



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

**ING - DIEGO CASALLAS**

**22810019-1**



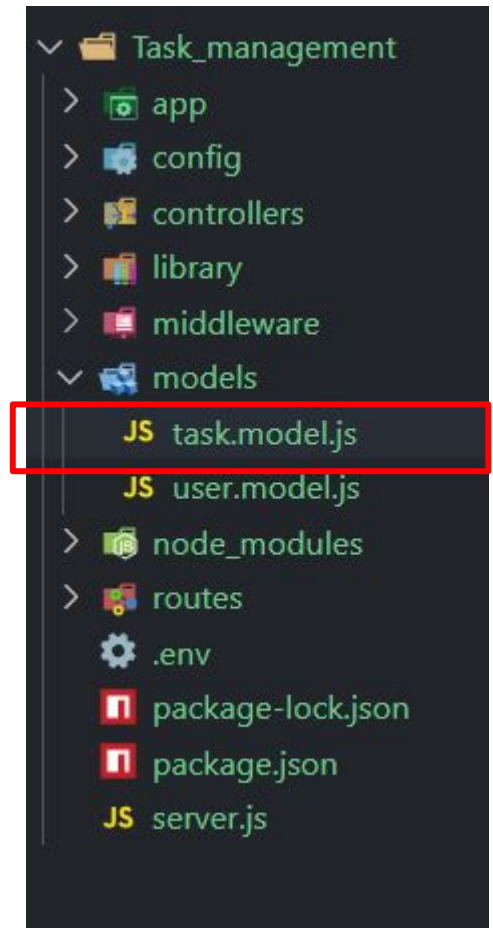
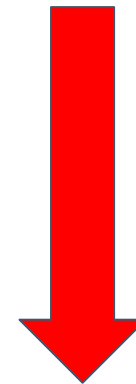
[www.sena.edu.co](http://www.sena.edu.co)

# Mongodb



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

```
Task_management > models > JS task.model.js > fileSchema
1  import { mongoose } from '../config/db/connection.js'; // Import mongoose from the connection
2  const Schema = mongoose.Schema;
3
4  // Subdocument for assigned users
5  const assigneeSchema = new Schema({
6    userId: { type: Schema.Types.ObjectId, required: true, ref: 'User' },
7    name: { type: String, required: true },
8    email: { type: String, required: true },
9    role: { type: String, required: true }
10 });
11
12 // Subdocument for files
13 const fileSchema = new Schema({
14   name: { type: String, required: true },
15   url: { type: String, required: true },
16   type: { type: String, enum: ['document', 'image', 'video', 'other'] },
17   size: { type: Number }, // in KB
18   uploadDate: { type: Date, default: Date.now }
19 });
20
21 // Subdocument for comments
22 const commentSchema = new Schema({
23   userId: { type: Schema.Types.ObjectId, required: true, ref: 'User' },
24   name: { type: String, required: true },
25   text: { type: String, required: true },
26   date: { type: Date, default: Date.now }
27 });
28
```

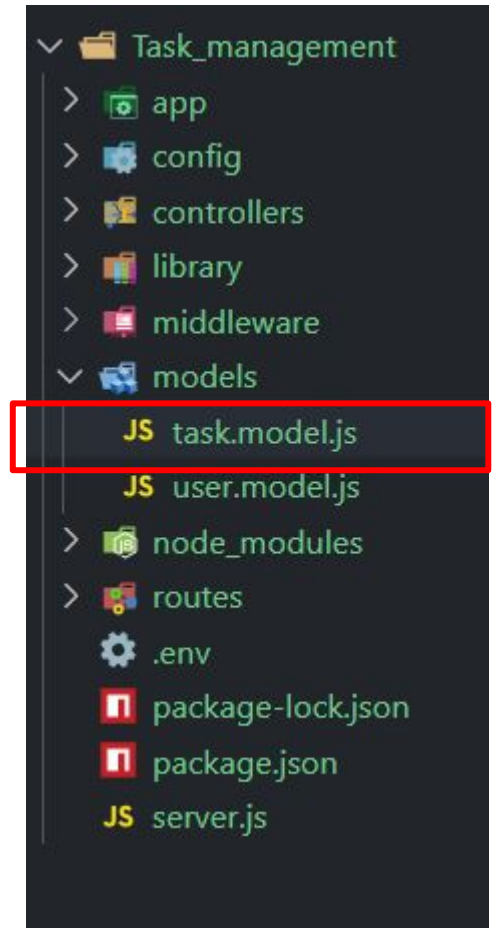
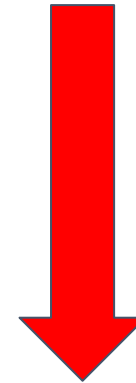


