**Documentation & Testing API dengan Postman part 2**

Melanjutkan sesi dari [part 1](https://medium.com/skyshidigital/documentation-testing-api-dengan-postman-part-1-5d33e430dca7#.6pgqr3y41) kita telah membahas bagaimana melakukan request ke API melalui postman kemudian kita bisa simpan request kedalam collection yang bisa dishare ke anggota tim lain. Dipart 2 yang akan kita bahas kali ini adalah

1. Design API yang Baik
2. Documentation & Mockup API dengan Apiary
3. Testing API dengan Postman

# Kenapa Design API itu Penting ?

API (*Application program interface*) adalah service yang menyediakan data dari server yang di request oleh client yang biasanya dalam format json, API ini adalah bagian yang paling penting dalam sebuah aplikasi Mobile / web frontend. Design API yang jelas akan memudahkan kita dalam mendevelop aplikasi.

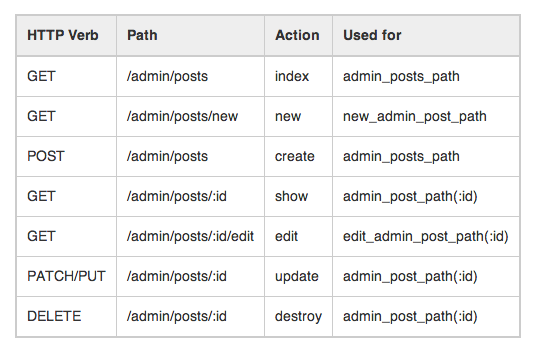
Bayangkan ketika sebuah API didesign dengan kembalian data yang tidak konsisten, nama path yang tidak merepresentasikan dari fungsi API, data yang dikembalikan terlalu berbelit belit dalam artian seharusnya bisa 1x panggil tapi harus 2x panggil dengan path API yang berbeda itu akan menambah effort untuk developing.

**Bagaimana API yang Baik?**

1. **Restful**

Untuk penamaan path API yang baik kita bisa memperhatikan aturan [Restful](https://en.wikipedia.org/wiki/Representational_state_transfer). Restful ini hanya menjadi acuan, mungkin ada beberapa kondisi tidak bisa menggunakan aturan Restful.

