Curso complementario Desarrollo de Back-end con Node.js - MongoDB

ING - DIEGO CASALLAS **22810019-1**

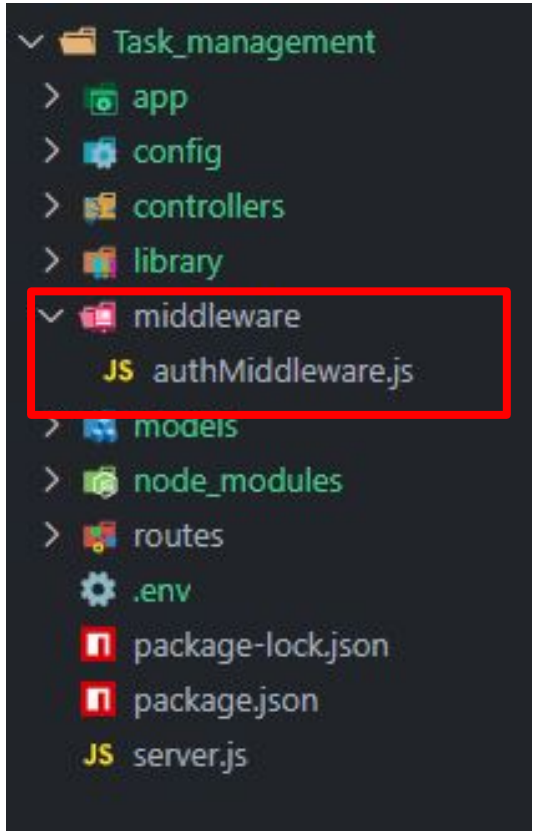
www.sena.edu.co

Proyecto Paso a Paso

- **middleware -> authMiddleware.js**
  - Validar token

Task_management > middleware > JS authMiddleware.js > verifyToken

```javascript
import jwt from "jsonwebtoken";
import dotenv from "dotenv";

dotenv.config();

// Middleware to verify the token
export const verifyToken = (req, res, next) => {
  const token = req.header("Authorization");
  if (!token) return res.status(401).json({ error: "Access denied" });

  try {
    const verified = jwt.verify(token.replace("Bearer ", ""), process.env.JWT_SECRET);
    req.user = verified;
    //See data token encrypted
    //console.log(verified);
    next();
  } catch (err) {
    res.status(400).json({ error: "Invalid Token" });
  }
};
```

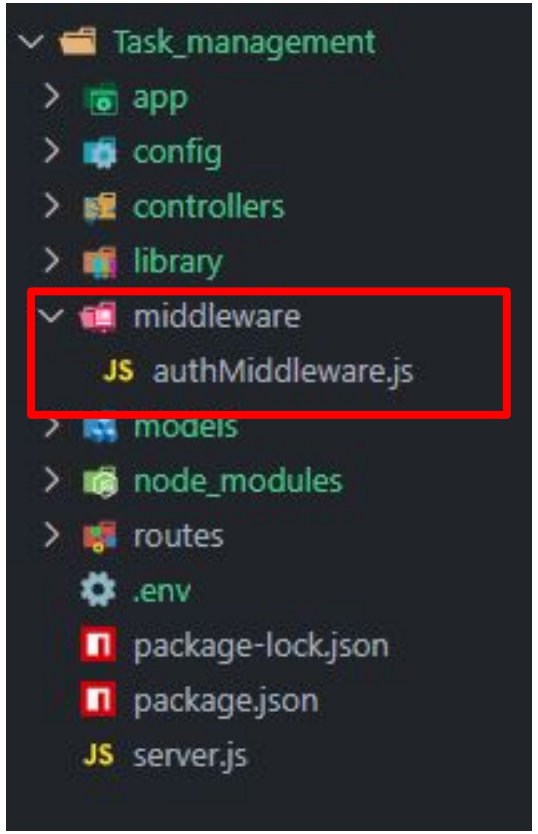


Proyecto Paso a Paso

- **Implementar en las rutas**
  - llamar el middleware a la ruta que se va a implementar.

Task_management > routes > JS auth.routes.js > default

```javascript
import { Router } from "express";
import AuthController from '../controllers/auth_controller.js';
import {verifyToken} from '../middleware/authMiddleware.js';
const router = Router();

// Public route
router.post('/auth/register',verifyToken, AuthController.register);
router.post('/auth/login', AuthController.login);

export default router;
```



Proyecto Paso a Paso

- **Probar**

TASK-MONGO-NODE / auth / **Register**

POST http://localhost:3000/api/auth/register

Params | Authorization | Headers (10) | Body | Scripts | Tests | Settings

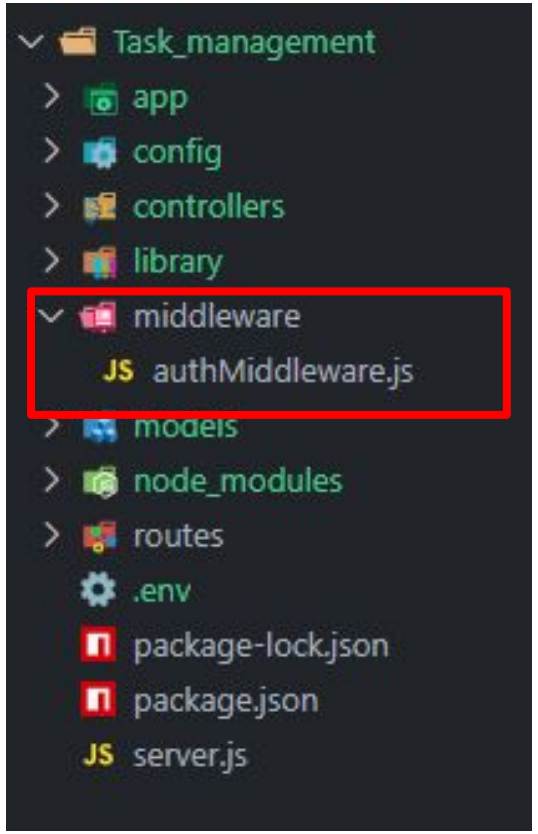
none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

```
1 {
2     "username":"diehercasvan1",
3     "email":"diehercasvan1@gmail.com",
4     "password":"Die80059877*"
5 }
```

