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

ING - DIEGO CASALLAS **22810019-1**

Node.js - Mongo.db

CRUD - Task

- **Controlador de tareas**
  - addTask
  - getTaskId
  - getTasks
  - updateTask
  - addComment
  - addFile

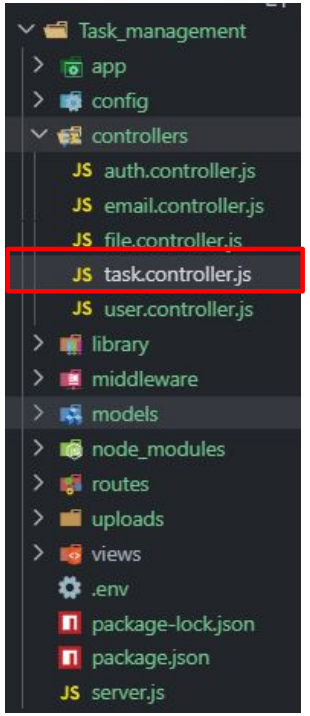
Task_management > controllers > JS task.controller.js > default

```
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) {···
79      };
80
81      //show all tasks
82  >   async getTaskId(req, res) {···
93      };
94      //show all tasks
95  >   async getTasks(req, res) {···
105     };
106
107     // Update a task add array of managers
108 >   async updateTask(req, res) {···
137     };
138
139     //Add comment to a task
140 >   async addComment(req, res) {···
162     };
163
164     // Add file to a task
165 >   async addFile(req, res) {···
192     };
193 }
194
195 export default new TaskController();
```

Node.js - Mongo.db

CRUD - Task

- Controlador de tareas
  - addTask



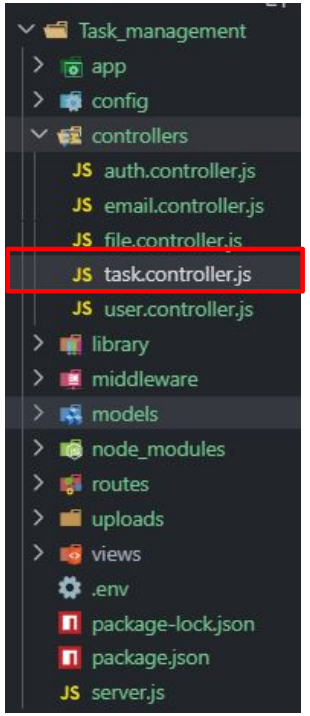
Task_management > controllers > JS task.controller.js > TaskController

```
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;

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

Node.js - Mongo.db

CRUD - Task

- Controlador de tareas
  - addTask



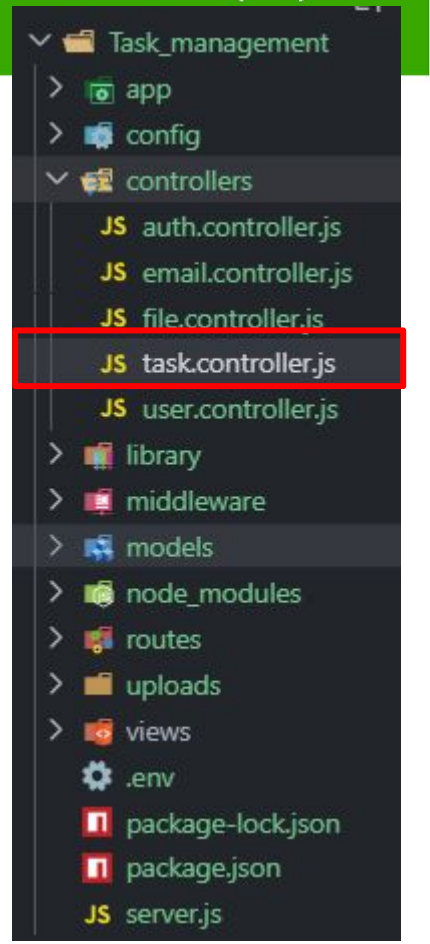

```
27
28          //Vailidate required fields
29          if (!title || !type || !dateStart || !dateEnd) {
30              return res.status(400).json({ error: 'Title, type, start date, and end date are required' });
31          }
32
33          // Convert date strings to Date objects
34          const 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
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          });
```

Node.js - Mongo.db

CRUD - Task

- Controlador de tareas
  - addTask

```
64
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      };
```

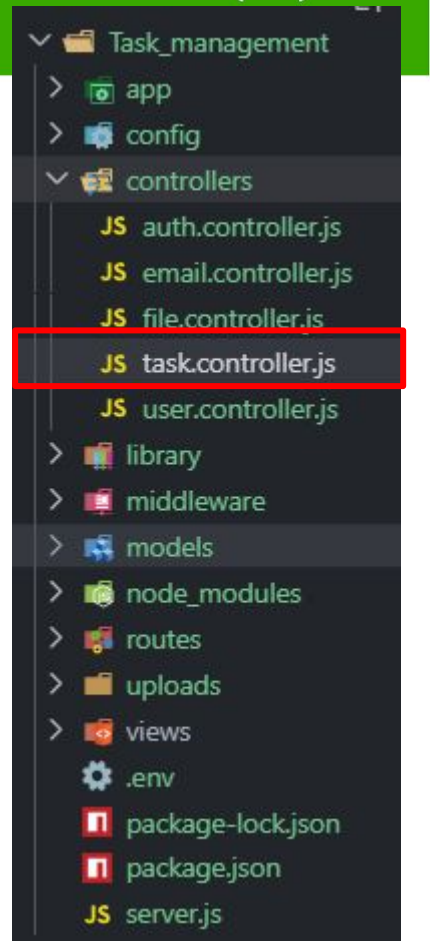


Node.js - Mongo.db

CRUD - Task

- Controlador de tareas
  - getTaskId

