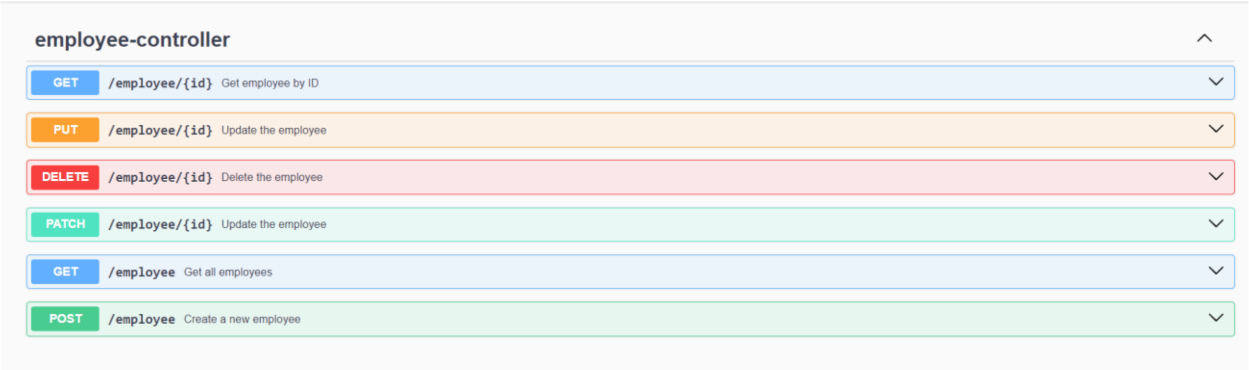
Develop an API to create, read, update and delete record



employee-controller

GET /employee/{id} Get employee by ID

PUT /employee/{id} Update the employee

DELETE /employee/{id} Delete the employee

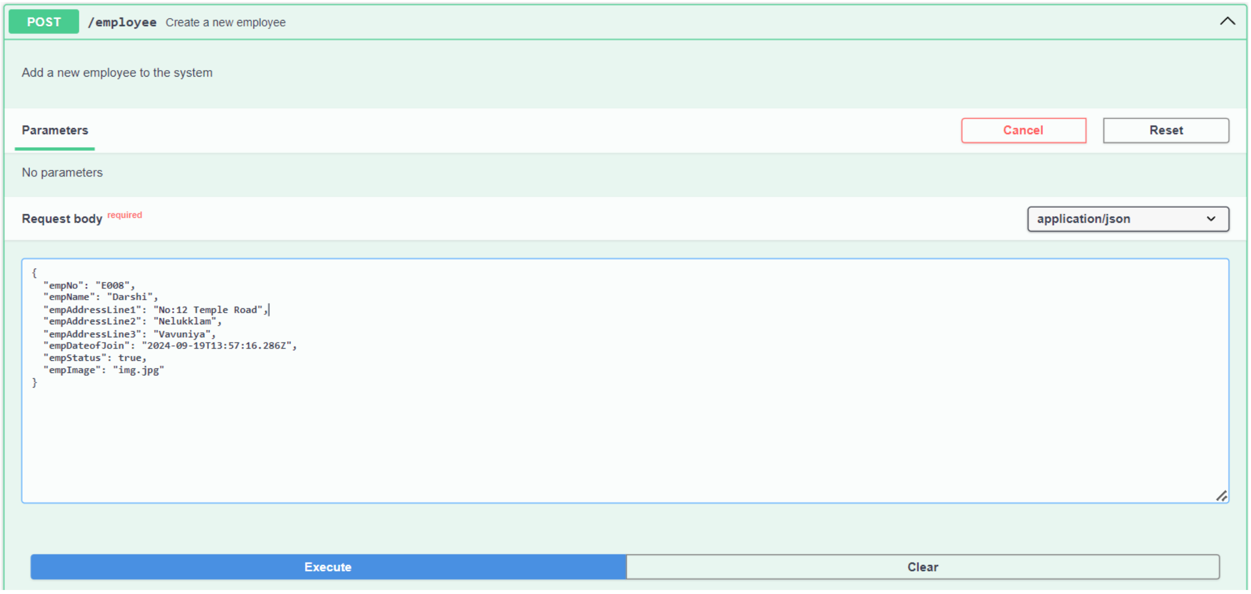
PATCH /employee/{id} Update the employee

GET /employee Get all employees

POST /employee Create a new employee

1. Create (HTTP POST):

The Create operation is used to add a new employee to the collection. The client sends a POST request to the API endpoint `/employees` with the employee details in the request body. The server validates the input and creates a new employee record in the database.

