



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

ING - DIEGO CASALLAS

22810019-1



www.sena.edu.co

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

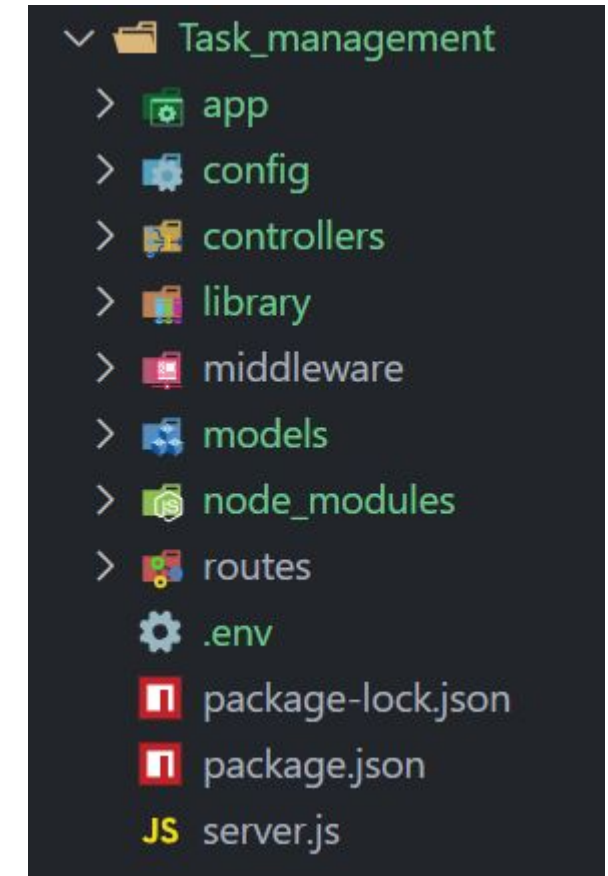
- mkdir "Nombre del proyecto"
- cd todo-app
- npm init -y ("Digitar por consola")
- npm install express mongoose body-parser dotenv ejs bcrypt jsonwebtoken cookie-parser nodemon nodemailer multer

