    },

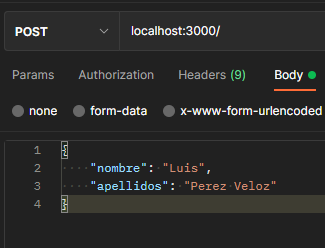
Luego para que el servidor puede recibir datos de json a js

En Api.js luego de asignar el puerto se escribe

*//toma todas las peticiones Json las convierte en js y las va a asignar a body*

app.use(express.json())

En postman se agrega

// los datos en formato JSON

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



Post completo con la base de datos

*//create*

*//post*

    create: async (*req*, *res*) => {

*//datos de las peticiones post viende desde body*

        console.log(*req*.body)

*//constante user instancia de modelo User*

*//new nueva instancia del modelo User*

*//datos que se encuentran en req.params*

        const user = new Users(*req*.body)

*//guardar en la base de datos*

*//await por que devuelve una promesa user.save()*

*//se guarda en una cosntante para imprimir en la consola (no es necesario)*

        const saveUser = *await* user.save()

*res*.status(201).send(saveUser.\_id)

    },

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*//obtener uno*

    get: async (*req*, *res*) => {

*//id es obtenido desde params*

        const{id}=*req*.params

*//se lo busca en la abse de datos*

*// se requiee traer uno findOne*

*//({})el atributo que se desa usar como filtro \_id:id (es el valor traido de params)*

        const user =*await* Users.findOne({\_id:id})

*//se envia el usuario que se encontro*

*res*.status(200).send(user)

    },

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Chat o mensaje de texto

Descripción generada automáticamente

*//actualizar*

*//put y patch*

    update: async (*req*, *res*) => {

*//id es traido desde params*

        const {id}=*req*.params;

*//trae el valor esperado en este caso por id*

        const user= *await* Users.findOne({\_id:id});

*//reemplazar los datos*

*//a quein se va a reemplar user*

*//la peticion que viene desde el cliente*

        Object.assign(user,*req*.body);

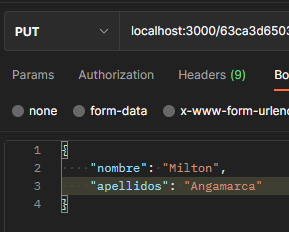
*//guarda*

*await* user.save();

*//estatus*

*res*.sendStatus(204)

    },

 Interfaz de usuario gráfica

Descripción generada automáticamenteTexto

Descripción generada automáticamente

*//eliminar*

*//delete*

    destroy: async (*req*, *res*) => {

*//id es traido desde params*

        const {id}=*req*.params;

*//buscamos el dato por el id*

        const user = *await* Users.findOne({\_id:id});

*//condicion si exite el valor esperado*

*if*(user){

*// si existe lo remueve*

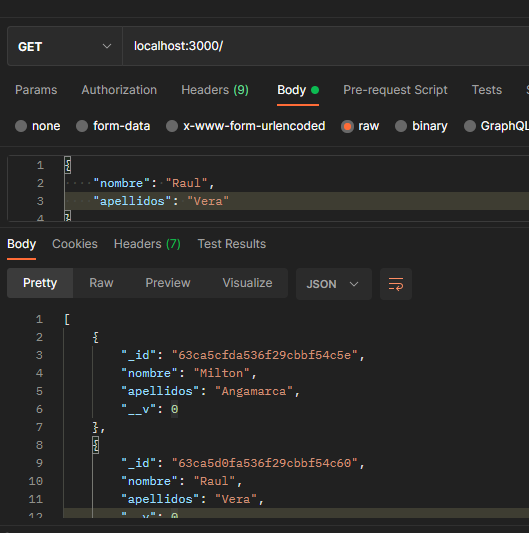
*await* user.remove();

        }

*//envia un estado de ok*

*res*.sendStatus(204)

    },



Texto

Descripción generada automáticamente 

