



Networking

Silver- Chapter 3- Topic 4

**Selamat datang di Chapter 3 Topic 4 pada online course
React Native dari Binar Academy!**





Haalloo teman-teman! 🙌

Di **Topic 2** kemarin, kita sudah belajar banyak tentang Eslint.
Di **Chapter 3 Topic 3** ini, kita bakal lanjut ke materi tentang networking dengan **RESTful API**.

Udah siap buat makin banyak tahu? Let's go!



Detailnya, kita bakal bahas hal-hal berikut ini:

- Pengertian RESTful API dan cara mengaksesnya
- Pengenalan dan penerapan JSON
- Pengertian dan aplikasi REST Client POSTMAN





Kalau dengar kata “**Networking**” apa sih yang ada di pikiran kamu, guys? Kalau dilihat dari artinya sih networking berarti jaringan atau koneksi.

Hmm tapi hubungan dan cara kerja “**Networking**” dalam aplikasi itu kayak gimana ya?





Nahh, untuk mengawali pembahasan tentang networking ini, kita harus mulai dari memahami **RESTful API** dan cara mengaksesnya yaa.

Langsung saja kita bahas yuks!





Guys, coba bayangin kalau aplikasi yang sudah kamu buat, tampilan kontennya statis alias yang itu-itu saja, nggak ganti-ganti. Duh, pasti bakal bikin pengunjung aplikasimu bosen deh~

Eh, tapi memangnya ada ya, cara supaya tampilan bisa jadi lebih dinamis?

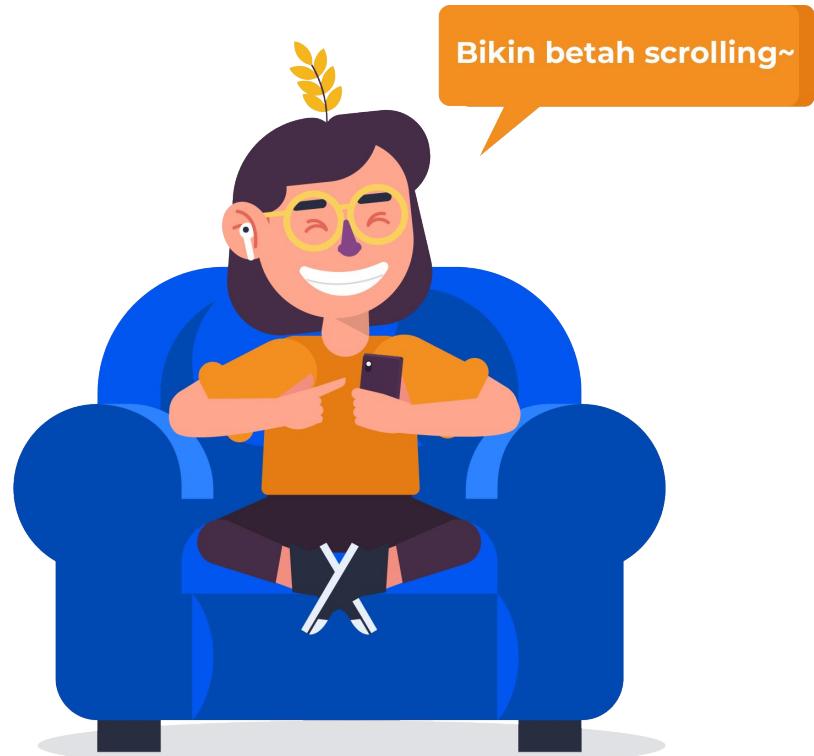
Yess, ada! Caranya adalah dengan mengintegrasikan aplikasi kita dengan server.





Biar eksis, kamu harus dinamis. Begini juga konten aplikasi~

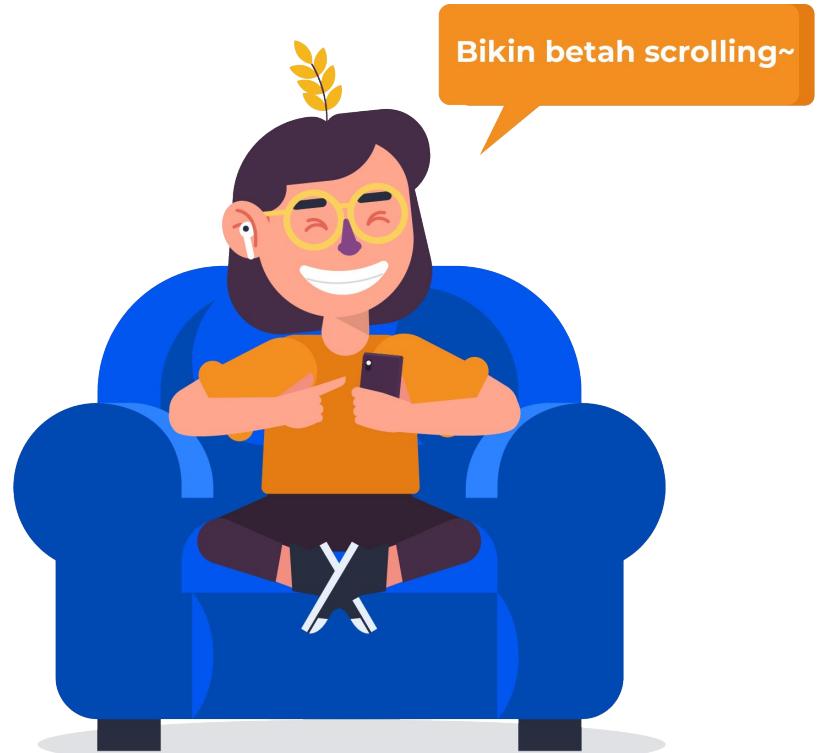
Mengintegrasikan aplikasi dengan server berfungsi untuk mengubah konten pada aplikasi jadi lebih dinamis (berubah-ubah), sesuai dengan data yang disimpan pada server.





Dengan adanya perubahan konten, aplikasimu akan tampak selalu ter-update sehingga pengunjung aplikasimu pun jadi tertarik melihat dan nggak bosen pas buka aplikasi.

Terus bagaimana cara melakukan integrasi aplikasi dengan server?



Kenalan dulu nih sama Restful API

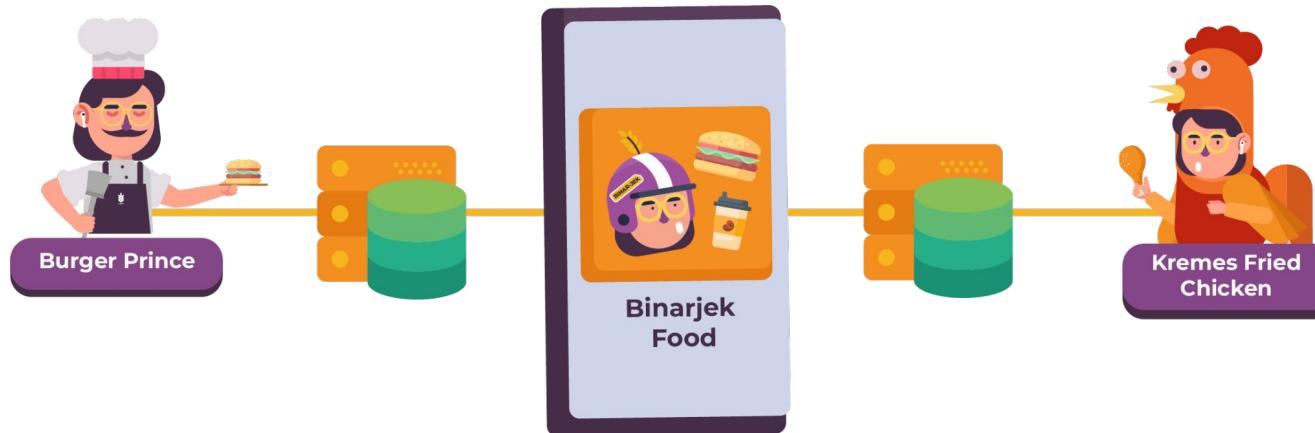
Ini adalah sebuah arsitektur metode komunikasi yang bisa bikin sistem melakukan pertukaran dan komunikasi data.

Jadi dengan RESTful API, konten dalam aplikasimu akan terkoneksi sehingga bisa bantu bikin kontenmu yang kudet jadi kelihatan up-to-date!



FYI, berikut pengertian RESTful API secara lebih lengkap.

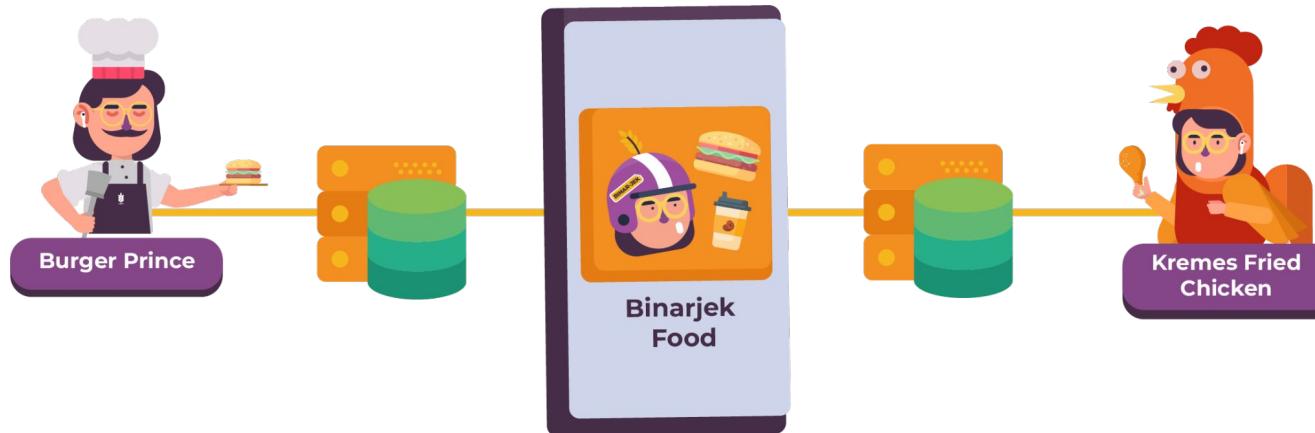
RESTful API merupakan **implementasi dari API** (Application Programming Interface). REST (REpresentational State Transfer) adalah suatu arsitektur **metode komunikasi** yang menggunakan protokol HTTP untuk **pertukaran data dan metode ini sering diterapkan dalam pengembangan aplikasi**.





Lalu tujuannya buat apa?

Tujuannya adalah untuk menjadikan sistem dengan performa yang baik, cepat, dan mudah untuk dikembangkan terutama dalam pertukaran dan komunikasi data.





Begini contoh cara kerja RESTful API, guys..

Putri mau cari informasi tentang film Spiderman lewat aplikasimu. Nah kalau aplikasimu udah terintegrasi dengan database server, maka pencarian film Spiderman akan diteruskan ke database server. Selanjutnya database di server akan memberikan data berupa film 'Spiderman' yang dicari oleh Putri.

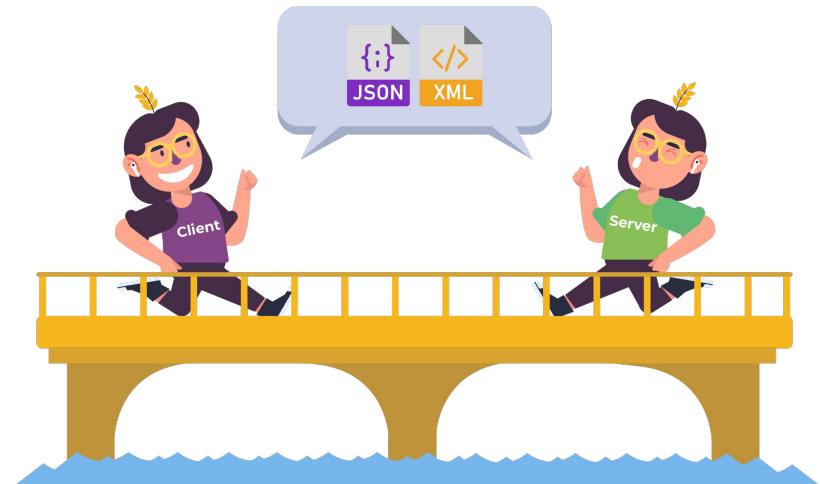
Inilah yang akan ditampilkan sebagai konten di aplikasi kamu.





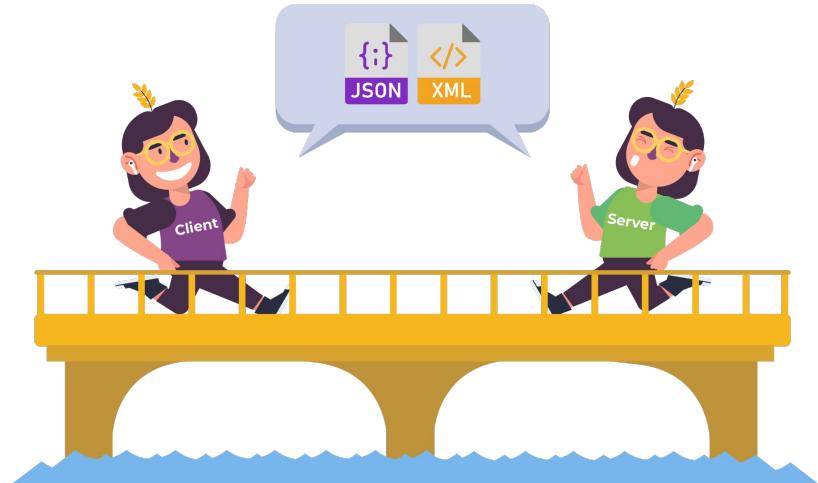
Kalau dipikir-pikir, fungsi RESTful API ini kayak kurir yaa..

