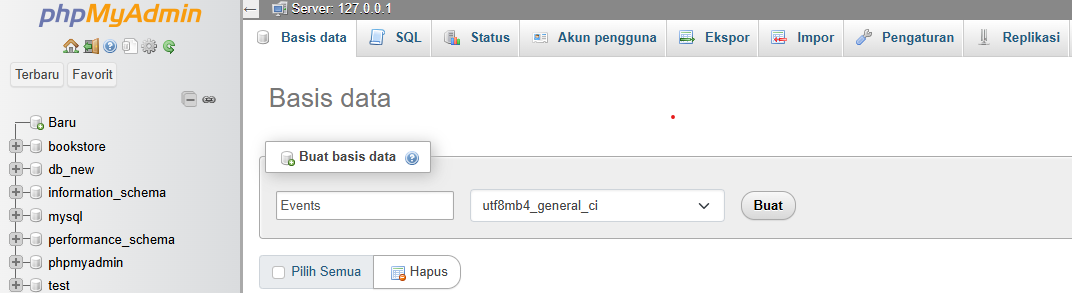
Nama : Sekar Wulan Ayu Lisanti

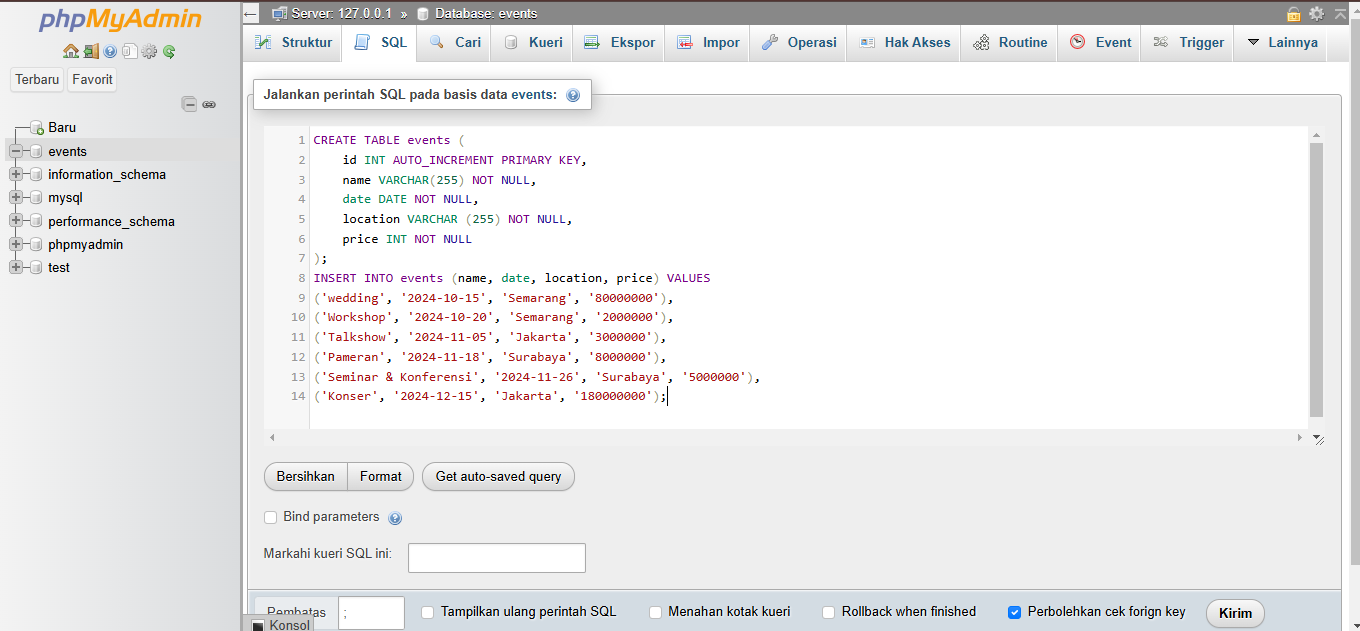
Nim : 21. 01.55.0005

Prodi : Sistem Informasi

1. Buat database “events”



1. Buat tabel “events” dengan menggunakan SQL sebagai berikut :



SQL Events :

CREATE TABLE events (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

date DATE NOT NULL,

location VARCHAR (255) NOT NULL,

price INT NOT NULL

);

