Kriteria Proyek

Seperti yang sudah Anda ketahui, kita akan membangun RESTful API untuk aplikasi catatan sederhana. Di mana aplikasi tersebut berfungsi untuk menyimpan (create), melihat (read), mengubah (update), dan menghapus (delete) catatan. Fungsionalitas ini dikenal sebagai operasi CRUD.

Dari segi Front-End (client), kami telah membuat aplikasi web-nya. Kami juga telah men-deploy aplikasi tersebut sehingga Anda dapat mengaksesnya melalui tautan ini: http://notesapp-v1.dicodingacademy.com/.

Namun ketika Anda mengaksesnya, aplikasi tersebut belum bisa digunakan. Anda tidak bisa melihat, dan menambahkan catatan apapun. Tetapi percayalah, aplikasi tersebut akan berfungsi dengan baik ketika Anda sudah membuat RESTful API sesuai dengan kriteria yang dibutuhkan. Lantas apa saja kriterianya?

Kriteria 1 - Web Server dapat menyimpan catatan

Kriteria pertama adalah web server dapat menyimpan catatan yang ditambahkan melalui aplikasi web. Tenang, untuk memenuhi kriteria ini Anda tidak perlu menggunakan database. Cukup simpan pada memory server dalam bentuk array JavaScript.

Berikut struktur dari objek catatan yang perlu disimpan oleh server:

```
{
  id: string,
  title: string,
  createdAt: string,
  updatedAt: string,
  tags: array of string,
  body: string,
},
```

Berikut contoh data nyatanya:

```
{
  id: 'notes-V1StGXR8_Z5jdHi6B-myT',
  title: 'Sejarah JavaScript',
  createdAt: '2020-12-23T23:00:09.686Z',
  updatedAt: '2020-12-23T23:00:09.686Z',
  tags: ['NodeJS', 'JavaScript'],
  body: 'JavaScript pertama kali dikembangkan oleh Brendan Eich dari Netscape di bawah nama Mocha, yang nantinya naman
},
```

Agar web server dapat menyimpan catatan melalui aplikasi client, web server harus menyediakan route dengan path '`/notes`' dan method `POST`.

Dalam menyimpan atau menambahkan notes, client akan mengirimkan permintaan ke path dan method tersebut dengan membawa data JSON berikut pada request body:

```
  "title": "Judul Catatan",
  "tags": ["Tag 1", "Tag 2"],
  "body": "Konten catatan"
}
```

tuk properti `id`, `createdAt`, dan `updatedAt` harus diolah di sisi server, jadi client tidak akan mengirimkan itu. rver harus memastikan properti `id` selalu unik.

Jika permintaan client berhasil dilakukan, respons dari server harus memiliki status code **201 (created)** dan mengembalikan data dalam bentuk JSON dengan format berikut:

```
{
  "status": "success",
  "message": "Catatan berhasil ditambahkan",
  "data": {
    "noteId": "V09YExygSUYogwWJ"
  }
}
```

Nilai dari properti `noteId` diambil dari properti `id` yang dibuat secara unik.

Bila permintaan gagal dilakukan, berikan status code **500** dan kembalikan dengan data JSON dengan format berikut:

```
{
  "status": "error",
  "message": "Catatan gagal untuk ditambahkan"
}
```

Kriteria 2 - Web Server dapat menampilkan catatan

Kriteria selanjutnya adalah web server dapat menampilkan catatan. Kriteria ini mengharuskan web server untuk mengirimkan seluruh atau secara spesifik data notes yang disimpan.

Ketika client melakukan permintaan pada path '`/notes`' dengan method '`GET`', maka server harus mengembalikan status code 200 (ok) serta seluruh data notes dalam bentuk array menggunakan JSON. Contohnya seperti ini:

```
  "status": "success",
  "data": {
    "notes": [
      {
        "id":"notes-V1StGXR8_Z5jdHi6B-myT",
        "title":"Catatan 1",
        "createdAt":"2020-12-23T23:00:09.686Z",
        "updatedAt":"2020-12-23T23:00:09.686Z",
        "tags":[
          "Tag 1",
          "Tag 2"
        ],
        "body":"Isi dari catatan 1"
      },
```

Jika belum ada catatan satu pun pada array, server bisa mengembalikan data notes dengan nilai array kosong seperti ini:

```
{
  "status": "success",
  "data": {
    "notes": []
  }
}
```

Selain itu, client juga bisa melakukan permintaan untuk mendapatkan catatan secara spesifik menggunakan id melalui path ‘ /notes/{id} ’ dengan method ‘GET’. Server harus mengembalikan status code **200** (ok) serta nilai satu objek catatan dalam bentuk JSON seperti berikut:

```
{
  "status": "success",
  "data": {
    "note": {
      "id":"notes-V1StGXR8_Z5jdHi6B-myT",
      "title":"Catatan 1",
      "createdAt":"2020-12-23T23:00:09.686Z",
      "updatedAt":"2020-12-23T23:00:09.686Z",
      "tags":[
        "Tag 1",
        "Tag 2"
      ],
      "body":"Isi dari catatan 1"
    }
```

```
{
  "status": "fail",
  "message": "Catatan tidak ditemukan"
}
```

Kriteria 3 - Web Server dapat mengubah catatan

Kriteria ketiga adalah web server harus dapat mengubah catatan. Perubahan yang dimaksud bisa berupa judul, isi, ataupun tag catatan. Ketika client meminta perubahan catatan, ia akan membuat permintaan ke path '`/notes/{id}`', menggunakan method '`PUT`', serta membawa data JSON pada body request yang merupakan data catatan terbaru.

```
{
  "title":"Judul Catatan Revisi",
  "tags":[
    "Tag 1",
    "Tag 2"
  ],
  "body":"Konten catatan"
}
```

Jika perubahan data berhasil dilakukan, server harus menanggapi dengan status code **200 (ok)** dan membawa data JSON objek berikut pada body respons.

```
{
  "status": "success",
  "message": "Catatan berhasil diperbaharui"
}
```

Perubahan data catatan harus disimpan ke catatan yang sesuai dengan `id` yang digunakan pada path parameter. Bila id catatan tidak ditemukan, maka server harus merespons dengan status code `404` (not found) dan data JSON seperti ini:

```
{
  "status": "fail",
  "message": "Gagal memperbarui catatan. Id catatan tidak ditemukan"
}
```

Kriteria 4 - Web Server dapat menghapus catatan

Kriteria terakhir adalah web server harus bisa menghapus catatan. Untuk menghapus catatan, client akan membuat permintaan pada path '`/notes/{id}`' dengan method '`DELETE`'. Ketika permintaan tersebut berhasil, maka server harus mengembalikan status code **200** (ok) serta data JSON berikut:

```
  "status": "success",
  "message": "Catatan berhasil dihapus"
}
```

catan yang dihapus harus sesuai dengan id catatan yang digunakan client pada path parameter. Bila id catatan ak ditemukan, maka server harus mengembalikan respons dengan status code **404** dan membawa data JSON rikut:

```
{
  "status": "fail",
  "message": "Catatan gagal dihapus. Id catatan tidak ditemukan"
}
```

-- akhir dari penjelasan kriteria

Bagaimana, sudah jelas? Itulah kriteria yang perlu dipenuhi oleh kita dalam mengembangkan RESTful API nanti. Untuk memastikan apakah web server yang dibuat sudah bekerja sesuai dengan kriteria, Anda perlu mencoba menggunakan aplikasi client yang dihubungkan dengan web server. Bagaimana caranya? Kita akan bahas itu nanti yah.