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

ING - DIEGO CASALLAS **22810019-1**

Proyecto Paso a Paso

Vamos a desarrollar una API RESTful con Node.js + Express + MongoDB para gestionar tareas (To-Do List) con autenticación de usuarios.

1. Configuración Inicial (2 horas)

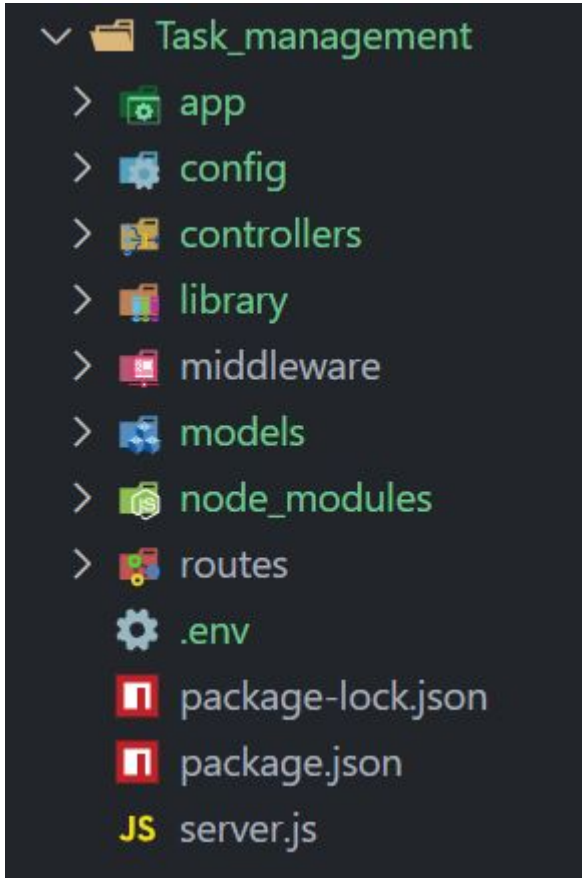
1.1. Crear el proyecto e instalar dependencias

- mkdir “Nombre del proyecto”
- cd todo-app
- npm init -y (“Digitar por consola”)
- npm install express mongoose body-parser dotenv ejs bcrypt jsonwebtoken cookie-parser nodemon nodemailer multer

Proyecto Paso a Paso

1.2. Estructura básica del proyecto

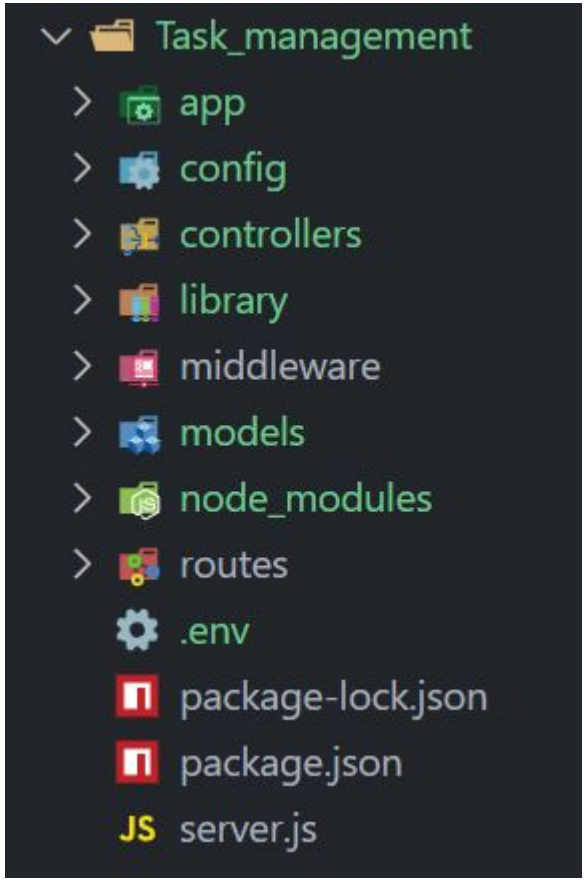
- **app (archivos de la aplicación)**
  - Archivo de la aplicación
- **config (archivos de configuración)**
  - Conexión a base de datos
- **controllers (controladores de la aplicación)**
  - Controladores para el manejo de las peticiones de las rutas al modelo



