

TASK: Build a small backend using Node.js (Express) and PostgreSQL that performs CRUD operations.

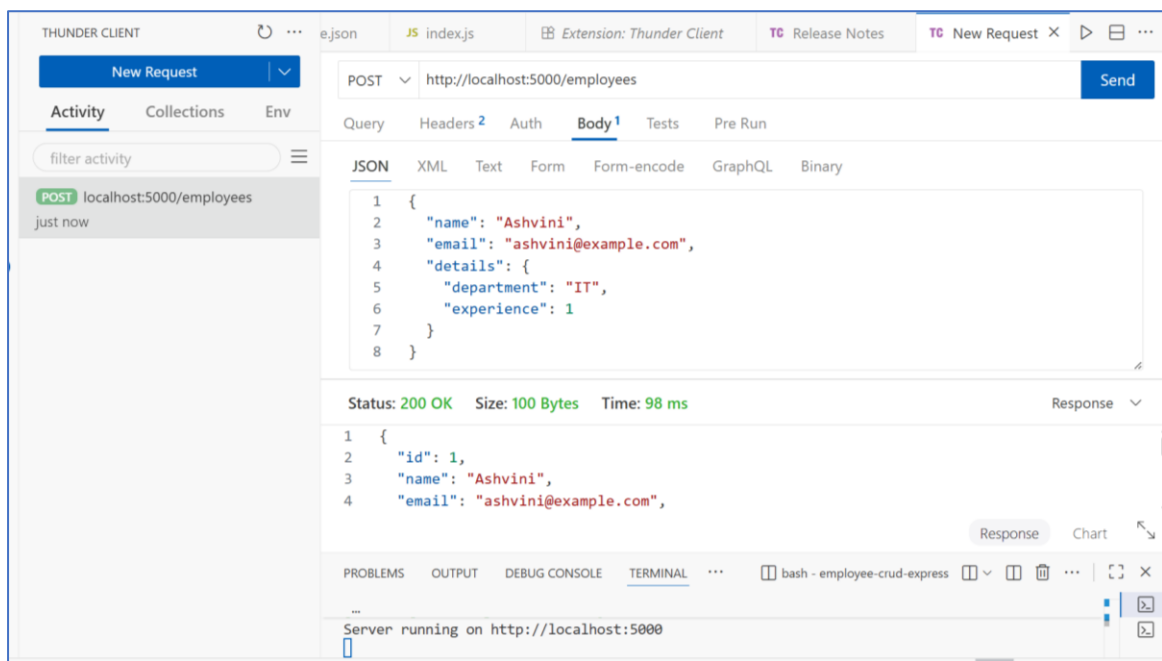
OUTPUT:

- **POST /employees → Add a new employee:**

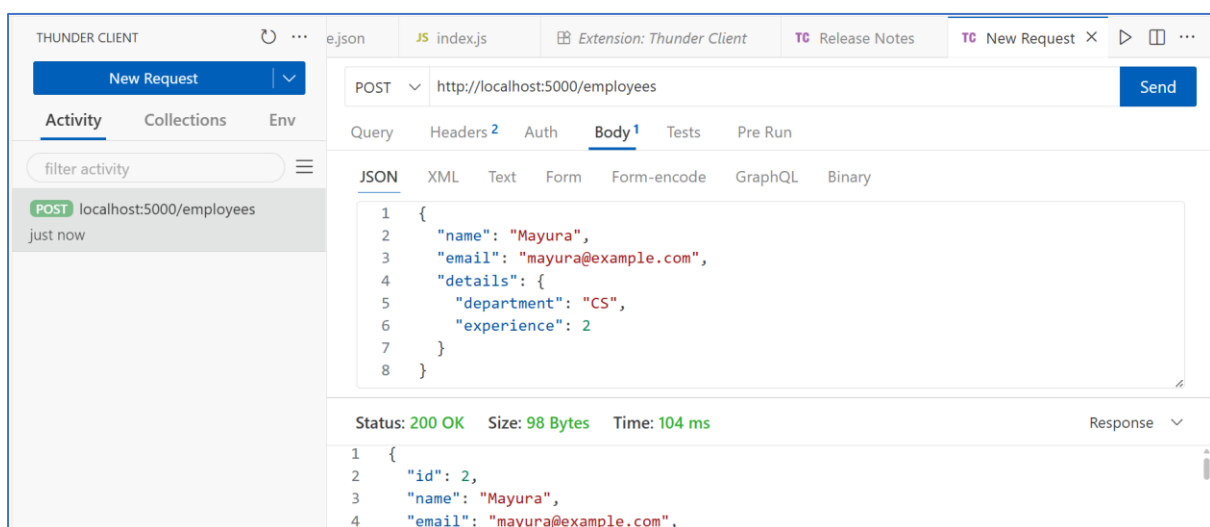
Tool, I used to send POST data is Thunder Client – VS Code extension

3 records are added using POST.

