IMPLEMENTASI REST API DATA PEGAWAI MENGGUNAKAN PHP DAN PENGUJIAN DENGAN POSTMAN

Dosen Pengampu:

Andi Iwan Nur Hidayat, S.Kom., M.Kom.

Disusun Oleh:

1. Feis Aulia Fatchuriani (22091397053)
2. Berlin Marsyah Yustina (22091397062)
3. Tsamarah Muadzah Lubis (22091397974)

PROGRAM STUDI D4 MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA

2024

❖ **REST API (Representational State Transfer Application Programming Interface)** adalah sebuah metode komunikasi antara perangkat lunak yang memungkinkan sistem yang berbeda untuk berinteraksi satu sama lain melalui protokol HTTP (Hypertext Transfer Protocol). REST API memungkinkan aplikasi untuk berkomunikasi dan berbagi data dengan sistem lain secara terstruktur.

❖ **Postman** adalah sebuah aplikasi yang memungkinkan pengembang perangkat lunak untuk melakukan pengujian, debugging, dan dokumentasi pada API. Dengan Postman, Anda dapat dengan mudah mengirim permintaan HTTP ke server, menangani respons, dan mengatur data yang dikirimkan dan diterima.

Pembuatan Database

Kami membuat database dengan nama ‘employees’ dan dengan tabel ‘employees’ yang memiliki entitas berikut:

	id	name	email	position	salary
Ubah Salin Hapus	1	Bahari	bahari@gmail.com	Manajer	20000000
Ubah Salin Hapus	2	Bambang Sriwijaya	bambangsus@gmail.com	Marketing	9000000
Ubah Salin Hapus	3	Fahrial	fahrial@gmail.com	Web Developer	30000000
Ubah Salin Hapus	6	Fafa	fafafafa@gmail.com	Manager	7000000
Ubah Salin Hapus	12	fhuh	bahari@gmail.com	Manajer	20000000
Ubah Salin Hapus	15	Soliha	soliha@gmail.com	CEO	9000

Pilih Semua *Dengan pilihan:* Ubah Salin Hapus Ekspor

Membuat Koneksi Database (helper.php)

helper.php > ...

```php
<?php

    define('HOST', 'localhost');
    define('USER', 'root');
    define('PASS', '');
    define('DB', 'employees');

    $db_connect = mysqli_connect(HOST, USER, PASS, DB) or die ('Unable Connect');

    header('Content-Type: application/json');
?>
```

Kode tersebut adalah bagian dari sebuah skrip PHP yang digunakan untuk menghubungkan ke database MySQL dan mengatur header untuk output dalam format JSON.

- **`define('HOST', 'localhost');`, `define('USER', 'root');`, `define('PASS', '');`, `define('DB', 'employees');`**

Mendefinisikan konstanta yang digunakan untuk menyimpan informasi koneksi database, seperti host (alamat server database), username, password, dan nama database.

- **$db_connect = mysqli_connect(HOST, USER, PASS, DB) or die ('Unable Connect');**
  Menggunakan fungsi `mysqli_connect()` untuk membuat koneksi ke database MySQL. Jika koneksi gagal, skrip akan berhenti dan menampilkan pesan 'Unable Connect'.
- **header('Content-Type: application/json');**
  Mengatur header HTTP untuk memberitahu browser bahwa konten yang dikirimkan adalah JSON. Hal ini penting jika Anda ingin mengirim data dalam format JSON dari server ke aplikasi klien, seperti yang umumnya digunakan dalam REST API.

Data.php

```php
<?php

    require_once('helper.php');

    $query = "SELECT * FROM employees order by id desc";
    $sql = mysqli_query($db_connect, $query);

    if ($sql) {
        $result = array();
        while ($row = mysqli_fetch_array($sql)) {
            array_push($result, array(
                'id' => $row['id'],
                'name' => $row ['name'],
                'email' => $row ['email'],
                'position' => $row ['position'],
                'salary' => $row ['salary'],

            ));
        }

    echo json_encode (array('result' => $result));

    }

?>
```

Kode diatas memuat file helper untuk detail koneksi database, mendefinisikan query SQL untuk mengambil data karyawan, mengeksekusi query, memeriksa keberhasilan, mengiterasi melalui baris yang diambil untuk membangun array asosiatif data karyawan, kemudian mengkodekan array data karyawan sebagai string JSON, dan menampilkannya, potensial sebagai respons terhadap permintaan AJAX atau untuk pemrosesan lebih lanjut.