# Mongodb



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

```
29 // Main Task schema
30 const taskSchema = new Schema({
31
32   title: {
33     type: String,
34     required: true,
35     trim: true,
36     maxlength: 100
37   },
38   description: {
39     type: String,
40     trim: true
41   },
42   type: {
43     type: String,
44     enum: ['Development', 'Design', 'Meeting', 'Documentation', 'Other'],
45     required: true
46   },
47   priority: {
48     type: String,
49     enum: ['High', 'Medium', 'Low'],
50     default: 'Medium'
51   },
52   color: {
53     type: String,
54     default: '#3498db',
55     match: /^[A-Fa-f0-9]{6}||[A-Fa-f0-9]{3}$/
56   },
57 });
```

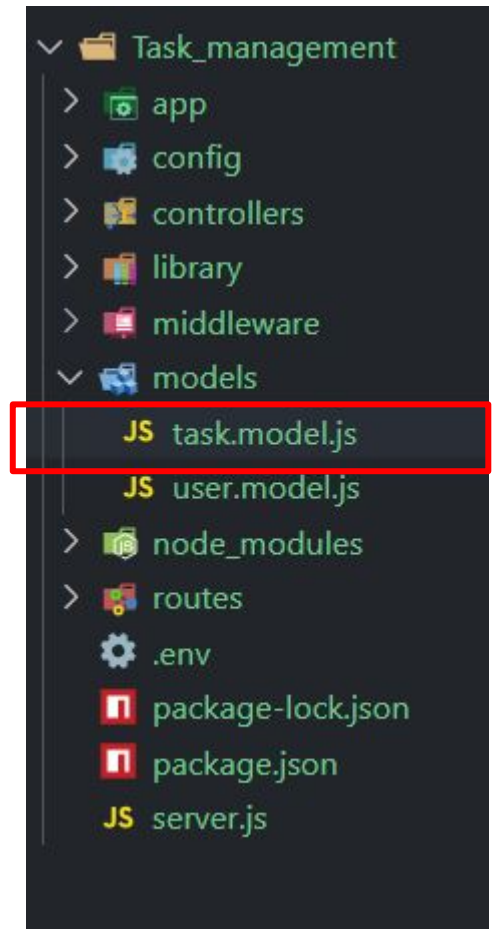
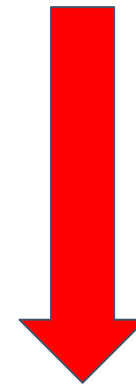


# Mongodb



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

```
57   startDate: {
58     type: Date,
59     required: true
60   },
61   endDate: {
62     type: Date,
63     required: true,
64     validate: {
65       validator: function (value) {
66         return value >= this.startDate;
67       },
68       message: 'End date must be after start date'
69     }
70   },
71   createdAt: {
72     type: Date,
73     default: Date.now
74   },
75   status: {
76     type: String,
77     enum: ['Pending', 'In Progress', 'Completed', 'Cancelled'],
78     default: 'Pending'
79   },
80   notificationEmail: {
81     type: String,
82     match: /^\\w+([\\.-]?\\w+)*@\\w+([\\.-]?\\w+)*\\.\\w{2,3}$/
83   },
```