Proyecto Paso a Paso

1.2. Estructura básica del proyecto

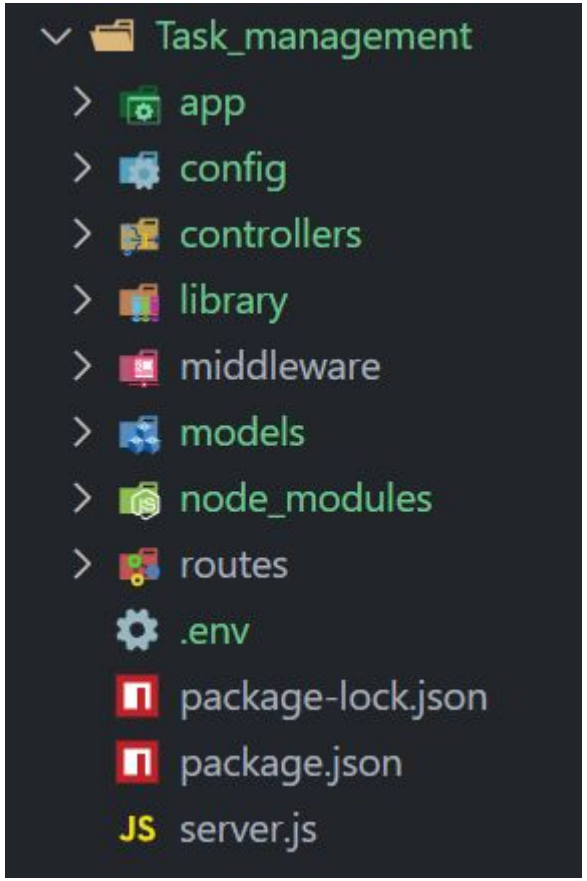
- **library (librerías de la aplicación)**
  - Archivo con funcionalidades específicas
- **middleware (funcionalidades específicas)**
  - Validar token en cada ruta
- **models (modelo de datos de la aplicación)**
  - Conjunto de datos para operar CRUD



Proyecto Paso a Paso

1.2. Estructura básica del proyecto

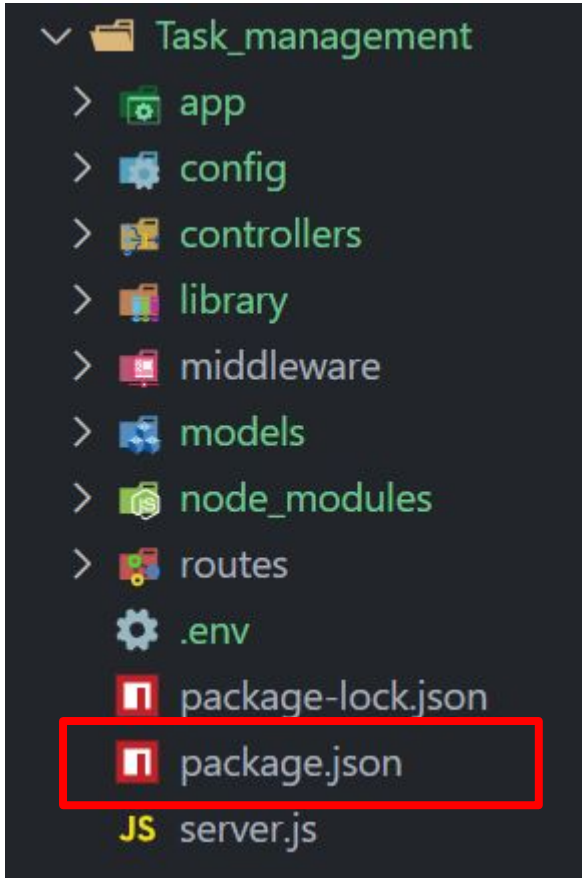
- **node_modules(componentes de npm)**
  - Archivos de dependencias
- **routes (manejo de rutas amigables)**
  - Rutas de la aplicación
- **env**
  - Variables del entorno
- **package.json**
  - Archivos de dependencias del proyecto



Proyecto Paso a Paso

- **package.json**
  - Ajustar lo siguiente:

```
"main": "server.js",
"type": "module",
▷Debug
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "node server.js",
  "dev": "nodemon server.js"
},
```