Yak! RESTful API berperan sebagai **jembatan komunikasi data antara client dan server**. Response dalam bentuk data inilah yang dapat digunakan untuk berbagai macam platform dari aplikasi yang berbeda bahasa pemrograman.



Kalo kurir bawa paket, kalau RESTful API bawa JSON atau XML

Respons dari API biasanya berformat JSON atau XML, tapi yang paling umum dan banyak digunakan adalah JSON. Karena itulah, pada topik ini kita hanya akan membahas tentang JSON.





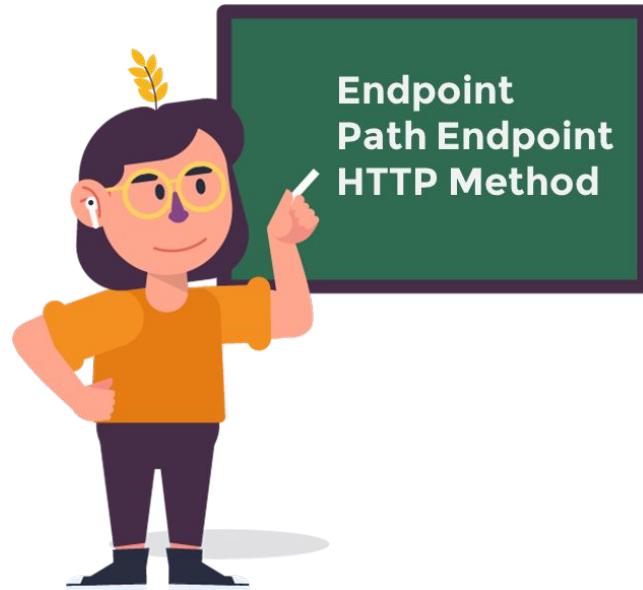
Untuk bisa mengakses sebuah API, kamu dapat menggunakan protokol HTTP atau HTTPS.

RESTful API memiliki beberapa komponen penting yang harus kamu pahami.

1. Endpoint

2. Path Endpoint

3. HTTP Method





ENDPOINT

Endpoint adalah sebuah alamat **url yang ditujukan untuk mengakses data** yang kamu inginkan.

<https://data.bmkg.go.id/DataMKG/MEWS/DigitalForecast/DigitalForecast-JawaTengah.xml>

ENDPOINT



Endpoint ini pasti sebenarnya sudah nggak asing buatmu. Ini contohnya!

Saat kamu mau mengakses data informasi gempa terbaru dari situs BMKG, BMKG sudah menyediakan sebuah RESTful API agar semua orang bisa mengakses data-data itu.

BMKG menyediakan sebuah **endpoint atau yang biasa kita kenal dengan alamat url** yaitu <https://data.bmkg.go.id>.

Alamat inilah yang akan kamu pakai untuk mendapatkan semua data yang bersumber dari BMKG.





PATH

PATH adalah sebuah alamat spesifik yang ditulis setelah endpoint.

Path bisa disebut sebuah pilihan kategori yang ingin kamu akses.

`https://data.bmkg.go.id/DataMKG/TEWS/gempaterkini.json`

Ini PATH untuk akses data List Gempa
di bawah 5.0 Skala Litter

`https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json`

Ini PATH untuk akses data gempa terbaru



Path seperti detil yang langsung mengarahkanmu pada tujuan

Misalnya saat kamu mau mengakses data informasi 15 gempa yang berkekuatan di bawah 5.0 skala richter, maka path yang dipakai adalah /DataMKG/TEWS/gempaterkini.json.

Saat mau mengakses data tersebut, kamu tinggal **menggabungkan ENDPOINT dan PATH-nya** jadi <https://data.bmkg.go.id/DataMKG/TEWS/gempaterkini.json>.

ENDPOINT

<https://data.bmkg.go.id>



PATH

`/DataMKG/TEWS/gempaterkini.json`



ENDPOINT

`https://data.bmkg.go.id`



PATH

`/DataMKG/TEWS/gempaterkini.json`

Contoh lainnya, misalnya kamu mau mengakses data gempa terbaru, maka path yang digunakan adalah `/DataMKG/TEWS/autogempa.json`.

Jadi, kamu bisa mengakses data informasi gempa terbaru dengan:

`https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json`.



HTTP METHOD

Adalah setiap request atau akses yang kamu lakukan terdapat sebuah verbs atau method, yang dipakai untuk menjelaskan maksud request yang diberikan.





Setiap method melambangkan jenis operasi/action yang berbeda-beda, contohnya kalau kamu ingin membaca data, kamu pakai method GET, tapi kalau kamu ingin menambahkan data, kamu pakai method POST.

Dengan kata lain **HTTP Method ini adalah sebagai penanda** apakah request HTTP ini tujuannya untuk ambil data, buat data baru, update data , atau hapus data.

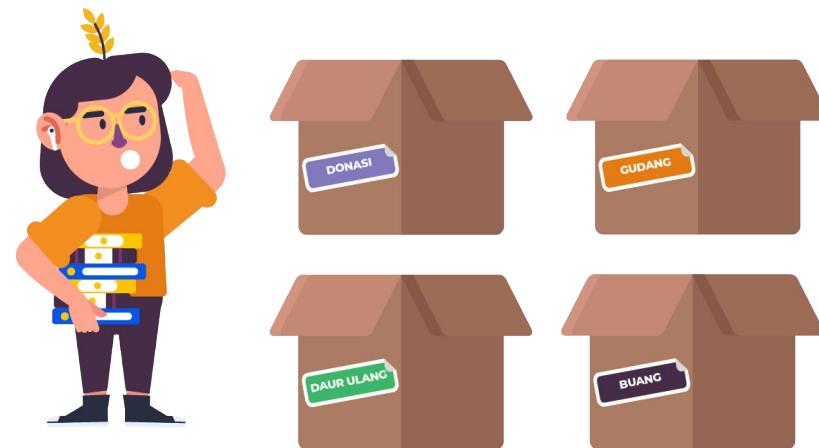


Kita bisa analogikan HTTP Method kayak kotak-kotak kardus~

Bayangin kamu lagi pindah rumah, pasti kamu akan butuh kotak-kotak khusus untuk memilah-milah barangmu dong?

Ada kotak untuk barang yang akan dibuang, ada kotak untuk barang yang akan kamu kirim ke panti asuhan dan ada kotak khusus untuk barang-barang kesayanganmu.

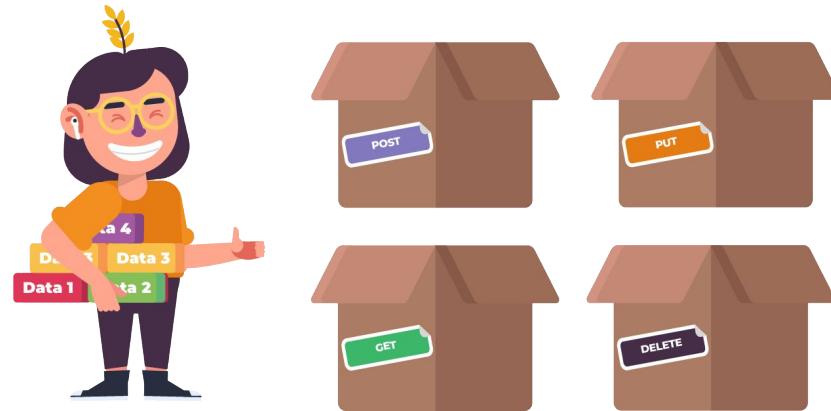
Tiap kotak punya tujuan mau diapain.





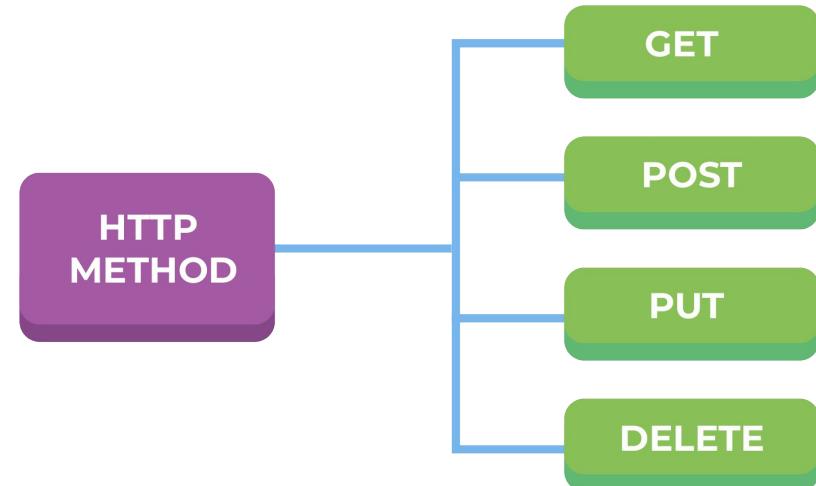
Nah, kotak-kotak itu sama kayak method. Gimana tuh?

Ada method yang digunakan untuk ambil data, ada method yang digunakan untuk buat data baru, update data , ataupun untuk hapus data.



Secara umum, ada **4 method yang biasa digunakan**, yaitu:

1. **Get**
2. **Post**
3. **Put**
4. **Delete**





Sekarang, mari kita bahas satu per satu
yuk supaya kamu paham fungsinya!





HTTP METHOD (GET)

Metode ini digunakan untuk **membaca atau mendapatkan data dari sumber.**

Seperti halnya pada kasus sebelumnya, kamu ingin mengakses/membaca data dari server BMKG, karena kebutuhanmu adalah membaca data, maka kamu harus menggunakan metode GET.

```
1 // 20220227094507
2 // https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json
3
4 {
5   "Infogempa": {
6     "gempa": {
7       "Tanggal": "27 Feb 2022",
8       "Jam": "04:55:49 WIB",
9       "DateTime": "2022-02-26T21:55:49+00:00",
10      "Coordinates": "-2.55,126.15",
11      "Lintang": "2.55 LS",
12      "Bujur": "126.15 BT",
13      "Magnitude": "4.7",
14      "Kedalaman": "10 km",
15      "Wilayah": "Pusat gempa berada di laut 58 km selatan Sanana",
16      "Potensi": "Gempa ini dirasakan untuk diteruskan pada masyarakat",
17      "Dirasakan": "III Sanana",
18      "Shakemap": "20220227045549.mmi.jpg"
19    }
20  }
21 }
```



Kalau kamu membuka RESTful API alamat website <https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json> pada browser seperti Google Chrome atau Mozilla, maka secara default metode yang kamu pakai ketika mengakses data tersebut adalah metode GET.

```
1 // 20220227094507
2 // https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json
3
4 {
5   "Infogempa": {
6     "gempa": {
7       "Tanggal": "27 Feb 2022",
8       "Jam": "04:55:49 WIB",
9       "DateTime": "2022-02-26T21:55:49+00:00",
10      "Coordinates": "-2.55,126.15",
11      "Lintang": "2.55 LS",
12      "Bujur": "126.15 BT",
13      "Magnitude": "4.7",
14      "Kedalaman": "10 km",
15      "Wilayah": "Pusat gempa berada di laut 58 km selatan Sanana",
16      "Potensi": "Gempa ini dirasakan untuk diteruskan pada masyarakat",
17      "Dirasakan": "III Sanana",
18      "Shakemap": "20220227045549.mmi.jpg"
19    }
20  }
21 }
```



HTTP METHOD (POST)

Metode ini dipakai untuk **membuat/menambahkan sebuah data baru pada server.**

Nah, nggak kayak method GET yang bisa kamu lakukan langsung lewat browser, operasi yang menggunakan **metode POST, PUT, DELETE ini nggak bisa kamu lakukan melalui browser**, melainkan harus melalui aplikasi yang berfungsi sebagai REST CLIENT.

Contoh:

Aplikasi yang bertugas sebagai REST CLIENT adalah aplikasi POSTMAN yang akan kita bahas di subtopik selanjutnya.



```
// 20220227094507
// https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

  "Infogempa": {
    "gempa": {
      "Tanggal": "27 Feb 2022",
      "Jam": "04:55:49 WIB",
      "DateTime": "2022-02-26T21:55:49+00:00",
      "Coordinates": "-2.55,126.15",
      "Lintang": "2.55 LS",
      "Bujur": "126.15 BT",
      "Magnitude": "4.7",
      "Kedalaman": "10 km",
      "Wilayah": "Pusat gempa berada di laut 58 km selatan Sanana",
      "Potensi": "Gempa ini dirasakan untuk diteruskan pada masyarakat",
      "Dirasakan": "III Sanana",
      "Shakemap": "20220227045549.mmi.jpg"
    }
  }
}
```



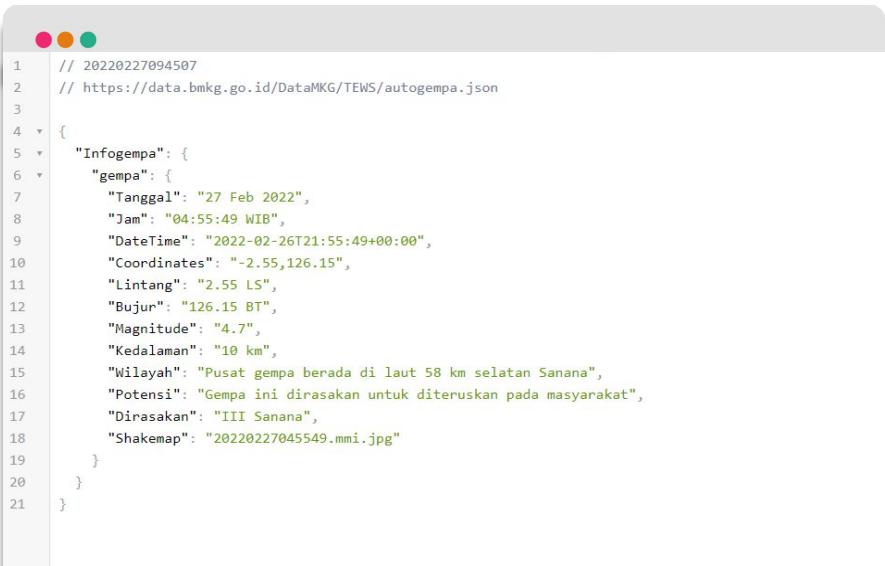
HTTP METHOD (PUT)

Metode ini digunakan untuk **mengupdate sebuah data yang telah terdapat pada server.**

Sebagai contoh, kamu ingin melakukan update nama akun Facebook kamu.

