



Spring 25

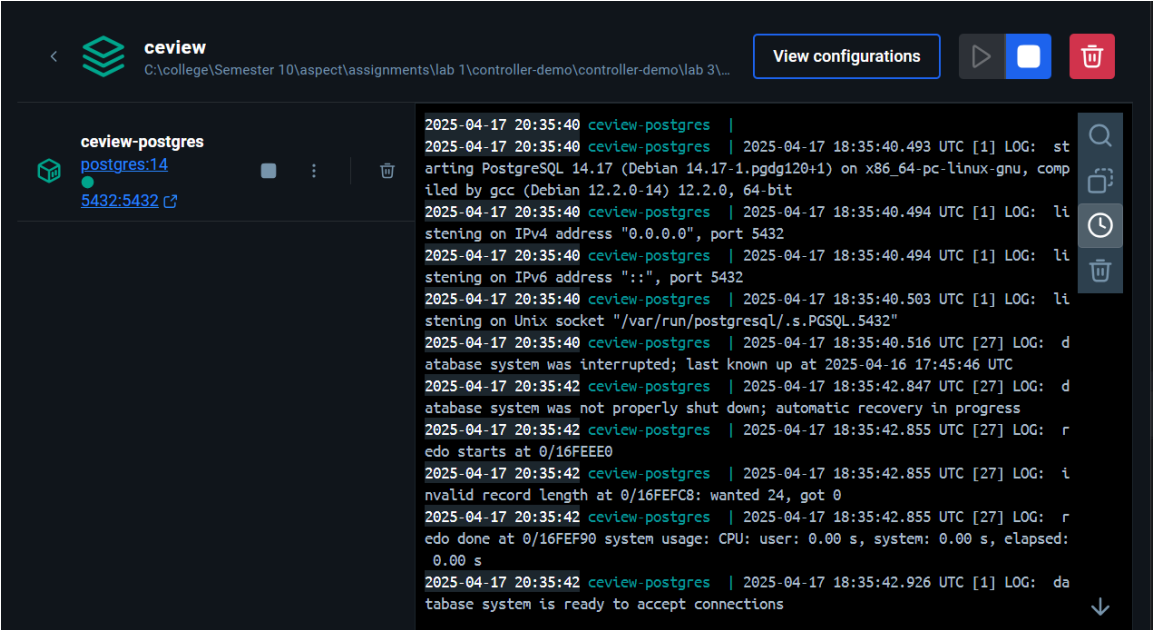
**CSE434 (UG2018) - Aspect- and Service-Oriented
Software Systems (37100)**

Lab assignment 3

Name: Salma Nasreldin Aboelela Ahmed

ID: 20p7105

Checking ceview in docker container



Checking docker running:

```
PS C:\college\Semester 10\aspect\assignments\lab 1\controller-demo\controller-demo\lab 3\ceview\ceview> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
f973dea24234   postgres:14   "docker-entrypoint.s..." 3 minutes ago  Up 3 minutes (healthy)  0.0.0.0:5432->5432/tcp             ceview-postgres
PS C:\college\Semester 10\aspect\assignments\lab 1\controller-demo\controller-demo\lab 3\ceview\ceview>
```

Listing databases on Docker CLI:

```
postgres=# \l

          List of databases
   Name   | Owner   | Encoding | Collate | Ctype   | Access privileges
-----+-----+-----+-----+-----+-----
ceview    | postgres | UTF8     | en_US.utf8 | en_US.utf8 | 
postgres | postgres | UTF8     | en_US.utf8 | en_US.utf8 | 
template0 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres      +
          |          |          |          |          | postgres=CTc/postgres
template1 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres      +
          |          |          |          |          | postgres=CTc/postgres
```

Connecting to ceview database:

```
postgres=# \c ceview
You are now connected to database "ceview" as user "postgres".
ceview=#
```

Checking tables:

```
ceview=# \dt
          List of relations
 Schema |   Name   | Type  | Owner
-----+-----+-----+-----
 public | restaurants | table | postgres
 public | reviews    | table | postgres
(2 rows)

ceview=#
```