- **require_once('helper.php');**
  Script ini memuat file helper.php yang diasumsikan berada di direktori yang sama. File ini hanya dimuat sekali untuk mencegah duplikasi dan potensi error. File ini dapat berisi detail koneksi database atau fungsi lain yang dapat digunakan kembali.
- **$query = "SELECT * FROM employees order by id desc";**

Baris ini mendefinisikan query SQL yang akan dijalankan. Query ini memilih semua kolom (*) dari tabel employees dan mengurutkannya berdasarkan kolom id secara descending (desc).

- **$sql = mysqli_query($db_connect, $query);**
  Baris ini mengeksekusi query yang telah didefinisikan menggunakan fungsi mysqli_query. Diasumsikan $db_connect adalah variabel yang berisi koneksi database yang telah dibuat di helper.php.
- **if ($sql) { ... }**
  Pernyataan if ini memeriksa apakah eksekusi query berhasil. Jika berhasil ($sql tidak false):
  - **$result = array();**
    Array kosong bernama $result dibuat untuk menyimpan data yang diambil.
  - **while ($row = mysqli_fetch_array($sql)) { ... }**
    Perulangan while ini berlanjut selama ada baris yang dapat diambil dari hasil query ($sql).
    - **$row = mysqli_fetch_array($sql);**
      Baris ini mengambil baris berikutnya dari hasil query dan menetapkannya ke variabel $row. $row menjadi array asosiatif yang berisi nama kolom sebagai kunci dan nilai kolom sebagai elemen.
    - **array_push($result, array(...));**
      Baris ini menambahkan elemen baru ke array $result. Elemen baru adalah array asosiatif dengan kunci (id, name, email, position, dan salary) dan nilai yang sesuai diekstrak dari baris saat ini menggunakan $row['nama_kolom'].
- **echo json_encode (array('result' => $result));**
  Baris ini mengkodekan array $result (yang berisi data karyawan) menjadi string JSON menggunakan fungsi json_encode. String JSON dibungkus dalam array dengan kunci result untuk struktur yang lebih baik.

Create.php

create.php > ...

```php
<?php

    require_once('helper.php');

    $name = $_POST['name'];
    $email = $_POST['email'];
    $position = $_POST['position'];
    $salary = $_POST['salary'];

    $query = "INSERT INTO employees (name, email, position, salary) VALUES ('$name', '$email', '$position', '$salary')";
    $sql = mysqli_query($db_connect, $query);

    if ($sql) {
        echo json_encode (array('message' => 'Employee Added'));
    } else {
        echo json_encode (array('message' => 'Employee Not Added'));
    }

?>
```

Kode PHP tersebut berfungsi untuk menambahkan data karyawan baru ke dalam database dan memberikan respon JSON berdasarkan keberhasilan prosesnya.

- **require_once('helper.php');**
  Baris ini memuat file helper.php yang diasumsikan berada di direktori yang sama. File ini dimuat sekali untuk mencegah duplikasi dan potensi error. File ini mungkin berisi detail koneksi database atau fungsi lain yang dapat digunakan kembali.
- **`$name = $_POST['name'];`,`$email = $_POST['email'];`,`$position = $_POST['position'];`,`$salary = $_POST['salary'];`**
  Baris-baris tersebut bertujuan untuk mengambil nilai yang dikirim melalui metode POST dari input form dengan nama masing-masing 'name', 'email', 'position', dan 'salary', kemudian menyimpan nilai-nilai tersebut ke dalam variabel yang sesuai: $name, $email, $position, dan $salary.
- **$query = "INSERT INTO employees (name, email, position, salary) VALUES ('$name', '$email', '$position', '$salary')";**
  Baris tersebut bertujuan untuk membangun sebuah query SQL untuk menambahkan data karyawan baru ke dalam tabel 'employees'. Ini dilakukan dengan menggunakan perintah SQL 'INSERT INTO', diikuti dengan nama tabel 'employees' dan daftar kolom (name, email, position, salary) di dalam kurung.
  Kemudian, dengan kata kunci 'VALUES', kita menentukan nilai-nilai yang akan dimasukkan ke dalam masing-masing kolom. Variabel $name, $email, $position, dan $salary menyimpan data yang akan dimasukkan ke dalam query.
  Dengan menggunakan tanda kutip tunggal di sekitar variabel-variabel tersebut ('$name', '$email', '$position', '$salary'), kita mengindikasikan bahwa nilai-nilai ini harus disisipkan secara langsung ke dalam query.
- **$sql = mysqli_query($db_connect, $query);**
  Baris ini mengeksekusi query yang telah dibangun menggunakan fungsi mysqli_query. Diasumsikan $db_connect adalah variabel yang berisi koneksi database yang telah dibuat di helper.php.
- **if ($sql) { ... } else { ... }**
  Pernyataan if ini memeriksa apakah eksekusi query berhasil ($sql tidak false).
  Jika query berhasil dieksekusi:
  - **echo json_encode (array('message' => 'Employee Added'));**
    Kode ini mengkodekan array yang berisi pesan "Employee Added" ke format JSON dan menampilkannya.