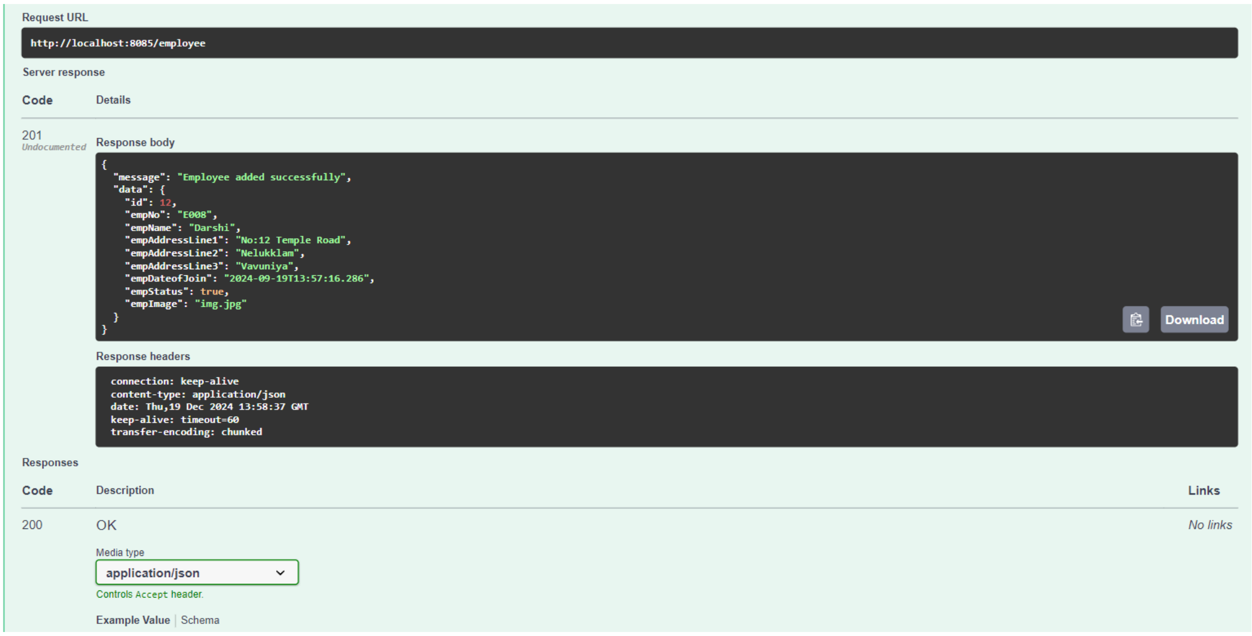


Add a new employee to the system

No parameters

Request body required

```
{
  "empNo": "E008",
  "empName": "Darshi",
  "empAddressLine1": "No:12 Temple Road",
  "empAddressLine2": "Nelukklam",
  "empAddressLine3": "Vavuniya",
  "empDateofJoin": "2024-09-19T13:57:16.286Z",
  "empStatus": true,
  "empImage": "img.jpg"
}
```