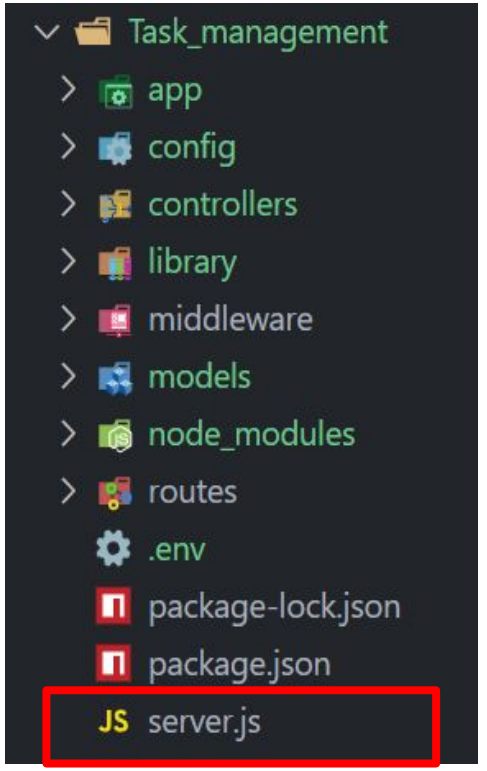


Proyecto Paso a Paso

1.3. Configurar
server.js

Task_management > JS server.js > ...

```
/**
 * Author:Diego Casallas
 * Date: 14/07/2025
 * Description: This is the main server file for the backend of the application.
*/

import app from './app/app.js';
import dotenv from 'dotenv';

dotenv.config();

const PORT = process.env.SERVER_PORT || 3000;

app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

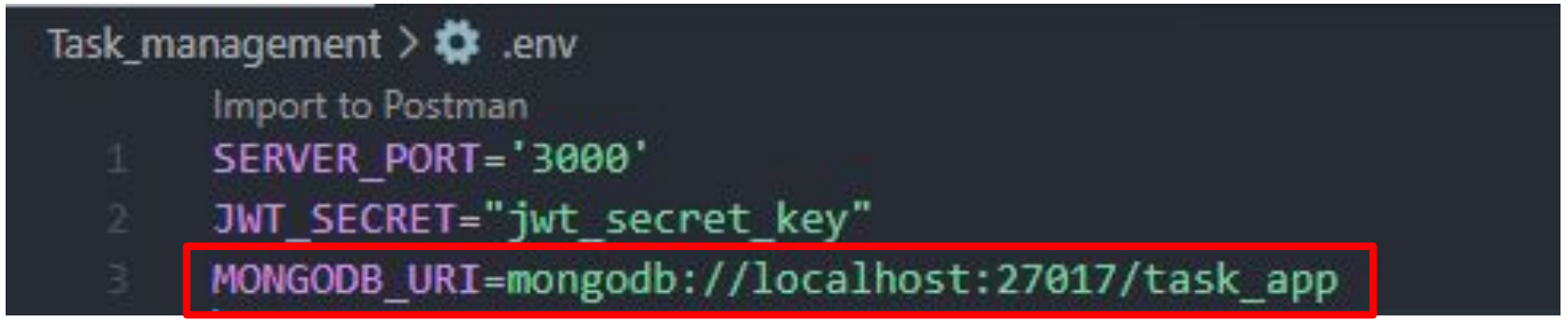


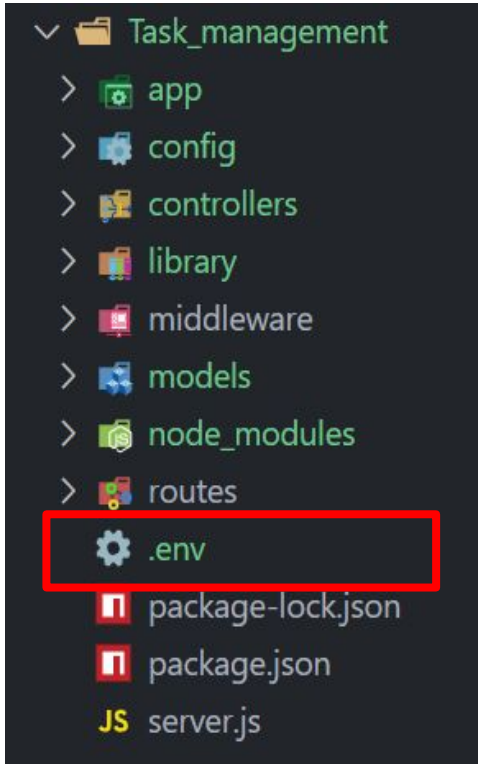
Proyecto Paso a Paso

1.4. Crear .env

- **env**

Tener presente para la configuración de la base de datos en mongo



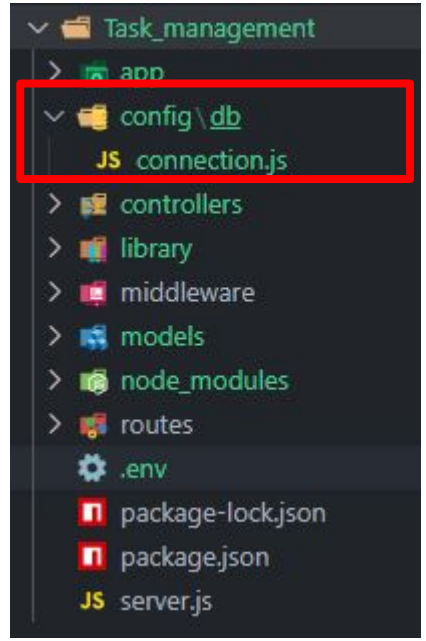


Proyecto Paso a Paso

- **config/db -> connection.js**

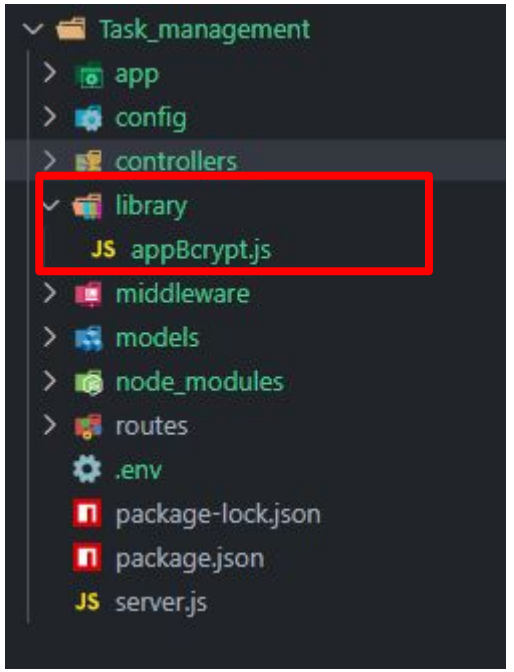
Task_management > config > db > JS connection.js > ...