  Jika query gagal dieksekusi:
  - **echo json_encode (array('message' => 'Employee Not Added'));**
    Kode ini mengkodekan array yang berisi pesan "Employee Not Added" ke format JSON dan menampilkannya.

Update.php

```php
<?php
require_once('helper.php');
parse_str(file_get_contents('php://input'), $value);

$id = $value['id'];
$name = isset($value['name']) ? $value['name'] : null;
$email = isset($value['email']) ? $value['email'] : null;
$position = isset($value['position']) ? $value['position'] : null;
$salary = isset($value['salary']) ? $value['salary'] : null;

$query = "UPDATE employees SET ";
$updateArray = array();

if (!is_null($name)) {
    $updateArray[] = "name='$name'";
}
if (!is_null($email)) {
    $updateArray[] = "email='$email'";
}
if (!is_null($position)) {
    $updateArray[] = "position='$position'";
}
if (!is_null($salary)) {
    $updateArray[] = "salary='$salary'";
}

$updateStr = implode(", ", $updateArray);

$query .= $updateStr . " WHERE id='$id'";

$sql = mysqli_query($db_connect, $query);

if ($sql) {
    echo json_encode(array('message' => 'Employee Updated'));
} else {
    echo json_encode(array('message' => 'Employee Not Updated'));
}
```

Kode ini memproses data yang diterima melalui request, membangun query update secara dinamis berdasarkan bidang yang disediakan, dan memberikan respon JSON yang menunjukkan keberhasilan atau kegagalan.

- **require_once('helper.php');**
  Baris ini memuat file helper.php sekali, diasumsikan file ini berisi detail koneksi database dan fungsi lain yang dapat digunakan kembali.
- **parse_str(file_get_contents('php://input'), $value);**
  Baris ini mengambil data dari request body (biasanya dikirim melalui metode POST atau PUT) dan menguraikannya ke dalam array asosiatif bernama $value.
- **$id = $value['id'];**
  Mengekstrak nilai id secara langsung.
- **$name = isset($value['name']) ? $value['name'] : null;**
  Memeriksa apakah kunci name ada di array $value. Jika ya, nilainya ditugaskan ke $name, jika tidak, $name disetel ke null. Pola ini diulang untuk bidang lain.
- **$query = "UPDATE employees SET ";**
  Baris ini menginisialisasi string query update.
- **$updateArray = array();**
  Array kosong bernama $updateArray dibuat untuk menyimpan klausa update.
- Kode kemudian mengulangi melalui bidang update potensial (name, email, position, dan salary):

Untuk setiap bidang, jika variabel yang sesuai ($name, $email, dll.) tidak null, string yang mewakili klausa update ditambahkan ke $updateArray:

- **$updateArray[] = "name='$name'";**
  Ini membangun string seperti "name='$name'", yang akan digunakan nanti.