Nah di belakang layar, aplikasi Facebook akan melakukan sebuah operasi PUT untuk mengupdate data nama akun facebook kamu ke server facebook.

Sama kayak POST, apabila kamu ingin mencoba operasi PUT ini kamu harus menggunakan aplikasi REST Client ya.



```
// 20220227094507
// https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json

1 {
2   "Infogempa": {
3     "gempa": {
4       "Tanggal": "27 Feb 2022",
5       "Jam": "04:55:49 WIB",
6       "DateTime": "2022-02-26T21:55:49+00:00",
7       "Coordinates": "-2.55,126.15",
8       "Lintang": "2.55 LS",
9       "Bujur": "126.15 BT",
10      "Magnitude": "4.7",
11      "Kedalaman": "10 km",
12      "Wilayah": "Pusat gempa berada di laut 58 km selatan Sanana",
13      "Potensi": "Gempa ini dirasakan untuk diteruskan pada masyarakat",
14      "Dirasakan": "III Sanana",
15      "Shakemap": "20220227045549.mmi.jpg"
16    }
17  }
18}
19}
20}
21}
```



HTTP METHOD (DELETE)

Metode ini digunakan untuk **menghapus sebuah data yang telah terdapat pada server.**

Misalnya kamu mau menghapus status Facebook yang kamu buat beberapa minggu yang lalu. Nah di belakang layar, aplikasi Facebook akan melakukan operasi DELETE untuk menghapus data status kamu tersebut di server.

Sama kayak POST dan PUT, kalau kamu mau mengoperasikan DELETE, kamu harus menggunakan aplikasi REST Client.

```
// 2022022709450/
// https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json
{
  "Infogempa": [
    {
      "gempa": {
        "Tanggal": "27 Feb 2022",
        "Jam": "04:55:49 WIB",
        "DateTime": "2022-02-26T21:55:49+00:00",
        "Coordinates": "-2.55,126.15",
        "Lintang": "2.55 LS",
        "Bujur": "126.15 BT",
        "Magnitude": "4.7",
        "Kedalaman": "10 km",
        "Wilayah": "Pusat gempa berada di laut 58 km selatan Sanana",
        "Potensi": "Gempa ini dirasakan untuk diteruskan pada masyarakat",
        "Dirasakan": "III Sanana",
        "Shakemap": "20220227045549.mmi.jpg"
      }
    }
  ]
}
```



Selanjutnya kamu akan belajar tentang
JSON. Kenapa?

Karena **JSON** merupakan respons yang
umum digunakan dalam API.





Wahai tukang jajan yang lagi belajar React Native, camkan petuah ini..

Apapun makanannya, minumnya Teh Botol Sosro. Apapun bahasa pemrograman yang kamu pakai, wajib hukumnya memahami JSON. Kenapa?

Karena JSON banyak digunakan dalam pembuatan aplikasi dan digunakan di hampir semua bahasa pemrograman.

Tapi kenapa sih harus JSON?



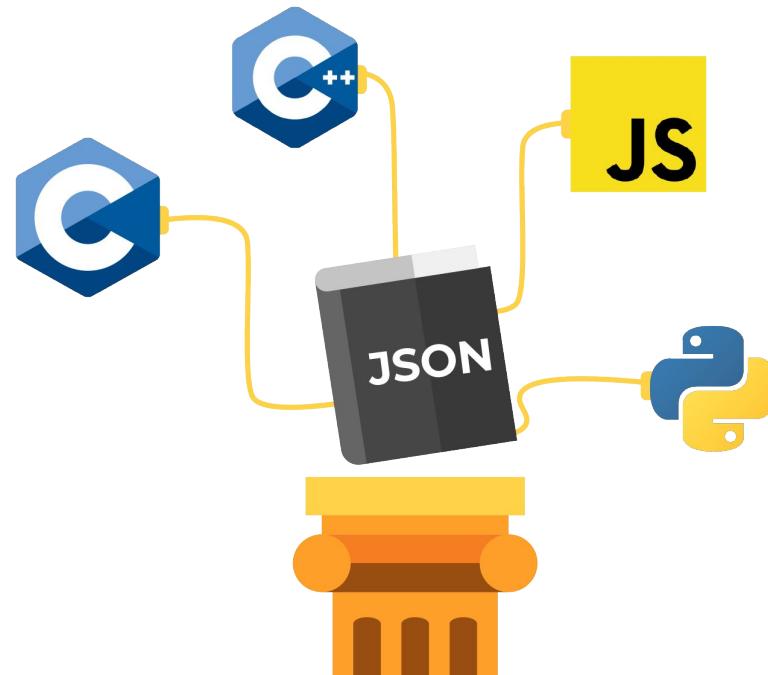


Begini guys..

JSON (JavaScript Object Notation) adalah sebuah **format data** yang digunakan untuk **pertukaran dan penyimpanan data**.

JSON adalah bagian (subset) dari Javascript yang bisa dibaca oleh berbagai macam bahasa pemrograman seperti C, C++, C#, Java, Javascript, Perl, Python, dan banyak lagi.

Hal inilah yang bikin **JSON jadi bahasa yang ideal untuk pertukaran data antar aplikasi**.





JSON bahkan mendominasi pendahulunya si XML (eXtensible Markup Language). Kalau dibandingkan dengan XML, JSON lebih sederhana dan mudah dibaca. Karena itulah, lebih banyak developer yang pakai JSON daripada XML.

Lihat saja nih tampilan codingnya, mana yang lebih simpel menurutmu?

XML

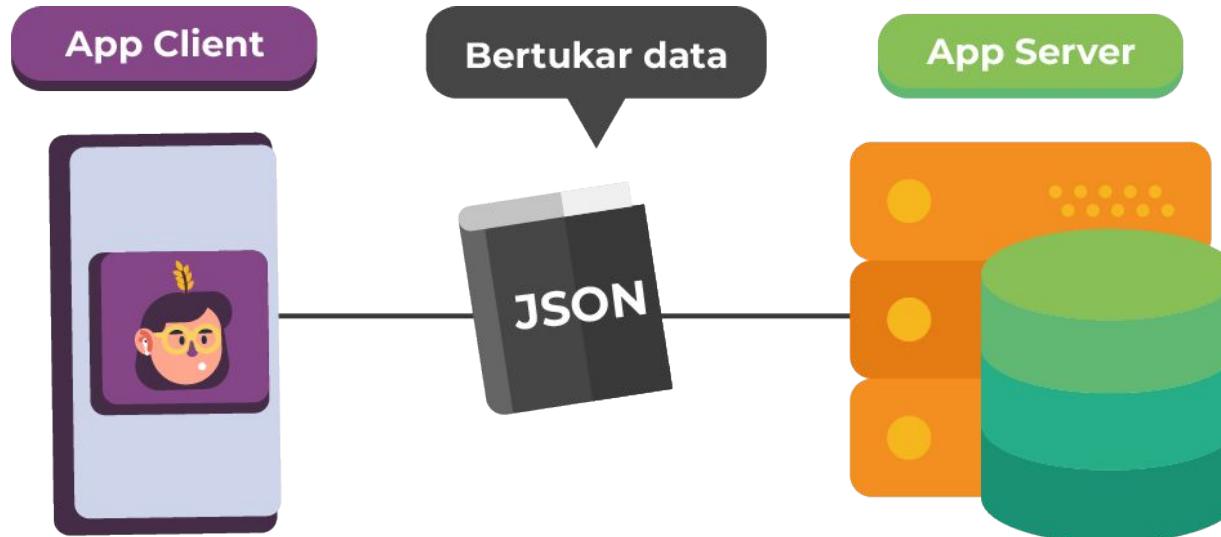
```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <biodata>
3   <name>Josua</name>
4   <jenisKelamin>Laki-laki</jenisKelamin>
5   <age>24</age>
6   <sosialMedia>
7     <twitter>@josssh</twitter>
8     <facebook>fb.me/joshhhha</facebook>
9   </sosialMedia>
10 </biodata>
```

JSON

```
1 {
2   "biodata": {
3     "name": "Josua",
4     "jenisKelamin": "Laki-laki",
5     "age": 24,
6     "sosialMedia": {
7       "twitter": "@josssh",
8       "facebook": "fb.me/joshhhha"
9     }
10   }
11 }
```

Begini nih penerapan JSON dalam Pemrograman

JSON biasanya digunakan sebagai format standar untuk bertukar data antar aplikasi.





Tapi sebenarnya masih ada fungsi lain dari JSON kok. Berikut ini enam penerapan tools atau aplikasi yang pakai JSON:

- JSON sebagai **format untuk bertukar data client dan server** atau antar aplikasi. Contoh: RESTFUL API
- JSON sebagai **tempat menyimpan data**, contoh: Database MongoDB
- JSON digunakan untuk **menyimpan konfigurasi project**, contoh: file composer.json pada project PHP dan package.json pada React;





Lanjutannya~

- JSON digunakan untuk **menyimpan konfigurasi dan penyimpanan data pada Hugo**;
- JSON digunakan untuk **menyimpan konfigurasi pada project nodejs**;
- JSON digunakan untuk **menyimpan data manifest**; dan masih banyak lagi.



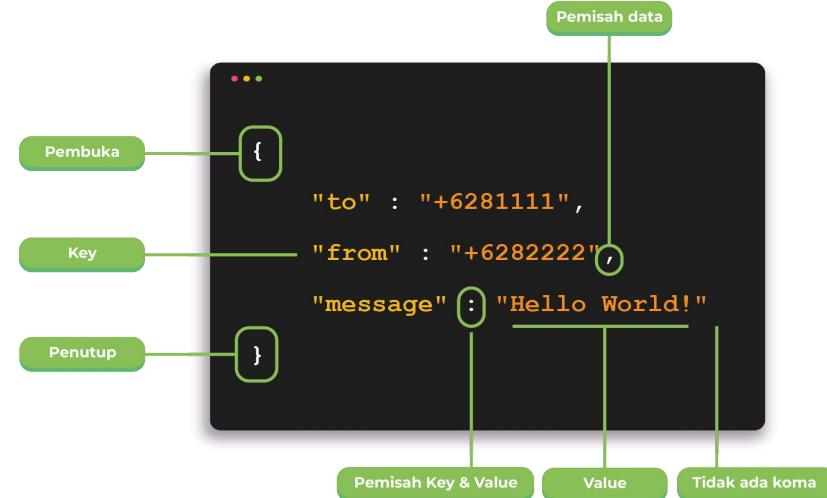


Struktur Dasar JSON

Coba perhatikan struktur JSON pada gambar di samping. Ini adalah struktur JSON yang paling sederhana.

JSON selalu dimulai dengan tanda kurung kurawal { dan ditutup dengan kurung }.

Di dalam kurung kurawal, berisi data yang formatnya key dan value. Kalau ada lebih dari satu data, maka dipisah dengan tanda koma dan di data terakhir nggak dikasih koma.





Oh iya, terus untuk value-nya..

Kamu bisa memberikan tipe data apa pun. Bahkan kamu juga bisa kamu isi dengan array dan objek, hingga tipe data null, number, dan Boolean.

```
1 {
2   "value1": null,
3   "value2": null,
4   "text1": null,
5   "text2": "hello",
6   "intValue": 0,
7   "myList": [],
8   "myEmptyList": null,
9   "boolean1": null,
10  "littleboolean": false
11 }
```



Kayak yang juga sudah sempat disebut di atas, kamu akan butuh Rest Client **POSTMAN** untuk melakukan operasi yang pakai metode **POST**, **PUT**, dan **DELETE**.

Karena ini penting, jadi ayolah kita pelajari **REST Client POSTMAN** juga.





Apa itu POSTMAN?

Iyaak betul.. postman adalah tukang pos. Eeeh tapi bukan postman itu yang akan kita bahas di sini hehe.

POSTMAN adalah sebuah **aplikasi yang berfungsi sebagai REST Client untuk uji coba REST API**.

POSTMAN biasa digunakan oleh developer pembuat API sebagai tools untuk menguji API yang telah mereka buat dan merupakan tool untuk melakukan proses development API.



POSTMAN

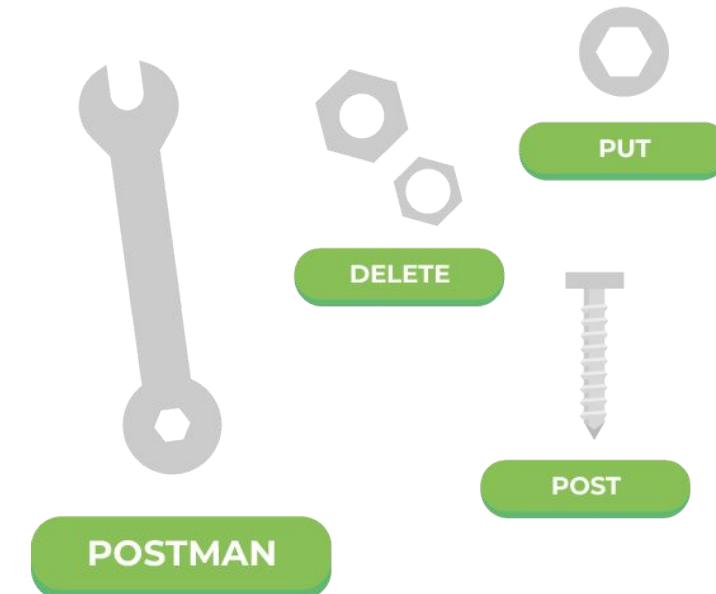


Eh tapi kenapa kamu perlu pakai POSTMAN ya?