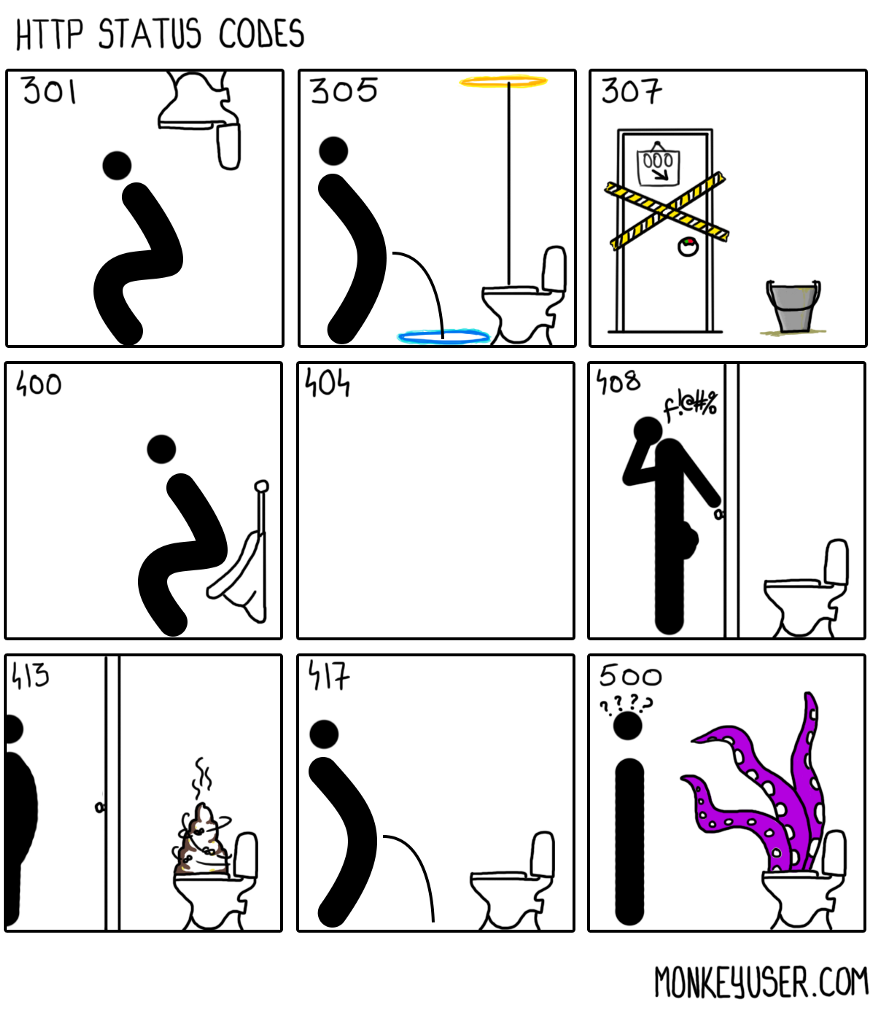
Table Restful API



Sebagai contoh ada Restful pada resource admin/post diatas, dengan path yang sama yaitu /admin/posts/ kita bisa membuat beberapa action sesuai dengan HTTP Verb. Untuk nama path yang biasa digunakan adalah plurar (jamak)

1. **HTTP Status Code**

Http Status code ini berfungsi untuk membantu kita dalam mengenali response balik dari API.



untuk menentukan Http Status Code bisa cek flowchart disini  [http://racksburg.com/choosing-an-http-status-code](http://racksburg.com/choosing-an-http-status-code/)

1. **Error Message**

Error message yang baik akan memudahkan developer dalam mengenali kesalahan seperti missing parameter, error parameter validation dll. berikut salah satu contoh error message di twitter

|  |
| --- |
| {  "errors": [  {  "message": "Sorry, that page does not exist",  "code": 34  }  ]  } |