Testing Workflow Example

1. Create a restaurant and save the returned UUID

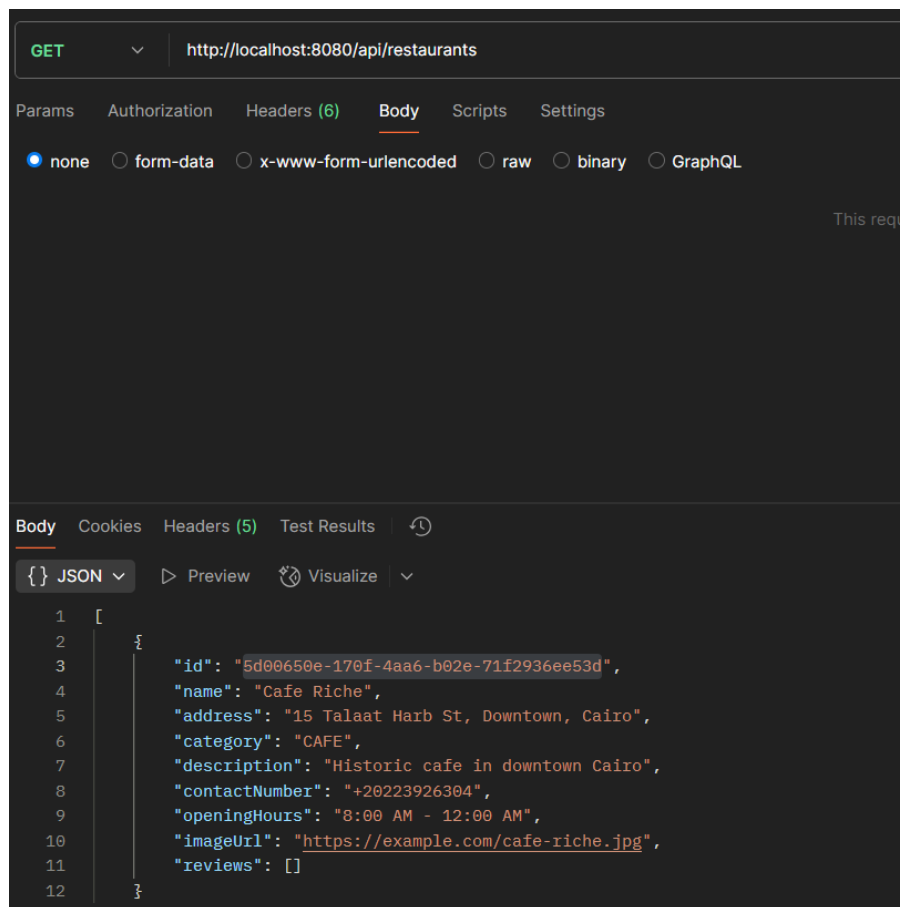
The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/api/restaurants
- Body:** A JSON object representing a restaurant:

```
{ "name": "Cafe Riche", "address": "15 Talaat Harb St, Downtown, Cairo", "category": "CAFE", "description": "Historic cafe in downtown Cairo", "contactNumber": "+20223926304", "openingHours": "8:00 AM - 12:00 AM", "imageUrl": "https://example.com/cafe-riche.jpg" }
```
- Response:** A 201 Created status with a response body:

```
{ "id": "5d00650e-170f-4aa6-b02e-71f2936ee53d", "name": "Cafe Riche", "address": "15 Talaat Harb St, Downtown, Cairo", "category": "CAFE", "description": "Historic cafe in downtown Cairo", "contactNumber": "+20223926304", "openingHours": "8:00 AM - 12:00 AM", "imageUrl": "https://example.com/cafe-riche.jpg", "reviews": [] }
```
- Performance:** 201 Created, 736 ms

2. Get all restaurants to verify it was created



3. Create a review for that restaurant using the UUID

The screenshot displays a REST client interface with a POST request to `http://localhost:8080/api/restaurants/5d00650e-170f-4aa6-b02e-71f2936ee53d/reviews`. The request body is raw JSON: `{ "rating": 4, "content": "Great atmosphere and excellent coffee. The historic ambiance adds to the experience." }`. The response body is also JSON, containing a full review object with an ID, rating, content, user information, timestamps, and a reported status.

POST `http://localhost:8080/api/restaurants/5d00650e-170f-4aa6-b02e-71f2936ee53d/reviews`

Params Authorization Headers (9) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 { "rating": 4, "content": "Great atmosphere and excellent coffee. The historic ambiance adds to the experience." }
```

Body Cookies Headers (5) Test Results ↻

{ } **JSON** ▾ ▶ Preview 🔄 Visualize ▾

```
1 {
2   "id": "497b2cc4-c11b-4372-81f6-69619812a91e",
3   "rating": 4,
4   "content": "Great atmosphere and excellent coffee. The historic ambiance adds to the experience.",
5   "userName": "Ahmed Mohamed",
6   "userEmail": "ahmed@example.com",
7   "createdAt": "2025-04-17T21:09:01.1920931",
8   "updatedAt": "2025-04-17T21:09:01.1920931",
9   "reported": false
10 }
```

4. Get all reviews for that restaurant

HTTP `http://localhost:8080/api/restaurants/5d00650e-170f-4aa6-b02e-71f2936ee53d/reviews`

GET `http://localhost:8080/api/restaurants/5d00650e-170f-4aa6-b02e-71f2936ee53d/reviews`

Params Authorization Headers (6) **Body** Scripts Settings

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (5) Test Results ⓘ

{ } JSON ▾ ▶ Preview 🔄 Visualize ▾

```
1  [
2    {
3      "id": "497b2cc4-c11b-4372-81f6-69619812a91e",
4      "rating": 4,
5      "content": "Great atmosphere and excellent coffee. The historic ambiance adds to the experience.",
6      "userName": "Ahmed Mohamed",
7      "userEmail": "ahmed@example.com",
8      "createdAt": "2025-04-17T21:09:01.192093",
9      "updatedAt": "2025-04-17T21:09:01.192093",
10     "reported": false
11   }
12 ]
```

5. Update the restaurant

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/api/restaurants/5d00650e-170f-4aa6-b02e-71f2936ee53d`. The request body is a JSON object with the following fields: `name`, `address`, `category`, `description`, `contactNumber`, `openingHours`, and `imageUrl`. The response status is `200 OK` with a response time of `33 ms`. The response body is a JSON object with the same fields as the request, plus an additional `reviews` array containing one review object with an `id`.