Body | Cookies (1) | Headers (7) | Test Results | 401 Unauthorized

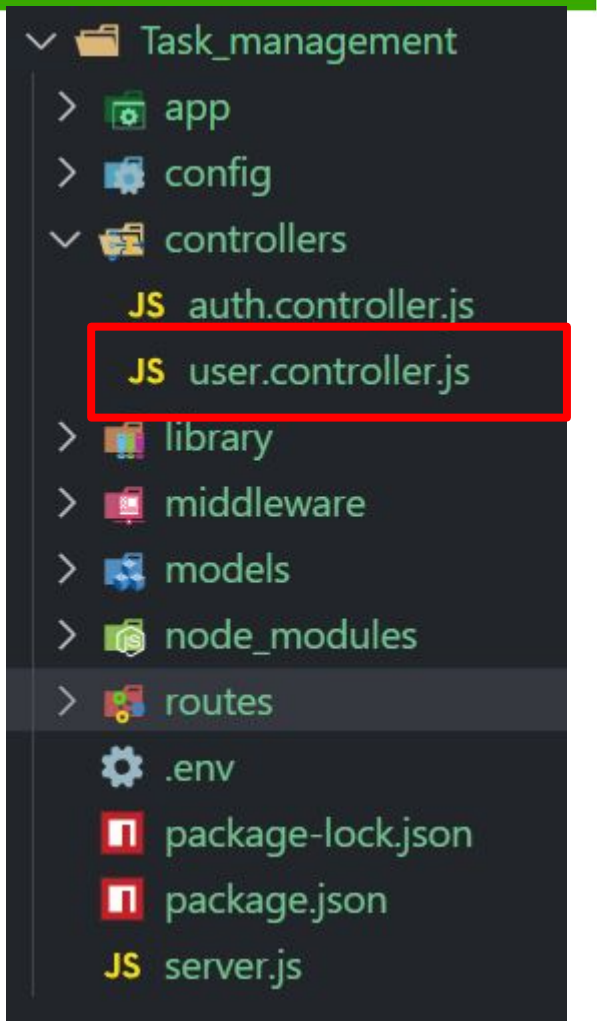
{} JSON | Preview | Visualize

```
1 {
2     "error": "Access denied"
3 }
```



Proyecto Paso a Paso

- **CRUD de usuario**
  - Usamos el mismo modelo de **user**
  - Crear el controlador
  - Crear la ruta
    - Implementar middleware
  - **Realizar pruebas**