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

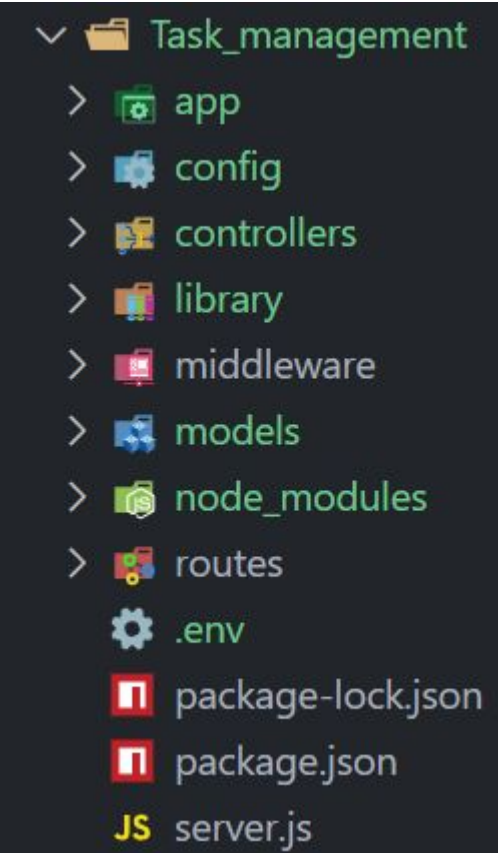


Proyecto Paso a Paso



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

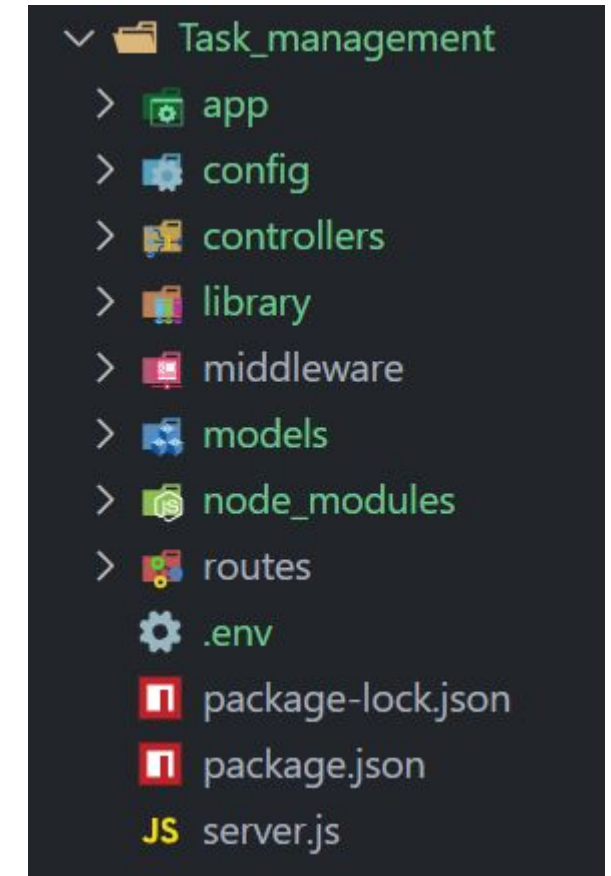


Proyecto Paso a Paso



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

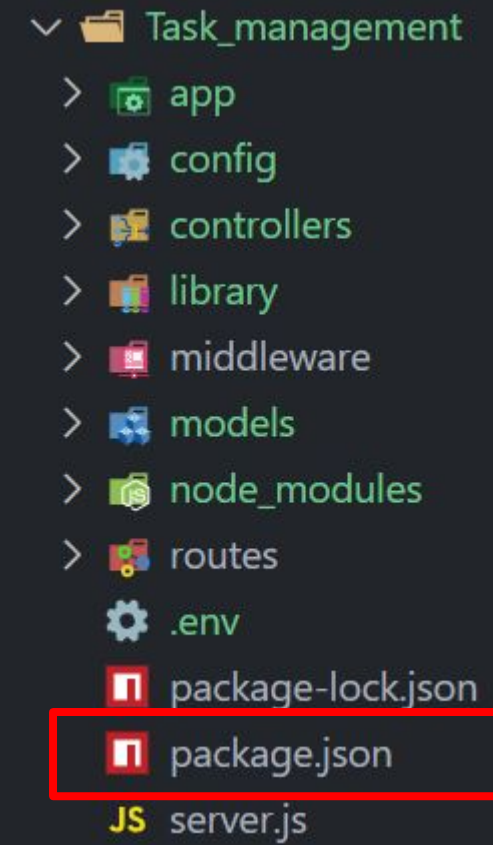


Proyecto Paso a Paso



- **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"  
},
```

A file explorer view of a project directory. The root folder is 'Task_management'. It contains several subfolders: 'app', 'config', 'controllers', 'library', 'middleware', 'models', 'node_modules', and 'routes'. There are also files: '.env', 'package-lock.json', 'package.json' (highlighted with a red box), and 'server.js'.

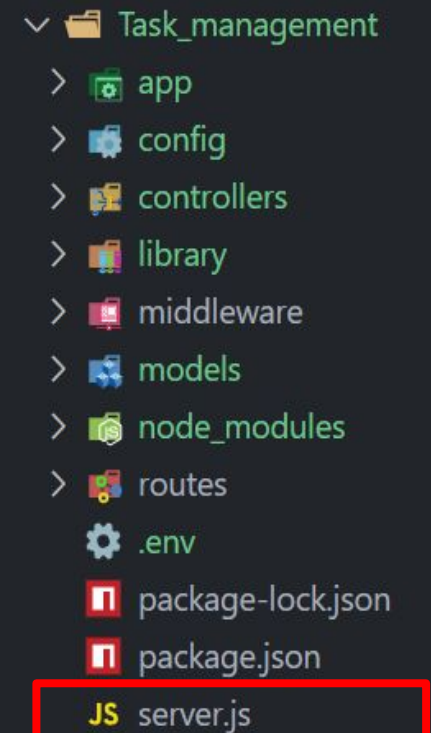
```
Task_management  
├── app  
├── config  
├── controllers  
├── library  
├── middleware  
├── models  
├── node_modules  
├── routes  
├── .env  
├── package-lock.json  
├── package.json  
└── server.js
```

Proyecto Paso a Paso



1.3. Configurar server.js