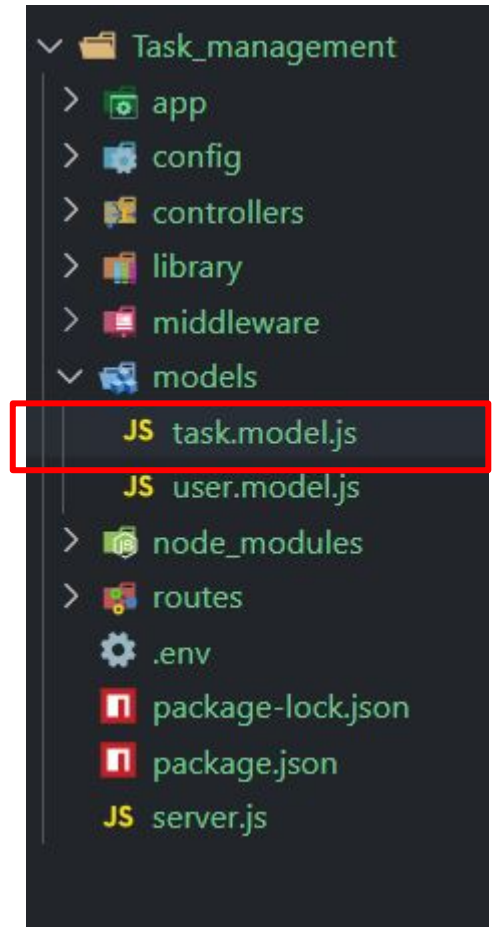
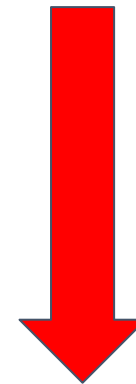


# Mongodb



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

```
84   assignees: [assigneeSchema],
85   files: [fileSchema],
86   comments: [commentSchema],
87   tags: [{
88     type: String,
89     trim: true
90   }],
91   projectId: {
92     type: Schema.Types.ObjectId,
93     ref: 'Project',
94     required: false
95   },
96   createdBy: {
97     type: Schema.Types.ObjectId,
98     ref: 'User',
99     required: true // Change from true to false
100  },
101  lastModified: {
102    type: Date,
103    default: Date.now
104  }
105  });
106
107  // Middleware to update modification date before saving
108  taskSchema.pre('save', function (next) {
109    this.lastModified = new Date();
110    next();
111  });
```



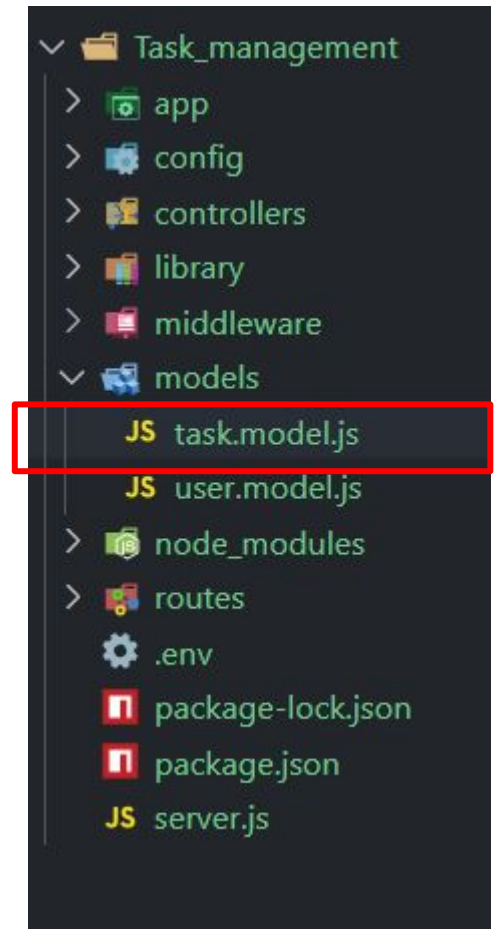


# Mongodb



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

```
112
113 // Indexes for query performance
114 taskSchema.index({ projectId: 1, status: 1 });
115 taskSchema.index({ startDate: 1, endDate: 1 });
116 taskSchema.index({ tags: 1 });
117
118 export default mongoose.model('Task', taskSchema);
119
120
```

