CRUD API Design

about CRUD

- CRUD adalah akronim yang berasal dari dunia database.
- Setiap huruf mewakili satu jenis tindakan yang dapat dilakukan pengguna pada sekumpulan data: C reate, R ead, U pdate, dan D elete.
- Dalam database relasional, empat aktivitas sesuai dengan perintah SQL INSERT, SELECT, UPDATE dan DELETE.
- Konsep CRUD dapat diterapkan ke world wide web dan protokol yang mendasarinya yaitu HTTP.
- Setiap URL mengarah ke sumber daya, yaitu sekumpulan data.
- Kita dapat mengaksesnya dengan kata kerja HTTP yang berbeda: GET, PUT, POST dan DELETE.
- Banyak API mengikuti paradigma ini

URL Design Best Practices

- URL mendeskripsikan sumber daya sedangkan kata kerja HTTP menjelaskan tindakan yang dapat dilakukan pada Sumber Daya
- URL hanya boleh berisi kata benda, tidak pernah kata kerja, kecuali jika dengan sengaja menambahkan sesuatu ke API yang tidak sesuai dengan paradigma CRUD.
- Kata benda di URL sama dengan mendeskripsikan model domain kita.

URL Design Best Practices

Saat memodelkan domain, perhatikan hal berikut:

- Jangan membuat kata benda terlalu panjang dan rumit (*users*, bukan *system-users*).
- Gunakan terminologi umum (*people* bukan *special-snowflakes*).
- Gunakan huruf kecil (*/items* bukan */Items*) untuk menghindari ambiguitas seputar sensitivitas huruf besar / kecil URL.
- Untuk sumber daya individu, sertakan pengidentifikasi sumber daya di jalur, bukan kueri (*/contacts/22* bukan */contacts?id=22*).
- Jalur yang diakhiri dengan nama sumber daya (dan biasanya tanpa garis miring) digunakan untuk mencantumkan beberapa item (*/files*) atau membuat item tanpa menentukan id.

URL Design Best Practices

Saat memodelkan domain, perhatikan hal berikut:

- Jalur bisa menunjukkan hierarki subresources (*/contacts/22/addresses*), tetapi desainer API harus menghindari struktur kompleks yang membutuhkan lebih dari dua tingkat penumpukan.
- Kata benda harus dalam bentuk jamak (*/files* bukan */file*) kecuali jika mendeskripsikan benda yang hanya satu saja.
- Kata benda yang merupakan gabungan dari beberapa kata harus menggunakan tanda hubung sebagai pemisah (*/legal-documents*). Alasannya: ketika URL adalah hyperlink, mereka sering digarisbawahi, yang akan membuat pemisah garis bawah tidak terlihat.
- URL tidak boleh mengungkapkan implementasi yang mendasarinya (*/resources* bukan */api.php/resources*).
- Bagian kueri dari URL adalah untuk penelusuran dan pemfilteran dan biasanya digunakan dengan daftar *endpoint* sumber daya (*/contacts?first_name=Anna&limit=20*).

The HTTP Verbs

Menggunakan kata kerja HTTP dengan benar adalah salah satu landasan desain API yang baik, dan dasar-dasarnya tidak terlalu rumit.

- **GET** untuk membaca atau mengambil data, R di CRUD.
  - Ketika membuat permintaan GET yang sama dua kali akan mengembalikan respons yang sama (kecuali jika data pokok diubah untuk sementara waktu) dan juga aman karena tidak mengubah status sisi server.
  - Saat mendesain API, tidak boleh menggunakan GET untuk sesuatu yang dapat mengubah sumber daya di server.

The HTTP Verbs

- **DELETE** adalah D di CRUD dan menghapus satu atau lebih sumber daya dari server.

The HTTP Verbs

- **PUT** memengaruhi sumber daya yang diidentifikasi oleh URL.
  - Misalnya, PUT di */users/1* membuat pengguna dengan ID 1 atau mengubah pengguna yang sudah ada dengan ID ini.
  - Karena sebagian besar API menghasilkan id di server dan tidak mengizinkan pembuatan sumber daya dengan cara ini, PUT dibatasi untuk operasi memperbarui.
  - Ketika membuat permintaan PUT yang sama dua kali akan mengembalikan respons yang sama dan tidak berbahaya

The HTTP Verbs

- **POST** mengirim data ke server dan menyerahkannya ke server untuk melakukan sesuatu dengannya. Ini adalah kata kerja yang paling fleksibel.
  - Dalam konteks CRUD API biasanya digunakan untuk membuat sumber daya di mana URL sumber daya paling akhir/baru tidak diketahui.
  - Misalnya, POST untuk /users membuat item user dengan ID baru misalkan ID=1, yang kemudian dapat diambil dengan, misalnya /users/1.
  - Apabila melakukan permintaan berulang ke /users maka akan membuat pengguna baru dengan ID baru misalkan ID=2 diakses dengan /users/2.

Example CRUD API in Action

Small CRM (Custom Relationship Management) system for a company with the following requirements:

- *The CRM has contacts, and each contact has notes and emails attached to them.*
- *API consumers can create, read, update and delete contacts as well as notes.*
- *Access to emails via the API is read-only*

Resources

Based on those requirements, these are our resources:

- *All contacts: /contacts*
- *Individual contact: /contacts/{cID}*
- *Notes for contact: /contacts/{cID}/notes*
- *Individual note: /contacts/{cID}/notes/{nID}*
- *Emails for contact: /contacts/{cID}/emails*
- *Individual email: /email/{eID}*

Contacts Operations

For contacts, we can do all CRUD actions, so there'll be the following:

- *GET /contacts*
- *POST /contacts*
- *GET /contacts/{cID}*
- *PUT /contacts/{cID}*
- *DELETE /contacts/{cID}*

Notes Operations

The same approach applies to notes, except that we have longer URL with (sometimes) two placeholders:

- *GET /contacts/{cID}/notes*
- *POST /contacts/{cID}/notes*
- *GET /contacts/{cID}/notes/{nID}*
- *PUT /contacts/{cID}/notes/{nID}*
- *DELETE /contacts/{cID}/notes/{nID}*

Emails Operations

For emails, we have fewer operations as they are read-only:

- *GET /contacts/{cID}/emails*
- *GET /email/{eID}*