


# Tutorial RESTful API Node JS + Express + MySQL


## Belajar RESTful API Node JS + Express (Create, Update, Delete)



Kiddy [Follow](#)

Aug 8, 2018 · 4 min read

 Login ke medium.com dengan Google ×



**doni92**  
nurramdandoni@gmail.com

LANJUTKAN SEBAGAI DONI92

Untuk membuat akun, Google akan membagikan nama, alamat email, dan gambar profil Anda kepada medium.com. Dengan melanjutkan, Anda menyetujui [kebijakan privasi](#) dan [persyaratan layanan](#) medium.com.

# BELAJAR



# BARENG KIDDY

{ CRUD TUTORIAL  
DENGAN NODE JS +  
EXPRESS DAN MY SQL }



Hello fellas!

Maaf banget nih baru sempet ngeblog dan ngelanjutin tutorial gue akibat kesibukan yang mendala seorang anak remaja usia 20 tahun muehehe ^^

Oke langsung to the point aja kali ini gue mau ngajarin lanjutan part 2 dari node js express dan mysql ini. Gue juga sempet baca komenan bahwa ada yang minta dilanjutin, akhirnya gue kabulkan permintaannya (sebenarnya emang harus dilanjutin sih hahaha)

cuss langsung ya, bagi yang belum baca part 1 diharapkan ngantri kesini dulu ya hehe.

Sekarang kita edit file **controller.js** kita dan masukin fungsi dibawah ini, kita akan ngebuat fungsi Search (Read dengan parameter), Create, Update, dan Delete sekaligus muehehe

```

exports.findUsers = function(req, res) {
    var user_id = req.params.user_id;

    connection.query('SELECT * FROM users WHERE user_id = ?',
    [ user_id ],
    function (error, rows, fields) {
        if(error){
            console.log(error);
        } else{
            response.ok(rows, res);
        }
    });
};

exports.createUsers = function(req, res) {
    var first_name = req.body.first_name;
    var last_name = req.body.last_name;

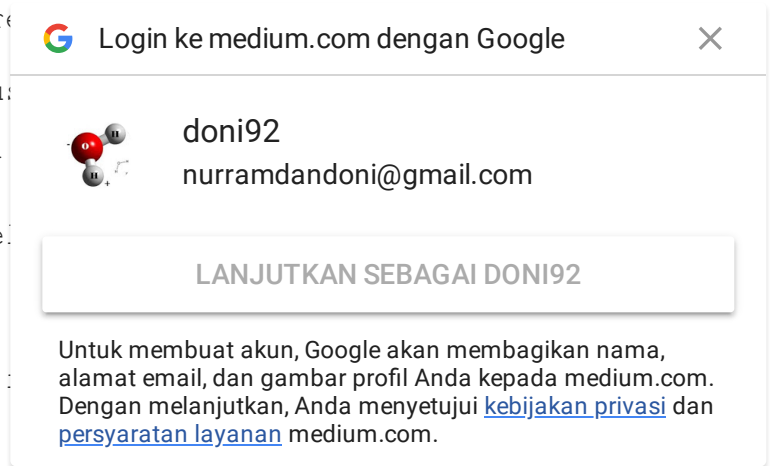
    connection.query('INSERT INTO person (first_name, last_name)
values (?,?)',
    [ first_name, last_name ],
    function (error, rows, fields){
        if(error){
            console.log(error);
        } else{
            response.ok("Berhasil menambahkan user!", res);
        }
    });
};

exports.updateUsers = function(req, res) {
    var user_id = req.body.user_id;
    var first_name = req.body.first_name;
    var last_name = req.body.last_name;

    connection.query('UPDATE person SET first_name = ?, last_name
= ? WHERE id = ?',
    [ first_name, last_name, user_id ],
    function (error, rows, fields){
        if(error){
            console.log(error);
        } else{
            response.ok("Berhasil merubah user!", res);
        }
    });
};

exports.deleteUsers = function(req, res) {
    var user_id = req.body.user_id;

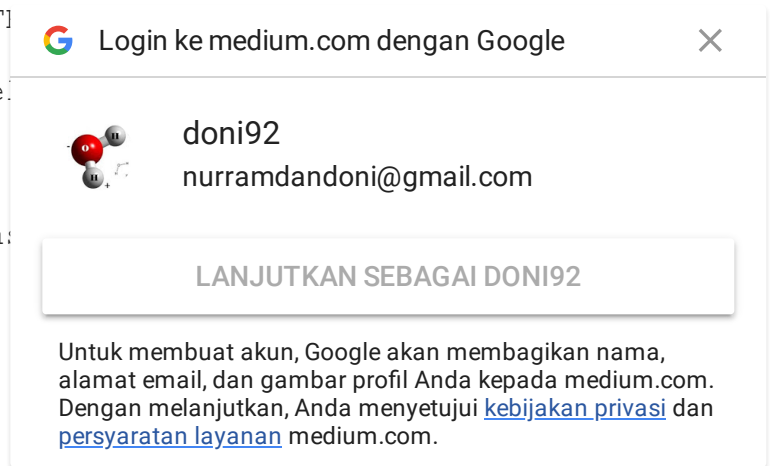
```



```

connection.query('DELETE FROM users WHERE user_id = ?;',
[ user_id ],
function (error, rows, fields) {
    if (error) {
        console.log(error);
    } else {
        response.ok("Berhasil menghapus user");
    }
});
};