INSERT INTO events (name, date, location, price) VALUES

('wedding', '2024-10-15, 'Semarang', '80000000'),

('Workshop', '2024-10-20', 'Semarang', '2000000'),

('Talkshow', '2024-11-28', 'Jakarta', '3000000'),

('Pameran', '2024-11-18', 'Surabaya', '8000000'),

('Seminar & Konferensi', '2024-11-26', 'Surabaya', '5000000'),

(Konser, '2024-12-15', 'Jakarta', '180000000'),

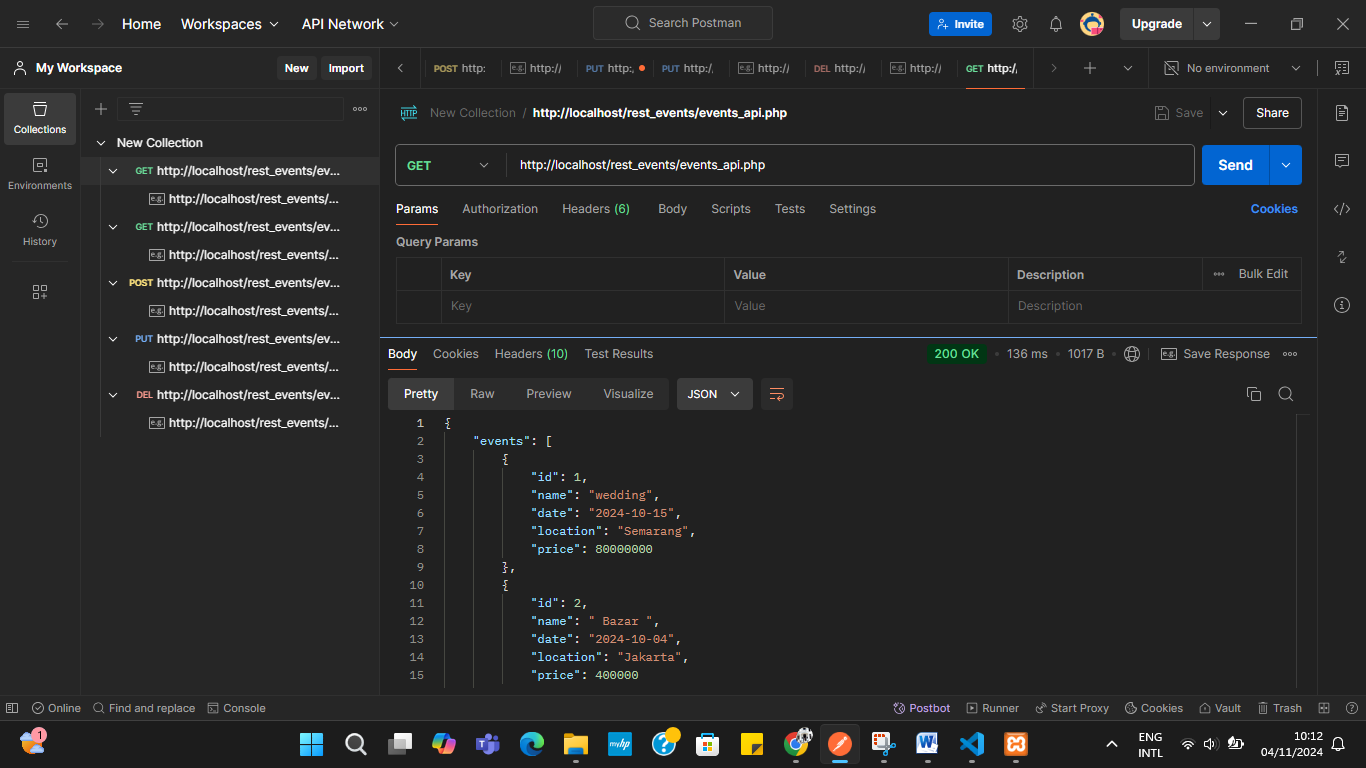
('wedding', '2024-12-28', 'Semarang', '80000000');

