

Upon examination of the screenshot, it's evident that the event list was successfully retrieved using a GET request.

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

- Method:** GET
- URL:** `http://127.0.0.1:8000/events/list/` (highlighted with a red circle)
- Body:** none (selected)
- Status:** 200 OK, 183 ms, 635 B
- Response Format:** JSON (selected)
- Response Body:** A JSON array of two event objects (highlighted with a red circle):

```
[
  {
    "id": 1,
    "event_name": "Prove by recognize",
    "city_name": "Robertstad",
    "date": "2024-04-01",
    "time": "17:00:00",
    "latitude": -66.50384758,
    "longitude": -26.49728437
  },
  {
    "id": 2,
    "event_name": "Support least recent",
    "city_name": "Carlville",
    "date": "2024-03-22",
    "time": "04:30:00",
    "latitude": -61.57435141,
    "longitude": 178.5312529
  }
]
```

An examination of the screenshot reveals the successful retrieval of a specific event using its ID through a GET request.

The screenshot displays a REST client interface with a GET request to `http://127.0.0.1:8000/events/list/1/`. The response is a JSON object containing event details, which is highlighted with a red circle. The interface includes tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The response status is 200 OK, with a response time of 71 ms and a size of 478 B. The response body is shown in a Pretty format, displaying a JSON object with the following fields:

```
{
  "id": 1,
  "event_name": "Prove by recognize",
  "city_name": "Robertstad",
  "date": "2024-04-01",
  "time": "17:00:00",
  "latitude": -66.50384758,
  "longitude": -26.49728437
}
```

The creation of a new event is clearly demonstrated in the provided screenshot using POST request

The screenshot displays a REST client interface with a POST request to `http://127.0.0.1:8000/events/create/`. The request body is a JSON object containing event details. The response is a 201 status code, indicating successful creation, with a response time of 48 ms and a body size of 487 B. The response body is also a JSON object containing the created event's details.

Request:

```
POST http://127.0.0.1:8000/events/create/

{
  "event_name": "Between thus table",
  "city_name": "Port Rebeccaberg",
  "date": "2024-03-01",
  "time": "18:00:00",
  "latitude": 38.33354302,
  "longitude": 157.9579286
}
```

Response:

```
201 Created 48 ms 487 B

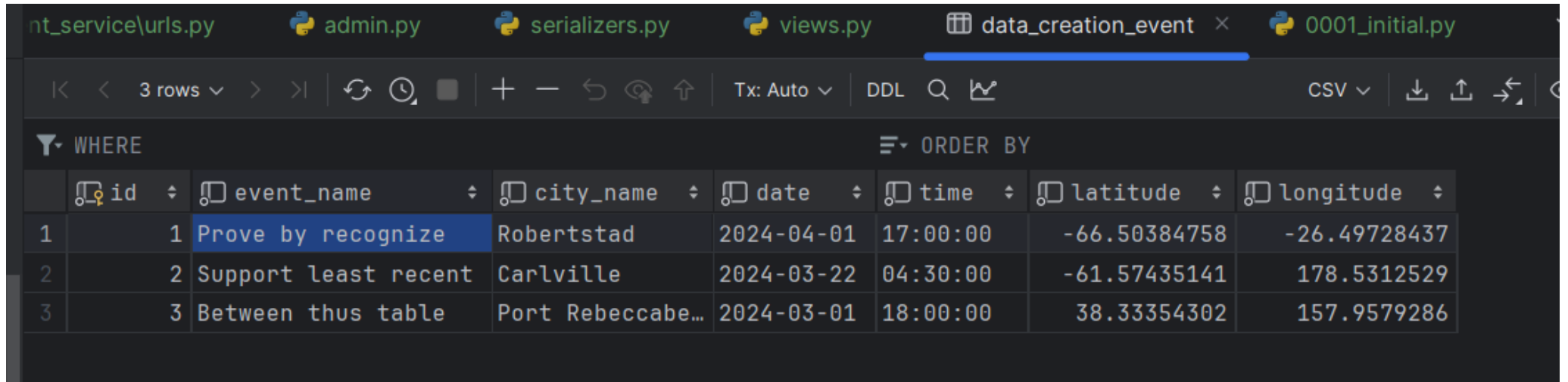
{
  "id": 3,
  "event_name": "Between thus table",
  "city_name": "Port Rebeccaberg",
  "date": "2024-03-01",
  "time": "18:00:00",
  "latitude": 38.33354302,
  "longitude": 157.9579286
}
```

The screenshot illustrates the update of the events list, incorporating a new event identified by the id =3.