```
81      //show all tasks
82      async getTaskId(req, res) {
83          try {
84              const task = await TaskModel.findById(req.params.id);
85              if (!task) {
86                  return res.status(404).json({ error: 'Task not found' });
87              }
88
89              res.status(200).json(task);
90          } catch (error) {
91              res.status(500).json({ error: 'Error in get task details' });
92          }
93      };
94
```

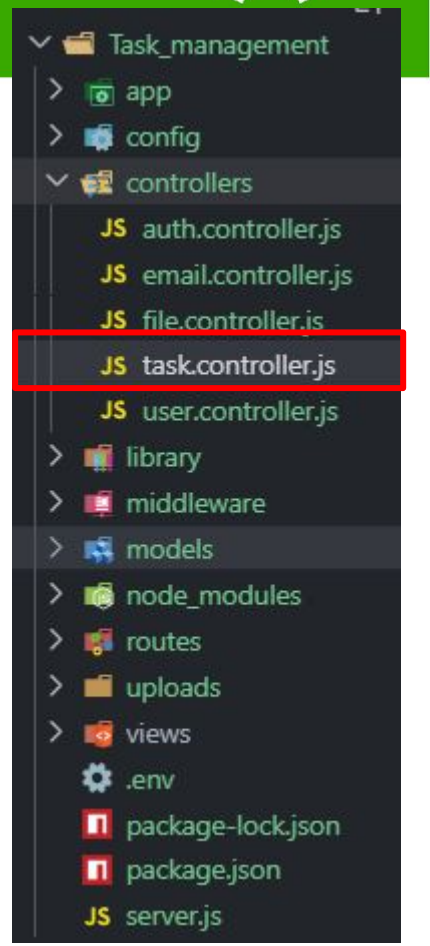


Node.js - Mongo.db

CRUD - Task

- Controlador de tareas
  - getTasks

```
95      //show all tasks
96      async getTasks(req, res) {
97          try {
98              const tasks = await TaskModel.find();
99              if (!tasks || tasks.length === 0) {
100                 return res.status(404).json({ error: 'No tasks found' });
101             }
102             res.status(200).json(tasks);
103         } catch (error) {
104             res.status(500).json({ error: 'Error in get task details' });
105         }
106     };
```

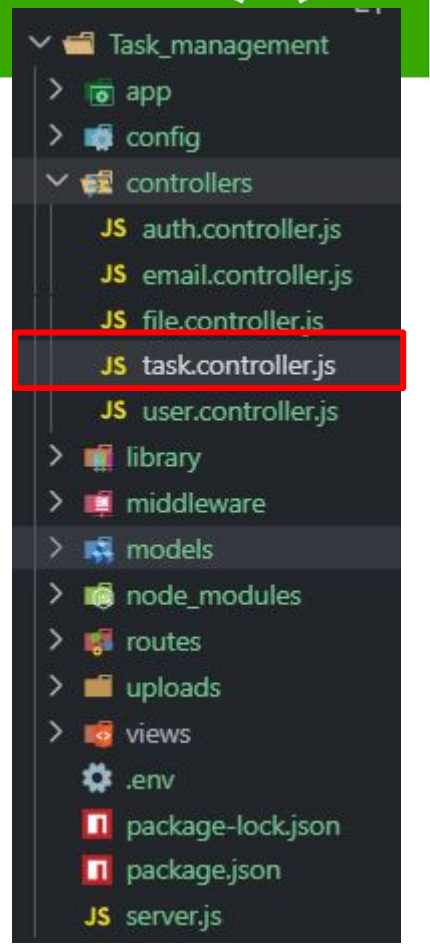


Node.js - Mongo.db

CRUD - Task

- Controlador de tareas
  - updateTask

```
108     // Update a task add array of managers
109     async updateTask(req, res) {
110         try {
111             const updates = req.body;
112 
113             // Convert date strings to Date objects
114             if (updates.dateStart) updates.dateStart = new Date(updates.dateStart);
115             if (updates.dateEnd) updates.dateEnd = new Date(updates.dateEnd);
116 
117             // Convert ids in managers to ObjectId
118             if (updates.managers) {
119                 updates.managers = updates.managers.map(inCharge => ({
120                     ...inCharge,
121                     userId: new mongoose.Types.ObjectId(inCharge.userId)
122                 }));
123             }
124 
125             const updateTask = await TaskModel.findByIdAndUpdate(
126                 req.params.id,
127                 { $set: updates },
128                 { new: true, runValidators: true }
129             );
130 
131             res.status(200).json(updateTask);
132         } catch (error) {
133             res.status(400).json({
134                 error: error.message,
135                 details: error.errors
136             });
137         }
138     };
139 
```

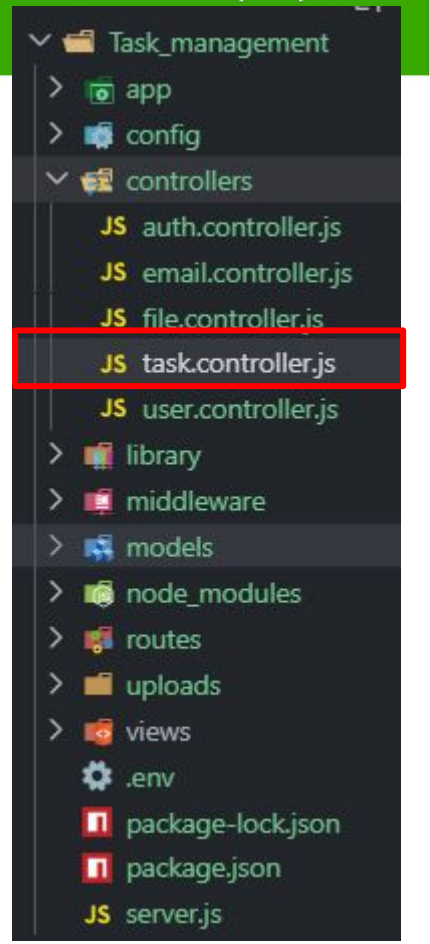


Node.js - Mongo.db

CRUD - Task

- Controlador de tareas
  - addComment

```
//Add comment to a task
async addComment(req, res) {
    try {
        const { userId, name, text } = req.body;
        // Validate required fields
        if (!userId || !name || !text) {
            return res.status(400).json({ error: 'User ID, name, and text are required' });
        }
        // Find the task and add the comment
        const task = await TaskModel.findById(req.params.id);

        if (!task) {
            return res.status(404).json({ error: 'Task not found' });
        }
        // Check if the comment already exists
        const existingComment = task.comments.find(comment => comment.userId.toString() === userId && comment.text === text);
        if (existingComment) {
            return res.status(400).json({ error: 'Comment already exists' });
        }
        // Add the comment
        task.comments.push({
            userId: userId,
            name: name,
            text: text,
            date: new Date()
        });
        // Save the updated task
        await task.save();