Request URL

```
http://localhost:8085/employee
```

Server response

Code	Details
201 *Undocumented*	Response body

```
{
  "message": "Employee added successfully",
  "data": {
    "id": 12,
    "empNo": "E008",
    "empName": "Darshi",
    "empAddressLine1": "No:12 Temple Road",
    "empAddressLine2": "Nelukklam",
    "empAddressLine3": "Vavuniya",
    "empDateofJoin": "2024-09-19T13:57:16.286",
    "empStatus": true,
    "empImage": "img.jpg"
  }
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Thu,19 Dec 2024 13:58:37 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

Media type: application/json

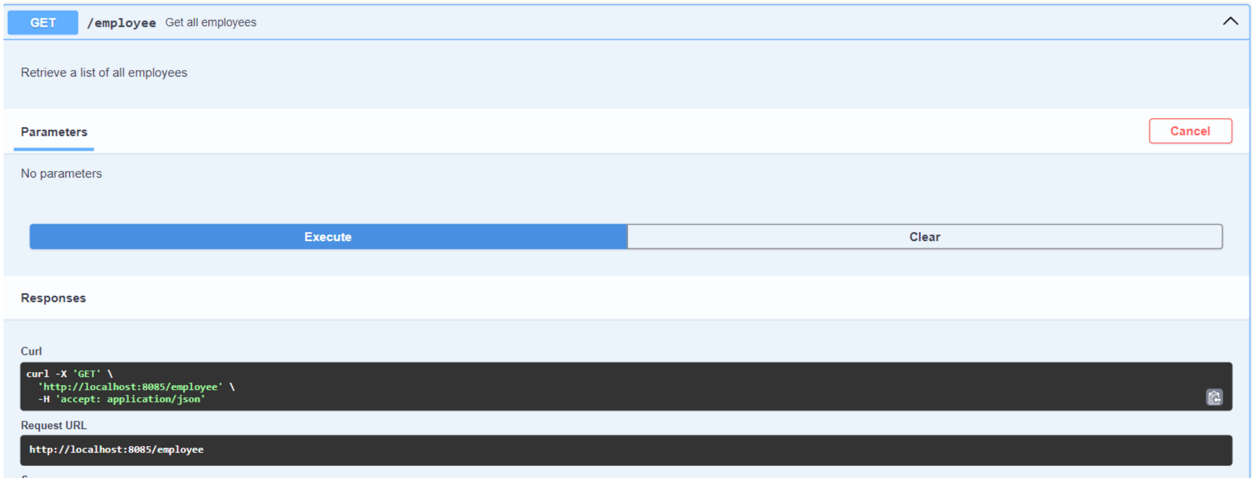
Controls Accept header.

2. Read (HTTP GET):

The Read operation retrieves information about one or more employee from the collection. There are two common scenarios:

a. Retrieve all employees:

The client sends a GET request to the `/employees` endpoint, and the server responds with a list of all employees available.



Retrieve a list of all employees

No parameters

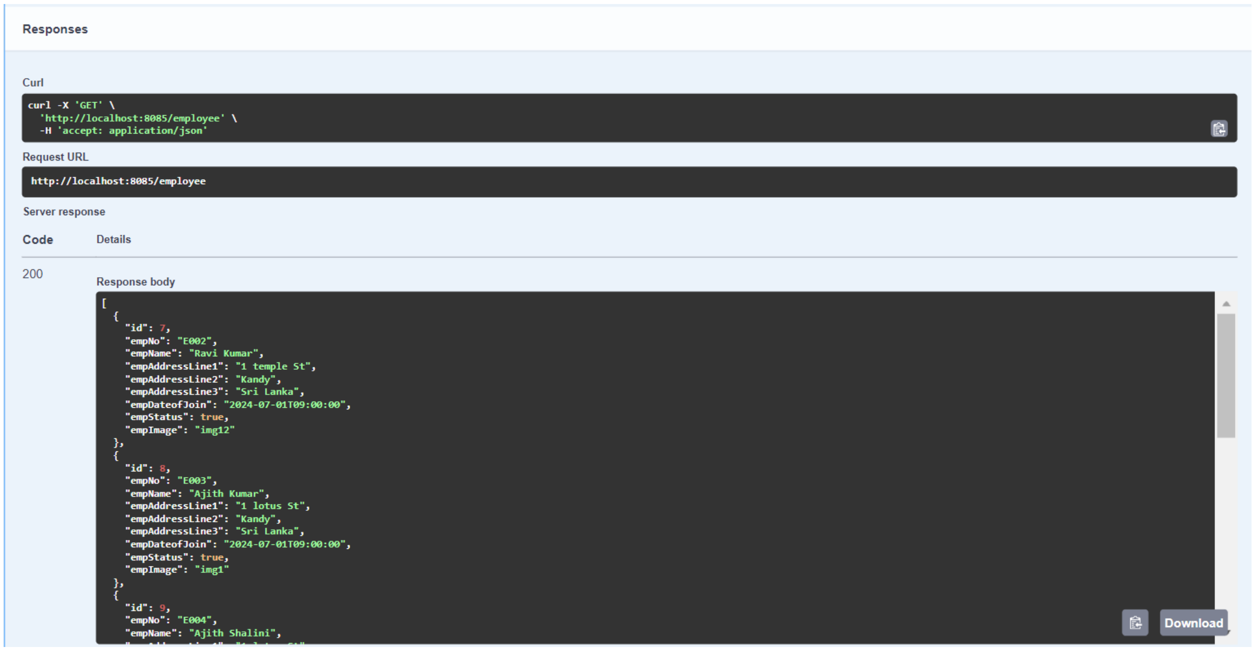
Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8085/employee' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8085/employee
```


Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8085/employee' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8085/employee
```

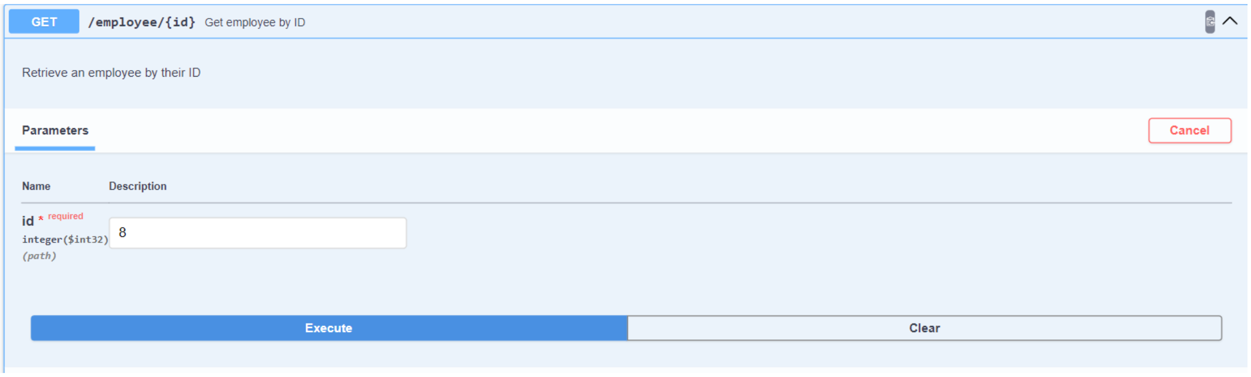
Server response

Code	Details
200	Response body

```
[
  {
    "id": 7,
    "empNo": "E002",
    "empName": "Ravi Kumar",
    "empAddressLine1": "1 temple St",
    "empAddressLine2": "Kandy",
    "empAddressLine3": "Sri Lanka",
    "empDateofJoin": "2024-07-01T09:00:00",
    "empStatus": true,
    "empImage": "img12"
  },
  {
    "id": 8,
    "empNo": "E003",
    "empName": "Ajith Kumar",
    "empAddressLine1": "1 lotus St",
    "empAddressLine2": "Kandy",
    "empAddressLine3": "Sri Lanka",
    "empDateofJoin": "2024-07-01T09:00:00",
    "empStatus": true,
    "empImage": "img1"
  },
  {
    "id": 9,
    "empNo": "E004",
    "empName": "Ajith Shalini",
```

b. Retrieve a specific employee:

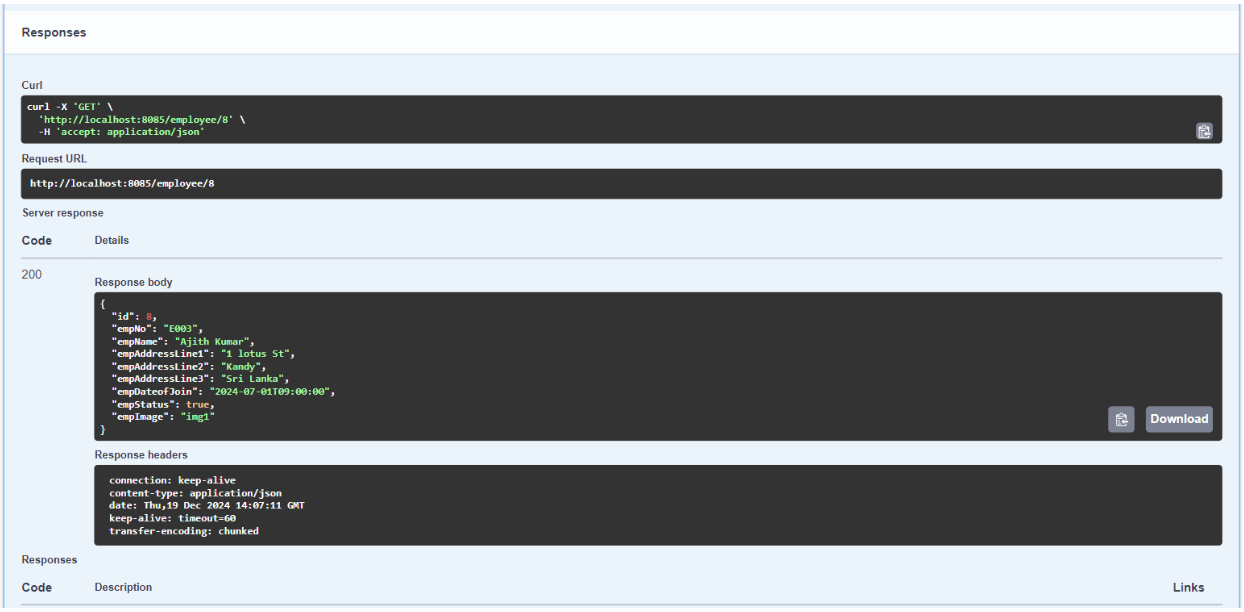
The client sends a GET request to the `/employees/{id}` endpoint, where `{id}` represents the unique identifier of the employee. The server retrieves the corresponding employee record and returns it to the client.



`/employee/{id}` Get employee by ID

Retrieve an employee by their ID

Name	Description
id * required integer($int32) *(path)*	8


Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8085/employee/8' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8085/employee/8
```