```



Kalo udah langsung pasang semua routesnya dibawah ini:

```

'use strict';

module.exports = function(app) {
    var todoList = require('./controller');

    app.route('/')
        .get(todoList.index);

    app.route('/users')
        .get(todoList.users);

    app.route('/users/:user_id')
        .get(todoList.findUsers);

    app.route('/users')
        .post(todoList.createUsers);

    app.route('/users')
        .put(todoList.updateUsers);

    app.route('/users')
        .delete(todoList.deleteUsers);
};

```

Sekarang saatnya bermain....

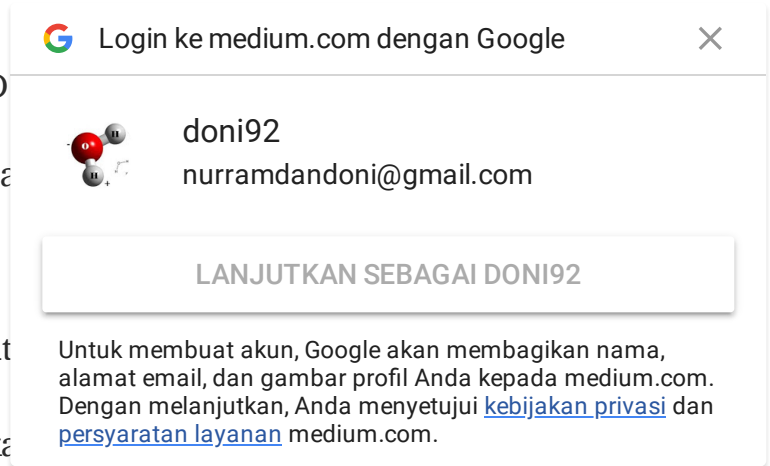
Jalankan dengan cara input “**node server.js**”