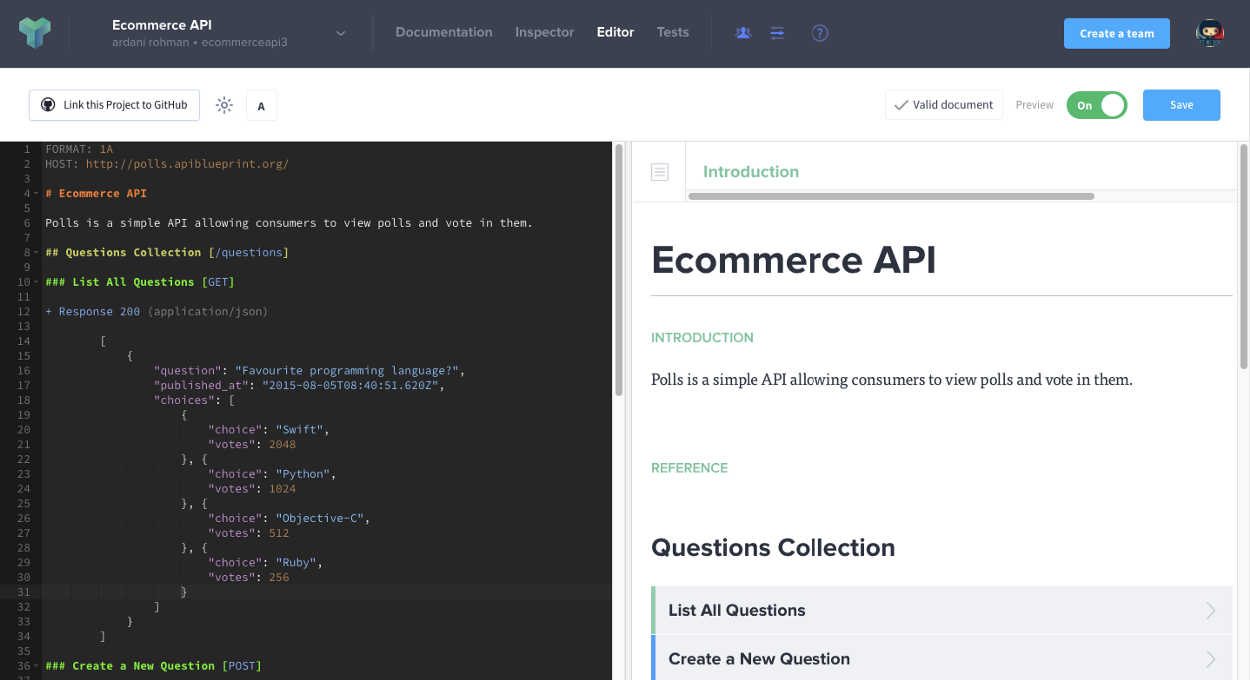
1. **Authentication**

Authentication ini diperlukan untuk membatasi akses pada API. Karena API bersifat stateless (tidak menggunakan session untuk menyimpan auth) sehingga menggunakan token yang dikirim melalui header request seperti [JWT](https://jwt.io/) dan [OAuth2](https://oauth.net/2/). untuk implementasi Auth akan kita bahas di artikel berikutnya :)

1. **Versioning**

Versioning ini digunakan ketika kita mau mengembangkan API ke versi yang baru dan masih support untuk API yang lama. teknik versioning API ada 2 macam yang pertama adalah dengan menambahkan path pada API /v2 (terbaru) dan /v1 (lama) bisa dimanage melalui Route dilevel aplikasi atau di level web server. Cara yang kedua menggunakan param header

# Documentation & Mockup API



Keperluan documentation API tidak kalah penting didalam pengembangan API, documentation API yang lengkap akan mempermudah dalam proses develop aplikasi mobile atau web frontend. Untuk urusan documentation dan mockup sudah ada beberapa service online yang bisa digunakan secara free salah satunya adalah [Apiary.io](http://apiary.io/)

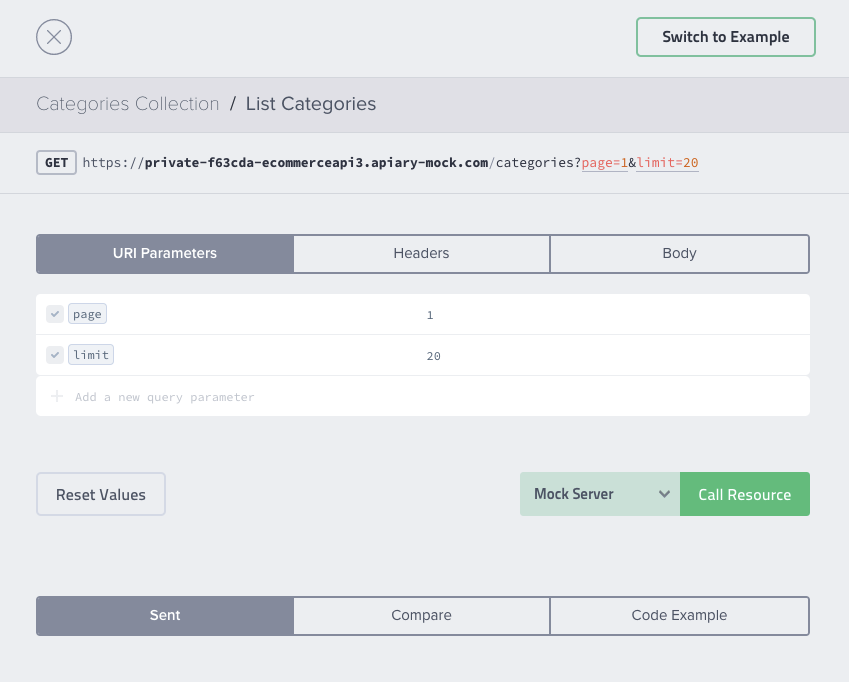
pada Apiary kita bisa membuat mockup API dan documentation sekaligus, fitur yang dimiliki apiary memungkinkan kita membuat Server Fake API bisa diakses secara public sehingga proses develop tim mobile / frontend tidak perlu menunggu sampai API siap tetapi bisa menggunakan API dari Apiary untuk sementara. (efisien waktu ⏰)

ini adalah sample API mockup yang saya buat bisa langsung di copy paste ke editor apiary atau bisa dicoba <http://docs.ecommerceapi3.apiary.io/>