1. Source code lengkap pada visual studio code
2. <?php
3. header("Content-Type: application/json; charset=UTF-8");
4. header("Access-Control-Allow-Origin: \*");
5. header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE");
6. header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");
7. $method = $\_SERVER['REQUEST\_METHOD'];
8. $request = [];
9. if (isset($\_SERVER['PATH\_INFO'])) {
10. $request = explode('/', trim($\_SERVER['PATH\_INFO'],'/'));
11. }
12. function getConnection() {
13. $host = 'localhost';
14. $db   = 'events';
15. $user = 'root';
16. $pass = ''; // Ganti dengan password MySQL Anda jika ada
17. $charset = 'utf8mb4';
18. $dsn = "mysql:host=$host;dbname=$db;charset=$charset";
19. $options = [
20. PDO::ATTR\_ERRMODE            => PDO::ERRMODE\_EXCEPTION,
21. PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC,
22. PDO::ATTR\_EMULATE\_PREPARES   => false,
23. ];
24. try {
25. return new PDO($dsn, $user, $pass, $options);
26. } catch (\PDOException $e) {
27. throw new \PDOException($e->getMessage(), (int)$e->getCode());
28. }
29. }
30. function response($status, $data = NULL) {
31. header("HTTP/1.1 " . $status);
32. if ($data) {
33. echo json\_encode($data);
34. }
35. exit();
36. }
37. function validateevents($name, $date, $location, $price) {
38. $errors = [];
39. if (empty($name)) {
40. $errors[] = "name is required";
41. }
42. if (empty($date)) {
43. $errors[] = "date is required";
44. }
45. if (empty($location)) {
46. $errors[] = "location is required";
47. }
48. if (empty($price)) {
49. $errors[] = "price is required";
50. }
51. return $errors;
52. }
53. $db = getConnection();
54. switch ($method) {
55. case 'GET':
56. if (!empty($request) && isset($request[0])) {
57. if ($request[0] === 'search') {
58. // 5.1 Search functionality
59. $searchTerm = $\_GET['term'] ?? '';
60. $stmt = $db->prepare("SELECT \* FROM events WHERE name LIKE ? OR location LIKE ?");
61. $searchTerm = "%$searchTerm%";
62. $stmt->execute([$searchTerm, $searchTerm]);
63. $events = $stmt->fetchAll();
64. response(200, $events);
65. } else {
66. // Get specific events
67. $id = $request[0];
68. $stmt = $db->prepare("SELECT \* FROM events WHERE id = ?");
69. $stmt->execute([$id]);
70. $events = $stmt->fetch();
71. if ($events) {
72. response(200, $events);
73. } else {
74. response(404, ["message" => "events not found"]);
75. }
76. }
77. } else {
78. // 5.2 Pagination
79. $page = isset($\_GET['page']) ? (int)$\_GET['page'] : 1;
80. $limit = isset($\_GET['limit']) ? (int)$\_GET['limit'] : 10;
81. $offset = ($page - 1) \* $limit;
82. $stmt = $db->prepare("SELECT \* FROM events LIMIT ? OFFSET ?");
83. $stmt->bindValue(1, $limit, PDO::PARAM\_INT);
84. $stmt->bindValue(2, $offset, PDO::PARAM\_INT);
85. $stmt->execute();
86. $events = $stmt->fetchAll();
87. $totalStmt = $db->query("SELECT COUNT(\*) FROM events");
88. $total = $totalStmt->fetchColumn();
89. response(200, [
90. 'events' => $events,
91. 'total' => $total,
92. 'page' => $page,
93. 'limit' => $limit
94. ]);
95. }
96. break;
98. case 'POST':
99. $data = json\_decode(file\_get\_contents("php://input"));
100. $errors = validateevents($data->name ?? '', $data->date ?? '', $data->location ?? '', $data->price ?? '' );
101. if (!empty($errors)) {
102. response(400, ["errors" => $errors]);
103. }
104. $sql = "INSERT INTO events (name, date, location, price) VALUES (?, ?, ?, ?)";
105. $stmt = $db->prepare($sql);
106. if ($stmt->execute([$data->name, $data->date, $data->location, $data->price])) {
107. response(201, ["message" => "events created", "id" => $db->lastInsertId()]);
108. } else {
109. response(500, ["message" => "Failed to create events"]);
110. }
111. break;
113. case 'PUT':
114. if (empty($request) || !isset($request[0])) {
115. response(400, ["message" => "events ID is required"]);
116. }
117. $id = $request[0];
118. $data = json\_decode(file\_get\_contents("php://input"));
119. $errors = validateevents($data->name ?? '', $data->date ?? '', $data->location ?? '', $data->price ?? '');
120. if (!empty($errors)) {
121. response(400, ["errors" => $errors]);
122. }
123. $sql = "UPDATE events SET name = ?, date = ?, location = ?, price = ? WHERE id =?";
124. $stmt = $db->prepare($sql);
125. if ($stmt->execute([$data->name, $data->date, $data->location, $data->price, $id])) {
126. response(200, ["message" => "events updated"]);
127. } else {
128. response(500, ["message" => "Failed to update events"]);
129. }
130. break;
132. case 'DELETE':
133. if (empty($request) || !isset($request[0])) {
134. response(400, ["message" => "events ID is required"]);
135. }
136. $id = $request[0];
137. $sql = "DELETE FROM events WHERE id = ?";
138. $stmt = $db->prepare($sql);
139. if ($stmt->execute([$id])) {
140. response(200, ["message" => "events deleted"]);
141. } else {
142. response(500, ["message" => "Failed to delete events"]);
143. }
144. break;
146. default:
147. response(405, ["message" => "Method not allowed"]);
148. break;
149. }
150. ?>

