**ALL ABOUT FULLSTACK**

Using Express & Sequelize

By Halim

**Npm install Sequelize pg**

* Sequelize : Mengubah data dari db menjadi objek
* Pg : Postgre (Database)

**npm install sequelize-cli**

* Buat command line interface

**Npx sequelize-cli init**

* Buat inisiasi awal (bakal ngebuild folder config,models,seeder)

**Npx sequelize-cli db:create**

* Ngebuat db nya (cara ceknya bisa lewat dbeaver trus tinggal buka postgre-nya )

**npx sequelize-cli model:generate --name todo --attributes task:string,status:boolean**

* Model:generate -> buat ngebuat blueprint tabelnya biar bisa jadi model dan tabel di dbnya
* --name todo -> nama tabelnya (kl bisa singular kek todo jangan todos karena bakal auto tambah “s”)
* --attributes -> ya berarti attribut si tabelnya ato columnnya (nama\_attribut:tipe\_data)

**npx sequelize-cli db:migrate**

* Memigrasi blueprint yang udah dibuat di project (folder migrate) ke db biar dibuat tabel-nya

**Npm start**

* Jalanin code tapi syaratnya di bagian package,json buat key “start” valuenya nama main filenya kek gini
* "scripts": {
* "start": "nodemon app.js",
* "test": "echo \"Error: no test specified\" && exit 1"
* },

**Get Data from DB**

* Import model-nya dulu caranya di top line kasih gini
* const {todo} = require('../models');
* Terus di bagian static functionnya kaya gini
* static getTodos(req,res){
* todo.findAll()
* .then(todos => {
* res.json(todos);
* })
* .catch(err=>{
* res.json(err);
* })
* }
* Todo disini berarti model yang tadi di require paling atas
* FindAll() ini function bawaan dari sequelize kl mau akses semua row
* Then -> promise untuk dijalankan kalau berhasil kirimnya apa
* Catch -> bila promise gagal ditepati maka bakal ngirim error status

**Validasi atau Filter untuk Query DB lewat sequelize**

static getTodos(req,res){

        todo.findAll({

            // Kalau mau kasih kondisi atau validasi bisa kayak gini

            // where: {

            //     id:1

            // }

            // Kalau mau diurutkan bisa kayak gini

            // ASC = ascending (dari kecil ke besar)

            // DESC = descending (dari besar ke kecil)

            // order: [

            //     ['id', 'DESC']

            // ]

        })

        .then(todos => {

            res.json(todos);

        })

        .catch(err=>{

            res.json(err);

        })

    }

* Yang where mirip kaya query SELECT \* From DB WHERE id = 1
* Untuk menampilkan data juga bisa diformat untuk berurutan (ASC,DESC)