Server response

Code	Details
200	Response body

```
{
  "id": 8,
  "empNo": "E003",
  "empName": "Ajith Kumar",
  "empAddressLine1": "1 lotus St",
  "empAddressLine2": "Kandy",
  "empAddressLine3": "Sri Lanka",
  "empDateofJoin": "2024-07-01T09:00:00",
  "empStatus": true,
  "empImage": "img1"
}
```

Response headers

```
 connection: keep-alive
 content-type: application/json
 date: Thu,19 Dec 2024 14:07:11 GMT
 keep-alive: timeout=60
 transfer-encoding: chunked
```

Responses

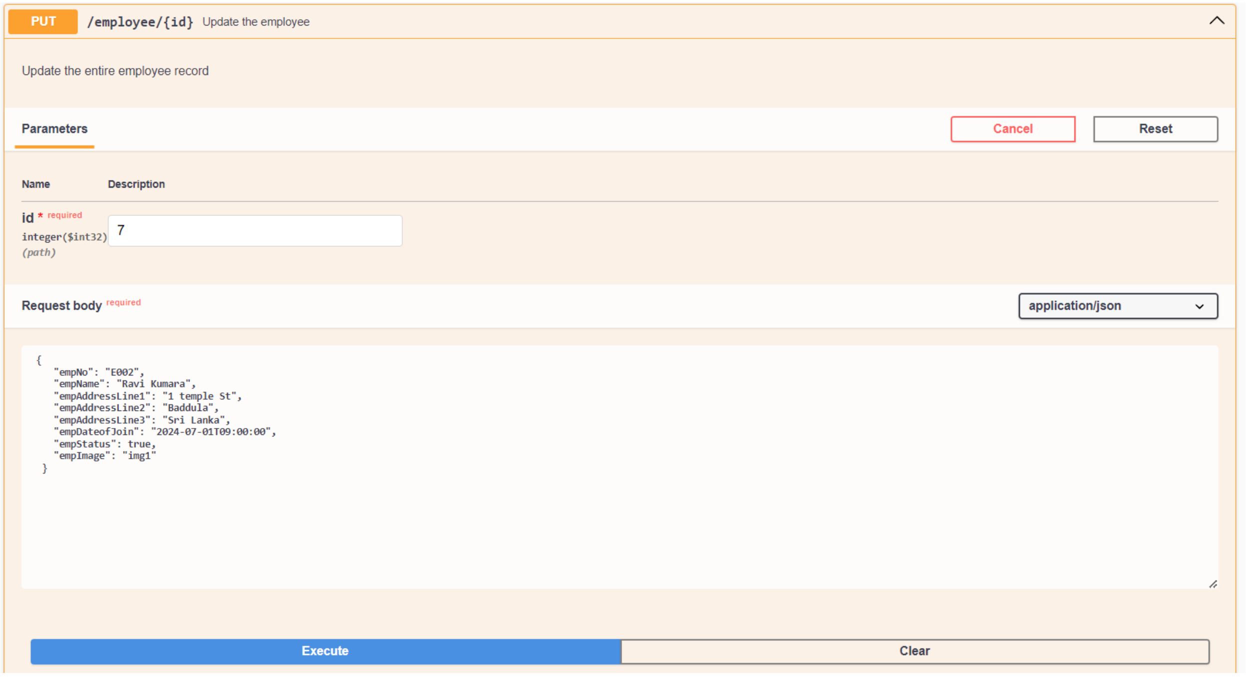
Code	Description	Links

3. Update (HTTP PUT or PATCH):

The Update operation allows modifying the information of an existing employee. There are two common approaches:

a. Update the entire employee record:

The client sends a PUT request to the `/employees/{id}` endpoint, providing the updated employee details in the request body. The server replaces the existing employee record with the new information.

