**API Documentation**

**Base URL**<http://localhost:3000/api>

**Authentication**

All secured endpoints require a JWT token, provided in the header of each request:

Authorization: Bearer <token>

**Endpoints**

1. **Login**

* Method: POST
* Endpoint: /api/login
* Description: Authenticates users and returns a JWT for accessing secured endpoints.
* Request Body:

{

"username": "admin",

"password": "password"

}

* Response:

{

"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c"

}

1. **Get All Tasks**

Method: GET

Endpoint: /api/tasks

Description: Retrieves all tasks from the database.

* Response:

[

{

"id": 1,

"name": "Test Task",

"description": "A sample task description.",

"is\_completed": false

}

]

1. **Create Task**

* Method: POST
* Endpoint: /api/tasks
* Description: Adds a new task to the database.
* Request Body:

{

"name": "New Task",

"description": "Details about the new task.",

"is\_completed": false

}

* Response:

{

"message": "Task created successfully"

}

1. **Update Task**

* Method: PUT
* Endpoint: /api/tasks/:id
* Description: Updates an existing task identified by its ID.
* Request Body:

{

"name": "Updated Task",

"description": "Updated description.",

"is\_completed": true

}

* Response:

{

"message": "Task updated successfully"

}

1. **Delete Task**

* Method: DELETE
* Endpoint: /api/tasks/:id
* Description: Deletes a task identified by its ID.
* Response:

{

"message": "Task deleted successfully"

}

**Usage Examples**

To interact with the API, you can use tools like Postman or CURL. Here’s how you might call the **Create Task** endpoint using Postman:

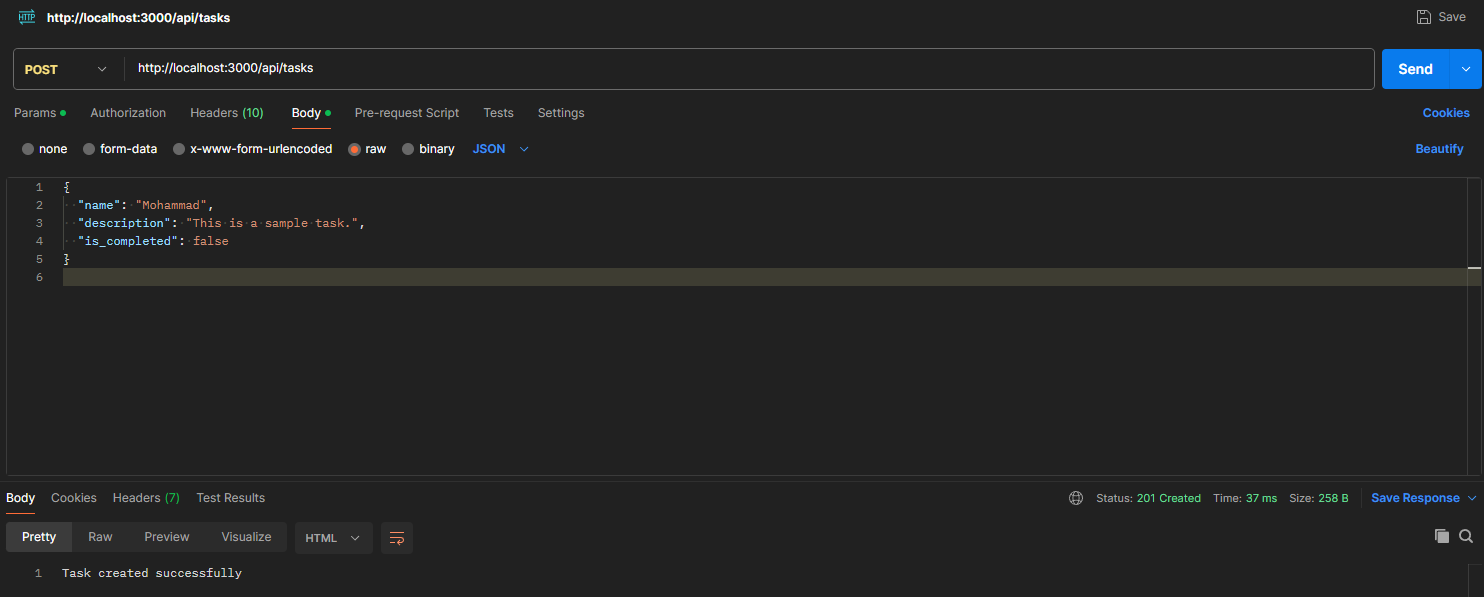


Figure 1 Image demonstrating the usage of the Create Task API

**Error Handling**

The API uses conventional HTTP response codes to indicate success or failure of an API request. In general:

* 200 OK - The request was successful.
* 400 Bad Request - The request was unacceptable, often due to missing a required parameter.
* 401 Unauthorized - No valid JWT token was provided.
* 404 Not Found - The requested resource doesn’t exist.
* 500 Internal Server Error - We had a problem with our server. Try again later.

**Express Status Dashboard**

* The Express Status Dashboard provides a real-time overview of server health and metrics.
* This dashboard is accessible at: <http://localhost:3300/status>
* This dashboard offers insights into various metrics such as CPU usage, memory usage, response times, and request rates, helping to monitor the application's health and performance efficiently.

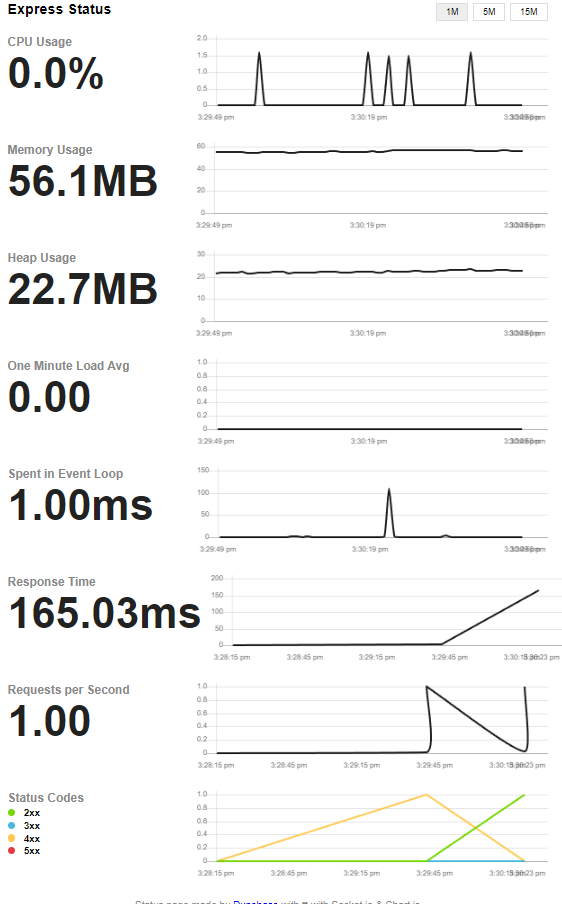


Figure 2 Image showing the Express Status data