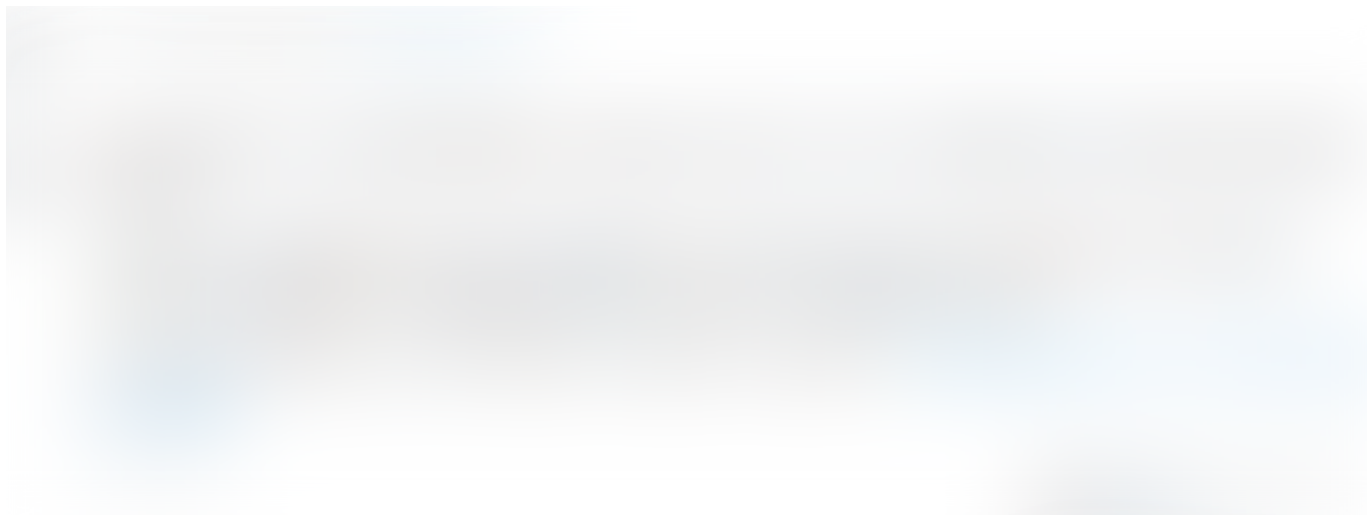
Sekarang kamu memiliki 4 routes baru

1. GET /users/:user\_id untuk search ID
2. POST /users untuk menambahkan data
3. PUT /users untuk mengubah data
4. DELETE /users untuk menghapus data

Kita mulai dari INSERT, buka Postman ke



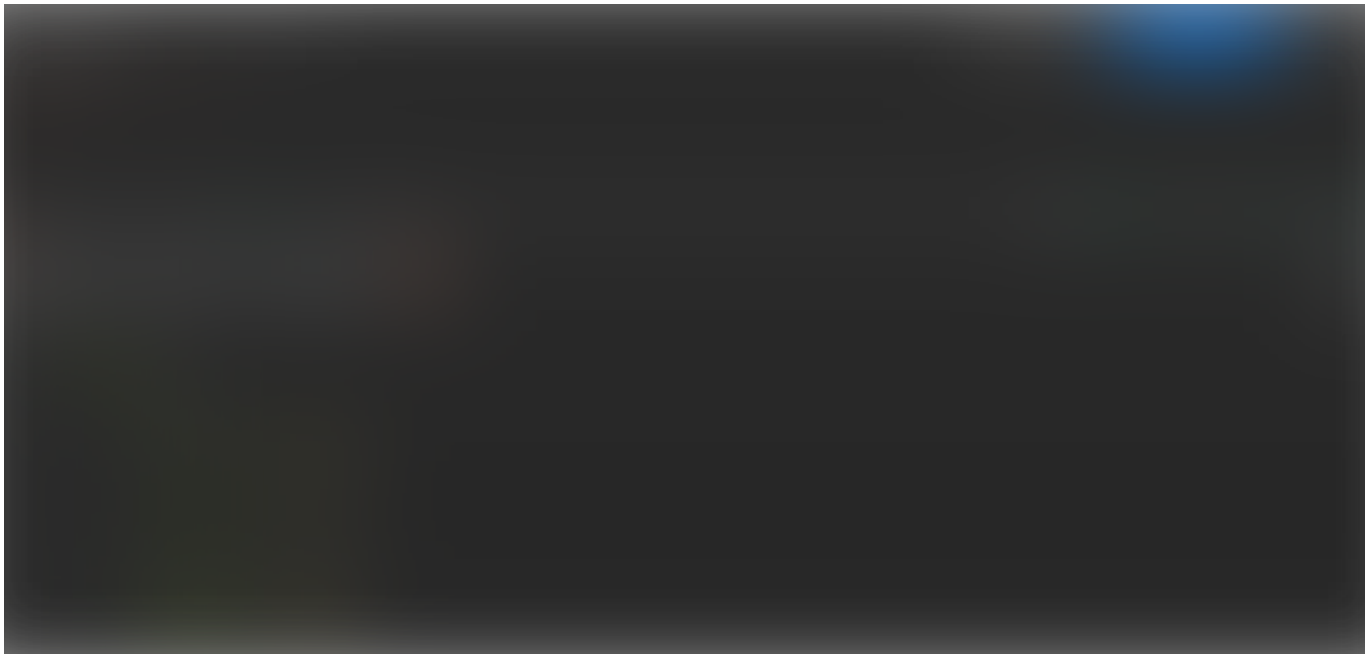
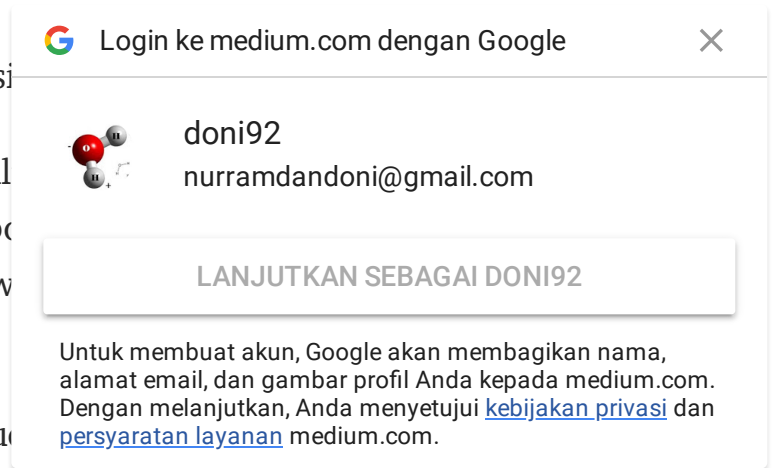
Perhatikan, POST diatas menggunakan **x-www-form-urlencoded**. Untuk Node JS hanya bisa menggunakan **Raw JSON** dan **x-www-form-urlencoded** dan tidak bisa menggunakan tipe form-data. Kenapa?