```
PUT http://localhost:8080/api/restaurants/5d00650e-170f-4aa6-b02e-71f2936ee53d

{
  "name": "Cafe Riche Updated",
  "address": "15 Talaat Harb St, Downtown, Cairo",
  "category": "CAFE",
  "description": "Updated description for historic cafe in downtown Cairo",
  "contactNumber": "+20223926384",
  "openingHours": "9:00 AM - 11:00 PM",
  "imageUrl": "https://example.com/cafe-riche-updated.jpg"
}
```

```
{
  "id": "5d00650e-170f-4aa6-b02e-71f2936ee53d",
  "name": "Cafe Riche Updated",
  "address": "15 Talaat Harb St, Downtown, Cairo",
  "category": "CAFE",
  "description": "Updated description for historic cafe in downtown Cairo",
  "contactNumber": "+20223926384",
  "openingHours": "9:00 AM - 11:00 PM",
  "imageUrl": "https://example.com/cafe-riche-updated.jpg",
  "reviews": [
    {
      "id": "497b2cc4-c11b-4372-81f6-69619812a91e",
    }
  ]
}
```

6. Update the review

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/api/reviews/497b2cc4-c11b-4372-81f6-69619812a91e`. The request body is a JSON object with the following fields: `userEmail`, `userName`, `rating`, and `content`. The response status is `200 OK` with a response time of `33 ms`. The response body is a JSON object with the same fields as the request, plus additional fields: `id`, `createdAt`, `updatedAt`, and `reported`.

```
PUT http://localhost:8080/api/reviews/497b2cc4-c11b-4372-81f6-69619812a91e

{
  "userEmail": "ahmed@example.com",
  "userName": "Ahmed Mohamed",
  "rating": 5,
  "content": "Updated review: Excellent service and amazing food. Highly recommended!"
}
```

```
{
  "id": "497b2cc4-c11b-4372-81f6-69619812a91e",
  "rating": 5,
  "content": "Updated review: Excellent service and amazing food. Highly recommended!",
  "userName": "Ahmed Mohamed",
  "userEmail": "ahmed@example.com",
  "createdAt": "2025-04-17T21:09:01.192093",
  "updatedAt": "2025-04-17T21:25:01.416881",
  "reported": false
}
```


7. Report the review

The screenshot shows a REST client interface with a PATCH request to `http://localhost:8080/api/reviews/497b2cc4-c11b-4372-81f6-69619812a91e/report`. The request body is a JSON object with the following fields: `id`, `rating`, `content`, `userName`, `userEmail`, `createdAt`, `updatedAt`, and `reported`. The response is also a JSON object with the same fields, where `reported` is set to `true`.

Request:

- Method: PATCH
- URL: `http://localhost:8080/api/reviews/497b2cc4-c11b-4372-81f6-69619812a91e/report`
- Body Type: none

Response:

```
1  {
2    "id": "497b2cc4-c11b-4372-81f6-69619812a91e",
3    "rating": 5,
4    "content": "Updated review: Excellent service and amazing food. Highly recommended!",
5    "userName": "Ahmed Mohamed",
6    "userEmail": "ahmed@example.com",
7    "createdAt": "2025-04-17T21:09:01.192093",
8    "updatedAt": "2025-04-17T21:28:05.3516636",
9    "reported": true
10 }
```

8. Get all reported reviews

GET <http://localhost:8080/api/reviews/reported>

Params Authorization Headers (6) **Body** Scripts Settings

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (5) Test Results ↺

{ } JSON ▾ ▶ Preview ↻ Visualize ▾

```
1  [
2    {
3      "id": "497b2cc4-c11b-4372-81f6-69619812a91e",
4      "rating": 5,
5      "content": "Updated review: Excellent service and amazing food. Highly recommended!",
6      "userName": "Ahmed Mohamed",
7      "userEmail": "ahmed@example.com",
8      "createdAt": "2025-04-17T21:09:01.192093",
9      "updatedAt": "2025-04-17T21:28:05.351664",
10     "reported": true
11   }
12 ]
```

9. Delete the review

DELETE <http://localhost:8080/api/reviews/497b2cc4-c11b-4372-81f6-69619812a91e>

Params Authorization Headers (6) **Body** Scripts Settings

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (3) Test Results ↺

Raw ▾ ▶ Preview ↻ Visualize ▾

1

204 No Content

10. Delete the restaurant