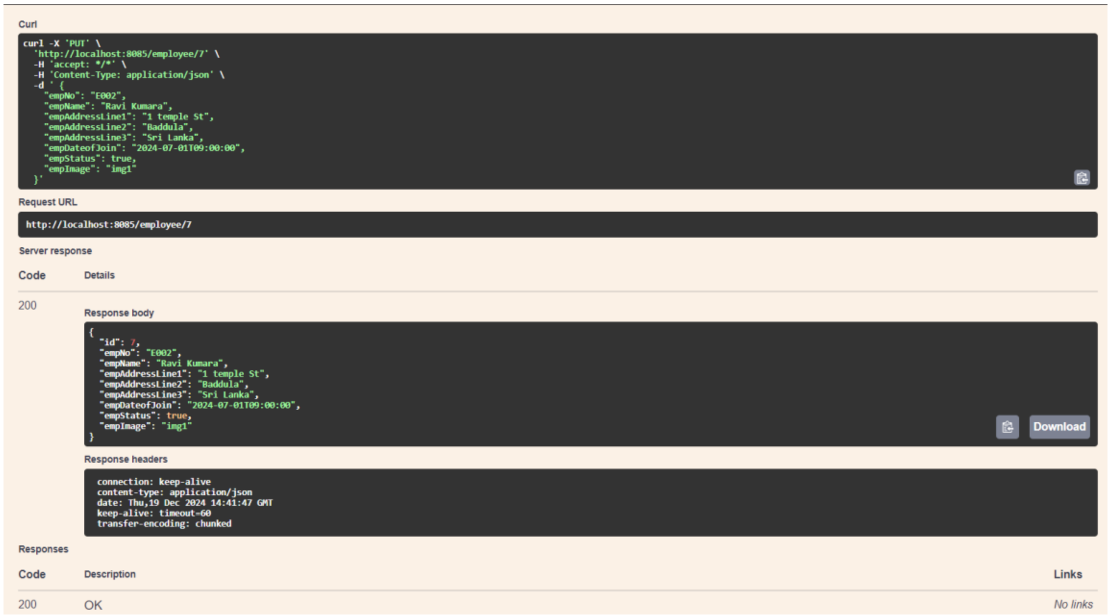


Update the entire employee record

Name	Description
id * required integer($int32) (path)	7

Request body required: application/json

```
{
  "empNo": "E002",
  "empName": "Ravi Kumara",
  "empAddressLine1": "1 temple St",
  "empAddressLine2": "Baddula",
  "empAddressLine3": "Sri Lanka",
  "empDateofJoin": "2024-07-01T09:00:00",
  "empStatus": true,
  "empImage": "img1"
}
```


Curl

```
curl -X 'PUT' \
  'http://localhost:8085/employee/7' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d ' {
    "empNo": "E002",
    "empName": "Ravi Kumara",
    "empAddressLine1": "1 temple St",
    "empAddressLine2": "Baddula",
    "empAddressLine3": "Sri Lanka",
    "empDateofJoin": "2024-07-01T09:00:00",
    "empStatus": true,
    "empImage": "img1"
  }'
```

Request URL

```
http://localhost:8085/employee/7
```

Server response

Code: 200

Response body

```
{
  "id": 7,
  "empNo": "E002",
  "empName": "Ravi Kumara",
  "empAddressLine1": "1 temple St",
  "empAddressLine2": "Baddula",
  "empAddressLine3": "Sri Lanka",
  "empDateofJoin": "2024-07-01T09:00:00",
  "empStatus": true,
  "empImage": "img1"
}
```

