SENA

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

22810019-1

Proyecto Paso a Paso

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

1. Configuración Inicial (2 horas)

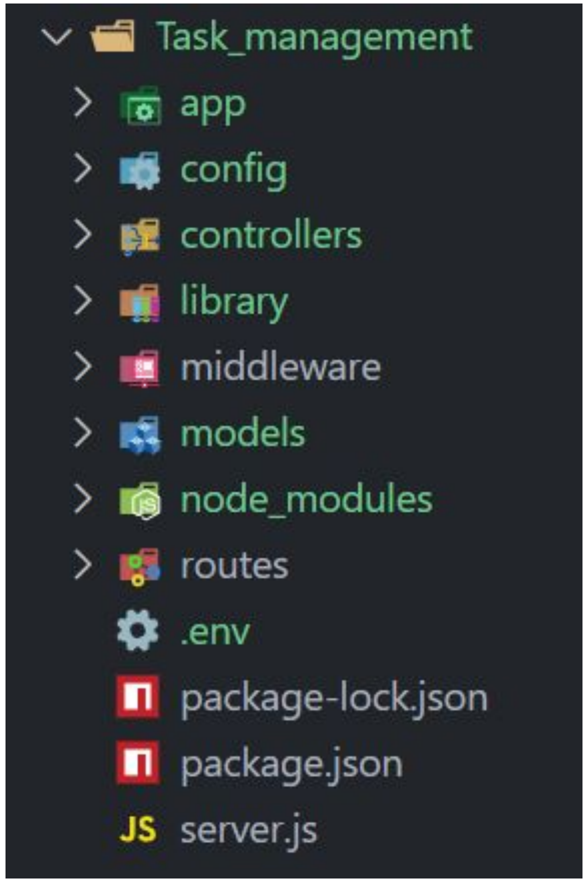
1.1. Crear el proyecto e instalar dependencias

- mkdir todo-app
- cd todo-app
- npm init -y
- npm install express mongoose body-parser dotenv ejs bcrypt jsonwebtoken cookie-parser nodemon

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

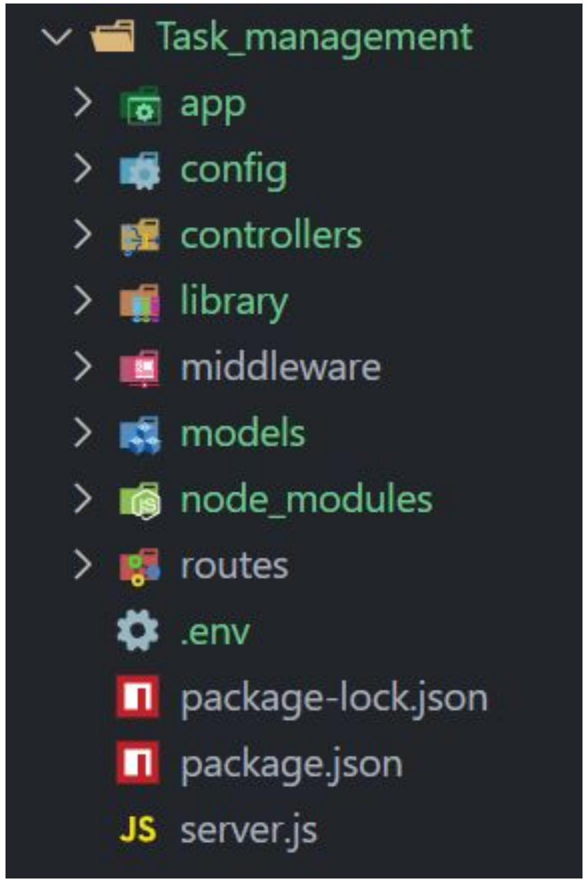


```
Task_management
  app
  config
  controllers
  library
  middleware
  models
  node_modules
  routes
  .env
  package-lock.json
  package.json
  server.js
```

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

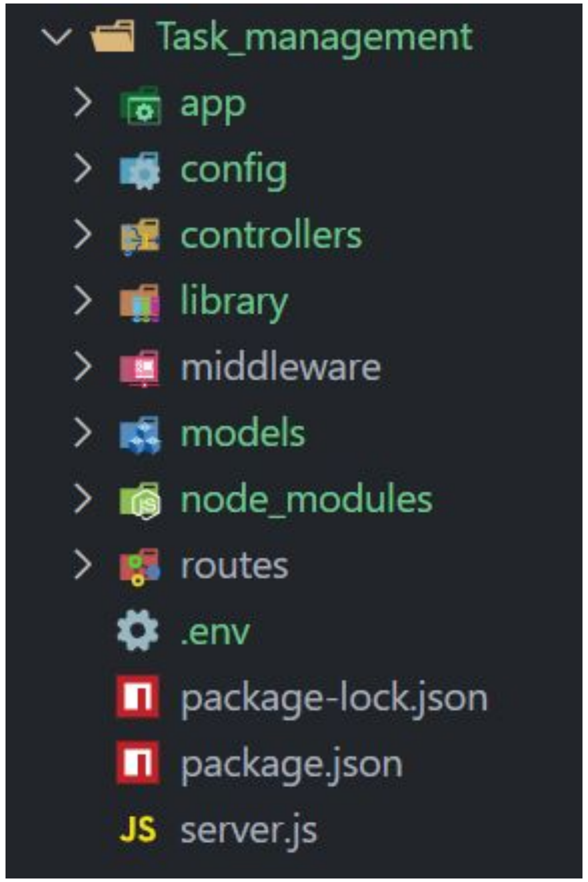


```
Task_management
  > app
  > config
  > controllers
  > library
  > middleware
  > models
  > node_modules
  > routes
    .env
    package-lock.json
    package.json
    server.js
```

Proyecto Paso a Paso

1.2. Estructura básica del proyecto

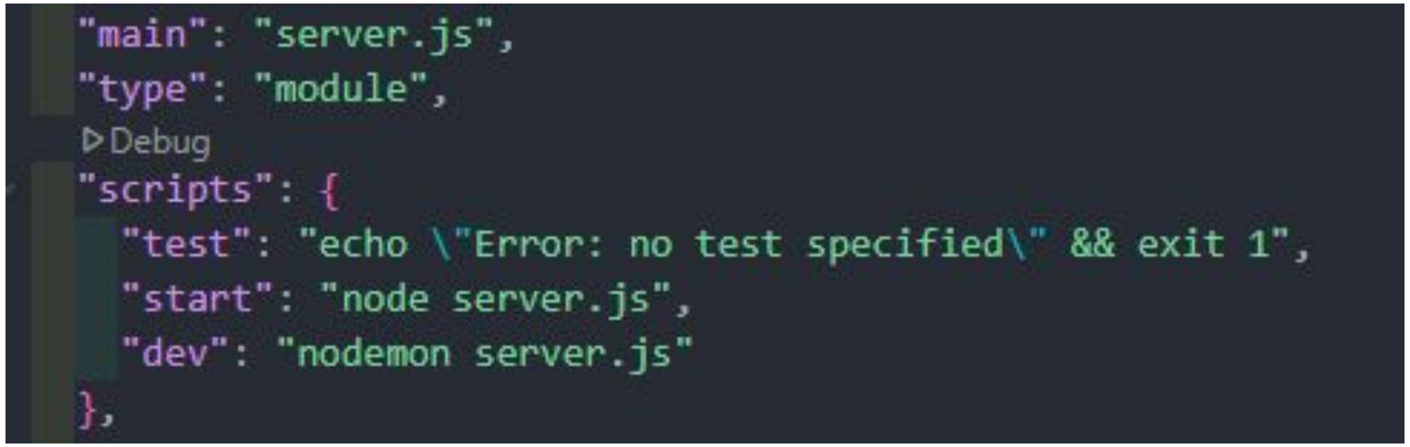
- **node_modules(componentes de npm)**
  - Archivos de dependencias
- **routes (manejo de rutas amigables)**
  - Rutas de la aplicación
- **env**
  - Variables del entorno
- **package.json**
  - Archivos de dependencias del proyecto