        // Return the updated task
        res.status(200).json(task);
    } catch (error) {
        res.status(400).json({ error: error.message });
    }
};
```

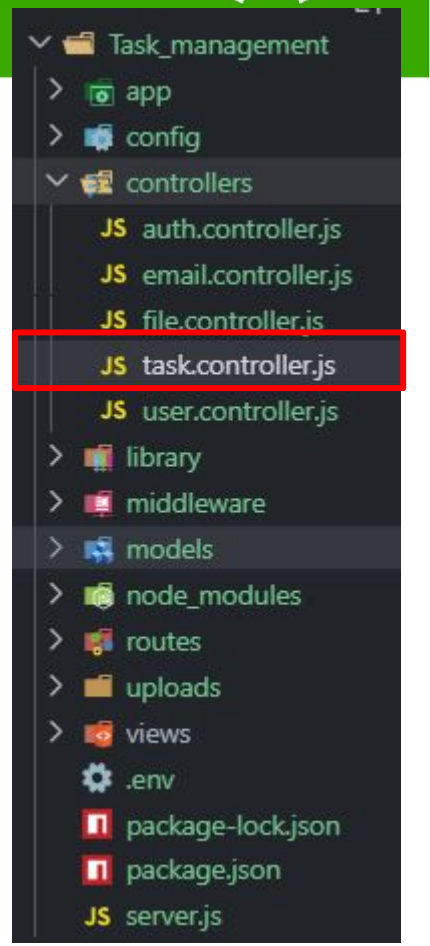


Node.js - Mongo.db

CRUD - Task

- Controlador de tareas
  - addFile

```
async addFile(req, res) {
    try {
        // Assuming multer uploaded the file and we have the info in req.file
        const { name, extension, type, size, url } =req.body;
        if (!name || !extension || !type || !size || !url) {
            return res.status(400).json({ error: 'File information is incomplete' });
        }
        const task = await TaskModel.findByIdAndUpdate(
            req.params.id,
            {
                $push: {
                    files: {
                        name: name,
                        url: url,
                        type: type,
                        extension: extension,
                        size: size
                    }
                }
            },
            { new: true }
        );

        res.status(200).json(task);
    } catch (error) {
        res.status(400).json({ error: error.message });
    }
};
}

export default new TaskController();
```

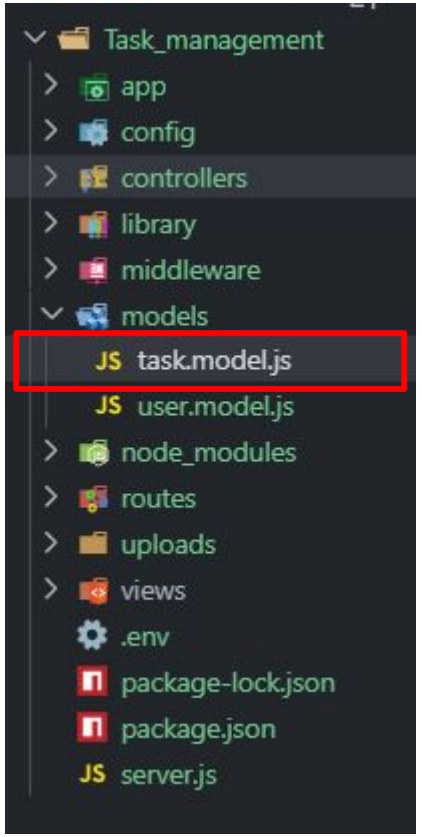


Node.js - Mongo.db

CRUD - Task

- Actualizar modelo
  - models ->task.model.js

```
// 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'] },
    extension: { type: String, required: true },
    size: { type: Number }, // in KB
    uploadDate: { type: Date, default: Date.now }
});
```



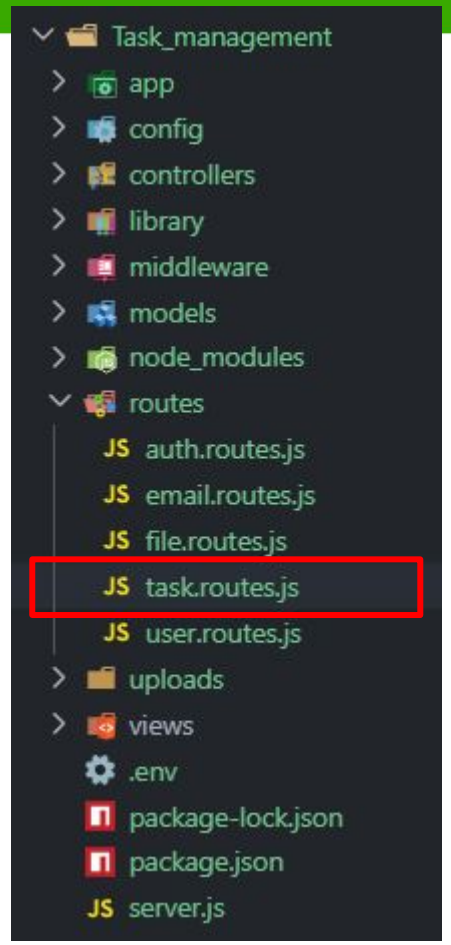
Node.js - Mongo.db

CRUD - Task

- Actualizar rutas
  - task.routes.js

Task_management > routes > JS task.routes.js > default

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

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

//Route for user by ID
router.route(`${name}/:id`)
    .get(TaskController.getTaskId)
    .put(TaskController.updateTask)

router.route(`${nameComment}/:id`).post(TaskController.addComment);

router.route(`${nameFiles}/:id`)
    .post(TaskController.addFile);

export default router;
```