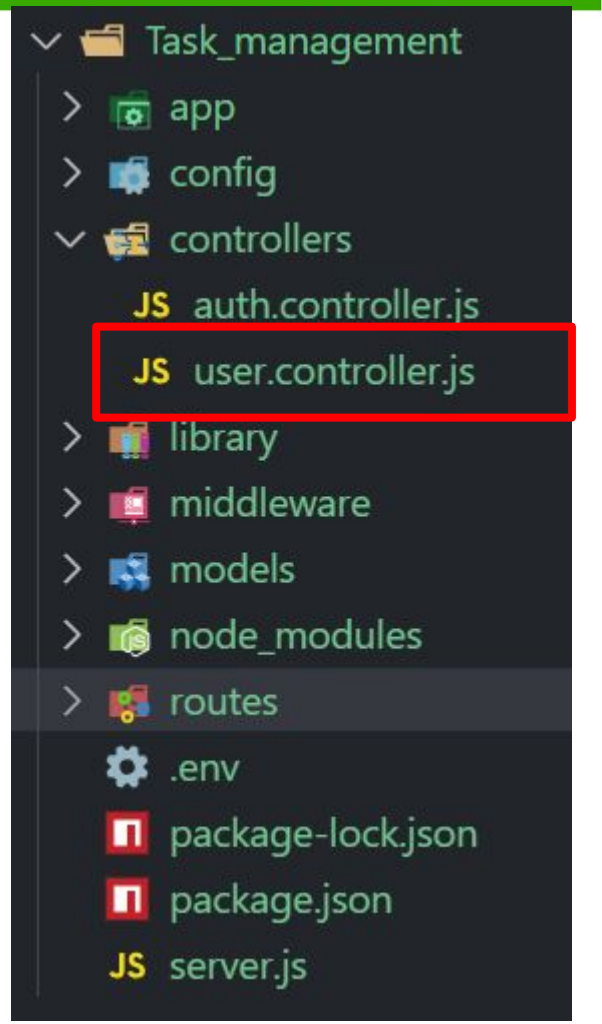


Proyecto Paso a Paso

- **controllers ->user.controller.js**

Task_management > controllers > JS user.controller.js > UserController

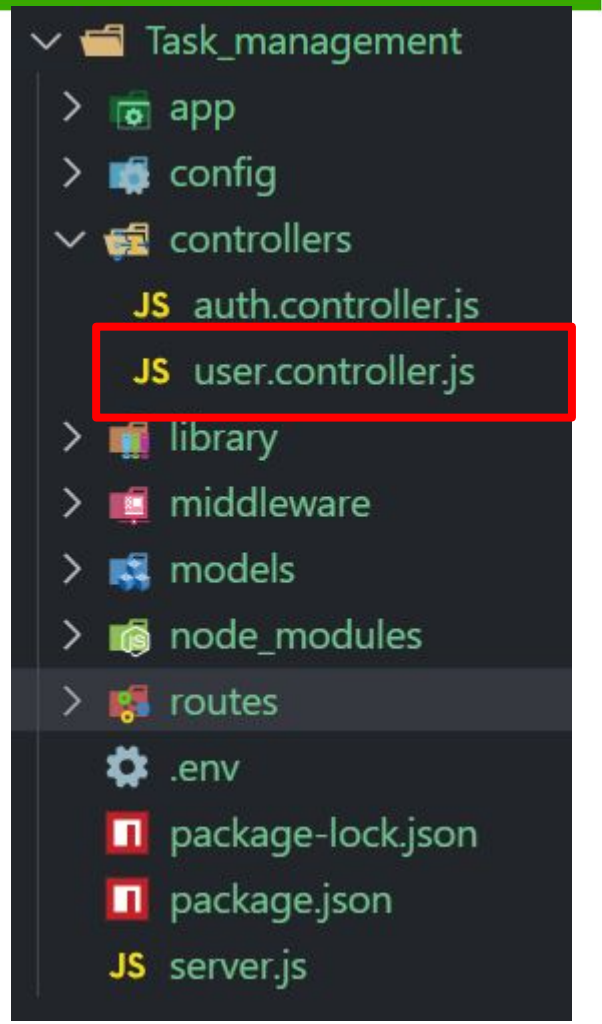
```javascript
import UserModel from '../models/user.model.js';//Import the UserModel
import dotenv from 'dotenv';

dotenv.config();
class UserController {
  // User registration
  async addUser(req, res) {
    const passwordRegex = /^(?=.*\d)(?=.*[\u0021-\u002b\u003c-\u0040])(?=.*[A-Z])(?=.*[a-z])\S{8,16}$/;
    try {
      const { username, email, password } = req.body;
      if (!username || !email || !password) {
        return res.status(400).json({ error: 'All data is mandatory for entry' });
      }
      if (!passwordRegex.test(password)) {
        return res.status(400).json({ error: 'The password must be between 8 and 16 characters long, with at least one digit, at least one lowercase letter, at least one uppercase letter, and at least one non-alphanumeric character.' });
      }
      const existingUserModel = await UserModel.findOne({ email });
      if (existingUserModel) {
        return res.status(400).json({ error: 'The user already exists' });
      }
      const newUserModel = new UserModel({ username, email, password });
      await newUserModel.save();
      return res.status(201).json({ message: 'User successfully registered' });
    } catch (err) {
      res.status(400).json({ error: err.message });
    }
  };
```



Proyecto Paso a Paso

- **controllers ->user.controller.js**

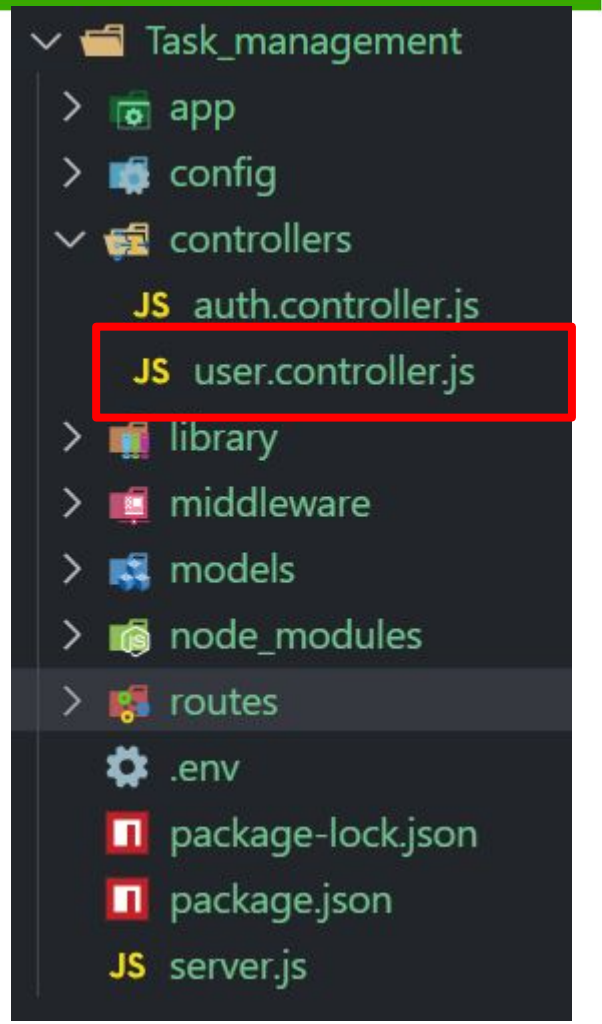
```javascript
// Users list
async show(req, res) {
  try {
    const userModel = await UserModel.find();
    if (!userModel) throw new Error('User not found');
    return res.status(200).json({ data: userModel });
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
};
```



Proyecto Paso a Paso

- **controllers ->user.controller.js**

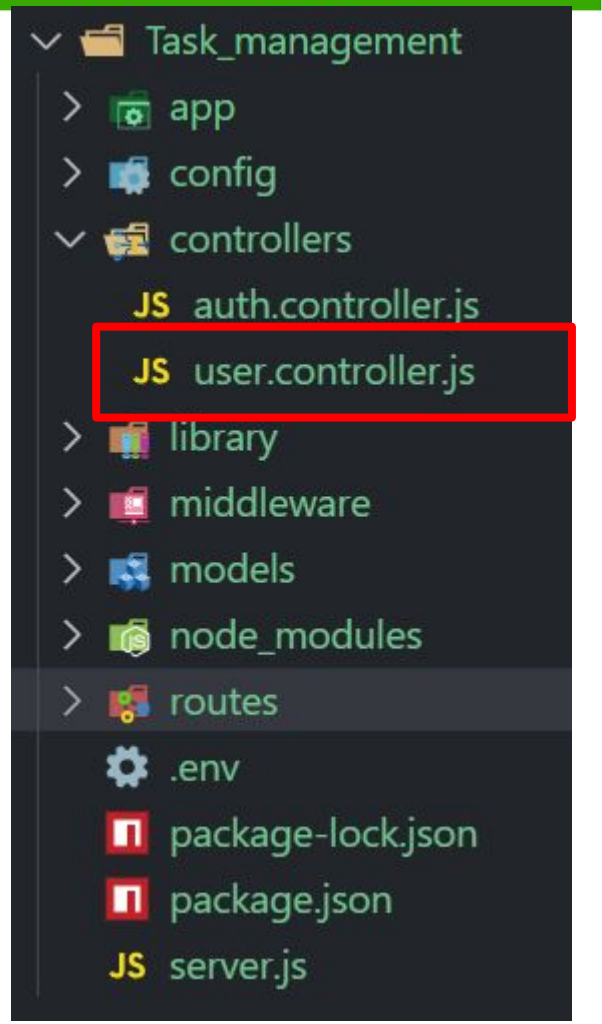
```javascript
// Find user by ID
async findById(req, res) {
  try {
    const id = req.params.id;
    if (!id) {
      return res.status(400).json({ error: 'User Id is required' });
    }
    const userModel = await UserModel.findOne({ _id: id });
    if (!userModel) throw new Error('User not found');
    return res.status(200).json({ data: userModel });
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
};
```