1) Ashvini user data



2) Mayura user data



3) Tanya user data

The screenshot shows the Thunder Client interface. The top bar includes the application name 'THUNDER CLIENT', a refresh button, and tabs for 'ejson', 'JS index.js', and 'Extension: Thunder Client'. On the right, there are links for 'TC Release Notes' and 'TC New Request'. The main panel is divided into two sections. The top section is for creating a new request, with a dropdown menu set to 'POST' and the URL 'http://localhost:5000/employees'. The bottom section shows the request details, including the 'Body' tab with a JSON payload:

```
{  "name": "Tanya",  "email": "tanya@example.com",  "details": {    "department": "MMS",    "experience": 3  }}
```

. The status bar at the bottom indicates 'Status: 200 OK', 'Size: 97 Bytes', and 'Time: 101 ms'. The response is also shown in JSON format:

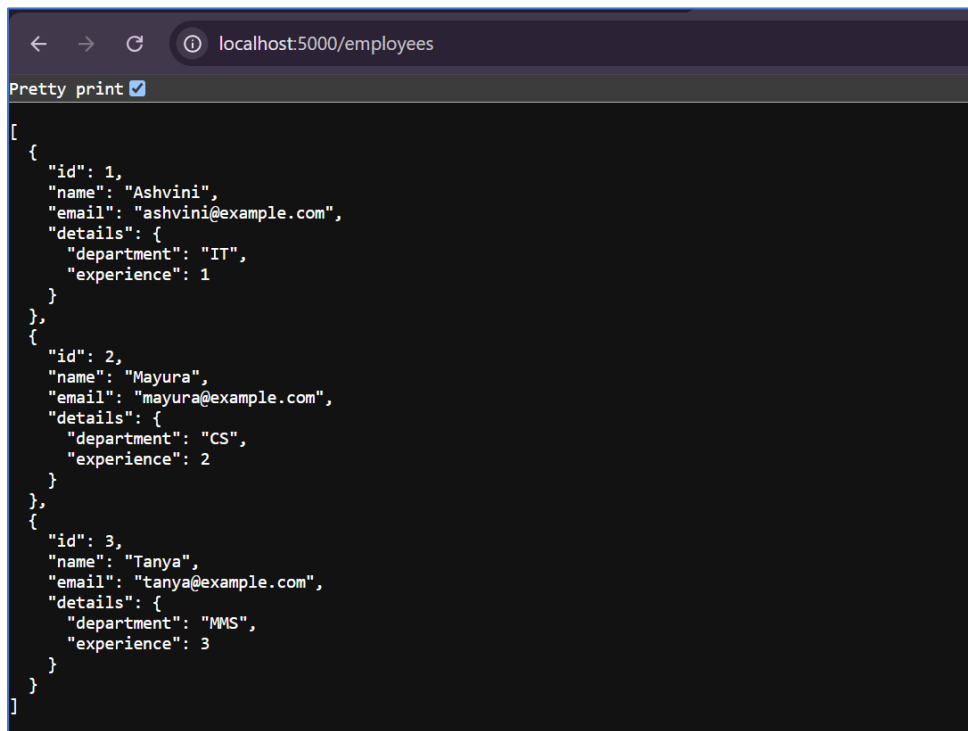
```
{  "id": 3,  "name": "Tanya",  "email": "tanya@example.com",  "details": {}}
```