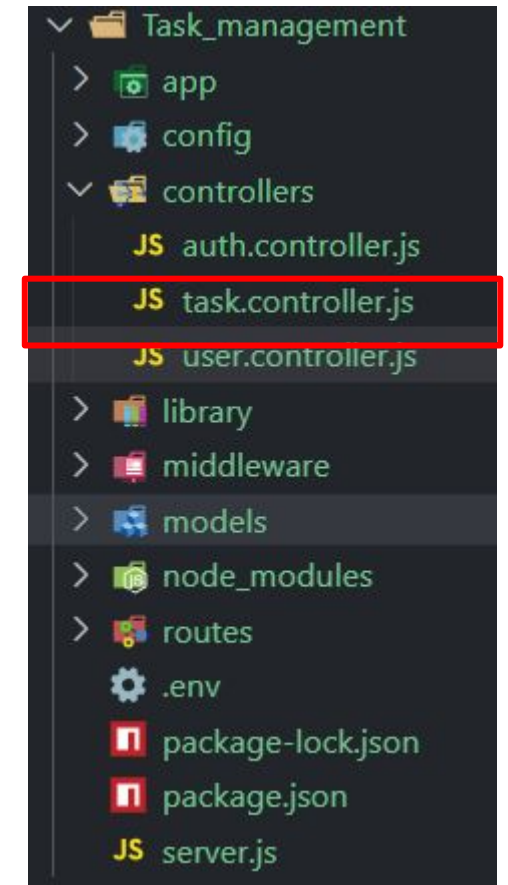


# Mongodb



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

```
Task_management > controllers > JS task.controller.js > TaskController > addTask > tags
1  import TaskModel from '../models/task.model.js'; // Import the user model
2  import { mongoose } from '../config/db/connection.js'; // Import mongoose from the connection
3
4  class TaskController {
5      // Create a new task
6      async addTask(req, res) {
7          try {
8              const {
9                  userId,
10                 title,
11                 description,
12                 type,
13                 priority,
14                 color,
15                 dateStart,
16                 dateEnd,
17                 notificationEmail,
18                 assignees,
19                 createdBy,
20                 tags
21             } = req.body;
```