Proyecto Paso a Paso

- **controllers ->user.controller.js**

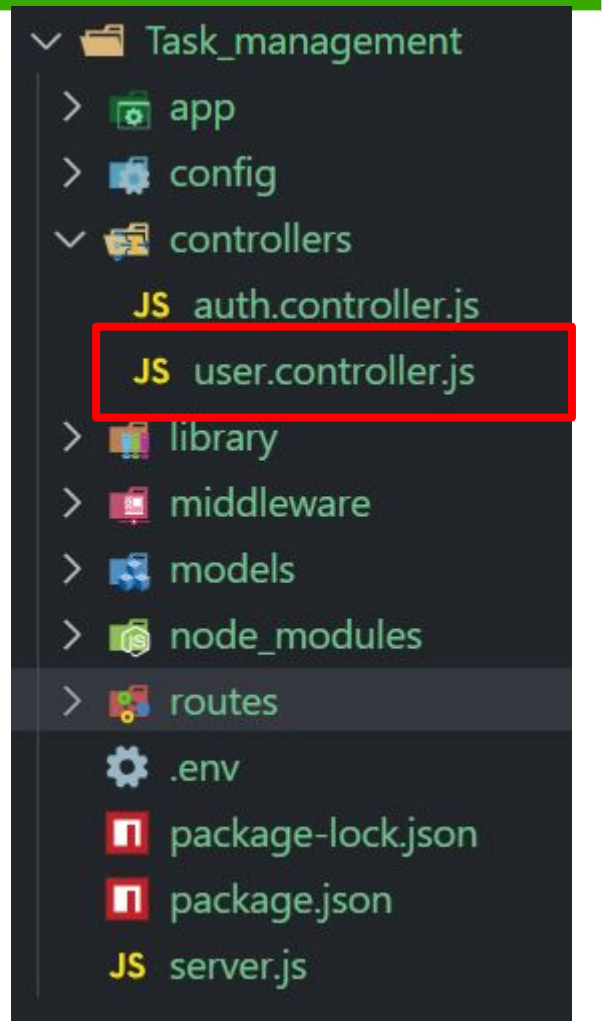
```javascript
// User update
async update(req, res) {
  const passwordRegex = /^(?=.*\d)(?=.*[\u0021-\u002b\u003c-\u0040])(?=.*[A-Z])(?=.*[a-z])\S{8,16}$/;
  try {
    const { username, email, password } = req.body;
    if (!username || !email || !password) {
      return res.status(400).json({ error: 'All data is mandatory for entry' });
    }
    if (!passwordRegex.test(password)) {
      return res.status(400).json({ error: 'The password must be between 8 and 16 characters long, with at least one digit, at least one lowercase letter, at least one uppercase letter, and at least one non-alphanumeric character.' });
    }
    const existingUserModel = await UserModel.findOne({ email });
    if (!existingUserModel) {
      return res.status(400).json({ error: 'User not found' });
    }
    const updateUser = await UserModel.findOneAndUpdate(
      { _id: req.params.id }, // Filtro
      { username, password }, // Datos nuevos
      { new: true } // Devuelve el documento actualizado
    );
    if (!updateUser) {
      return res.status(404).json({ error: 'User not updated' });
    }
    res.status(200).json(updateUser);
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
};
```



Proyecto Paso a Paso

- **controllers ->user.controller.js**

```javascript
  // User deletion
  async delete(req, res) {
    try {
      const deletedUser = await UserModel.findOneAndDelete({
        _id: req.params.id
      });
      if (!deletedUser) {
        return res.status(404).json({ error: 'User not updated' });
      }
      res.status(200).json({ message: 'Deleted successfully' });
    } catch (error) {
      res.status(500).json({ error: 'Error deleting user' });
    }
  };

}
export default new UserController();
```



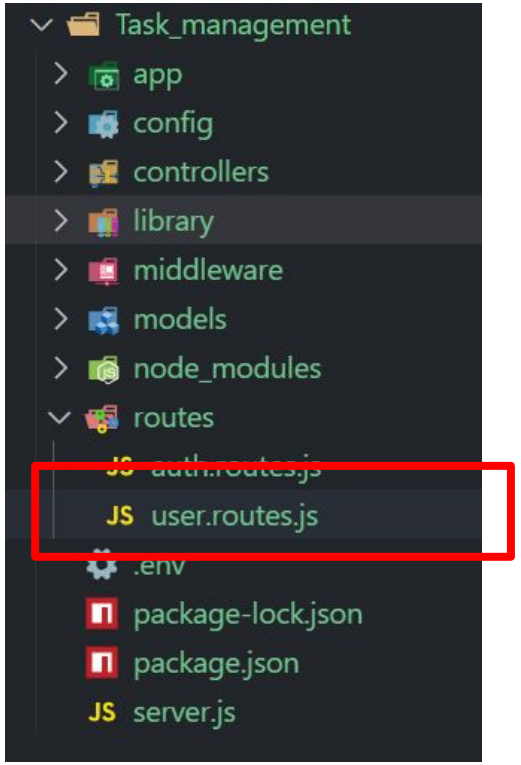
Proyecto Paso a Paso

- **routes ->user.routes.js**

Task_management > routes > JS user.routes.js > ...

```js
import { Router } from "express";
import UserController from '../controllers/user.controller.js';
import {verifyToken} from '../middleware/authMiddleware.js';

const router = Router();
const name="/user";
// Verify token middleware
router.use(verifyToken);
// Route for user registration and list
router.route(name)
.post(UserController.addUser)
.get(UserController.show);
//Route for user by ID
router.route(`${name}/:id`)
  .get(UserController.findById)
  .put(UserController.update)
  .delete(UserController.delete);

export default router;
```

