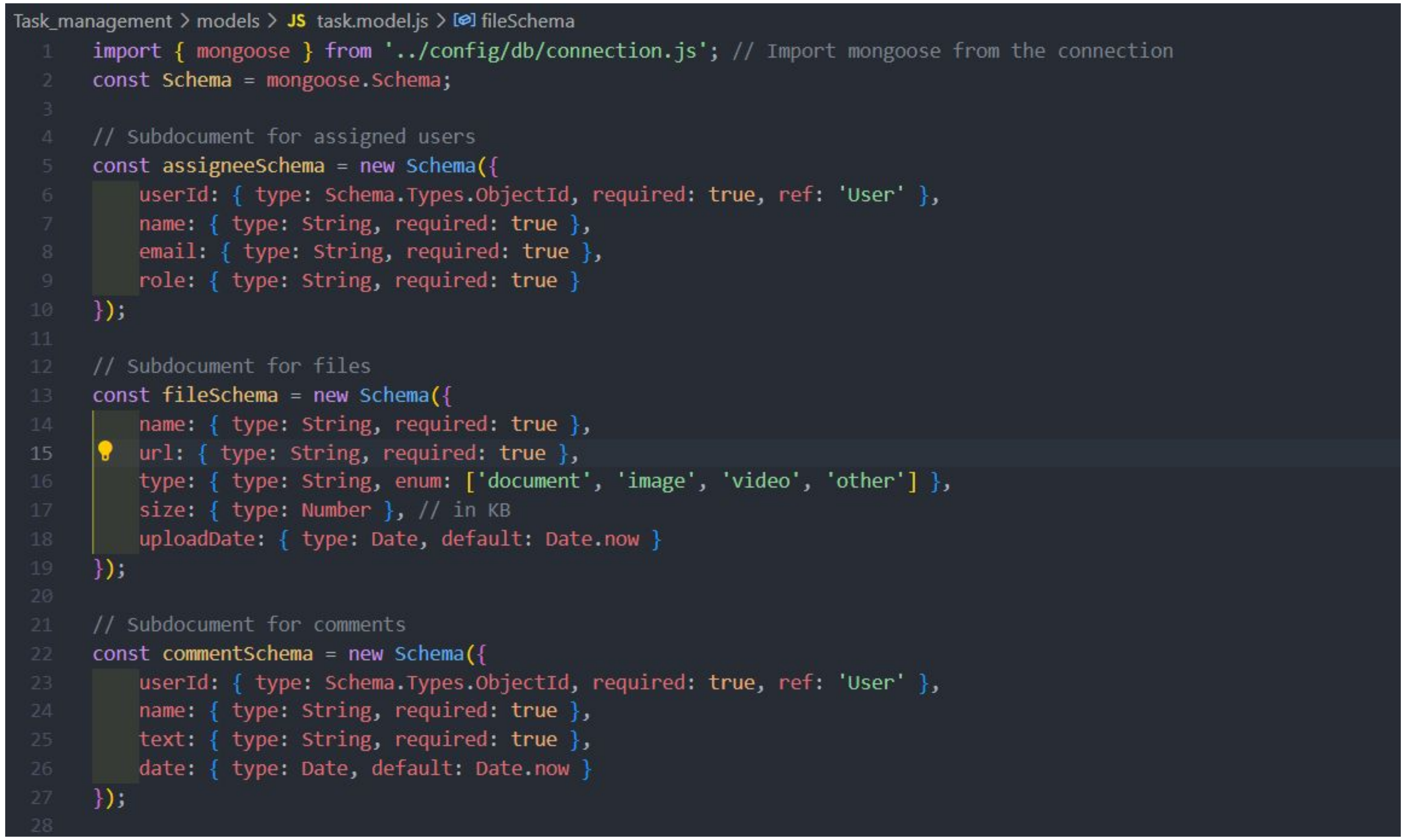
Curso complementario Desarrollo de Back-end con Node.js - MongoDB

ING - DIEGO CASALLAS

22810019-1

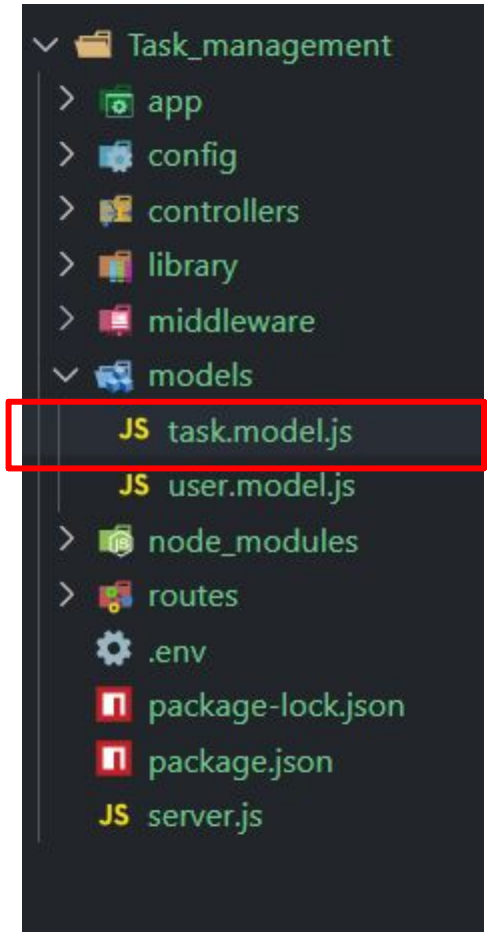
Mongodb

Crear Modelo de Tarea
models->task.model.js



Task_management > models > JS task.model.js > fileSchema

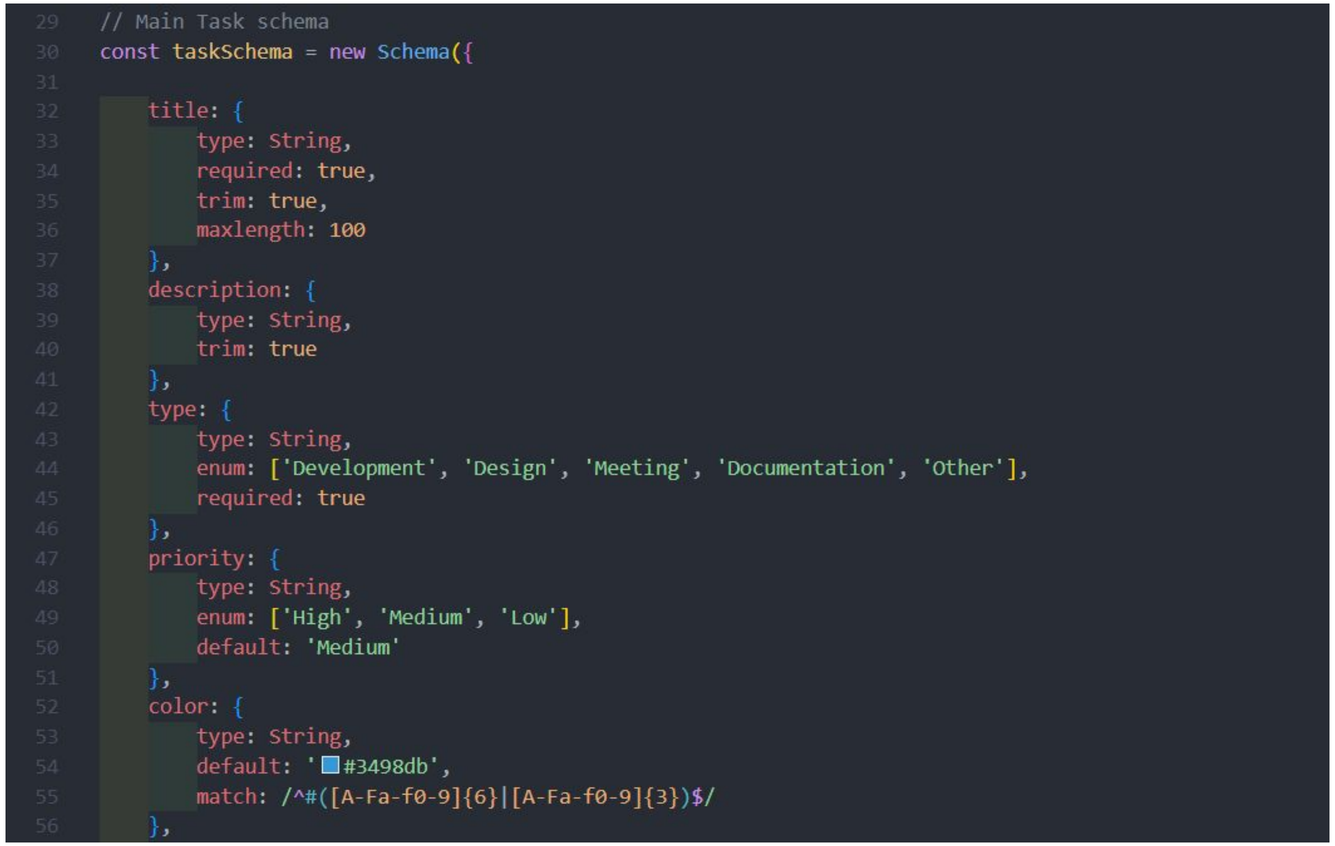
```javascript
import { mongoose } from '../config/db/connection.js'; // Import mongoose from the connection
const Schema = mongoose.Schema;

// Subdocument for assigned users
const assigneeSchema = new Schema({
    userId: { type: Schema.Types.ObjectId, required: true, ref: 'User' },
    name: { type: String, required: true },
    email: { type: String, required: true },
    role: { type: String, required: true }
});

// Subdocument for files
const fileSchema = new Schema({
    name: { type: String, required: true },
    url: { type: String, required: true },
    type: { type: String, enum: ['document', 'image', 'video', 'other'] },
    size: { type: Number }, // in KB
    uploadDate: { type: Date, default: Date.now }
});

// Subdocument for comments
const commentSchema = new Schema({
    userId: { type: Schema.Types.ObjectId, required: true, ref: 'User' },
    name: { type: String, required: true },
    text: { type: String, required: true },
    date: { type: Date, default: Date.now }
});