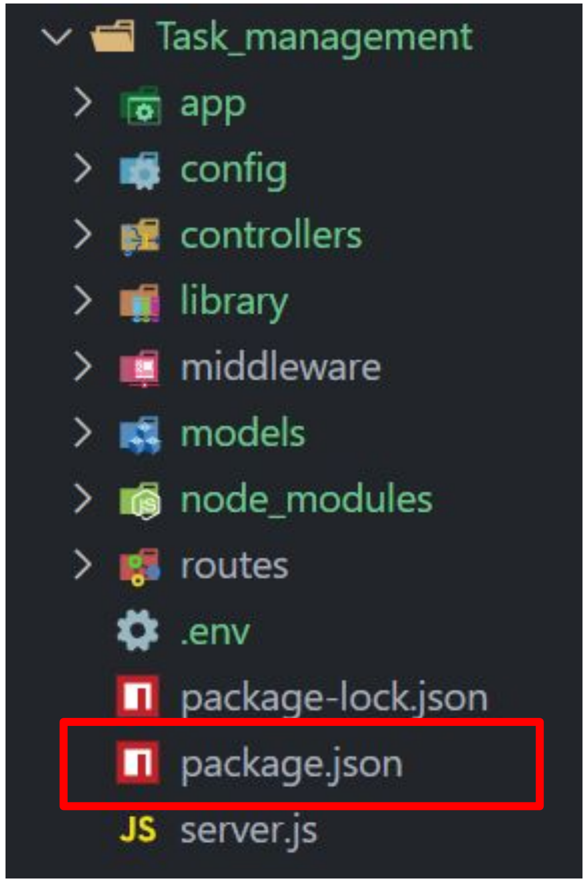
```
Task_management
  > app
  > config
  > controllers
  > library
  > middleware
  > models
  > node_modules
  > routes
    .env
    package-lock.json
    package.json
    server.js
```

Proyecto Paso a Paso

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



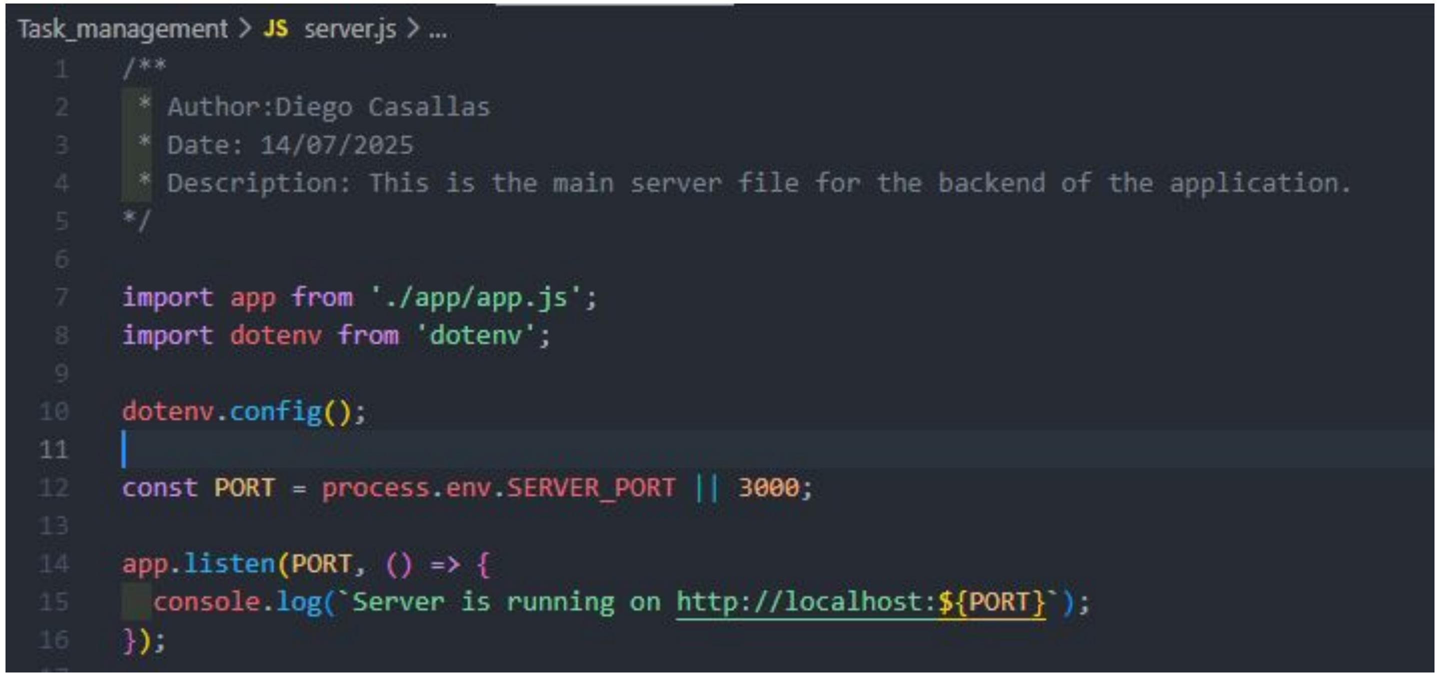
```
"main": "server.js",
"type": "module",
▷ Debug
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "node server.js",
  "dev": "nodemon server.js"
},
```



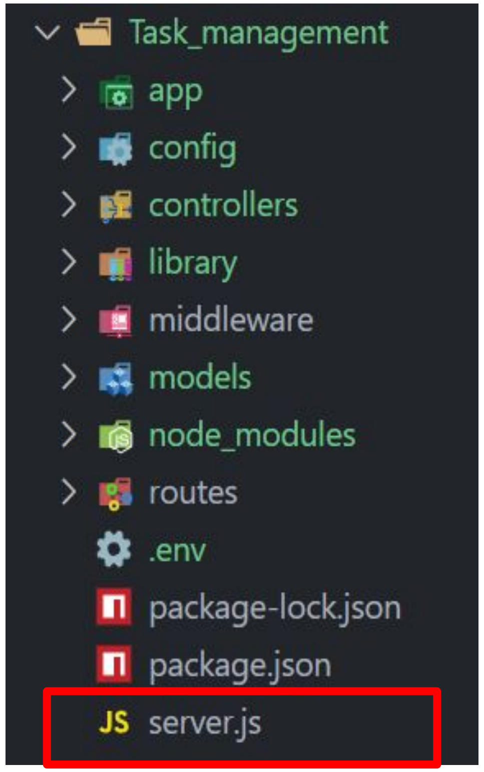
```
Task_management
  app
  config
  controllers
  library
  middleware
  models
  node_modules
  routes
  .env
  package-lock.json
  package.json
  server.js
```

Proyecto Paso a Paso

1.3. Configurar server.js



```
Task_management > JS server.js > ...
/**
 * 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}`);
});
```



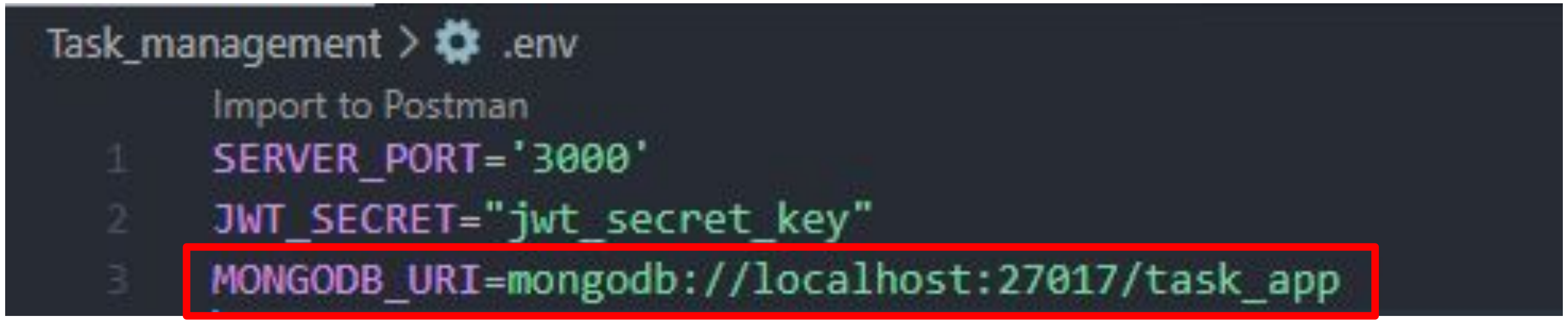
```
Task_management
  app
  config
  controllers
  library
  middleware
  models
  node_modules
  routes
  .env
  package-lock.json
  package.json
  server.js
```