Nah, seperti yang dijelaskan sebelumnya. kalau kamu mau mengakses RESTful API dengan metode GET, kamu bisa melakukannya langsung melalui browser saja.

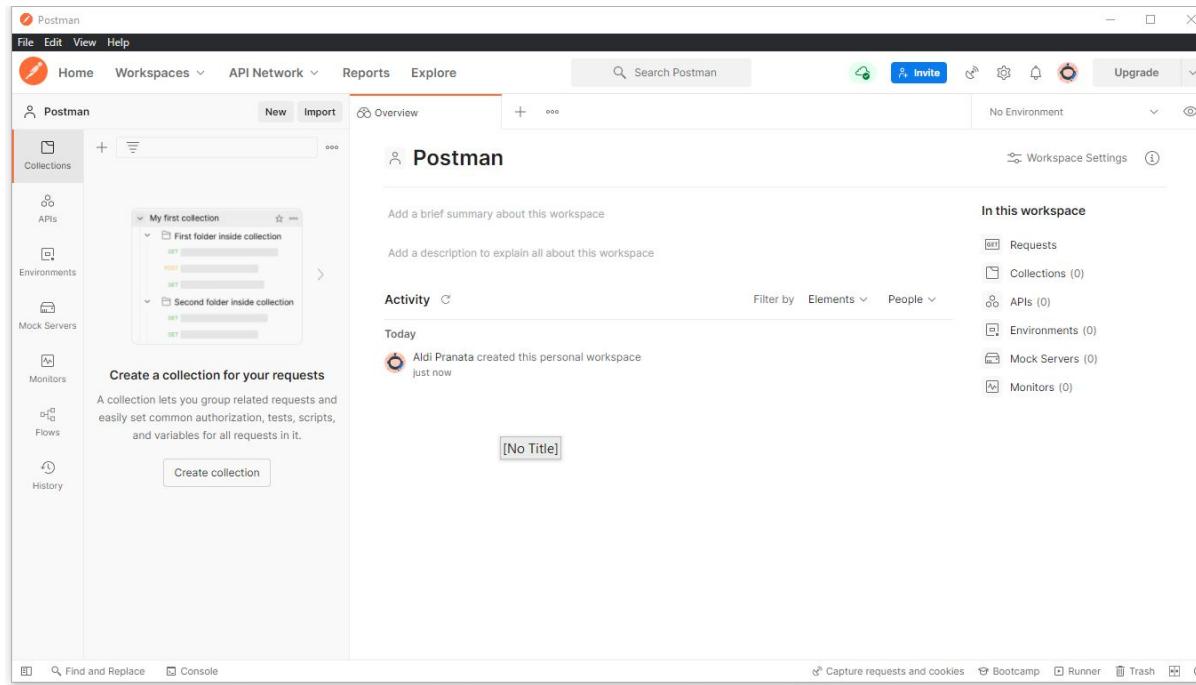
Tapi.. kalau kamu mau melakukan operasi dengan metode lain seperti POST, PUT, dan DELETE, kamu harus menggunakan sebuah tool REST Client. Salah satu tools pada kategori REST Client ini adalah **POSTMAN**.

Dengan POSTMAN kamu dapat membuat, mengupdate, menghapus data pada sebuah server melalui RESTful API.





Biar kamu kebayang, begini tampilan aplikasi POSTMAN



The screenshot shows the Postman application window. The left sidebar has sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a workspace titled "Postman". It includes fields for adding a brief summary and a description, an "Activity" feed showing a recent creation by "Aldi Pranata", and a section for "In this workspace" which lists Requests, Collections (0), APIs (0), Environments (0), Mock Servers (0), and Monitors (0). A "Create a collection for your requests" section is also present.



Terus, bagaimana caranya nih supaya bisa mengunduh POSTMAN?

Gampang banget! Untuk mengunduh aplikasi POSTMAN ini, kamu bisa langsung download di situs web resmi POSTMAN [di sini](#) ya!

Cara instalasinya juga gampang banget, guys. Setelah filenya ter-download, kamu tinggal klik tombol next, next, next saja deh.



Download Postman

Download the app to quickly get started using the Postman API Platform. Or, if you prefer a browser experience, you can try the new web version of Postman.

The Postman app

The ever-improving Postman app (a new release every week) gives you a full-featured Postman experience.

[Windows 64-bit](#)

By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

Version 9.13.0 · [Release Notes](#) · [Product Roadmap](#)

Not your OS? Download for Mac (Intel Chip / Apple Chip) or Linux (x64).

Twitter API v2 / Tweet Lookup Single Tweet

GET > https://api.twitter.com/2/tweets/:id

Params Authorization Headers Body Pre-request Scripts Tests Settings

Query param

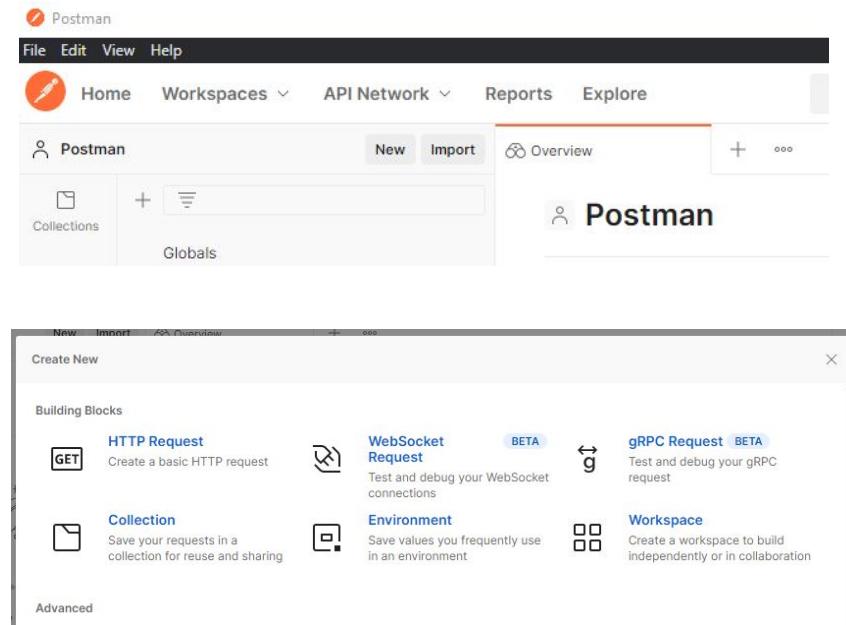
Key	Value	Description
tweet.fields	attachments,author_id,context_annotations,entities,in_reply_to_user_id,lang,public_metrics,possibly_sensitive,referenced_tweets,source,text	Comma-separated list of fields to expand.
expansions	Comma-separated list of fields to expand.	
media.fields	duration_ms,height,media_key,non_pu	Comma-separated list of fields for the media object.
post.fields	Comma-separated list of fields for the post object.	
place.fields	Comma-separated list of fields for the place object.	



Oke, pertanyaan selanjutnya, bagaimana ya cara menggunakan POSTMAN?

Yaudah, yuk langsung saja kita sambil belajar eksekusi.

Kita akan coba buat sebuah request menggunakan method GET untuk menampilkan data Gempa BMKG yang sudah kita bahas sebelumnya.



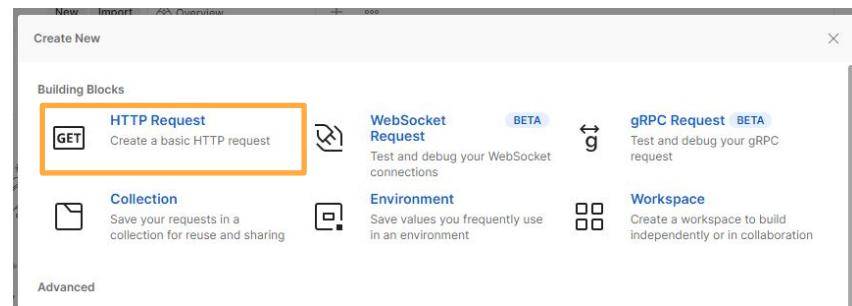
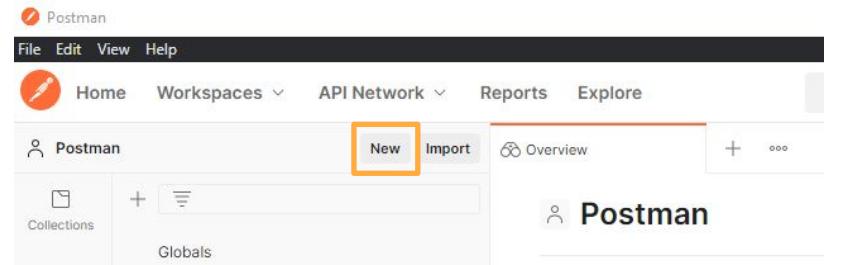
The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', and 'Help' options. Below the navigation bar, the main menu includes 'Home', 'Workspaces', 'API Network', 'Reports', and 'Explore'. The 'Home' option is highlighted. On the left side, there are buttons for 'Collections' and 'Globals'. The right side features a user profile with the name 'Postman'. A central panel displays a 'Create New' dialog box with the following options:

- Building Blocks**
 - HTTP Request** (BETA): Create a basic HTTP request. It has a 'GET' icon and a brief description.
 - Collection**: Save your requests in a collection for reuse and sharing. It has a folder icon and a brief description.
 - WebSocket Request** (BETA): Test and debug your WebSocket connections. It has a socket icon and a brief description.
 - gRPC Request** (BETA): Test and debug your gRPC request. It has a gRPC icon and a brief description.
 - Environment**: Save values you frequently use in an environment. It has an environment icon and a brief description.
 - Workspace**: Create a workspace to build independently or in collaboration. It has a workspace icon and a brief description.
- Advanced**: A section labeled 'Advanced'.



Langsung aja! Langkah yang pertama

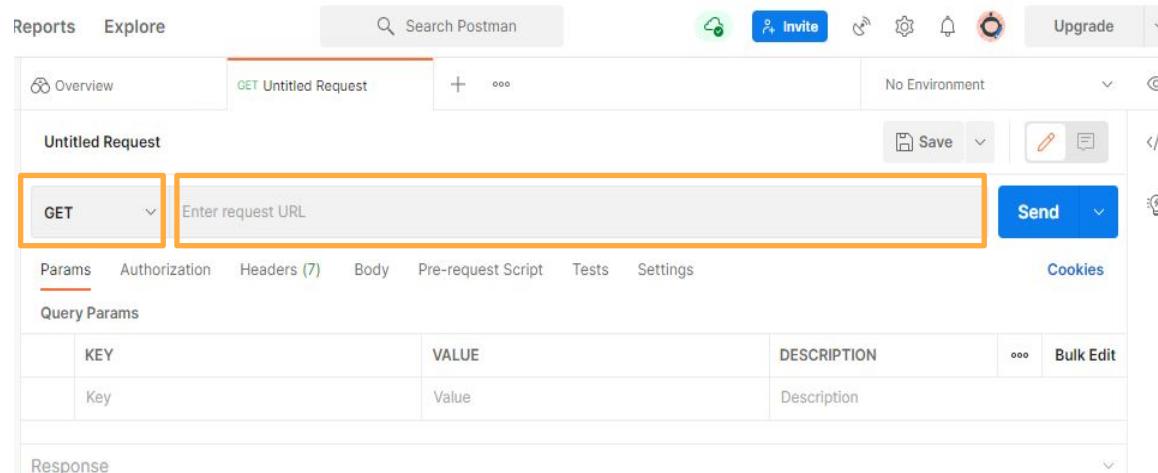
Buka aplikasi POSTMAN-mu, kemudian klik tombol **new**. Setelah itu akan muncul sebuah popup, kemudian klik **HTTP Request**.





Langkah kedua

Selanjutnya akan muncul sebuah tab baru pada POSTMAN, nah di tampilan inilah kamu akan melakukan request data ke server.