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:8000/events/list/`. The response is a 200 OK status with a 24 ms latency and 794 B of data. The response body is displayed in JSON format, showing a list of three events. The third event, with id 3, is the new event added to the list.

```
4      "event_name": "Prove by recognize",
5      "city_name": "Robertstad",
6      "date": "2024-04-01",
7      "time": "17:00:00",
8      "latitude": -66.50384758,
9      "longitude": -26.49728437
10    },
11    {
12      "id": 2,
13      "event_name": "Support least recent",
14      "city_name": "Carlville",
15      "date": "2024-03-22",
16      "time": "04:30:00",
17      "latitude": -61.57435141,
18      "longitude": 178.5312529
19    },
20    {
21      "id": 3,
22      "event_name": "Between thus table",
23      "city_name": "Port Rebeccaberg",
24      "date": "2024-03-01",
25      "time": "18:00:00",
26      "latitude": 38.33354302,
27      "longitude": 157.9579286
28    }
29  ]
```

Furthermore, the modifications are appropriately reflected within the internal database, as confirmed by the screenshot.



The screenshot displays a database management interface with a table named 'data_creation_event'. The table contains 3 rows of data. The first row is highlighted. The columns are: id, event_name, city_name, date, time, latitude, and longitude. The interface also shows a toolbar with various icons and a search bar.

	id	event_name	city_name	date	time	latitude	longitude
1	1	Prove by recognize	Robertstad	2024-04-01	17:00:00	-66.50384758	-26.49728437
2	2	Support least recent	Carlville	2024-03-22	04:30:00	-61.57435141	178.5312529
3	3	Between thus table	Port Rebeccabe...	2024-03-01	18:00:00	38.33354302	157.9579286

Test Case for the finder API as per the assignment :

Input: User's Source Latitude: 40.7128, User's Source Longitude: -74.0060, Search Date: 2024-03-15

The screenshot displays a code editor with a project structure on the left and a debug console at the bottom. The project structure includes folders like `event_service`, `data_creation`, and `event_finder`, with files such as `urls.py` and `views.py`. The main editor window shows Python code in `event_finder\urls.py` for the `find` endpoint. The code calculates the distance between the user's location and event locations and serializes the event data.

```
event_data = [] * 10 # Create a list to store event data for the current page

# Calculate start and end index for events based on page number
start_index = page * 10
end_index = start_index + 10

# Retrieve events for the current page
current_events = events[start_index:end_index]

for event in current_events:
    # Retrieve weather conditions for the event
    weather = get_weather(event.city_name, event.date)

    # Calculate distance between user's location and event location
    distance_km = calculate_distance(user_latitude, user_longitude, event.latitude, event.longitude)

    # Serialize event data
```

The debug console shows the execution of the `find` endpoint. The input parameters are `user_latitude: 38.33354302168567`, `user_longitude: 157.95792863097148`, and `search_date: 2024-03-01`. The output shows the calculated distance and the event data for the search date 2024-03-15. A red circle highlights the output for the search date 2024-03-15, and a red arrow points to it.

```
user_latitude: 38.33354302168567 user_longitude: 157.95792863097148 search_date: 2024-03-01
search_date: 2024-03-01 end_date: 2024-03-15
[03/Apr/2024 11:21:35] "GET /events/find/ HTTP/1.1" 200 7004
user_latitude: 40.7128 user_longitude: -74.006 search_date: 2024-03-15
search_date: 2024-03-15 end_date: 2024-03-29
[03/Apr/2024 11:26:12] "GET /events/find/ HTTP/1.1" 200 6907
```

Output format as per the assignment in (/events/find REST API endpoint) :

Page 1

The screenshot shows a REST client interface with a GET request to `http://localhost:8000/event-finder/find/`. The request body is highlighted with a red circle and contains the following JSON:

```
{
  "date": "2024-03-15",
  "latitude": 40.7128,
  "longitude": -74.0060
}
```

The response body is also highlighted with a red circle and contains the following JSON:

```
{
  "Page1": {
    "events": [
      {
        "event_name": "Structure support choice",
        "city_name": "Fryland",
        "date": "2024-03-15",
        "weather": "Rainy 25C",
        "distance_km": "8910.23984646717"
      },
      {
        "event_name": "Party development available",
        "city_name": "Port Alexander",
        "date": "2024-03-15",
        "weather": "Windy 27C",
        "distance_km": "12710.135679990923"
      },
      {
        "event_name": "Of ask open",

```

Red arrows point to the request body and the response body. The status bar at the bottom indicates a 200 OK response with a time of 1 m 57.90 s and a size of 7.06 KB.

GET ⌵ http://localhost:8000/events/find/ Send ⌵

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary **JSON** ⌵ Beautify

```
1  [
2    "date": "2024-03-15",
3    "latitude": 40.7128,
4    "longitude": -74.0060
5  ]
```

Body Cookies Headers (10) Test Results ⌵ 🌐 Status: 200 OK Time: 1 m 57.90 s Size: 7.06 KB Save Response ⌵

Pretty Raw Preview Visualize **JSON** ⌵ ≡ 🔍

```
73     }
74   ],
75   "page": 1,
76   "pageSize": 10,
77   "totalEvents": 44,
78   "totalPages": 5
79 },
80 "Page2": {
81   "events": [
82     {
83       "event_name": "Player",
84       "city_name": "Lewischester",
85       "date": "2024-03-19",
86       "weather": "Cloudy 16C",
87       "distance_km": "15232.42376978484"
88     },
89     {
90       "event_name": "May",
91       "city_name": "New Brittany",
```