Proyecto Paso a Paso

1.4. Crear .env

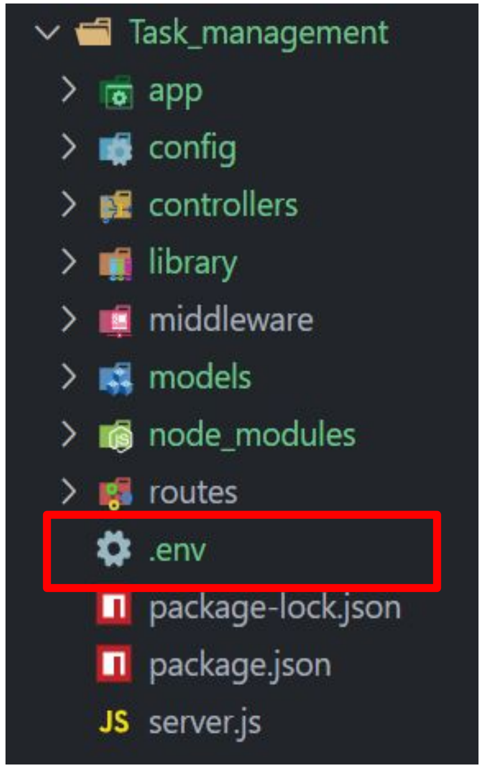
- **env**

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



Task_management > .env

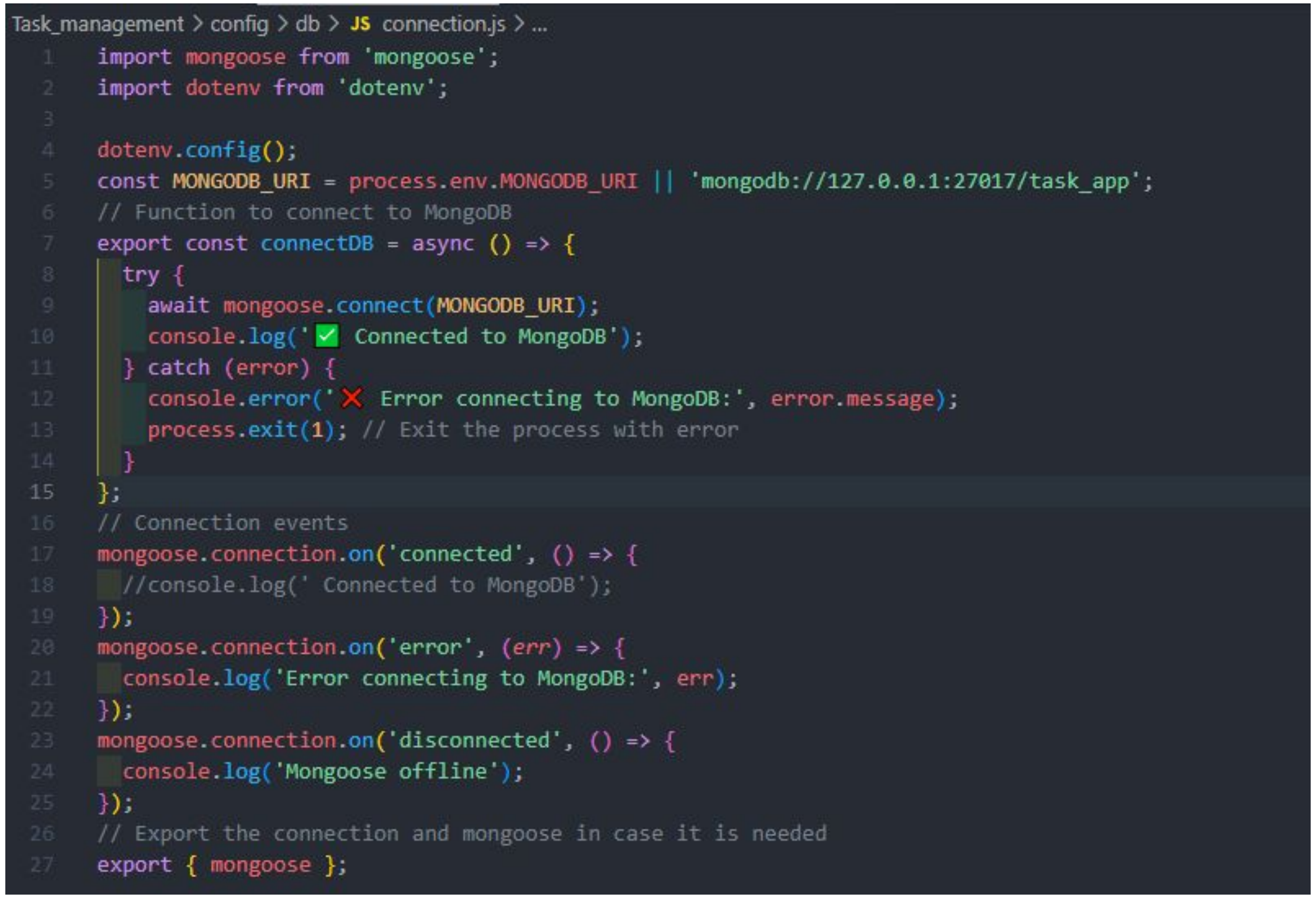
```
Import to Postman
SERVER_PORT='3000'
JWT_SECRET="jwt_secret_key"
MONGODB_URI=mongodb://localhost:27017/task_app
```



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

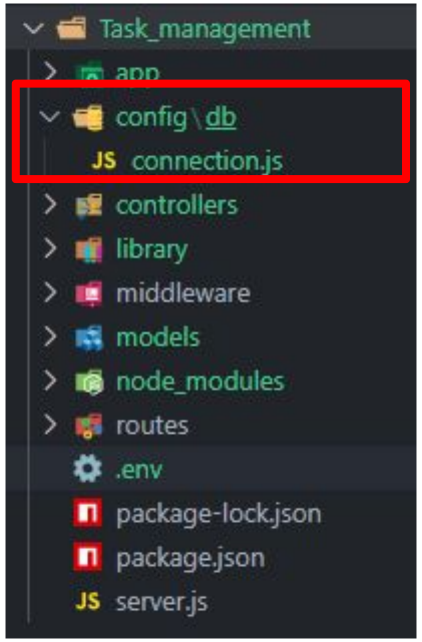
Proyecto Paso a Paso

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



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

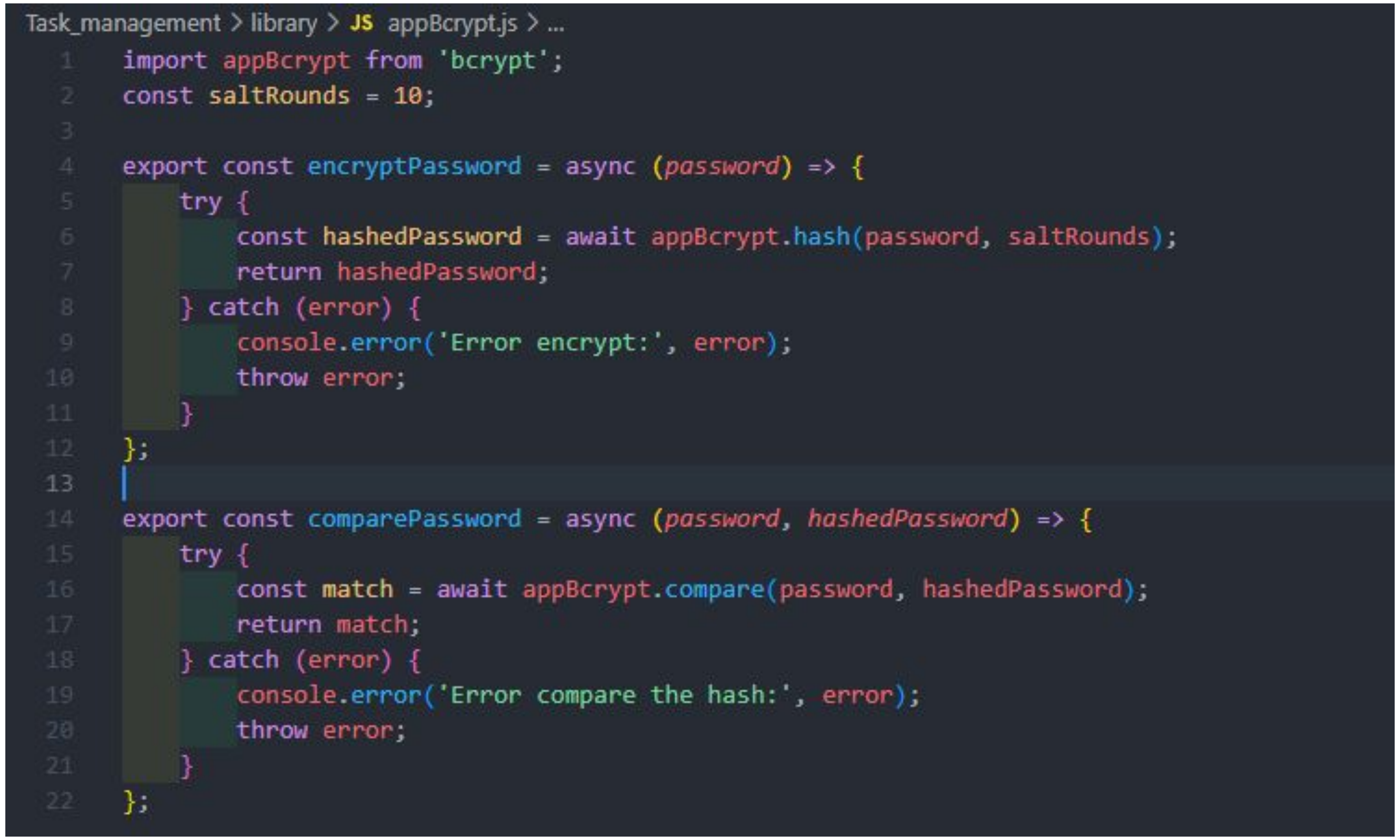
```javascript
import mongoose from 'mongoose';
import dotenv from 'dotenv';

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 };
```



```
Task_management
  app
  config\db
    connection.js
  controllers
  library
  middleware
  models
  node_modules
  routes
  .env
  package-lock.json
  package.json
  server.js
```