```
import mongoose from 'mongoose';
import dotenv from 'dotenv';

dotenv.config();
const MONGODB_URI = process.env.MONGODB_URI || 'mongodb://127.0.0.1:27017/task_app';
// Function to connect to MongoDB
export const connectDB = async () => {
  try {
    await mongoose.connect(MONGODB_URI);
    console.log('✅ Connected to MongoDB');
  } catch (error) {
    console.error('❌ Error connecting to MongoDB:', error.message);
    process.exit(1); // Exit the process with error
  }
};
// Connection events
mongoose.connection.on('connected', () => {
  //console.log(' Connected to MongoDB');
});
mongoose.connection.on('error', (err) => {
  console.log('Error connecting to MongoDB:', err);
});
mongoose.connection.on('disconnected', () => {
  console.log('Mongoose offline');
});
// Export the connection and mongoose in case it is needed
export { mongoose };
```



Proyecto Paso a Paso

- **library -> appBcrypt.js**



Task_management > library > JS appBcrypt.js > ...

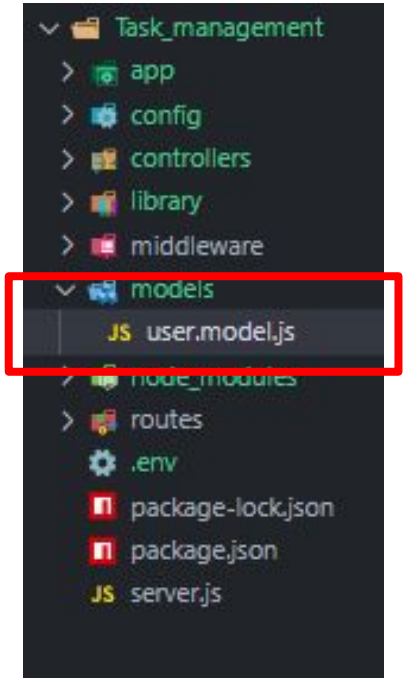
```
import appBcrypt from 'bcrypt';
const saltRounds = 10;

export const encryptPassword = async (password) => {
    try {
        const hashedPassword = await appBcrypt.hash(password, saltRounds);
        return hashedPassword;
    } catch (error) {
        console.error('Error encrypt:', error);
        throw error;
    }
};

export const comparePassword = async (password, hashedPassword) => {
    try {
        const match = await appBcrypt.compare(password, hashedPassword);
        return match;
    } catch (error) {
        console.error('Error compare the hash:', error);
        throw error;
    }
};
```

Proyecto Paso a Paso

- **models -> user.model.js**

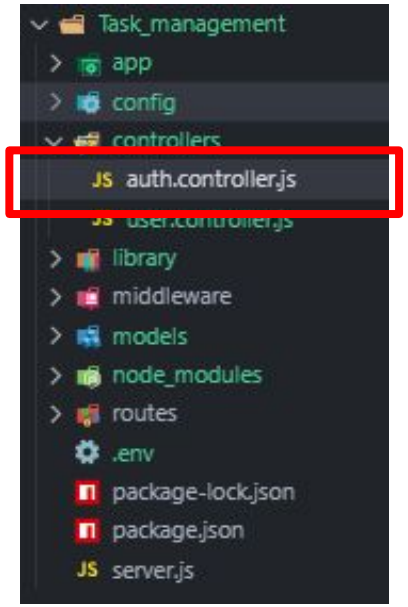
Task_management > models > JS user.model.js > userSchema.pre('save') callback