```



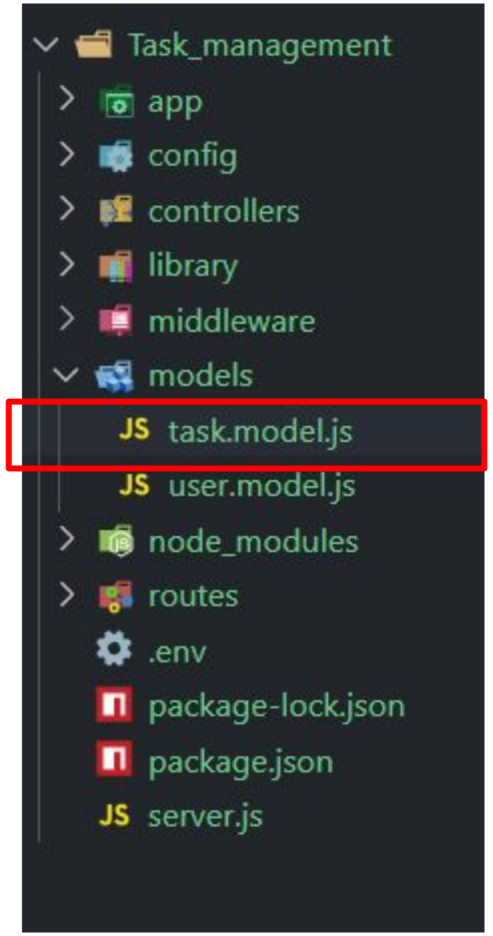
```
Task_management
  app
  config
  controllers
  library
  middleware
  models
    task.model.js
    user.model.js
  node_modules
  routes
  .env
  package-lock.json
  package.json
  server.js
```

Mongodb

Crear Modelo de Tarea
models->task.model.js



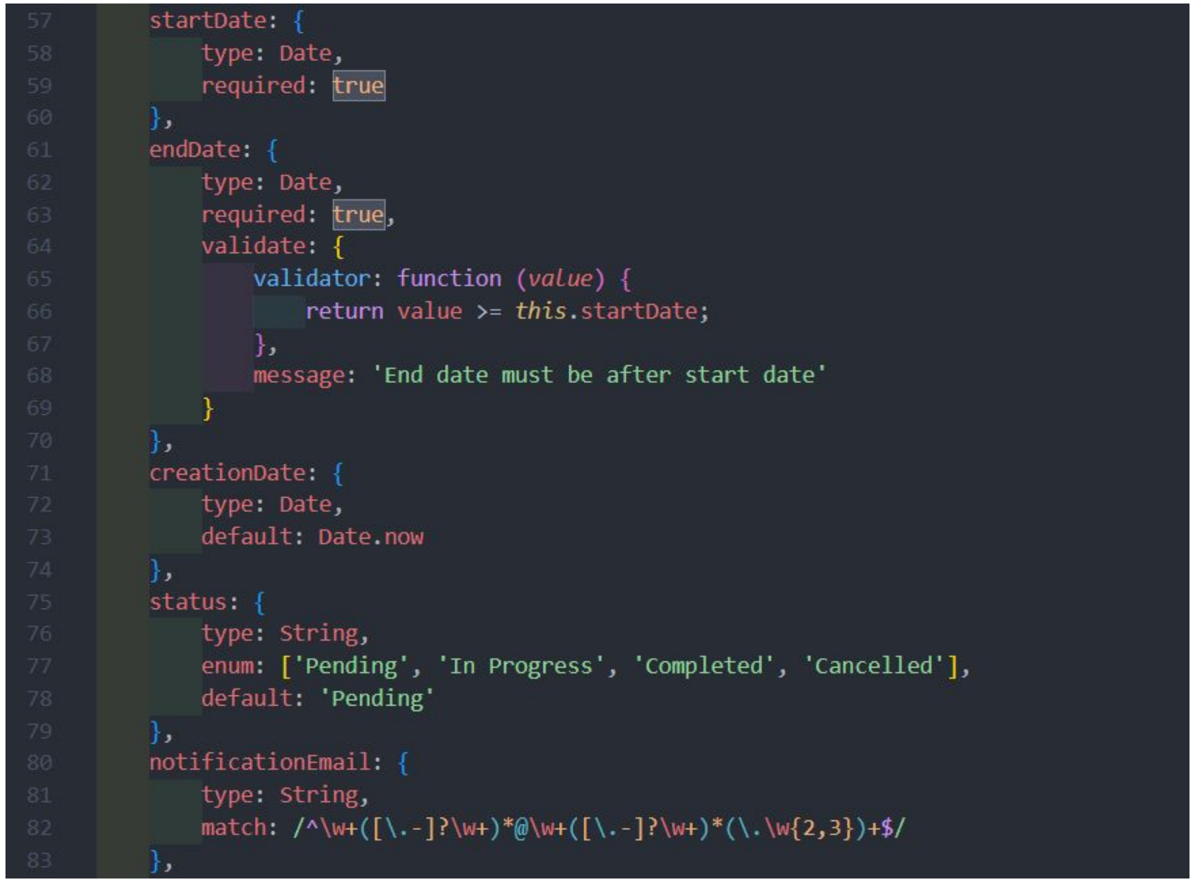
```
// Main Task schema
const taskSchema = new Schema({

    title: {
        type: String,
        required: true,
        trim: true,
        maxlength: 100
    },
    description: {
        type: String,
        trim: true
    },
    type: {
        type: String,
        enum: ['Development', 'Design', 'Meeting', 'Documentation', 'Other'],
        required: true
    },
    priority: {
        type: String,
        enum: ['High', 'Medium', 'Low'],
        default: 'Medium'
    },
    color: {
        type: String,
        default: '#3498db',
        match: /^#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})$/
    },
```



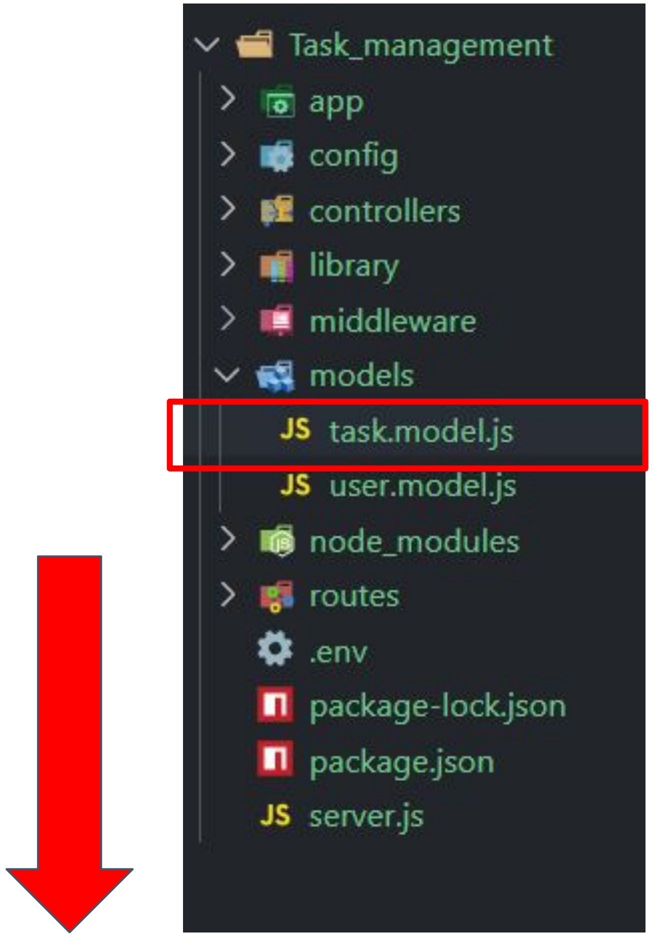
```
Task_management
  app
  config
  controllers
  library
  middleware
  models
    task.model.js
    user.model.js
  node_modules
  routes
  .env
  package-lock.json
  package.json
  server.js
```