- **$updateStr = implode(", ", $updateArray);**
  Baris ini menggabungkan elemen dalam $updateArray dengan koma dan spasi, membentuk string klausa update final.
- **$query .= $updateStr . " WHERE id='$id'";**
  Baris ini menambahkan $updateStr (berisi klausa update) dan klausa "WHERE" dengan $id yang disediakan ke string query utama.
- **$sql = mysqli_query($db_connect, $query);**
  Baris ini mengeksekusi query update yang dibuat menggunakan mysqli_query. Diasumsikan $db_connect menyimpan koneksi database yang dibuat di helper.php.
- **if ($sql) { ... } else { ... }**
  Pernyataan kondisional ini memeriksa hasil eksekusi query:
  Jika query berhasil:
  - **echo json_encode(array('message' => 'Karyawan Diperbarui'));**
    Mengkodekan pesan sukses dan menggemakannya sebagai string JSON.
    Jika query gagal:
  - **echo json_encode(array('message' => 'Karyawan Tidak Diperbarui'));**
    Mengkodekan pesan gagal dan menggemakannya sebagai string JSON.

Delete.php

```php
<?php

    require_once('helper.php');
    parse_str(file_get_contents('php://input'), $value);

    $id = $value['id'];

    $query = "DELETE FROM employees WHERE id='$id'";
    $sql = mysqli_query($db_connect, $query);

    if ($sql) {
        echo json_encode (array('message' => 'Employee Deleted'));
    } else {
        echo json_encode (array('message' => 'Employee Not Deleted'));
    }

?>
```

Script ini berfungsi untuk penghapusan data karyawan dari tabel "employees" berdasarkan nilai "id" yang disediakan yang dikirim melalui request body. Ini menangani proses koneksi ke database, membangun query, mengeksekusinya, dan memberikan respon JSON yang menunjukkan keberhasilan atau kegagalan.

- **require_once('helper.php');**
  Baris ini memuat file helper.php sekali, diasumsikan file ini berisi detail koneksi database dan fungsi lain yang dapat digunakan kembali.
- **parse_str(file_get_contents('php://input'), $value);**

Baris ini mengambil data dari request body (biasanya dikirim melalui metode DELETE) dan menguraikannya ke dalam array asosiatif bernama $value.

- **$id = $value['id'];**
  Baris ini mengekstrak nilai id dari array $value, mengasumsikan request berisi data dengan kunci bernama "id".
- **$query = "DELETE FROM employees WHERE id='$id'";**
  Baris ini membangun string query delete. DELETE FROM employees digunakan untuk menentukan operasi dan tabel target, sedangkan WHERE id='$id' digunakan untuk menyaring penghapusan berdasarkan id yang disediakan.
- **$sql = mysqli_query($db_connect, $query);**
  Baris ini mengeksekusi query delete menggunakan mysqli_query. Diasumsikan $db_connect menyimpan koneksi database yang dibuat di helper.php.
- **if ($sql) { ... } else { ... }**
  Pernyataan kondisional ini memeriksa hasil eksekusi query:
  Jika query berhasil:
  - **echo json_encode(array('message' => 'Karyawan Dihapus'));**
    Mengkodekan pesan sukses dan menggemakannya sebagai string JSON.

  Jika query gagal:
  - **echo json_encode(array('message' => 'Karyawan Tidak Dihapus'));**
    Mengkodekan pesan gagal dan menggemakannya sebagai string JSON.

Index.php

index.php > html > head > style > .employee

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Employees</title>
<style>
  body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
  }
  .container {
    max-width: 800px;
    margin: 20px auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  }
  h1 {
  text-align: center;
  color: #333;
  border-bottom: 5px solid #333;
  display: inline-block;
  padding-bottom: 15px;
  width: 100%;
  box-sizing: border-box;
}
```

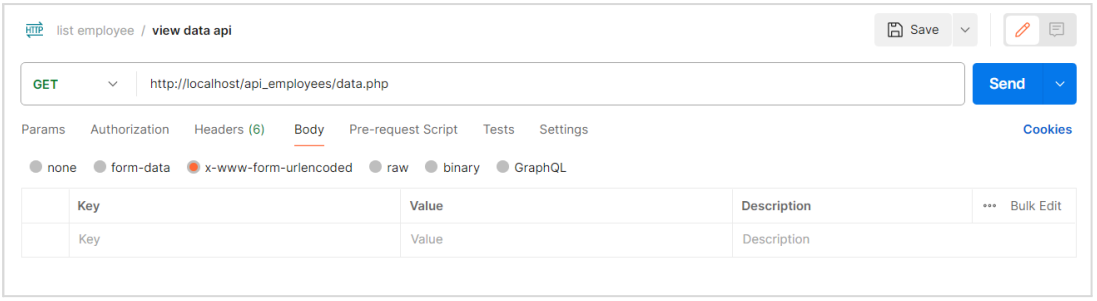
```
  .employee {
    background-color: #f9f9f9;
    margin-bottom: 10px;
    padding: 10px;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    display: flex;
    justify-content: space-between;
    align-items: center;
  }
  .employee-details {
    flex: 1;
  }
  .view-button {
    background-color: #007bff;
    color: #fff;
    border: none;
    padding: 5px 10px;
    border-radius: 3px;
    cursor: pointer;
  }
</style>
</head>
<body>
<div class="container">
  <h1>Employee List</h1>
  <div id="employee-list"></div>
</div>