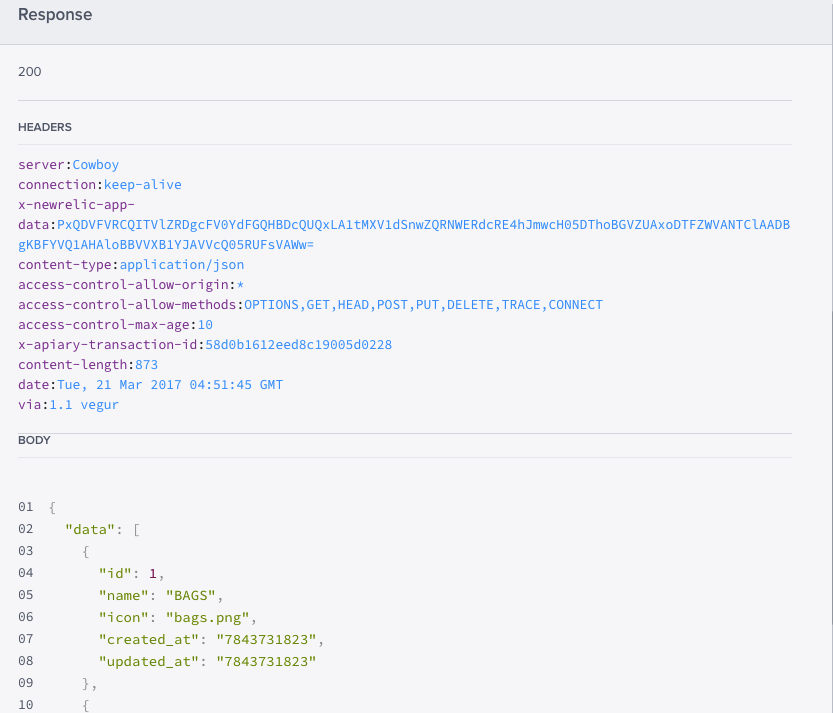
|  |
| --- |
| FORMAT: 1A  HOST: https://api.ecommerce.com/  # Ecommerce API  API documentation for E-commerce  ## Categories Collection [/categories{?page}{?limit}]  ### List Categories [GET]  + Parameters  + page : 1 (number) - optional  + limit : 20 (number) - optional  + Response 200 (application/json)  {  "data":[  {  "id": 1,  "name": "BAGS",  "icon": "bags.png",  "created\_at": "7843731823",  "updated\_at": "7843731823"  },  {  "id": 2,  "name": "ACCESSORIES",  "icon": "accessories.png",  "created\_at": "7843731823",  "updated\_at": "7843731823"  },  {  "id": 3,  "name": "SHOES",  "icon": "shoes.png",  "created\_at": "7843731823",  "updated\_at": "7843731823"  },  {  "id": 4,  "name": "SMALL LEATHER GOODS",  "icon": "small-leather-goods.png",  "created\_at": "7843731823",  "updated\_at": "7843731823"  }  ],  "meta": {  "total": 100,  "page": 1,  "limit":20    }  }  ### Create a New Category [POST]  + Request (application/json)  {  "name": "TIMEPIECE",  "icon": "timepiece.png"  }  + Response 201 (application/json)  + Headers  Location: /categories/15  + Body  {  "name": "TIMEPIECE",  "icon": "timepiece.png",  "created\_at": "7843731823",  "updated\_at": "7843731823"  }  ### Update Category [PUT /{id}]  + Parameters  + id (number) - ID of category must integer  + Request (application/json)  {  "name": "TIMEPIECE",  "icon": "timepiece.png"  }  + Response 204 (application/json)  ### Detail Category [GET /{id}]  + Parameters  + id: 1 (number) - ID of category must integer  + Response 200 (application/json)  {  "name": "TIMEPIECE",  "icon": "timepiece.png",  "created\_at": "7843731823",  "updated\_at": "7843731823"  }  ### Delete Category [DELETE /{id}]  + Parameters  + id (number) - ID of category must integer  + Response 204 (application/json) |