```
import { mongoose } from '../config/db/connection.js'; // Import mongoose from the connection
import { encryptPassword  } from '../library/appBcrypt.js';

const userSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
});

// Middleware for password hashing
userSchema.pre('save', async function (next) {
  if (!this.isModified('password')) return next();

  try {
    this.password = await encryptPassword(this.password);
    next();
  } catch (error) {
    next(error);
  }
});

export default mongoose.model('User', userSchema);
```



Proyecto Paso a Paso

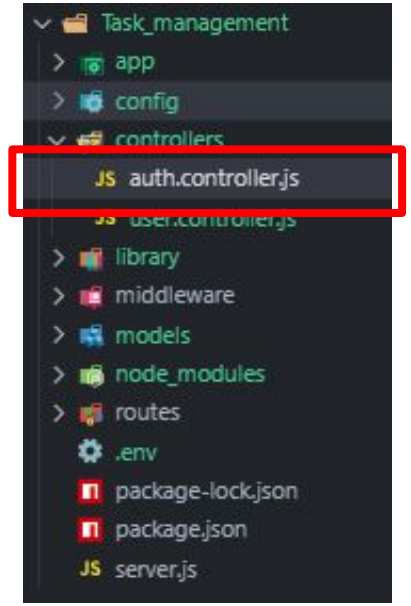
- **controllers -> auth.controller.js**



Proyecto Paso a Paso

- **controllers -> auth.controller.js**

```
// Login
  async login(req, res) {
    try {
      const { email, password } = req.body;
      const userModel = await UserModel.findOne({ email });
      if (!userModel) throw new Error('User not found');

      const isMatch = await comparePassword(password, userModel.password);
      if (!isMatch) throw new Error('Incorrect password');

      const token = jwt.sign({ id: userModel._id }, process.env.JWT_SECRET, { expiresIn: '1h' });
      res.cookie('token', token, { httpOnly: true });
      res.status(200).json({ message: 'Successful login', token: token });
    } catch (err) {
      res.status(400).json({ error: err.message });
    }
  };

}

export default new AuthController();
```

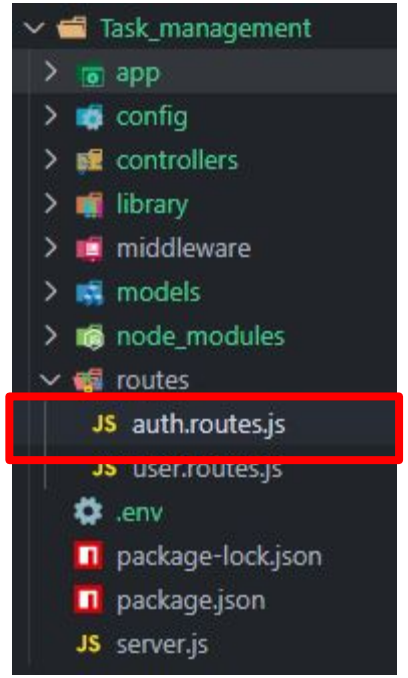


Proyecto Paso a Paso

- **routes -> auth.routes.js**

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