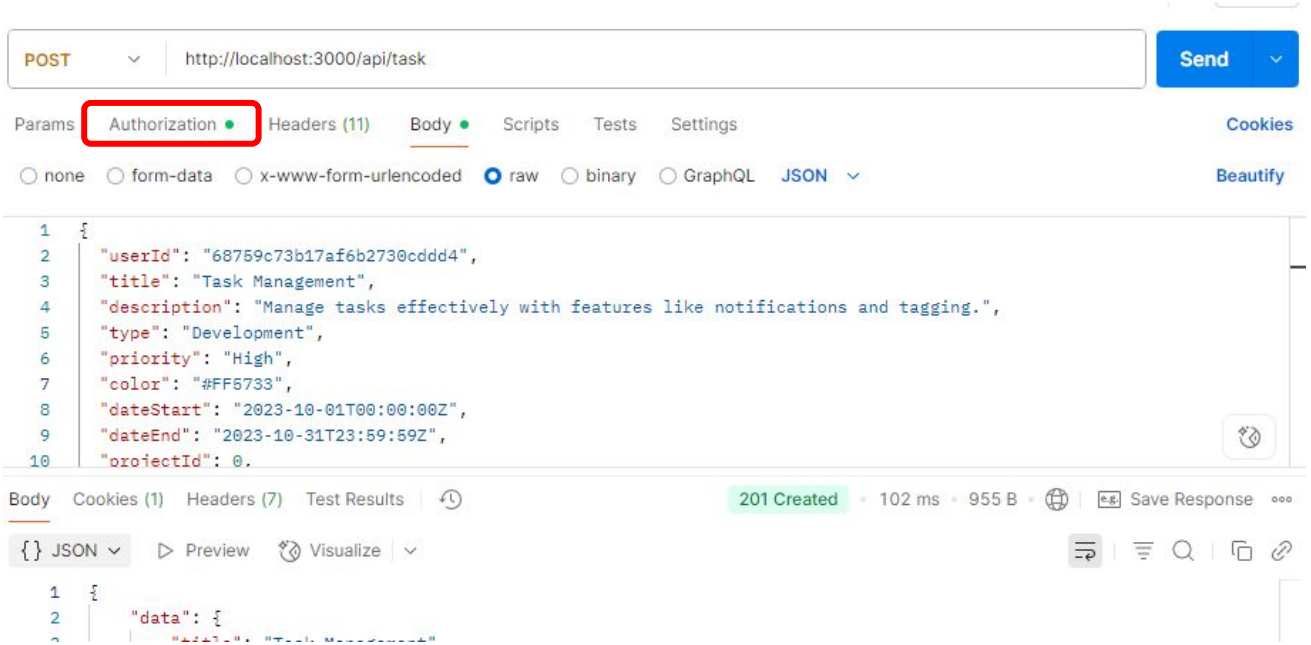
Node.js - Mongo.db

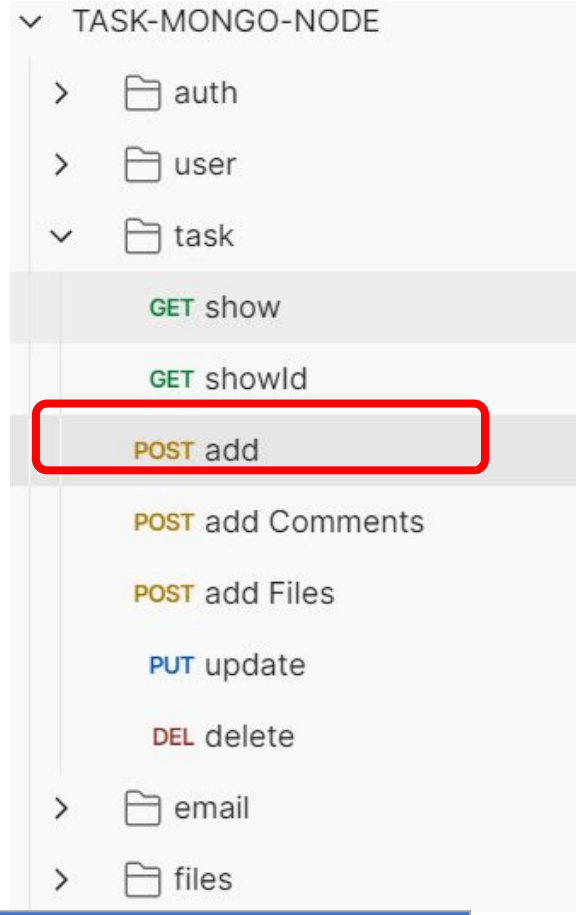
CRUD - Task

- Pruebas

TOKEN

DATOS





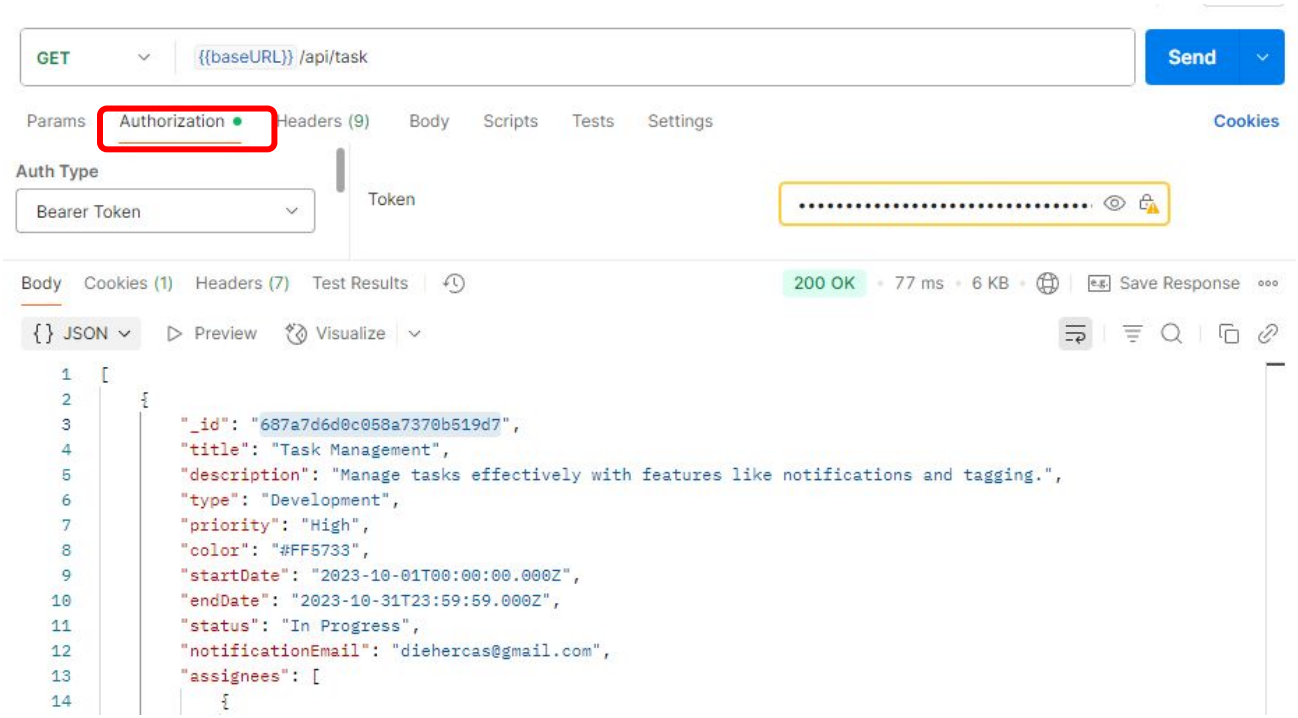
MÉTODO	RUTA	CONTROLADOR	MODELO
POST	http://localhost:3000/api/task	Tesk	Task