<script>
fetch('http://localhost/api_employees/data.php')
  .then(response => response.json())
  .then(data => {
    const employeeList = document.getElementById('employee-list');
    data.result.forEach(employee => {
      const div = document.createElement('div');
      div.classList.add('employee');
      div.innerHTML = `
        <div class="employee-details">
          <h3>${employee.name}</h3>
          <p>${employee.position}</p>
        </div>
        <button class="view-button" onclick="viewEmployee(${employee.id})">View</button>
      `;
      employeeList.appendChild(div);
    });
  });

function viewEmployee(id) {
  window.location.href = `http://localhost/api_employees/data.php`;
}
</script>
</body>
</html>
```

Kode di atas adalah sebuah halaman HTML yang menampilkan daftar karyawan dari data yang diambil dari sebuah API.

- **`<!DOCTYPE html>`:** Mendefinisikan tipe dokumen HTML yang digunakan.
- **`<html lang="en">`:** Menandakan awal dari dokumen HTML dengan menentukan bahasa yang digunakan (dalam hal ini bahasa Inggris).
- **`<head>`:** Bagian kepala dokumen HTML yang berisi informasi tentang dokumen seperti meta tag, judul, dan link ke file eksternal.
    - **`<meta charset="UTF-8">`:** Mendefinisikan set karakter yang digunakan dalam dokumen sebagai UTF-8.

    - **`<meta name="viewport" content="width=device-width, initial-scale=1.0">`:** Mendefinisikan pengaturan tampilan untuk perangkat mobile.
    - **`<title>Employees</title>`:** Menetapkan judul halaman.
    - **`<style>`:** Menyediakan CSS internal untuk mengatur tata letak dan tampilan elemen-elemen di halaman.
