What is REST?

1. REPRESENTATIONAL STATE TRANSFER

- REST, atau REpresentational State Transfer, adalah gaya arsitektur untuk menyediakan standar antara sistem komputer di web, sehingga memudahkan sistem untuk berkomunikasi satu sama lain
- Sistem RESTcompliant, sering disebut sistem RESTful, dicirikan oleh bagaimana mereka bersifat stateless (tanpa status) dan memisahkan kepentingan antara klien dan server.

2. SEPARATION OF CLIENT AND SERVER

- Dalam gaya arsitektur REST, implementasi klien dan implementasi server dapat dilakukan secara independen tanpa saling mengetahui satu sama lain.
- Ini berarti kode di sisi klien dapat diubah kapan saja tanpa memengaruhi pengoperasian server, dan kode di sisi server dapat diubah tanpa memengaruhi operasi klien.
- Selama masing-masing pihak mengetahui format pesan apa yang akan dikirim ke pihak lain, pesan tersebut dapat dibuat secara modular dan terpisah
- Dengan memisahkan masalah antarmuka pengguna dari masalah penyimpanan data, akan meningkatkan fleksibilitas antarmuka di seluruh platform dan meningkatkan skalabilitas dengan menyederhanakan komponen server
- Dengan menggunakan antarmuka REST, klien yang berbeda mencapai titik akhir REST yang sama, melakukan tindakan yang sama, dan menerima respons yang sama.

3. STATELESSNESS

- Sistem yang mengikuti paradigma REST tidak memiliki status/state,artinya server tidak perlu mengetahui apa pun tentang status klien dan sebaliknya.
- Batasan keadaan stateless ini diterapkan melalui penggunaan sumber daya, bukan melalui perintah.
- Sumber daya adalah kata benda di Web - sumber daya mendeskripsikan objek, dokumen, atau hal apa pun yang mungkin perlu Anda simpan atau kirim ke layanan lain.
- Karena sistem REST berinteraksi melalui operasi standar pada sumber daya, mereka tidak bergantung pada implementasi antarmuka.

4. COMMUNICATION BETWEEN CLIENT AND SERVER

- Dalam arsitektur REST, klien mengirim permintaan untuk mengambil atau mengubah sumber daya, dan server mengirim respons ke permintaan ini

5. MAKING REQUESTS

- REST mengharuskan klien membuat permintaan/request ke server untuk mengambil atau mengubah data di server.
- Permintaan biasanya terdiri dari:
  - an HTTP verb, yang mendefinisikan jenis operasi apa yang harus dilakukan
  - a header, yang memungkinkan klien untuk menyampaikan informasi tentang permintaan tersebut
  - a path to a resource, alamat dari sumber daya
  - an optional message body containing data, pesan tambahan

5.1. HTTP VERBS

- Ada 4 kata kerja (Verbs) HTTP dasar yang dapat kita gunakan dalam permintaan untuk berinteraksi dengan sumber daya dalam sistem REST:
  - GET - ambil sumber daya tertentu (menurut id) atau kumpulan sumber daya
  - POST - buat sumber daya baru
  - PUT - perbarui sumber daya tertentu (menurut id)
  - HAPUS - hapus sumber daya tertentu dengan id

5.2. HEADERS AND ACCEPT PARAMETERS

- Di header permintaan, klien mengirimkan jenis konten yang dapat diterima dari server. Opsi untuk tipe konten adalah MIME Type.
- MIME Types, digunakan untuk menentukan tipe konten pada field Accept, terdiri dari type dan subtype. Contoh:
  - Text – text/html, text/css, text/plain
  - image — image/png, image/jpeg, image/gif
  - audio — audio/wav, image/mpeg
  - video — video/mp4, video/ogg
  - application — application/json, application/pdf, application/xml, application/octet-stream
- Contoh header: klien mengakses sumber daya artikel dengan Id 23

```
GET /articles/23
Accept: text/html, application/xhtml
```