Mongodb

Crear Modelo de Tarea
models->task.model.js



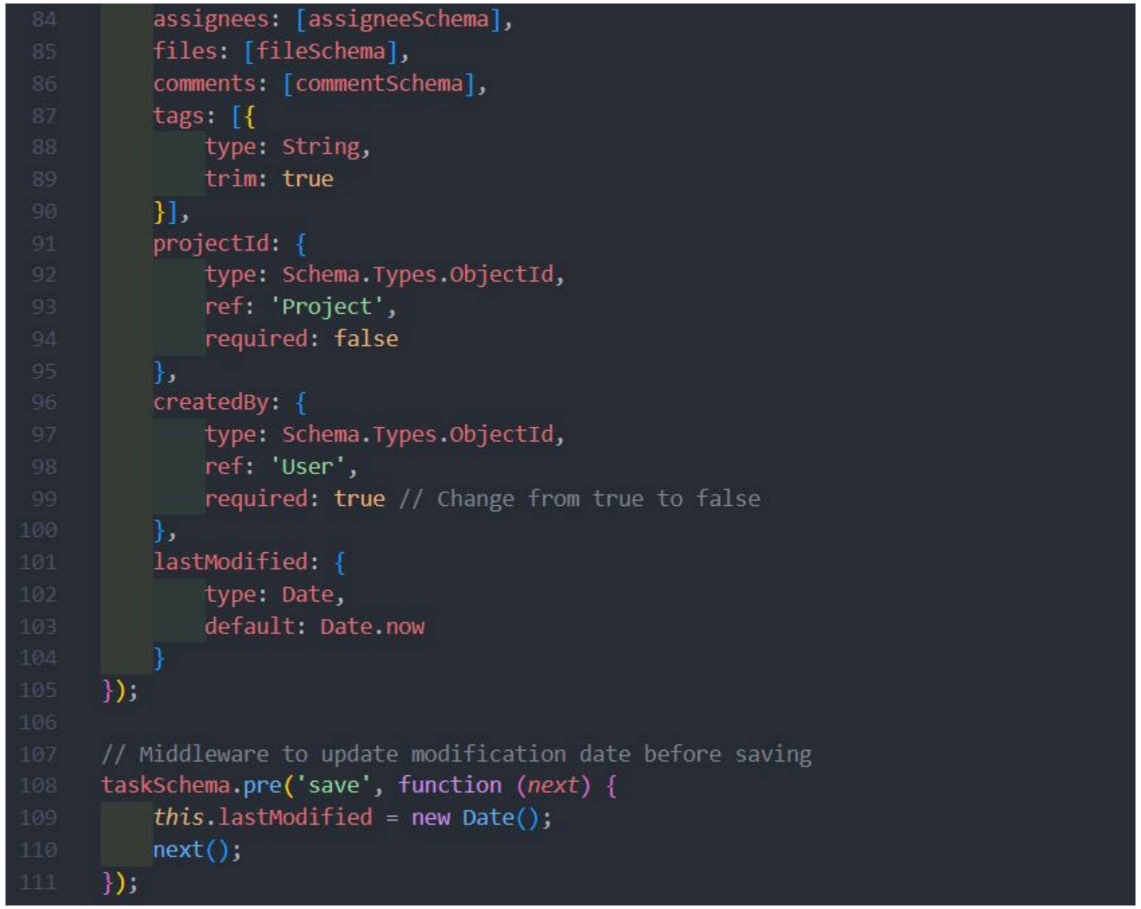
```
    startDate: {
        type: Date,
        required: true
    },
    endDate: {
        type: Date,
        required: true,
        validate: {
            validator: function (value) {
                return value >= this.startDate;
            },
            message: 'End date must be after start date'
        }
    },
    creationDate: {
        type: Date,
        default: Date.now
    },
    status: {
        type: String,
        enum: ['Pending', 'In Progress', 'Completed', 'Cancelled'],
        default: 'Pending'
    },
    notificationEmail: {
        type: String,
        match: /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/
    },
```



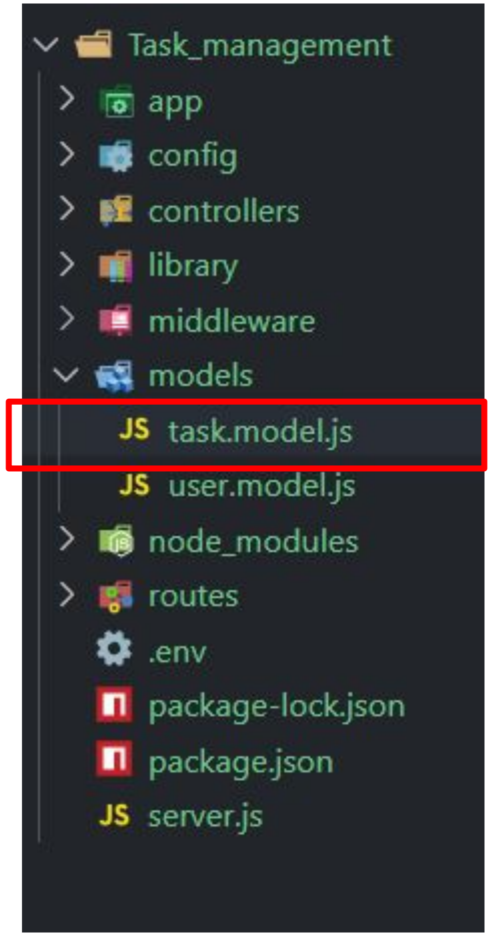
```
Task_management
  app
  config
  controllers
  library
  middleware
  models
    task.model.js
    user.model.js
  node_modules
  routes
  .env
  package-lock.json
  package.json
  server.js
```

Mongodb

Crear Modelo de Tarea
models->task.model.js

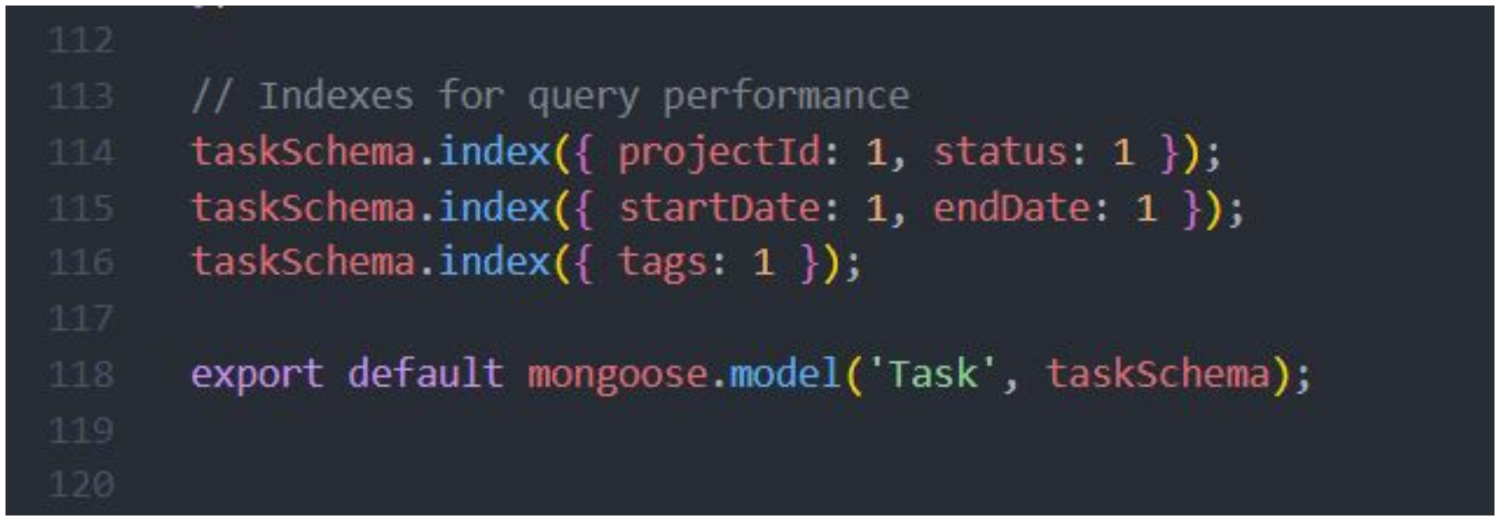


```
        assignees: [assigneeSchema],
        files: [fileSchema],
        comments: [commentSchema],
        tags: [{
            type: String,
            trim: true
        }],
        projectId: {
            type: Schema.Types.ObjectId,
            ref: 'Project',
            required: false
        },
        createdBy: {
            type: Schema.Types.ObjectId,
            ref: 'User',
            required: true // Change from true to false
        },
        lastModified: {
            type: Date,
            default: Date.now
        }
    });

    // Middleware to update modification date before saving
    taskSchema.pre('save', function (next) {
        this.lastModified = new Date();
        next();
    });
```