- **`<body>`:** Bagian tubuh dokumen HTML yang berisi konten yang akan ditampilkan di dalam halaman.
    - **`<div class="container">`**: Sebuah div yang menampung konten halaman dan memiliki beberapa gaya CSS untuk tata letak dan penampilan.
    - **`<h1>Employee List</h1>`:** Heading utama yang menampilkan judul halaman.
    - **`<div id="employee-list"></div>`:** Tempat di mana daftar karyawan akan ditampilkan. Ini adalah sebuah div kosong yang akan diisi dengan data dari API menggunakan JavaScript.
    - **`<script>`:** Mendefinisikan script JavaScript yang digunakan untuk mengambil data dari API dan menampilkan informasi karyawan.
    - **`fetch('http://localhost/api_employees/data.php')...`:** Mengambil data dari API menggunakan metode Fetch API.
    - **`function viewEmployee(id)...`:** Sebuah fungsi JavaScript yang dipanggil saat tombol "View" ditekan. Ini mengarahkan pengguna ke halaman detail karyawan dengan menggunakan id karyawan yang dipilih.
- Penutup tag **`</body>` dan `</html>`** menandakan akhir dari dokumen HTML.
- Dalam kode tersebut, ada juga beberapa gaya CSS yang mengatur tampilan elemen-elemen di halaman, seperti tata letak, warna, dan ukuran teks.

GET- Postman

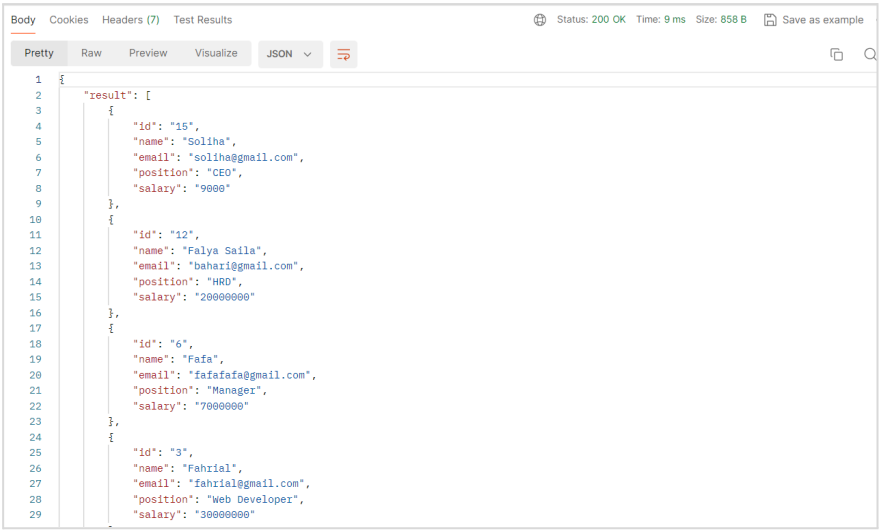


REST API GET digunakan untuk mengambil data dari server. Dalam konteks penggunaan Postman dengan URL localhost/api_employees/data.php, REST API tersebut dapat digunakan untuk mengambil data pegawai dari database.

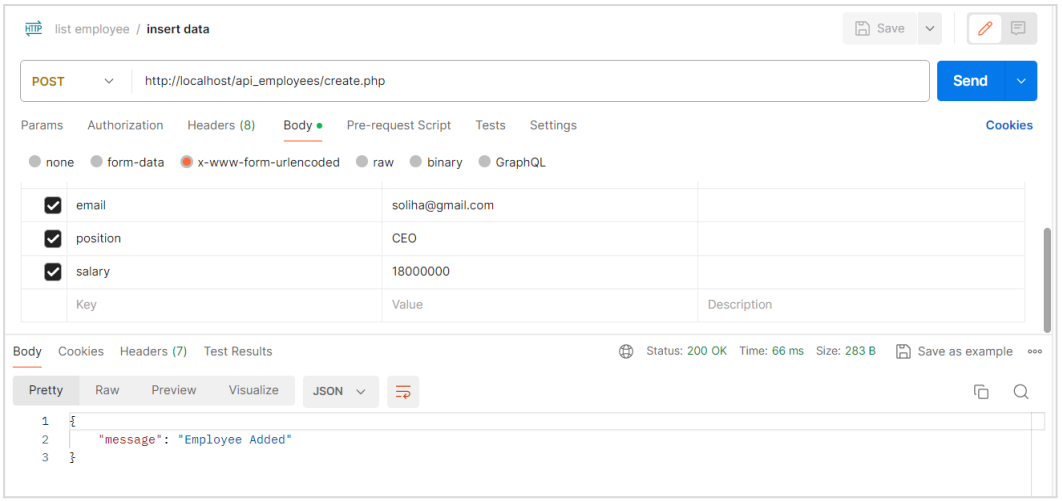
1. Pastikan server lokal Anda berjalan dan dapat diakses dengan menggunakan URL localhost.
2. Buatlah request GET baru di Postman dengan URL lengkap localhost/api_employees/data.php.