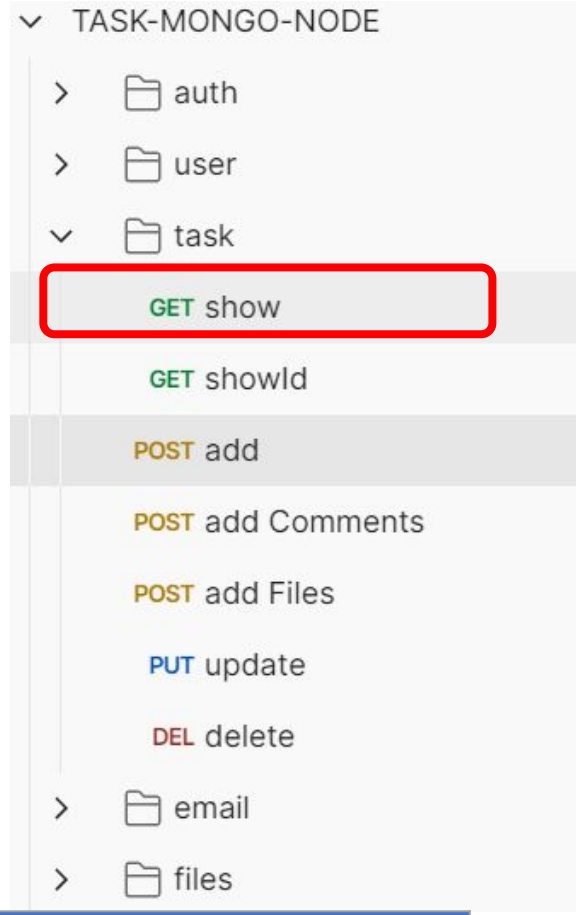
Node.js - Mongo.db

CRUD - Task

- Pruebas

TOKEN





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

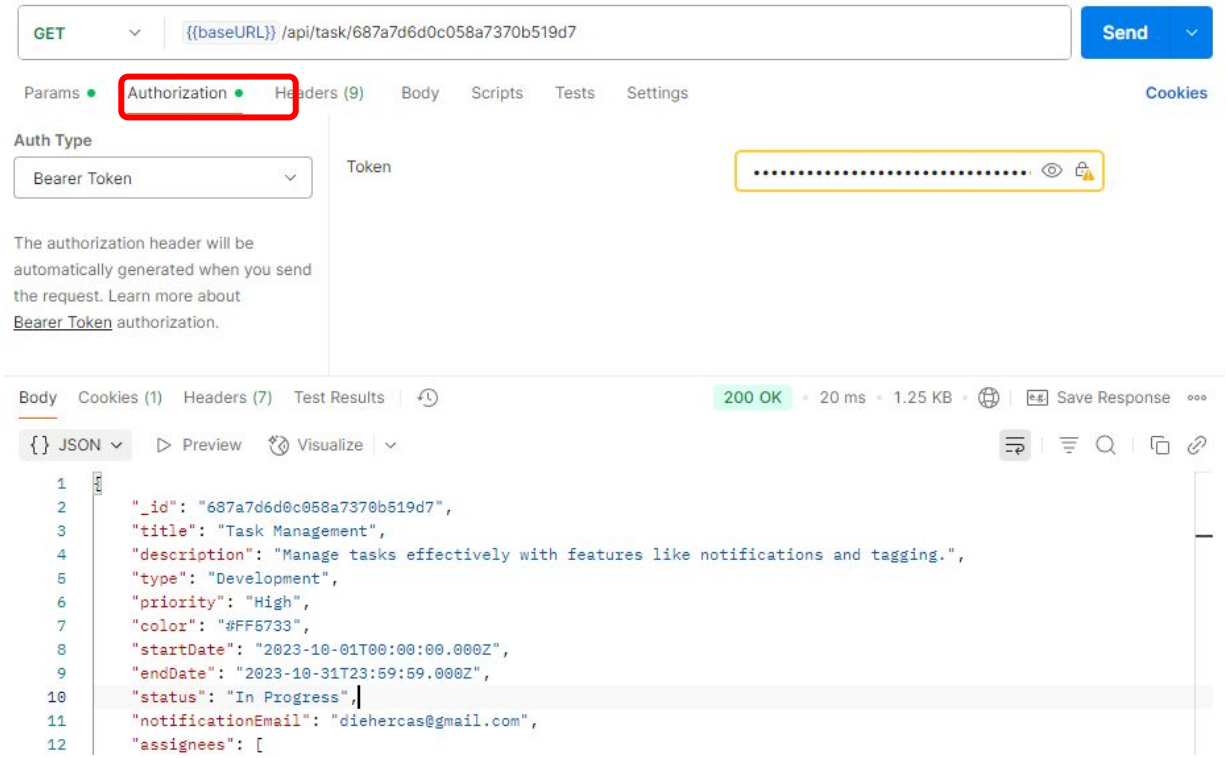
Node.js - Mongo.db

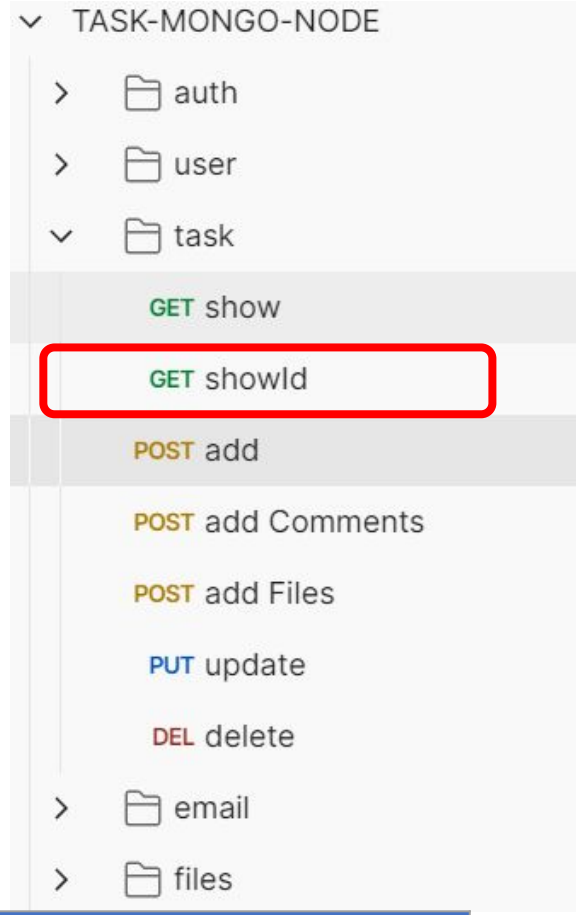
CRUD - Task

- Pruebas

ID

TOKEN



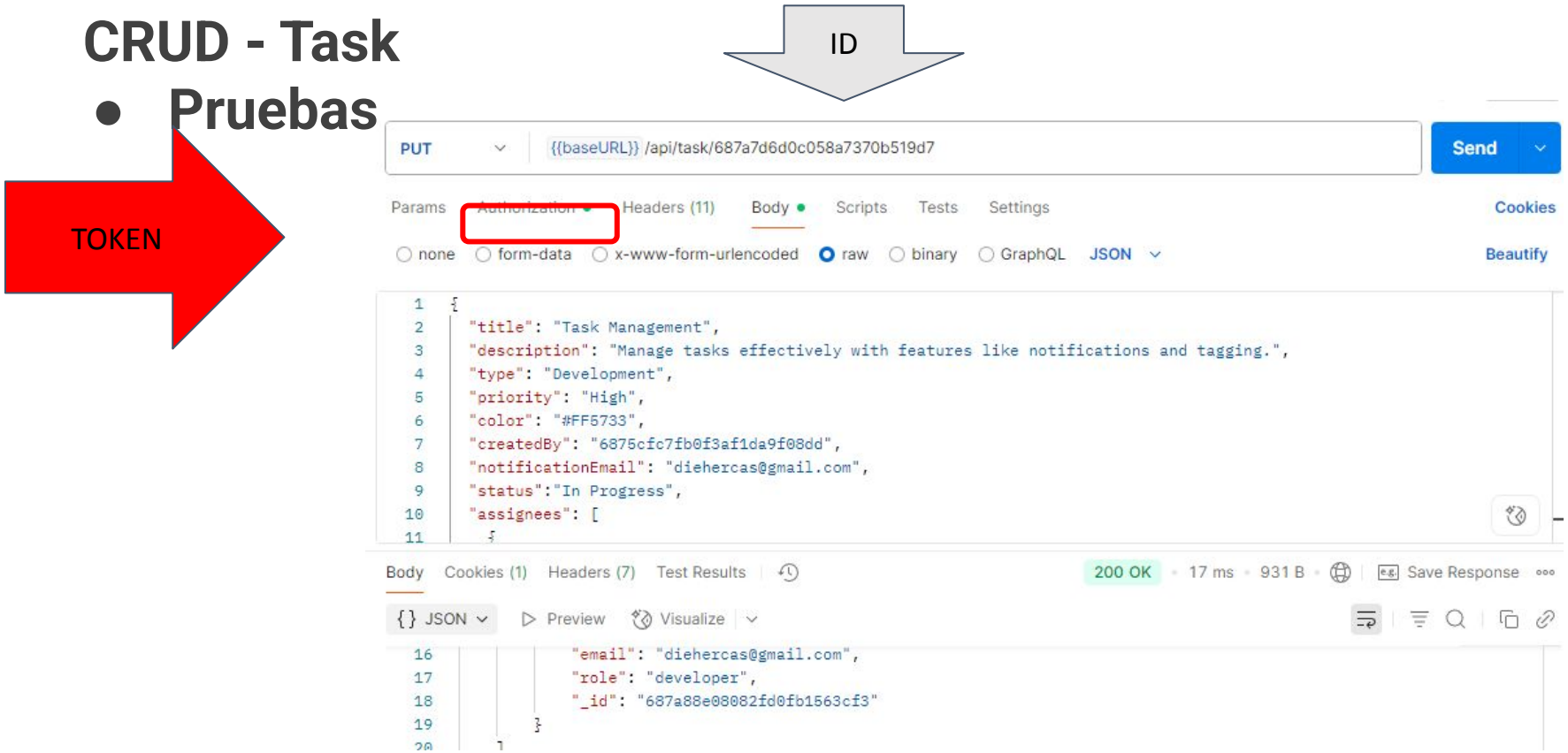