```
Task_management > JS server.js > ...
1  /**
2   * Author:Diego Casallas
3   * Date: 14/07/2025
4   * Description: This is the main server file for the backend of the application.
5   */
6
7  import app from './app/app.js';
8  import dotenv from 'dotenv';
9
10  dotenv.config();
11  |
12  const PORT = process.env.SERVER_PORT || 3000;
13
14  app.listen(PORT, () => {
15    console.log(`Server is running on http://localhost:${PORT}`);
16  });
```



Proyecto Paso a Paso

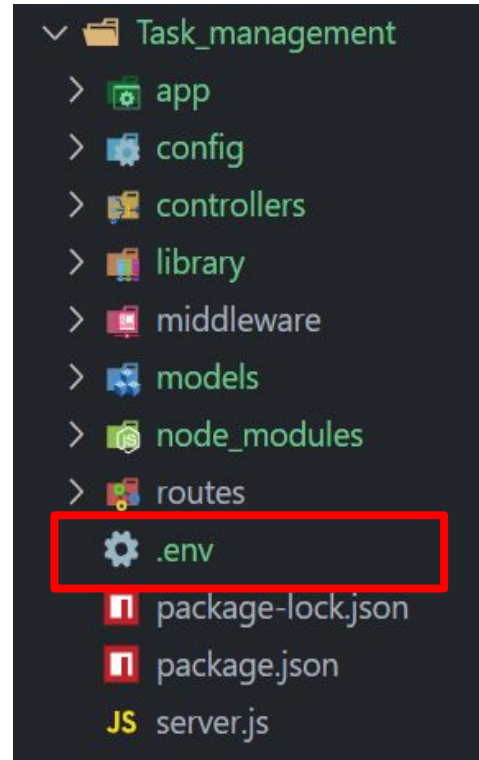


1.4. Crear .env

- **env**

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

```
Task_management > ⚙ .env
  Import to Postman
1  SERVER_PORT='3000'
2  JWT_SECRET="jwt_secret_key"
3  MONGODB_URI=mongodb://localhost:27017/task_app
```

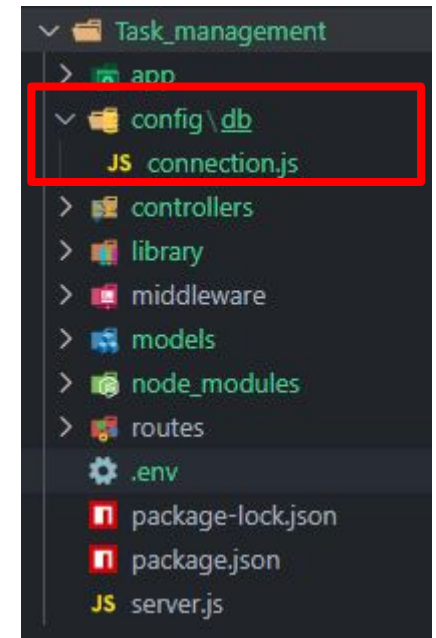


Proyecto Paso a Paso



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