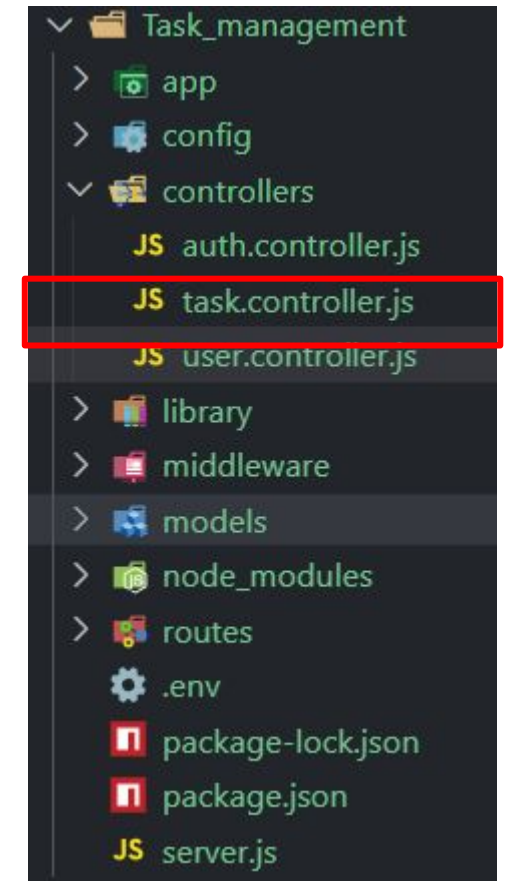


# Mongodb



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

```
22
23 // Validate required fields
24 if (!userId) {
25   return res.status(400).json({ error: 'User ID is required' });
26 }
27
28 //Vailidate required fields
29 if (!title || !type || !dateStart || !dateEnd) {
30   return res.status(400).json({ error: 'Title, type, start date, and end date are required' });
31 }
32
33 // Convert date strings to Date objects
34 const startDate = new Date(dateStart);
35 const endDate = new Date(dateEnd);
36 if (isNaN(startDate) || isNaN(endDate)) {
37   return res.status(400).json({ error: 'Invalid date format' });
38 }
39 if (endDate < startDate) {
40   return res.status(400).json({ error: 'End date must be after start date' });
41 }
42 // Create a new task
```



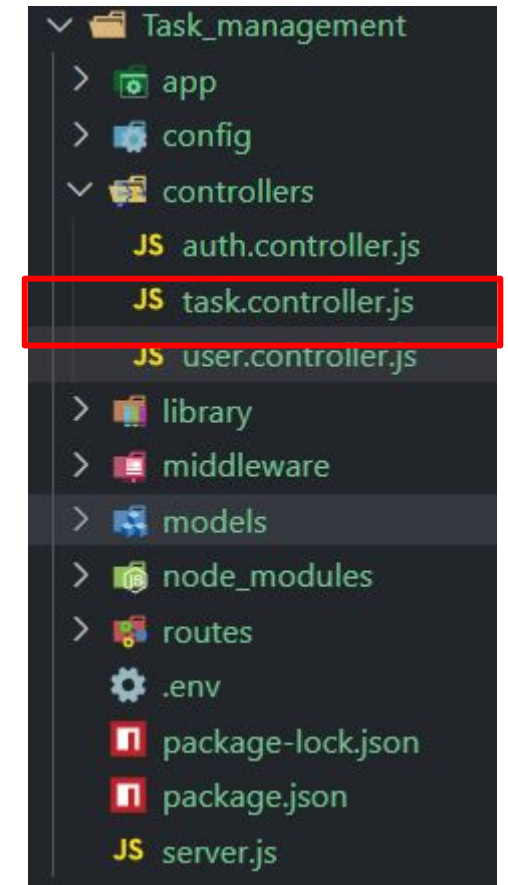


# Mongodb



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

```
43 const newTasks = new TaskModel({
44   title,
45   description,
46   type,
47   priority,
48   color,
49   startDate,
50   endDate,
51   notificationEmail,
52   assignees: Array.isArray(assignees)
53     ? assignees.map(assignee => ({
54       name: assignee.name,
55       email: assignee.email,
56       role: assignee.role,
57       userId: assignee.userId // Do not convert to ObjectId here
58     }))
59     : [],
60   projectId: req.body.projectId ? new mongoose.Types.ObjectId(req.body.projectId) : null, // Convert to ObjectId if
    provided
61   createdBy: createdBy || userId, // Use userId if createdBy is not provided
62   tags: Array.isArray(tags) ? tags.map(tag => tag.name) : []
63 });
```