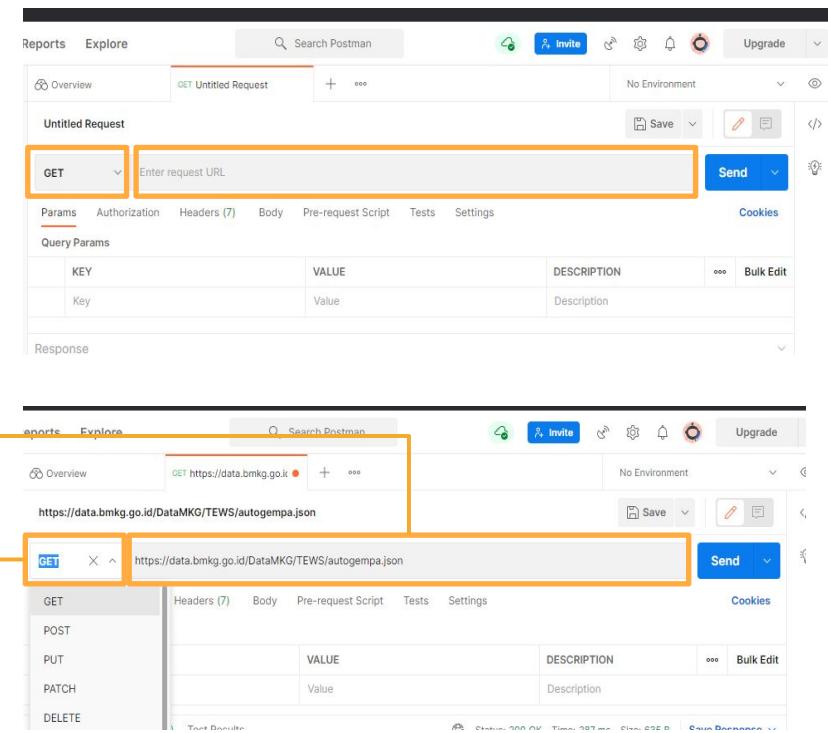
The screenshot shows the Postman application interface. At the top, there are tabs for 'Reports' and 'Explore', a search bar, and various icons for account management and upgrades. Below the header, a navigation bar includes 'Overview', 'Untitled Request' (which is currently active), a '+' button, and an 'Add' button. The main workspace is titled 'Untitled Request' and contains a 'Send' button. A dropdown menu for the method ('GET') is open, and an input field for the 'Enter request URL' is highlighted with an orange box. Below this, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. Under the 'Params' tab, there is a table for 'Query Params' with columns for 'KEY', 'VALUE', 'DESCRIPTION', and 'Bulk Edit'. The first row shows a 'Key' column with the value 'Description'. At the bottom of the interface, there is a section labeled 'Response'.



Lanjut ke langkah yang ketiga, guys!

Langkah selanjutnya yang harus kamu lakukan adalah mengisi URL dan memilih HTTP method pada POSTMAN. Begini caranya :

1. Pilih method GET, karena kamu ingin mendapatkan data dari BMKG
2. Isi ENDPOINT dan PATH pada bagian kotak 'Enter Request URL'. Masukan ENDPOINT dan PATH nya yaitu <https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json>



The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Reports', 'Explore', a search bar ('Search Postman'), and various account and settings icons. Below the bar, the main area is titled 'Untitled Request' with a 'GET' method selected. A large orange box highlights the 'Enter request URL' input field, which contains the URL 'https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json'. To the left of this field, another orange box highlights the 'GET' button. Below the URL input, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. Under the 'Headers (7)' tab, there's a table with one row: 'KEY' (Key) and 'VALUE' (Value). At the bottom of the interface, there are buttons for 'Send' and 'Cookies'.



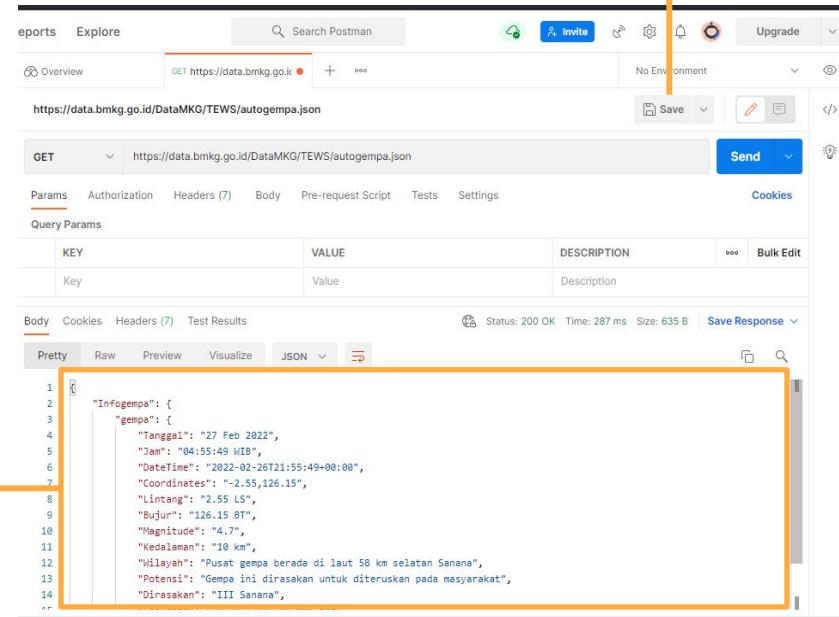
Langkah keempat, alias langkah terakhir!

Setelah itu, kamu klik tombol "Send", maka POSTMAN akan mengirimkan request untuk membaca info gempa terbaru ke server, setelah proses request selesai, server akan mengirimkan data yang telah kamu request, yaitu data Info gempa terbaru.

Nah, pada bagian bawah POSTMAN adalah data yang diterima dan ditampilkan dari server milik BMKG.

Response data dari
POSTMAN

Pilih method GET, karena kita ingin mendapatkan data dari BMKG



The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Reports', 'Explore', a search bar ('Search Postman'), and various buttons like 'Invite', 'Upgrade', and a profile icon. Below the navigation is a card for a collection named 'Overview' with a 'GET' request to 'https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json'. The request details show 'Params' selected, with a single parameter 'Key' listed. The 'Body' tab is active, displaying the JSON response received from the server. The response is as follows:

```
1  {
2   "Infogempa": [
3     "gempa": [
4       {
5         "Tanggal": "27 Feb 2022",
6         "Jam": "04:55:49 WIB",
7         "Datetime": "2022-02-26T21:55:49+00:00",
8         "Coordinates": "-2.55,126.15",
9         "Lintang": "2.55 LS",
10        "Bujur": "126.15 BT",
11        "Magnitude": "4.7",
12        "Kedalaman": "10 km",
13        "Wilayah": "Pusat gempa berada di laut 58 km selatan Sanana",
14        "Potensi": "Gempa ini dirasakan untuk diteruskan pada masyarakat",
15        "Dirasakan": "III Sanana"
16      }
17    ]
18  ]
```



Kalau sudah berhasil download POSTMAN,
sekarang kita lanjut ke materi selanjutnya
yaitu **HTTP Status Code**.

Apa fungsinya dalam POSTMAN? Cuss cari
tahu..

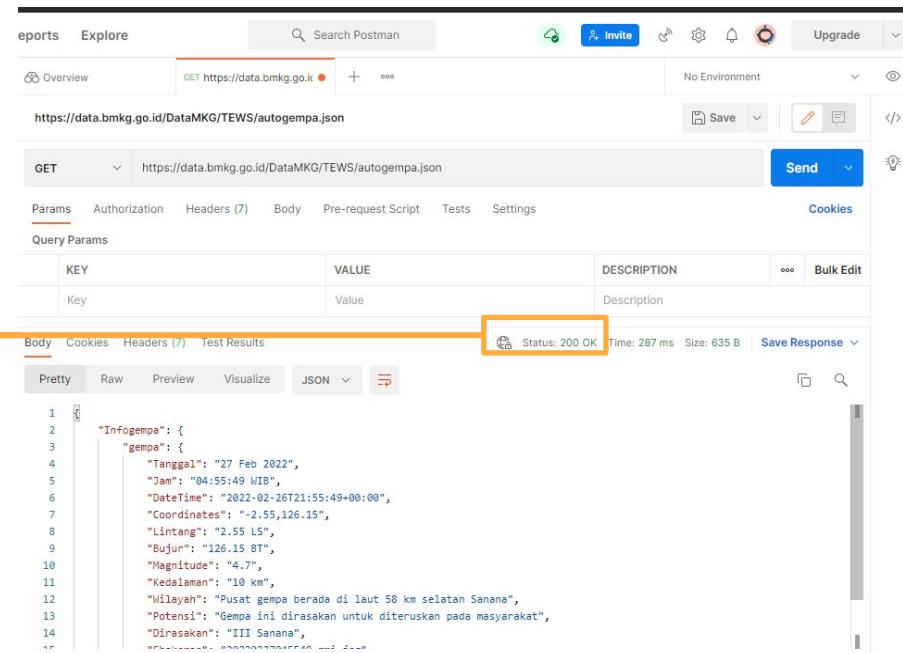




HTTP Status Code

Kalau kamu perhatikan, selain menampilkan Response data dari server, POSTMAN juga menampilkan kode response status lho.

Jadi apa sih kode response status ini?



The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Reports', 'Explore', a search bar ('Search Postman'), and various icons for account management and upgrades. Below the bar, a tab labeled 'Overview' is selected, showing the URL 'https://data.bmkg.go.id/TEWS/autogempa.json'. The main workspace shows a 'GET' request to the same URL. Below the request, tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings' are visible. A 'Cookies' tab is also present. Under 'Query Params', there's a table with columns 'KEY', 'VALUE', 'DESCRIPTION', and 'Bulk Edit'. The 'Body' tab is active, displaying the JSON response from the server. The response body is as follows:

```
1 "Infogempa": {  
2     "gempa": {  
3         "Tanggal": "27 Feb 2022",  
4         "Jam": "04:55:49 WIB",  
5         "DateTime": "2022-02-26T21:55:49+00:00",  
6         "Coordinates": "-2.55,126.15",  
7         "Lintang": "2.55 S",  
8         "Bujur": "126.15 E",  
9         "Magnitude": "4.7",  
10        "Kedalaman": "10 km",  
11        "Wileayah": "Pusat gempa berada di laut 58 km selatan Sanana",  
12        "Potensi": "Gempa ini dirasakan untuk diteruskan pada masyarakat",  
13        "Dirasakan": "III Sanana",  
14        "Rasakan": "Rasakan gempa di Sanana"}
```



Kode responses status ini adalah salah satu **indikator** yang menandakan apabila request kamu **berhasil**, atau dalam proses request ke server ada kendala **error**.

Singkatnya, Kode HTTP Status adalah response server atas request data yang kamu lakukan.

Kode response status ini berbentuk kode tiga digit dan biasanya kamu nggak akan sadar kalau proses pertukaran data berjalan dengan lancar.

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Reports', 'Explore', a search bar ('Search Postman'), and various icons for account management and upgrades. Below the bar, a tab labeled 'Overview' is selected, showing the URL 'https://data.bmkg.go.id/DataMKG/TEWS/autogempa.json'. The main workspace shows a 'GET' request to the same URL. Underneath, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is active, showing a table for 'Query Params' with columns for 'KEY', 'VALUE', 'DESCRIPTION', and 'Bulk Edit'. Below this, the 'Body' tab displays the raw JSON response. The status bar at the bottom of the interface indicates 'Status: 200 OK', along with other details like 'Time: 287 ms' and 'Size: 635 B'. The JSON response content is partially visible, showing details about an earthquake event.

```
1 "Infogempa": {  
2     "gempa": {  
3         "Tanggal": "27 Feb 2022",  
4         "Jam": "04:55:49 WIB",  
5         "DateTime": "2022-02-26T21:55:49+00:00",  
6         "Coordinates": "-2.55,126.15",  
7         "Lintang": "2.55 LS",  
8         "Bujur": "126.15 BT",  
9         "Magnitude": "4.7",  
10        "Kedalaman": "10 km",  
11        "Wileayah": "Pusat gempa berada di laut 58 km selatan Sanana",  
12        "Potensi": "Gempa ini dirasakan untuk diteruskan pada masyarakat",  
13        "Dirasakan": "III Sanana",  
14        "Rasakan": "Rasakan gempa di Sanana"}
```



Dalam **HTTP Status Code**, ada tiga macam kode yang umum dipakai.

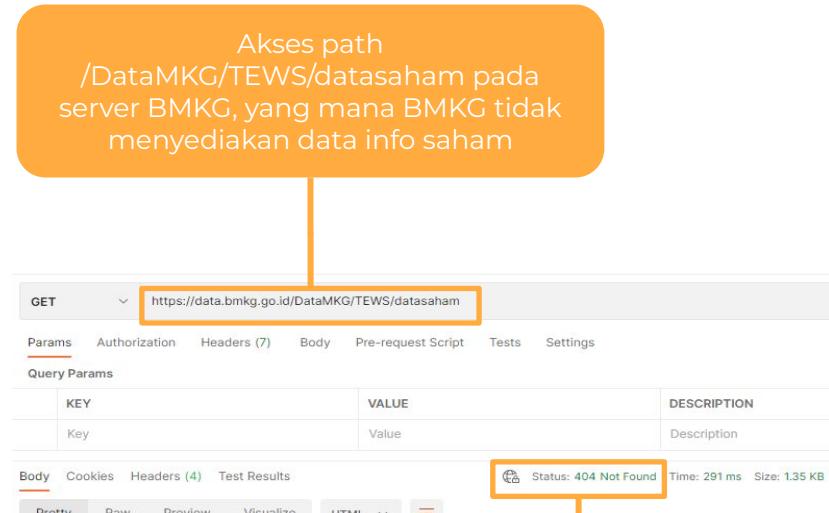
Simak baik-baik yaa, supaya kamu nggak salah mengartikan~





2xx – Kode Berhasil

Kalau http status code-mu diawali dengan angka 2, maka request kamu terbilang sukses, tanpa ada kendala error.



The screenshot shows the Postman interface with a failed API call. The URL in the header is `https://data.bmkg.go.id/DataMKG/TEWS/datasaham`. The status bar at the bottom indicates `Status: 404 Not Found`. A callout box points to this status message with the text: "Request akan gagal, dan server akan memberikan response error berawalan dengan angka 4, yang artinya kesalahan dari sisi kamu yang meminta info data yang tidak ada".