```
1   import { Router } from "express";
2   import AuthController from '../controllers/auth.controller.js';
3
4   const router = Router();
5
6   // Public route
7   router.post('/auth/register', AuthController.register);
8   router.post('/auth/login', AuthController.login);
9
10  export default router;
```

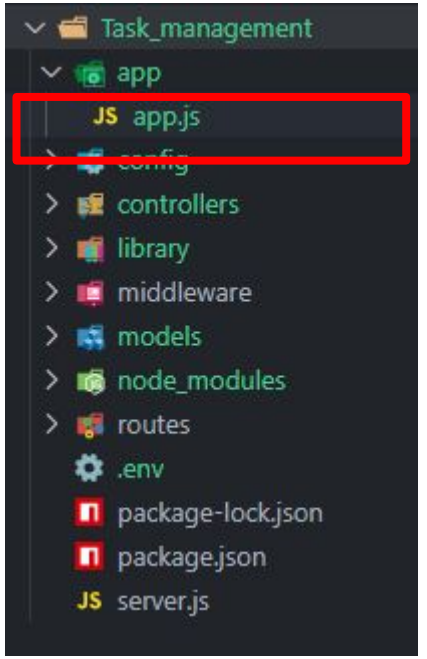


Proyecto Paso a Paso

- **app-> app.js**

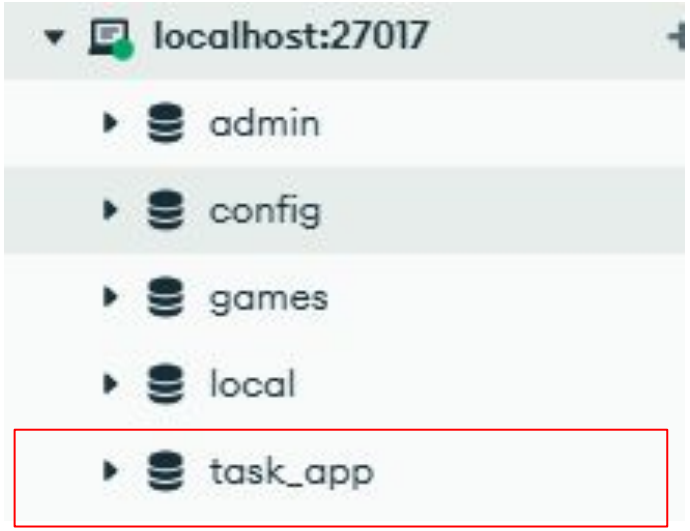
Task_management > app > JS app.js > ...

```
import express from 'express';
import { connectDB } from '../config/db/connection.js'; // Import the connection
// Import routes
import authRouter from '../routes/auth.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((rep, res, nex) => {
  res.status(404).json({
    message: 'Endpoint losses'
  });
});