sekarang kita coba untuk test mockup API dengan klik switch console untuk mencoba hasil response API, yang perlu diingat adalah mockup API ini bersifat statis jadi format yang kita tulis akan menjadi response dan pastikan response format yang kalian tulis itu valid

console mockup Apiary

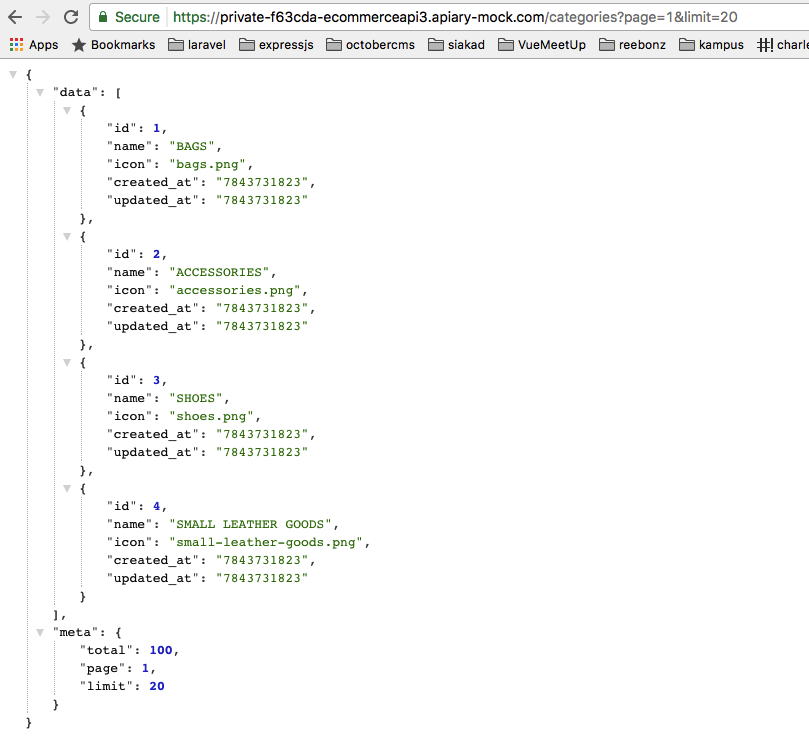


response mockup



selain itu kita bisa call langsung url private dari generate apiary <https://private-f63cda-ecommerceapi3.apiary-mock.com/categories?page=1&limit=20> dan hasilnya:

response json



# Testing API

setelah documentation selesai dibuat saatnya kita bikin testing API. banyak cara untuk testing API bisa melalui unit test memanfaatkan library seperti Codeception / Laravel Dusk tapi kali ini kita bahas yang paling simple menggunakan Postman

sebelumnya kita sudah bikin mockup API di Apiary sekarang kita bikin testnya. Untuk postman kita bisa menggunakan script JS untuk membuat testingnya doc : <https://www.getpostman.com/docs/testing_examples>

Aktifkan Postman Console pada menu view show postman console untuk keperluan debugging.