CONTROLLER



Proyecto Paso a Paso

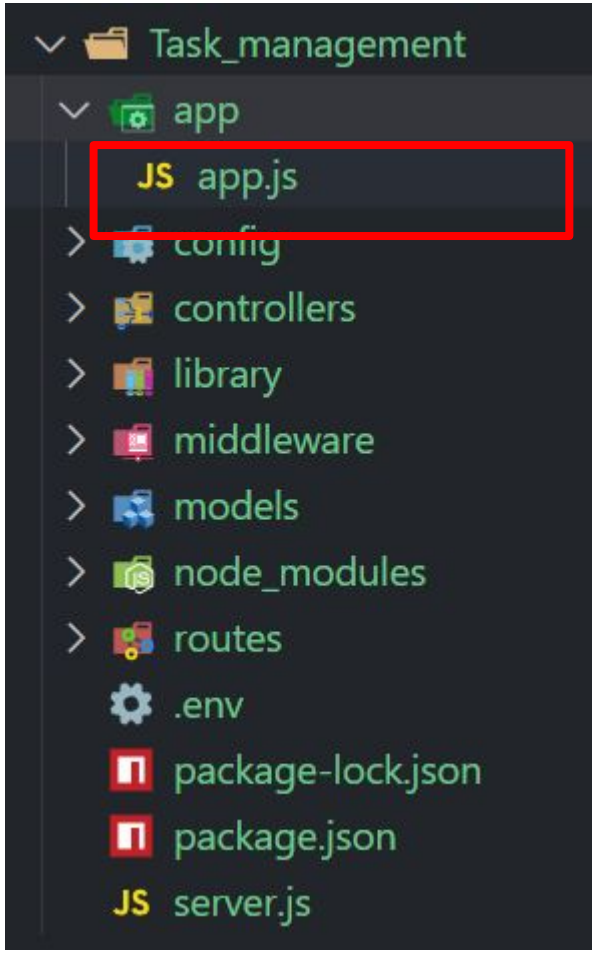
- **app ->app.js**

ROUTER

ADD ROUTER

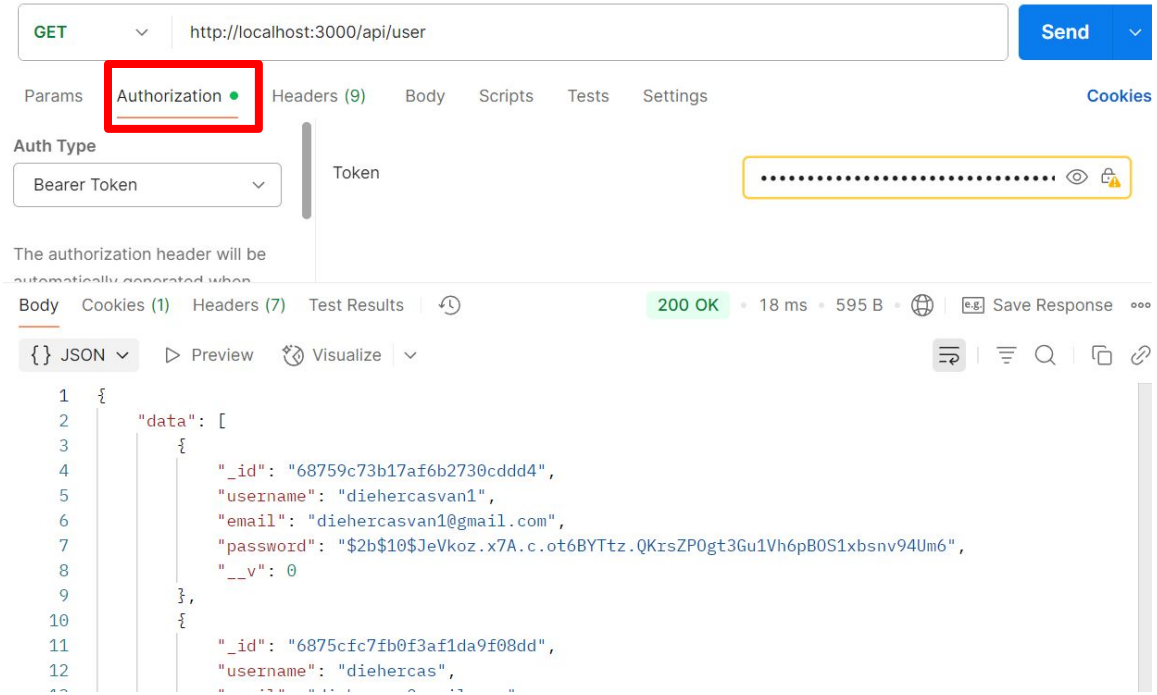
Task_management > app > JS app.js > ...

```js
import express from 'express';
import { connectDB } from '../config/db/connection.js';
// Import routes
import authRouter from '../routes/auth.routes.js';
import userRouter from '../routes/user.routes.js';

const app = express();
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Connect to MongoDB
connectDB();// Call the connection function

// Use routes
app.use('/api', authRouter);
app.use('/api', userRouter);

app.use((rep, res, nex) => {
  res.status(404).json({
    message: 'Endpoint losses'
  });
});

export default app;
```