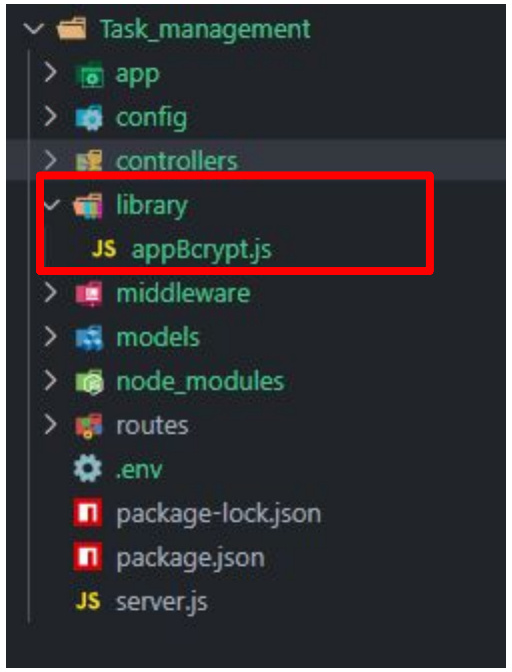
Proyecto Paso a Paso

- **library -> appBcrypt.js**



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

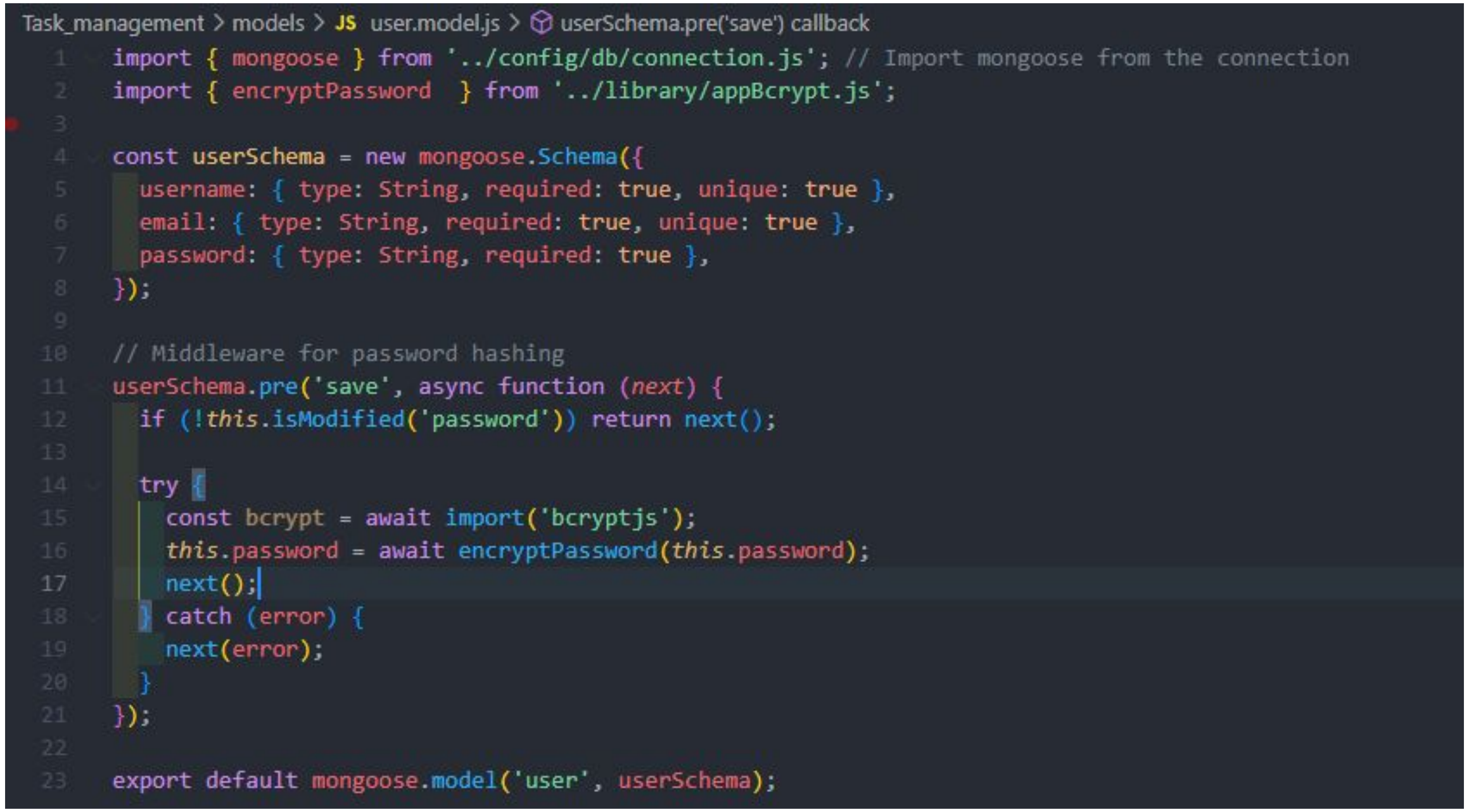
```js
import appBcrypt from 'bcrypt';
const saltRounds = 10;

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;
    }
};
```



```
Task_management
  > app
  > config
  > controllers
  v library
      appBcrypt.js
  > middleware
  > models
  > node_modules
  > routes
    .env
    package-lock.json
    package.json
    server.js
```