Response headers

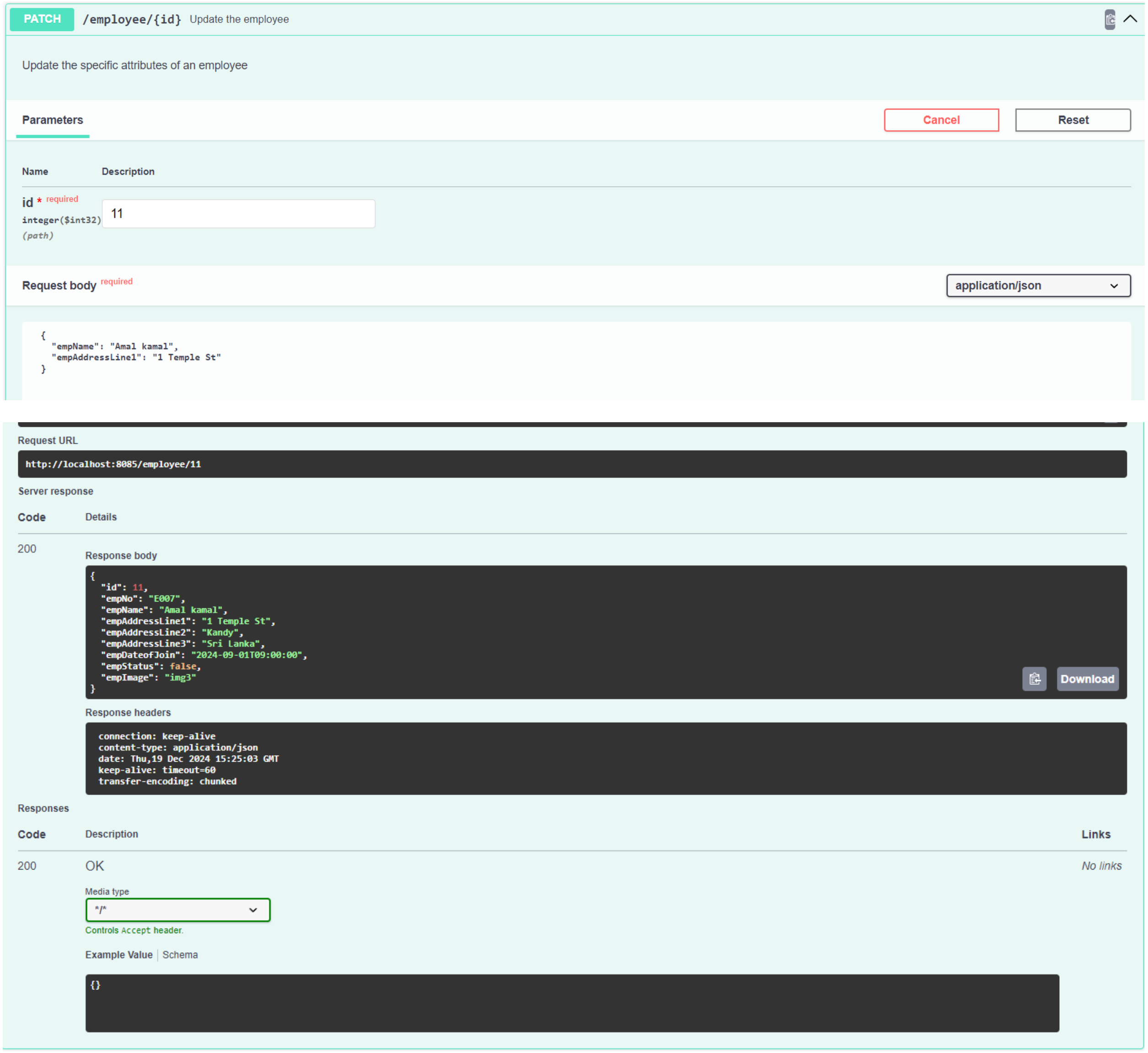
```
connection: keep-alive
content-type: application/json
date: Thu,19 Dec 2024 14:41:47 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

b. Update specific attributes of an employee:

The client sends a PATCH request to the `/employees/{id}` endpoint, providing a partial representation of the employee with only the fields that need to be updated. The server applies the changes to the corresponding employee record.



/employee/{id} Update the employee

Update the specific attributes of an employee

Name	Description
id * required integer($int32) (path)	11

Request body required

```
{
  "empName": "Amal kamal",
  "empAddressLine1": "1 Temple St"
}
```

Request URL

```
http://localhost:8085/employee/11
```

Server response

Code	Details
200	Response body

```
{
  "id": 11,
  "empNo": "E007",
  "empName": "Amal kamal",
  "empAddressLine1": "1 Temple St",
  "empAddressLine2": "Kandy",
  "empAddressLine3": "Sri Lanka",
  "empDateofJoin": "2024-09-01T09:00:00",
  "empStatus": false,
  "empImage": "img3"
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Thu,19 Dec 2024 15:25:03 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