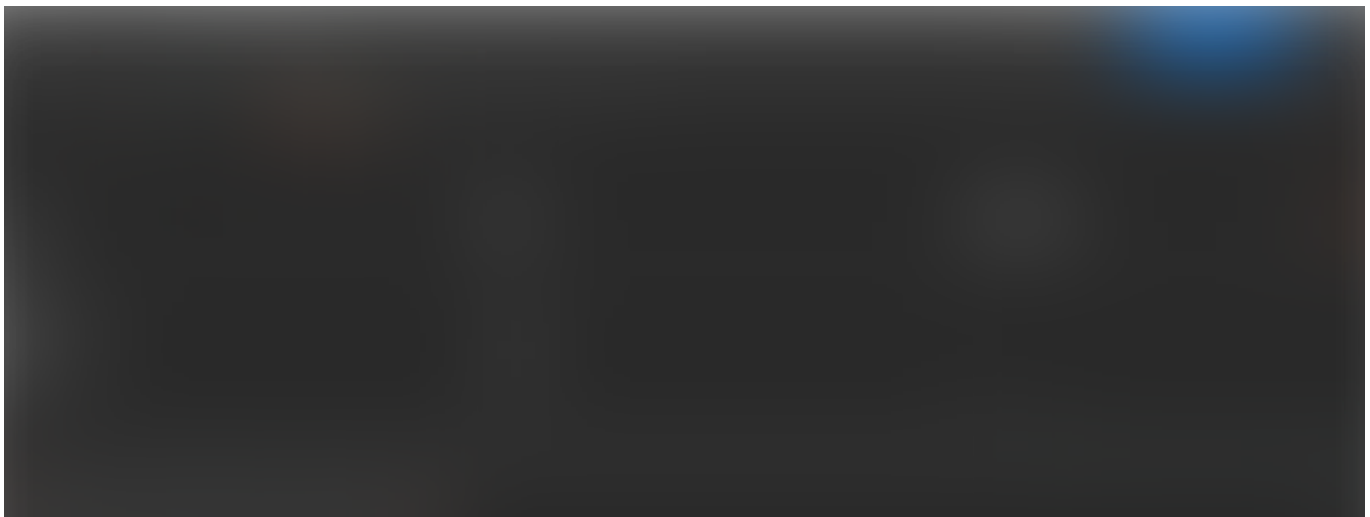
Nah tuh kata salah satu jawaban di websi

Intinya gini, lu gabisa milih tipe data mul  
terlalu besar kalo dimasukin ke library bo  
itu salah satu tipe form yang standar di w  
www-form-urlencoded (CMIIW!).

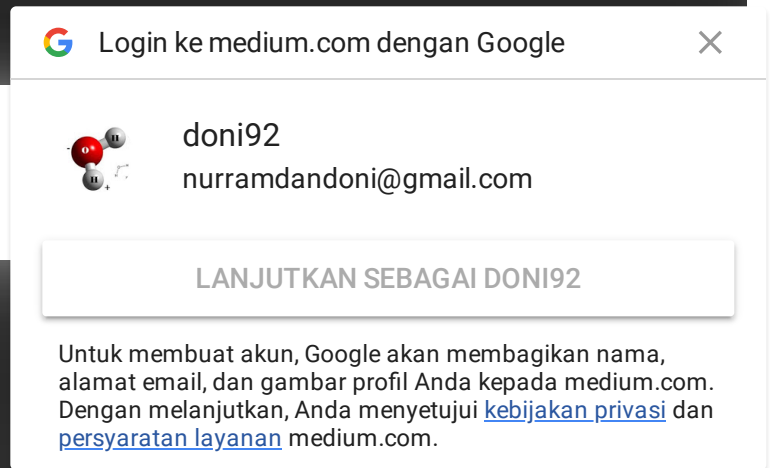
Oke lanjut aja, sekarang cek user kamu u



Oke gue udah berhasil nambah, sekarang kita akan update salah satu ID. Gue akan update ID nomor 2.