Akses path
`/DataMKG/TEWS/datasaham` pada
server BMKG, yang mana BMKG tidak
menyediakan data info saham

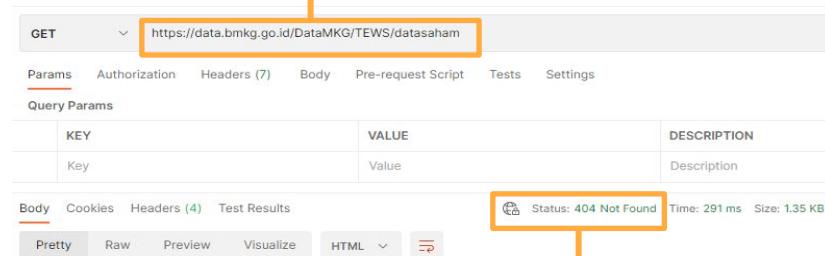


Akses path
/DataMKG/TEWS/datasaham pada
server BMKG, yang mana BMKG tidak
menyediakan data info saham

4xx – Kode Error karena kesalahan ketika request

Error kode yang berawalan angka 4 ini maksudnya adalah error kategori yang disebabkan oleh request yang kamu buat.

Misalnya kamu mau mengakses data informasi saham di server BMKG, nah saat kamu mengakses data tersebut, BMKG akan memberi respon Error 4xx, karena BMKG nggak pernah punya data infromasi saham.



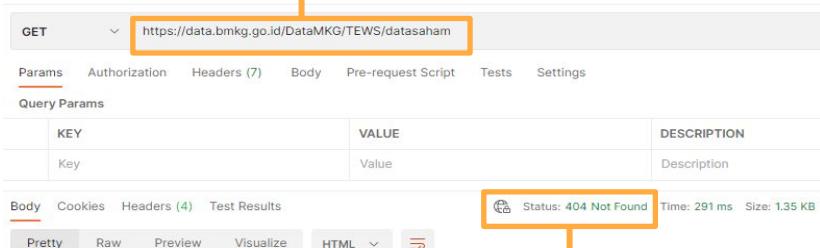
The screenshot shows the Postman interface with a failed GET request. The URL in the header is highlighted with an orange box: `https://data.bmkg.go.id/DataMKG/TEWS/datasaham`. In the bottom right corner of the response area, there is an orange box containing the text "Status: 404 Not Found Time: 291 ms Size: 1.35 KB".

Request akan gagal, dan server akan memberikan response error berawalan dengan angka 4, yang artinya kesalahan dari sisi kamu yang meminta info data yang tidak ada



5xx – Kode Error karena server lagi error

Error kode yang berawalan dengan angka 5 adalah error yang disebabkan ada kesalahan dari sisi server, seperti misalnya servernya lagi mati.



Akses path
/DataMKG/TEWS/datasaham pada
server BMKG, yang mana BMKG tidak
menyediakan data info saham

Request akan gagal, dan server akan memberikan response error berawalan dengan angka 4, yang artinya kesalahan dari sisi kamu yang meminta info data yang nggak ada



Oke, sekarang kita masuk ke materi yang paling penting nih, guys. Ups, apa tuh?

Cara menggunakan **POSTMAN**!





Cara bikin Request POST, PUT, DELETE dengan POSTMAN

Nah, sekarang kamu akan coba pakai server lain untuk melakuan proses Pembuatan Data (POST), mengupdate Data (PUT) dan menghapus data (DELETE).

Kamu akan menggunakan server yang punya Restful API untuk membuat to do list dan menggunakan ENDPOINT server code.aldipee.com untuk melakukan semua operasi method request (GET, POST, PUT, dan DELETE).



POSTMAN

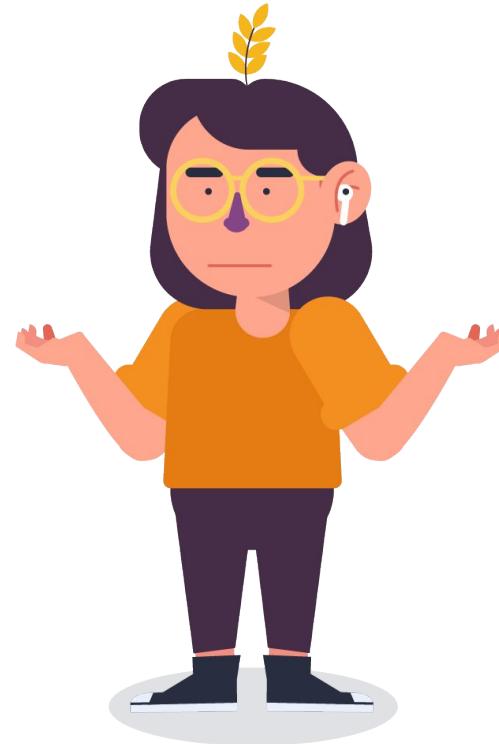


Eittss tapi..

Kamu kan sudah tahu nih, kalau bakal pake server/endpoint code.aldipee.com untuk bikin data, hapus data, update data, dan baca data.

Masalahnya kamu belum tahu Path apa yang akan kamu pakai untuk mengakses data-data pada server/endpoint code.aldipee.com?

Di mana ya kamu bisa dapatkan informasi Path yang tersedia di server code.aldipee.com tersebut?





Jawabannya adalah..

API Documentation/Dokumentasi API!

Jadi setiap RESTful API itu pasti ada yang namanya dokumen dokumentasi

Nah, di dalam dokumen itulah nanti kamu bisa mendapatkan banyak informasi yang kamu butuhkan, berhubungan dengan Path yang akan kamu pakai.





Namanya juga dokumentasi, ya pasti fungsinya untuk ngasih informasi

Jangan dianggurin.

Sama kayak kalau kamu mau bikin kue brownies, maka kamu harus lihat dokumentasi resepnya dulu untuk tahu bahan-bahan apa saja yang kamu butuhkan untuk membuat bolu brownies tersebut.

RESTful API juga begitu. Kamu **harus melihat dokumentasinya lebih dulu untuk mendapatkan PATH yang akan kamu gunakan** untuk mengakses data tertentu.





Nah, untuk server `code.aldipee.com`, kamu bisa lihat dokumentasinya di [link ini](#), guys.

The screenshot shows the API Playground interface with the following details:

- ENVIRONMENT:** Playground - Prod
- LAYOUT:** Double Column
- LANGUAGE:** CURL
- API PLAYGROUND - PUBLISHED:** Todo
- Introduction:** Create, Read, Update, and Delete all to-do items. You can create unlimited todo items without Authentication.
- Authentication:** Rate limit
- Todo:** A section containing two endpoints:
 - GET All todo list items**:
HTTP Request: `http://code.aldipee.com/api/v2/todos`
Description: Fetch all to-do items. Each time you perform a request, the server will return a max of 10 items.
 - GET Todo item detail by id**:
HTTP Request: `http://code.aldipee.com/api/v2/todos/:id`
Description: Get todo item details by ID.
- Example Request:** curl --location --request GET 'http://code.aldipee.com/api/v2/todos'
- Response Success Sample:** 200 OK
- Body:** Header (1/8)

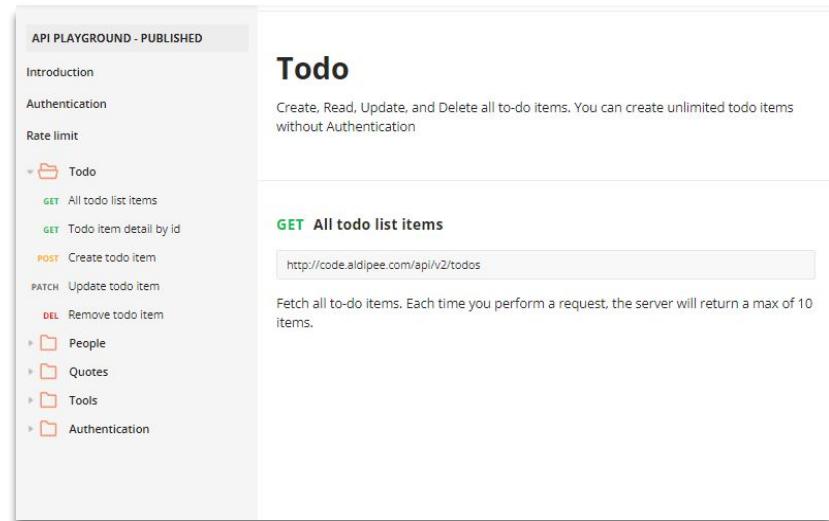
```
{ "success": true, "message": "Successfully retrieved items", "data": { "results": [ { "status": "PENDING", "title": "Buy flour", "created_at": "2022-02-26T0", "id": "6239883a7f72754880a1" } ] } }
```
- Example Response:** curl --location --request GET 'http://code.aldipee.com/api/v2/todos/6239883a7f72754880a1'
- Response Success Sample:** 200 OK



Perhatikan gambar di samping deh..

Karena kamu ingin membuat To-do app, berarti **kamu perlu buka tab di sebelah kiri pada document** dokumentasi, yang berhubungan dengan data to-do app ya, guys.

Nah, di folder '**Todo**' tersebut, ada beberapa operasi yang bisa kamu lakukan beserta method HTTP-nya.

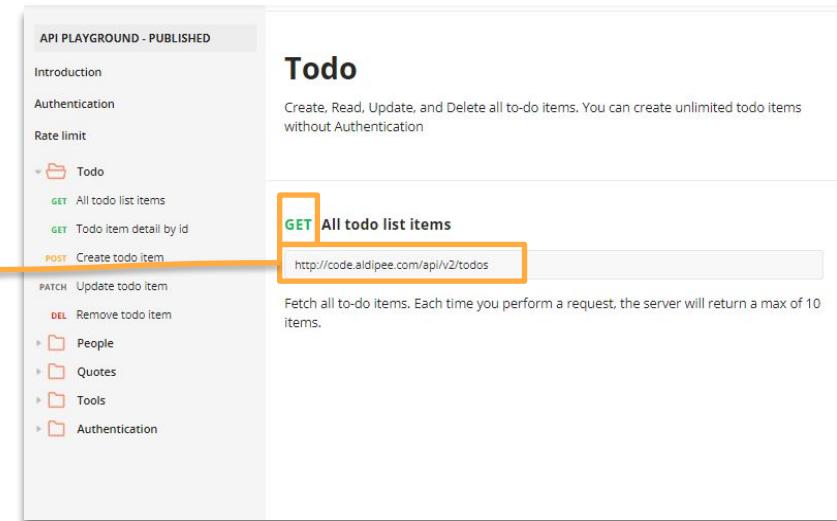


The screenshot shows the API Playground interface. On the left, there's a sidebar titled "API PLAYGROUND - PUBLISHED" with a tree icon. It lists several sections: Introduction, Authentication, Rate limit, Todo (which is expanded), People, Quotes, Tools, and Authentication. Under the Todo section, it shows methods: GET All todo list items, GET Todo item detail by id, POST Create todo item, PATCH Update todo item, and DEL Remove todo item. To the right, under the "Todo" heading, it says "Create, Read, Update, and Delete all to-do items. You can create unlimited todo items without Authentication". Below that is a "GET All todo list items" section with a URL input field containing "http://code.aldipee.com/api/v2/todos" and a description: "Fetch all to-do items. Each time you perform a request, the server will return a max of 10 items."



Apa yang bisa kamu lihat dari gambar ini, guys?

Dari dokumentasi ini kamu bisa tahu bahwa, kalau kamu mau mengakses semua data to-do list, maka kamu harus pakai path **/api/v2/todos** dan pake method **GET**



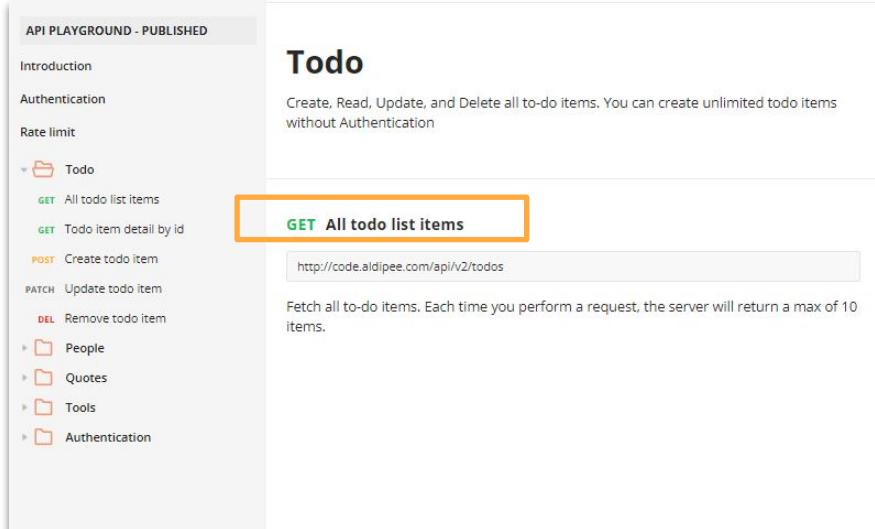
The screenshot shows the API Playground interface. On the left, there's a sidebar titled "API PLAYGROUND - PUBLISHED" with sections for Introduction, Authentication, Rate limit, and a "Todo" category containing endpoints for GET, POST, PATCH, and DEL methods. A red arrow points from the text above to the "GET All todo list items" entry. The main content area has a title "Todo" with a description: "Create, Read, Update, and Delete all to-do items. You can create unlimited todo items without Authentication". Below this is a detailed endpoint entry for "All todo list items": "GET All todo list items" with the URL "http://code.aldipee.com/api/v2/todos". A red box highlights this entire section. The description for this endpoint states: "Fetch all to-do items. Each time you perform a request, the server will return a max of 10 items."



Selanjutnya bagaimana?

Untuk mengakses semua data to-do list, pada dokumentasi dibilang kalau kamu harus pakai method **GET**.