```
Task_management > config > db > JS connection.js > ...
1  import mongoose from 'mongoose';
2  import dotenv from 'dotenv';
3
4  dotenv.config();
5  const MONGODB_URI = process.env.MONGODB_URI || 'mongodb://127.0.0.1:27017/task_app';
6  // Function to connect to MongoDB
7  export const connectDB = async () => {
8    try {
9      await mongoose.connect(MONGODB_URI);
10     console.log('✅ Connected to MongoDB');
11   } catch (error) {
12     console.error('❌ Error connecting to MongoDB:', error.message);
13     process.exit(1); // Exit the process with error
14   }
15 };
16 // Connection events
17 mongoose.connection.on('connected', () => {
18   //console.log(' Connected to MongoDB');
19 });
20 mongoose.connection.on('error', (err) => {
21   console.log('Error connecting to MongoDB:', err);
22 });
23 mongoose.connection.on('disconnected', () => {
24   console.log('Mongoose offline');
25 });
26 // Export the connection and mongoose in case it is needed
27 export { mongoose };
```

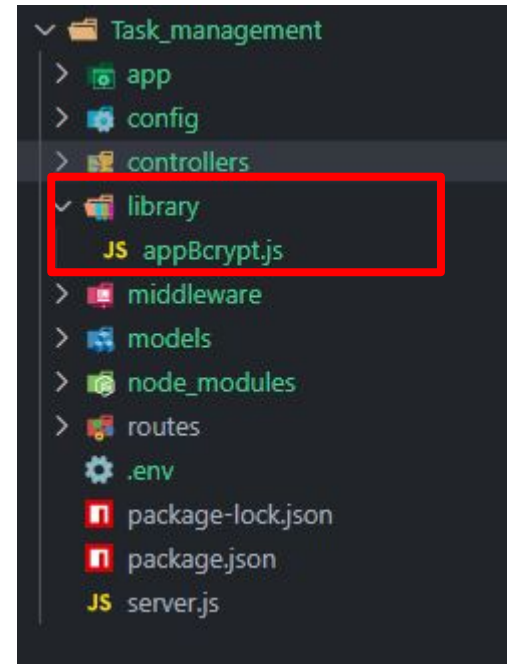


Proyecto Paso a Paso



- **library -> appBcrypt.js**

```
Task_management > library > JS appBcrypt.js > ...
1  import appBcrypt from 'bcrypt';
2  const saltRounds = 10;
3
4  export const encryptPassword = async (password) => {
5    try {
6      const hashedPassword = await appBcrypt.hash(password, saltRounds);
7      return hashedPassword;
8    } catch (error) {
9      console.error('Error encrypt:', error);
10     throw error;
11   }
12 };
13
14 export const comparePassword = async (password, hashedPassword) => {
15   try {
16     const match = await appBcrypt.compare(password, hashedPassword);
17     return match;
18   } catch (error) {
19     console.error('Error compare the hash:', error);
20     throw error;
21   }
22 };
```

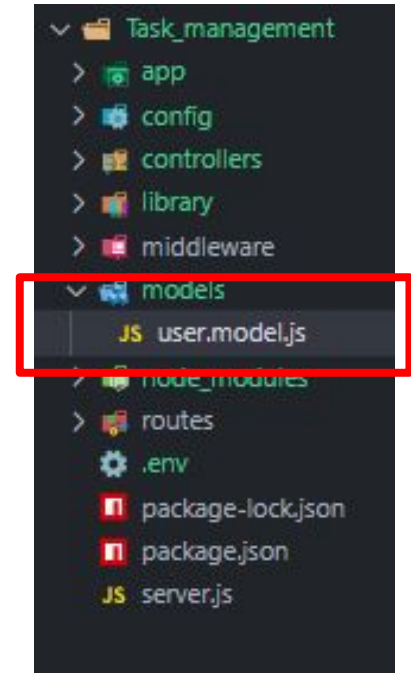


Proyecto Paso a Paso



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