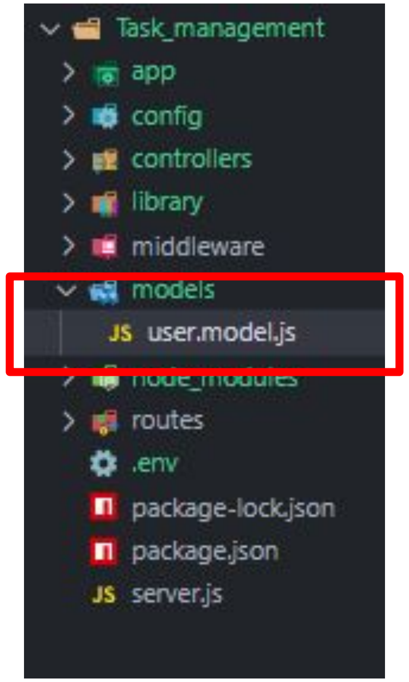
Proyecto Paso a Paso

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



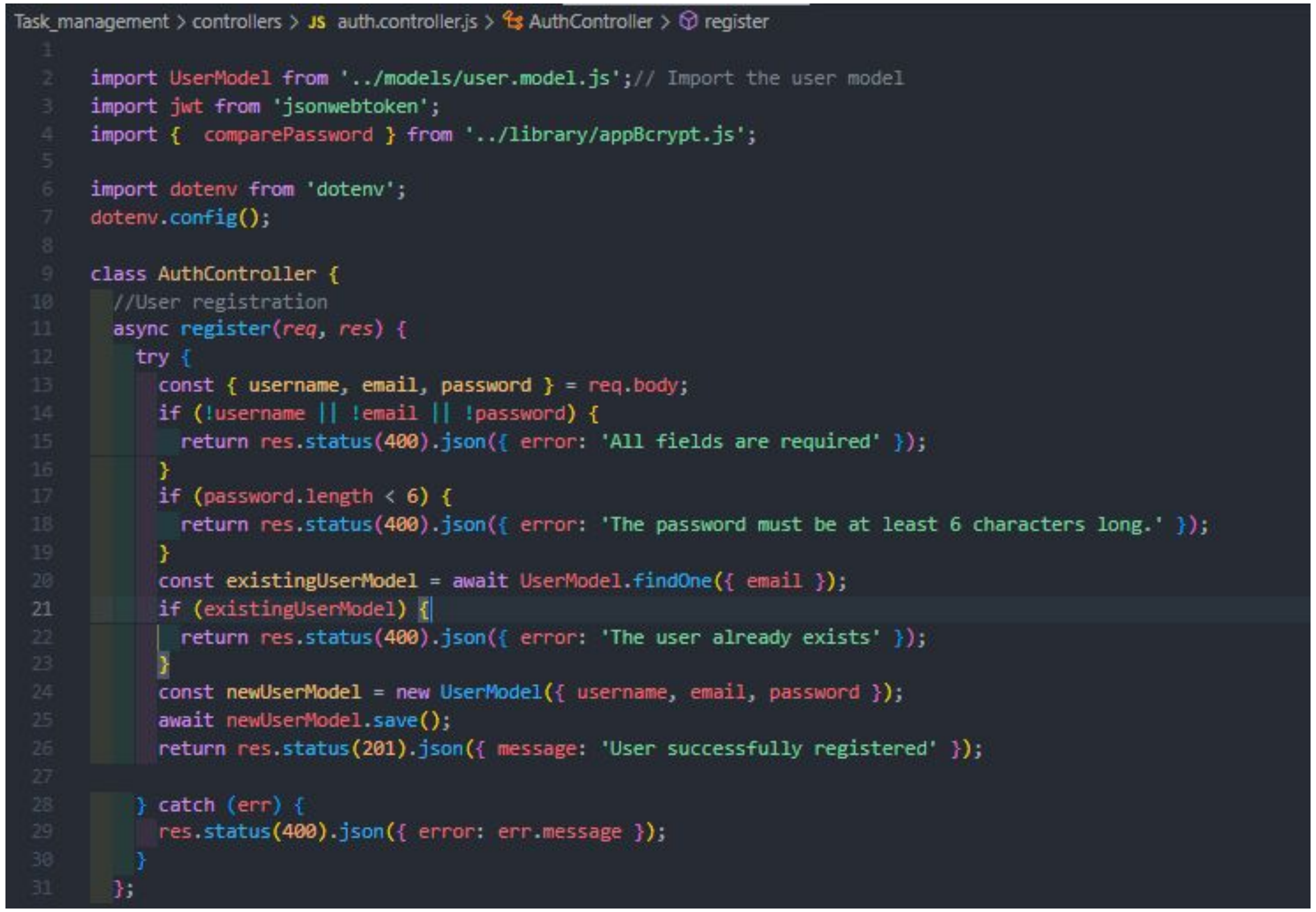
Task_management > models > JS user.model.js > userSchema.pre('save') callback

```javascript
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 {
    const bcrypt = await import('bcryptjs');
    this.password = await encryptPassword(this.password);
    next();
  } catch (error) {
    next(error);
  }
});

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



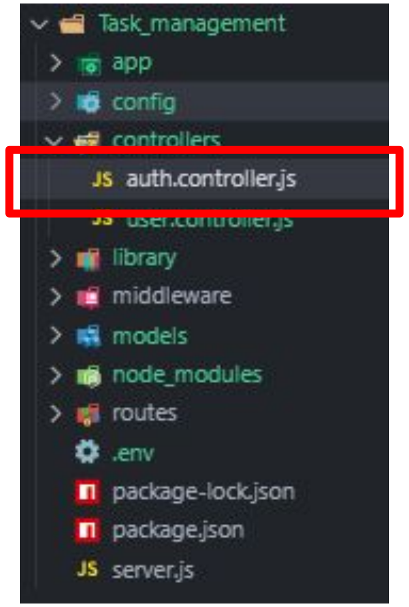
Proyecto Paso a Paso

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



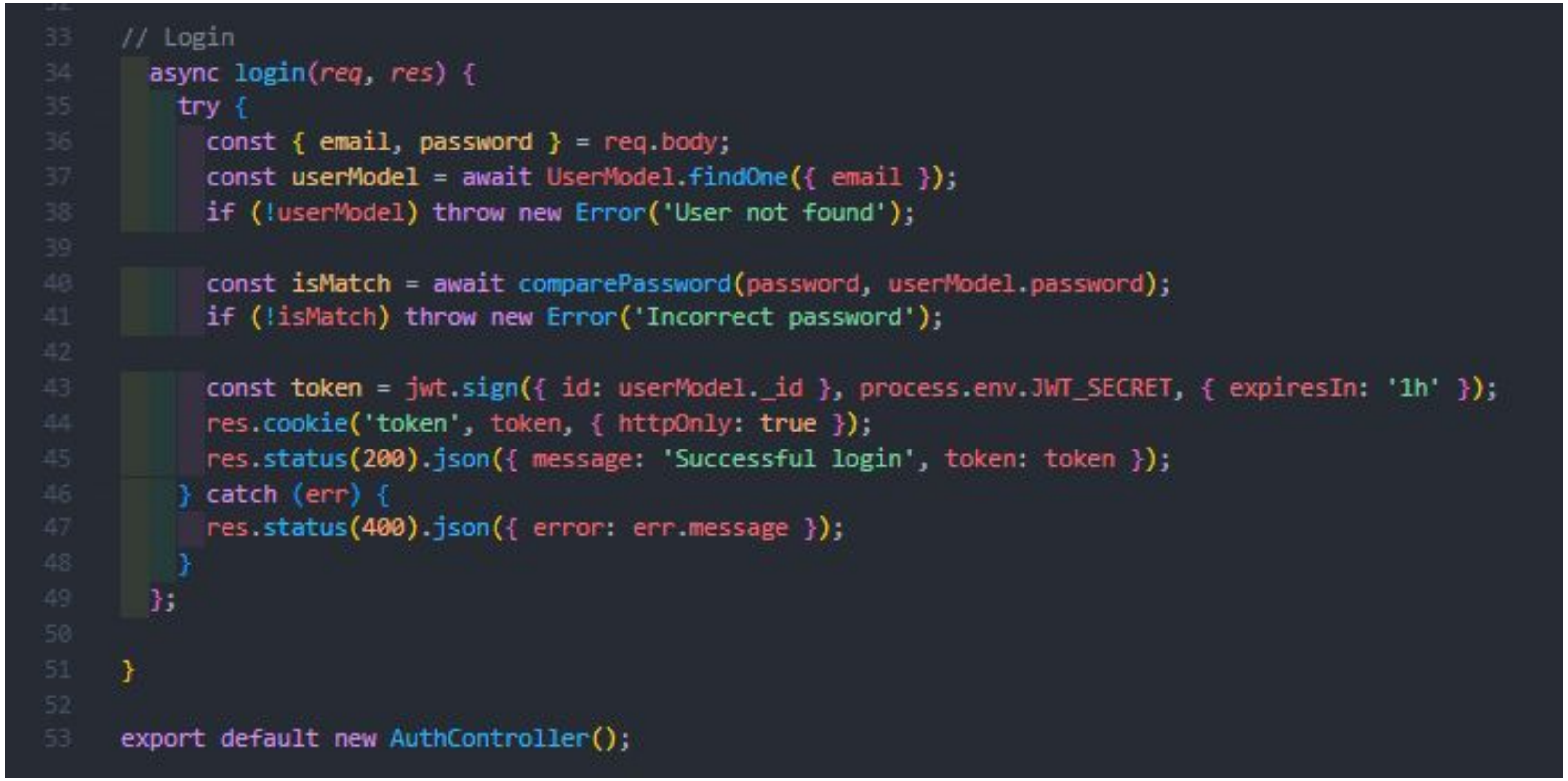
Task_management > controllers > JS auth.controller.js > AuthController > register