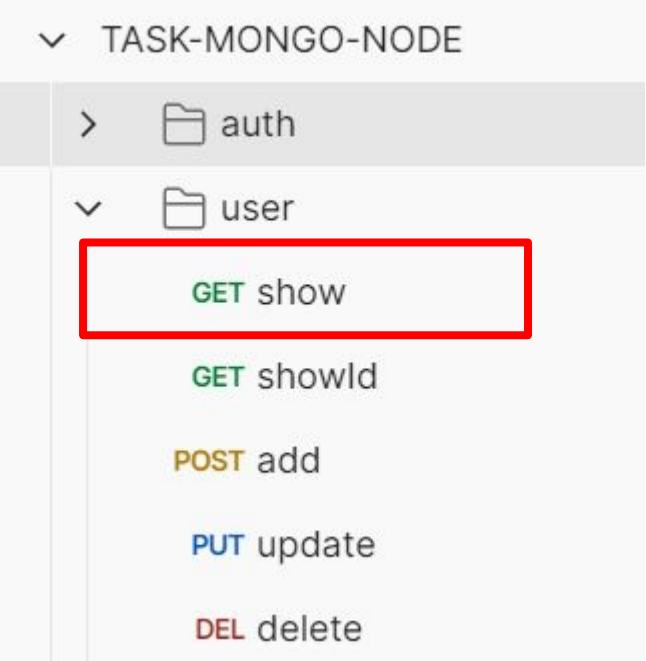
Proyecto Paso a Paso

- **Pruebas**



TOKEN

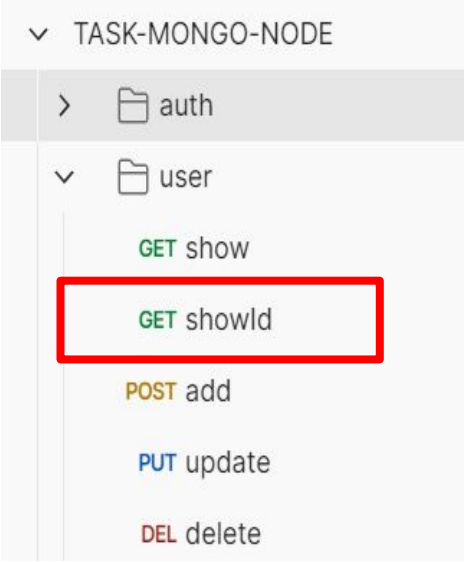
RESULTADO



Método	Ruta	Controlador	Modelo
GET	http://localhost:3000/api/user	User	User

Proyecto Paso a Paso

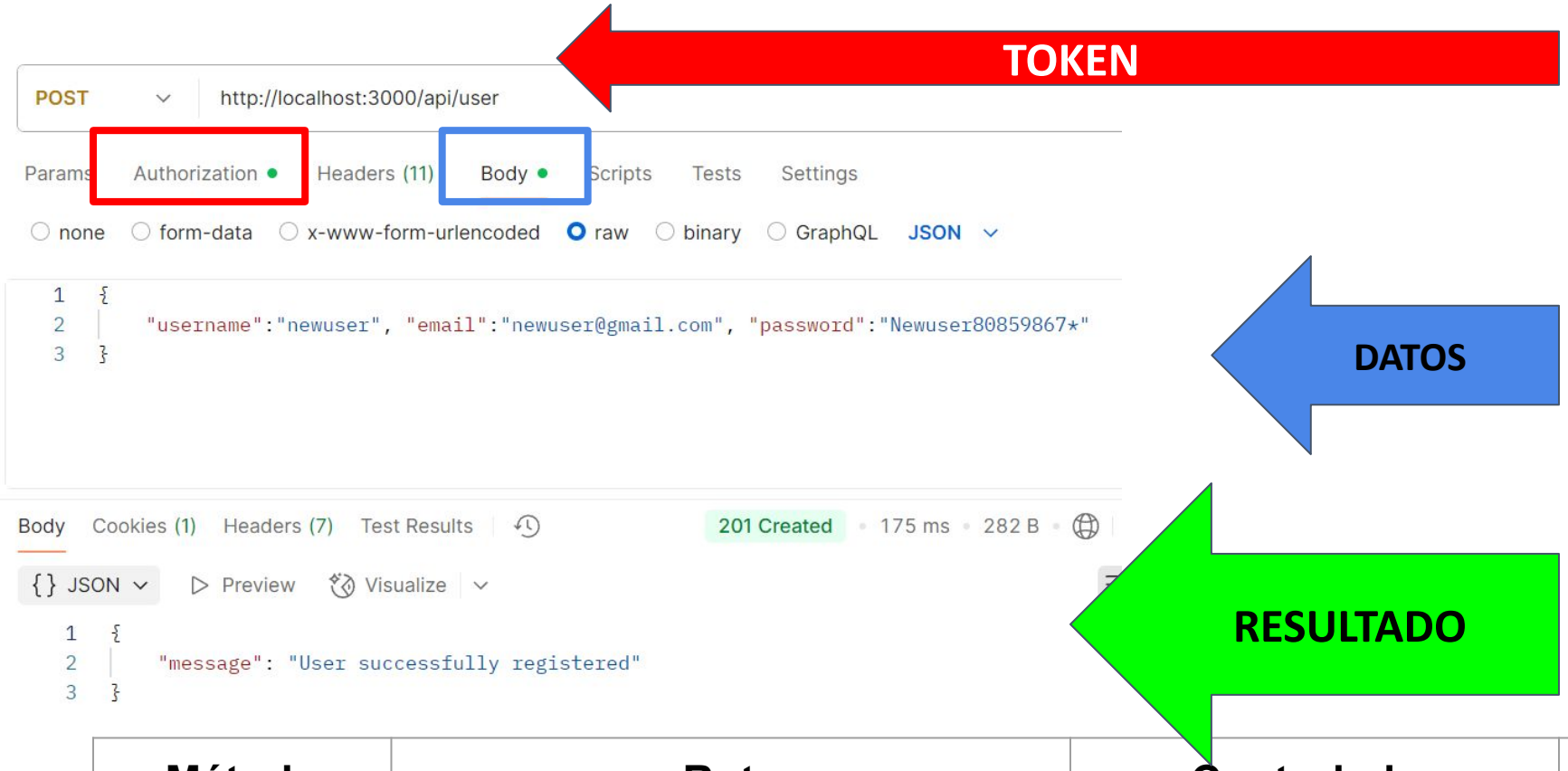
- **Pruebas**

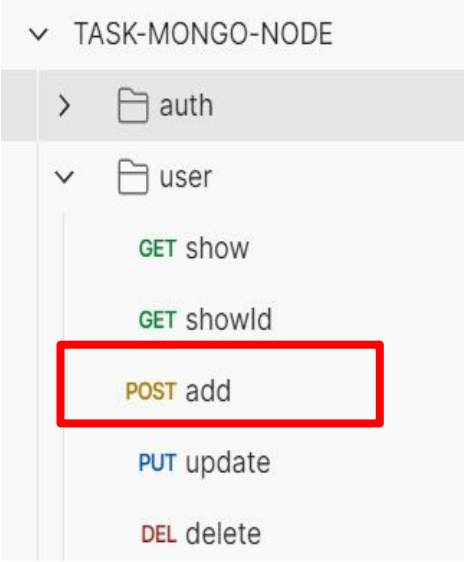


Método	Ruta	Controlador	Modelo
GET	http://localhost:3000/api/user:id	User	User

Proyecto Paso a Paso

- **Pruebas**





Método	Ruta	Controlador	Modelo
POST	http://localhost:3000/api/user	User	User

Proyecto Paso a Paso

- **Pruebas**

TOKEN

PUT http://localhost:3000/api/user/68768746acdd080a7c6e3b6f

ID

Params Authorization Headers (11) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "username":"NewUse", "email":"newuser@gmail.com","password":"Newuser80800867*"
3 }
```

DATOS

Body Cookies (1) Headers (7) Test Results 200 OK 35 ms 356 B

{} JSON Preview Visualize

```
1 {
2     "_id": "68768746acdd080a7c6e3b6f",
3     "username": "NewUse",
4     "email": "newuser@gmail.com",
5     "password": "Newuser80800867*",
6     "__v": 0
7 }
```

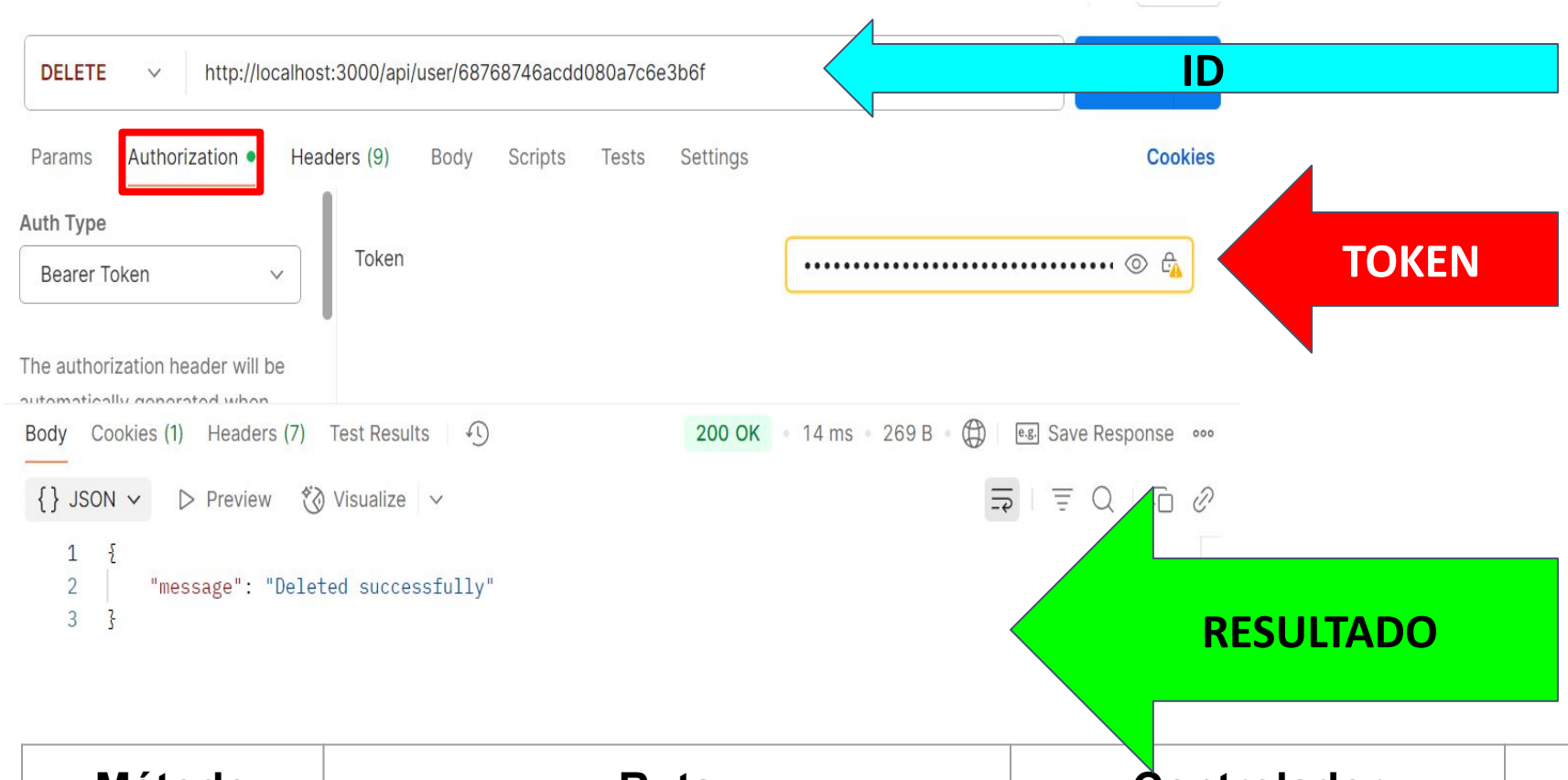
RESULTADO

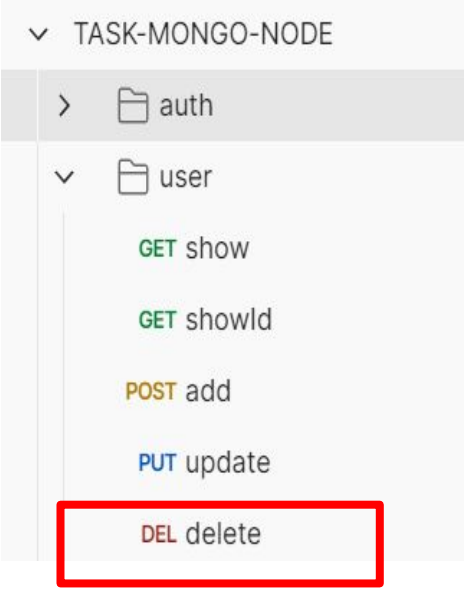
TASK-MONGO-NODE
- auth
- user
  - GET show
  - GET showId
  - POST add
  - PUT update
  - DEL delete

Método	Ruta	Controlador	Modelo
PUT	http://localhost:3000/api/user:id	User	User

Proyecto Paso a Paso

- **Pruebas**





Método	Ruta	Controlador	Modelo
DELETE	http://localhost:3000/api/user:id	User	User

Actividad

- **Crear el modelo de datos para almacenar tareas**
  - **Con los siguientes campos:**
    - **Información básica:**