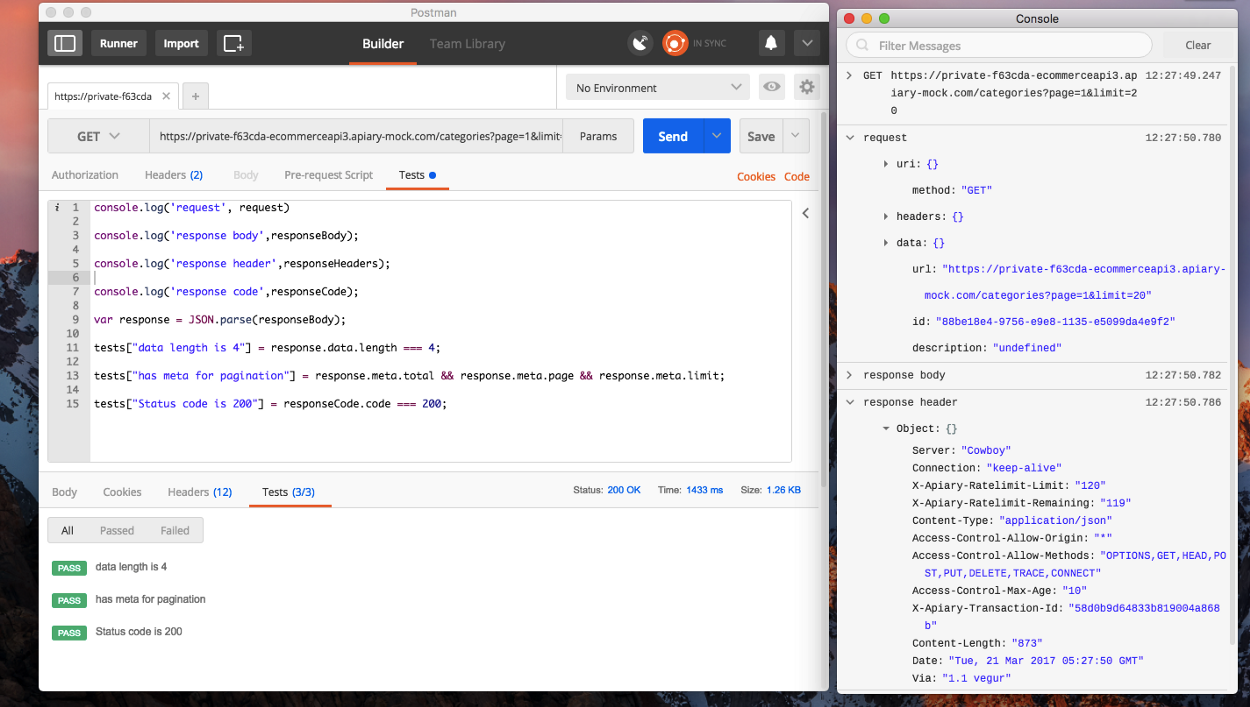
variable yang telah disediakan oleh postman adalah request, responBody, responHeaders, responseCode bisa langsung console.log untuk melihat isinya.

Test API ini konsepnya adalah membuat membuat perbandingan response code dan struktur json response yang dikeluarkan

|  |
| --- |
| console.log('request', request)  console.log('response body',responseBody);  console.log('response header',responseHeaders);  console.log('response code',responseCode);  var response = JSON.parse(responseBody);  tests["data length is 4"] = response.data.length === 4;  tests["has meta for pagination"] = response.meta.total && response.meta.page && response.meta.limit;  tests["Status code is 200"] = responseCode.code === 200; |

untuk menjalankan script test API caranya sama dengan call API lewat postman dan hasilnya bisa dilihat seperti pada tampilan dibawah ini 😀

hasil test API



Dari hasil test API diatas semua pass / success, setelah test yang kita buat sudah sesuai yang diharapkan maka ketika API sudah selesai develop kita bisa menganti environmentnya di postman untuk melakukan test di staging misalnya. Ketika ada perubahan API kita hanya perlu menjalankan test yang sudah kita buat dan apabila success bisa dilanjutkan deploy ke production.

# Kesimpulan

API menjadi bagian penting dalam menghubungkan antara backend dengan frontend dan mobile. Kualitas API bisa dilihat dari response body, status code, error message yang dihasilkan. Pastikan ketika develop API kita tidak lupa untuk membuat design dan documentation sebab kebanyakan bottleneck antara backend dan frontend sering terjadi pada API yang tidak jelas. Next kita akan membahas tentang How To Developing API 🎯