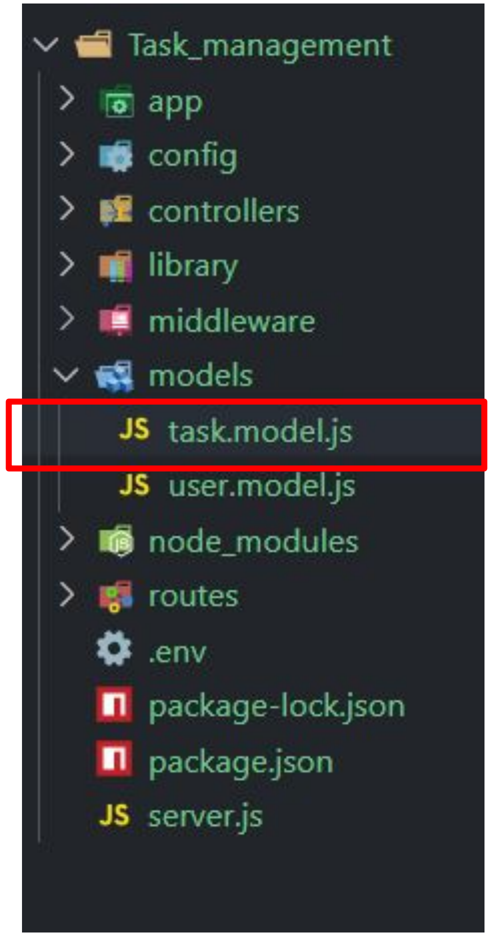


Mongodb

Crear Modelo de Tarea
models->task.model.js



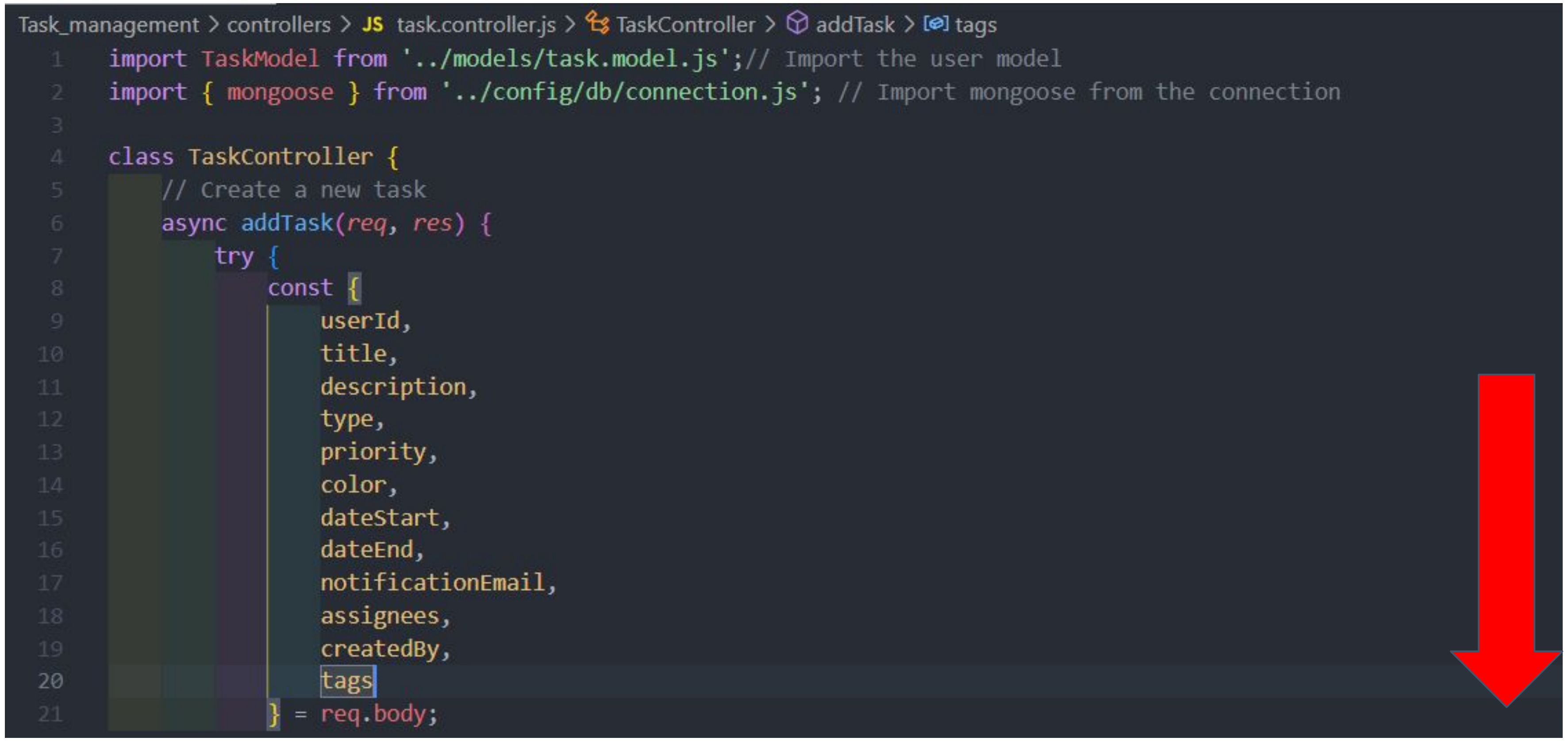
```
// Indexes for query performance
taskSchema.index({ projectId: 1, status: 1 });
taskSchema.index({ startDate: 1, endDate: 1 });
taskSchema.index({ tags: 1 });

export default mongoose.model('Task', taskSchema);
```



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

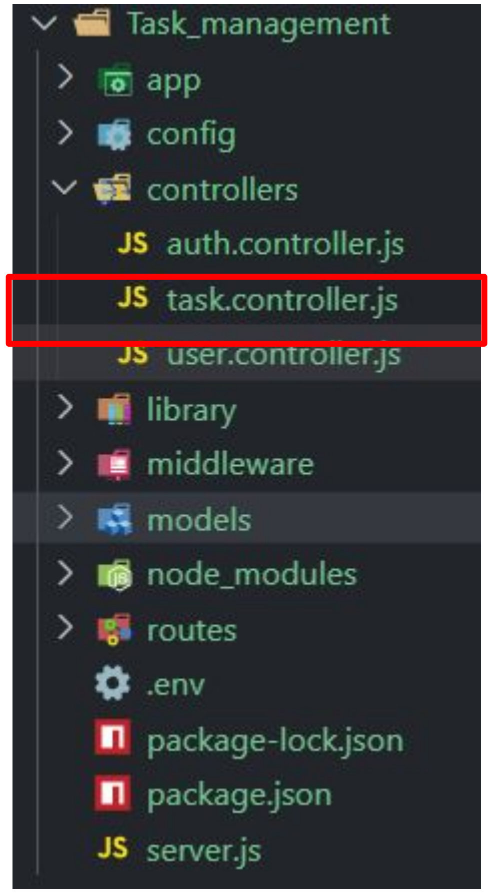
Mongodb

Crear Controlador de Tarea
controllers->task.controller.js



Task_management > controllers > JS task.controller.js > TaskController > addTask > tags

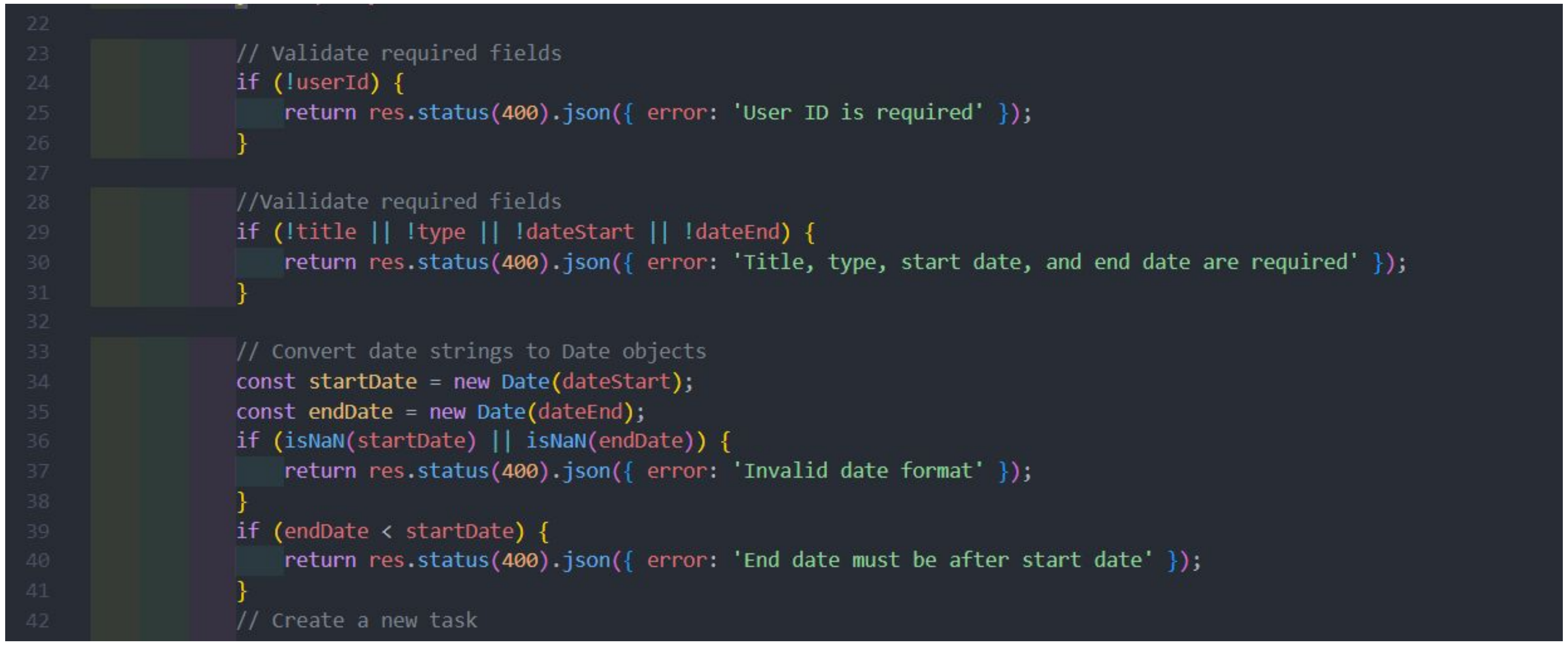
```javascript
import TaskModel from '../models/task.model.js';// Import the user model
import { mongoose } from '../config/db/connection.js'; // Import mongoose from the connection

class TaskController {
    // Create a new task
    async addTask(req, res) {
        try {
            const {
                userId,
                title,
                description,
                type,
                priority,
                color,
                dateStart,
                dateEnd,
                notificationEmail,
                assignees,
                createdBy,
                tags
            } = req.body;
```