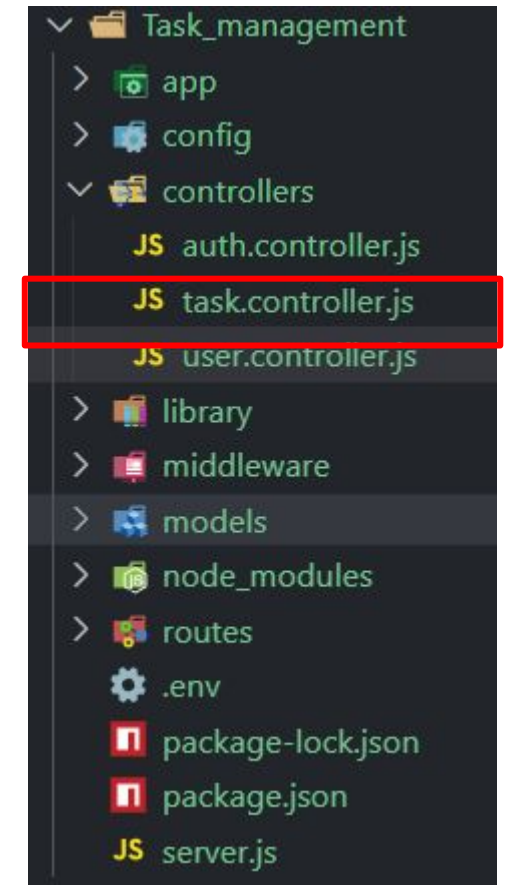
# Mongodb



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

```
65     const savedTasks = await newTasks.save();
66     // Optional: Send notification by email
67     if (notificationEmail) {
68         // Here you would integrate the emailController
69         // emailController.sendNotificationEmail(notificationEmail, savedTasks);
70     }
71     // Return the saved task as a response
72     res.status(201).json({"data":savedTasks});
73 } catch (error) {
74     res.status(400).json({
75         error: error.message,
76         details: error.errors
77     });
78 }
79 };
80 }
```

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

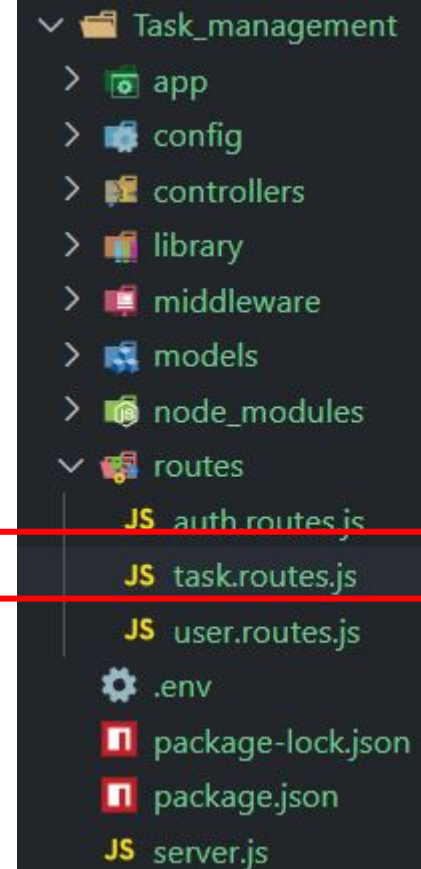


# Mongodb



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

```
Task_management > routes > JS task.routes.js > [?] default
1  import { Router } from "express";
2  import TaskController from '../controllers/task.controller.js';
3  import {verifyToken} from '../middleware/authMiddleware.js';
4
5  const router = Router();
6  const name="/task";
7  // Verify token middleware
8  router.use(verifyToken);
9  // Route for user registration and list
10 router.route(name)
11   .post(TaskController.addTask);
12
13
14 export default router;
```



A file explorer view of a project named 'Task\_management'. The 'routes' folder is expanded, showing three files: 'auth.routes.js', 'task.routes.js' (highlighted with a red box), and 'user.routes.js'. Below the 'routes' folder are other files: '.env', 'package-lock.json', 'package.json', and 'server.js'. A large red arrow points downwards from the 'task.routes.js' file towards the code editor on the left.