```
import UserModel from '../models/user.model.js';// Import the user model
import jwt from 'jsonwebtoken';
import {  comparePassword } from '../library/appBcrypt.js';

import dotenv from 'dotenv';
dotenv.config();

class AuthController {
  //User registration
  async register(req, res) {
    try {
      const { username, email, password } = req.body;
      if (!username || !email || !password) {
        return res.status(400).json({ error: 'All fields are required' });
      }
      if (password.length < 6) {
        return res.status(400).json({ error: 'The password must be at least 6 characters long.' });
      }
      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 });
    }
  };
```

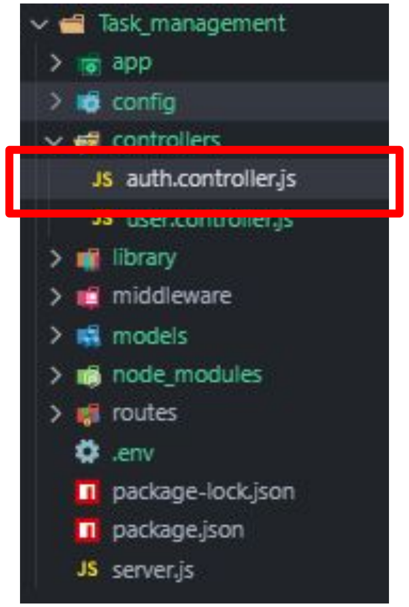


Proyecto Paso a Paso

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



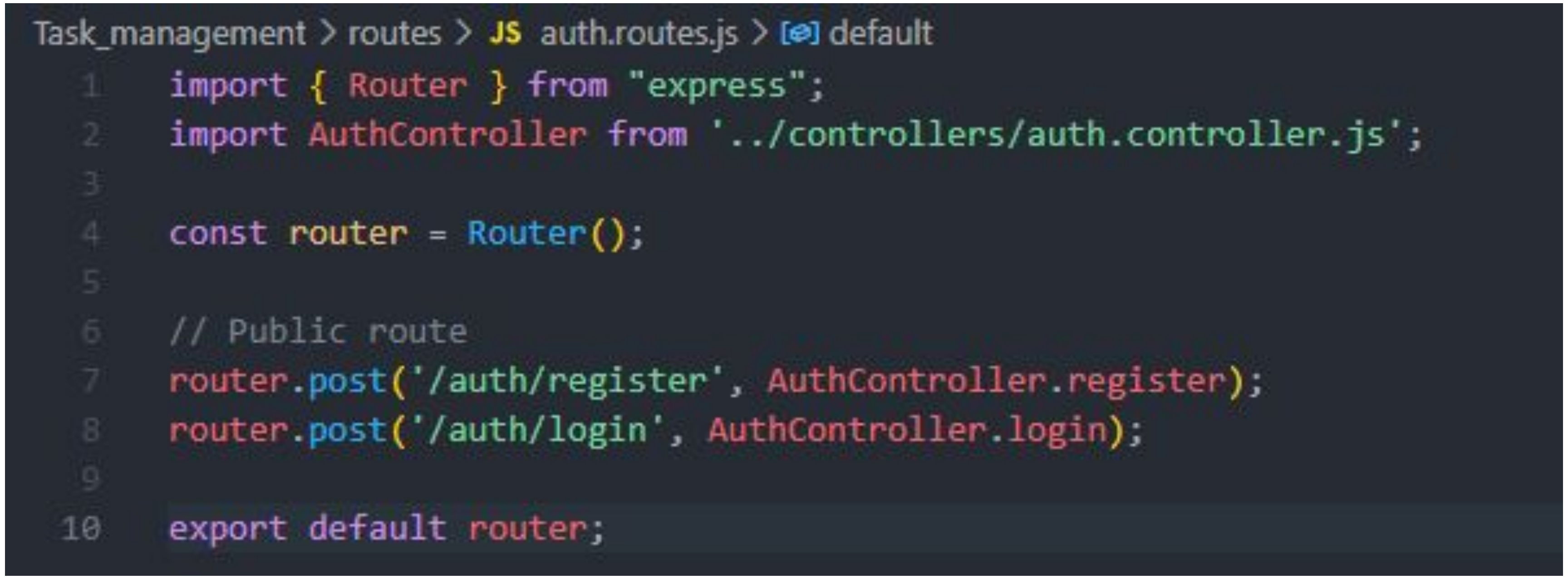
```
    // Login
    async login(req, res) {
        try {
            const { email, password } = req.body;
            const userModel = await UserModel.findOne({ email });
            if (!userModel) throw new Error('User not found');

            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();
```



```
Task_management
  app
  config
  controllers
    auth.controller.js
    user.controller.js
  library
  middleware
  models
  node_modules
  routes
  .env
  package-lock.json
  package.json
  server.js
```

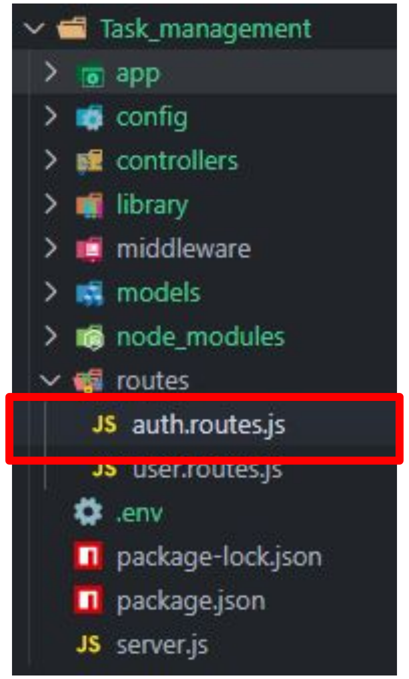
Proyecto Paso a Paso

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



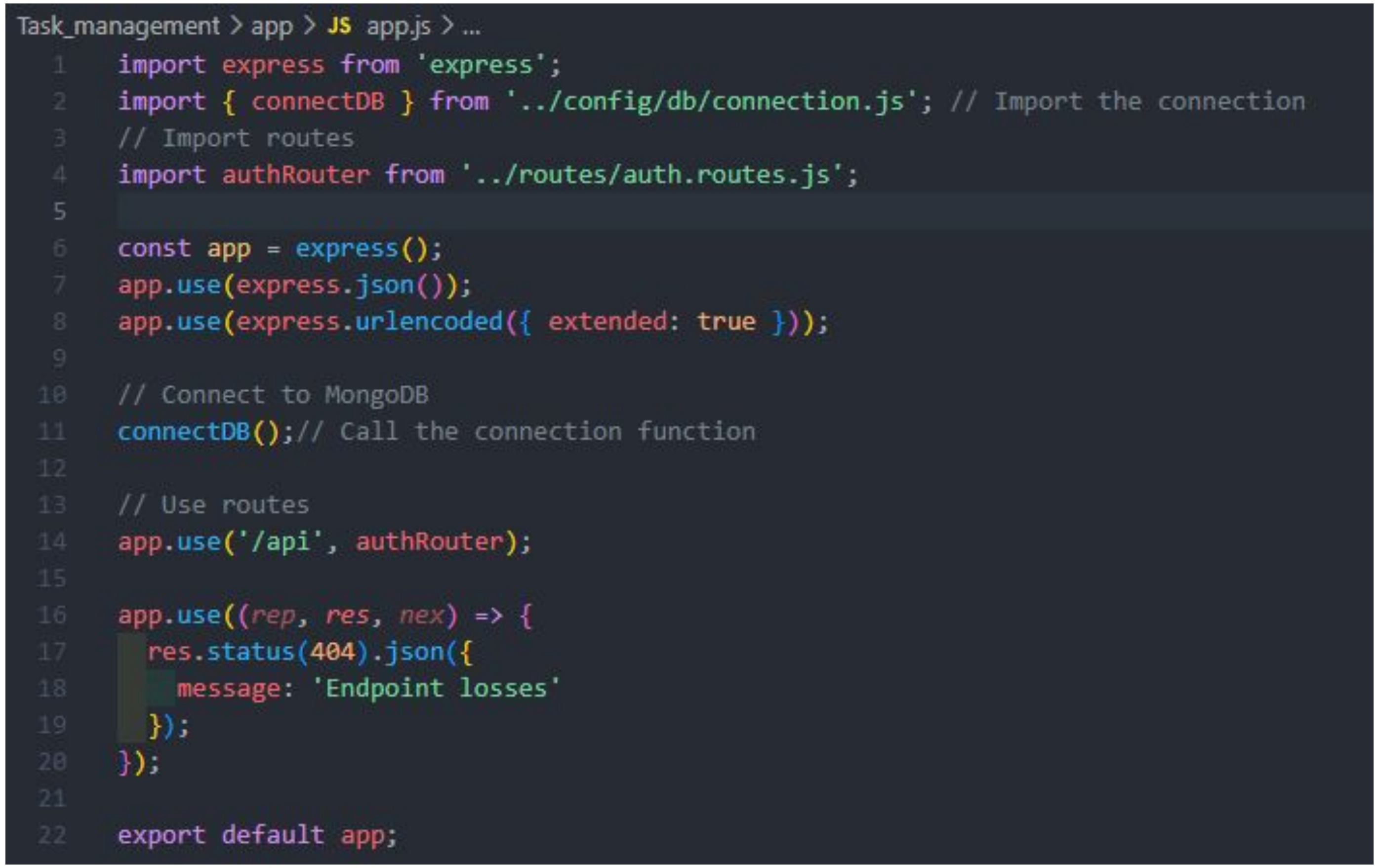
Task_management > routes > JS auth.routes.js > [e] default

```javascript
import { Router } from "express";
import AuthController from '../controllers/auth.controller.js';

const router = Router();