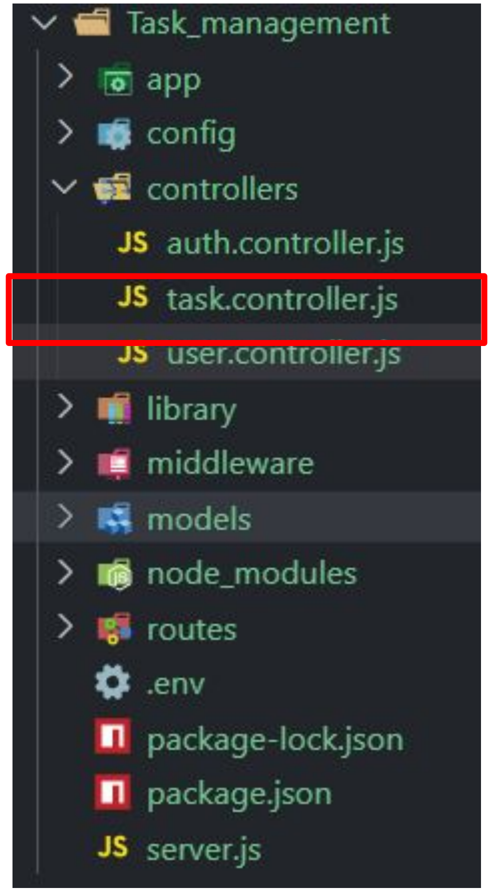
```
Task_management
  app
  config
  controllers
    auth.controller.js
    task.controller.js
    user.controller.js
  library
  middleware
  models
  node_modules
  routes
  .env
  package-lock.json
  package.json
  server.js
```

Mongodb

Crear Controlador de Tarea
controllers->task.controller.js



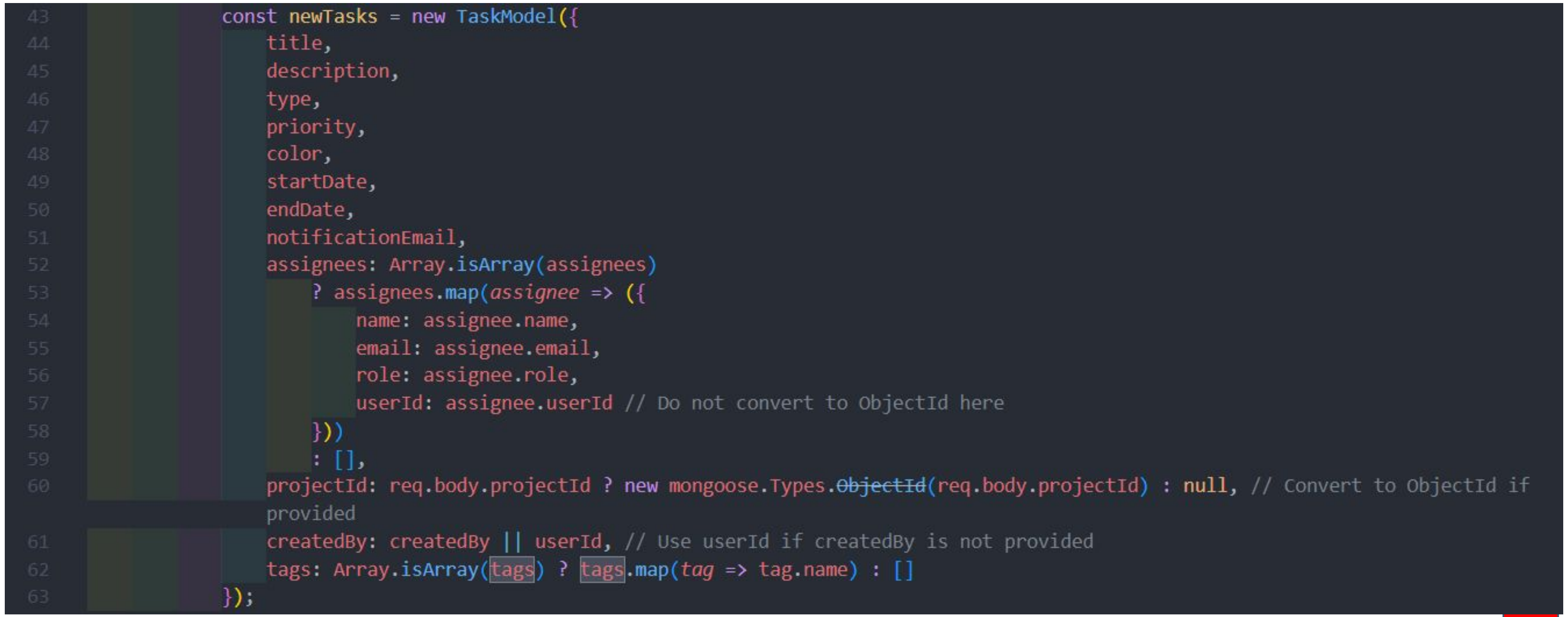
```
        // Validate required fields
        if (!userId) {
            return res.status(400).json({ error: 'User ID is required' });
        }

        //Vailidate required fields
        if (!title || !type || !dateStart || !dateEnd) {
            return res.status(400).json({ error: 'Title, type, start date, and end date are required' });
        }

        // Convert date strings to Date objects
        const startDate = new Date(dateStart);
        const endDate = new Date(dateEnd);
        if (isNaN(startDate) || isNaN(endDate)) {
            return res.status(400).json({ error: 'Invalid date format' });
        }
        if (endDate < startDate) {
            return res.status(400).json({ error: 'End date must be after start date' });
        }
        // Create a new task
```