3. Ketika request GET dikirimkan, skrip PHP di data.php akan dijalankan. Skrip ini kemudian akan berinteraksi dengan database (yang telah dihubungkan sebelumnya menggunakan koneksi MySQLi) untuk mengambil data pegawai dari tabel employees.
4. Setelah itu, data tersebut akan diubah menjadi format JSON dan dikirimkan kembali ke Postman sebagai respons.

Di Postman, akan melihat respons dalam bentuk JSON yang berisi daftar pegawai beserta informasi mereka seperti nama dan posisi. Dengan menggunakan REST API GET, dapat dengan mudah mengambil data dari server dan menggunakannya dalam aplikasi atau layanan lain yang Anda buat.

Hasil:



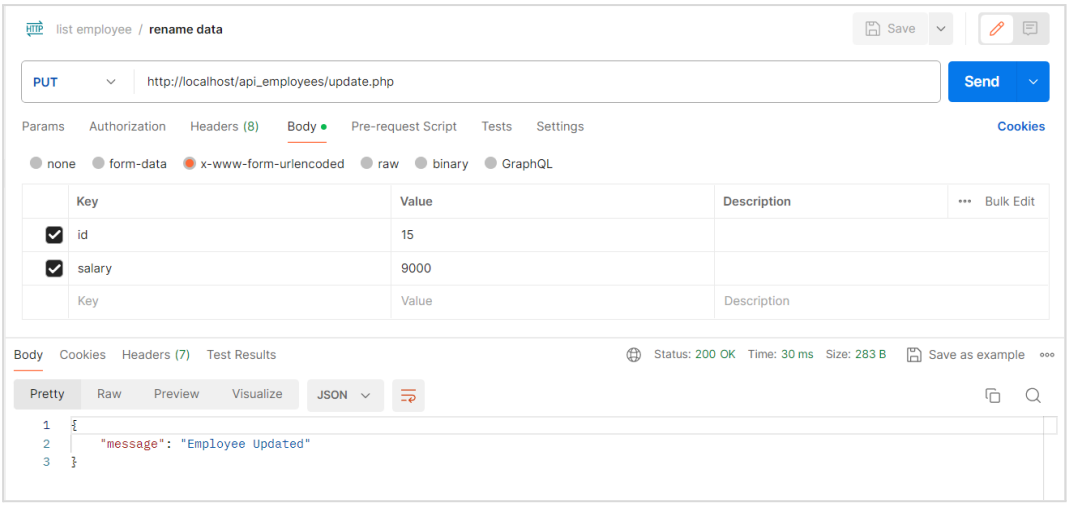
POST- Postman



REST API POST menggunakan Postman dengan URL localhost/api_employees/create.php digunakan untuk menambahkan data pegawai ke dalam database.

1. Pertama, pastikan server PHP Anda berjalan dan terhubung dengan database.
2. Kemudian, buka Postman dan atur metode request menjadi POST. Pada bagian URL, masukkan localhost/api_employees/create.php.
3. Selanjutnya, pilih tab "Body" di Postman, kemudian pilih format x-www-form-urlencoded. Masukkan key dan value untuk setiap data pegawai yang ingin ditambahkan, seperti name, email, position, dan salary.
4. Klik tombol "Send" untuk mengirimkan request ke API. Jika request berhasil, Anda akan menerima response berupa data pegawai yang telah ditambahkan dalam format JSON beserta status code 200. Jika terdapat kesalahan, akan menerima response dengan status code lain beserta pesan error yang dapat membantu untuk memperbaiki masalah.

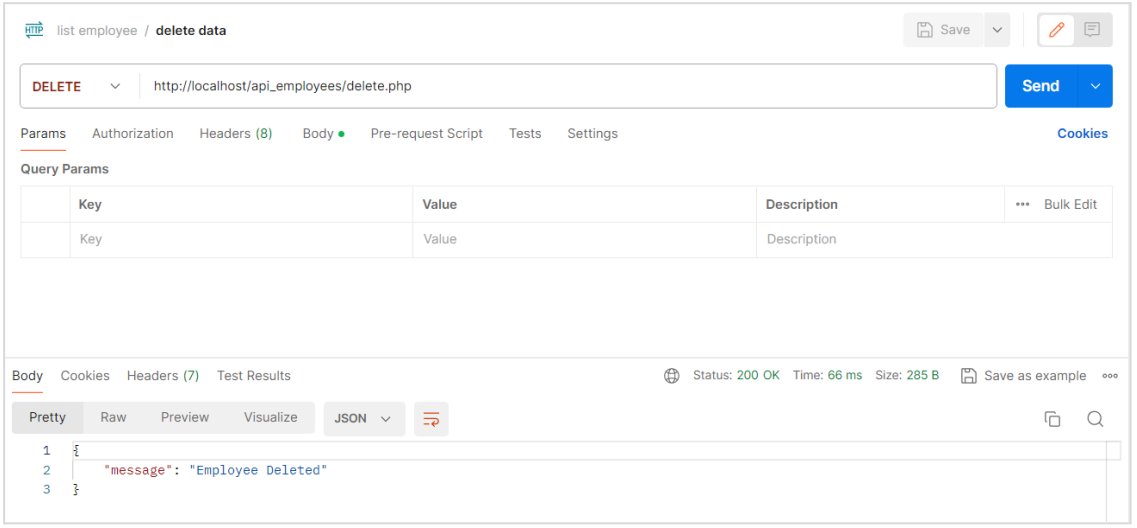
PUT - Postman