Nah, sekarang kita coba akses yuk datanya melalui **POSTMAN**

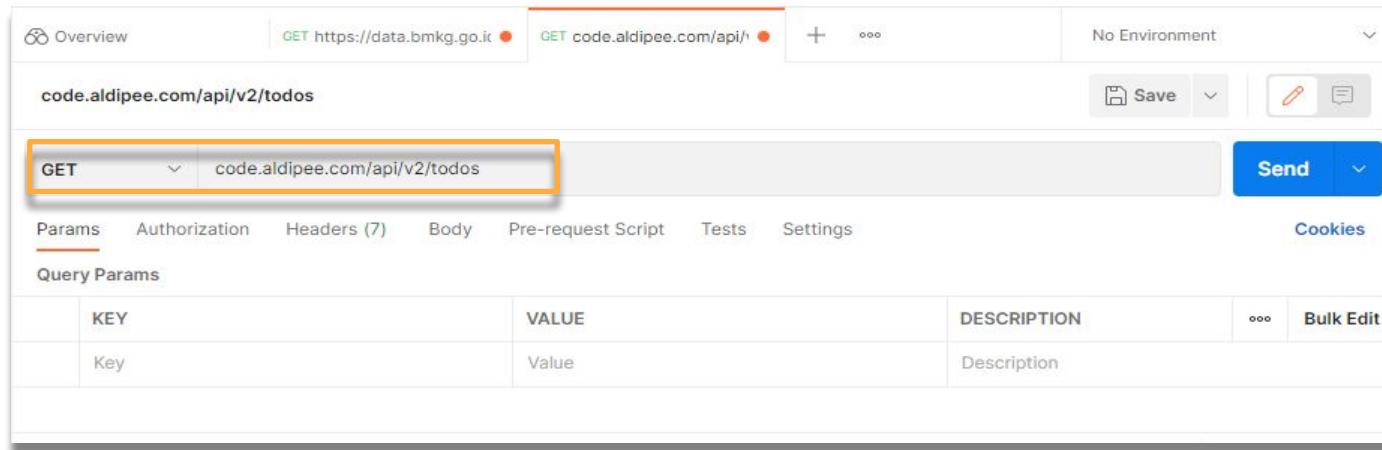


The screenshot shows the API Playground interface. On the left, there's a sidebar with links like 'API PLAYGROUND - PUBLISHED', 'Introduction', 'Authentication', and 'Rate limit'. Below that is a tree view with 'Todo' expanded, showing 'All todo list items' (highlighted with an orange border), 'Todo item detail by id', 'Create todo item', 'Update todo item', 'Remove todo item', and sub-folders for 'People', 'Quotes', 'Tools', and 'Authentication'. To the right, under the 'Todo' section, the title 'Todo' is displayed with the sub-instruction 'Create, Read, Update, and Delete all to-do items. You can create unlimited todo items without Authentication'. A button labeled 'GET All todo list items' is shown, along with the URL 'http://code.aldipee.com/api/v2/todos'. Below the URL, a note says 'Fetch all to-do items. Each time you perform a request, the server will return a max of 10 items.'



Langkah mengakses data melalui POSTMAN:

Masukan ENDPOINT-nya yaitu code.aldipee.com, kemudian masukan juga Path yang kamu dapatkan dari dokumentasi tadi, yaitu /api/v2/todos. Kemudian pilih method sesuai dengan yang ada dokumentasi tadi yaitu GET, lalu tekan tombol 'Send'

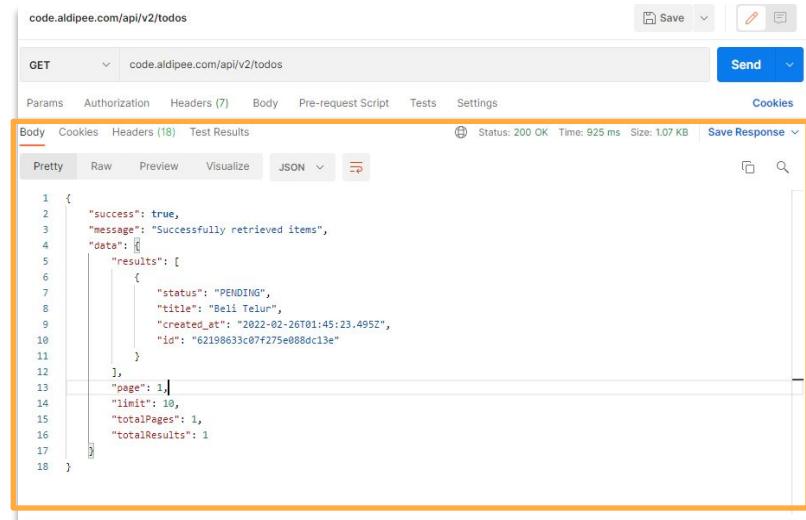


The screenshot shows the Postman application interface. At the top, there are tabs for Overview, GET https://data.bmkg.go.id (selected), and GET code.aldipee.com/api/. Below these are buttons for Save, Edit, and Delete. The main area shows a GET request to code.aldipee.com/api/v2/todos. The 'Params' tab is selected, showing a table for Query Params:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		



Nah, Request yang telah kamu buat barusan berhasil nih, guys. POSTMAN-nya Sudah munculin data JSON.



The screenshot shows the Postman interface with a successful API call to `code.aldipee.com/api/v2/todos`. The response is a JSON object:

```
1 {  
2   "success": true,  
3   "message": "Successfully retrieved items",  
4   "data": [  
5     "results": [  
6       {  
7         "status": "PENDING",  
8         "title": "Beli Telur",  
9         "created_at": "2022-02-26T01:45:23.495Z",  
10        "id": "62198633c07f275e088dc13e"  
11      }  
12    ],  
13    "page": 1,  
14    "limit": 10,  
15    "totalPages": 1,  
16    "totalResults": 1  
17  ]  
18 }
```



Jangan salah masukin Path Iho, guys..

Eh tapi kira-kira, apa yang terjadi ya kalau misalnya kamu salah masukin Path yang nggak sesuai di dokumentasi?

Misal di dokumentasi dijelaskan pake **/api/v2/todos** tapi kamu malah pake **/api/v2/todo** yang nggak ada di dokumentasi.

The screenshot shows the Postman application interface. A GET request is being made to the URL "code.aldipee.com/api/v2/todo". The response status is 404 Not Found, with a message indicating "Not found".

```
1  {
2     "code": 404,
3     "message": "Not found"
4 }
```



Kalau kamu salah masukin Path, maka ini nih yang akan terjadi..

Kamu nggak bakal berhasil mendapat data yang kamu inginkan. Sebaliknya, kamu akan mendapatkan respon status 404.

Seperti kita bahas sebelumnya setiap respon status yang awalnya 4, itu adalah error yang disebabkan oleh kesalahan kamu dalam membuat request, guys.

The screenshot shows the Postman application interface. A GET request is made to `code.aldipee.com/api/v2/todo`. The response status is 404 Not Found, with a time of 92 ms and a size of 888 B. The response body is a JSON object:

```
1  {
2   "code": 404,
3   "message": "Not found"
4 }
```



Nah sekarang kita akan bahas bagaimana cara membuat data pada sebuah server melalui POSTMAN.

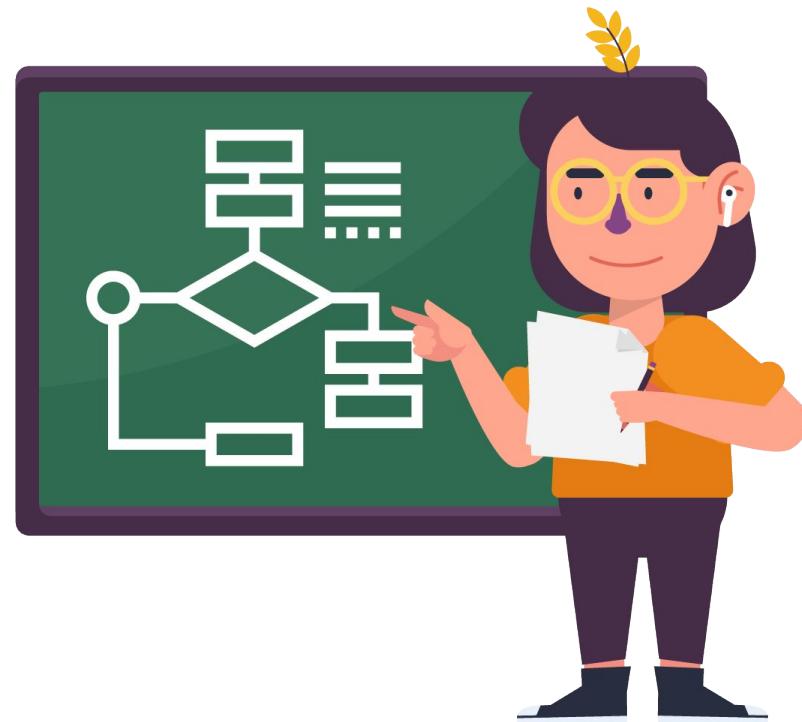
Kamu akan coba melakukan request dengan menggunakan method POST.





Hmm.. lalu darimana kita akan memulainya yaa?

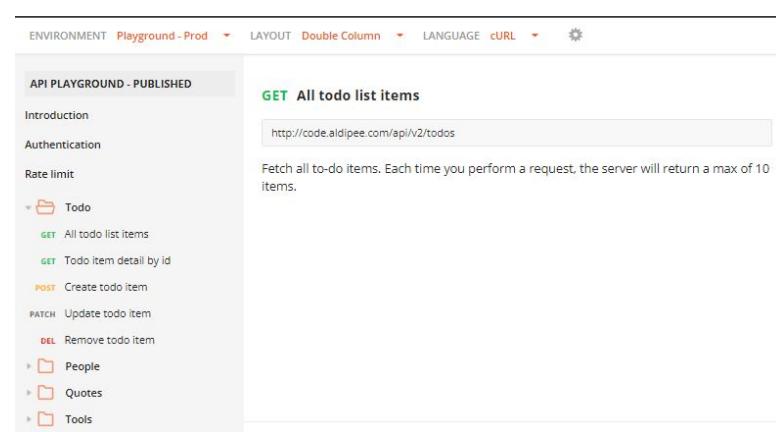
Sama seperti sebelumnya, kamu harus lihat terlebih dahulu pada document dokumentasi yang telah diberikan agar kamu dapat informasi berupa path dan method yang digunakan untuk membuat data to-do baru pada server.





Lihat gambar di samping dulu, lalu ikuti langkah-langkah berikut ya:

1. Pada bagian navigasi menu sebelah kiri terdapat nama operasi yang kita inginkan yaitu “Create todo item”. Klik menu tersebut.



The screenshot shows the Postman API playground interface. At the top, there are environment, layout, and language settings. Below that, the API playground header indicates it's published. On the left, a sidebar lists sections: Introduction, Authentication, Rate limit, Todo (with sub-options: GET All todo list items, GET Todo item detail by id, POST Create todo item, PATCH Update todo item, and DEL Remove todo item), People, Quotes, and Tools. The main content area shows a GET request for 'All todo list items' with the URL `http://code.aldipee.com/api/v2/todos`. A description below the URL states: 'Fetch all to-do items. Each time you perform a request, the server will return a max of 10 items.'



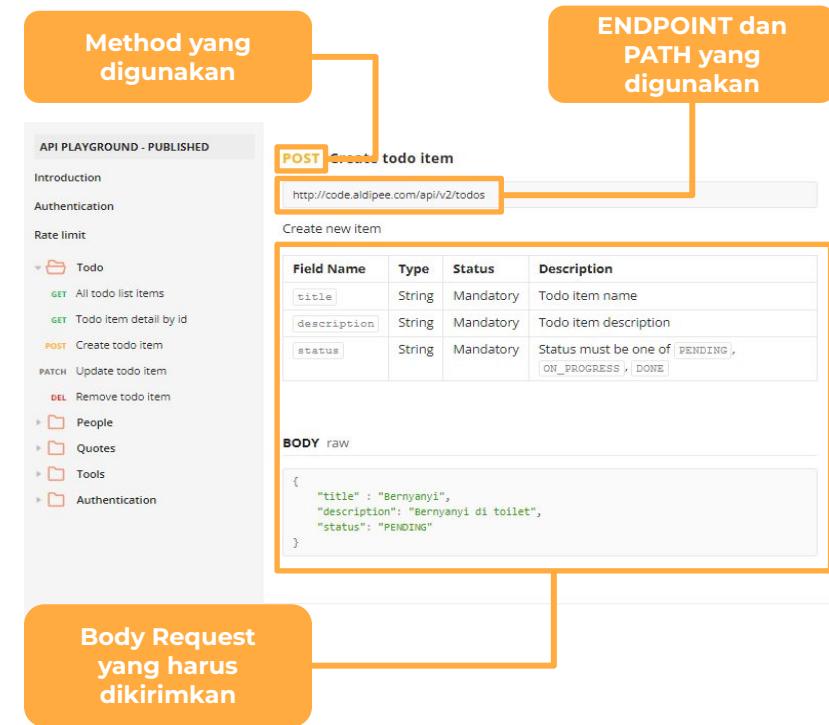
2. Setelah itu akan ada informasi berupa Path dan Method yang kamu butuhkan untuk bikin data baru pada server.

Selain itu, ada juga yang namanya body request dalam method POST.

Apa itu body request?

Jadi karena kamu mau bikin data baru pada server, maka secara otomatis kamu juga akan **mengirimkan body request yang berisi data-data** apa yang ingin kamu buat pada server.

Yuk, kita coba di POSTMAN.