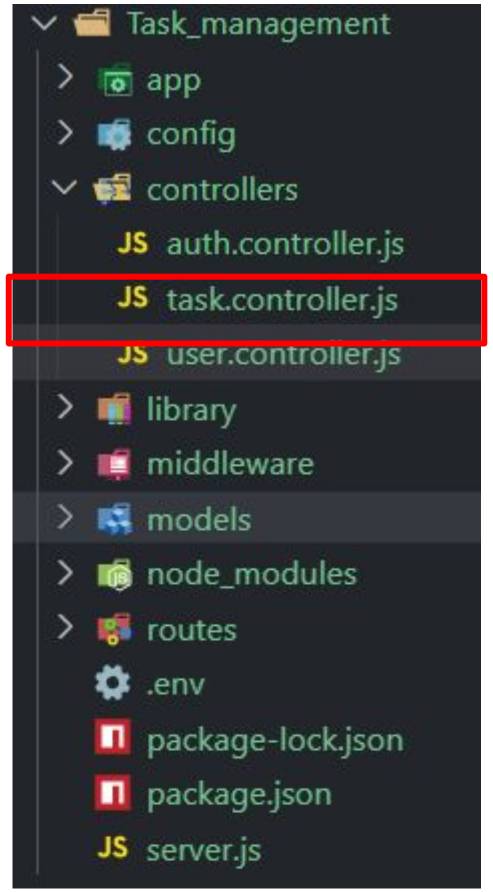
Mongodb

Crear Controlador de Tarea

controllers->task.controller.js



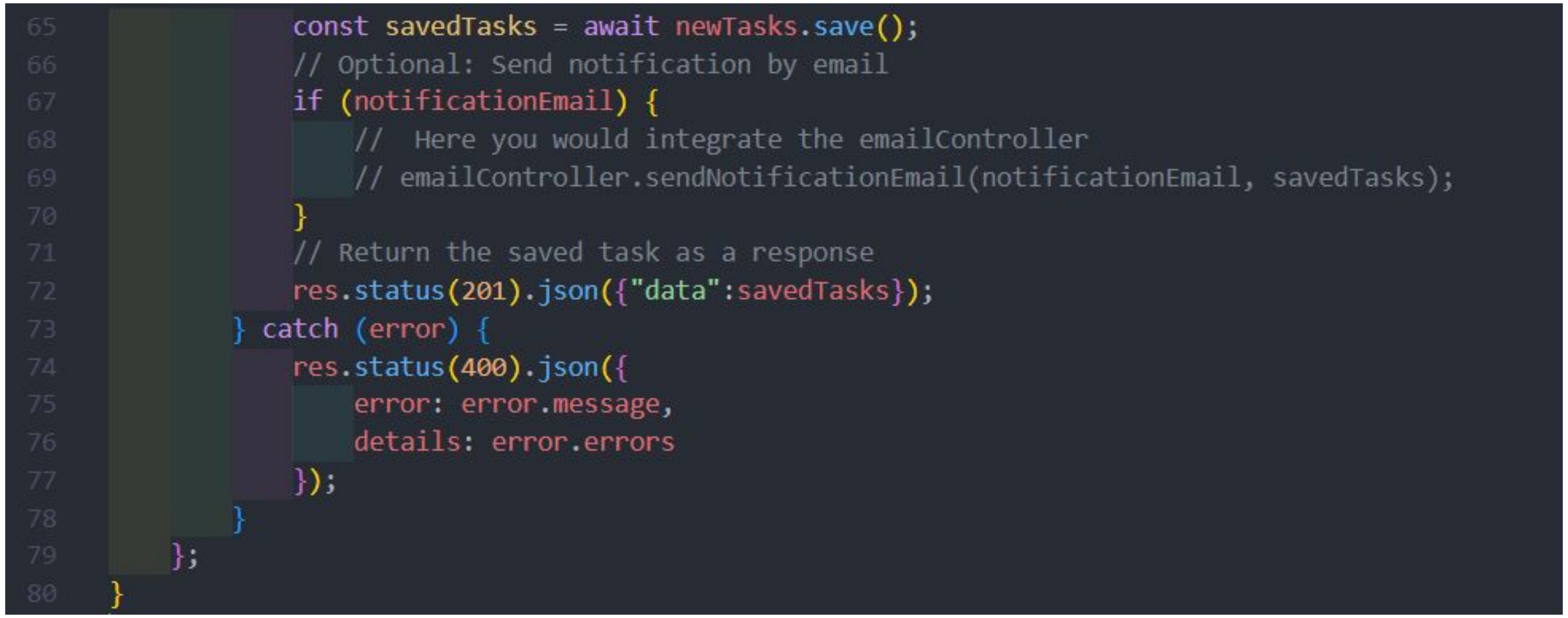
```
        const newTasks = new TaskModel({
            title,
            description,
            type,
            priority,
            color,
            startDate,
            endDate,
            notificationEmail,
            assignees: Array.isArray(assignees)
                ? assignees.map(assignee => ({
                    name: assignee.name,
                    email: assignee.email,
                    role: assignee.role,
                    userId: assignee.userId // Do not convert to ObjectId here
                }))
                : [],
            projectId: req.body.projectId ? new mongoose.Types.~~ObjectId~~(req.body.projectId) : null, // Convert to ObjectId if provided
            createdBy: createdBy || userId, // Use userId if createdBy is not provided
            tags: Array.isArray(tags) ? tags.map(tag => tag.name) : []
        });
```



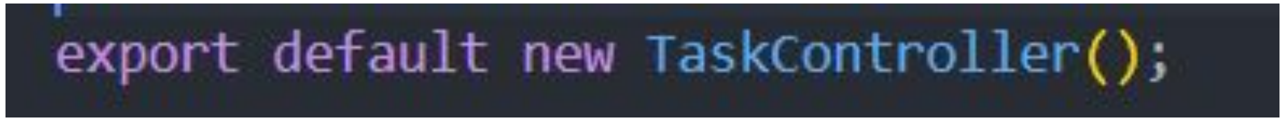
```
Task_management
  app
  config
  controllers
    auth.controller.js
    task.controller.js
    user.controller.js
  library
  middleware
  models
  node_modules
  routes
  .env
  package-lock.json
  package.json
  server.js
```

Mongodb

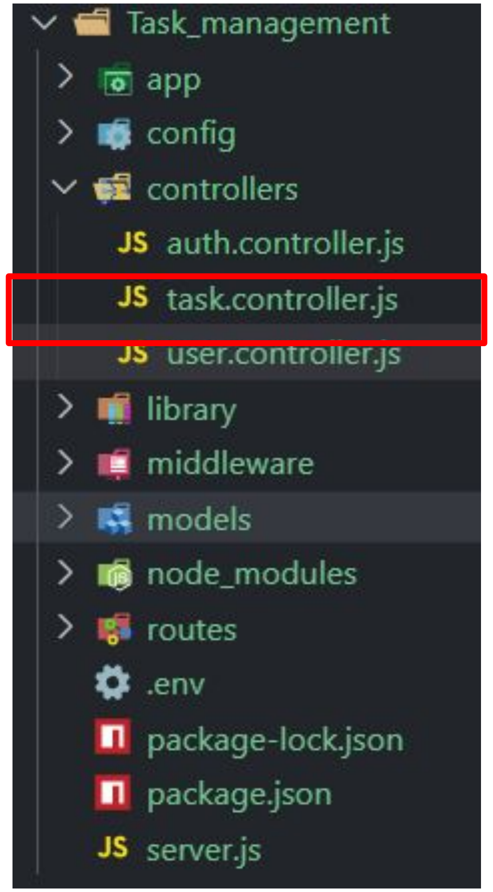
Crear Controlador de Tarea controllers->task.controller.js



```
            const savedTasks = await newTasks.save();
            // Optional: Send notification by email
            if (notificationEmail) {
                //  Here you would integrate the emailController
                // emailController.sendNotificationEmail(notificationEmail, savedTasks);
            }
            // Return the saved task as a response
            res.status(201).json({"data":savedTasks});
        } catch (error) {
            res.status(400).json({
                error: error.message,
                details: error.errors
            });
        }
    };
}
```



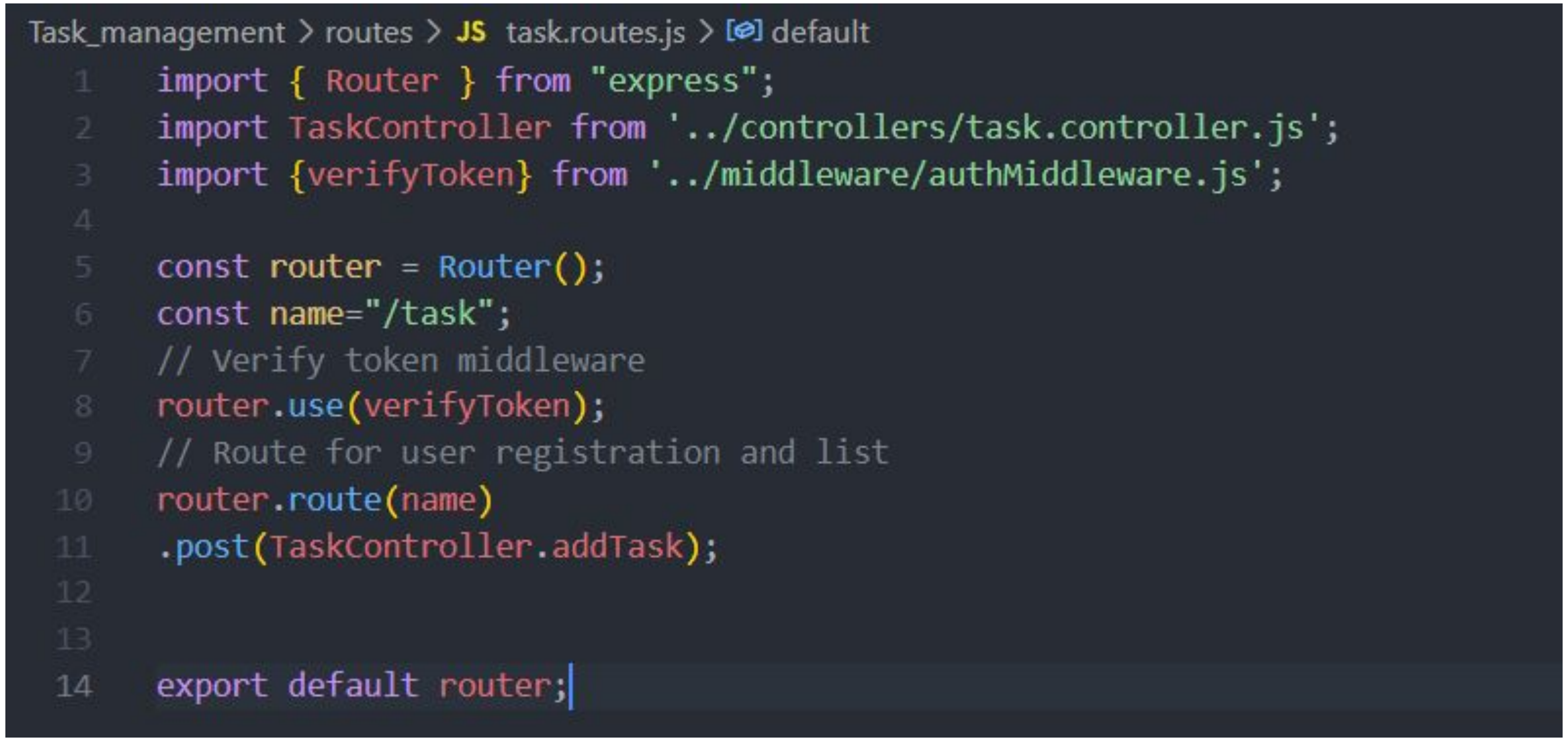
```
export default new TaskController();
```