export default app;
```



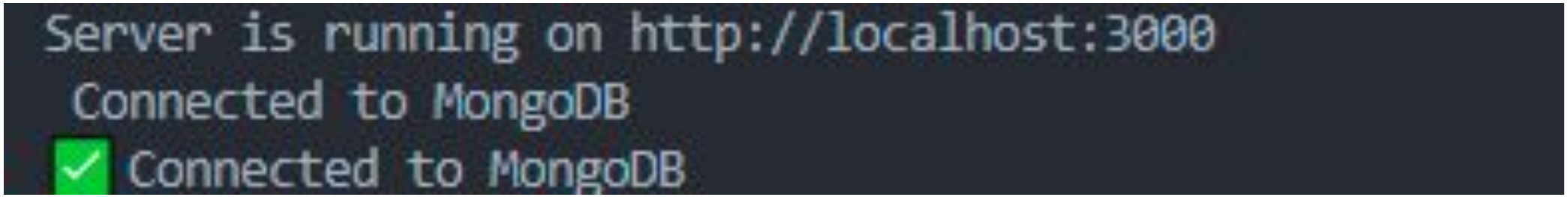
Proyecto Paso a Paso

- **Configurar base de datos Mongo**
  - Abrir mongo compass
  - Crear base de datos
    - task_app



Proyecto Paso a Paso

- **Subir los servicios del servidor**
  - Buscar la raíz del proyecto
  - Ejecutar **npm run dev**

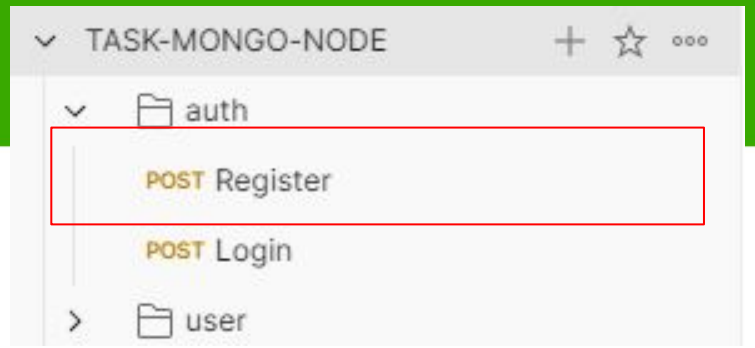


Proyecto Paso a Paso

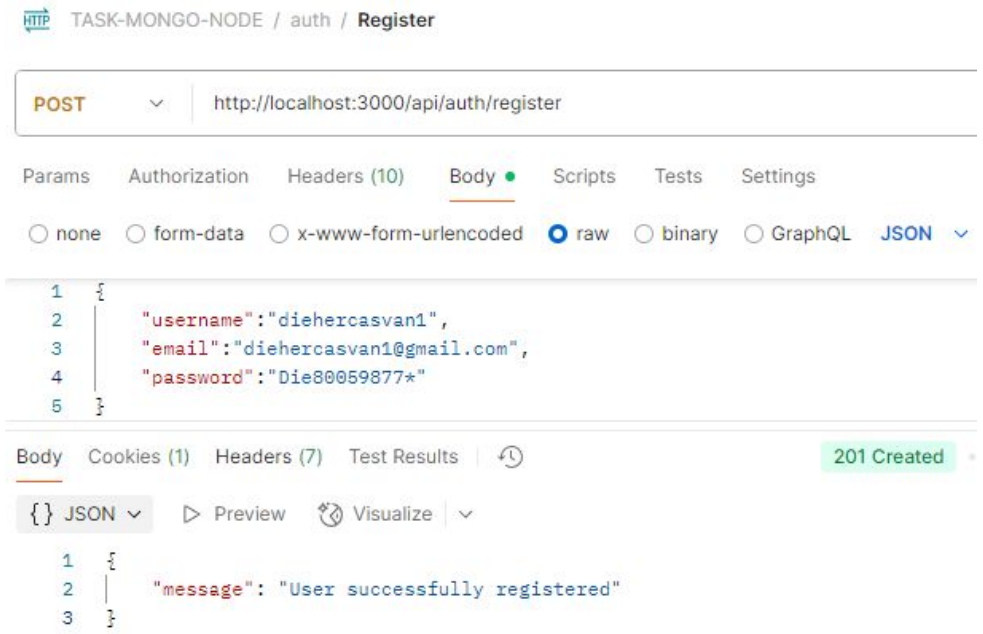
Pruebas

- **Ejecutar la aplicación postman**