Daftar endpoint dan pengujian pada postman

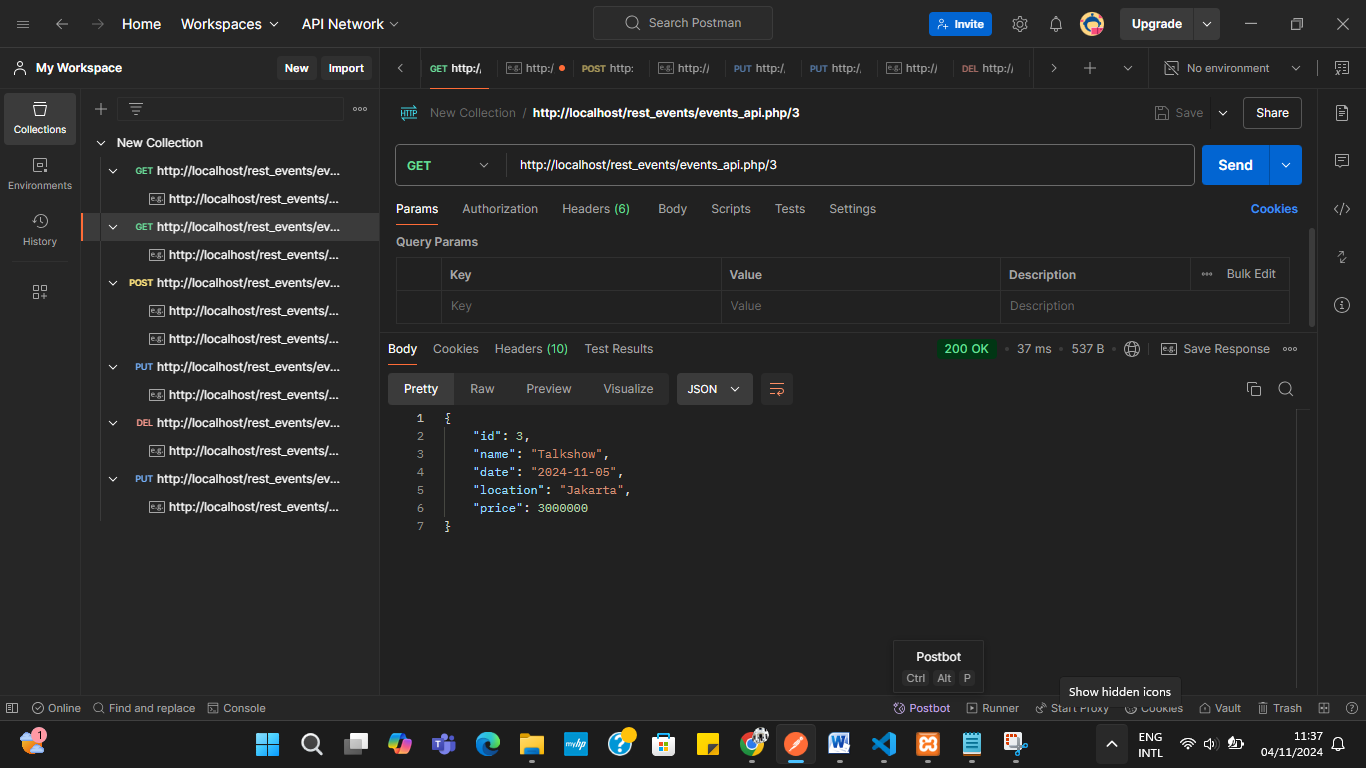
1. Menampilkan seluruh data dengan menggunakan metode get. Lalu uji