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

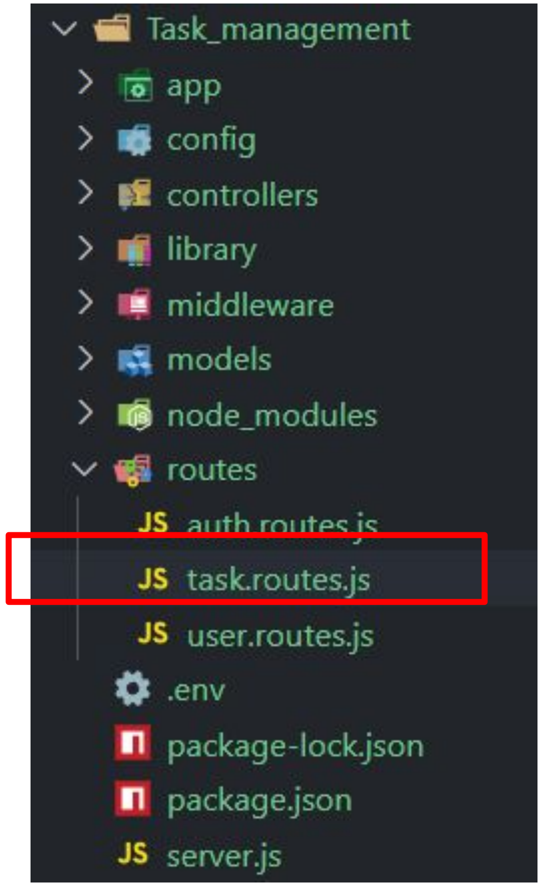
Mongodb

Crear Rutas de Tarea
routes->task.routes.js



Task_management > routes > JS task.routes.js > [ø] default

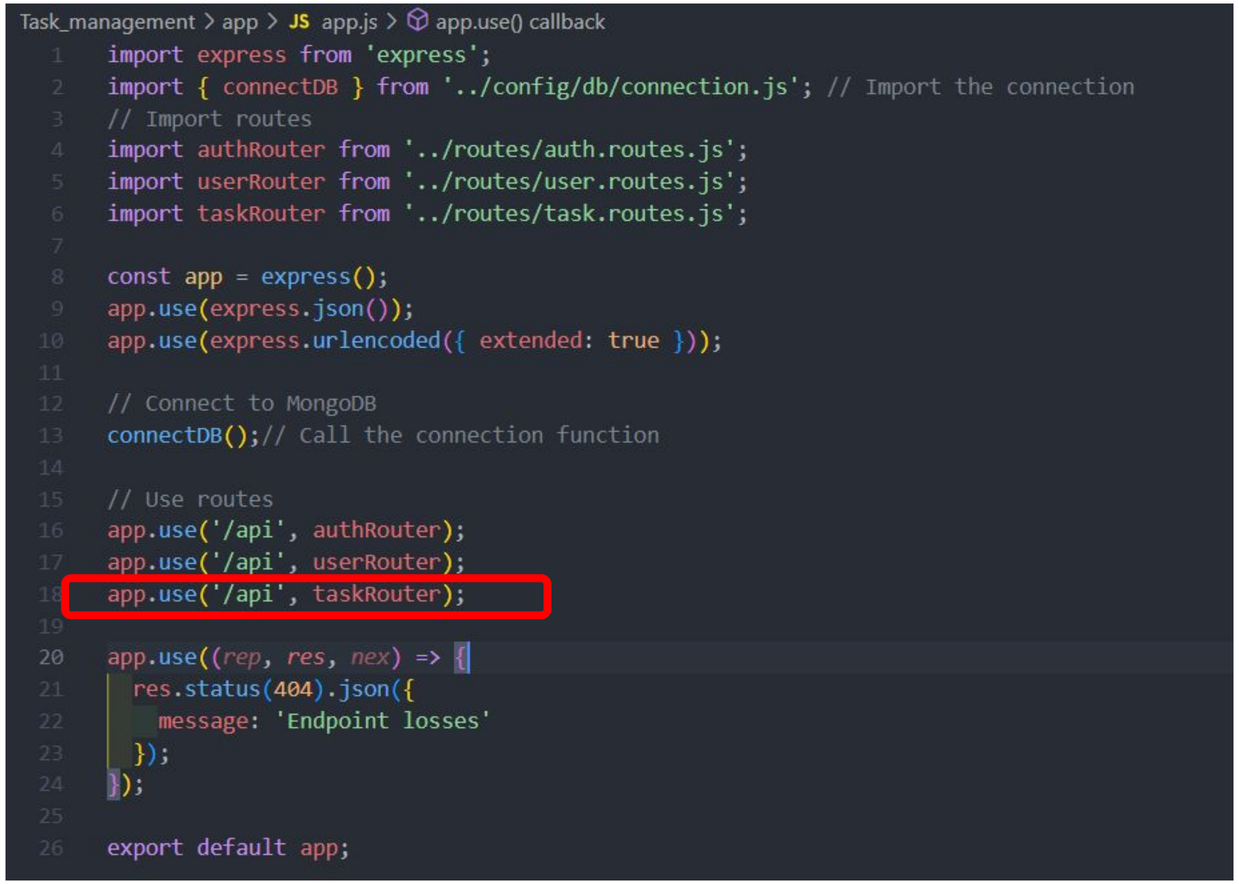
```
import { Router } from "express";
import TaskController from '../controllers/task.controller.js';
import {verifyToken} from '../middleware/authMiddleware.js';

const router = Router();
const name="/task";
// Verify token middleware
router.use(verifyToken);
// Route for user registration and list
router.route(name)
.post(TaskController.addTask);


export default router;
```



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

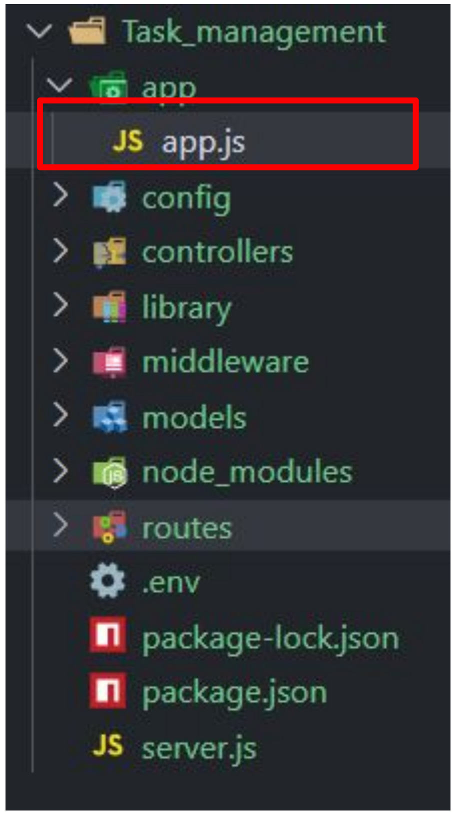
Mongodb

Agregar rutas al proyecto
app->app.js



Task_management > app > JS app.js > app.use() callback

```javascript
import express from 'express';
import { connectDB } from '../config/db/connection.js'; // Import the connection
// Import routes
import authRouter from '../routes/auth.routes.js';
import userRouter from '../routes/user.routes.js';
import taskRouter from '../routes/task.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('/api', taskRouter);

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

export default app;
```