Cek lagi



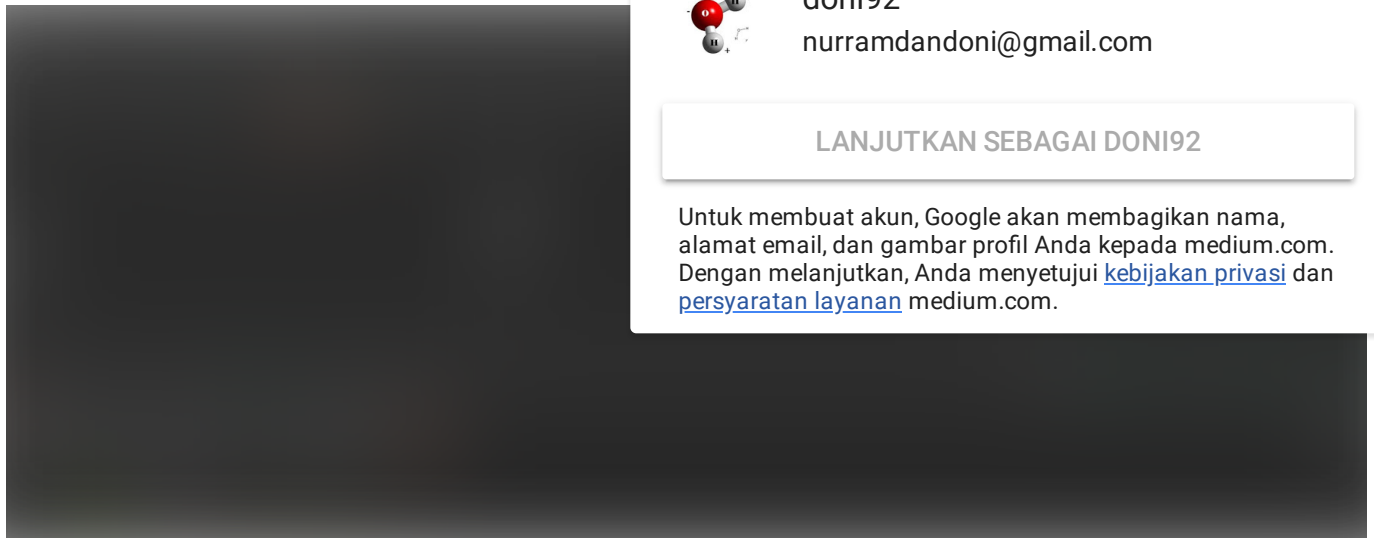
Berubah tuh jadi Kiddy Gates (Diangkat jadi anaknya Bill Gates HAHAAHAHA).

Sekarang kita mau menampilkan ID nomor 2 aja (Read dengan parameter / Search)