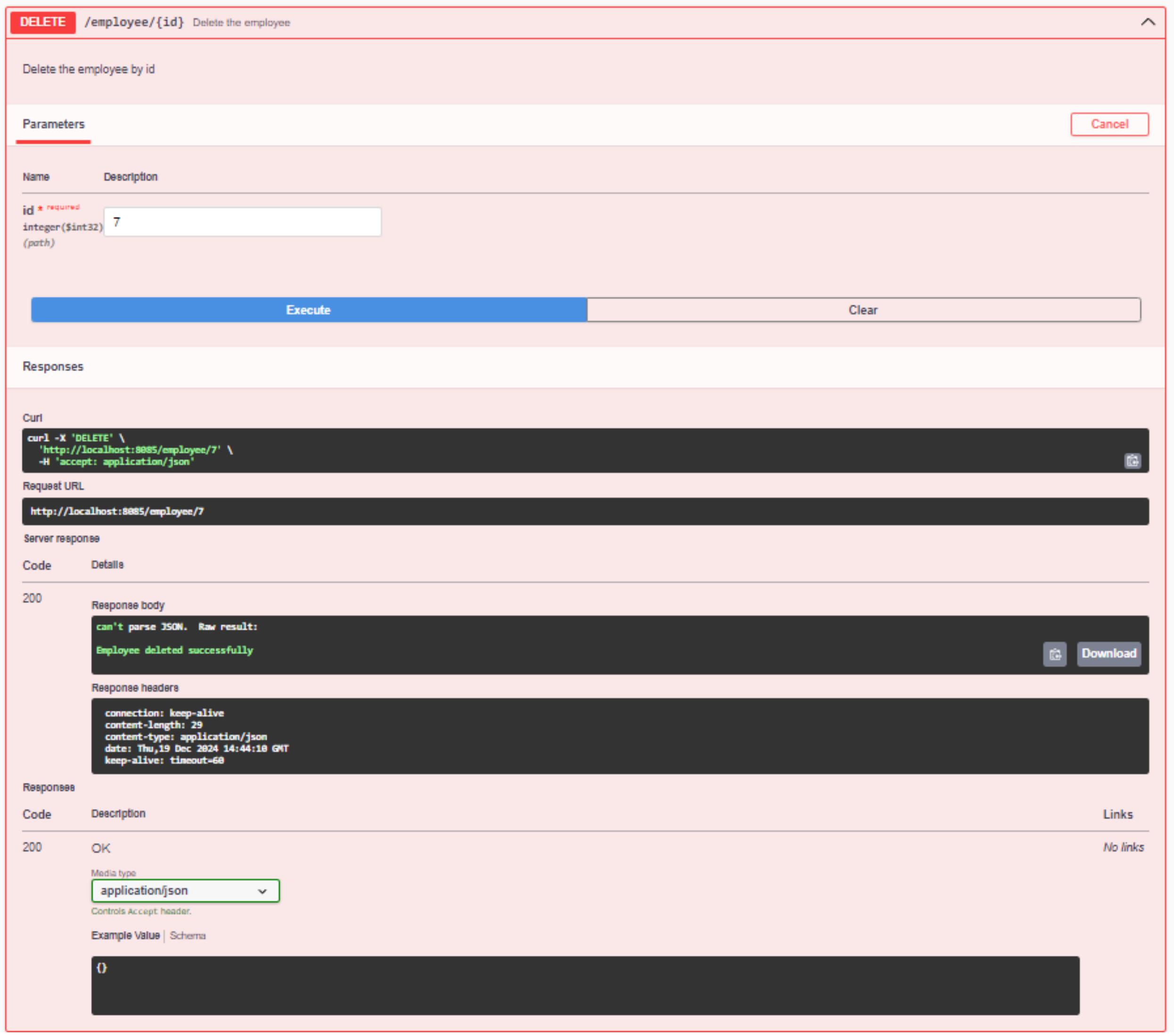
/

Controls Accept header.

```
{}
```

4. Delete (HTTP DELETE):

The Delete operation removes an employee from the collection. The client sends a DELETE request to the `/employees/{id}` endpoint, where `{id}` represents the unique identifier of the employee to be deleted. The server deletes the corresponding employee record from the database.



id: 7

Curl

```
curl -X 'DELETE' \
  'http://localhost:8085/employee/7' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8085/employee/7
```

Server response

Code: 200

Response body

```
can't parse JSON.  Raw result:

Employee deleted successfully
```

Response headers

```
connection: keep-alive
content-length: 29
content-type: application/json
date: Thu,19 Dec 2024 14:44:10 GMT
keep-alive: timeout=60
```

Responses

Code	Description	Links
200	OK	No links

Media type: application/json

```
{}
```