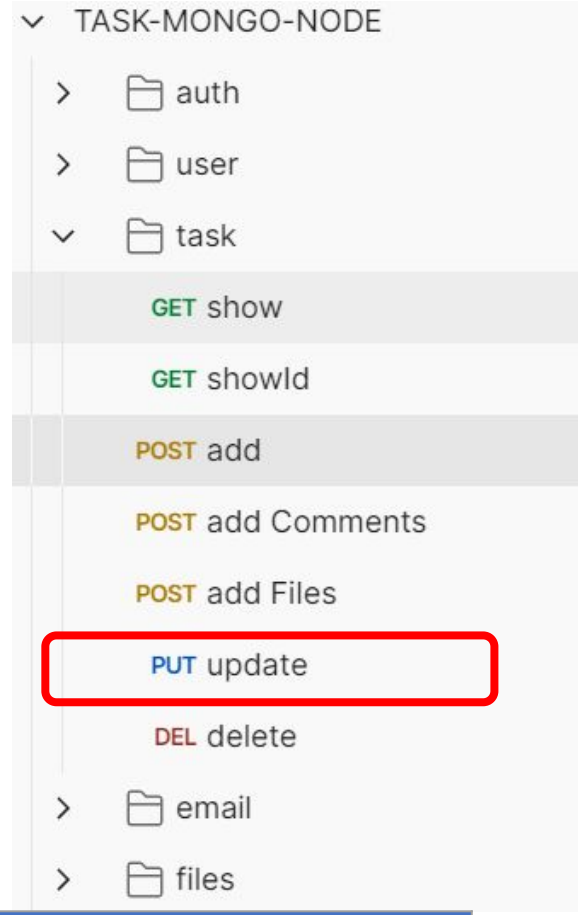
MÉTODO	RUTA	CONTROLADOR	MODELO
GET	http://localhost:3000/api/task:id	Tesk	Task

Node.js - Mongo.db

CRUD - Task

- Pruebas



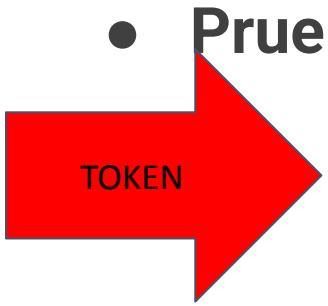


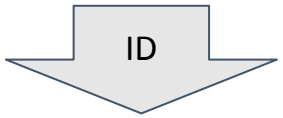
MÉTODO	RUTA	CONTROLADOR	MODELO
PUT	http://localhost:3000/api/task:id	Tesk	Task