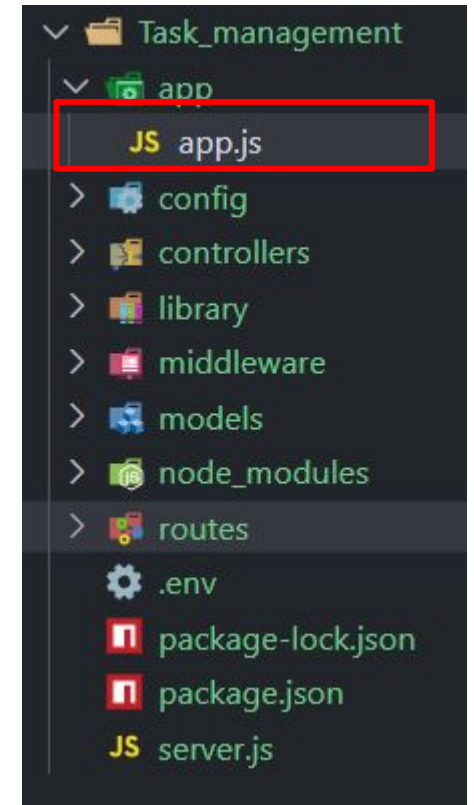
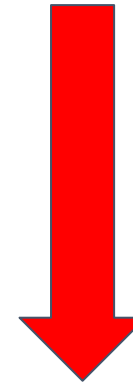
```
Task_management
├── app
├── config
├── controllers
├── library
├── middleware
├── models
├── node_modules
├── routes
│   ├── JS auth.routes.js
│   ├── JS task.routes.js
│   └── JS user.routes.js
├── .env
├── package-lock.json
├── package.json
└── JS server.js
```

# Mongodb



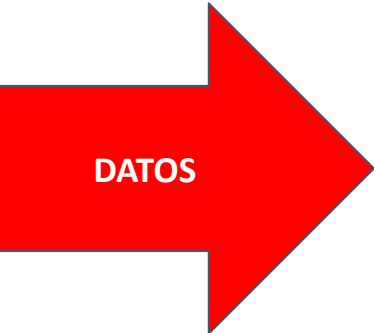
## Agregar rutas al proyecto app->app.js

```
Task_management > app > JS app.js > app.use() callback
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  import userRouter from '../routes/user.routes.js';
6  import taskRouter from '../routes/task.routes.js';
7
8  const app = express();
9  app.use(express.json());
10 app.use(express.urlencoded({ extended: true }));
11
12 // Connect to MongoDB
13 connectDB();// Call the connection function
14
15 // Use routes
16 app.use('/api', authRouter);
17 app.use('/api', userRouter);
18 app.use('/api', taskRouter);
19
20 app.use((req, res, next) => {
21   res.status(404).json({
22     message: 'Endpoint losses'
23   });
24 });
25
26 export default app;
```





## Pruebas



POST

http://localhost:3000/api/task

Params

Authorization

Headers (11)

Body

Scripts

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1 {

2   "userId": "68759c73b17af6b2730cddd4",

3   "title": "Task Management",

4   "description": "Manage tasks effectively with features like notifications and tagging.",

5   "type": "Development",

6   "priority": "High",

7   "color": "#FF5733",

8   "dateStart": "2023-10-01T00:00:00Z",

9   "dateEnd": "2023-10-31T23:59:59Z",

10   "projectId": 0,

11   "createdBy": "6875cfc7fb0f3af1da9f08dd",

12   "notificationEmail": "diehercas@gmail.com",

13   "assignees": [{ "userId": "68759c73b17af6b2730cddd4", "name": "diego casallas", "email": "diehercas@gmail.com",

14     "role": "developer" }],

15   "tags": [{ "name": "urgent" }, { "name": "development" }]

16 }

Body

Cookies (1)

Headers (7)

Test Results

201 Created

104 ms

955 B

Save F

TASK-MONGO-NODE

>

auth

>

user

>

task

GET show

GET showId

POST add

PUT update

DEL delete

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



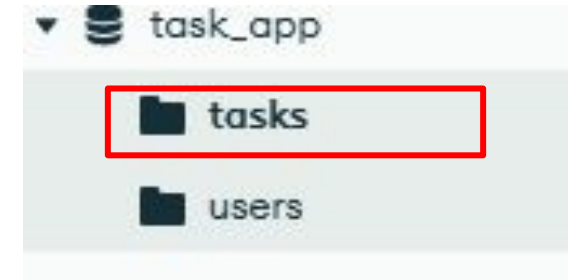
# Mongodb



## 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
```





# GRACIAS

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



[www.sena.edu.co](http://www.sena.edu.co)