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

export default router;
```



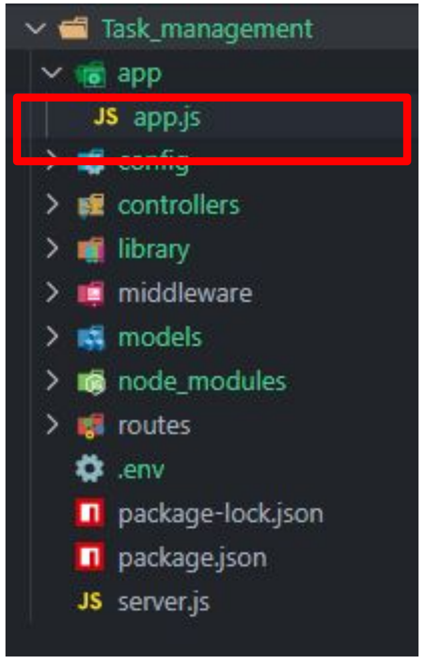
Proyecto Paso a Paso

- **app-> app.js**



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

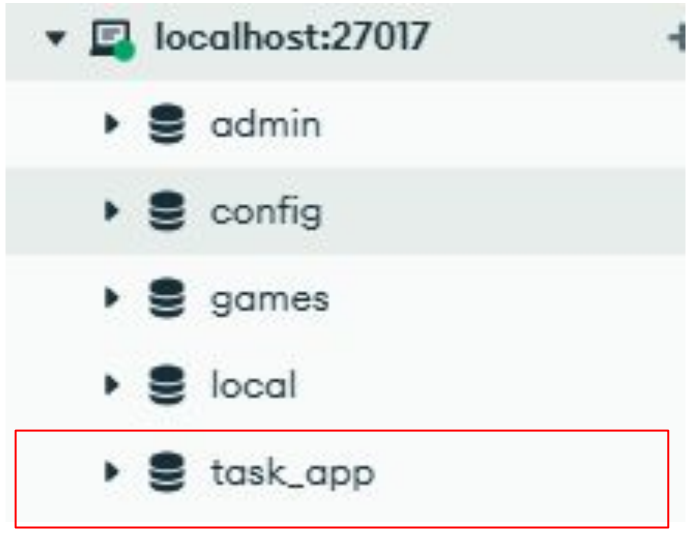
```javascript
import express from 'express';
import { connectDB } from '../config/db/connection.js'; // Import the connection
// Import routes
import authRouter from '../routes/auth.routes.js';

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;
```



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

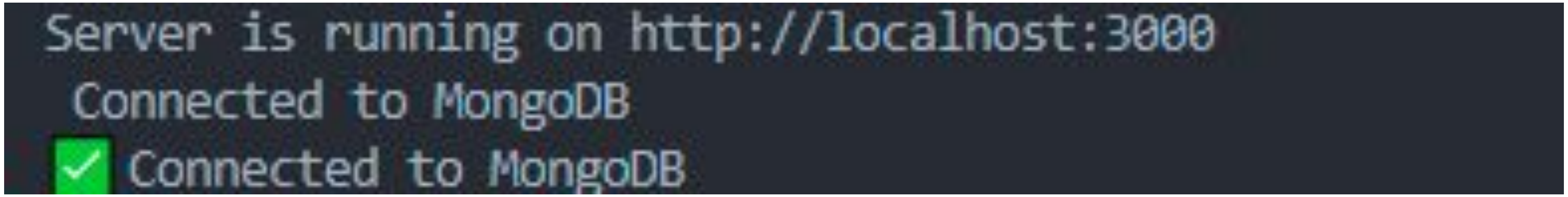
Proyecto Paso a Paso

- **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**

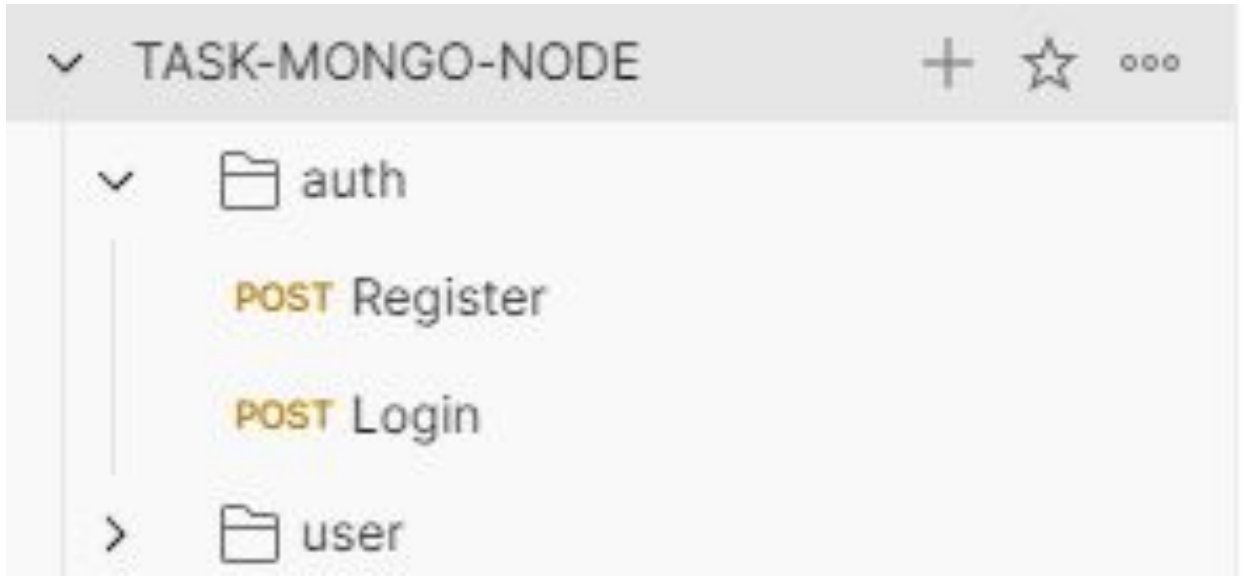