URL : http://localhost/rest\_events/events\_api.php



1. Menampilkan data berdasarkan id dengan menggunakan metode get , lalu uji

URL : <http://localhost/rest_events/events_api.php/3>



1. Menambah data baru

Dengan menggunakan method post dengan menambahkan

**Headers:**

Key: Content-Type

Value: application/json

**Body:**

Pilih "raw" dan "JSON"

{

    "name": "ulang tahun",

    "date": "2024-11-19",

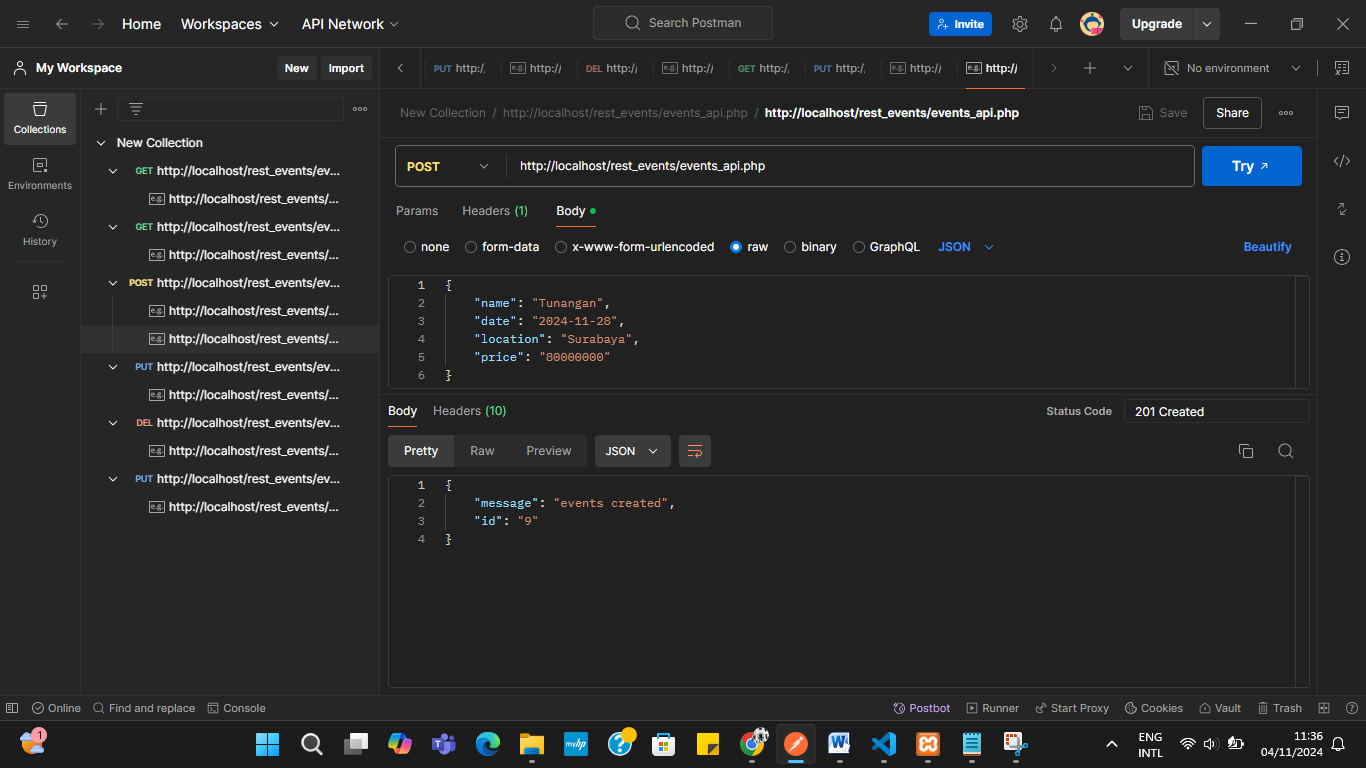
    "location": "Surabaya",

    "price": 5000000

}

Lalu uji dengan

URL : http://localhost/rest\_events/events\_api.php



1. Mengupdate data

Methode : PUT

**Headers:**

Key: Content-Type

Value: application/json

**Body:**

Pilih "raw" dan "JSON"