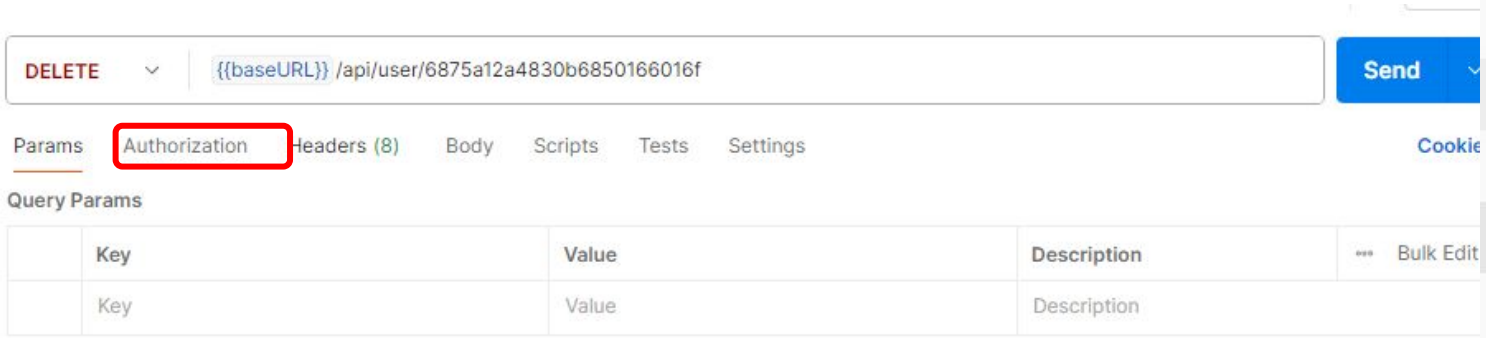
Node.js - Mongo.db

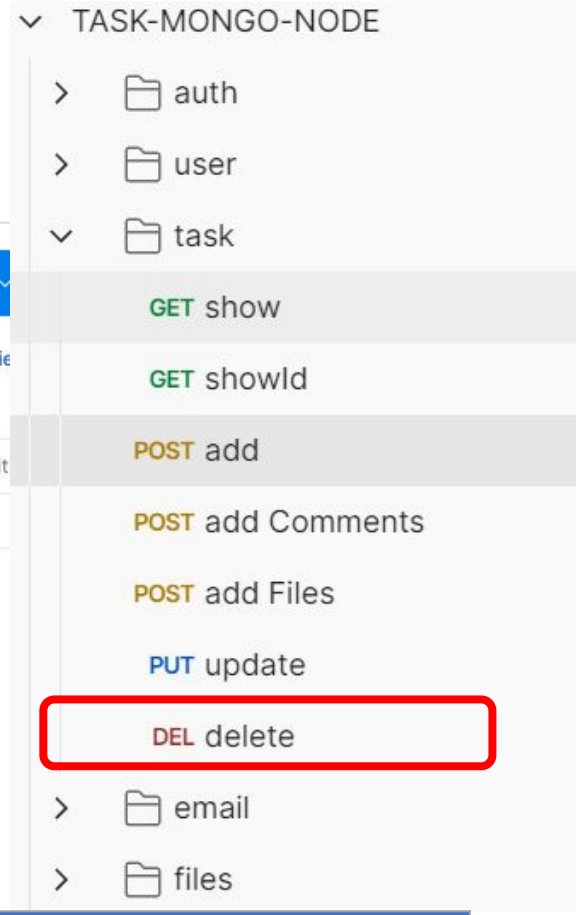
CRUD - Task

- Pruebas









MÉTODO	RUTA	CONTROLADOR	MODELO
DELETE	http://localhost:3000/api/task:id	Tesk	Task