```
Server is running on http://localhost:3000
 Connected to MongoDB
✅ Connected to MongoDB
```

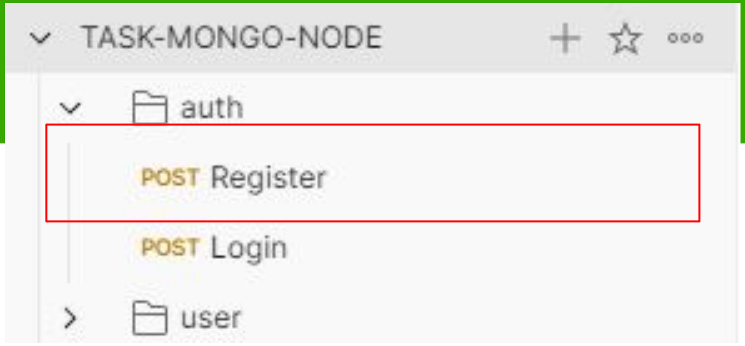
Proyecto Paso a Paso

Pruebas

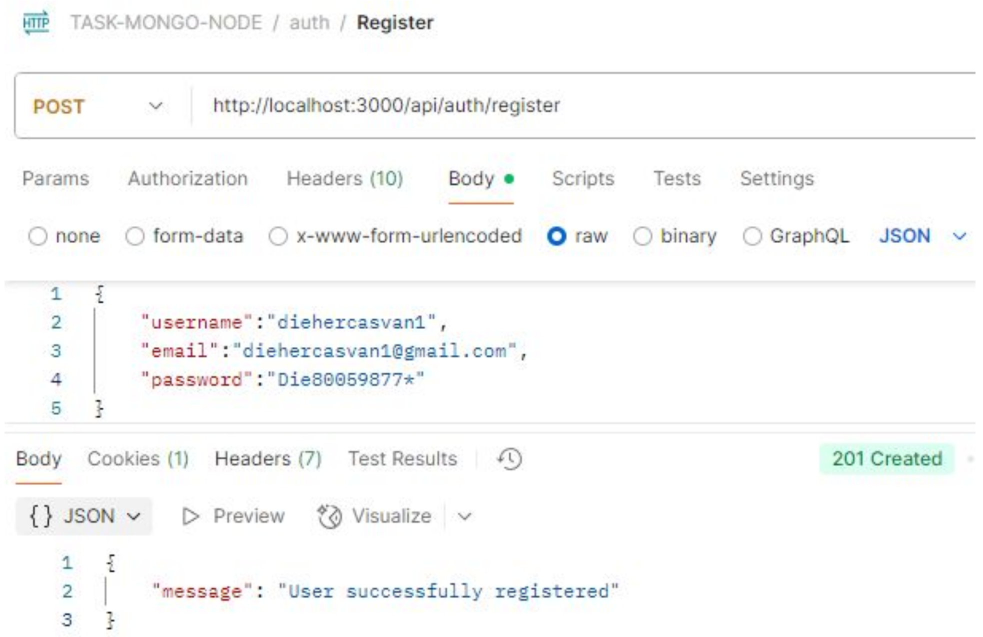
- **Ejecutar la aplicación postman**
  - **Crear una colección**



Proyecto Paso a Paso



Pruebas



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

Body: raw [✓] JSON

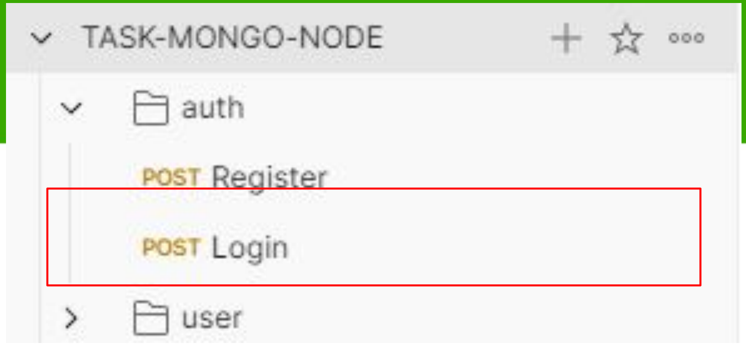
```
{
    "username":"diehercasvan1",
    "email":"diehercasvan1@gmail.com",
    "password":"Die80059877*"
}
```

201 Created

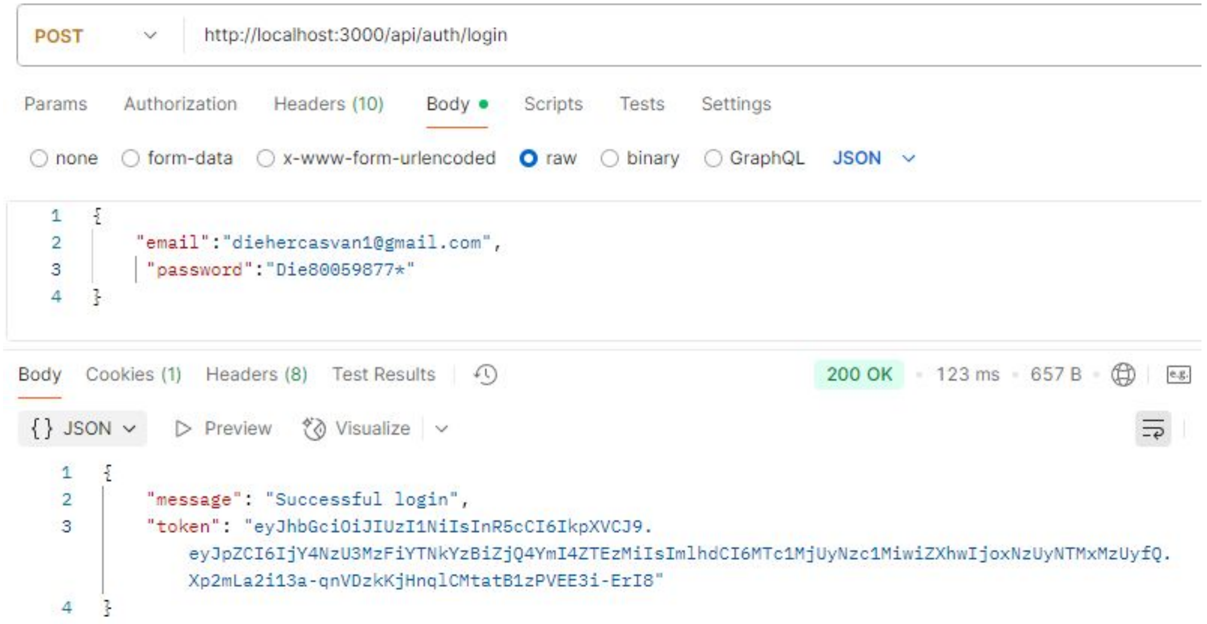
```
{
    "message": "User successfully registered"
}
```

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

Proyecto Paso a Paso



Pruebas



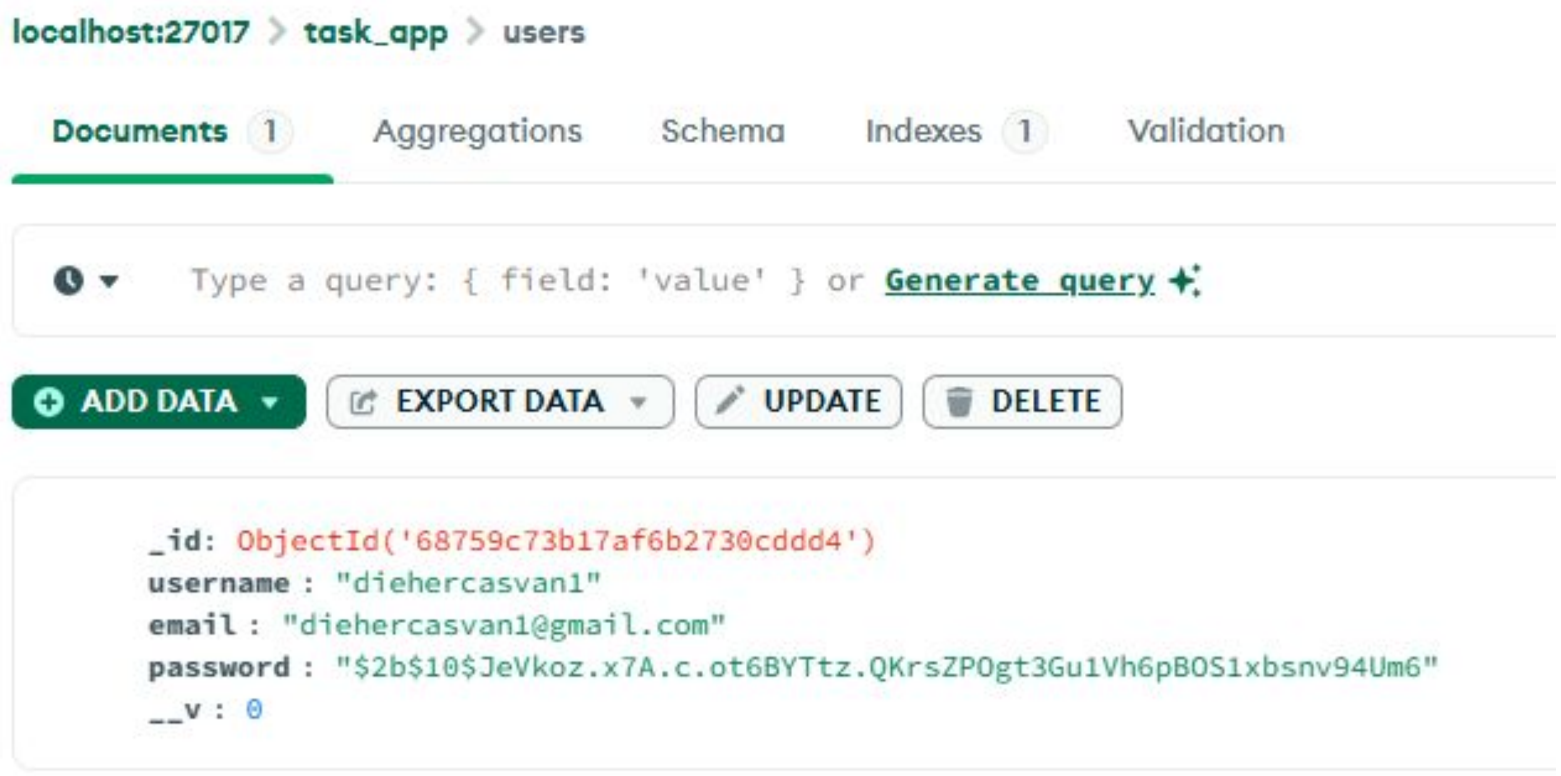
```json
{
    "email":"diehercasvan1@gmail.com",
    "password":"Die80059877*"
}
```

```json
{
    "message": "Successful login",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6IjY4NzU3MzFiYTNkYzBiZjQ4YmI4ZTEzMiIsImlhdCI6MTc1MjUyNzc1MiwiZXhwIjoxNzUyNTMxMzUyfQ.
        Xp2mLa2i13a-qnVDzkKjHnqlCMtatB1zPVEE3i-ErI8"
}
```

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.QKrsZPOgt3Gu1Vh6pBOS1xbsnv94Um6"
__v : 0
```

SENA

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