**REST API**

**Authentication and Authorization**

Pada modul sebelumnya, kita telah mempelajari tentang pembuatan endpoint untuk akses, menambah, mengubah, dan menghapus data pada *database (data resource)*. *Data Resource* merupakan sesuatu yang harus diamankan karena berpotensi adanya pencurian atau penyalahgunaan data oleh pihak yang tidak dikenal. Oleh karena itu, tidak sembarang orang dapat menggunakan endpoint yang kita buat. Endpoint hanya dapat digunakan oleh orang-orang yang diizinkan oleh sistem. Dalam literasi IT terdapat istilah *Authentication* dan *Authorization*. ***Authentication*** adalah proses mengenal user yang akan masuk pada sebuah sistem. ***Authorization*** adalah proses pemberian izin akses user terhadap sistem.

Pada konteks pemanfaatan REST API ini, proses *authentication* dilakukan dengan memberikan kode token kepada user yang telah terverifikasi oleh sistem melalui proses login. Jika user sudah mendapatkan kode token, user dapat memanfaatkan endpoint yang sudah disediakan. Namun setiap kali user melakukan *request* pada suatu endpoint, akan terjadi proses *authorization* terlebih dahulu dengan memvalidasi kode token yang ada. Jika kode token valid, maka user akan mendapatkan *response* sesuai dengan karakteristik endpoint yang dibuat.

**Langkah Praktikum**

Pada praktikum bagian ini, kita masih menggunakan project REST API pelanggaran siswa sebelumnya. Silahkan ikuti Langkah-langkah berikut ini.

* Buka tabel “user” pada database “pelanggaran\_siswa” melalui phpMyAdmin.
* Pastikan data tabel “user” sudah terisi dan untuk kolom password telah menggunakan hash **MD5**.



* Install *library* **cryptr** dan **md5** pada project.

**npm install --save cryptr md5**

*Library* **cryptr** digunakan untuk *encrypt* dan *decrypt* data, sedangkan *library* **md5** digunakan untuk *hashing string* format MD5.

* Pada file api.js (*source code* sebelumnya), tambahkan code berikut untuk memanggil *library* cryptr dan md5.

|  |
| --- |
| const md5 = require("md5")  const Cryptr = require("cryptr")  const crypt = new Cryptr("140533601726") // secret key, boleh diganti kok |

* Selanjutnya kita akan membuat endpoint untuk proses *authentication* menggunakan method POST.

|  |
| --- |
| // endpoint login user (authentication)  app.post("/user/auth", (req, res) => {      // tampung username dan password      let param = [          req.body.username, //username          md5(req.body.password) // password      ]        // create sql query      let sql = "select \* from user where username = ? and password = ?"      // run query      db.query(sql, param, (error, result) => {          if (error) throw error          // cek jumlah data hasil query          if (result.length > 0) {              // user tersedia              res.json({                  message: "Logged",                  token: crypt.encrypt(result[0].id\_user), // generate token                  data: result              })          } else {              // user tidak tersedia              res.json({                  message: "Invalid username/password"              })          }      })  }) |

* Setelah itu kita akan membuat sebuah fungsi **validateToken** sebagai proses *authorization* yang digunakan untuk memvalidasi token yang dikirimkan. (**letakkan code berikut ini sebelum pembuatan semua endpoint**)

|  |
| --- |
| validateToken = () => {      return (req, res, next) => {          // cek keberadaan "Token" pada request header          if (!req.get("Token")) {              // jika "Token" tidak ada              res.json({                  message: "Access Forbidden"              })          } else {              // tampung nilai Token              let token  = req.get("Token")                // decrypt token menjadi id\_user              let decryptToken = crypt.decrypt(token)              // sql cek id\_user              let sql = "select \* from user where ?"              // set parameter              let param = { id\_user: decryptToken}              // run query              db.query(sql, param, (error, result) => {                  if (error) throw error                   // cek keberadaan id\_user                  if (result.length > 0) {                      // id\_user tersedia                      next()                  } else {                      // jika user tidak tersedia                      res.json({                          message: "Invalid Token"                      })                  }              })          }      }  } |

* Kemudian kita masukkan fungsi **validateToken()** pada salah satu endpoint, misal endpoint untuk akses data siswa (**/siswa**) dengan method GET.



* Simpan dan jalankan ulang file app.js (dengan perintah **node api.js**)
* Buka Postman, dan kita akan memanggil endpoint untuk akses data siswa seperti berikut ini.



Akan memunculkan response seperti berikut ini.



Kita mendapatkan pesan “Access Forbidden” karena kita tidak mengirimkan kode token pada saat *request* endpoint ini.

* Untuk mendapatkan kode token, kita perlu melakukan proses *authentication*, yaitu dengan memanggil endpoint “/user/auth” menggunakan method POST.



Dan kita kirimkan username dan password sesuai dengan data yang tersimpan pada tabel user.



Jika berhasil akan mendapatkan *response* seperti berikut.



Simpan kode token pada clipboard (lakukan copy pada token).

* Kita akses kembali endpoint untuk mengambil data siswa



Pada **tab Headers**, tambahkan **key “Token”** dan paste kode token sebelumnya untuk mengisi value dari “Token”



Klik tombol **Send** dan kita akan mendapatkan data siswa seperti yang diharapkan.



* Lakukan kembali Langkah nomor 7 pada semua endpoint (kecuali endpoint untuk authentication) guna mengamankan endpoint yang telah kita buat.