- **GET /employees → Fetch all employees:**

The screenshot shows the Thunder Client interface. The top bar includes the application name 'THUNDER CLIENT', a refresh button, and tabs for 'ejson', 'JS index.js', and 'Extension: Thunder Client'. On the right, there are links for 'TC Release Notes' and 'TC New Request'. The main panel is divided into two sections. The top section is for creating a new request, with a dropdown menu set to 'GET' and the URL 'localhost:5000/employees'. The bottom section shows the request details, including the 'Body' tab with a JSON payload:

```
{  "id": 1,  "name": "Ashvini",  "email": "ashvini@example.com",  "details": {    "department": "IT",    "experience": 1  }}, {  "id": 2,  "name": "Mayura",  "email": "mayura@example.com",  "details": {    "department": "CS",    "experience": 2  }}, {  "id": 3,  "name": "Tanya",  "email": "tanya@example.com",  "details": {    "department": "MMS",
```

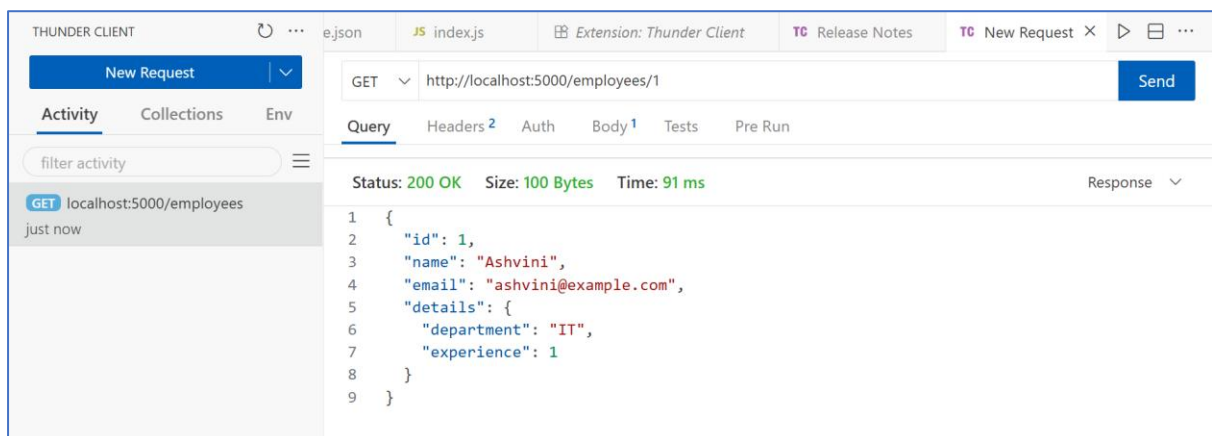
Accessing From Chrome:

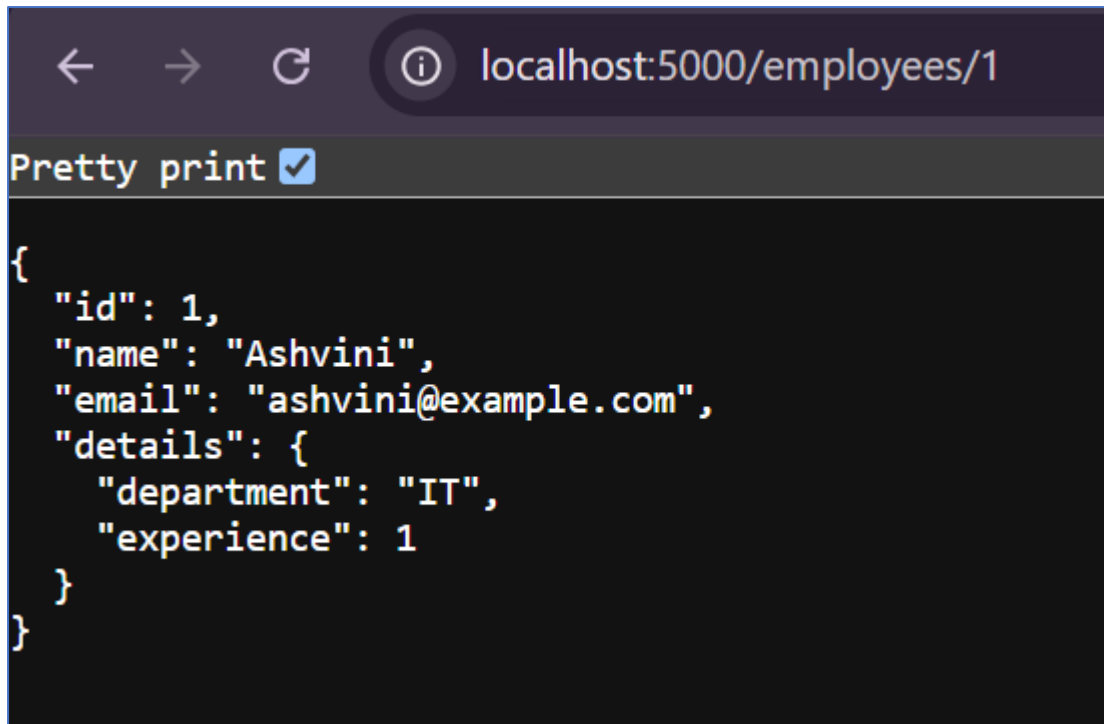


A screenshot of a web browser window with the address bar showing `localhost:5000/employees`. The page content displays a JSON array of three employee objects, formatted with 'Pretty print' checked. The JSON data is as follows:

```
[
  {
    "id": 1,
    "name": "Ashvini",
    "email": "ashvini@example.com",
    "details": {
      "department": "IT",
      "experience": 1
    }
  },
  {
    "id": 2,
    "name": "Mayura",
    "email": "mayura@example.com",
    "details": {
      "department": "CS",
      "experience": 2
    }
  },
  {
    "id": 3,
    "name": "Tanya",
    "email": "tanya@example.com",
    "details": {
      "department": "MMS",
      "experience": 3
    }
  }
]
```

- **GET /employees/:id → Fetch one employee by id:**





```
localhost:5000/employees/1

Pretty print ☒

{
  "id": 1,
  "name": "Ashvini",
  "email": "ashvini@example.com",
  "details": {
    "department": "IT",
    "experience": 1
  }
}
```

- **PUT /employees/:id → Update an employee's details:**

>> Update user with id 2

```
{
  "name": "Samarth Updated",
  "email": "sam@example.com",
  "details": {
    "department": "CS",
    "experience": 2
  }
}
```

New Request ▼

PUT ▼ http://localhost:5000/employees/2 Send

Activity Collections Env

filter activity ☰

PUT localhost:5000/employees just now

Query Headers **2** Auth **Body** **1** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

```
1 {
2   "name": "Samarth Updated",
3   "email": "sam@example.com",
4   "details": {
5     "department": "CS",
6     "experience": 2
7   }
8 }
```