Node.js - Mongo.db

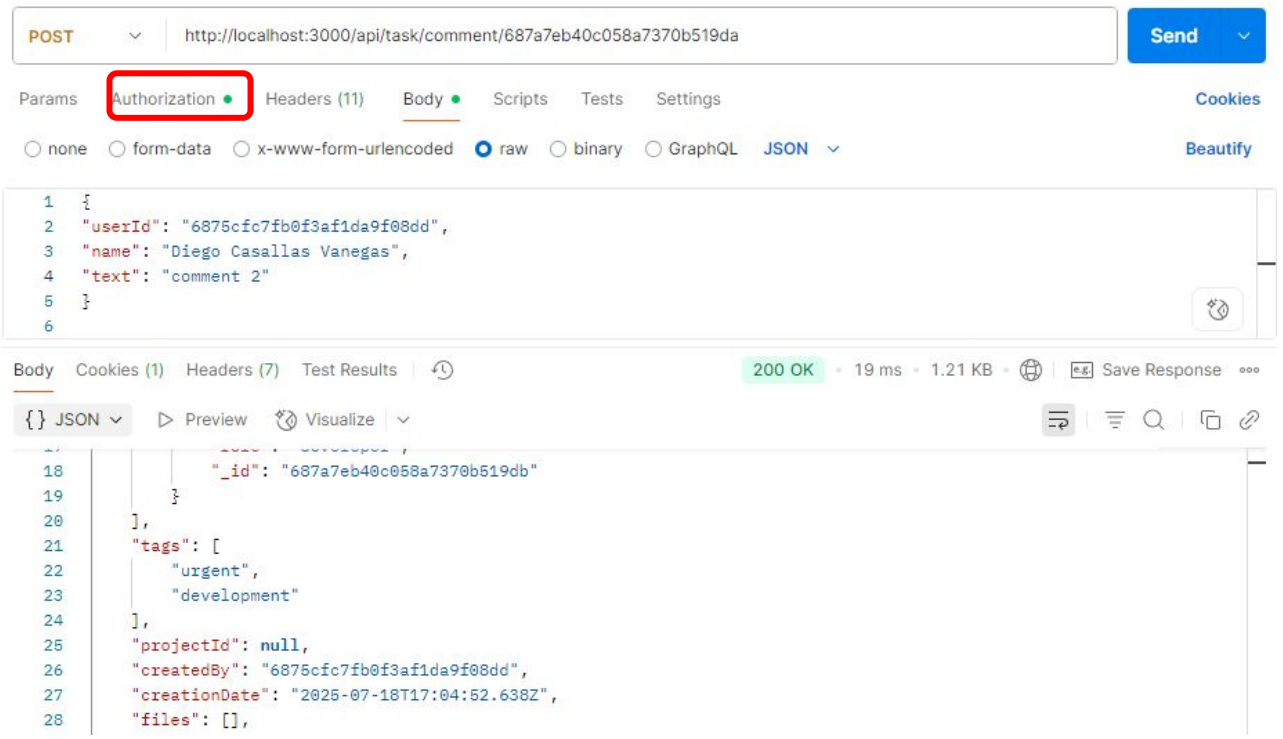
CRUD - Task

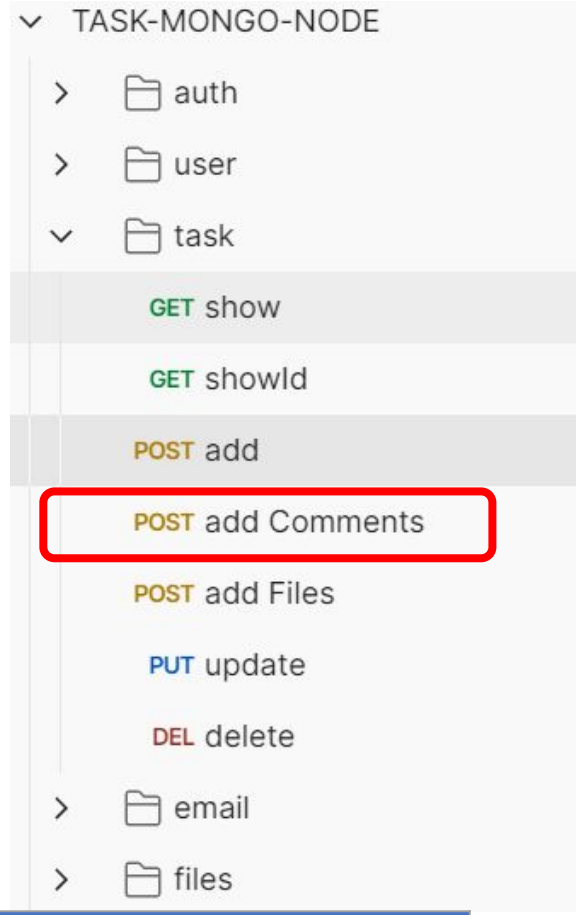
- Pruebas

TOKEN

DATOS

ID





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

Node.js - Mongo.db

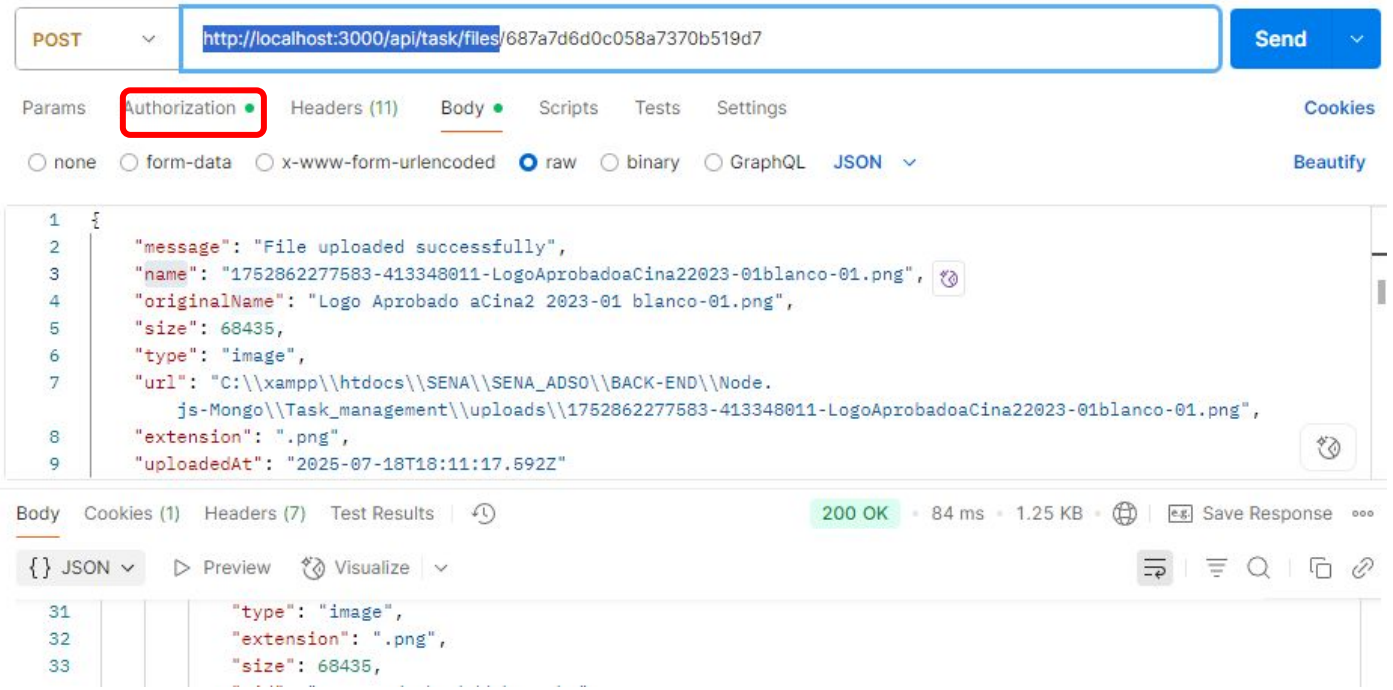
CRUD - Task

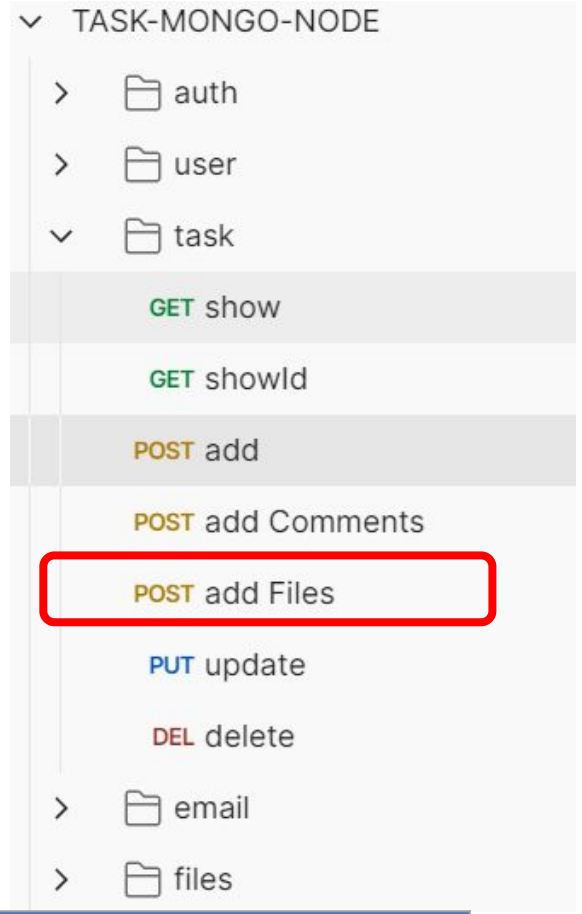
- Pruebas

ID

TOKEN

DATOS





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

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