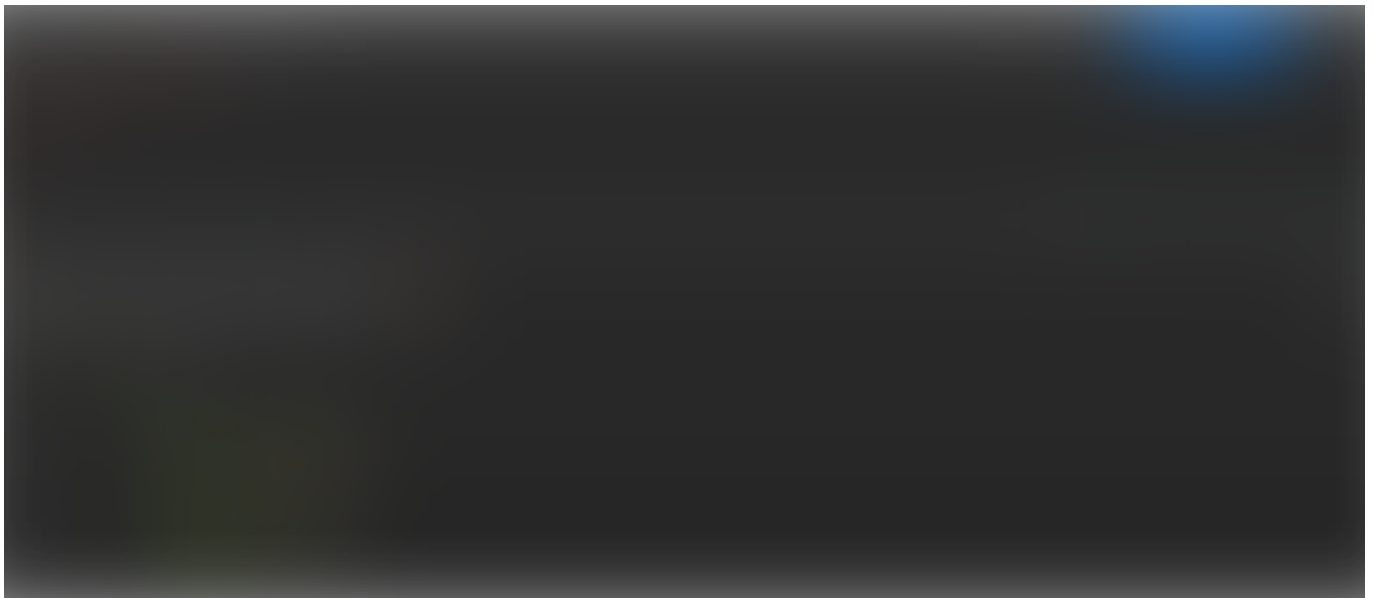


Berhasil untuk nampilin user dengan ID nomor 2, ini digunakan kalo kalian mau liat detail profile user nya.

Oke sekarang kita akan coba fungsi Hapus. kita akan hapus si ID nomor 3 dengan fungsi DELETE.



Kita cek lagi



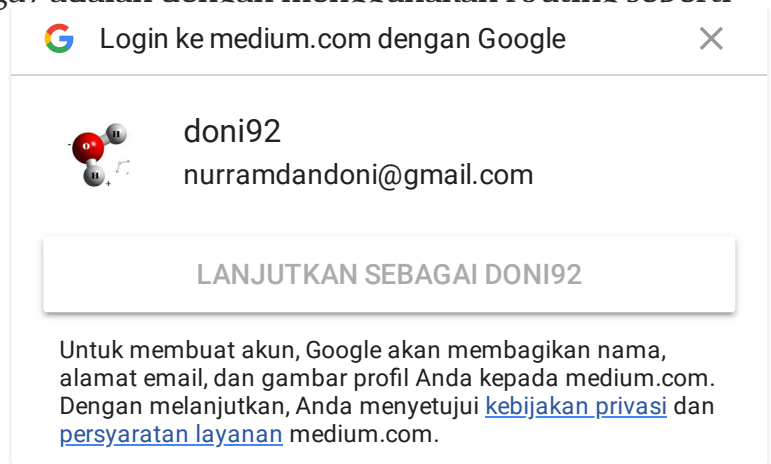
VOILA! Sudah terhapus!

Oke mungkin ada yang bingung kenapa gue pake fungsi GET, POST, PUT, DELETE?

Karena sebenarnya REST API yang bener sih (menurut gue aja kali ya?) itu emang satu URL yang bisa digunakan dengan satu method saja. Tujuannya apa? sebenarnya tujuannya untuk mengirit URL aja.

Kan kebiasaan dari kita (kadang saya juga) adalah dengan menggunakan routing seperti ini:

1. `/url/users/insert`
2. `/url/users/detail/{ID User}`
3. `/url/users/update/{ID User}`
4. `/url/users/delete/{ID User}`



Nah dengan REST API itu semua gaperlu ribet-ribet nulis Update, Delete, Detail. Mainkan semuanya dengan satu URL saja, biar ga ribet-ribet penamaan URLnya.

Tapi itu semua kembali ke developer dan team masing-masing. Tidak ada yang salah baik menggunakan metode diatas atau dengan REST API yang benar, yang penting adalah code rapih, terstruktur, reusable, dan pastinya harus well-maintaned ^^

Sekian yang dapat saya tulis, semoga apa yang saya tulis bermanfaat. Apabila kamu merasa blog ini bermanfaat untuk teman kamu yang baru belajar, jangan ragu untuk membagikannya ya ^^ Semakin banyak ilmu yang kamu bagikan, maka semakin banyak ilmu yang kamu serap.

Nodejs   MySQL   Express   Expressjs

About   Help   Legal