Status: 200 OK Size: 104 Bytes Time: 121 ms Response ▼

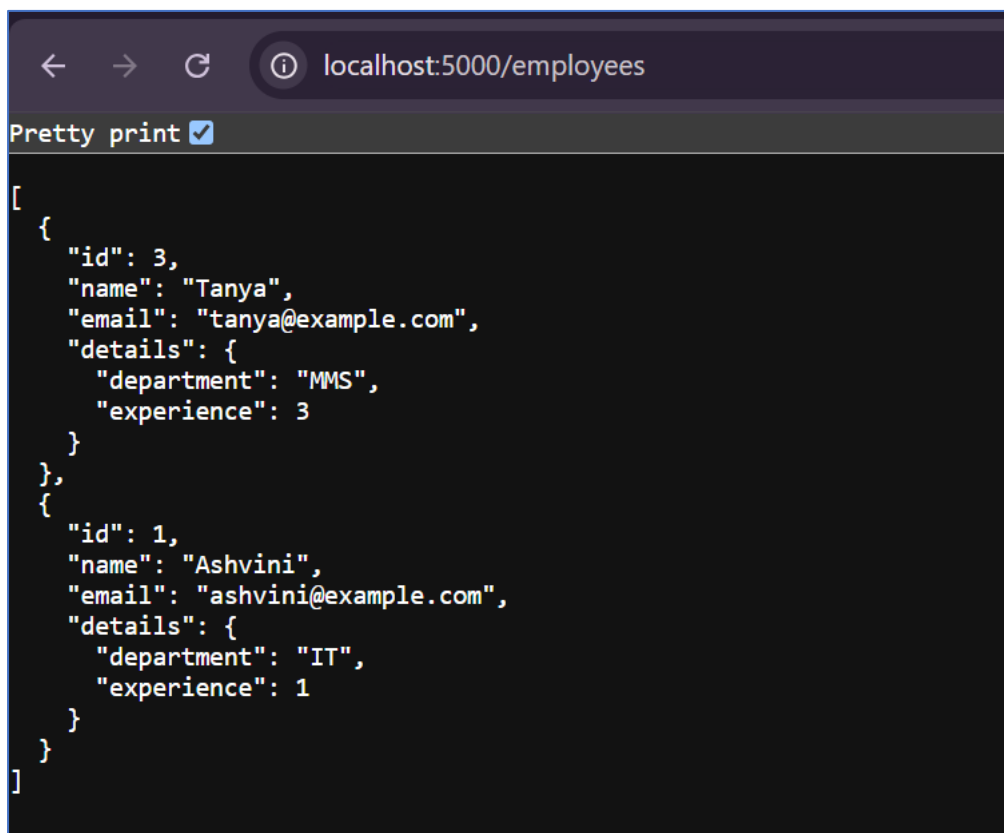
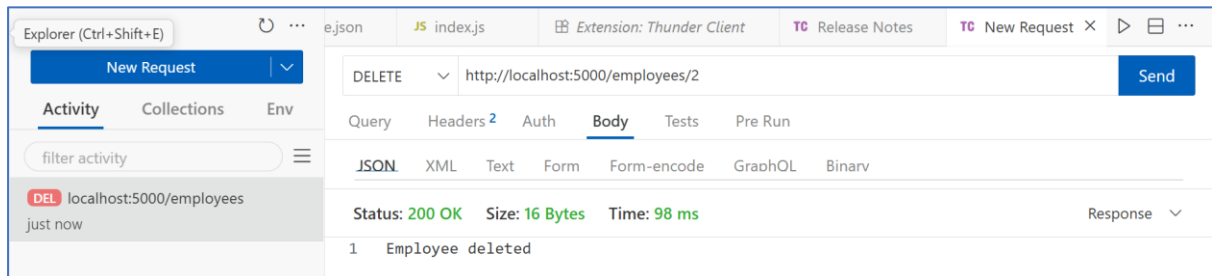
```
1 {
2   "id": 2,
3   "name": "Samarth Updated",
4   "email": "sam@example.com",
5   "details": {
6     "department": "CS",
7     "experience": 2
8   }
9 }
```

← → ↻ ⓘ localhost:5000/employees

Pretty print ☒

```
[
  {
    "id": 3,
    "name": "Tanya",
    "email": "tanya@example.com",
    "details": {
      "department": "MMS",
      "experience": 3
    }
  },
  {
    "id": 2,
    "name": "Samarth Updated",
    "email": "sam@example.com",
    "details": {
      "department": "CS",
      "experience": 2
    }
  },
  {
    "id": 1,
    "name": "Ashvini",
    "email": "ashvini@example.com",
    "details": {
      "department": "IT",
      "experience": 1
    }
  }
]
```

- **DELETE /employees/:id → Delete an employee:**
-



- **In PostgreSQL Final Data Left:**

pgAdmin 4

File Object Tools Help

Object Explorer

- > climatrack_db
- > employees_db
 - > Casts
 - > Catalogs
 - > Event Triggers
 - > Extensions
 - > Foreign Data Wrappers
 - > Languages
 - > Publications
 - > Schemas (1)
 - > public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > 1.3 Sequences
 - > Tables (1)
 - > employees

Dashboard x Properties x SQL x employees_db/postgres@PostgreSQL 16* x employees_db/po... x

employees_db/postgres@PostgreSQL 16

Query Query History

```
1
2
3 CREATE TABLE employees (
4     id SERIAL PRIMARY KEY,
5     name VARCHAR(100),
6     email VARCHAR(100),
7     details JSON
8 );
9
10 select * from employees;
```

Data Output Messages Notifications

	id [PK] integer	name character varying (100)	email character varying (100)	details json
1	3	Tanya	tanya@example.com	{"department":"MMS","experience":3}
2	1	Ashvini	ashvini@example.com	{"department":"IT","experience":1}

Data Output Messages Notifications

	id [PK] integer	name character varying (100)	email character varying (100)	details json
1	3	Tanya	tanya@example.com	{"department":"MMS","experience":3}
2	1	Ashvini	ashvini@example.com	{"department":"IT","experience":1}