```
Task_management > models > JS user.model.js > userSchema.pre('save') callback
1  import { mongoose } from '../config/db/connection.js'; // Import mongoose from the connection
2  import { encryptPassword } from '../library/appBcrypt.js';
3
4  const userSchema = new mongoose.Schema({
5    username: { type: String, required: true, unique: true },
6    email: { type: String, required: true, unique: true },
7    password: { type: String, required: true },
8  });
9
10 // Middleware for password hashing
11 userSchema.pre('save', async function (next) {
12   if (!this.isModified('password')) return next();
13
14   try {
15     this.password = await encryptPassword(this.password);
16     next();
17   } catch (error) {
18     next(error);
19   }
20 });
21
22 export default mongoose.model('User', userSchema);
```

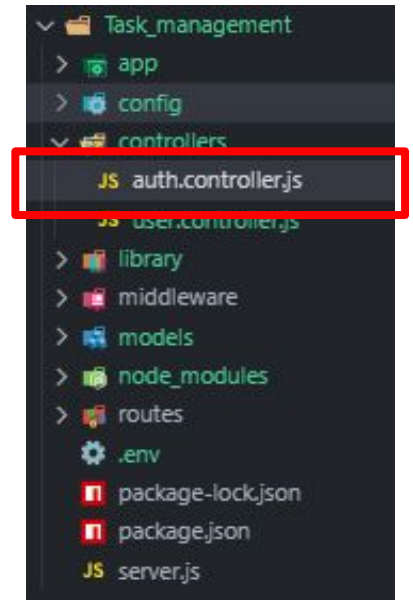


Proyecto Paso a Paso



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

```
Task_management > controllers > JS auth.controller.js > AuthController > register
1
2 import UserModel from '../models/user.model.js'; // Import the user model
3 import jwt from 'jsonwebtoken';
4 import { comparePassword } from '../library/appBcrypt.js';
5
6 import dotenv from 'dotenv';
7 dotenv.config();
8
9 class AuthController {
10   //User registration
11   async register(req, res) {
12     try {
13       const { username, email, password } = req.body;
14       if (!username || !email || !password) {
15         return res.status(400).json({ error: 'All fields are required' });
16       }
17       if (password.length < 6) {
18         return res.status(400).json({ error: 'The password must be at least 6 characters long.' });
19       }
20       const existingUserModel = await UserModel.findOne({ email });
21       if (existingUserModel) {
22         return res.status(400).json({ error: 'The user already exists' });
23       }
24       const newUserModel = new UserModel({ username, email, password });
25       await newUserModel.save();
26       return res.status(201).json({ message: 'User successfully registered' });
27     } catch (err) {
28       res.status(400).json({ error: err.message });
29     }
30   }
31 }
```

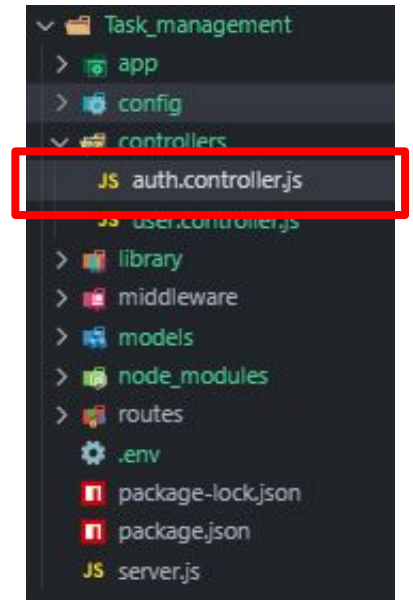


Proyecto Paso a Paso



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

```
33 // Login
34 async login(req, res) {
35   try {
36     const { email, password } = req.body;
37     const userModel = await UserModel.findOne({ email });
38     if (!userModel) throw new Error('User not found');
39
40     const isMatch = await comparePassword(password, userModel.password);
41     if (!isMatch) throw new Error('Incorrect password');
42
43     const token = jwt.sign({ id: userModel._id }, process.env.JWT_SECRET, { expiresIn: '1h' });
44     res.cookie('token', token, { httpOnly: true });
45     res.status(200).json({ message: 'Successful login', token: token });
46   } catch (err) {
47     res.status(400).json({ error: err.message });
48   }
49 };
50
51 }
52
53 export default new AuthController();
```

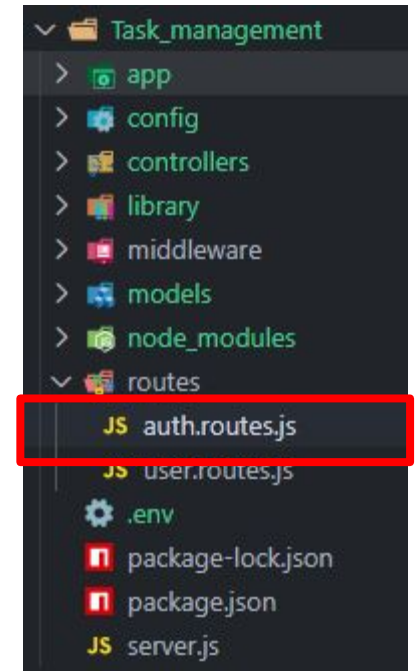


Proyecto Paso a Paso



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