The screenshot shows the Postman API playground interface. On the left, there's a sidebar with sections like Introduction, Authentication, Rate limit, Todo (with sub-options for GET, POST, PATCH, and DEL), People, Quotes, Tools, and Authentication. The main area shows a POST request to 'Create todo item' with the URL 'http://code.aldipee.com/api/v2/todos'. Below the URL, there's a 'Create new item' section with a table for field names, types, statuses, and descriptions. The 'title' field is set to String, Mandatory, and Todo item name. The 'description' field is also String, Mandatory, and Todo item description. The 'status' field is String, Mandatory, with a note that it must be one of PENDING, ON_PROGRESS, or DONE. At the bottom, there's a 'BODY raw' section containing a JSON object:

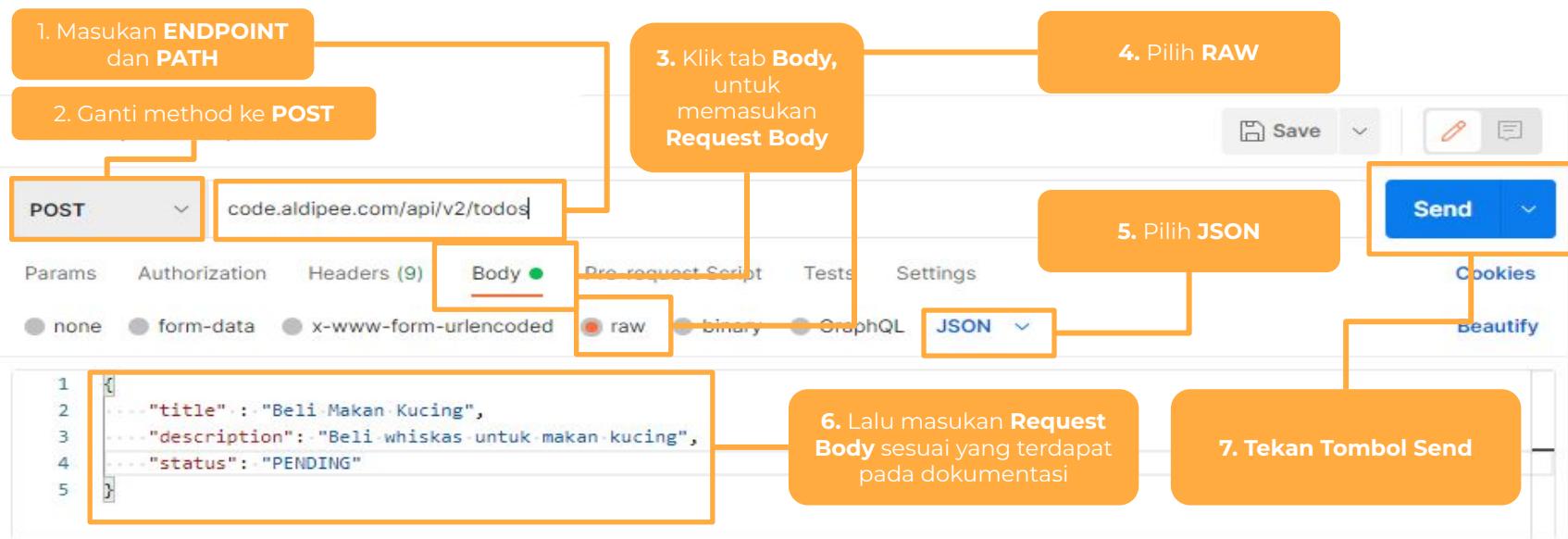
```
{  
  "title": "Bernyanyi",  
  "description": "Bernyanyi di toilet",  
  "status": "PENDING"  
}
```

Three callout boxes highlight specific parts of the interface:

- Method yang digunakan** (Method used) points to the 'POST' button.
- ENDPOINT dan PATH yang digunakan** (Endpoint and PATH used) points to the URL input field.
- Body Request yang harus dikirimkan** (Body Request that must be sent) points to the JSON body in the 'BODY raw' section.



Langkah-langkah bikin body request:



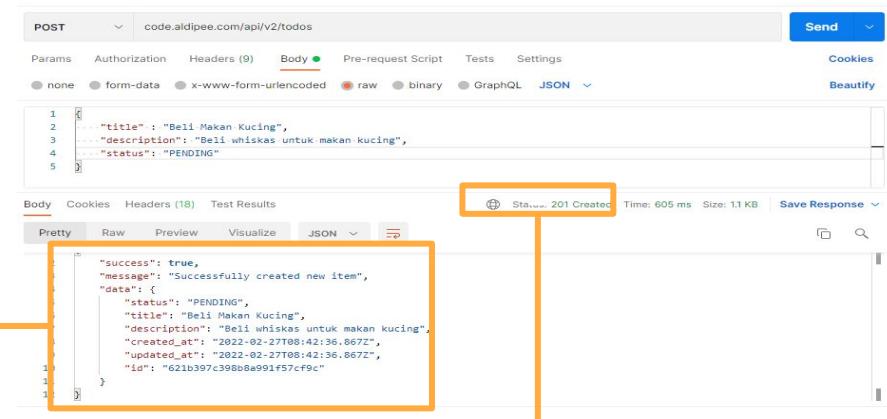


Nah, setelah kamu berhasil mengirimkan Request POST ke server, maka server akan memberikan Respon berupa data yang kamu kirim tadi.

3. Selanjutnya, ada message "Successfully created new Item" dan Kode Status HTTP yang diawali dengan angka 2, yang menandakan request pembuat data baru telah berhasil diterima oleh server. Sekarang data **dengan title "Beli Makan Kucing"** telah berhasil disimpan.

Terdapat informasi response dari server yang menandakan request berhasil

HTTP Status Code diawali dengan angka 2, yang menandakan Request Berhasil



```
POST code.aldipee.com/api/v2/todos
Body (JSON)
{
  "title": "Beli Makan Kucing",
  "description": "Beli whiskas untuk makan kucing",
  "status": "PENDING"
}

Body (Pretty)
{
  "success": true,
  "message": "Successfully created new item",
  "data": {
    "status": "PENDING",
    "title": "Beli Makan Kucing",
    "description": "Beli whiskas untuk makan kucing",
    "created_at": "2022-02-27T08:42:36.867Z",
    "updated_at": "2022-02-27T08:42:36.867Z",
    "id": "621b397c398b8a991f57cf9c"
  }
}
```



- Untuk mengkonfirmasi data kamu telah tersimpan di server, kamu bisa coba request untuk membaca semua data to-do item pada server, seperti yang telah kita lakukan sebelumnya.

Nah sekarang data yang telah kamu tambahkan pada server pada method POST sebelumnya sudah muncul Ketika dibaca.

```
GET code.aldipee.com/api/v2/todos
Send
Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies
Status: 200 OK Time: 1037 ms Size: 1.19 KB Save Response
Body Cookies Headers (18) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "success": true,
3   "message": "Successfully retrieved items",
4   "data": [
5     "results": [
6       {
7         "status": "PENDING",
8         "title": "Beli Telur",
9         "created_at": "2022-02-26T01:45:23.495Z",
10        "id": "62198633c07f275e088dc13e"
11      },
12      {
13        "status": "PENDING",
14        "title": "Beli Makanan Kucing",
15        "created_at": "2022-02-27T08:42:36.867Z",
16        "id": "621b397c39808a991f57cf9c"
17      }
18    ],
19    "page": 1,
20    "limit": 10,
21    "total": 2
22  ]
```



**Yeaayy!! Selamat yaa, guys! Akhirnya
kamu berhasil menuntaskan Topik 4
tentang Networking ini. Biar makin jago,
yuk kerjain quiz berikut ini!**



Saatnya kita Quiz!





1. Berdasarkan gambar berikut, mana yang disebut sebagai PATH?



1 <https://online.binar.co.id/api/v1/data/students>

- A. https://
- B. online.binar.co.id
- C. /api/v1/data/students



1. Berdasarkan gambar berikut, mana yang disebut sebagai PATH?



1 <https://online.binar.co.id/api/v1/data/students>

- A. https://
- B. online.binar.co.id
- C. /api/v1/data/students

/api/v1/data/students
PATH selalu terletak setelah HOST.



2. Berdasarkan gambar berikut, mana yang disebut sebagai ENDPOINT?



1 <https://online.binar.co.id/api/v1/data/students>

- A. https://
- B. online.binar.co.id
- C. /api/v1/data/students



2. Berdasarkan gambar berikut, mana yang disebut sebagai ENDPOINT?



1 https://online.binar.co.id/api/v1/data/students

- A. https://
- B. online.binar.co.id
- C. /api/v1/data/students

Host selalu terletak setelah https:// atau http://



3. Manakah pernyataan-pernyataan yang benar di bawah ini?

- A. Format data JSON lebih sering digunakan pada RESTful API, karena formatnya lebih mudah dibaca
- B. Format data XML lebih sering digunakan pada RESTful API, karena formatnya lebih mudah dibaca
- C. Dalam sebuah API format data JSON dan XML bisa bergabung menjadi satu



3. Manakah pernyataan-pernyataan yang benar di bawah ini?

- A. Format data JSON lebih sering digunakan pada RESTFul API, karena formatnya lebih mudah dibaca
- B. Format data XML lebih sering digunakan pada RESTFul API, karena formatnya lebih mudah dibaca
- C. Dalam sebuah API format data JSON dan XML bisa bergabung menjadi satu

Format data JSON lebih sering digunakan pada RESTFul API, karena formatnya lebih mudah dibaca



4. Apakah kamu dapat membuat method request yang berbeda pada ENDPOINT dan PATH yang sama persis?



1 <https://online.binar.co.id/api/v1/data/students>

- A. Tidak, setiap method request harus memiliki ENDPOINT dan PATH yang berbeda
- B. Bisa, setiap method request yang berbeda dapat dilakukan pada ENDPOINT dan PATH yang sama
- C. Setiap beda Method Request, ENDPOINT harus beda pula, tapi PATH harus sama



Apakah kamu dapat membuat method request yang berbeda pada ENDPOINT dan PATH yang sama persis?



1 <https://online.binar.co.id/api/v1/data/students>

- A. Tidak, setiap method request harus memiliki ENDPOINT dan PATH yang berbeda
- B. Bisa, setiap method request yang berbeda dapat dilakukan pada ENDPOINT dan PATH yang sama
- C. Setiap beda Method Request, ENDPOINT harus beda pula, tapi PATH harus sama

Bisa, setiap method request yang berbeda dapat dilakukan pada ENDPOINT dan PATH yang sama karena setiap ENDPOINT dan PATH pada sebuah RESTFull API bisa memiliki beberapa HTTP Method



5. Mengapa method POST harus memiliki Request Body, sedangkan method GET tidak harus memiliki Request Body

- A. Karena method POST bertugas mengirimkan data pada server, sehingga harus membawa request body
- B. Karena Method GET tidak support Request Body
- C. Karena method POST dan GET adalah jenis method yang sama tapi hanya beda nama saja



5. Mengapa method POST harus memiliki Request Body, sedangkan method GET tidak harus memiliki Request Body

- A. Karena method POST bertugas mengirimkan data pada server, sehingga harus membawa request body
- B. Karena Method GET tidak support Request Body
- C. Karena method POST dan GET adalah jenis method yang sama tapi hanya beda nama saja

Karena method POST berfungsi mengirimkan data ke server, sedangkan method GET hanya mengambil data dari server. Nah karena POST berfungsi mengirimkan data, otomatis dalam pembuatan requestnya harus ada data yang dikirimkan, nah data yang akan dikirimkan ini lah yang dimaksud request body.



1

Berdasarkan gambar di atas, mana yang disebut sebagai **PATH**?

- A https://
online.binar.co.id
- B /api/v1/data/students
- C



- C /api/v1/data/students
PATH selalu terletak setelah HOST.



2

Berdasarkan gambar di atas, mana yang disebut sebagai **ENDPOINT**?

- A https://
online.binar.co.id
- B /api/v1/data/students
- C



2

B

online.binar.co.id

Host selalu terletak setelah https:// atau http://



3

Manakah pernyataan-pernyataan yang benar di bawah ini?

- A Format data JSON lebih sering digunakan pada RESTFul API, karena formatnya lebih mudah dibaca
- B Format data XML lebih sering digunakan pada RESTFul API, karena formatnya lebih mudah dibaca
- C Dalam sebuah API format data JSON dan XML bisa bergabung menjadi satu



3

A

Format data JSON lebih sering digunakan pada RESTFul API, karena formatnya lebih mudah dibaca

Format data JSON lebih sering digunakan pada RESTFul API, karena formatnya lebih mudah dibaca



4

Apakah kamu dapat membuat method request yang berbeda pada ENDPOINT dan PATH yang sama persis?

- A Tidak, setiap method request harus memiliki ENDPOINT dan PATH yang berbeda
- B Bisa, setiap method request yang berbeda dapat dilakukan pada ENDPOINT dan PATH yang sama
- C Setiap beda Method Request, ENDPOINT harus beda pula, tapi PATH harus sama



4

B

Bisa, setiap method request yang berbeda dapat dilakukan pada ENDPOINT dan PATH yang sama

Karena setiap ENDPOINT dan PATH pada sebuah RESTFull API bisa memiliki beberapa HTTP Method



5

Mengapa method POST harus memiliki Request Body, sedangkan method GET tidak harus memiliki Request Body

- A Karena method POST bertugas mengirimkan data pada server, sehingga harus membawa request body
- B Karena Method GET tidak support Request Body
- C Karena method POST dan GET adalah jenis method yang sama tapi hanya beda nama saja



5

A

Karena method POST bertugas mengirimkan data pada server, sehingga harus membawa request body.

Karena method POST berfungsi mengirimkan data ke server, sedangkan method GET hanya mengambil data dari server. Nah karena POST berfungsi mengirimkan data, otomatis dalam pembuatan requestnya harus ada data yang dikirimkan, nah data yang akan dikirimkan ini lah yang dimaksud request body.

Terima Kasih!



Next Topic

loading...