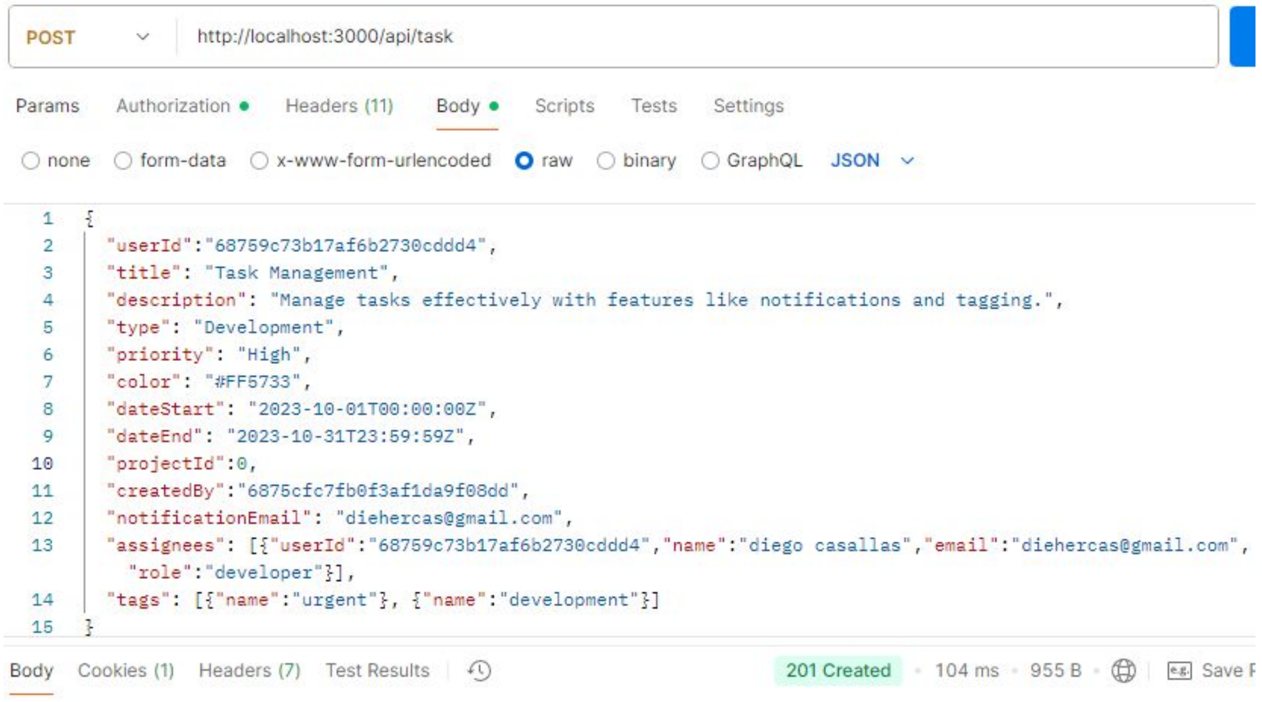


```
Task_management
  app
    app.js
  config
  controllers
  library
  middleware
  models
  node_modules
  routes
  .env
  package-lock.json
  package.json
  server.js
```

Mongodb

Pruebas

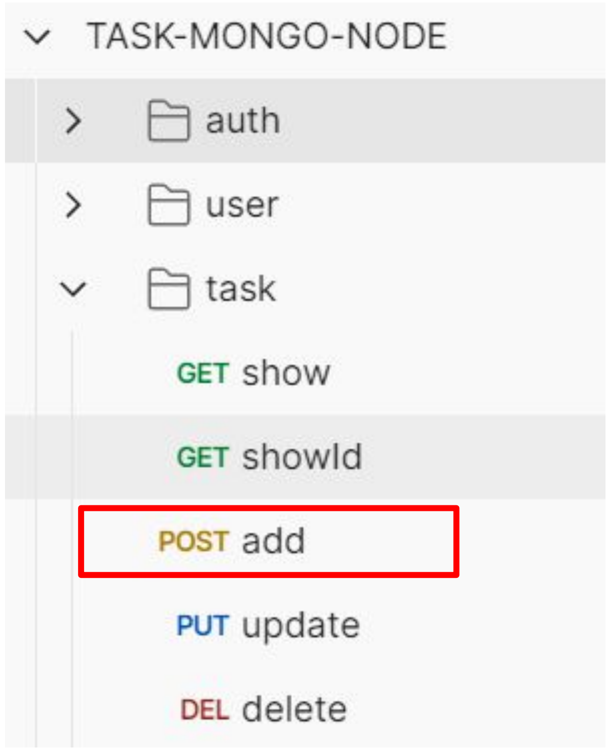
DATOS



```
http://localhost:3000/api/task
```

```
{
  "userId":"68759c73b17af6b2730cddd4",
  "title": "Task Management",
  "description": "Manage tasks effectively with features like notifications and tagging.",
  "type": "Development",
  "priority": "High",
  "color": "#FF5733",
  "dateStart": "2023-10-01T00:00:00Z",
  "dateEnd": "2023-10-31T23:59:59Z",
  "projectId":0,
  "createdBy":"6875cfc7fb0f3af1da9f08dd",
  "notificationEmail": "diehercas@gmail.com",
  "assignees": [{"userId":"68759c73b17af6b2730cddd4","name":"diego casallas","email":"diehercas@gmail.com",
    "role":"developer"}],
  "tags": [{"name":"urgent"}, {"name":"development"}]
}
```

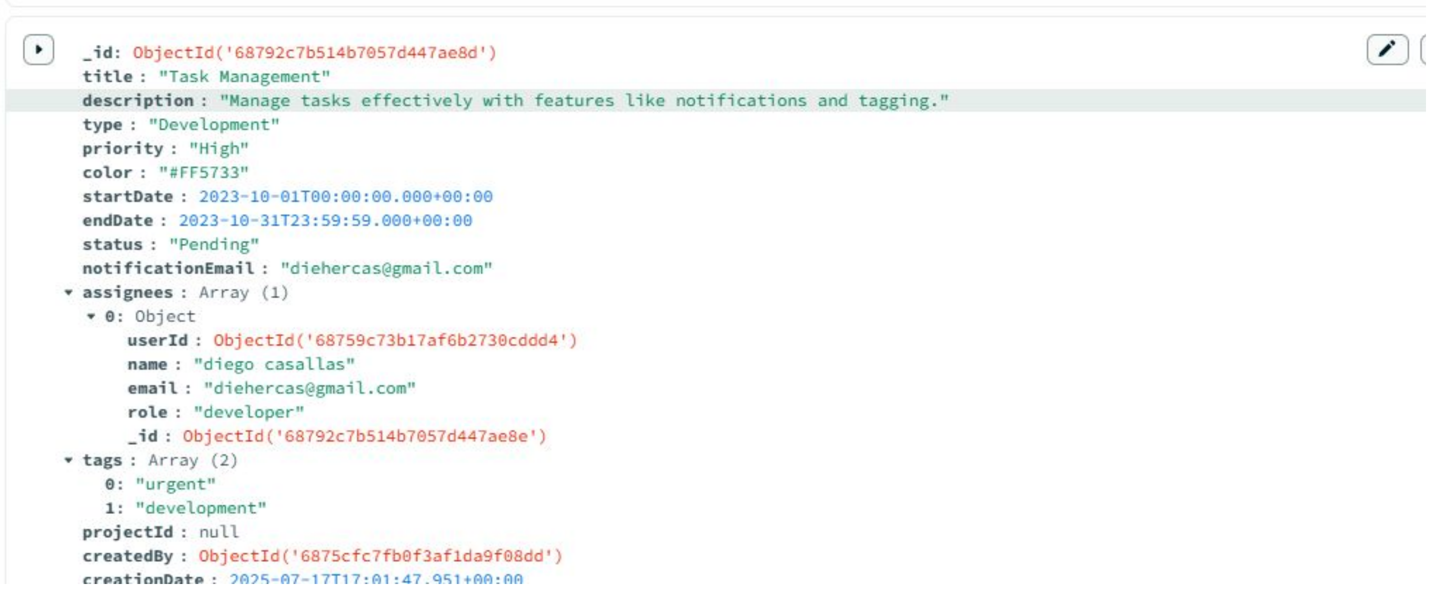
201 Created · 104 ms · 955 B



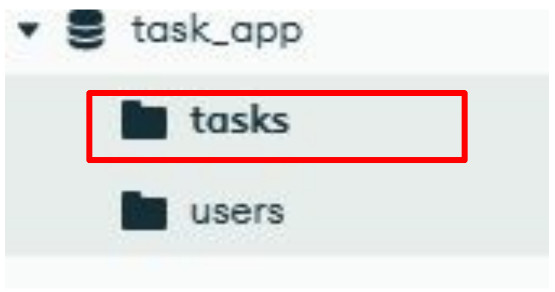
MÉTODO	RUTA	CONTROLADOR	MODELO
POST	http://localhost:3000/api/task	Task	Task

Mongodb

MongoDB



```
_id: ObjectId('68792c7b514b7057d447ae8d')
title : "Task Management"
description : "Manage tasks effectively with features like notifications and tagging."
type : "Development"
priority : "High"
color : "#FF5733"
startDate : 2023-10-01T00:00:00.000+00:00
endDate : 2023-10-31T23:59:59.000+00:00
status : "Pending"
notificationEmail : "diehercas@gmail.com"
assignees : Array (1)
  0: Object
    userId : ObjectId('68759c73b17af6b2730cddd4')
    name : "diego casallas"
    email : "diehercas@gmail.com"
    role : "developer"
    _id : ObjectId('68792c7b514b7057d447ae8e')
tags : Array (2)
  0: "urgent"
  1: "development"
projectId : null
createdBy : ObjectId('6875cfc7fb0f3af1da9f08dd')
creationDate : 2025-07-17T17:01:47.951+00:00
```



G R A C I A S