      - **_id: Identificador único de MongoDB**
      - **título, descripción: Detalles de la tarea**
      - **tipo: Categoría de la tarea (Desarrollo, Diseño, Reunión, etc.)**
      - **prioridad: Nivel de prioridad (Alta, Media, Baja)**
      - **color: Código hexadecimal para identificación visual**
      - **estado: Estado actual (Pendiente, En progreso, Completada, etc.)**
    - **Fechas:**
      - **fechaInicio, fechaFin: Rango de tiempo para la tarea**
      - **fechaCreacion: Cuando se creó el registro**
      - **ultimaModificacion: Última actualización**
    - **Personas involucradas:**
      - **correoNotificacion: Email para notificaciones**
      - **encargados: Array con los responsables**
      - **creadoPor: Quién creó la tarea**
    - **Archivos adjuntos:**
      - **archivos: Array con documentos relacionados**
    - **Otros:**
      - **comentarios: Discusiones sobre la tarea**
      - **etiquetas: Palabras clave**
      - **proyectoId: Proyecto al que pertenece**

Documentación Oficial

https://mongoosejs.com/docs/guide.html

- **Crear modelo**
- **Crear el controlador**
- **Crear las rutas**
  - **validar token**
- **Realizar las pruebas (Postman)**
  - **Crear documentación**

Mongoose JS con Node.js

- **¿Qué es Mongoose?**
  - ODM (Object Document Mapper):
  - Permite interactuar con MongoDB usando objetos y esquemas en JavaScript.
- **Ventajas:**
  - Validación de datos.
  - Middlewares (hooks).
  - Consultas más intuitivas.
- **Diferencia con el Driver Nativo:**
  - Mongoose añade estructura, mientras que el driver trabaja con MongoDB directamente.

Mongoose JS con Node.js

- **Instalación y Configuración**