  - **Crear una colección**

Proyecto Paso a Paso

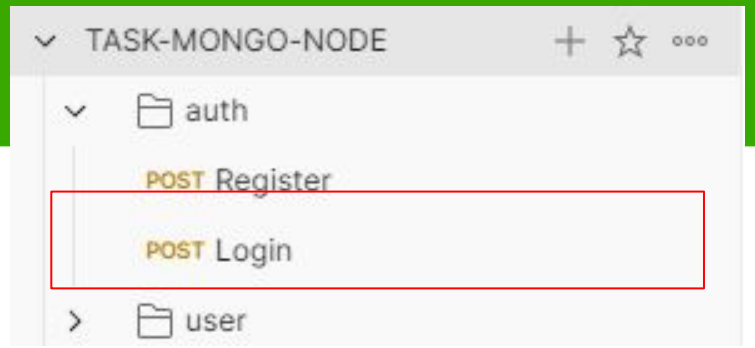


Pruebas

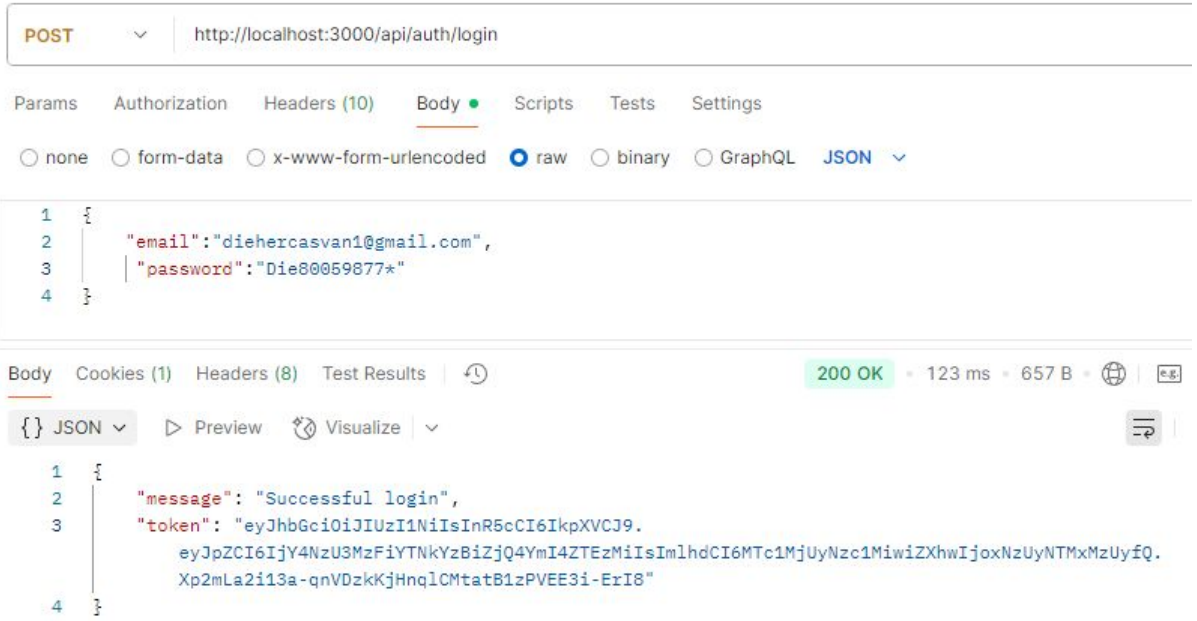


Método	Ruta	Controlador	Modelo
POST	http://localhost:3000/api/auth/register	Auth	User

Proyecto Paso a Paso



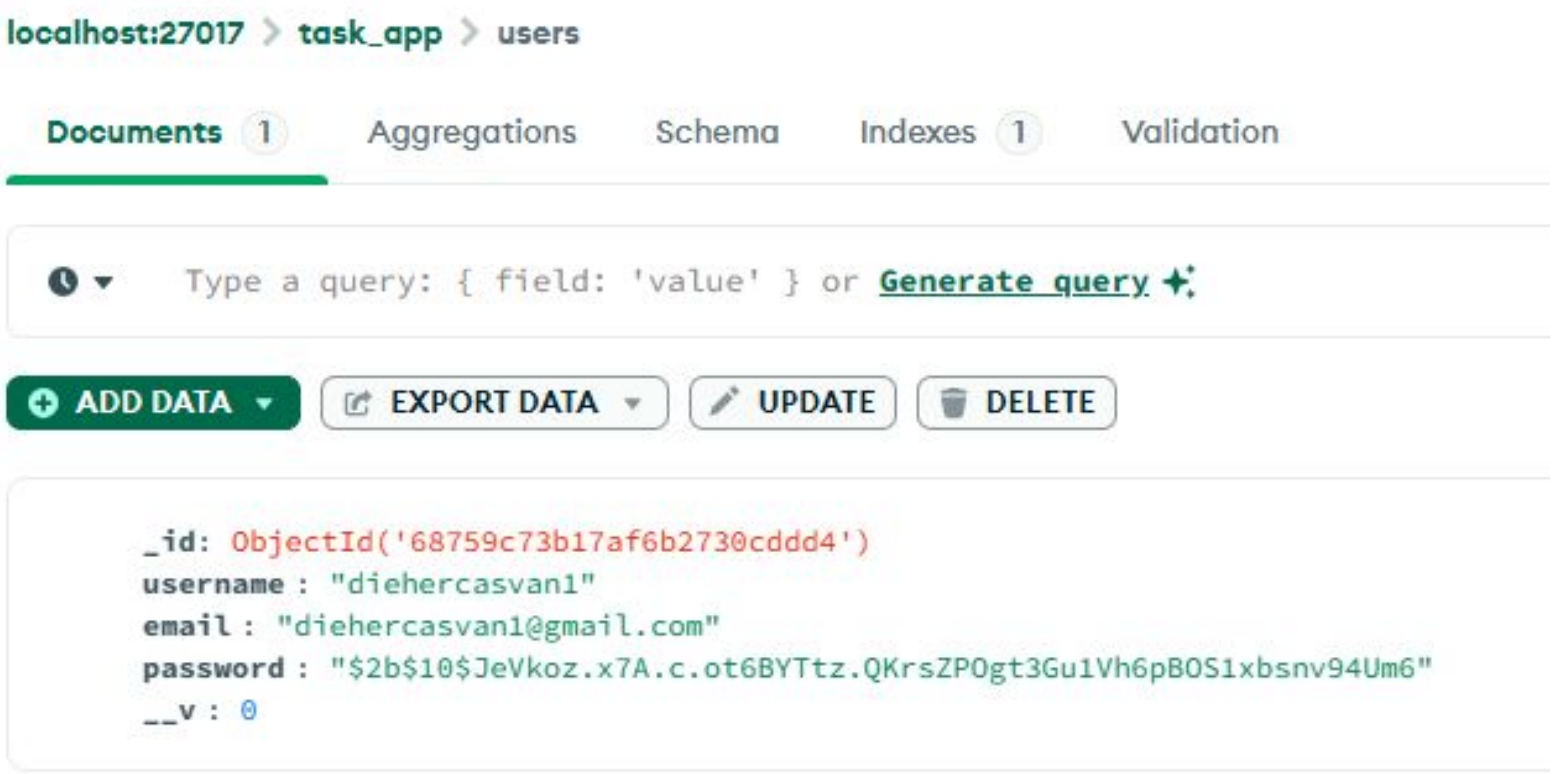
Pruebas



Método	Ruta	Controlador	Modelo
POST	http://localhost:3000/api/auth/login	Auth	User

Proyecto Paso a Paso

Pruebas



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