```
Task_management > routes > JS auth.routes.js > [e] 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;
```

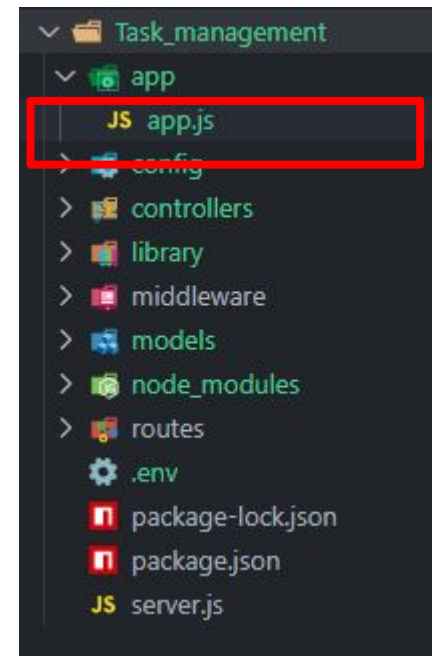


Proyecto Paso a Paso



- **app-> app.js**

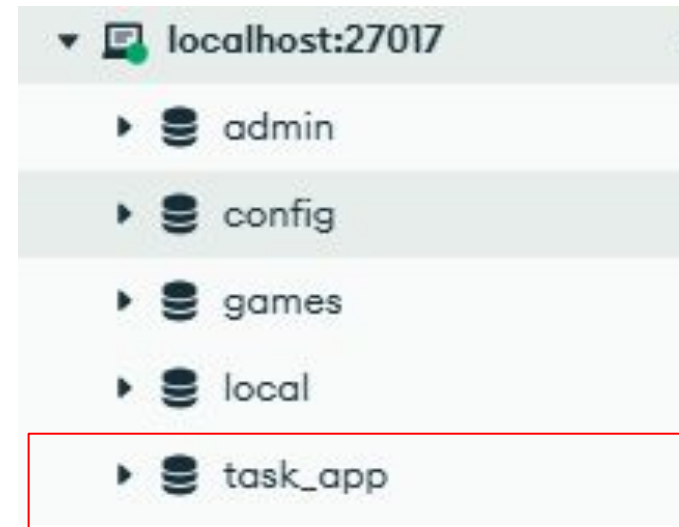
```
Task_management > app > JS app.js > ...
1  import express from 'express';
2  import { connectDB } from '../config/db/connection.js'; // Import the connection
3  // Import routes
4  import authRouter from '../routes/auth.routes.js';
5
6  const app = express();
7  app.use(express.json());
8  app.use(express.urlencoded({ extended: true }));
9
10 // Connect to MongoDB
11 connectDB();// Call the connection function
12
13 // Use routes
14 app.use('/api', authRouter);
15
16 app.use((req, res, nex) => {
17   res.status(404).json({
18     message: 'Endpoint losses'
19   });
20 });
21
22 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**

```
Server is running on http://localhost:3000  
Connected to MongoDB  
✓ Connected to MongoDB
```