REST API PUT digunakan untuk memperbarui data yang ada di server. Dengan menggunakan Postman dan URL localhost/api_employees/update.php, dapat mengakses endpoint update.php pada API dengan base URL localhost/api_employees untuk melakukan operasi PUT. Untuk menggunakan REST API PUT dengan Postman, langkah-langkahnya adalah sebagai berikut:

1. Pertama, buka Postman dan pilih metode PUT.
2. Selanjutnya, masukkan URL localhost/api_employees/update.php sebagai endpoint. Kemudian, pilih tab "Body" di Postman dan pilih format "x-www-form-urlencoded".
3. Setelah itu, masukkan data yang ingin diperbarui dalam format yang sesuai dengan format yang dipilih. Terakhir, kirimkan permintaan PUT.

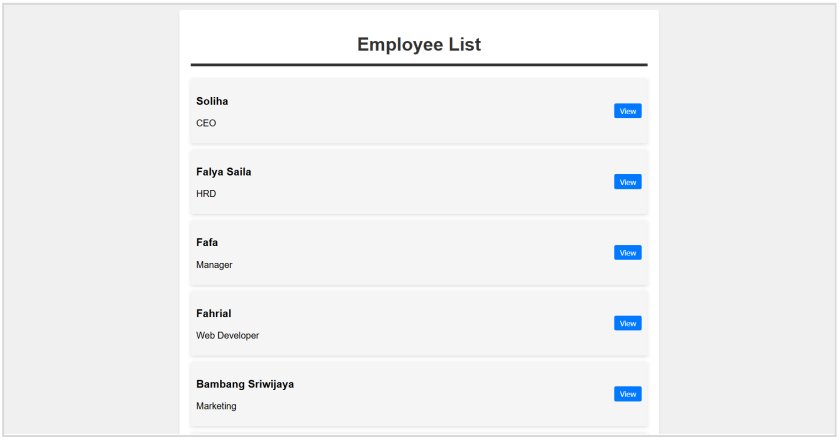
DELETE - Postman

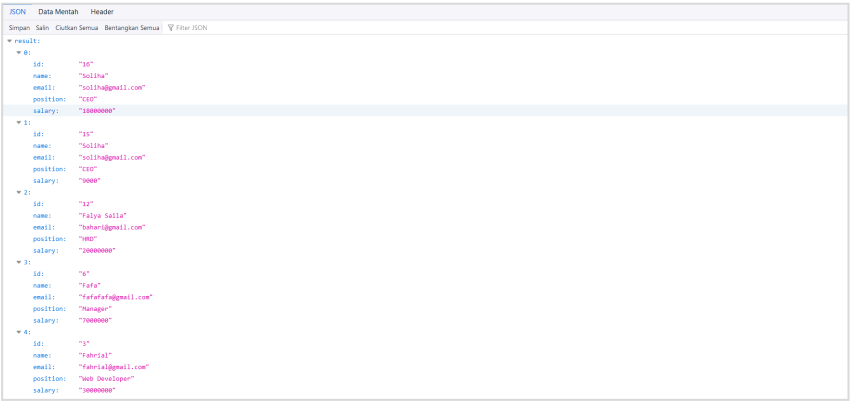


REST API delete menggunakan Postman memungkinkan untuk menghapus data dari server menggunakan metode HTTP DELETE. Untuk menggunakan fitur ini, perlu mengirimkan permintaan DELETE ke URL yang sesuai dengan data yang ingin dihapus.

1. Pertama, pastikan server Anda telah diatur untuk menanggapi permintaan DELETE dengan benar.
2. Kemudian, buat permintaan DELETE di Postman dengan URL http://localhost/api_employees/delete.php dan pastikan untuk mengirimkan data yang sesuai untuk mengidentifikasi data yang akan dihapus, misalnya, ID pegawai.
3. Setelah itu, kirim permintaan tersebut dengan menggunakan Postman. Jika data berhasil dihapus, server akan memberikan respons yang sesuai, misalnya, pesan sukses atau status kode 200. Jika terjadi masalah, server akan memberikan respons yang sesuai, misalnya, pesan error atau status kode 404.

Tampilan Website





Link GitHub:
[berlinmrsyh/IMPLEMENTASI-REST-API (github.com)]