{

    "name": " Bazar ",

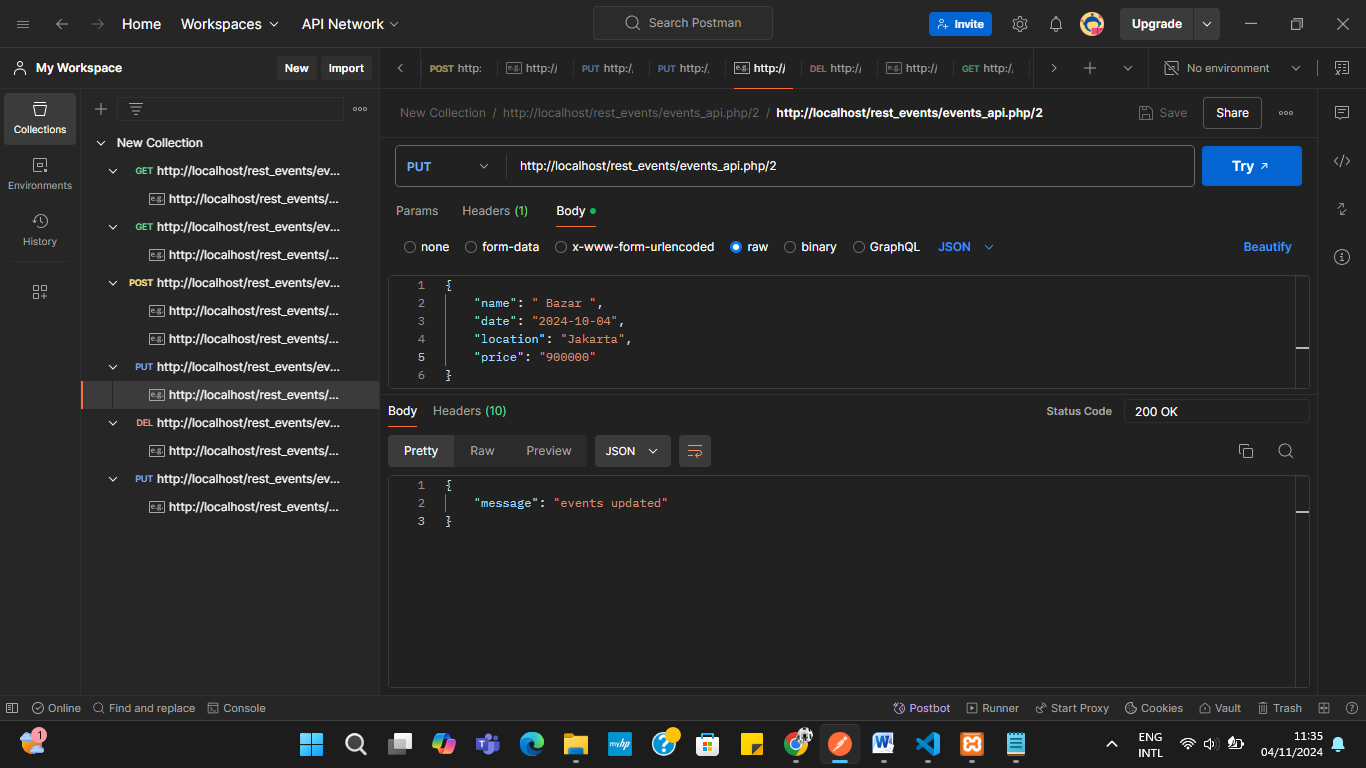
    "date": "2024-10-04",

    "location": "Jakarta",

    "price": "900000"

}

URL : http://localhost/rest\_events/events\_api.php/2



1. Menghapus data

Method : Delete

URL : http://localhost/rest\_events/events\_api.php/4