Proyecto Paso a Paso

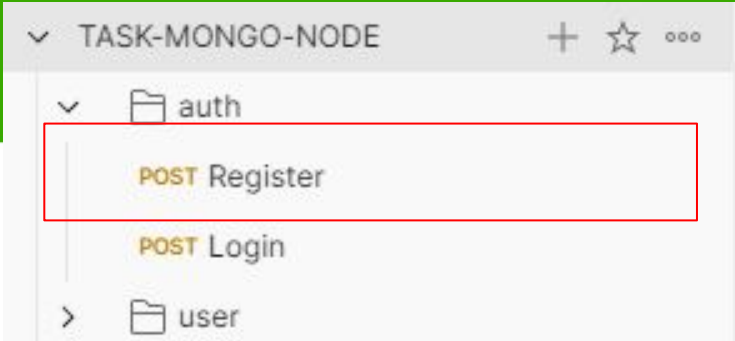


Pruebas

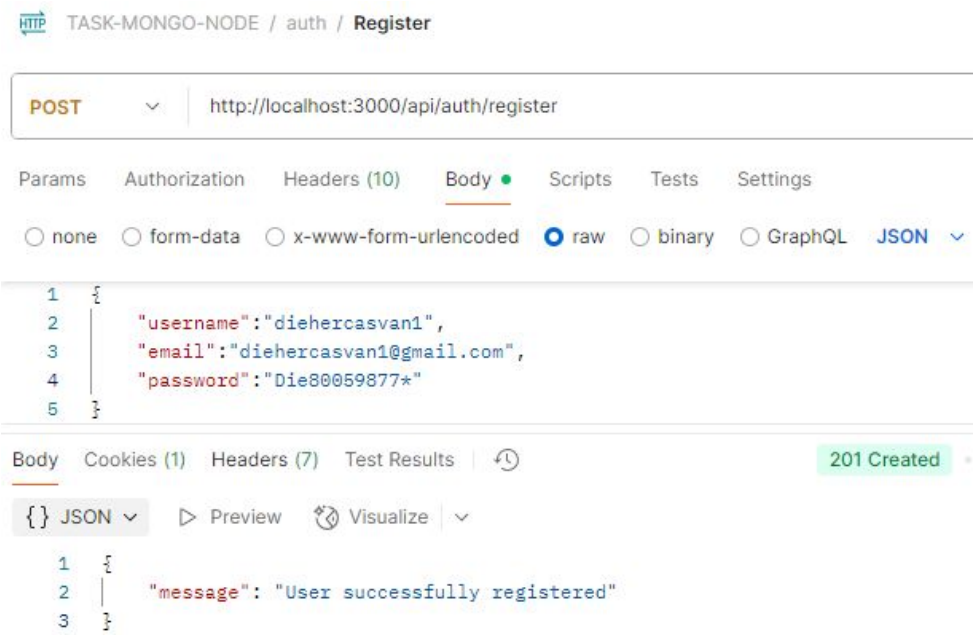
- **Ejecutar la aplicación postman**
 - **Crear una colección**



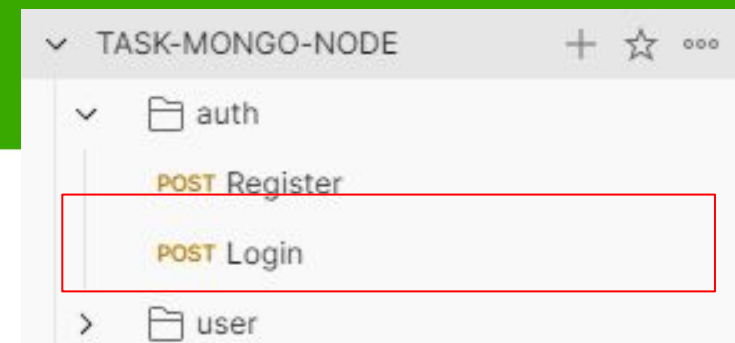
Proyecto Paso a Paso



Pruebas



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



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

Proyecto Paso a Paso



Pruebas

localhost:27017 > task_app > users

Documents 1

Aggregations

Schema

Indexes 1

Validation



Type a query: { field: 'value' } or [Generate query](#) ✚

+ ADD DATA ▾

EXPORT DATA ▾

UPDATE

DELETE

```
_id: ObjectId('68759c73b17af6b2730cddd4')
username : "diehercasvan1"
email : "diehercasvan1@gmail.com"
password : "$2b$10$JeVkoz.x7A.c.ot6BYTtz.QKrsZP0gt3Gu1Vh6pBOS1xbsnv94Um6"
__v : 0
```




GRACIAS

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



www.sena.edu.co