```
npm install mongoose
```

- **Conexión a MongoDB:**

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/miDB', {
  useNewUrlParser: true,
  useUnifiedTopology: true
 });
```

Mongoose JS con Node.js

- **Esquemas (Schemas) y Modelos**

```
const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  age: { type: Number, min: 18 },
  email: { type: String, unique: true }
});
```

- **Modelo: Representación de una colección en la DB.**

```
const User = mongoose.model('User', userSchema);
```

Mongoose JS con Node.js

- **Operaciones CRUD**
  - **Create**

```
const newUser = new User({ name: "Ana", age: 25 });
newUser.save();
```

  - **Read:**

```
User.find({ age: { $gte: 18 } });
```

  - **Update:**

```
User.updateOne({ name: "Ana" }, { age: 26 });
```

  - **Delete:**

```
User.deleteOne({ name: "Ana" });
```

Mongoose JS con Node.js

- **Middlewares (Hooks)**
    - **Usos comunes:**
    - Encriptar contraseñas antes de guardar.
    - Validaciones personalizadas.

```
userSchema.pre('save', function(next) {
  console.log("Guardando usuario...");
  next();
 });
```

Mongoose JS con Node.js

- **Populate (Relaciones entre Colecciones)**
  - **con Referencias:**

```
const postSchema = new mongoose.Schema({
  title: String,
  author: { type: mongoose.Schema.Types.ObjectId, ref: 'User' }
 });
```

  - **con Populate:**

```
Post.findOne({ title: "Hola Mundo" }).populate('author');
```

Mongoose JS con Node.js

- **Ejemplo**

```
import mongoose from 'mongoose';
const { Schema } = mongoose;

const blogSchema = new Schema({
      title: String, // String is shorthand for {type: String}
      author: String,
      body: String,
      comments: [{ body: String, date: Date }],
      date: { type: Date, default: Date.now },
      hidden: Boolean,
      meta: {
        votes: Number,
        favs: Number
  }
 });
```

Mongoose JS con Node.js

- **Recursos Adicionales**

  - Documentación Oficial: mongoosejs.com
  - Repositorio GitHub: github.com/Automattic/mongoose
  - Curso Recomendado: MongoDB University

G R A C I A S

Línea de atención al ciudadano: 01 8000 910270
Línea de atención al empresario: 01 8000 910682

@SENAComunica

www.sena.edu.co