5.3. PATHS

- Permintaan harus berisi jalur/paths ke sumber daya tempat operasi harus dilakukan. Di RESTful API, jalur harus dirancang untuk membantu klien mengetahui apa yang sedang terjadi.
- Contoh:
  - POST **fashionboutique.com/customers** – menambah sumber daya baru
  - GET **fashionboutique.com/customers/:id** – mengambil item di sumber daya customer sesuai dengan id
  - DELETE **fashionboutique.com/customers/:id** – menghapus item di sumber daya customer sesuai dengan id

6. SENDING RESPONSES
6.1. CONTENT TYPES/JENIS KONTEN

- Dalam kasus di mana server mengirimkan muatan data ke klien, server harus menyertakan file Jenis konten di header tanggapan. Field Jenis konten pada header memberi tahu klien tentang jenis data yang dikirimnya pada isi respons.
- Jenis konten ini adalah MIME TYPES, sama seperti yang tertera pada field Accept di header request.

For example, when a client is accessing a resource with id 23 in an articles resource with this GET Request:

```
GET /articles/23 HTTP/1.1
Accept: text/html, application/xhtml
```

The server might send back the content with the response header:

```
HTTP/1.1 200 (OK)
Content-Type: text/html
```

6.2. RESPONSE CODES

- Tanggapan dari server berisi kode status untuk mengingatkan klien tentang informasi keberhasilan operasi.
- Beberapa response code yang umum:

Status code	Meaning
200 (OK)	This is the standard response for successful HTTP requests.
201 (CREATED)	This is the standard response for an HTTP request that resulted in an item being successfully created.
204 (NO CONTENT)	This is the standard response for successful HTTP requests, where nothing is being returned in the response body.
400 (BAD REQUEST)	The request cannot be processed because of bad request syntax, excessive size, or another client error.
403 (FORBIDDEN)	The client does not have permission to access this resource.
404 (NOT FOUND)	The resource could not be found at this time. It is possible it was deleted, or does not exist yet.
500 (INTERNAL SERVER ERROR)	The generic answer for an unexpected failure if there is no more specific information available.

6.2.1. Expected Status codes

- Untuk setiap kata kerja/Verbs HTTP , ada kode status yang diharapkan yang harus dikembalikan server setelah berhasil:
  - GET — return 200 (OK)
  - POST — return 201 (CREATED)
  - PUT — return 200 (OK)
  - DELETE — return 204 (NO CONTENT) If the operation fails, return the most specific status code possible corresponding to the problem that was encountered.

EXAMPLES OF REQUESTS AND RESPONSES >> VIEW All Customers

If we wanted to view all customers, the request would look like this:

```
GET http://fashionboutique.com/customers
Accept: application/json
```

A possible response header would look like:

```
Status Code: 200 (OK)
Content-type: application/json
```

followed by the customers data requested in application/json format.

EXAMPLES OF REQUESTS AND RESPONSES >> CREATE a new customer

Create a new customer by posting the data:

```
POST http://fashionboutique.com/customers
Body:
{
  "customer": {
    "name" = "Scylla Buss"
    "email" = "scylla.buss@codecademy.org"
  }
}
```

The server then generates an id for that object and returns it back to the client, with a header like:

```
201 (CREATED)
Content-type: application/json
```

EXAMPLES OF REQUESTS AND RESPONSES >> VIEW a single customer

To view a single customer we GET it by specifying that customer's id:

```
GET http://fashionboutique.com/customers/123
Accept: application/json
```

A possible response header would look like:

```
Status Code: 200 (OK)
Content-type: application/json
```

followed by the data for the customer resource with id 23 in application/json format.

EXAMPLES OF REQUESTS AND RESPONSES >> UPDATE a single customer

We can update that customer by _PUT_ the new data:

```
PUT http://fashionboutique.com/customers/123
Body:
{
  "customer": {
    "name" = "Scylla Buss"
    "email" = "scyllabuss1@codecademy.com"
  }
}
```

A possible response header would have Status Code: 200 (OK), to notify the client that the item with id 123 has been modified.

EXAMPLES OF REQUESTS AND RESPONSES >> DELETE a single customer

We can also DELETE that customer by specifying its id:

```
DELETE http://fashionboutique.com/customers/123
```

The response would have a header containing Status Code: 204 (NO CONTENT), notifying the client that the item with id 123 has been deleted, and nothing in the body.